DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE MISSÃO ATRAVÉS DAS REDES DE PETRI

Luciano Menezes Junior

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Ramon Romankevicius Costa

Rio de Janeiro
Outubro de 2014

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE DE MISSÃO ATRAVÉS DAS REDES DE PETRI

Luciano Menezes Junior

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Ramon Romankevicius Costa, D.Sc.

Prof. Fernando Cesar Lizarralde, D. Sc.

Prof. Marco Antonio Meggiolaro, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2014

Menezes Junior, Luciano

Desenvolvimento de um sistema de controle de missão através das Redes de Petri/Luciano Menezes Junior. – Rio de Janeiro: UFRJ/COPPE, 2014.

XVIII, 118 p.: il.; 29, 7cm.

Orientador: Ramon Romankevicius Costa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2014.

Referências Bibliográficas: p. 109 – 118.

1. Sistema de Controle de Missão. 2. Rede de Petri. 3. Veículo Autônomo Submarino. I. Costa, Ramon Romankevicius. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

A meus pais, Luciano e Maria José, a esposa Jordana, aos irmãos e aos demais amigos.

Agradecimentos

Primeiramente a Deus, pela condução e pela sabedoria para superar os desafios durante toda trajetória. Ao orientador Ramon Romankevicius Costa, pelos ensinamentos e pela serenidade. Aos professores do Labcon: Liu Hsu e Fernando Lizarralde, por toda colaboração em minha formação acadêmica. Aos amigos de estudo, Alessandro Santos, Alcidney Valério, Ivanko Yannick Yanque Tomasevich, Marcela Tarazona e Nerito Aminde, pelo apoio e troca de conhecimentos. Em especial, ao Diego Pereira Dias, pessoa que me auxiliou na execução, na correção e na coordenação deste trabalho. De modo indireto, aos meus pais, aos meus irmãos e principalmente à minha esposa Jordana, pelo compressão e paciência.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

Luciano Menezes Junior

Outubro/2014

Orientador: Ramon Romankevicius Costa

Programa: Engenharia Elétrica

Os sistemas de controle de missão (SCM) são elementos fundamentais na robótica autônoma de maneira geral. É a partir dessa estrutura que se constrói os diversos métodos para elaboração de trajetórias e tomadas de decisão dentro do plano de missão. O trabalho apresentado cria um SCM modular, capaz de executar missões, com um bom nível de detalhamento, de modo totalmente autônomo em atividades de inspeção e monitoração de tubulações de dutos submarinos. O simulador virtual do ROV LUMA foi utilizado para a validação das técnicas implementadas. O modelo desenvolvido foi através do formalismo das Redes de Petri, cuja grande vantagem é a realização de análises comportamentais prévia do sistema.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DEVELOPMENT OF A MISSION CONTROL SYSTEM WITH PETRI NET

Luciano Menezes Junior

October/2014

Advisor: Ramon Romankevicius Costa

Department: Electrical Engineering

The mission control systems (MCS) are fundamental elements in autonomous robotics in general. The various methods for create trajectories and decision making within the mission plan are built from this structure. The work presented here, creates a modular MCS able of executing missions, with a good level of detail, in a fully autonomous manner in activities for inspection and monitoring of subsea pipelines. The virtual simulator ROV LUMA was used to validate the techniques implemented. The model was developed through the formalism of Petri Nets, whose the great advantage is the possibility to perform preliminary system behavioral analysis this objective.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas

AC	Escolha Assimétrica (*Assymmetric Choice*)
AI	Inteligência Artificial (*Artificial Intelligence*)
ASC	Embarcação de Superfície Autônoma (*Autonomous Surface Craft*)
AuRA	Inteligência Artificial (*Artificial Intelligence*)
AUV	Veículo Submarino Autônomo (*Autonomous Underwater Veichle*)
AVG	Veículo Guiado Autonomamente (*Automated Guided Vehicle*)
CRFC	Fibra de Carbono Reforçado por Compostos (*Carbon Fiber Reinforced Composites*)
CVL	Correlação de Distância e Velocidade (*Correlation Velocity Log*)
DEC	Controlador de Evento Discreto (*Discrete Event Controller*)
DoD	Departamento de Defesa (*Department of Defense*)
DSL	Linguagem Específica de Domínio (*Domain Specific Language*)
EKF	Filtro de Kalman Estendido
FC	Livre Escolha (*Free-Choice*)
FCE	Livre Escolha Estendida (*Extended Free-Choice*)
FEI	Filtro Estendido de Informações
GFRP	Fibra de Vidro Reforçada com Plástico (*Glass-Fiber Reinforced Plastic*)
GPS	Sistema de Posicionamento Global (*Global System Position*)
INS	Sistema de Navegação Inercial (*Inercial Navigation Systems*)
ISE	Engenharia Submarina Internacional (*International Submarine Engineering*)
ISR	Instituto de Sistemas e Robótica
IST	Instituto Superior Técnico
ITPM	Instituto de Tecnologia de Problemas Marinhos (*Institute of Marine Technology Problems*)
LBL	Linhas de Bases Longas (*Long Base Lines*)
MARCBot	Robô à Controle Remoto Avançado Multi-Função (*Multi-Function Advanced Remote Control Robot)*
MAS	Sistema Multi-Agente (*Multi Agent System*)
MCL	Linguagem de Controle de Missão (*Mission Control Language*)
MG	Grafo Marcado (*Graph Marked*)
MMC	Composto de Matriz de Metal (*Metal Matrix Composites*)

NASA	Administração Espacial e Aeronáutica Nacional (*National Aeronautics and Space Administration*)
NPS	Escola Naval de Pós-Graduação (*Naval Pos-graduate School*)
PEE	Programa de Engenharia Elétrica
PIG	Dispositivo de Inspeção de Dutos (*Pipeway Inspections Gate*)
PN	Redes de Petri (*Petri Net*)
PNBB	Blocos de Arquiteturas de Redes de Petri (*Petri Nets Building Block*)
Remus	Unidade de Monitoramento em Ambiente Remoto (*Remote Environmental Monitoring Units*)
ROS	Sistema Operacional de Robôs (*Robot Operation System*)
ROV	Veículo Submarino Operado Remotamente (*Remotely Operated Underwater Vehicle*)
SBL	Linhas de Bases Curtas (*Short Base Lines*)
SCM	Sistema de Controle de Missão
SED	Sistemas à Eventos Discretos
SLAM	Mapeamento e Localização Simultâneas (*Simultaneous Localization and Mapping*)
SM	Máquina de Estado (*State Machine*)
SMACH	Máquina de Estrado (*State MACHine*)
SPA	Sentir, Planejar e Agir
SPURV	Veiculo Submarino de Pesquisas de Auto-propulsão (*Self-Propelled Underwater Research Vehicle*)
TDL	Linguagem de Descrição de Tarefa (*Task Description Language*)
TMS	Sistema de Gerenciamento de Tether (*Tether Manager System*)
T-REX	Executiva e Tele-Reativa (*Teleo-Reactive EXecutive*)
USBL	Linhas de Bases Super-Curtas (*Ultra Short Base Lines*)
UUV	Veículo Submarino Não-Tripulado (*Unmanned Under Vehicle*)
VA	Veículos de Autônomos
VAA	Veículos Aéreos Autônomos
VMA	Veículos Marinhos Autônomos
VTA	Veículos Terrestres Autônomos
WHOI	Instituto Oceanográfico Woods Hole (*Woods Hole Oceanographic Institution*)

Lista de símbolos

$u(t)$	Entrada
$y(t)$	Saída
$x(t)$	Estado
$r(t)$	Sinal de Referência
t_0	Instante inicial
$x(t_0)$	Estado Inicial
$u(t) = \gamma(r(t), t)$	Lei de Controle
$\in$	Pertence
$\notin$	Não Pertence
$\cup$	União
$\cap$	Interseção
$\subset$	Contém
$\supset$	Contido
$\exists$	Existe
$\leq$	Menor ou Igual
$\geq$	Maior ou Igual
$\neq$	Diferente
$\equiv$	Equivalente
$\oslash$	Vazio
$\rightarrow$	Implicação
α	Alpha
γ	Gamma
μ	Mu
ψ	Psi

Capítulo 1

Introdução

O atual crescimento dos sistemas autônomos está diretamente ligado aos sucessos obtidos em aplicações civis, militares e comerciais. Graças ao desenvolvimento das tecnologias associadas a esses equipamentos, pesquisadores constroem as mais variadas soluções, implementadas nas plataformas de autonomia móvel em diversos campos da engenharia. A ideia deste primeiro capítulo é realizar uma breve análise dos principais veículos autônomos existentes, apresentando conceitos fundamentais, descrevendo a motivação, os objetivos e a organização desta dissertação.

1.1 Autonomia de um robô

Autonomia é uma expressão usada para descrever uma área de estudos de robôs parcialmente ou totalmente independentes. Um robô é dito autônomo se, durante o seu funcionamento, nenhum agente externo exerce interferência em sua atividade. Por outro lado, um robô é considerado semi-autônomo se, durante o seu funcionamento, for necessária a intervenção de um agente externo, mesmo que ele seja capaz de tomar certas decisões de forma automática.

A teoria dos Veículos de Autônomos (VA) é construída sobre esse conceito. Um VA é conhecido como uma plataforma não tripulada que possui comportamento próprio, com ou sem intervenção direta de um agente externo. Os VAs dispõem de uma diversidade enorme de configurações, podendo se submeter a adaptações que os possibilitam cumprir missões em diferentes ambientes, tais como: superfícies terrestres, ambientes marinhos, espaços aéreos e até espaciais (MURPHY, 2000). Define-se missão como um conjunto de tarefas que o robô desempenha a fim de concluir um objetivo.

(ENTREPRISE, 2006) cita veículos terrestres que podem se locomover através de rodas ou pernas. Outros exemplos são apresentados por (PASCOAL; SILVESTRE; OLIVEIRA, 2006), que detalha modelos de veículos aquáticos com formato de embarcações de superfícies, submersas (como os submarinos), ou híbridas (com-

binando características aquáticas e terrestres). Já em uma missão cujo objetivo é inspecionar fronteiras, ou áreas de desmatamento, a melhor alternativa são os aviões autônomos que têm amplo alcance visual (CAMBONE et al., 2005). Em missões aero-espaciais é usual o emprego de robôs autônomos com a finalidade de reduzir os custos operacionais e evitar o envolvimento de astronautas, pois a exposição humana é letal nessas condições (CRUZEN; THOMPSON, 2012).

De acordo com (PEREIRA, 2005), determinados aspectos devem estar presentes para endossar a utilização dos VA, entre eles estão:

- *Hostilidade do ambiente*: Torna-se inviável ou impossível a presença segura de um operador;
- *Viabilidade econômica*: Redução dos custos ao remover a participação do operador.
- *Aperfeiçoamento técnico*: A permanência do operador nas atividades não garante resultados satisfatórios, necessitando a intervenção de funcionalidades autônomas.

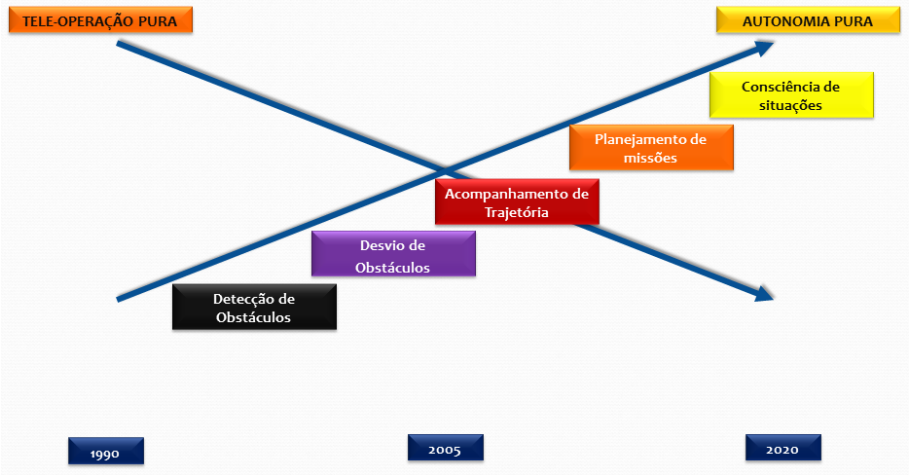


Figura 1.1: Representação da evolução dos robôs ao longo dos anos. Figura baseada em (PEREIRA, 2005).

1.1.1 Estrutura de um veículo autônomo

Outro aspecto relevante no projeto de um VA é o seu grau da autonomia (nível de "inteligência"). Esse parâmetro cria modelos operacionais bem distintos, divididos em três categorias:

- *Pré-programados*: São VAs projetados para missões simples, constituídas praticamente por atividades com um baixo nível de complexidade e limitadas a movimentos repetitivos.

- *Supervisionados*: Nesta categoria parte da missão é executada de forma automática, geralmente são eventos associados às camadas de níveis inferiores da hierarquia (controle de atuadores, motores, sensores). Por outro lado, as funções de alto nível (tomadas de decisão) são executadas por um operador.

- *Independentes*: São sistemas mais elaborados onde é incorporada uma arquitetura inteligente para obter as melhores escolhas durante uma missão. Além de um eficiente sistema de tomada de decisões, estes VAs possuem sistemas de diagnósticos de falhas e monitoração de estados, entre outros.

Dentro destes níveis de "inteligência" todo VA é organizado sobre uma estrutura composta em sub-sistemas responsáveis pelo funcionamento e gerenciamento de recursos. Resumidamente, (PEREIRA, 2005) e (BASSI, 2008) citam alguns desses sub-sistemas:

- *Sistema de Suporte, Monitoração e Detecção de Falhas*: responsável pelo controle e distribuição de energia, ao mesmo tempo monitora o consumo para os módulos embarcados. Ademais, permite a verificação de possíveis falhas no hardware e detecção de erros em comportamentos, gerando alternativas para contornar tais eventos.

- *Sistema de Comando e Controle*: combina as trajetórias de referências fornecidas pelo *Sistema de Controle de Missão* com todas as informações necessárias do *Sistema de Navegação*, criando comandos para conduzir o veículo na trajetória desejada.

- *Sistemas de Atuação*: atua na velocidade de deslocamento e/ou na inclinação do veículo fazendo com que essas informações se equiparem com as referências do *Sistema de Comando e Controle*.

- *Sistema de Navegação*: determina a posição e as velocidades angular e linear do corpo em relação a um referencial. Essas informações são obtidas pela instrumentação embarcada. Dentre esses instrumentos pode-se destacar a presença dos: profundímetros, giroscópios, acelerômetros, magnetômetros, etc. Ou se apropriando de dados de agentes externos, como: GPSs, posicionamentos acústicos etc. Essas informações são enviadas para os *Sistemas de Comando e Controle* e para o *Sistema de Controle de Missão*.

- *Sistema de Comunicação*: gerencia as informações de acordo com as ações previstas nos *Sistemas de Controle de Missão*. Esses dados podem ser trocados entre o veículo e o meio externo de cooperação, ou podem circular internamente entre os módulos dos subsistemas.

- *Sistema de Carga*: é responsável por armazenar e gerenciar os instrumentos que irão coletar os dados relativos às missões. Se preciso for, esse sistema pode transmitir os dados aquisitados para algum módulo externo. São vários sensores e atuadores que são subservientes a esse módulo, tais como: sonares, sensores de umidade, sensores de temperatura, câmeras de vídeo, iluminação, lasers, manipuladores, etc.

- *Sistema de Interface*: cria um elo de informações entre o veículo e o ponto de operação. Pode ser um simples monitor para acompanhamento de missões, ou pode ser um terminal iterativo, interferindo diretamente na conclusão da missão.

- *Sistema de Controle de Missão*: transforma o plano de missão (geralmente definido em uma linguagem de alto nível ou em uma plataforma dedicada) em um conjunto de comandos para cada um dos subsistemas. Esses subsistemas têm um relativo grau de discernimento e interpretação. Deve-se respeitar os níveis hierárquicos de informações durante a execução da missão e gerenciar as decisões de forma a alcançar o objetivo dentro dos requisitos especificados.

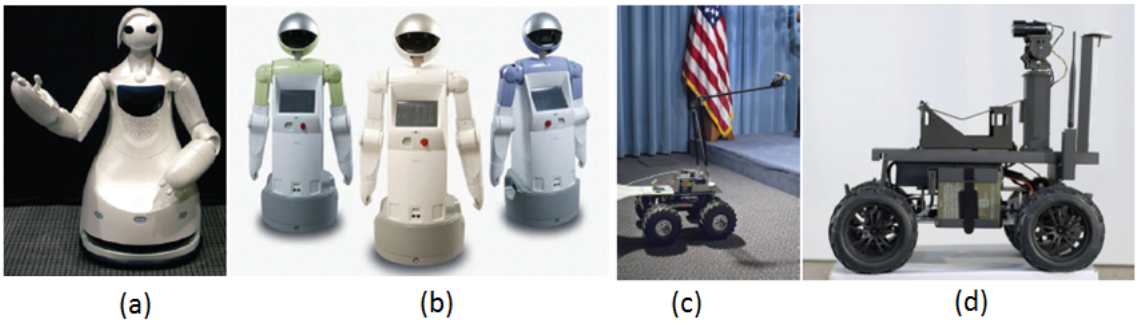


Figura 1.2: Humanoides desenvolvidos pelas empresas japonesas. (a) Robina da Toyota (b) Fujitsu da Enon - veículos desenvolvidos pelo U.S. Department of Defense (DoD) (c) Multi-Function Advanced Remote Control Robot (MARCBot) (d) BomBot. Fonte: (CHANG; BIAN; SHI, 2004).

1.1.2 Veículos aéreos autônomos

Os Veículos Aéreos Autônomos (VAA), conhecidos também como *drones* é qualquer aeronave que não precisa do piloto embarcado para ser guiada. Sua geometria, modelo, custo e aerodinâmica podem ser apresentados de diversas formas. Existem VAAs de asas fixas, que utilizam motores de propulsão, e há outros tipos que funcionam através de hélices, como os helicópteros (PEREIRA, 2005). Essa ferramenta desperta um grande interesse militar e é tratado como assunto de estratégico de defesa em muitos países. Histórias envolvendo VAAs nas guerras do Iraque (Operação

Tempestade no Deserto e Iraque Freedom) e no Afeganistão, citados em (CAMBONE et al., 2005), revelam que o domínio dessa tecnologia pode criar táticas de espionagens e soberania nacional. A grande maioria das missões dessa natureza é de longa duração, desempenhadas por veículos de grande porte equipados com instrumentos de ótima precisão. Operações utilizando os VAAs são vantajosas também pelo lado econômico, pois manter tropas deslocadas em território inimigo é muito mais oneroso do que manter VAAs e uma equipe de coordenação no local.

Além do interesse militar, outras áreas da ciência também demonstram apreço a esta tecnologia. A versatilidade e a possibilidade da cobertura em grandes áreas atraem pesquisadores de todas as naturezas. A vantagem desta plataforma é o baixo custo operacional.

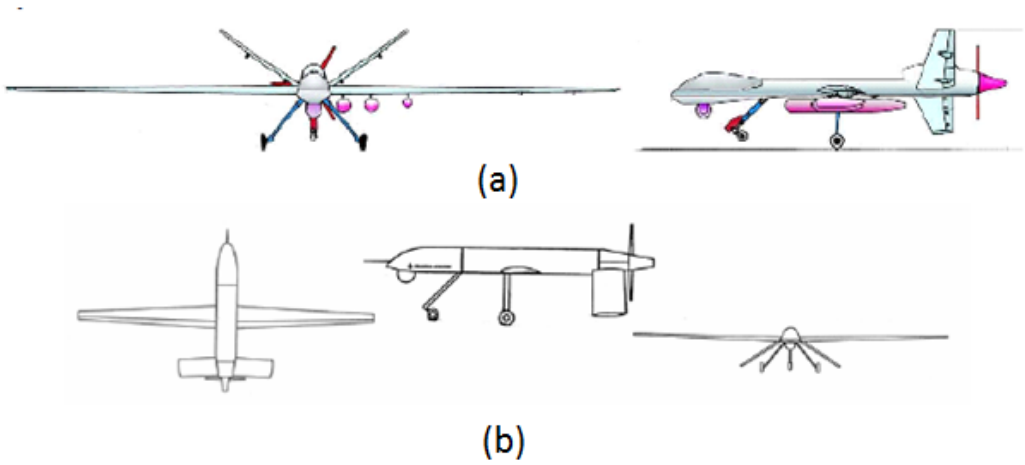


Figura 1.3: (a) de uso militar, I-Gnat-ER dentre várias missões executadas, ele foi empregado na segurança dos Chefes de Estado Reunião do G-8 em Alberta no Canadá em 2002. (b) O MQ-9 Predator B, é uma arma de guerra letal que é capaz de atingir alvos críticos e pode ser utilizado como um equipamento de espionagem. Figuras extraídas do documento (CAMBONE et al., 2005)

1.1.3 Veículos terrestres autônomos

Os Veículos Terrestres Autônomos (VTAs) representam uma categoria de robôs que se locomovem sobre os mais variados terrenos através de rodas, esteiras, pernas ou qualquer outra combinação mecânica que auxilie seu deslocamento. Há uma gama de veículos que podem ser adaptados para as mais variadas funções. Os casos do gigante Abramns Panther (aproximadamente 40 toneladas) e do singelo Dragon Runner, que pode facilmente ser carregado por qualquer ser humano adulto (PEREIRA, 2005), são exemplos dessas adaptações.

Assim como ocorre em outras áreas da robótica móvel, os VTAs se destacam em aplicações militares e civis. Intervenções militares são encontradas em: desarme

de minas terrestres, reconhecimento e apoio de tropas e neutralização de munições explosivas (NAVAL OPERATIONS, 2005). Na área civil pode-se destacar: inspeção em plantações agrícolas, operações de combates a incêndios, operações de busca e salvamento e missões em áreas contaminadas (ENTREPRISE, 2006).

O uso desta ferramenta em ambientes não estruturados tem sido alvo de intensas pesquisas ao longo dos últimos anos. Trabalhos em parcerias com institutos tecnológicos permitiram que fábricas e galpões de estocagem ganhassem projetos de destaque. Um exemplo disso foi a criação do veículo Automated Guided Vehicle (AGV) (ROCHA, 2001), robô guiado automaticamente que realiza armazenagem e distribuição de matérias-prima em uma fábrica de transformadores. O AGV se desloca entre as prateleiras utilizando faixas coloridas marcadas em solo que servem de referência aos sensores ópticos do veículo.

Figura 1.4: (a) de uso militar, o robô RONS, é um dispositivo autônomo que atua em áreas de difícil acesso humano, fazendo manipulação de substâncias, resgate de dispositivos e equipamentos. (b) Robô ARTS, semi-autônomo operado em plataforma móvel, utilizado pelas forças armadas dos EUA. Figuras extraídas de

(CAMBONE et al., 2005)

1.1.4 Veículos marinhos autônomos

A classe dos Veículos Marinhos Autônomos (VMAs) é composta por dois grandes segmentos de desenvolvimento: os denominados Veículos de Superfície Autônomos ou Autonomous Surface Craft (ASC) e os veículos submersos chamados de Autonomous Underwater Veichle (AUV). O primeiro grupo será apresentado de forma superficial na próxima seção. O segundo grupo será exposto de forma mais detalhada, descrevendo a evolução, destacando a importância no cenário atual e principalmente, focando os sistemas de controle de missão para AUVs, que representa o tópico principal desta dissertação.

Veículos marinhos autônomos de superfície

Nas últimas décadas, o desenvolvimento tecnológico na área naval favoreceu estudos científicos para aperfeiçoar a exploração dos oceanos. A partir desse progresso,

veículos de superfície, os ASC, têm surgido em várias partes do mundo nas mais diversas atividades, tais como: vigilância em águas abrigadas ou em mar aberto, recolhimento de amostras e objetos em situações de catástrofes (PASCOAL; SILVESTRE; OLIVEIRA, 2006). A hidrodinâmica e as dimensões do veículo variam de acordo com as atividades envolvidas. Lanchas, catamarãs, mono cascos, podem utilizar motores à combustão ou motores elétricos alimentados por baterias para propulsão.

(a) (b) (c)

Figura 1.5: (a) ASC monocasco tipo kayak da Robotic Marine Systems (b) Catamaran Delfim do Instituto de Sistemas e Robótica (ISR) (c) Catamaran Charlie do CNRISSIA em Itália do Instituto Superior Técnico (IST). Figuras retiradas de (PASCOAL; SILVESTRE; OLIVEIRA, 2006).

1.2 Motivação

Desde 2004, o Laboratório de Controle do PEE vem desenvolvendo um veículo submarino de operação remota (Remotely Operated Underwater Vehicle - ROV), denominado LUMA. Esse projeto pioneiro foi inicialmente concebido para inspecionar barragens e túneis de adução, que levam as águas das represas até as turbinas das hidrelétricas (GOULART, Junho 2007).

Após obter resultados satisfatórios operando em barragens, uma série de melhorias estruturais têm sido implementadas para adaptá-lo à cenários complexos. Em 2007, por exemplo, o desafio foi introduzir modificações para utilizar o ROV LUMA na Baia do Almirantado, Antártica, onde colaborou com um projeto envolvendo o censo da vida marinhas daquela região.

Dentre as melhorias realizadas, um avanço importante foi o sistema de controle e o simulador desenvolvido em (GOULART, Junho 2007). Utilizando técnicas de identificação, os parâmetros do sistema foram refinados o que permitam obter resultados de simulação bastante similares aos obtidos em testes práticos com o ROV LUMA.

Recentemente, a direção da COPPE enfatizou a importância estratégica do desenvolvimento de tecnologia de sistemas robóticos autônomos em sua diretriz. Financiado pelo CNPq, desde 2010 o ROV LUMA vem passando por uma série de

aperfeiçoamentos tais como: inclusão de uma eletrônica embarcada baseada em PC104, nova interface baseada em ROS, novos sensores, novos propulsores, novo sistema de baterias, etc. Em fase final de montagem, o ROV LUMA ganhará um hardware instalado que permitirá a sua utilização de forma híbrida, tanto como ROV quanto AUV. Para operar como AUV falta basicamente definir um sistema de controle de missão.

Uma aplicação imediata para o AUVs que está sendo desenvolvido é de interesse da Petrobras, que atualmente tem cerca de 6000Km de dutos submarinos, em condições estruturais pouco conhecidas, devido à dificuldade de se estabelecer uma rotina de acompanhamento e de inspeção. Neste caso há duas saídas: utilizar os ROVs para tal atividade, mesmo apresentando uma onerosa logística de funcionamento, ou uma segunda alternativa seria utilizar os AUVs.

Motivado por essa demanda técnica, o foco principal desta tese é apresentar um modelo de sistema de controle de missões para AUVs, empregado em inspeções de monitoração de tubulações de óleo e gás. Será apresentado um sistema teórico, capaz de realizar o planejamento de trajetórias, de modo totalmente autônomo, em condições de mar bem próximo das condições reais.

Toda teoria desenvolvida é aplicada a partir da adaptação do simulador apresentado na tese (GOULART, Junho 2007). Nesse trabalho, o autor focou no desenvolvimento de um ROV para inspeção de túneis de adução em usinas hidrelétricas. A semelhança entre a tese (GOULART, Junho 2007) e este trabalho é basicamente o veículo virtual (blocos de controle de configuração da dinâmica e os blocos dos acoplamento dos propulsores). A parte de comando foi totalmente reestruturada. O sistema poderá ser programado por um controlador de missão baseado em Redes de Petri. O cenário da missão é bem semelhante, porém, ao invés do iniciar a inspeção dentro do duto, o veículo inicia a inspeção em um ponto fora do duto. O projeto tem aspectos que o torna o mais genérico possível e não há a necessidade do domínio amplo de todas as funcionalidades que compõe um AUV para programá-lo.

1.3 Objetivo

O objetivo desta dissertação é criar um sistema capaz de definir e executar missões de um veículo autônomo submarino, que seja simples tanto da perspectiva do programador quanto do usuário.

Desafio

O desafio é criar um modelo flexível, genérico e simples para o usuário, podendo ser adaptado em qualquer outro tipo de missão compatível com o AUV utilizado.

O desempenho do sistema pode ser testado no protótipo de AUV desenvolvido no Laboratório de Controle da COPPE e, em uma eventual situação futura, em testes experimentais.

Metodologia

1. Estudar da história dos AUVs pesquisando detalhadamente suas principais partes;

2. Descrever o estado da arte do controle de missão, focando os sistemas predominantes nas aplicações voltadas aos AUVs mais importantes na literatura;

3. Desenvolver um sistema de controle de missão em redes de Petri, aplicando suas principais propriedades;

4. Permitir que esse sistema de controle seja flexível ao ponto de adaptá-lo à outras missões sem perder suas propriedades;

5. Qualificar o item anterior através de testes experimentais realizados no simulador virtual;

1.4 Organização da tese

A tese está organizada da seguinte forma:

- *Capítulo 2*: Esse capítulo começa fazendo um rápido resumo sobre a evolução dos veículos submarinos ao longo dos anos até chegar aos AUVs atuais. Depois desta introdução inicial, o capítulo detalha as principais partes que compõe um veículo submarino e finaliza com uma breve descrição de um AUV de uso comercial, incluindo o ROV LUMA.

- *Capítulo 3*: Faz uma abordagem ampla sobre os principais sistemas de controle de missão, divididos entre aplicações: militar, pesquisa, comercial e genéricas (categorias presentes na literatura). Um resumo sobre a evolução e estrutura de uma arquitetura de controle também está presente.

- *Capítulo 4*: Mostra a teoria dos sistemas à eventos discretos modeladas em redes de Petri. O capítulo também introduz os principais conceitos sobre as ações primitivas. A principal característica dessa estrutura é que suas propriedades são transmitidas das redes menores para as redes maiores.

- *Capítulo 5*: Traz uma breve introdução sobre inspeção em dutos submarinos, mostrando quais são os grandes desafios envolvendo este tipo de atividade.

Esse capítulo descreve a missão planejada com o passo-a-passo das ações tomadas pelo AUV. Uma bateria de testes é realizada em ambiente virtual, sendo apresentado uma análise dos resultados obtidos no final do capítulo.

- *Capítulo 6*: Esse capítulo apresenta as principais conclusões e sugere futuros trabalhos aos pesquisadores desta área.

-

Capítulo 2

Veículos Submarinos Autônomos

2.1 Introdução

Os veículos submarinos não tripulados - Unmanned Under Vehicle (UUV) - foram desenvolvidos para realizar tarefas consideradas inviáveis para os seres humanos quer seja por motivos técnicos ou por questões de segurança. Dentre os tipos de UUVs existentes, podemos destacar os veículo submarinos de operação remota - Remotely Operated Underwater Vehicle (ROVs) - e os veículos submarinos autônomos - Autonomous Underwater Vehicle (AUV) - como as principais ferramentas empregadas no meio industrial e acadêmico.

O ROV é uma ferramenta tele-operada, por meio de uma estação de trabalho. É necessária a utilização do cabo umbilical para estabelecer esse elo entre o veículo e a superfície. Através deste cabo que se transmite todos os dados dos sensores e comandos para o controle de movimento da ferramenta. Há necessidade do uso de uma embarcação dedicado a ferramenta, bem como manter uma tripulação capacitada e treinada a fim de viabilizar as operações (GOULART, Junho 2007). Determinados tipos de missões exige-se que essa embarcação possua dispositivos caros e complexos. Por exemplo, para operações que envolvam mergulhos profundos e que dependa posicionamento preciso, as embarcações devem ter um equipamento chamado *Tether Manager System* (TMS), dispositivo que atenua o efeito da correnteza, gerenciando o comprimento do cabo umbilical. Além disso é preciso estar equipada com um sistema de controle de posição, orientado por GPS, de alta confiabilidade e exatidão. Ademais, para uma operação ser bem sucedida, é fundamental que ela seja executada com habilidade pelo operador (RIBAS; RIDAO; NEIRA, 2010).

Os primeiros AUVs foram criados na década de 60. Desenvolvidos sob a coordenação da marinha estadunidense, o objetivo era executar missões de inspeção e atividades militares (WERNLI, 2001). Nos anos 70, universidades, institutos e organizações não governamentais iniciaram a pesquisa experimental do AUV. A Universi-

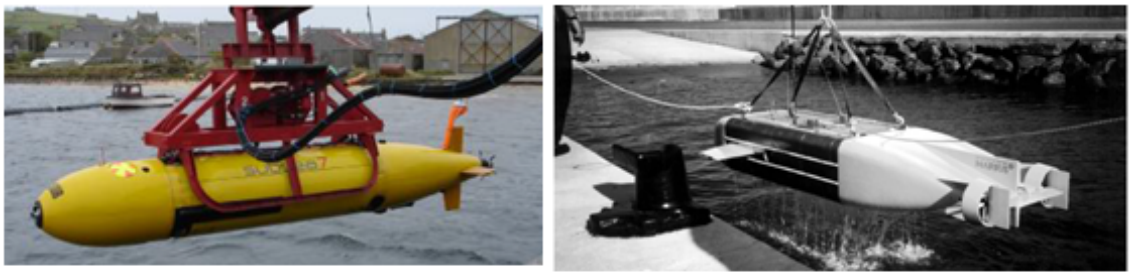
dade de Washington desenvolveu ensaios com o protótipo *Self-Propelled Underwater Research Vehicle* (SPURV), na região Ártica. Na mesma época, a Universidade de New Hampshire desenvolveu o AUV EAVE em conjunto com a marinha estadunidense e Universidade de San Diego. Os russos, neste mesmo período, também deram suas contribuições para alavancar a tecnologia. O *Institute of Marine Technology Problems* (IMTP) e o *Russian Academy of Science*, iniciaram seus programas em 1976 através de um AUV capaz de operar à 6000 metros. Dois protótipos foram criados para esta atividade, o L1 e o L2 e os testes foram feitos nos Oceanos Atlântico e Pacífico (BLIDBERG; ALI; CHECHIN, 1995). Nos anos 80, os estudos na área de AUV tiveram um avanço significativo. O progresso da eletrônica e o surgimento de processadores cada vez menores e mais eficientes, permitiu aos pesquisadores a introdução de algoritmos complexos embarcados nos módulos dos veículos autônomos. Em continuidade, outros setores da indústria começaram a se interessar pela tecnologia. A possibilidade de operar em águas profundas aproximou indústria de óleo e gás das pesquisas, com o financiamento de importantes projetos. Nas décadas de 90 e anos 2000, com a consolidação dos sistemas operacionais, as plataformas de AUV se tornaram bastante robustas sendo capazes de manipular um grande volume de informações. Por outro lado, com o financiamento em pesquisas através da iniciativa privada impulsionada pelos estudos de oceanografia, ajudaram a delinear as missões bem como os tratamentos dos dados obtidos durante as inspeções (BLIDBERG, 2001). Nestas décadas, iniciou-se uma nova fase para os AUV, preparando para o processo de comercialização da ferramenta.

Mesmo sendo economicamente oneroso, o AUV é utilizado em várias aplicações, tais como: localização de navios naufragadas, (BALLARD, 1987), mapeamento do leito marinho (YOERGER et al., 2007), detecção e resgate de objetos (KONDOA H. & URA, 2004), segurança de portos e localização de minas (WILLCOX et al., 2001) e em estudos científicos no leito marinho (KENCONLEY, 2014), (CURTIN; BELLINGHAM, 2001).

Nas próximas seções desse capítulo será apresentado detalhes da estrutura de um AUV, destacando importantes elementos construtivos, tais como: hidrodinâmica, navegação e controle, propulsão, orientação e flutuabilidade, sensores e instrumentação, alimentação e posicionamento.

2.2 Casco

Umas das mais importantes partes de um AUV é o casco. Segundo (ALLMENDINGER, 1990) existem cascos das mais variadas formas, modelos e tamanhos, projetados de acordo com as particularidades de cada missão. Apesar da grande diversidade de cascos existentes, todos eles respeitam alguns critérios de construção,



(a) (b)

Figura 2.1: (a) AUV Subsea7 GEOSUB AUV, este veículo proporciona boa autonomia e resistência utilizando bons sensores, garantindo qualidade na pesquisa (EVANS et al., 2009)(b) MARIUS AUV, veículo de desenvolvimento português utilizados em diversas aplicações civis (OLIVEIRA et al., 1996)

entre eles estão:

- Pressão e profundidade máximas;
- Faixa de temperatura de operação;
- Resistência mecânica à impactos;
- Permeabilidade à água;
- Apelo visual;
- Acessibilidade para intervenções;
- Versatilidade;
- Praticidade;
- Restrições para futuras expansões;
- Tamanho;
- Peso;
- Resistência a corrosão e a produtos químicos;

Além dos itens acima citados, a hidrodinâmica é outra importante característica exigida na construção. Na prática, isso significa que o projeto do casco deve levar em consideração a redução dos efeitos das forças de arraste. Projetos bem sucedidos de hidrodinâmica reduzem o consumo de energia e favorecem a estabilidade durante a navegação do AUV. A geometria ideal para o casco é a cilíndrica, pois, reduz os efeitos das forças externas, aprimora o desempenho de navegação e suporta altas pressões (ROSS, 2006).

Segundo (ROSS, 2006), pode-se destacar algumas vantagens desta geometria, tais como:

- Ótima estrutura para resistir aos efeitos das pressões hidrostática;
- O aumento do volume interno pode ser conseguido se ampliado o comprimento;
- Facilita manipulação e armazenamento;

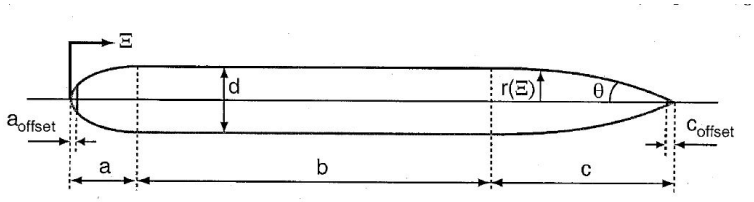


Figura 2.2: Figura do casco do barco retirada do documento (CUTIPA-LUQUE, 2012)

Segundo (CUTIPA-LUQUE, 2012), o casco da Figura 2.2 pode ter um melhor desempenho hidrodinâmico, reduzindo as forças de arraste caso seu projeto leve em consideração as seguintes relações:

$$r_{proa} = \frac{d}{2}\left[1 - \left(\frac{x + a_{off} - a}{a}\right)^2\right]^{\frac{1}{n}}, \tag{2.1}$$

$$r_{médio} = \frac{d}{2}, \tag{2.2}$$

$$r_{popa} = \frac{d}{2} - \left[\frac{3d}{2c^2} - \frac{\tan\theta_m}{c}\right](x - l_f)^2 + \left[\frac{d}{c^3} - \frac{\tan\theta_m}{c^2}\right](x - l_f)^3 . \tag{2.3}$$

Conforme (2.3), a geometria de cascos esbeltos facilitam a navegação favorecendo a hidrodinâmica, sugerindo a relação $l/d \geq 6$. A forma geométricas da proa e da popa pode ser alteradas pelos coeficientes a e θ.

Uma desvantagem da forma cilíndrica é o aparecimento de cavitação. Mesmo em baixas velocidades, ela é um fenômeno que ocorre pela distribuição de pressão na estrutura do casco. (PASTER, 1986). A geometria do veículo provoca diferentes velocidades locais distribuídas ao longo do seu corpo e, como consequência, surge uma pressão distribuída no casco. Essa pressão tem seu menor valor na área de maior curvatura, e no momento que ela se iguala à pressão exercida pela água, há um aumento da temperatura na região e bolhas aparecem. Esse fenômeno eleva o índice de turbulências e, dependendo da magnitude, pode danificar o veículo (ROSS, 2006). O formato frontal do AUV (nariz) é mais eficiente quando se assemelha ao formato de gota d'água, essa geometria reduz a instabilidade e as forças de arraste cerca de 2 a 10 vezes em relação à forma esférica (PASTER, 1986).

O material para a construção do casco também tem sua importância dentro do projeto. É desejável que ele apresente uma boa resistência e possua características anti-corrosivas. No passado, usava-se muito o aço pela alta resistência mecânica e preço acessível, mas como desvantagem, a relação peso resistência é baixa. O alumínio tem uma melhor relação entre peso e resistência em comparação ao aço, mas é susceptível a corrosões. O titânio tem a melhor relação peso resistência entre todos os metais estudados, além de ser inerte as corrosões. Porém, o alto valor agregado de mercado, o torna inviável para esta aplicação.

O material mais comum utilizado para aplicações submersas é o *Glass-Fiber Reinforced Plastic* (GFRP) ou fibra de vidro. Além do baixo custo e boa resistência, esse material não sofre com reações químicas e corrosões. Da mesma forma que o (GFRP), o material *Carbon Fiber Reinforced Composites* (CFRC) ou fibra de carbono, também tem uma ótima relação peso-resistência, mas o seu custo chega a ser até três vez acima do valor da fibra de vidro. Outro material em estudo para veículos submarinos, mas não tão comum é o *Metal Matrix Composites* (MMC). Ele possui vantagens técnicas sobre os GFRP e CFRC, mas o valor de mercado o torna pouco atraentes, cerca de até 15 vezes mais caro que o GFRP, por exemplo.

O plástico acrílico tem suas vantagens, não corrói e tem boa relação entre resistência e peso. Inclusive esse material tem sido empregado com sucesso em visores resistente a pressão (STACHIW, 2004). A desvantagem está em sua alta transparência que pode dificultar as operações subaquáticas. Para projetos com baixos investimentos o PVC pode ser uma boa alternativa. Esse material é muito usual em sistemas que operam em baixas profundidades (MONTEEN; WARNER; RYLE, 2000). A tabela 2.1, mostra as propriedades de cada material citado e apresenta uma relação de densidade, resistência e tensão de cada um deles.

Tabela 2.1: propriedades dos materiais adaptada de (CHANG; BIAN; SHI, 2004)

Material	Densidade (kg/dm^3)	Resist. à Def. (MPa)	Mód. de Tensão (GPa)	Resist. Espec. (KNm/Kg)
Aço de Alta Resistência	7,86	550	207	70
Liga de Alumínio	2,9	503	70	173
Liga de Titânio	4,5	830	120	184
GFRP	2,1	1200	65	571
CFRP	1,7	1200	210	706
MCC	2,7	3000	140	1111
Acrílico	1,2	103	3,1	86
PVC	1,4	48	35	34

2.3 Propulsão, mergulho e flutuação

Os *thrusters* - propulsores dos AUVs- são leves, robustos, precisos e o tipo de recurso mais utilizados para o deslocamento. Em (VALAVANIS et al., 1997) relaciona cerca de 25 modelos de AUVs em seu artigo, sendo que a maioria deles utilizam os *thrusters* para movimentos horizontais. O movimento vertical pode ser feito de dois modos: pelos*thrusters* ou por mecanismos de flutuação.

Técnicas de deslocamentos através dos *thrusters* exigem que o controle de flutuabilidade do veículo esteja ligeiramente positiva. Deste modo, em caso de pane elétrica, o veículo retorna à superfície. A desvantagem desta aplicação é o consumo excessivo de energia, já que os *thrusters* devem permanecer ligados para manter a profundidade a todo momento. Nesta linha de pesquisa, existe o desenvolvimento de (CAVALLO; MICHELINI, 2004) onde um sistema articulado (punho) fixado na base do *thruster* permite a mudança de direção em dois graus de liberdade, possibilitando o deslocamento horizontal e vertical. O veículo descrito em (MAURYA et al., 2007) também combina movimentação vertical e horizontal, mas ao invés de adotar um punho articulado, utiliza hidroplanos parecido com os empregados nas asas das aeronaves para deslocamento vertical.

Na categoria dos mecanismos de flutuação, os mais populares são: i) tanque de lastro de pistão, ii) bomba hidráulica com base em sistema de lastro, iii) sistema de ar comprimido e iv) bombeamento. O tanque de lastro de pistão é um dos sistemas mais comuns. Seu princípio de funcionamento é muito similar ao de uma seringa. Uma extremidade do cilindro está ligada ao ambiente e o movimento do pistão faz com que a água entre ou saia. Quando é necessário o movimento de descida, o pistão suga a água para dentro do cilindro tornando o AUV com flutuabilidade negativa. Por outro lado quando é solicitado a subida, o pistão empurra a água para fora do cilindro tornando o AUV com flutuabilidade positiva. Para obter uma maior precisão de movimento, um pistão com acionamento linear pode ser empregado, possibilitando uma melhor estabilidade por uma simples malha de controle.

O sistema de bombeamento hidráulico é similar ao do tanque de lastro, entretanto a grande diferença está na utilização de um reservatório interno com fluido hidráulico e uma bomba para movimentar o pistão linear. A flutuabilidade é controlada pelo fluido hidráulico que atua o pistão introduzindo água nos cilindro. Sistemas de ar comprimido são alternativas comuns em algumas classes de submarinos. O mecanismo é composto por um cilindro de ar comprimido, um tanque de água e uma válvula normalmente fechada nas respectivas entradas (WASSERMAN et al., 2003). Por exemplo, no caso de um empuxo positivo, os tanques são preenchido com ar comprimido e no caso de empuxo negativo são preenchidos com água. (TANGIRALA S. & DZIELSKI, 2007) mostram um sistema de flutuabilidade que contém

dois tanques que se comunicam ao ambiente externo. Se for necessário o deslocamento para baixo, os tanques se enchem com água. Caso for necessário flutuar, a água é bombeada para fora dos tanques.

2.4 Instrumentação

A maioria dos sensores que compõe o AUV estão relacionados à sua localização. Por exemplo, os sensores de pressão são capazes de medir a força externa exercida pela coluna d'água convertendo os valores em profundidade (WILLIAMS et al., 2000). Para medir velocidade, existem várias formas. Uma delas é descrito por (MODARRESS et al., 2007) que usa elementos de difração ótica para medir o deslocamento das micro-partículas presentes na água. A velocidade do AUV é estimada através da passagem destas micro partículas por um feixe de luz de dispersão. Os sensores são muito pequenos e precisos, porém podem sofrer interferências por mudanças de temperatura e pressão. O grande desafio deste método é a prevenção contra eventuais correntes marinhas, que podem mascarar as reais velocidades. Em operações realizadas próximas ao leito marinho pode-se utilizar os sensores tipo Doppler, que são ondas sonoras que tocam o chão e retornam ao sensor revelando a leitura de velocidade (BOLTRYK et al., 2004). Para melhorar a posição estimada é usual o emprego dos métodos de filtragem de Kalman (LEE et al., 2007). Outro instrumento de medição de velocidade é o *Correlation Velocity Log* (CVL), cujo princípio de funcionamento é idêntico aos sensores Doppler. A única diferença é a emissão de dois pulsos consecutivos, tornando essa técnica a mais precisa para aplicações em baixas velocidades (BOLTRYK et al., 2004).

Os *Inercial Navigation Systems* (INS) são sensores que medem aceleração e velocidade angular. Podem ser usados para medição da posição pela combinação dos acelerômetros, giroscópios e magnetômetros (STUTTERS et al., 2008). O acelerômetro mede aceleração linear, o giroscópio mede a velocidade angular e o magnetômetro fornece as informações sobre a posição do norte magnético. Com esses três sensores é possível analisar o movimento total do robô. Existem INSs equipados com giroscópios à laser que elimina os efeitos mecânicos do atrito e aumenta a precisão das medidas. A fusão sensorial diminui os erros das estimativas. Utilizando filtro de Kalman, vários sensores podem ser empregados simultaneamente aumentando assim a precisão da medição (MAJUMDER; SCHEDING; DURRANT-WHYTE, 2001).

Para operações autônomas a detecção de obstáculos permite que o veículo localize obstruções físicas ao longo da trajetória e crie alternativas para evitar colisões. Dentre os possíveis instrumentos que podem realizar essa função estão os sonares e câmeras. Essa é a configuração sugerida por (MAJUMDER; SCHEDING; DURRANT-WHYTE, 2001), que descreve a importância da utilização de, no mí-

nimo, esses dois instrumentos para a detecção de obstáculos. Segundo esta proposta, sensores de ultra-som localizam objetos à longas distâncias, onde as câmeras de vídeos são quase inoperantes. Esses sonares emitem ondas de altas frequência que percorrem longas distâncias, embora configuração com ondas de baixa frequência detalhes nas imagens (TOAL et al., 2005). Ainda neste contexto, (TOAL et al., 2005) desenvolveu um novo conceito sobre detecção de obstáculos a partir do uso de fibras óticas com dois diferentes sensores. Um deles utiliza uma técnica sem contato, emitindo um feixe de luz e quando houver houver um obstáculo na frente, a luz refletida é detectada. O segundo sensor opera por contato. Um feixe detectável de luz dentro da fibra é alterado quando ela toca um objeto. Outra proposta de sistema anti-colisões foi apresentada por (WILLIAMS et al., 2000), cujo informações do profundímetro são combinadas com imagens geradas por um sonar para a construção do mapa do ambiente.

2.5 Fonte de alimentação

A bateria elétrica é o tipo de armazenamento de energia mais utilizados nos AUVs ainda que em (JALBERT et al., 2003), é citado o emprego de energia solar para algumas categorias especiais. Algumas vantagens tais como, operação silenciosa e facilidade para controlar a velocidade do veículo, a torna popular dentre as fontes de energia. Devido à sua composição química, as baterias de zinco e de prata tiveram uma grande importância durante a década de 40, oferecendo bom desempenho em picos de potência apesar de baixos ciclos de recargas. Entretanto, desenvolvidas mais recentemente e mais atraentes, as baterias de íons de lítio são um dos modelos mais usados na atualidade (WINCHESTER et al., 2002).

As baterias podem ser primárias ou secundárias, ou seja, não carregáveis ou carregáveis respectivamente. Em geral a maioria das baterias empregadas em AUVs são secundárias, apesar das baterias primárias serem mais eficientes. No grupo das baterias primárias (BRADLEY et al., 2001) descreve a alcalina como a mais popular, eficiente na relação custo benefício, embora apresente sensibilidade a variações de temperatura e eventuais vazamentos de gás. Apesar de mais caras, as baterias primárias compostas de lítio possuem alta densidade de energia e são bastante eficientes. Na família das baterias secundárias destacam-se as de chumbo-ácido (muito populares, porém podem apresentar vazamentos de hidrogênio), as baterias de níquel-cádmio (que apresentam uma curva suave de descarga) e as de zinco e níquel (que possuem um bom ciclo de vida e uma boa densidade de energia). As baterias de lítio destacam-se pela melhor densidade de energia. A sua desvantagem é a necessidade de um complexo circuito eletrônico para operação.

(TAKAGAWA, 2007) relata sobre as células de combustível para a alimentação

de AUVs. Apesar da maior capacidade de potência em relação a outras baterias, as células de combustíveis necessitam de um sistemas de compensação de pressão instalado no vaso de abrigo. Para contornar este problema é proposto um mecanismo semelhante aos de compensação de pressão para baterias e recipientes de combustíveis e oxidantes. As células podem diminuir os ciclos de carga do veículo se caso o composto for armazenado em formato de alta densidade energética. As células de combustível, que operam em hidrogênio e oxigênio, são fontes de energia atraentes para AUVs, pois são eficientes, silenciosas, compactas e de fácil manutenção. A energia total fornecida por uma célula de combustível é limitada apenas pelo combustível e oxigênio disponível (HABERBUSCH et al., 2002). A Tabela 2.2 mostra um apanhado geral das principais baterias aplicadas em AUVs.

Tabela 2.2: característica dos tipos de energia adaptada de (BRADLEY et al., 2001)

Material	Densidade de Energia (kg/m^3)	Compensação de Pressão	Desgaseificação	Ciclos
Alcalina	140	não	provavelmente	1
Lítio	375	não	não	1
Chumbo-Ácido	32,5	sim	sim	100
Níquel-Cádmio	33	não	provavelmente	100
Níquel-Zinco	58,5	sim	não	500
Íons de Lítio	144	não	não	500
Polímero de Lítio	193	sim	não	500
Óxido de Prata	100	não	sim	500

2.6 Sistema de localização

O sistema de posicionamento global - *Global System Position (GPS)* - é uma poderosa ferramenta capaz de localizar, com grande precisão qualquer objeto na superfície da terra. Entretanto, para que haja a comunicação, o receptor precisa se encontrar dentro linha da visada dos satélites em órbita. Isto significa que a localização não é acessível aos objetos que estão em ambientes fechados ou em ambientes subaquáticos, onde as ondas de rádio enviadas pelos satélites não conseguem penetrar. Por esse motivo, a utilização de GPS na localização de veículos submarinos se torna totalmente inviável.

Com a ausência de um sistema de localização externo, os AUV são obrigados a utilizarem outros métodos. Por exemplo, para determinar a localização de um veículo submarino é integrar no tempo o sinal de velocidade (para sensores de velocidade) e integrar duas vezes sinal de aceleração (para sensores inerciais). O grande problema é que as medições geradas por esses instrumentos acumulam erros inerentes

às interferências das correntezas dos oceanos (LEE et al., 2007). Como consequência, os cálculos de navegação estimada são comprometidos, gerando um erro de posição. Outro problema parecido ocorre com os sensores efeito Doppler. O erro cresce significativamente com o aumento da distância percorrida, mesmo utilizando filtro de Kalman. Portanto, esse método para estabelecer a posição é impraticável em longas distâncias.

Para determinados casos, a utilização da navegação acústica torna-se uma solução viável de localização. Ao contrário das ondas eletromagnéticas, a energia acústica tem uma boa propagação nos oceanos, assim, transponders hidroacústicos é uma boa alternativa para orientação dos AUVs. As técnicas mais utilizadas para este fim são: *Long Base Lines* (LBL), *Short Base Lines* (SBL) e as *Ultra Short Base Lines* (USBL) (LEONARD et al., 1998). Todas elas utilizam um conjunto de transdutores externos (balizas localizadas no leito marinho, ou na própria embarcação de apoio), que auxiliam na navegação. O cálculo da posição é determinado através do intervalo de tempo que a onda acústica leva para partir do transponder (localizado no AUV) e retornar através das balizas auxiliares. Este tipo de técnica pode ser muito bem empregado em áreas limitadas ou águas abrigadas. Entretanto em casos mais específicos (onde o objetivo é cobrir grandes distâncias), esta configuração se torna ineficiente. A necessidade da integração de sistema externo ao AUV bem como as limitações nas áreas de trabalho, inviabilizam a utilização deste método em várias missões.

Os modelos de localização geofísicos, segundo (LEONARD et al., 1998), podem ser empregados em situações onde se tem um mapa a priori, da região a ser explorada. Nestas condições, necessita-se da instalação de balizas auxiliares. A aproximação de uma referência global é calculada (pelo emprego de medições geofísicas, como batimétricos, magnetômetros e gravitacionais) e permite a localização precisa do AUV. Mesmo assim, esse recurso não é simples. O mapa carregado na memória do veículo é confrontado com inúmeras informações, oriundas dos sensores. Para estabelecer, por exemplo, a correlação entre informações do mapa e os dados dos instrumentos é necessário uma analogia entre as leituras externas e as referências relacionadas na memoria. A resolução do mapa também interfere na localização do AUV. Excesso ou falta de detalhes podem mostrar ou esconder importantes pontos para a localização do veículo. O grande desafio desta técnica é criar referências confiáveis para a localização do AUV.

Os sistemas de mapeamento e localização simultânea - *Simultaneous Localization and Mapping* (SLAM) - é uma abordagem que busca uma solução aos problemas de localização em ambientes desconhecidos. A ideia central é criar um sistema de mapeamento global de qualquer ambiente, com ou sem conhecimento prévio. Portanto, o mapa é construído de acordo com deslocamento do veículo. Várias técnicas

com diferentes abordagens são propostas, tais como: Filtro Estendido de Informações (FEI), Filtro de Kalman Estendido (EFK), Modelos Gaussianos de Incerteza, Mapeamento estocásticas desassociadas, filtros comprimidos, mapas sequencias, e o sub-mapa de filtros locais (THRUN; BURGARD; FOX, 2005). Em resposta ao crescimento do SLAM e o sucesso do emprego das técnicas de mapeamento e localização, as apostas estão nas mais diversas áreas. Aplicações em ambientes abertos, em aeronaves e em veículos submarinos mostram a versatilidade deste recurso. Apesar do futuro promissor, há algumas questões que precisam ainda ser aperfeiçoadas (RIBAS; RIDAO; NEIRA, 2010).

2.7 Dinâmica e arquitetura de controle

O funcionamento do AUV requer o modelo matemático para determinar a dinâmica do veículo. De acordo com (CUTIPA-LUQUE, 2012), o modelo pode ser obtido por equações empíricas e semi-empíricas, recebendo do nome de modelo nominal, que apresenta um certo grau de incerteza devido às aproximações e dinâmicas não modeladas. Como resultado, são gerados dezenas de coeficientes associados aos termos hidrodinâmicos que pode ser analisado com maiores detalhes em (CUTIPA-LUQUE, 2012) e em (FOSSEN, 1994).

Após determinar um modelo para o veículo cria-se o controlador. O controle escalar (baseado na formatação das funções de transferência de malha aberta), embora tenha-se mostrado eficiente às aplicações industriais, não garante condições de estabilidade e desempenho robusto ao controle de AUVs. Todas estas fragilidades são manifestadas quando o sistema apresenta desequilíbrio, perturbações e incertezas nas variáveis observadas. No levantamento dos parâmetros de controle do veículo, observa-se um acoplamento entre as coordenadas do sistema. Isso ocorre entre os movimentos de balanço e guinada, entre os movimentos de caturro e arfagem além de estar presente movimentos acoplados entre coordenadas verticais e as coordenadas horizontais (CUTIPA-LUQUE, 2012). Uma estratégia que pode ser utilizada para tornar o movimento do veículo menos complexo é o desacoplamento dos movimentos verticais e horizontais (WILLIAMS et al., 2000), (MAURYA et al., 2007).

Devido à dificuldade de controlar a dinâmica do veículo utilizando apenas um controlador, frequentemente se utiliza um controlador de ganho programado (KAMINER et al., 1995). Nesta abordagem, um número finito de controladores é desenvolvido para um número finito de modelos linearizados aplicando a teoria de controle robusto H_∞. Essa pode ser uma boa alternativa para o controle do AUV. Diversos artigos e estudos relacionam uma grande variedade de controladores para aplicações em AUVS que podem ser vistos em (CUTIPA-LUQUE, 2007), (FOSSEN, 1994), (JALVING, 1994).

Outro tópico de grande importância na robótica móvel é a arquitetura do controle. Essa estrutura define onde e como as instruções são executadas com a finalidade de completar as missões programadas. Em seu artigo, (VALAVANIS et al., 1997) demonstra algumas abordagens utilizadas em projetos de AUVs. Ele cita as técnicas *Deliberativas*, onde as informações são centralizadas em uma unidade de processamento, permitindo reproduzir o ambiente de acordo com a metodologia: Sentir, Planejar e Agir (SPA). É uma estrutura bem definida, mas com sérios problemas de flexibilidade à ambientes dinâmicos. Outra arquitetura abordada e desenvolvida por (BROOKS, 1986), permite que diferentes níveis de informações trabalham em paralelo, no entanto, divididas em camadas. Esta arquitetura é robusta e apresenta um verdadeiro comportamento reativo dinâmico.

Pelo excesso de informações e pela ausência de uma central de informações, esses sistemas encontram dificuldades de sincronização. E finalmente, tem-se a arquitetura *Híbrida*, que é a integração dos modelos citados anteriormente, *Deliberativas* e *Reativo*. A proposta é dividida em dois níveis: O nível mais alto usa a metodologia SPA, enquanto nível mais baixo utiliza a metodologia de subsunção para controle do hardware. (MEDEIROS, 1998) relaciona os principais tipos de arquiteturas e realiza um interessante resumo com as vantagens e as desvantagens de cada uma dessas abordagem. Na camada deliberativa da arquitetura de controle está o sistema de controle de missão. Ele é responsável em transformar o plano de missão (geralmente definido em uma linguagem de alto nível) em um conjunto de comandos para cada um dos subsistemas. Esses subsistemas têm um relativo grau de discernimento e interpretação. Deve-se respeitar os níveis hierárquicos de informações durante a execução da missão e gerenciar as decisões de forma a alcançar o objetivo dentro dos requisitos especificados.

2.8 Aplicação comercial

2.8.1 REMUS

O resultado de décadas de desenvolvimento na área de AUVs provocaram o aparecimento de poderosas ferramentas. Uma delas é o *Remote Environmental Monitoring UnitS* (REMUS) desenvolvido pelo *Oceanographic System Laboratory*. É um dos veículos mais requisitados do mundo, sempre participando de ensaios, estudos científicos e missões da marinha estadunidense (ALLEN et al., 1997).

O REMUS é um pequeno AUV com cerca de 19 cm de diâmetro e $1,6$ metros de comprimento (vide Figura 2.3). Ele está equipado com sonares, sensor Doppler, sensor de condutividade, temperatura e profundidade. Sua operação é controlada por uma placa PC-104 que está montado sobre a placa-mãe com 16 portas conver-

soras analógica/digital. O veículo é equipado com três motores de propulsão, com controle de yaw e picth.

Ao longo de seu desenvolvimento esse veículo sofreu melhorias em sua velocidade, resistência e segurança. A segurança foi aperfeiçoada com uma combinação da redução das forças de arraste e uso de nova tecnologia de baterias. Melhorias na mecânica nos eixos dos propulsores resultaram em um ganho de velocidade de 2m/s para aproximadamente 3m/s. Mais detalhes desse veículo pode ser encontrado no artigo (ALLEN et al., 1997).

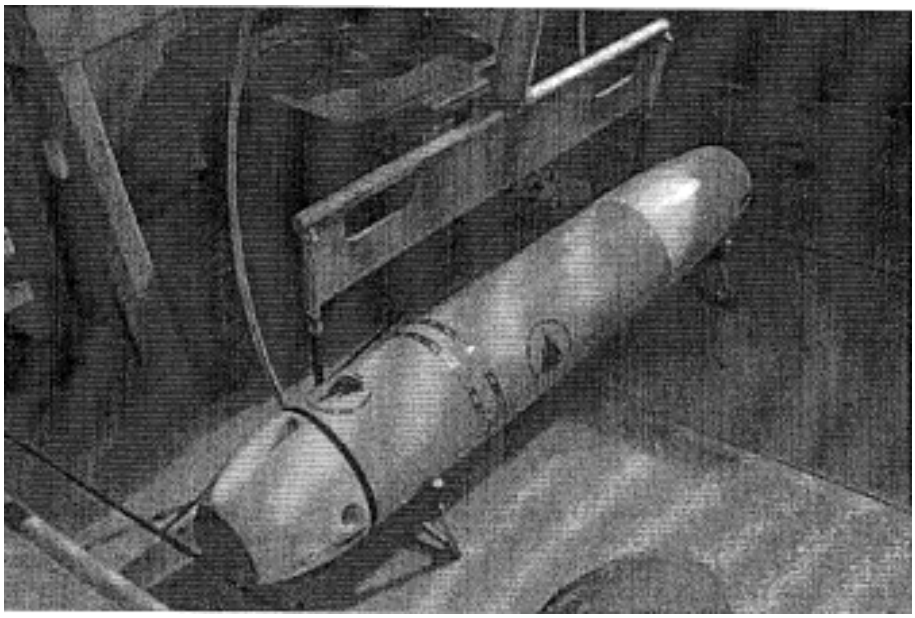


Figura 2.3: Foto do AUV Remus (ALLEN et al., 1997)

2.8.2 ROV Luma

O ROV Luma é um veículo tele-operado desenvolvido pelo Laboratório de Controle do PEE em parceria com a Ampla cujo objetivo é inspecionar barragens e túneis de adução que levam as águas das represas até as turbinas das hidrelétricas. Atualmente, depois de algumas melhorias, o veículo está habilitado a receber um sistema de controle de missão, podendo tornar assim, um veículo autônomo para desempenhar diversas tarefas.

O projeto mecânico do LUMA é do tipo estrutura aberta (open frame), como mostrado na Figura 2.4, e suas dimensões são: $63cm$ de altura, $88cm$ de largura e $1m$ de comprimento. O peso aproximado é de $40kg$. O sistema é composto por:

- Estrutura mecânica em alumínio;
- Dois vasos de pressão em PVC para a eletrônica e as baterias;
- Câmera de vídeo colorida com pan e tilt de alta resolução;

Figura 2.4: Foto do ROV LUMA extraída de (GOULART, Junho 2007)

- Câmera de vídeo preto e branco de alta sensibilidade;
- Sonar tipo *profilling*;
- Duas luminárias de leds de alta intensidade;
- Cabo umbilical de 1km de fibra ótica;
- Alimentação embarcada usando baterias de $72V$ e $40Ah$;
- Quatro propulsores;
- Doze flutuadores;
- Lastro de chumbo;
- Eletrônica embarcada;

O frame é feito de cantoneiras de alumínio, perfil "L", aliando resistência mecânica e leveza. Na parte central do LUMA estão os propulsores, câmeras e luminárias. Na parte superior estão os 12 flutuadores distribuídos uniformemente nas extremidades com um vão central, onde a água pode-se escoar livremente. O posicionamento do propulsor vertical na parte superior eleva o ponto de flutuação. Na parte inferior estão os componentes mais pesados, tais como: vaso da eletrônica embarcada, vaso das baterias e os lastros de chumbo. Essa disposição desloca o centro de gravidade e cria uma maior estabilidade de navegação (GOULART, Junho 2007).

A arquitetura é parcialmente modular e distribuída, utilizando microcontroladores AVR de 8 bits da *Atmel*. Cada módulo contem dispositivos e funções próprias,

ou seja, se o operador desejar o desligamento de um dos motores, apenas o os respectivos módulos responsáveis pela tarefa operam.

2.9 Comentários finais

Neste capítulo foi introduzida uma pequena revisão bibliográfica sobre as principais características e os principais componentes de um AUVs. No primeiro momento foi feita uma releitura da evolução da ferramenta ao longo dos anos até a consolidação dos modelos atuais. Alguns pontos como: sistemas de comunicação, características estruturais, opções de fontes de energia, propulsão etc, foram objetivamente abordados a fim de preencher a lacuna teórica e detalhar um pouco mais sobre universo do veículo em questão. Nos sub-capítulos finais são apresentados alguns modelos de veículos disponíveis atualmente no mercado focando as particularidades de cada um deles, dentre os citados, está o LUMA, cujo o desenvolvimento da tese é baseada em seu modelo virtual.

-

Capítulo 3

Sistemas de Controle de Missão

3.1 Introdução

Todo robô é desenvolvido para realizar uma ou mais tarefas. Para qual tipo de missão o veículo será desenvolvido? Qual é o conjunto de ações que serão tomadas durante as missões? Quem são os usuários do robô? Quais são os dados necessários coletados durante as missões? Perguntas como estas ajudam a estabelecer o grau de destreza que o dispositivo terá que apresentar. Para (TURNER, 1995), um dos maiores desafios dos AUVs é determinar o nível de "inteligência" que o veículo deverá possuir para desempenhar com sucesso as missões programadas.

O Sistema de Controle de Missão (SCM) é um subsistema dentro da arquitetura de controle localizado no nível superior, que define a sequência e a forma que as instruções serão executadas com o objetivo de concluir a missão programada. Esse capítulo realiza uma revisão bibliográfica das arquiteturas de controle, apontando suas principais evoluções estruturais nas últimas décadas. Posteriormente é apresentado os tipos de SCM predominantes na área da robótica submarina, nos segmentos militar, industrial e acadêmico.

3.2 Arquitetura de controle

A arquitetura de controle é baseada em paradigmas. Segundo (AUGUSTO, 2007), em Inteligência Artificial - *Artificial Intelligence* (AI) -, paradigma representa uma filosofia ou conjunto de situações e técnicas onde se buscam soluções para a abordagem de um problema. Não existe uma solução única para cada tipo de problema e sim, melhores ou piores soluções para o mesmo problema.

Os primeiros veículos autônomos foram concebidos em meados do século passado, após o surgimento da AI. Nesse momento, o paradigma utilizado, denominado de *Deliberativo*, era caracterizado pela integração das informações oriundas dos senso-

res, direcionando-as à unidade central que permitia reproduzir o ambiente seguindo a metodologia *Sentir, Planejar e Agir* (SPA). Esse paradigma funciona de forma sequencial e organizado. Primeiro o robô "sente o mundo" e constrói o mapa global do ambiente; depois planeja todas os caminhos possíveis para alcançar o objetivo; e por último, o robô age. Esse evento é cíclico e termina apenas quando o objetivo da missão for alcançado (MURPHY, 2000). Esse modelo pode ser visto na Figura 3.1.

A flexibilidade de programar diferentes missões dentro da mesma aplicação, rendeu bons projetos (FIKES; NHSSON, 1971). Pelos resultados alcançados, muitos outros sistemas adotaram esse mesmo paradigma, porém perceberam-se algumas falhas estruturais que comprometiam as missões. A grande dificuldade era contornar situações onde o algoritmo encontrava soluções não triviais, tornando o modelo do ambiente deficiente, impossibilitando o planejamento da missão. A abordagem *Deliberativa* era limitada a ambientes previsíveis e em ambientes dinâmicos sofria sérios problemas.

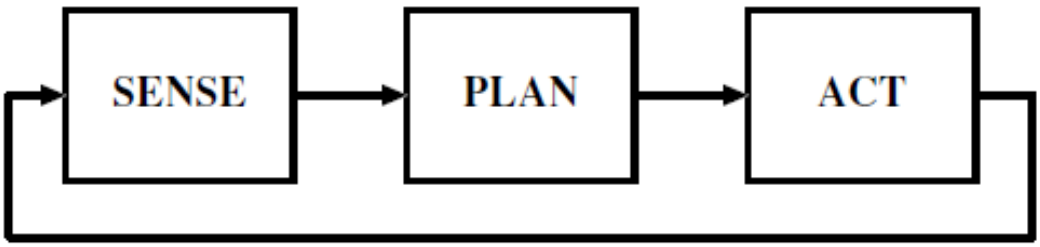


Figura 3.1: Esquema SPA retirado do livro (MURPHY, 2000)

As críticas e as falhas apresentadas na modelagem de ambientes dinâmicos da abordagem *Deliberativa*, levaram um grupo de cientistas nos anos 80 a repensar na concepção de paradigma em AI. Os pesquisadores questionavam a utilização do ambiente simbólico, predominante até o momento, e propuseram desenvolver uma arquitetura baseada na abordagem reativa (MURPHY, 2000). Para isto, uma importante ligação entre a percepção do ambiente (sensores) e as ações (atuadores) foi aprimorada. Uma rede de computadores foi criada para conectar os sensores de forma distribuída, eliminando a central de controle. O ambiente real era a todo o momento atualizado, permitindo assim, o uso desse paradigma em ambientes dinâmicos.

A consolidação desse modelo foi idealizada por (BROOKS, 1986), que propôs uma arquitetura de *Subsunção*, baseada em um controle robusto em camadas. O novo paradigma, conhecido como *Paradigma Reativo* baseava-se no comportamento animal, tendo a "inteligência" classificada em níveis.

Depois de uma análise prévia, o robô cria um modelo agrupando um conjunto de comportamentos, que faz o sistema reagir da melhor forma para a conclusão da

missão. Novas camadas geradas podem reutilizar as camadas mais baixas, inibi-las ou até mesmo traças caminhos alternativos, como mostra a Figura 3.2. O paradigma *Deliberativo*, baseado na rotina cíclica, lenta e tediosa do SPA deu lugar ao paradigma rápido e reativo. Neste sentido, não houve problemas com o modelamento de ambiente em tempo real, pois o sistema de controle era capaz de perceber as mudanças de cenário. O sucesso alcançado por esse paradigma permitiu algumas aplicações tais como, por exemplo, o controle preciso em aplicações especiais de servo-motores (CONNELL, 1992).

Apesar do sucesso, o modelo em camadas sugerido pelo *Paradigma Reativo*, tinha limitações. Missões de longa duração e dificuldades de adaptação aos robôs emergentes na época, trouxeram novos questionamentos. Além disso, a dificuldade de criar uma hierarquia de prioridades, evitando que diversos comportamentos fossem ativados simultaneamente, dificultava a sincronização.

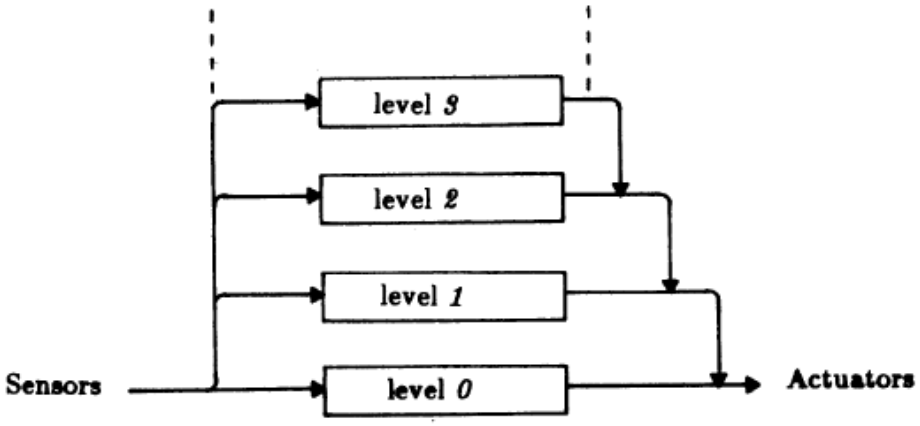


Figura 3.2: Controle em camadas composto em vários níveis e cada uma responsável por uma diferente atividade.(BROOKS, 1986)

A solução que poderia ser adotada com sucesso, integrava planejamento (herdado do paradigma *Deliberativo*), com o comportamento reativo (herdado do paradigma *Reativo*) ou seja, a união das melhores características dos paradigmas existentes na época. Surgiu nesse período o modelo de *Arquitetura-Híbrida* que ainda é a configuração de arquitetura mais bem sucedido nas aplicações da robótica móvel. Atualmente, o controle deste paradigma está distribuído em três camadas, como mostrado na Figura 3.3:

- A *Camada Reativa* é uma estrutura de baixo nível, responsável pela interação com o meio em tempo real, onde está localizado os elementos primários do sistema, tais como: sensores e atuadores. Em determinados momentos, há

uma comunicação envolvendo outras camadas de controle para atualização de alguns dados durante a missão.

- A camada intermediária de *Execução e Controle* supervisiona as tarefas. Constantemente, ela interage com as demais, atuando entre os níveis superior e inferior (camada de planejamento e a camada reativa respectivamente). A camada de *Execução e Controle* estabelece uma comunicação para que o planejamento desenvolvido em camadas superiores alcance os níveis inferiores, habilitando/desabilitando atuadores e sensores durante a trajetória.

- Na parte superior da arquitetura, encontra-se a camada *Deliberativa*, responsável por transformar o objetivo real da missão em uma sequência de ações. O grau de deliberação e reação encontrado nas arquiteturas pode variar de aplicação para aplicação.

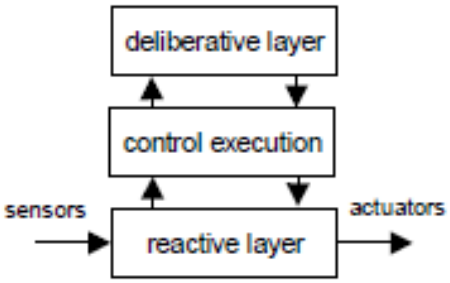


Figura 3.3: Modelo híbrido de arquitetura

(FIRBY, 1989) foi um dos primeiros pesquisadores a propor essa arquitetura, integrando os modelos deliberativo e reativo em três camadas. Pouco depois, (ARKIN; BALCH, 1997) desenvolveu o *Autonomous Robot Architecture* (AuRA). Embora muitas aplicações na área de robótica móvel tenham adotados este modelo em três camadas de controle, poucos sistemas possuem capacidade deliberativa plena. A maioria dos casos as missões sofrem a interferência do homem, não podendo ser geradas automaticamente por um hardware embarcado.

3.2.1 Arquitetura deliberativa

A característica principal desse sistema é a utilização das informações armazenadas do modelo de mundo para a construção do conhecimento prévio do ambiente, através das informações sensoriais recebidas do robô. Um dispositivo que emprega as técnicas deliberativas exige um amplo conhecimento do mundo e usa estas informações para prever o resultado de todas as suas ações. Essa habilidade permite

otimizar seu desempenho em função do modelo armazenado em sua memória. Portanto, a arquitetura deliberativa precisa de informações detalhadas, sobretudo que sejam coerentes, confiáveis e seguras. Se as informações são incompletas, confusas ou dinâmicas (onde os objetos podem se mover), a resposta do sistema se torna inconsistente. Um corredor lotado ou um campo de batalha, por exemplo, é potencialmente perigoso invocar o passado já que as informações podem não ser mais válidas (ARKIN; BALCH, 1997).

As formas mais usadas para o robô representar o mundo são: a representação simbólica e a representação geométrica. O simbólico é baseado em lógica, utilizando geralmente, a inteligência artificial. No geométrico o ambiente é representado espacialmente por espaços livres e regiões de obstáculos, permitindo que o robô execute a atividade de mover-se de forma autônoma de uma região com espaços livres e sem colidir com os obstáculos. Por esse motivo o modelo simbólico é o mais utilizado (JUNIOR, 2006).

Existem outras formas de representação do modelo de mundo, porém o que importa é definir o objetivo da tarefa executada pelo robô. Uma vez estabelecido o objetivo, o robô deve ser capaz de planejar para que suas ações o levem ao cumprimento de sua tarefa de forma eficiente e ótima. As arquiteturas deliberativas são melhores empregadas a ambientes estáticos e muito bem controlados, pela dependência do mundo interno. Há casos em que as mudanças do ambiente (que devem ser percebidas pelo robô durante sua ação) impede a realização da tarefa. Assim o modelo interno tem que ser atualizado, e portanto, o planejamento tem que ser refeito.

3.2.2 Arquitetura reativa

A arquitetura reativa é composta por uma série de comportamentos que ligam os dados sensoriais diretamente às ações comportamentais do robô, possibilitando uma correta coordenação desses comportamentos (BROOKS, 1989). Diferentemente da arquitetura deliberativa, esse modelo evita utilizar uma representação de mundo interno, permitindo uma resposta rápida do sistema à estímulos e transformações externas (mudança do modelo de mundo). Essa característica torna essa arquitetura ideal para aplicações em ambientes dinâmicos. Esta rapidez se deve a simplicidade no tratamento das informações sensoriais e a forma direta pela qual a percepção, ou estímulo, estão associados às ações ou respostas (JUNIOR, 2006).

A coordenação das ações em uma arquitetura reativa é feita de modo competitivo ou cooperativo. Na coordenação competitiva o comportamento ativo prevalece dentro de uma hierarquia ou arbitrariedade, determinando a ação que o robô irá realizar. Ja na coordenação cooperativa todos os comportamentos ativos contribuem

com a ação do robô. Segue os principais tipos de coordenação para a arquitetura reativa.

Arquitetura de subsunção

Esta abordagem, Subsunção - *Subsumption* - (BROOKS, 1986), é um tipo de arquitetura reativa, organizada em camadas através de uma hierarquização de comportamentos. Os níveis superiores correspondem à execução dos objetivos da tarefa, já os níveis inferiores são responsáveis pelas funcionalidades básicas da tarefa, como as que garantem a integridade e a sobrevivência do robô. Os níveis mais altos suprime o fluxo de dados das camadas inferiores. As camadas são independentes e decidem quando e como agir, sem o uso de sub-rotinas de outras camadas (JUNIOR, 2006).

Arquitetura esquema motor

O modelo citado por (ARBIB, 1992), descreve os comportamentos como módulos que se relacionam entre o controle motor e a percepção. Esses módulos agem paralelamente ou concorrentemente havendo uma cooperação mútua para determinar a resposta final do sistema. As respostas motora aos estímulos de cada comportamento são apresentadas sob forma vetorial gerada a partir de métodos de campos potencias artificiais. A soma do vetor desses elementos resulta na ação executada. Desta forma não há uma decisão de comportamento prevalecendo sobre os demais, pois todos contribuem com a parcela sobre a resposta final do sistema (JUNIOR, 2006).

Arquitetura de circuito

Essa arquitetura é uma hibridização das melhores características da reatividade (como a tipificação pela arquitetura de subsunção), utilizando as abstrações e formalismos lógicos de (ARKIN; BALCH, 1997). Criada por (ROSENSCHEIN S., 1986), suas reações são fruto dos comportamentos mais primitivos, que consequentemente formam outros comportamentos. Dessa forma há uma grande abstração e a formação de grupos mais complexos de ações comportamentais.

3.2.3 Arquitetura híbrida

Esse é o modelo predominante de arquitetura que utiliza as técnicas deliberativa para planejar as ações do modelo de mundo, de forma que os objetivos do robô possam ser alcançados. Quando as ações já foram planejadas, a execução fica por conta das camadas inferiores, as reativas, que responde em alta velocidade as variações da dinâmica do ambiente. Portanto, esta arquitetura se torna apropriada para soluções

em situações mais complexos, atingindo objetivo de forma eficiente, através do uso da deliberação aliada as respostas dinâmicas dos sistemas reativos. A arquitetura híbrida é responsável por definir a função da parte deliberativa e reativa da unidade de controle. Também é de responsabilidade da arquitetura coordenar a interface entre deliberação e reação dentro do sistema (JUNIOR, 2006).

Entre esses dois extremos, podem existir arquiteturas com diferentes níveis de hibridização. Entretanto, segundo (MURPHY, 2000), podem-se classificar os vários métodos existentes do paradigma híbrido em três categorias :

- Arquiteturas Gerencias: tem como característica a divisão da porção deliberativa em subcamadas. Na parte superior, ficam as camadas de planejamento de alto nível que repassa o plano às camadas de mais baixo nível. O planejamento é refinado e entregue às camadas reativas da arquitetura. Há uma supervisão das camadas de nível mais alto sobre as camadas de níveis mais baixo, podendo passar instruções e, eventualmente, modificar as camadas imediatamente inferiores a elas.

- Arquitetura de Hierarquia de Estado: gerencia as atividades reativas e deliberativas de acordo com o conhecimento dos estados do robô e também através do tempo (passado, presente, futuro). Os comportamentos reativos são efetivos somente em relação ao tempo presente, isto é, estado atual do robô. Já as atividades deliberativas são divididas entre as que possuem o conhecimento dos estados passados do robô e as que precisam de conhecimento sobre os estados futuros (missão e planejamento de rota). Portanto, essa estrutura de hierarquia híbrida possui três camadas, sendo que as superiores gerenciam as diretamente inferiores a elas.

- Arquiteturas Orientadas a Modelos: os comportamentos podem acessar porções de um modelo de mundo global, além de suas próprias percepções de mundo específico. Esse modelo de mundo global funciona como um sensor virtual, provendo percepções de alguns comportamentos. Esse modelo se assemelha ao comportamento monolítico de um mundo global das arquiteturas hierárquicas puras.

3.2.4 Controle baseado em comportamento

Os robôs móveis, em sua maioria, são programados para desempenhar as mais complexas missões em ambientes pouco conhecidos. Essa situação cria uma série de dificuldades para o desenvolvimento de um único sistema de controle. Portanto, alguns trabalhos focam em fragmentar uma tarefa complexa (global), em múltiplas

tarefas mais simples (sub-tarefas), permitindo a criação de sistemas de controle independentes, caracterizando assim, uma navegação descentralizada (PIRJANIAN, 1999).

O controle baseado em comportamento é uma técnica inspirada em modelos da natureza, que busca um padrão de arquitetura a partir do cérebro dos animais. Problemas complexos, tando de planejamento quanto de execução, se transformam em um conjunto de módulos de interação (comportamentos), que coletivamente atingem um nível complexo de planejamento. De acordo com um observador externo, o comportamento pode ser definido como padrões de atividade resultante das interações entre o robô e o meio. Para o programador, ele também pode ser definido como módulos de controle que descreve um conjunto de ferramentas para alcançar o objetivo (SANTéRIO, 2010).

Os comportamentos são executados em paralelos, havendo uma iteração direta entre os sensores e atuadores com todas as camadas. Essa flexibilidade possibilita que novas camadas sejam adicionadas ao sistema sem a necessidade de configurá-lo. Assim, sistemas complexos e abrangentes podem ser construídos a partir de comportamentos mais simples.

(SANTéRIO, 2010) destaca algumas características desses sistema, estre elas estão:

- Comportamentos são implementados como as leis de controle (por vezes semelhantes às usadas na teoria de controle), seja em software ou hardware, como um elemento de processamento ou como um processo.

- Cada comportamento pode receber entradas de sensores do robô (sensores de proximidade, detectores de ultrassom, sensores de contato, câmera) e de outros módulos. As saídas são enviadas para os atuadores do robô (rodas, garras, manipuladores) e outros módulos.

- Comportamentos diferentes podem receber contribuições de forma independente do mesmo sensor e comandar ações de saída para o mesmo atuador.

- Os comportamentos são codificados para serem de relativa simplicidade, e são adicionados ao sistema de forma incremental.

- Comportamentos (ou seus sub-grupos) são executados simultaneamente e não sequencialmente, a fim de explorar o paralelismo e velocidade de computação, bem como a dinâmica de interação entre os comportamentos e entre os comportamentos e o ambiente.

O controle baseado em comportamento tem que ser capacitado em determinar a melhor opção de ação ou comportamento dentre várias possibilidades. Há métodos

que permitem esta escolha, por exemplo, a abordagem de hierarquia de comportamento, cujo o comando em alto nível são ativos e enviadas a outros níveis inferiores (SANTéRIO, 2010). Outras abordagens e técnicas de controle em camadas podem ser vistas e estudadas em (PIRJANIAN, 1999).

Classificação das arquiteturas baseadas em comportamento

Vários tipos de arquiteturas de navegação vêm sendo desenvolvidas nos últimos anos em busca de progressos voltados a autonomia de robôs móveis. Como o aumento da complexibilidade dos sistemas, várias exigências foram agregadas à navegação, para atender diversas especificações, que por si só, demandam um sistema de controle robusto e independente. Uma arquitetura de navegação baseada em comportamentos recebe a classificação qualitativa, de acordo com a seleção dos comportamentos - sendo competitiva (arbitração) ou cooperativa (método fusão de comandos)(PIRJANIAN, 1999) .

Dentre as técnicas de arbitração, um único comportamento é responsável pela reação de todo o sistema. Existem três formas de utilizar esse modelo, são elas: sistemas baseado em prioridades (BROOKS, 1989), *winner-take-all*(MAES, 1989) e sistemas baseados em estados (KOSECKA, 1994).

Já nas técnica fusão de comando, o gerenciamento das informações é feito por um conjunto de controladores. Existem quatro modos distintos para essa aplicação, são elas: sinais de controle por votação (DAMN, 1995), por superposição (ARKIN, 1987), por lógicas fuzzy (SAFFIOTTI; KONOLIGE; RUSPINI, 1995) e multiagentes (PIRJANIAN, 1999).

Outros métodos levam em consideração atribuições estatísticas como filtros de Kalman e filtro de informação(THRUN; BURGARD; FOX, 2005). No fluxograma da da Figura 3.4 pode-se observar o a forma que se classifica a arquitetura baseada em comportamento.

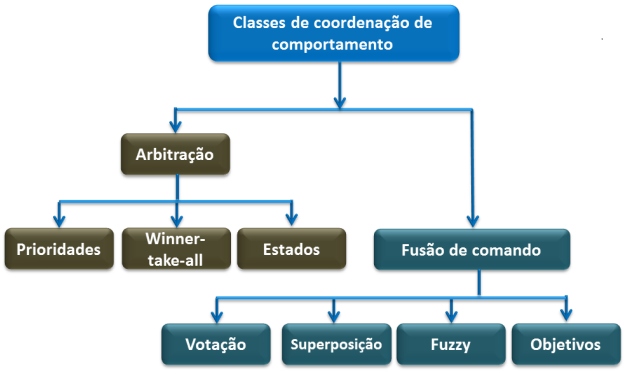


Figura 3.4: Mecanismos de seleção das classes de ações. Figura adaptada de (PIRJANIAN, 1999)

3.3 Revisão de sistemas de controle de missões

Do ponto de vista técnico, um tópico comum a qualquer veículo autônomo é a utilização dos SCM em sua arquitetura. No caso dos AUVs é muito importante delimitar o tipo de missão que o robô irá executar para determinar o critério utilizado. Geralmente, os AUVs são classificados em três grandes categorias: as militares, as científicos e as comerciais.

Embora possuam diferentes finalidades, a similaridade encontrada nas três modalidades é a representada na Figura 3.5. Todos os modelos de arquitetura híbrida, que pode ser empregado com ou sem técnicas deliberativas *on-bord*, têm uma característica em comum: executam um conjunto de instruções primitivas, que descreve um grupo de tarefas simples, cujo o veículo seja capacitado à realizar de forma autônoma.

Por exemplo, *"manter velocidade constante"*, *"nível baixo de carga na bateria"*, *"ir a um ponto"*, *"seguir o duto"* ou *"alcançar uma determinada profundidade"* etc, são modelos recorrentes de ações primitivas usados em projetos de SCMs.

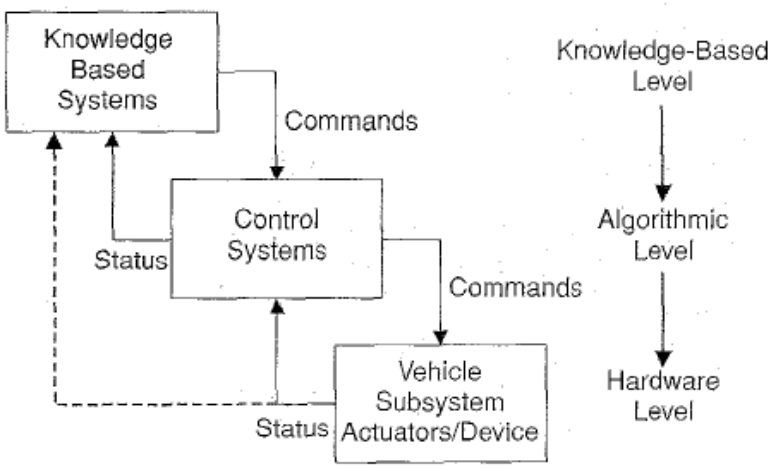


Figura 3.5: Arquitetura de controle de um veículo autônomo (VALAVANIS et al., 1997)

Técnicas de planejamento de missões são divididas em dois grupos: um deles utiliza o planejamento da missão pré-definida pelo usuário e o outro grupo utiliza técnicas de AI para executar planejamento de missões de forma automática. (TURNER, 1995) cita algumas técnicas de planejamento automático de missões. Posteriormente, outros trabalhos também foram elaborados sobre este assunto, que basicamente possibilita o replanejamento da missão de forma automática pelo veículo durante a execução. Geralmente, missões de longa duração ou missões realizadas em ambientes desconhecidos, têm bons resultados a partir do planejamento automático.

Outro grupo de planejamento delega toda a responsabilidade e o sucesso das missões ao programador. Através de um estudo prévio, ele tem que ser capaz de antever

todas as situações possíveis e, para cada uma delas, criar uma solução buscando sempre o cumprimento da missão. Apesar de parecer improvável o detalhamento das missões, aumenta substancialmente a possibilidade de sucesso.

A linguagem específica de domínio - *Domain Specific Language* (DSL) - são linguagens de programação executa os planejamentos das missões. Existem algumas representações, que são usados *scripts* mais básicas, até as estruturas desenvolvidas em *scripts* alto complexibilidade. Esse assunto pode ser visto com mais detalhes em (KAO et al., 1992), (JOHNS; TAYLOR, 2008).

Os DSLs, descrevem as ações primitiva para cumprir uma missão, combinam mecanismos de manipulação de erros, incluindo por exemplo, valores de profundidade máxima ou mínima, atitude permitida ou zonas de segurança etc. Para ilustrar o esquema de uma SCM por DSL, a Figura 3.6 pode ser consultada. O primeiro bloco, *user input* é responsável pela interação do sistema com o usuário. O perfil da missão é dado pelo programador através de uma lista de comportamentos primários ou um conjunto de metas arquivadas na memória do robô, de acordo com a necessidade de cada missão.

Antes de ser iniciado, existe um sistema que verifica todas as possibilidades e alternativas que podem ocorrer durante o planejamento. O bloco de execução interpreta a trajetória, armazenado na memória do veículo e o transforma em uma sequência de comandos e sinais. Se for detectado um erro, esse bloco solicita o replanejamento da missão. Para os sistemas que não possuem recursos *Deliberativo on-board*, todas as ações reativas devem ser incluídas na programação.

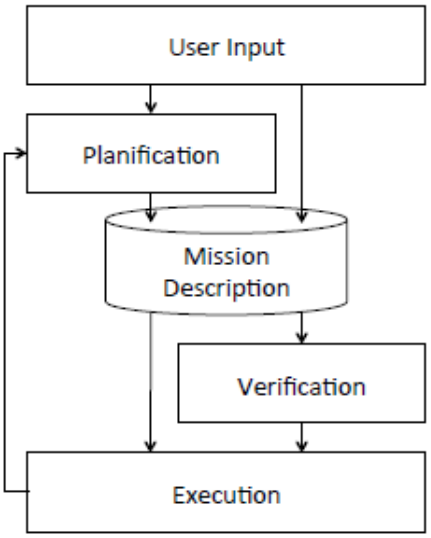


Figura 3.6: Estrutura de um MCS

Finalmente, um caminho para simplificar as análises e diminuir a complexidade das arquiteturas SCMs são os formalismos de linguagens. Entre os mais conhecidos temos as Redes de Petri (BARBIER et al., 2011) e as Maquinas de Estado

(BOUSSINOT; SIMONE, 1991), que na realidade é um caso particular das Redes de Petri.

3.4 Sistemas deliberativos de planejamento de missão

O método *Deliberativo on-board*, tem uma aplicação bem sucedida no projeto *ORCA*. (TURNER, 1995) descreve a estrutura desenvolvida em um AUV de pesquisas oceânicas de missões de longo alcance em ambientes desconhecidos. A ideia desta arquitetura é dividir uma missão em um conjunto de pequenas missões e cada uma com seu respectivo objetivo. Esses pequenos grupos criam uma lista com um conjunto de metas.

O sistema seleciona a tarefa de maior prioridade (dentro desta lista) e aplica um método sistemático, para executar individualmente cada meta. Esse sistema apresenta uma biblioteca de ações primitivas e ações sistemáticas, que se repetem várias vezes até que todas as tarefas da lista estejam concluídas. Outro modelo de AUV, que utiliza a abordagem *Deliberativa* em planejamento de missões, foi apresentado por (CHANG; BIAN; SHI, 2004). É uma técnica mais elaborada que compartilha informações entre o AUV e uma estação de trabalho localizada em algum ponto estratégico na superfície. Toda estrutura é fragmentada em três partes: nível de tarefa, nível de missão e nível de comportamento. Uma estação de gerenciamento elabora a missão, associando as informações da carta náutica eletrônica com as informações do arquivo da missão. O nível de tarefa cria as missões e as sequências de execução. Esse nível é dividido em módulos. O módulo de coordenação sincroniza todos os eventos (falhas, obstáculos, eventos, etc.) com o *clock* principal. Eventualmente, este módulo de coordenação pode solicitar o replanejamento da missão quando o sincronismo não é alcançado, ou até mesmo abortar a missão (executando uma sequência de comandos para levar o AUV a uma posição de segurança). O nível de comportamento estipula uma arbitragem entre as *ações primitivas* do veículo, enviando essas informações para a estação de controle. Todo planejamento da missão e coordenação de tarefas é formalizada em Redes de Petri.

Umas das implementações mais complexas das técnicas *Deliberativa on-board* em SCM foi consolidada no *Monterey Bay Aquarium Research Institute* em parceria com a *National Aeronautics and Space Administration* (NASA)'s combinando planejamento e complexas técnicas de programação em SCM (RAJAN et al., 2007). Trata-se do *Teleo-Reactive EXecutive* (T-REX) que é um conceito elaborado para integrar *ações primitivas* de robôs às tarefas de níveis mais elevados. A técnica adotada é baseada no paradigma de planejamento de restrições temporais, onde o

aspecto tempo é a chave da representação dos estados do passado, presente e futuro. Desta forma, o paradigma SPA é o núcleo de uma malha de controle colocado de forma sistemática, possibilitando atingir um espectro de vários comportamentos reativos e deliberativos. Tudo isso é possível no T-REX, com a programação em alto nível, garantindo robustez e segurança para o comportamento do robô.

3.5 Sistemas de missões pré-definidas

A maioria das aplicações envolvendo AUVs utiliza a configuração de missões pré-definidas para os SCMs no lugar das técnicas *Deliberativas*. Desta forma, o usuário é responsável pelo sucesso das missões. O programador tem que ser extremamente detalhista, ao ponto de prever todos os passos da atividade, criando alternativas para solucionar cada uma das possíveis situações encontradas durante a execução da missão. Com esse nível de detalhamento, as configurações pré-definidas alcançam um grande sucesso em suas respectivas aplicações.

Existem dois métodos para abordar as missões pré-definida em SCMs: o primeiro retrata os problemas através dos *scripts* ou DSLs, e o segundo retrata o sistema baseando em formalismos. Ambos têm suas particularidades, possuindo suas vantagens ou desvantagens de acordo com as aplicações. A seguir serão apresentados alguns detalhes desses dois métodos.

3.5.1 Linguagens por *Scripts*

A maioria dos sistemas empregados em AUVs usam as linguagens por *scripts* ou DSLs. As aplicações são divididas em três grandes áreas de atuação: os sistemas empregados nos centros de pesquisa, aplicações para fins comerciais e as soluções genéricas, que abrangem boa parte do mercado.

Aplicações na Pesquisa

A *Naval Pos-graduate School* (NPS) desenvolveu um sistema de controle híbrido, distribuído em três níveis: o nível estratégico, o nível execução e o nível tático. Uma arquitetura de controle contendo uma linguagem denominada "PROLOG" citada em (MARCO; HEALEY; MCGHEE, 1996), caracterizada por regras pré-definidas, por gerenciamento de transições de estados, incluindo procedimentos de recuperação de erros e falhas na execução de missões.

Um dos primeiros SCMs em DSLs foi elaborado pelo *International Submarine Engineering* (ISE), chamado de PROMETEUS (KAO et al., 1992). Uma arquitetura de subsunção (BROOKS, 1989) desenvolvida por *script* conecta os sentidos de percepção à ação do robô. Nesta arquitetura, o comportamento é estimulado pelas

informações sensoriais através de várias malhas interligadas a diferentes atividades. Nas camadas superiores, os comportamentos são direcionados ao cumprimento da missão e nas camadas mais baixas eles são ligados às ações. As duas camadas funcionam de forma independente, mas respeitam uma hierarquia de inibição e supressão. Em ambos os modos, sempre a prioridade será das camadas superiores.

Aplicações Comerciais

Em paralelo com os estudos dos grandes centros de pesquisa, empresas comerciais começaram a desenvolver e a disponibilizar no mercado soluções prontas de AUVs para a realização de trabalhos, principalmente para os estudos do leito marinhos. A companhia *Hydroid* comercializa o *Remote Environmental Monitoring Units* (Remus) (ALLEN et al., 1997), inicialmente desenvolvido pela *Woods Hole Oceanographic Institution* (WHOI).

O Remus é programado em ASCII e contém arquivos de missões formando uma lista de objetivos que são cumpridos em sequência. Uma missão é concluída quando toda a lista for finalizada. A grande vantagem desse AUV é a simplicidade do *script* de controle que possui uma flexibilidade no algoritmo e é facilmente adaptado à outros tipos de missões.

Outro AUV de uso comercial que utiliza recursos de missões pré-definidas é o *GAVIA AUV System* .O sistema possui uma arquitetura de software mais elaborada, chamada de *Multi Agent System* (MAS) e é descrito de forma bastante detalhada em (GAVIA, 2014).

Aplicações Genéricas

As aplicações genéricas definem um campo de missões que podem ser adaptadas para quaisquer áreas de atuação, onde ainda não há uma linha de dedicação exclusiva de pesquisa.

Algumas soluções são padronizadas pela linguagem *urbiscript* que tem sido desenvolvido desde 2003 no laboratório *Cognitive Robotics Lab of ENSTA* em Paris. Esse componente é utilizado para agrupar funções escritas em C++ com elementos de comportamento funcional. Esse modelo cria vantagens nas manutenções e nas iterações. O *urbiscript* apresenta recursos para programação em eventos, como parte da semântica de linguagem.

Outra aplicação genérica por *script* é a *Task Description Language* (TDL) (SIMMONS; APFELBAUM, 1998), que também utiliza a linguagem de programação C++ e inclui procedimentos assíncronos (tarefas). O modelo compila missões em um arquivo C++, executado no veículo por uma biblioteca de objetos independentes. A TDL garante certa simplicidade na linguagem, devido à complexibilidade

apresentada neste tipo de programação.

3.5.2 Formalismo de Linguagens

O formalismo de linguagens mostra como o *Discrete Event Controller* (DEC) é controlado. Ele é responsável por ativar/desativar estados e verificar o cumprimento de *ações primitivas* e transições de eventos do veículo. Sequências primitivas, comportamentos do veículo e fluxo de dados são representados em um único sistema, tornando possíveis análises das propriedades, tais como: alcançabilidade, vivacidade e limitada.

Descrição Formal da Missão

A solução desenvolvida por (KENCONLEY, 2014) é constituída por uma arquitetura em máquinas de estado. O *State MACHine* (SMACH), utiliza uma biblioteca em *Phyton*, executando planos complexos, onde todos os possíveis estados e transições podem ser explicitamente descritos. *SMACH* está integrado ao *Robot Operation System* (ROS), por meio de um interface chamada (actionlibs).

Outro sistema muito popular, o Esterel (BOUSSINOT; SIMONE, 1991), vem evoluindo e se desenvolvendo por mais de 20 anos, com a colaboração da *École des Mines de Paris* e *Institut National de Recherche en Informatique et en Automatique* (INRIA). Através de uma programação síncrona destinada à sistemas reativos complexos, é possível modelar missões e estruturas de controle do veículo. Apesar de ótima eficiência e facilidade, este sistema é restrito para determinados tipos de missão.

No LSTS/FEUP (SILVA; MARTINS; PEREIRA, 1999) o TEJA CASE (DESHPANDE; SOUSA, 1997) é um aplicativo que define graficamente a dinâmica de comportamento utilizando o paradigma de máquina de estado. A partir do modelo criado graficamente da aplicação, todo o código fonte é gerado em C++, sendo o projeto modelado em três camadas. Existe a camada de abstração (responsável pelo acesso às informações de baixo nível do robô, tais como: motores, sensores e atuadores), existe a camada funcional (constituída por algoritmos de ações primitivas de navegação) e existe a camada de coordenação (gerencia o comportamento da missão).

Missão Formal em Rede de Petri - aplicações para AUVs

O *Institute for Systems and Robotics* (ISR) Lisboa, Portugal, desenvolveu uma arquitetura de controle para o AUV Marius (OLIVEIRA et al., 1998), cujo o fluxo de controle de informações das ações primitivas são coordenados pelo software chamado CORAL. A Rede de Petri é o formalismo para descrever o sistema (vide Figura 3.7).

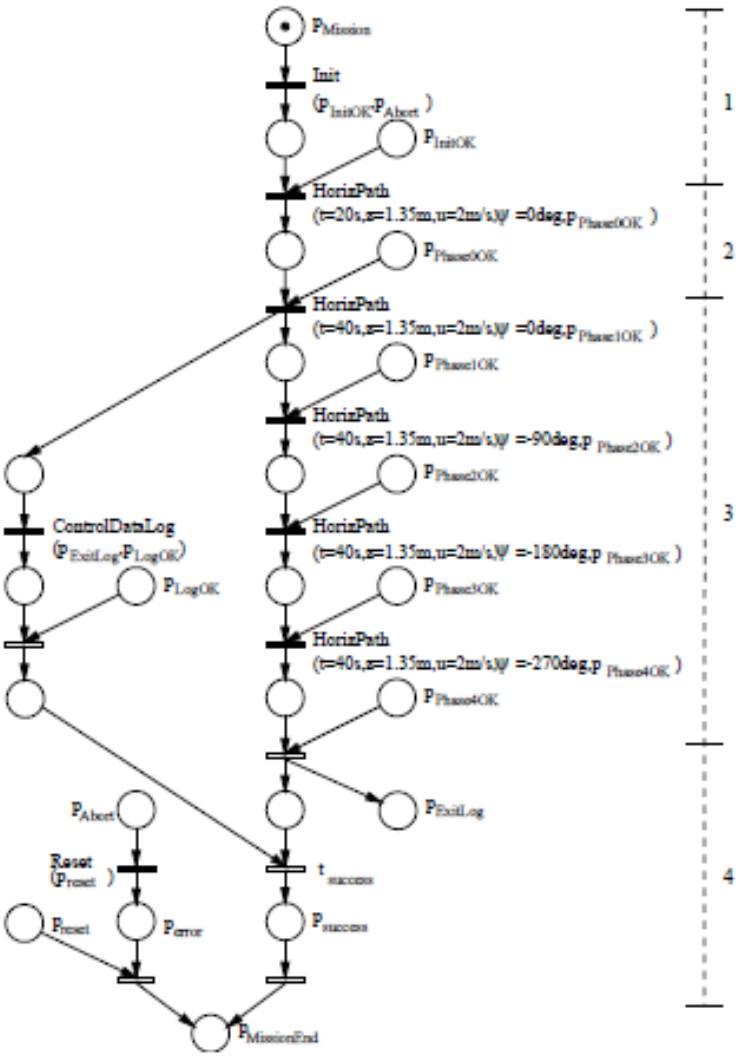


Figura 3.7: Estrutura de um SCM (OLIVEIRA et al., 1996)

Na Itália existe outra similar aplicação dos - AUV Romeo - baseada nas Redes de Petri (CACCIA et al., 2013). Desenvolvido para atender os projetos de robôs submarinos, a arquitetura é baseada nos conceitos de níveis de controle de execução originalmente introduzidos por (ALAMI et al., 1998). A camada deliberativa utiliza as Rede de Petri para acompanhar os planos de execução. Quando algum comportamento da tarefa é habilitado pelos sensores ou atuadores, um demultiplexador - baseado em redes de Petri -, estabelece os melhores comportamentos (respostas) aos estímulos recebidas pelos sensores e atuadores do sistema.

Existem soluções híbridas que combinam o uso da linguagem formal - rede de Petri - com os *scripts*, aliadas às funções de manipulação de erros. Esse é o caso do AUV de uso militar francês *Redermor* (BARROUIL; LEMAIRE, 2005) e (BARBIER et al., 2011), que identifica e recupera minas submarinas. O *Redermor* tem uma programação e um sistema de execução chamado ProCoSA. Esse sistema é composto por uma interface gráfica, para a construção das redes de Petri (lugares são representados por estados comportamentais e as transições são as ações eventos e mensagens). As ações descritas em redes de Petri são implementadas no AUV por sub-sistemas independentes. Para especificar a missão, o usuário tem que configurar uma série de parâmetros, tais como: regiões proibidas, limitas de profundidade, restrições etc. Esses dados são otimizados e a melhor rota de navegação é escolhida por um complexo sistema de planejamento em alto nível. Finalmente, o controlador da missão supervisiona as etapas e monitora a execução de cada comportamento.

E finalmente, o departamento de robótica da University of Girona - Espanha - desenvolveu sistema de controle de missões em aplicações genéricas para AUVs, baseados em rede de Petri (PALOMERAS et al., 2009). O conceito das Petri Nets Building Block (PNBBs) descreve e executa as missões facilmente implementadas nas camadas deliberativas. A flexibilidade é a grande vantagem do sistema tornando-se possível a adaptação a diferentes configurações de arquiteturas de controle. Para descrever a missão, ao invés de manipular diretamente as redes de Petri utilizando ferramentas gráficas, um sistema em alto nível e de linguagem imperativa, chamado *Mission Control Language* (MCL) é utilizado (PALOMERAS et al., 2012).

3.6 Comentários finais

A robótica é uma área de percepção e manipulação de informações do mundo real. A interpretação dos dados pelos instrumentos bem como o movimento executado pelo veículo autônomo, sofrem interferências de diversas fontes, dentre elas pode-se citar:

- *Ambiente*: o mundo físico é inerentemente imprevisível. O nível de incerteza

em ambientes estruturados é baixo, por outro lado, em ambientes desestruturados e dinâmicos a incerteza é altíssima.

- *Sensores*: resolução, precisão e exatidão são propriedades comuns à todos instrumentos de navegação e essas características são cercadas de incertezas.
- *Modelos*: os modelos de mapeamento e navegação são imprevisíveis por natureza, pois são uma abstração do mundo real.
- *Robôs*: os deslocamentos do robô são realizados por motores e atuadores que depreendem de um certo grau de incerteza.
- *Computação*: os robôs são sistema em tempo real que têm as informações distribuídas através de várias centrais de processamento. A discretização dos sinais e a utilização de algorítimos simplificados já são suficientes para introduzir incertezas ao sistema.

No passado foi comum ignorar a maioria dessas incertezas nos modelos computacionais, mas com a evolução da robótica e o desafio de criar tarefas em ambientas cada vez mais desestruturados, a utilização da incerteza dentro da modelagem dos veículos autônomos está sendo um fator fundamental (THRUN; BURGARD; FOX, 2005).

O mundo real é não-determinístico, o que dificulta a elaboração dos planos de missão e trajetória. A arquitetura baseada em comportamentos facilita a implementação da autonomia em ambientes desestruturados. Essa arquitetura divide a missão em sub-tarefas, cria controles independentes e descentraliza a navegação. A manipulação das informações probabilísticas pode ser feita através da fusão de comando. Dentro deste método, a lógica Fuzzy (ao contrário da lógica booleana, admite valores intermediários entre 0 e 1) reproduz bem conceitos estatísticos, tornando uma boa ferramenta para modelar ambientes não-determinísticos.

Outra alternativa foi desenvolvida por (GHALLAB; NAU; TRAVERS, 2004) para facilitar a modelagem de ambientes desestruturados. Essa proposta baseia-se na superposição restritiva, que possibilita implementar técnicas de planejamento em modelos reais a partir de formas simplificadas. Com isso, o mundo real se transforma em uma modelo determinístico e segue as propriedades:

- O sistema tem um conjunto finito e totalmente observável de estados;
- A programação toda é feita em ações restritivas;
- O plano de soluções tem uma sequência ordinária, linear, finita e atemporal (não depende do tempo)

Algorítimos heurísticos são utilizadas para descrever o planejamento utilizado na superposição. É assim nos projetos dos AUVs (PALOMERAS et al., 2009), (OLIVEIRA et al., 1996) cujo sistemas de alto nível mantém a representação simplificada do mundo real, sendo toda modelada em um banco de dados (HERMAN et al., 1988). Isso justifica utilização dos sistemas à eventos discretos para modelar ambientes não-determinísticos. A grande vantagem dos sistemas à eventos são suas propriedades de causalidade e diagnósticos de conflitos.

A proposta desde trabalho é aplicar os conceitos de arquitetura híbrida com elementos prioritariamente deliberativa em um sistemas de controle de missão baseados nos Sistema à Eventos Discretos (SED). Essa aplicação será implementada em um simulador de AUV, que realiza inspeção em dutos submarinos em ambiente virtual onde as informações são determinísticas, ou seja, não apresentam ambiguidades.

No entanto, para melhor compreensão do sistema de controle de missões proposto é necessário apresentar os fundamentos de sistemas à eventos discretos usados no desenvolvimento da arquitetura de navegação, cujo seus conceitos serão vistos no próximo capítulo.

-

Capítulo 4

Teoria das Redes de Petri e Eventos de Sistemas Discretos

4.1 Introdução

Um Sistema a Evento Discreto (SED) é um sistema de estados baseados a eventos, ou seja, a evolução dos estados depende apenas da ocorrência de eventos discretos.

(MURATA, 1989) (CASSANDRAS; LAFORTUNE, 2008), citam que: "um sistema é uma parte limitada do Universo que interage com o mundo externo através das fronteiras que o delimitam". " Os sistemas de interesse percebem as ocorrências no mundo externo através da recepção de estímulos, denominados eventos. ". " A ocorrência de um evento causa, em geral, uma mudança interna no sistema, a qual pode ou não se manifestar a um observador externo. Além disso, uma mudança pode ser causada pela ocorrência de um evento interno ao próprio sistema, tal como o término de uma atividade ou o fim de uma temporização. Em qualquer caso, essas mudanças se caracterizam por serem abruptas e instantâneas: ao perceber um evento, o sistema reage imediatamente, acomodando-se em tempo nulo numa nova situação, onde permanece até que ocorra um novo evento. Desta forma, a simples passagem do tempo não é suficiente para garantir que o sistema evolua; para tanto, é necessário que ocorram eventos, sejam estes internos ou externos. Note ainda que a ocorrência desses eventos pode depender de fatores alheios ao sistema, de modo que este não tem, em geral, como prevê-los".

Neste capítulo será abordado os conceitos e fundamentos que envolvem a teoria dos eventos discretos e também uma pequena introdução sobre os princípios das redes de Petri.

4.2 Sistemas a Eventos Discretos

4.2.1 Eventos e Modelagem de um SED

Eventos são ocorrências que geram transições e mudam imediatamente o estado de um sistema. Segundo a transcrição das normas *IEEE*, que padronizam os termos de dispositivos eletro-eletrônicos, um sistema é a combinação de componentes que agem em conjunto para realizar uma função, cujo seria impossível ser desempenhada por qualquer uma das partes individualmente. Um sistema dinâmico à eventos discretos pode ser modelado para detectar problemas de projetos, decorridos por informações incompletas ou ambíguas.

O modelo é um recurso utilizado que reproduz os estados do sistema, criando condições semelhantes ao sistema de origem. A *modelagem* é definida por um conjunto de variáveis mensuradas a partir de um sistema físico apresentado. Medindo-se essas variáveis por um período de tempo, é possível coletar dados. Selecionando-se um conjunto destas variáveis e supondo que seja possível variá-las no tempo, define-se um conjunto de funções de tempo, chamadas variáveis de entrada (CASSANDRAS; LAFORTUNE, 2008). Entretanto, os dados que podem ser medidos diretamente enquanto as entradas são alteradas, são as variáveis de saída, ou seja, as respostas aos estímulos ocorridos pelas variáveis de entrada.

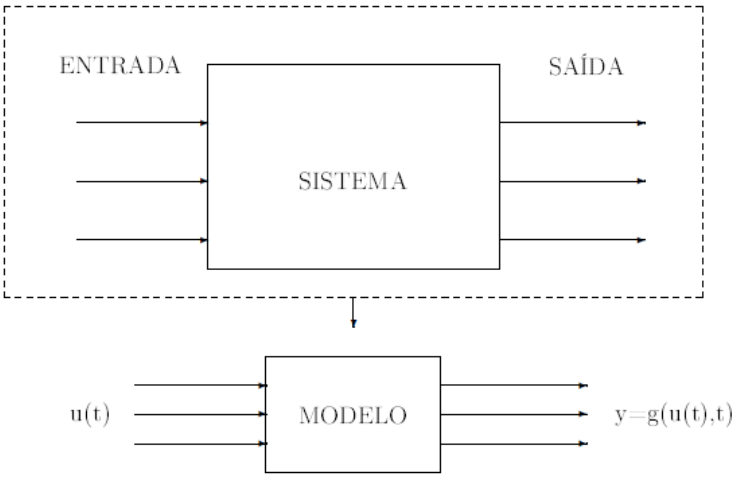


Figura 4.1: Figura representa um modelo dinâmico

Pode-se representar matematicamente, por equações, a relação entre entradas e saídas de um sistema. Sendo $u(t)$, $y(t)$ e $x(t)$, as entradas, as saídas e os estados, respectivamente e o estado inicial $x(t_0)$, a modelagem pode ser definida como um conjunto de equações necessárias para especificar a saída $y(t)$, para todo $t \geq t_0$, conhecendo-se $y(t_0)$ e $u(t)$. A equação diferencial 4.1 é que define um bom número de sistemas encontrados na natureza:

$$x(t) = f(x(t), u(t), t) \tag{4.1}$$

Logo o conjunto de equações que representam o espaço de estado é dado por:

$$x(t) = f(x(t), u(t), t), x(t_0) = x_0 \tag{4.2}$$

$$y(t) = g(x(t), u(t), t) \tag{4.3}$$

A equação 4.2 descreve um conjunto de equações de estado com condições iniciais específicas. Já equação 4.3, forma o conjunto das equações de saída.

A solução geral do sistema é dada por uma função particular, cujo sistema é controlado, selecionando uma entrada correta para o comportamento de saída ser alcançado. Sendo $r(t)$ o sinal de referência, tem-se a entrada de controle é definida pela lei de controle:

$$u(t) = \gamma(r(t), t) \tag{4.4}$$

A realimentação é um conceito que utiliza as informações coletadas na saída do sistema para retroalimentar a entrada de controle. Matematicamente, usar uma realimentação consiste em estender a lei de controle 4.4, para incluir a referência $r(t)$, e a saída observada $y(t)$ ou o estado $x(t)$, na seguinte forma:

$$u(t) = \gamma(r(t), x(t), t) \quad \text{ou} \quad u(t) = \gamma(r(t), y(t), t) \tag{4.5}$$

A etapa da modelagem é o primeiro passo para a compreensão do sistema e a partir dele, cria-se um modelo para representar o sistema físico. Se o resultado for satisfatório, estudam-se diferentes condições com diferentes parâmetros ou funções de entrada. As etapas de projeto e síntese são iniciadas quando se tem um número significativo disponíveis. É necessária uma triagem de parâmetros e valores que leve o comportamento do projeto às condições satisfatórias. Geralmente é preciso refinar e ajustar os esquemas de controle para selecionar alguns parâmetros que satisfaçam o objetivo do desempenho (CASSANDRAS; LAFORTUNE, 2008).

4.2.2 Sistemas controlados pelo tempo e em eventos

O sincronismo das ocorrências, em sistemas controlados pelo tempo, é em função do sinal de "clock". Caso nenhum evento ocorra neste intervalo, significa dizer que o "evento nulo" *ocorreu*, ou seja, um evento que não gera nenhuma transição de estados. Neste caso, o tempo é responsável pela evolução do sistema.

Em outros sistemas, o funcionamento é determinado pela ocorrências de eventos.

Todo o evento é definido pelo processo de transição de estados, sendo a combinação de elementos assíncronos, simultâneos e independentes o tempo.

Pela definição de (CASSANDRAS; LAFORTUNE, 2008), um SED é um sistema de estado discreto, no qual o espaço de estados é um conjunto discreto, cujo o mecanismo de transição de estados é baseado em eventos. A evolução dos estados depende somente da ocorrência de eventos discretos, que acontecem pontualmente no tempo, e assíncronos, que ocorrem em intervalos irregulares, e não importa o que ocorre entre eles.

Sistemas industriais são bons exemplos de sistemas discretos. Em um processo industrial, os clientes são as peças ou parte das peças que estão dispostas para acesso às maquinas (tais como: robôs, correias) que executam as operações. Quando as peças estão sendo trabalhadas, elas são armazenadas em uma fila, até que seja liberado o acesso a próxima operação disponível. Essas filas em uma planta industrial têm capacidade finita devido às limitações físicas da planta.

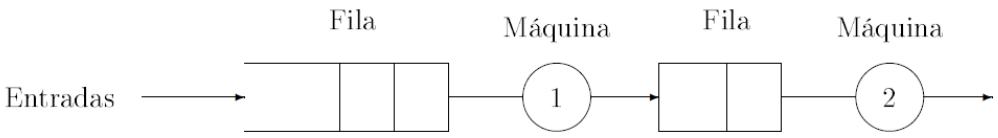


Figura 4.2: Figura representa sistemas de produção em fila

De acordo com a figura 4.2, que representa um processo industrial contendo duas máquinas em produção em série, a capacidade da Maquina 1 é infinita, enquanto a capacidade da Máquina 2, não. Com isso é possível ter situações em que a produção da Máquina 1 seja completada, mas a Maquina 2 ainda esteja ocupada com uma fila de peças à terminar. Assim, uma peça pode estar terminada e permanecer na Maquina 1, embora não requeira mais nenhum serviço. Além do mais, outras as peças são forçadas a esperar na fila da Máquina 1, aguardando na fila de entrada do processo. Portanto, o conjunto de eventos neste caso é $E = \{a, c_1, d_2\}$, sendo a a chegada de peças na Máquina 1, c_1 conclusão de serviço da Máquina 1 e d_2 a partida de peças para a Máquina 2. O estado do sistema é definido pelo vetor $x = [x_1, x_2]$, que corresponde ao comprimento da fila de ambas as Máquinas. Nota-se que x_2 é restrito em $\{0, 1, 2, 3\}$, e quando $x_2 = 3$, a Máquina 1 é bloqueada (acabou o serviço na peça e a fila de espera da Máquina 2 está cheia). O Modelamento deste bloqueio requer uma condição inicial B que x_2 pode indicar que o comprimento da fila é 3 (1 peça processada e 2 peças na espera) e que a Máquina 1 está bloqueada (embora possa ter uma situação onde $x_1 \notin 0$). O espaço de estado deste conjunto discreto é definido por: $X = \{(x_1, x_2) : x_1 \geq 0, x_2 \in \{0, 1, 2, 3, B\}\}$.

4.3 Redes de Petri

Nesta seção será apresentado um formalismo para modelagem dos SEDs, as Redes de Petri - *Petri Nets* (PNs). Desenvolvidas por Carl A. Petri, no início dos anos 60 (MURATA, 1989), elas são ferramentas que manipulam eventos de acordo com regras bem definidas, explicitando quais eventos podem ser habilitados. Possuem recursos matemáticos, do tipo rede, descritos graficamente por grafos que são intuitivos e capturam um conjunto de informações estruturais sobre o sistema. Sua implementação é indicada para estudos de estruturas concorrentes, assíncronos, paralelos, não determinístico e estocástico. Isto permite a representação de vários tipos de sistemas, dos mais simples até os que possuem um alto grau de complexibilidade.

Como linguagem gráfica, as PNs podem ser usadas como redes de comunicação visual (similar aos fluxogramas e aos diagramas de blocos). Uma "*ficha*" é introduzida ao sistema, para simular atividades e as dinâmicas de evolução da rede. Como ferramenta matemática, ela permite a construção de equações algébricas que determinam o comportamento do sistema. Essas duas formas de representação possibilita uma análises em focos distintos, representados pelos modelos teórico e prático respectivamente.

4.4 Transições e conjunto

As regras PNs têm sua atividade baseada nas regras de disparo de suas fichas durante a evolução da rede. Embora apresente uma regra relativamente simples, as considerações por de trás desses conceitos determinam uma teoria profunda e complexa.

As redes de Petri é um modelo particular de grafo que inclui uma condição inicial chamada de μ_0. As *transições - Transitions (T)* - são os eventos ocorrido no sistema. Para que haja uma transição, um conjunto de condições tem que ser satisfeitas. Estados relacionados a esse conjunto de condições são chamados de *lugares - Places (P)*. Locais que sofrem mudanças após uma *transição*, são denominados entradas. Locais que sofrem ocorrências após *transições* e são pré-requisitos para que as *transições* ocorram, são as saídas. Os *Arcos (A)* sempre conectam os elementos P aos T e vice-versa, nunca conectando elementos de mesma natureza, isto é, não podem existir auto laços. Os círculos definem a representação gráfica dos *lugares*. As *transições* são representadas por barras ou caixas. Os *arcos* são indicado pelas setas. As *fichas*, são definidas por uma marcação no dentro dos *lugares*, que sinaliza a o estado desses elementos. Um *lugar* com o número de fichas maior ou igual sua capacidade, é dito habilitado.

Os *lugares* de entrada podem ser representados pelos: dados e sinais de entrada,

recursos e condições necessárias, pré-condições, etc. Os *lugares* de saída podem ser representados por: pós-condições, dados de saída, ações concluídas etc. As *transições* são definidas por: eventos, passo computacional, processamento, tarefas ou trabalhos etc. A Tabela 4.1 relaciona os conjuntos citados acima:

Tabela 4.1: característica das entradas e saídas das redes de Petri

Lugares de Entrada	Transições	Lugares de Saída
Pré-Condições	Eventos	Pós-Condições
Dados de Entrada	Passo Computacional	Dados de Saída
Sinais de Entrada	Sinal do Processador	Sinais de Saída
Recursos Necessários	Tarefa ou Trabalho	Recursos Liberados
Condições	Condições Lógica	Conclusões
Buffers	Processados	Buffers

De acordo com (CASSANDRAS; LAFORTUNE, 2008), o conjunto das redes de Petri $N = (P, T, A, \omega, \mu_0)$ é definido como uma quíntupla onde:

$P = \{p_1, p_2, ..., p_n\}$; um conjunto finito de *lugares* (P),
$T = \{t_1, t_2, ..., t_n\}$; um conjunto finito de Transições (T),
onde, $P \cap T = \oslash \quad$ e $\quad P \cup T \neq \oslash$;
$A \subseteq (PXT) \cup (TXP)$; é o conjunto de arcos (relacionamento de fluxo),
$\omega : F \rightarrow \{1, 2, 3, ...\}$; é o peso de uma função,
$\mu_0 : P \rightarrow \{1, 2, 3, ...\}$; isto é, o número inicial de fichas em cada lugar (P),

Uma RP é definida por: (N, μ_0), onde μ_m é o conjunto de estados marcados da PN;

Comportamento

O comportamento das PNs pode ser descrito pelo fluxo de ocorrências do sistema. Existe uma regra de evolução para a dinâmica de transições, que leva a uma sequência de mudanças de estados da rede, dada por:

1. Uma transição T é dita habilitada se cada respectiva entrada estiver habilitada com no mínimo $\omega(P, T)$, onde $\omega(P, T)$ é o peso do arco de T para P, ou seja, o peso mínimo para o arco ser habilitado;

2. Uma transição habilitada, pode ou não ser disparada (dependerá se o evento acontece ou não);

3. Um disparo de uma transição habilitada T e remove $\omega(P, T)$ fichas de cada lugar de entrada P de T, e adiciona $\omega(P, T)$, $\omega(P, T)$ fichas para cada local de saída P de T, com $\omega(P, T)$ sendo o peso dos arcos de T a P;

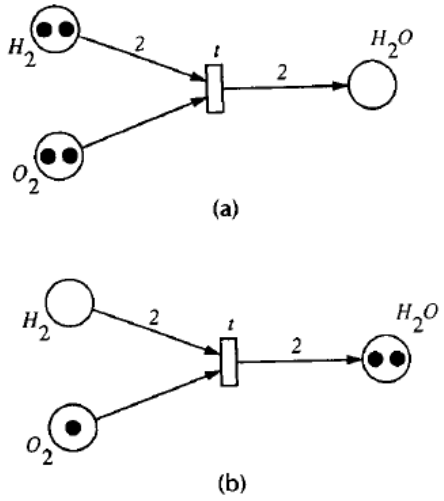


Figura 4.3: Figura retirada de (MURATA, 1989). Ilustração de uma regra de transição: (a) antes da transição T ser habilitada; (b) após transição T ser habilitada.

Como exemplo a figura 4.3, mostra uma regra de evolução da PN.

O modelo do sistema na Figura 4.3, define transições habilitadas assumido que P pode armazenar um número ilimitado de fichas (as PNs são definidas como um sistema de capacidade infinita). Para as redes de capacidade finita é prudente considerar um número superior de fichas ao limite que cada P, sendo a capacidade da rede definida por (N, μ_0), associada a $K(p)$, ou seja, o máximo número de fichas que P pode armazenar.

4.5 Sub-classes das Redes de Petri

As PNs são classificadas em seis diferentes tipos, utilizando símbolos que definem os pré-conjuntos e os pós-conjunto dessas sub-classes. A seguir estão apresentados os símbolos dos prés e pós-conjuntos, onde F é denominado de conjunto dos arcos:

$\bullet t = \{p|\,(p,t) \in F\}$ = um conjunto de entradas de "*lugares*" t,

$t\bullet = \{p|\,(t,p) \in F\}$ = um conjunto de saída de "*lugares*" t,

$\bullet p = \{t|\,(p,t) \in F\}$ = um conjunto de entradas de "transições" p,

$p\bullet = \{t|\,(t,p) \in F\}$ = um conjunto finito de "transições" p,

Os símbolos acima são ilustrados na Figura 4.4, essa notação pode ser aplicada a um subconjunto. Por exemplo, seja $S_1 \subseteq P_1$, então $\bullet S_1$ é uma união de todos dos $\bullet p$ onde $p \in S_1$. Através da notação acima pode-se estabelecer uma relação sobre suas bases estruturais (MURATA, 1989).

As PNs recebem uma classificação de acordo com sua estrutura de evolução. Dentre essas classificações estão: os conflitos, as concorrências e as confusões. Por exemplo, Figura (a) 4.5 mostra uma PN que possui duas transições marcadas, em-

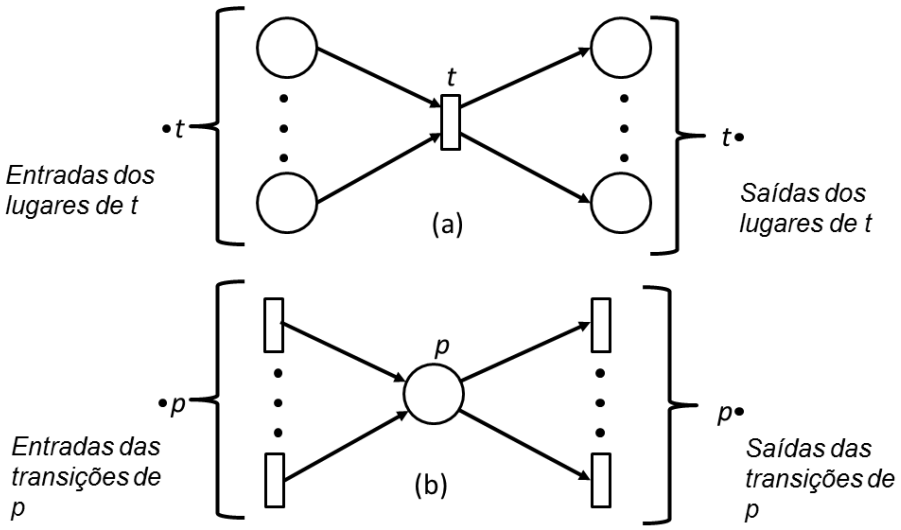


Figura 4.4: Simbologia para (a) o conjunto de entradas e saídas para os *lugares* de t (b) o conjunto de entradas e saídas para as transições de p

bora apenas uma delas pode ser disparada. Essa configuração é chamada de conflito, escolha ou decisão. Outro exemplo mostrado na Figura (b) 4.5 , indica uma atividade paralela determinística, que dois *lugares* estão simultaneamente marcados. No último exemplo, exibido nas Figuras (c) e (d) 4.5, mostra um caso de conflito e concorrência na mesma rede. Isso ocorre devido T_0 e T_2 serem concorrentes, mas conflitam com T_1, ou seja, ou há o disparo de T_0 e T_2, ou o disparado T_1. Assim, há uma confusão assimétrica sendo T_0 concorrente com T_1, conflitante com T_2 caso T_1 disparar antes de T_2.

4.5.1 Máquinas de Estado

As Máquinas de Estado - *State Machine (SM)* - são uma RP ordinária (todos os arcos têm peso igual a 1) cujo as transições T possuem apenas 1 entrada e 1 saída para cada lugar.

isto é:

$$|t\bullet| = |\bullet t| = 1, \forall t \in T$$

Isto significa que não há concorrência, mas pode haver conflitos pelos disparos de várias transições simultaneamente. Obviamente, que neste último caso, corresponde aos SMs não determinísticos.

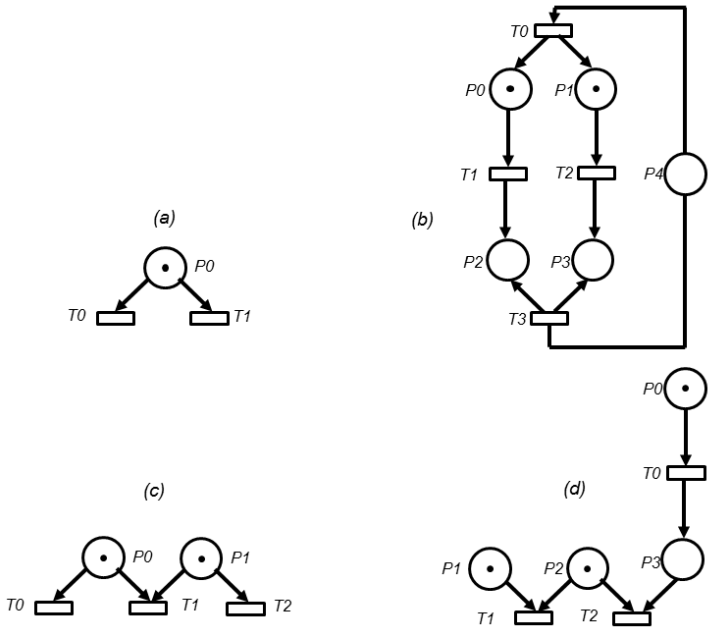


Figura 4.5: Simbologia para (a) RP com conflito na estrutura, rede não determinística (b) RP contendo uma ação determinística paralela. (c) RP contendo confusão simétrica, onde T_0 e T_2 são concorrentes e estão em conflito com T_1. (d) RP com confusão assimétrica: T_0 é concorrente com T_0 porém pode estar em conflito com T_2 se T_1 disparar antes de T_2

4.5.2 Grafo Marcado

Os Grafos Marcados - *Marked Graphs (MG)* - é uma RP ordinária (todos os arcos têm peso igual a 1) onde cada lugar P tem apenas uma transição de entrada de uma transição de saída. Portanto esse sistema não há conflito, porém há concorrência,

isto é:

$$|p\bullet| = |\bullet p| = 1, \quad \forall p \in P$$

4.5.3 Livre Escolha

A rede de Livre Escolha - *Free-Choice (FC)* - é uma RP ordinária (todos os arcos têm peso igual a 1) onde cada arco de saída deve possuir uma única entrada para a transição correspondente. Por isso, podem existir concorrências ou conflitos, mas nunca ambos simultaneamente.

isto é:

$$\forall p \in P, |\bullet p| \leq 1, ou, \bullet(p\bullet) = \{p\};$$

- equivalentemente:

$$\forall p_1, p_2 \in P, p_1\bullet \cap p_2\bullet \neq \oslash \Rightarrow |p_1\bullet| = |p_2\bullet| = 1;$$

4.5.4 Livre Escolha Estendida

Uma rede de Livre Escolha Estendida - *Extended Free-Choice (EFC)* - é uma RP que para cada transição T, contendo mais de um arco de entrada $(p_1, p_1, ..., p_n)$, e o conjunto de saída, leva para o mesmo conjunto de transições.

isto é:

$$p_1 \bullet \cap p_2 \bullet \neq \oslash \Rightarrow p_1 \bullet = p_2 \bullet, \quad \text{para} \quad \forall \{p_1, p_2 \in P\}$$

4.5.5 Escolha Assimétrica

Uma rede de Escolha Assimétrica - *Assymmetric Choice (AC)* - também conhecida como rede simples é uma RP ordinária (todos os arcos têm peso igual a 1) permitindo-se concorrências e conflitos, mas não de modo assimétrico.

isto é:

$$p_1 \bullet \cap p_2 \bullet \neq \oslash \Rightarrow p_1 \bullet \subseteq p_2 \bullet, \text{ ou } p_1 \bullet \supseteq p_2 \bullet, \text{ para } \forall \{p_1, p_2 \in P\}$$

4.5.6 Resumo Esquemático das Sub-Classes das Redes de Petri

Esquemático dos cinco tipos de subclasses das RP na Figura 4.6 a seguir:

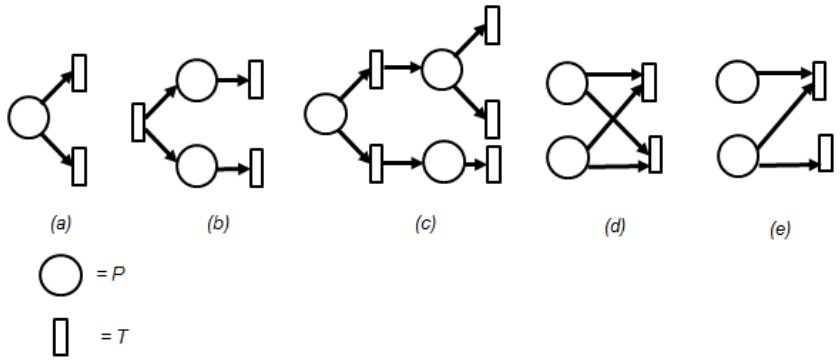


Figura 4.6: Sub-Classes das RP: (a) Máquina de Estado (b) Grafo Marcado (c) Livre Escolha (d) Livre Escolha Estendida (e) Escolha Assimétrica

Diagrama Venn onde estabelece a configuração das sub-classes:

4.6 Propriedades

Uma das principais vantagens em se modelar os SEDs em PN é a utilização das ferramentas de análise para avaliar o comportamento do sistema. Existem dois tipos

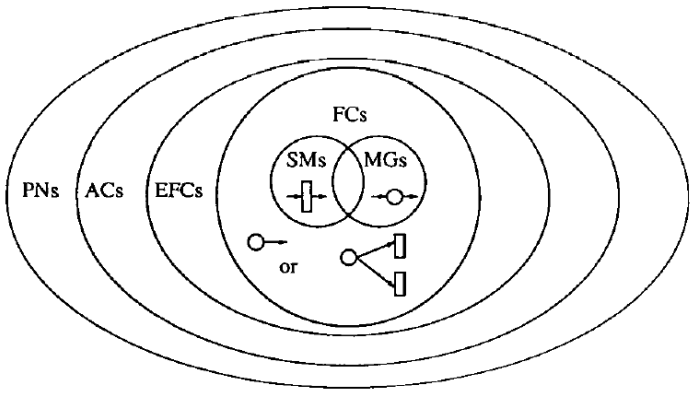


Figura 4.7: Figura retirada do artigo (MURATA, 1989), onde mostra a configuração das sub-classes de uma RP de acordo com diagrama Venn

de propriedades: as comportamentais, que estuda a rede a partir de uma marcação inicial; e as estruturais, que independe da marcação inicial.

A seguir serão detalhadas as principais características de cada uma delas. Para exemplificar, as transições da Figura 4.8 apresenta uma PN com sua sequência de evoluções.

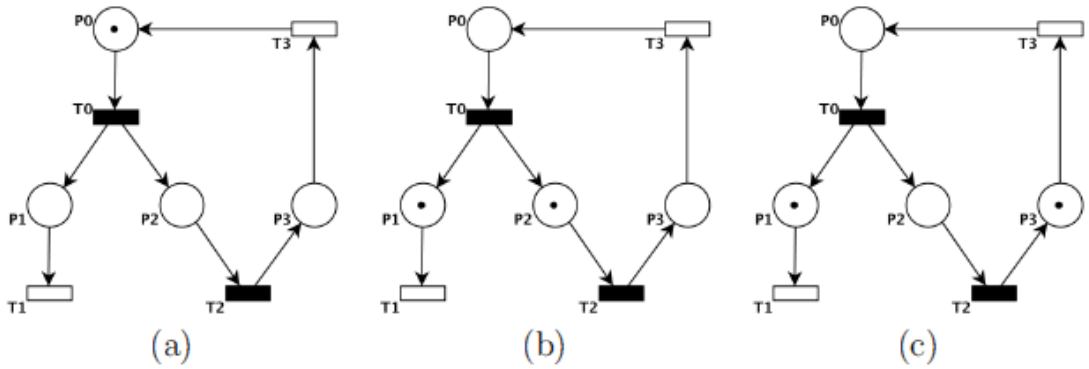


Figura 4.8: (a) RP em seu estado inicial, (b) RP estado intermediário após disparo de T_0 (c) RP em estado final após disparo das transições T_1 e T_2

Alcançabilidade

O disparo de uma transição habilitada para alterar a distribuição de fichas na PN, ocorre de acordo as regra de transições. Uma sequência de disparos irá resultar numa sequência de marcações. Uma marcação μ_m é dita acessível a partir de uma marcação μ_0 se existir uma sequência de disparos que leve a PN evoluir de μ_0 a μ_m. O conjunto de todas as possíveis marcações alcançáveis a partir de μ_0 em uma rede (N, μ_0) é indicado por $R(\mu_0)$. O conjunto de todas as possíveis sequências de transições a partir de μ_0 em uma rede (N, μ_0) é denominada por $L(\mu_0)$. Por exemplo,

na Figura 4.8, podemos alcançar um número infinito de diferentes estados, porém podemos reduzir o em apenas sete transições: (Figura 4.8 (a) $\{1,0,0,0\}$, (Figura 4.8 (b) $\{0,1,1,0\}$, (Figura 4.8 (c) $\{0,1,0,1\}$, $\{0,0,0,1\}$, $\{0,\omega,1,0\}$ e $\{0,\omega,0,1\}$, sendo ω qualquer número inteiro positivo maior que 1.

Limitada

Uma rede de Petri (N,μ_0) é dita ser limitada se o número de fichas de cada lugar P não exceda um número k-finito para qualquer marcação alcançável a partir do estado inicial μ_0, isto é, $\mu(P) \in k$ para cada lugar P e cada marcação $\mu \in R(\mu_0)$. Uma rede limitada significa dizer que não haverá sobre fluxo em buffers ou registros, independente da sequência de disparos da rede. Por exemplo, a Figura 4.8 mostras P_0, P_2 e P_3 limitadas em 1 ficha, porém P_1 é ilimitado. Significa dizer que a PN (N,μ_0) é limitada em 1.

Vivacidade

A rede de Petri (N,μ_0) é dita ser viva se, não importando qual marcação alcançada a partir de μ_0, é possível progredir em qualquer sequência de transição de rede sem interromper a dinâmica da PN. Isto significa que a rede está livre e desimpedida de bloqueios operacionais, independente de quais caminhos de transições for seguido. Portanto, a PN em 4.8 é dita livre, ou seja, sem bloqueios.

4.7 Equações matriciais

Nessa seção será introduzido a noção das equações matriciais aplicadas em SEDs demostradas em (IORDACHE; MOODY; ANTSAKLIS, 2002), com o objetivo de qualificar a análise de uma RP. Para esse método é importante definir alguns elementos.

Seja o conjunto de números inteiros Z, com n números de *lugares* e m o números de transições. Os arcos de conexão entre as transição e os *lugares* são descritos pela matriz $D^- \in Z^{n\times m}$, já os arcos entre os *lugares* e as transições são descritos pela matriz $D^+ \in Z^{n\times m}$. Para a construção da matriz D^- os elementos $i_{n\times m}$ leva o número 0, se não há conexões entre os transições m com os *lugares* n, caso contrário leva o peso do arco igual a $(\omega\,(m,n))$. De maneira semelhante, para a construção da matriz D^+ os elementos $i_{n\times m}$ leva o número 0, se não há conexões entre os *lugares* n com as transições m, caso contrário leva o peso do arco igual a $(\omega\,(m,n))$. Com essa configuração, os elementos das matrizes D^- e D^+ são maiores ou iguais a zero.

A representação dos estados marcados é por um vetor coluna n, cujo o símbolo é μ, seu comprimento está associado ao número *lugares*. O número de fichas em cada

lugar, representa os elementos do vetor na condição $p_n\left(\mu\left(\mu, p_n\right)\right)$. A evolução de uma RP é em etapas. O vetor coluna q representa as *transições* que serão disparados naquele etapa, onde os $j-$ésimos elementos que o compõe tem valor 0 se a transição não for disparada e valor 1 se as *transições* forem disparadas.

A partir do vetor de *transições* q, sua validação é determinada analisando as entradas não nulas habilitadas em 1 das. Essa condição pode ser determinada utilizando as seguintes expressões:

$$\mu \geq D^{-} q \tag{4.6}$$

A matriz de incidência da RP é definida como:

$$D = D^{+} - D^{-} q \tag{4.7}$$

Utilizando o exemplo da Figura 4.8 para descrever o comportamento das matrizes D, D^{+} e D^{-}

$$D^{+} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.8}$$

$$D^{-} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{4.10}$$

Quando uma RP está desprovida de auto laço, utiliza-se a matriz D^{+} e D^{-} para definir a matriz D, portanto podemos utilizar a desigualdade 4.6, obtendo:

$$\begin{aligned} \mu + D^{+} q &\geq D^{-} q \\ \mu + \left(D^{+} q + D^{-} q\right) q &\geq 0 \\ \mu + D q &\geq 0 \end{aligned} \tag{4.11}$$

Precauções devem ser tomadas ao utilizar as desigualdades 4.6 e 4.11. Quando q (vetor que caracteriza disparos concorrentes de múltiplas *transições*), for um vetor de zeros com apenas um elemento igual a 1, esta desigualdade não poderá ser usada. Se isso ocorrer o vetor q da RP assume valores descritos por:

$$\mu \leftarrow \mu + D^{+}q \tag{4.12}$$

Desta forma, a regra de evolução da RP se caracteriza por:

$$\mu_{k+1} = \mu_k + Dq_k \tag{4.13}$$

$$D \in Z^{n \times m}, \mu \in Z^n, q \in Z^m, (\mu, q \geq 0) \tag{4.14}$$

4.8 Comentários finais

A modelagem de SEDs representa de formam sólida e confiável grande parte dos sistemas existentes. O formalismo baseado em redes de Petri é uma ótima alternativa simular seu comportamento e restrições. A implementação é mais utilizada em estudos de estruturas determinísticas, mas em alguns casos específicos, elas também podem ser aplicadas em sistemas não-determinística e estocásticos.

Assim como é possível desenvolver lógicas de programação utilizando código C, também é possível desenvolver estruturas de programação utilizando as redes de Petri. A representação dos sistemas de controle de missão em uma linguagem formal em alto nível, aliada às análises das propriedades, torna esse formalismo uma ferramenta atrativa para o descrever de sistemas complexos, tal qual o utilizado nesta dissertação.

-

Capítulo 5

Sistemas de Controle de Missão Definidos por Redes de Petri

No capítulo anterior foi mostrado que as Redes de Petri - Petri Nets (PNs) - são uma boa ferramenta para elaborar um Sistema de Controle de Missão (SCM) e modelar as *ações primitivas* de forma simples e eficiente. As PNs possuem recursos interessantes para descrever os Sistemas de Eventos Discretos (SEDs), bem como apresentam objetos de diagnósticos e análise de comportamento estrutural do SCM.

A proposta de (PALOMERAS et al., 2012) define uma PN capaz de modelar e gerenciar as *ações primitivas* de um sistema de controle de missões de um AUV. Chamado de *Petri Net Building Block* (PNBB), essa estrutura interpreta as *ações primitivas* formadas em pequenas PNs que, ao ser agrupadas, descrevem o comportamento completo do sistema de controle de missão. Os elementos que compõem as PNBBs estão classificados em dois grupos. Existe o grupo *tarefas* que supervisiona as atividades das *ações primitivas* e há o grupo *estruturas de controle* que coordena o fluxo entre as redes de forma sequencial, paralela, iterativa ou condicional. Pode-se garantir o resultado final das estruturas PNBBs herdará as mesmas propriedades das redes menores.

5.1 Composição do Sistema

As sequências de eventos que ocorrem durante as missões envolvem dois fatores. Um deles é o objetivo da missão e o outro são as mudanças sofridas pelo meio e/ou pelo AUV durante a atividade. Uma missão é estruturalmente definida pela sequência de habilita/desabilita das *ações primitivas*. Essas ações geram fluxos de eventos que atualizam as informações do ambiente de navegação de acordo com o conceito dos SEDs. (CASSANDRAS; LAFORTUNE, 2008) definiu em seu livro: “Um SED é um sistema orientado à eventos, ou seja, sua ocorrência depende completamente da

ocorrência dos eventos assíncronos ao longo do tempo." A definição formal de um SED é:

$$\Sigma = \{S, A, E, \gamma\} \ , \tag{5.1}$$

onde:

- $S = \{s_1, s_2, \ldots, s_n\}$ é um conjunto finito de estados;
- $A = \{\lambda, a_1, a_2, \ldots, a_n\}$ é um conjunto de ações incluindo a ação nula λ. A é conhecido também como um conjunto de entrada;
- $E = \{\lambda, e_1, e_2, \ldots, e_n\}$ é um conjunto de eventos incluindo a ação nula λ. E é conhecido também como um conjunto de saída;
- $\gamma : S \times E \rightarrow 2^S \times A$ é uma função de transição de estado.

A função de transição de estado é dada por $(s, e) = (s', a)$, onde s, $s' \in S$, $e \in E$ e $a \in A$, e representa o comportamento do sistema em resposta à detecção de eventos. Assim, quando o sistema está no estado s e o evento e é recebido, a rede pode evoluir para o estado s' e executar a ação a . A ação nula (λ), inclusa nos conjuntos E e A, permite que a rede possa mudar de estado mesmo se nenhum eventos for recebido $(\gamma, \lambda) = (s', a)$. Da mesma forma, o estado de transição pode sofrer mudanças, mesmo não recebendo ações, logo: $(\gamma, e) = (s', \lambda)$.

5.2 Ações Primitivas

Em robótica, *ações primitivas* são funcionalidades básicas que descrevem os comportamentos indispensáveis de um robô. Por exemplo, as *ações primitivas* de um AUV são: Atingir a profundidade "x" (*AchieveDepth*), navegar para um ponto específico (*Goto*), abortar operação (*Alarm*), etc. Em todos esses casos, as *ações primitivas* são funções que conduzem o robô a uma situação pré-estabelecida. A proposta do trabalho é modelar apenas as camadas superiores dentro da arquitetura de controle, ou seja, as camadas deliberativas como mostrado no Capítulo 3. De acordo com (PALOMERAS et al., 2009) as seguintes condições das *ações primitivas* devem ser asseguradas para seu modelamento:

- As *ações primitivas* são independentes, ou seja, podem ser ativadas livremente, mesmo que outras *ações primitivas* já estejam ativas.
- Se uma ou mais *ações primitivas* não podem ser habilitadas simultaneamente é porque compartilham os mesmos recursos, portanto formam uma única *ação primitiva*.

- As *ações primitivas* devem ser protegidas e, mesmo em casos de falhas (hardware e/ou software), devem ser capazes de emitir informações confiáveis.

A preocupação durante o desenvolvimento das *ações primitivas* é fornecer um modelo genérico onde somente dois eventos poderão ser enviados: habilita (*enabling*) ou desabilita (*disabling*). Para elaborar uma ação mais refinada, pode-se agregar módulos ou sistemas específicos do AUV. Por exemplo, pode-se habilitar uma *ação primitiva* que leve o robô a um local específico na orientação (x, y, z) e essa PNBB compartilha dados possibilitando habilitar ou desabilitar vários outras *ações primitivas* a partir das informações geradas do seu estado atual. Três modelos genéricos de PNs são propostos para descrever as ações e os eventos, esses sistemas são mostrados pelas figuras abaixo:

A Figura 5.1 ilustra as três subclasses de PNs.

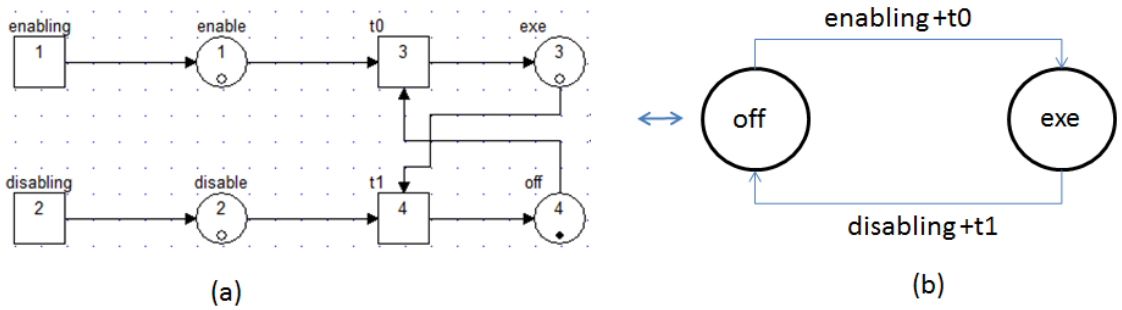


Figura 5.1: Figura referente ao modelo off/exe empregado nas *ações primitivas*

A Figura 5.1(a), desenvolvida no software *Netlab*, mostra as funções (*enable*)/(*disable*) genéricas aplicadas a quaisquer eventos. Para habilitar as *ações primitivas*, o lugar (*enable*) tem que estar marcado. Em seu estado inicial, a transição t_0 dispara, marcando o lugar (*exe*) e indicando que a rede está em execução. Em seguida, basta a transição t_1 habilitar o lugar (off) da PNBB, para desabilitar a rede.

A Figura 5.2(a) mostra outra função (*enable*)/ (*disable*) e (*primitive ok*) para o modelamento de outros casos específicos. Para habilitar a ação, o lugar (*exe*) tem que estar marcado. Se o objetivo for alcançado, a transição t_2 dispara a ficha habilitado o lugar *ok*. A ação pode ser desabilitada em duas situações: quando *exe* ou *primitive ok* estiverem marcados, desde que as transições t_2 ou t_5 estejam habilitadas.

A Figura 5.3(a) mostra um PNBB com quatro estados: *(i)* desabilitado, *(ii)* buscando objetivo, *(iii)* objetivo atingido e *(iv)* erro. Os dois primeiros, *off* e *exe* estão habilitados. Enquanto a rede executa a ação, o estado *exe* está habilitado e o SCM acompanha os eventos da missão. Quando as pré-condições dos estados são

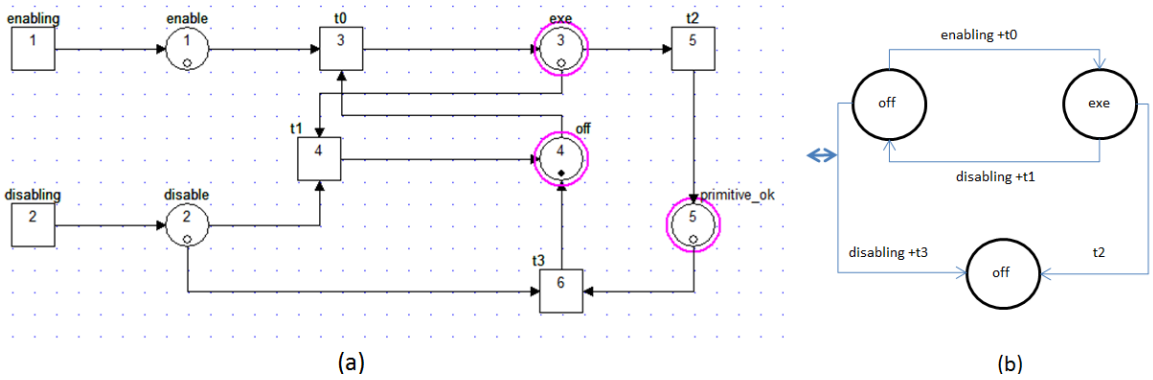


Figura 5.2: Figura referente ao modelo off/exe/primitive ok empregado nas *ações primitivas*

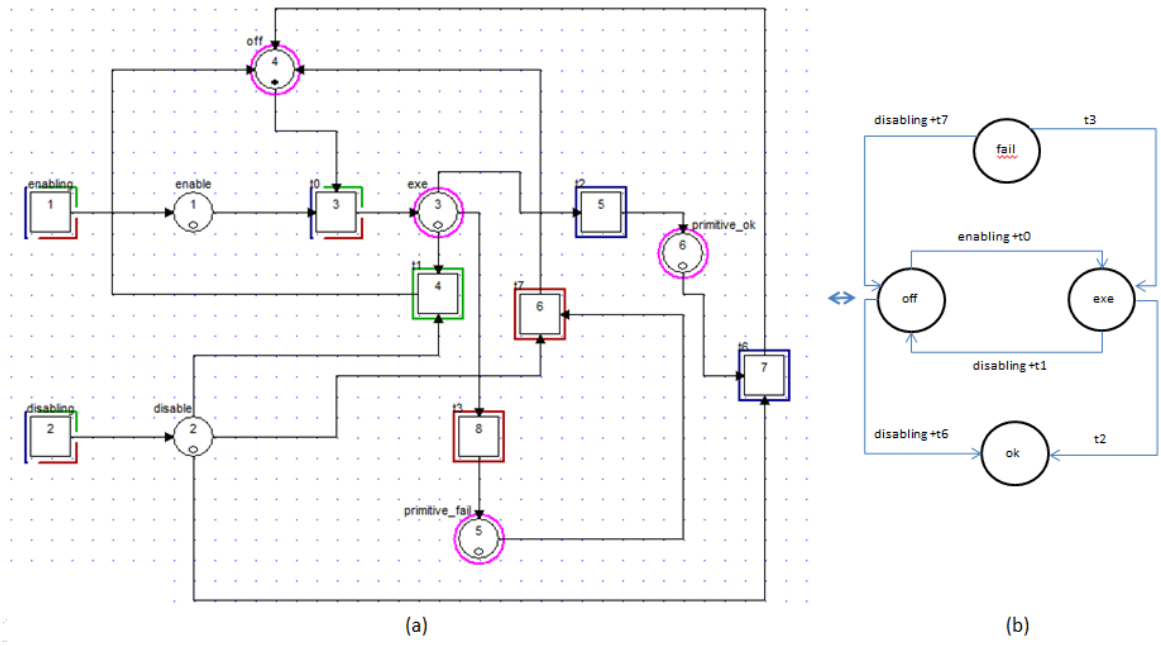


Figura 5.3: Figura referente ao modelo off/exe/primitive/fail ok empregado nas *ações primitivas*

atingidos, a transição t_2 é disparada e a ficha evolui para o lugar *ok*. Entretanto, se uma falha ocorrer durante a execução desta PNBB, a transição t_3 é disparada e o estado *fail* é habilitado. Nestas condições são habilitadas as transições t_1, t_6 ou t_7, marcando os lugares *exe*, *ok*, ou *fail* respectivamente. A Figura 5.3 (b) mostra o esquemático do modelo em PN.

Essas configurações certificam que os estados de entrada/saída estão sendo executados seguramente pelas PNs. As três redes apresentam transições não imediatas de contenção, denominadas ativação e desativação. Os eventos são habilitados pela outras rede hierarquicamente superiores.

Nas Figuras 5.1, 5.2 e 5.3, quando t_0 é disparada, suas fichas habilitam as PNBBs. Mas se t_1 habilitar em 5.1 ou t_1 / t_3 em 5.2 ou t_1 / t_6 / t_7 em 5.3, os comandos de desativação são disparados e as transições levam as fichas para o lugar *off*, desabilitando a PNBB.

Quando a rede alcança o objetivo da *ação primitiva*, um evento e uma transição são gerados informando à camada reativa. Cada *ação primitiva* gera seus respectivos eventos e disparam suas respectivas transições, dando permissão para habilitar/desabilitar as camadas superiores e/ou elementos nas camadas reativas.

5.2.1 Verificação das *Ações Primitivas*

Após definir o PNBB utilizado é importante que algumas propriedades sejam verificadas. A rede tem que ser alcançável, ou seja, a partir da marcação inicial todos os estados terão que ser atingidos. A rede tem que ser desobstruída, isto é, pode evoluir em qualquer sequência de transição sem ter sua dinâmica interrompida. Esses e outros estudos de propriedades de rede podem ser vistos no com mais detalhes no Capítulo 4.

Em grandes estruturas de redes contendo um número elevado de estados, tais análises demandam um grande custo computacional. Para isto, as PNBBs são mantidas reduzidas afim de tornar a análise viável.

As propriedades de alcançabilidade e vivacidade pode ser verificadas na Figura 5.3. Esses estados marcados representam um vetor da rede, sendo: $S_{\mu 1} = \{\mu(enable), \mu(disable), \mu(exe), \mu(off), \mu(ok), \mu(fail)\}$. A PNBB pode evoluir livremente sem bloqueios a partir do estado inicial $S_{\mu off}$, logo a rede poderá atingir qualquer outro estado e retornar para $S_{\mu off}$. Isso significa que a rede é viva, ou sem *deadlocks*. Os estados alcançáveis são: $S_{\mu off} = \{0, 0, 0, 1, 0, 0\}$, $S_{\mu exe} = \{0, 0, 1, 0, 0, 0\}$, $S_{\mu ok} = \{0, 0, 0, 0, 1, 0\}$, $S_{\mu fail} = \{0, 0, 0, 0, 0, 1\}$. O lugar marcado $S_{\mu off}$ indica a rede pronta para ser ativada. O $S_{\mu exe}$ informa que a *ação primitiva* está em busca de seu objetivo. O $S_{\mu ok}$ determina que o objetivo foi alcançado e finalmente, o $S_{\mu fail}$ indica que houve alguma falha durante a ação. Os quatro outros estados podem

ocorrer sendo: $S_{\mu enabled} = \{0,1,0,0,0,0\}$, $S_{\mu disabled1} = \{0,1,1,0,0,0\}$, $S_{\mu disabled2} = \{0,1,0,0,1,0\}$ e $S_{\mu disabled3} = \{0,1,0,0,0,1\}$. Pelas informações do grafo de evolução da rede, representadas pelos vetores marcados acima da rede da Figura 5.3, pode-se perceber que esta PNBB é alcançável.

Nesse mesmo grupo é possível determinar também a propriedade de limite da rede da Figura 5.3, ou seja, a PNBB pode evoluir sem que o número de fichas de cada lugar não exceda um número de 1 ficha para qualquer marcação alcançável a partir do estado inicial. Essa propriedade é importante, pois evita que a estrutura tenha sobre-fluxo de informações em quaisquer direções que o sistema evoluir.

5.2.2 Construção dos Blocos

Segundo (PALOMERAS et al., 2012), as PNBBs são estruturas básicas utilizadas para codificar as *ações primitivas* em redes de Petri. Sua implementação necessita de *interfaces* específicas que estabelecem a conexão entre as demais PNBBs que constituem o SCM. Todas as estruturas das PNBBs têm que apresentar, no mínimo, uma *interface externa* I no conjunto de lugares onde:

$$\begin{aligned} I &= I_{\mu input} \cup I_{\mu output} \\ I_{\mu input} &\cap I_{\mu output} \neq \oslash \\ \forall p \in I_{\mu input},\ &\bullet p = \oslash \\ \forall p \in I_{\mu output},\ &p\bullet = \oslash \end{aligned} \tag{5.2}$$

Os estados de entrada marcados $p \in I_{\mu input}$ são chamados de *estados de entrada* e os estados de saída marcados $p \in I_{\mu output}$ são chamados de *estados de saída*. Os diagramas de blocos a seguir definem o comportamento de cada um dele:

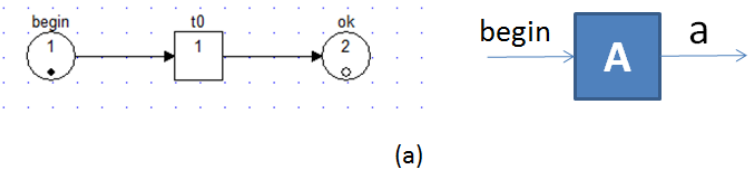
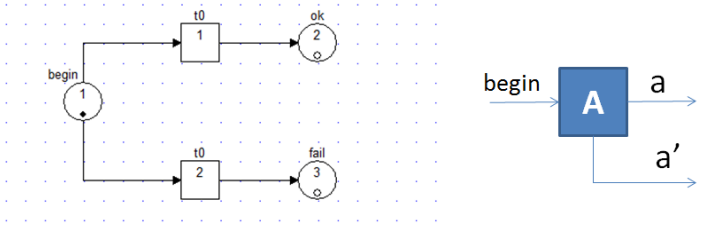


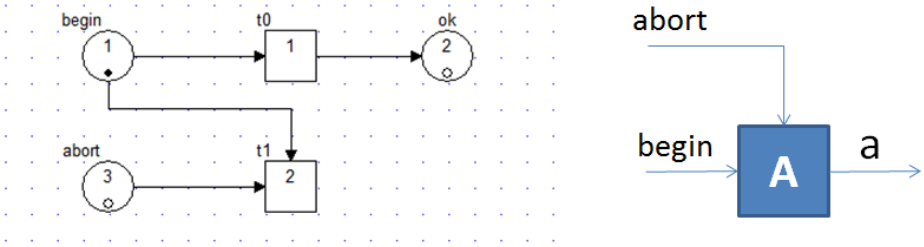
Figura 5.4: Figura da rede begin/ok empregado na interface das PNBB

- A Figura 5.4 mostra a estrutura de uma PNBB sendo $I_{\mu input} = \{begin\}$, $I_{\mu output} = \{ok\}$. O lugar *ok* é somente marcado quando aquela rede alcança o objetivo, ao disparar t_0 .



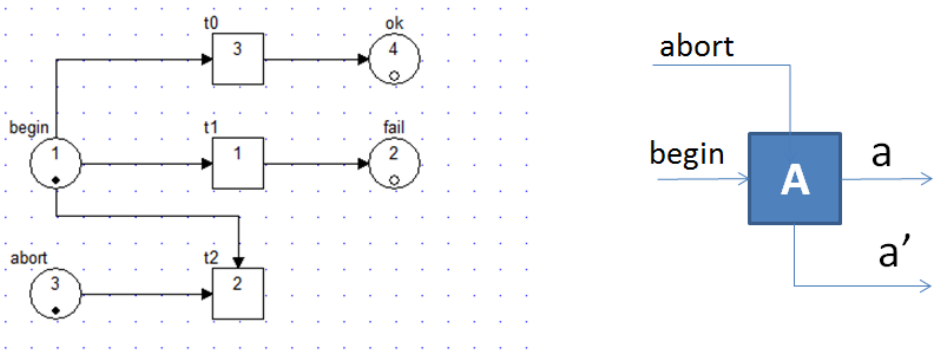
(b)

Figura 5.5: Figura da rede begin/ok/fail empregado na interface das PNBB



(c)

Figura 5.6: Figura da rede begin/ok/abort empregado na interface das PNBB



(d)

Figura 5.7: Figura da rede begin/ok/fail/abort empregado na interface das PNBB

- A Figura 5.5 mostra a estrutura de uma PNBB sendo $I_{\mu input} = \{begin\}$, $I_{\mu input} = \{ok, fail\}$. Uma vez a rede habilitada, ela pode alcançar dois estados: *conclusão da ação* ou *falha da ação*, marcando os lugares *ok* ou *fail*, respectivamente. Essa PNBB possui um conflito entre t_0 e t_1, portanto, a transição que disparar primeiro definirá o estado final da rede.

- A Figura 5.6 mostra a estrutura de uma PNBB sendo $I_{\mu input} = \{begin, abort\}$, $I_{\mu output} = \{ok\}$. Essa rede inclui dois lugares de entrada e um de saída. Quando ela é habilitada, o lugar *begin* é marcado, porém, em caso de anomalias, t_1 dispara e imediatamente e a ficha é removida, desativando da rede. Caso t_0 dispare, o lugar de saída é marcado e o estado *ok* é habilitado.

- A Figura 5.7 mostra a estrutura de uma PNBB sendo $I_{\mu input} = \{begin, abort\}$, $I_{\mu output} = \{ok, fail\}$. Essa rede inclui dois lugares de entrada e dois de saída, sendo eles, *conclusão da ação* ou *falha da ação*. O disparo da transição t_0 e t_1 marca o lugar *ok* ou *fail*, respectivamente. Entretanto se o lugar *abort* for marcado e t_2 for habilitada, a rede é desativada, removendo as fichas dos lugares *ok* e *fail*.

Para a construção das estruturas dos PNBBs, dois modelos de rede são necessários: as redes de *tarefas* e as redes de *estrutura de controle*. A rede de *tarefas* é utilizada para supervisionar as *ações primitivas* definidas em PNBBs. Ela promove somente uma interface, a *interface externa* (I), estabelecendo a conexão entres todas as PNBBs do SCM. Já a rede de *controle de estrutura* tem o objetivo de fazer uma composição entre tarefas (I_1) e composições múltiplas entre várias *tarefas*, podendo ser $(I_2, \ldots, I_n)$.

Exemplo

A instrução que permite iniciar a rede PNBB e consequentemente, iniciar a execução da *ações primitivas* é promovida pelo SCM. Por exemplo, a Figura 5.8 é uma rede integrada entre a *ação primitiva* da Figura 5.3 e o bloco PNBB da Figura 5.7. O conjunto de eventos é caracterizado por:

$$
\begin{aligned}
&I_{\mu input} = \{begin, abort\} \\
&I_{\mu output} = \{ok, fail\} \\
&I = I_{\mu input} \cup I_{\mu output} \{begin, abort, ok, fail\}
\end{aligned}
$$

Quando tt_0 da Figura 5.8 dispara sua ficha a rede é habilitada. As transições tt_1, tt_2, tt_3 permitem o funcionamento da *ação primitiva*, movendo a ficha para o lugar *enable*. Essa PNBB envia eventos para as camadas superiores e inferiores nas

seguintes situações: quando é recebido o evento *ok*, a transição t_2 dispara, marcando os lugares *ok* e *isok*, habilitando ambos estados. A mesma situação ocorre em eventos como *fail*, tendo lugares como *primitivefail* e *isfail* marcados. Uma importante diferença acontece (no estado *off*) entre as Figuras 5.3 e 5.8. Na Figura 5.8 esse lugar atua com uma proteção, evitando a execução da *ação primitiva* mesmo que ela já esteja em funcionamento. Para isto, ao invés do lugar ser conectado pela transição t_0, ela é conectada indiretamente com a transição tt_0. A estrutura e os métodos da rede foram inspirados, em sua essência, pelas ideias discutidas em (IORDACHE; MOODY; ANTSAKLIS, 2002).

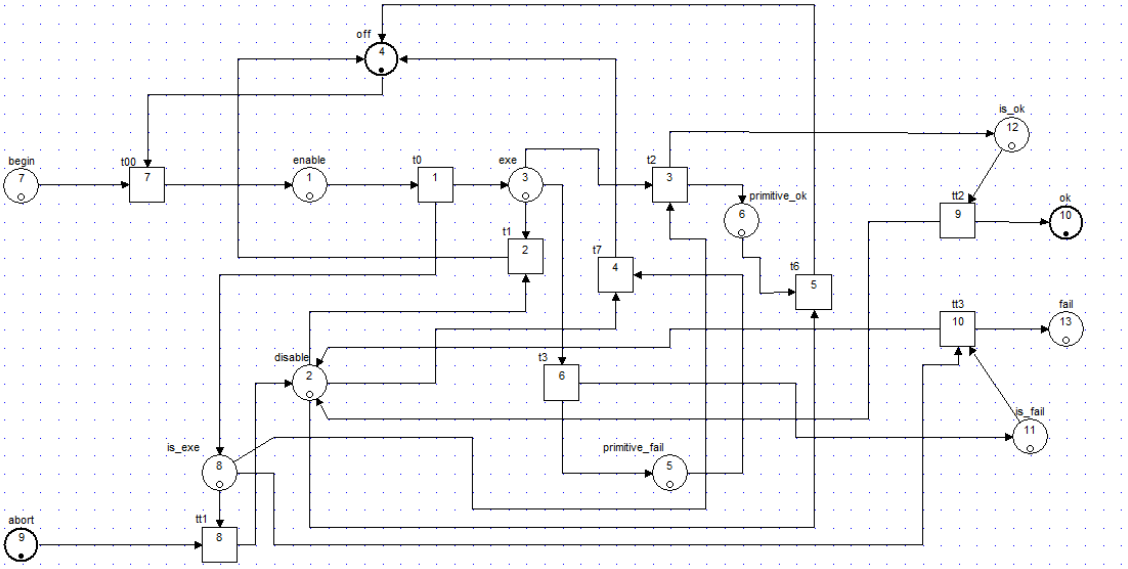


Figura 5.8: Figura da integrada de uma PBNN com um modelo de *ação primitiva* feito no software *Netlab)*

Dinâmica

A *ação primitiva* da Figura 5.8 é sempre habilitada pela estrutura superior. Quando a transição tt_0 é disparada a rede é habilitada, e será desabilitada se somente se:

1. O lugar *ok* se marcado, através do disparo de t_2;
2. Se uma falha for detectada durante a execução da ação, t_3 dispara, marca o lugar *fail*, produz o disparo de tt_3 e desabilita a missão;
3. Caso uma falha seja detectada durante a missão, a estrutura superior marcar o lugar *abort*. Quando a *ação primitiva* estiver em execução, a transição tt_1 dispara e desabilita a função.

Portanto, de acordo com o grafo de alcançabilidade da Figura 5.8 e no esquemático representado na Figura 5.9, pode-se resumir essa rede como um conjunto de oito

estados onde: $S_{\mu i} = \{\mu\,(begin),\ \mu\,(abort),\ \mu\,(ok),\ \mu\,(fail),\ \mu\,(enable),\ \mu\,(disable),\ \mu\,(off),\ \mu\,(primitiveok)\}$. Sendo o conjunto de lugares que compõe a interface: $\{begin, abort, ok, fail\}$. O conjunto de lugares das *ações primitivas*: {*enable*, *disable*, *off*, *exe*, *primitive ok*, *primitive fail*}. Os demais conjuntos de lugares das *ação primitiva* são: $\{isexe, isok, isfail\}$. Os quatro estados alcançáveis são descritos por:

$$
\begin{aligned}
S_{\mu taskdisabled} &= \{0,0,0,0,0,0,0,0,0,1,0,0,0\}\ , \\
S_{\mu taskexe} &= \{0,0,0,0,1,0,0,0,0,0,1,0,0\}\ , \\
S_{\mu taskok} &= \{0,0,1,0,0,0,0,0,0,1,0,0,0\}\ , \\
S_{\mu taskfail} &= \{0,0,0,1,0,0,0,0,0,1,0,0,0\}\ .
\end{aligned}
$$

A partir do estado inicial ($S_{\mu taskdisabled}$), apenas dois estados finais podem ser alcançados. Ambas situações tem o estado final *off* marcado juntamente com o *ok* ($S_{\mu taskok}$) ou o lugar *fail* ($S_{\mu taskfail}$), dependendo do tipo de evento que será produzido durante a execução da *ação primitiva*. Por outro lado, a missão pode ser comprometida se uma *ação primitiva* hierarquicamente superior abortar a missão, marcando o lugar *abort*.

Essa rede é livre de bloqueios e pode alcançar quaisquer estados a partir do inicial.

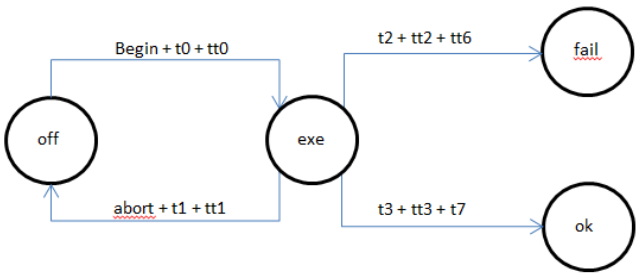


Figura 5.9: Grafo de alcançabilidade da PN apresentada na Figura 5.8

A característica da PPNB de *interface* é dependente do comportamento das *ações primitivas*. Se uma *ação primitiva* for constituída por apenas uma entrada/saída é possível utilizar o modelo representado da Figura 5.4. Em casos mais sofisticados, onde dois eventos de saída são necessários, a melhor representação está na Figura 5.5. Sistemas ainda mais refinados e complexos podem ser modelados pela rede da Figura 5.7. Embora haja uma boa quantidade de variações de interfaces, de acordo com do fluxo informações, outras estruturas podem ser obtidos a partir de combinações dos modelos descritos acima. Na lista de variações há a possibilidade da criação de estruturas paralelas, sequencias e condicionais.

A Figura 5.11 apresenta a mesma PNBB de interface mostrada na Figura 5.1, integrada a respectiva *ação primitiva*. Do ponto de vista da *interface*, o compor-

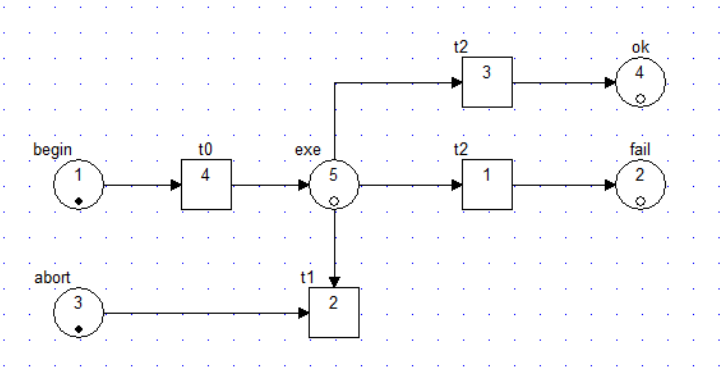


Figura 5.10: Exemplo de um simples PNBB com estados como: *begin ok abort fail exe*

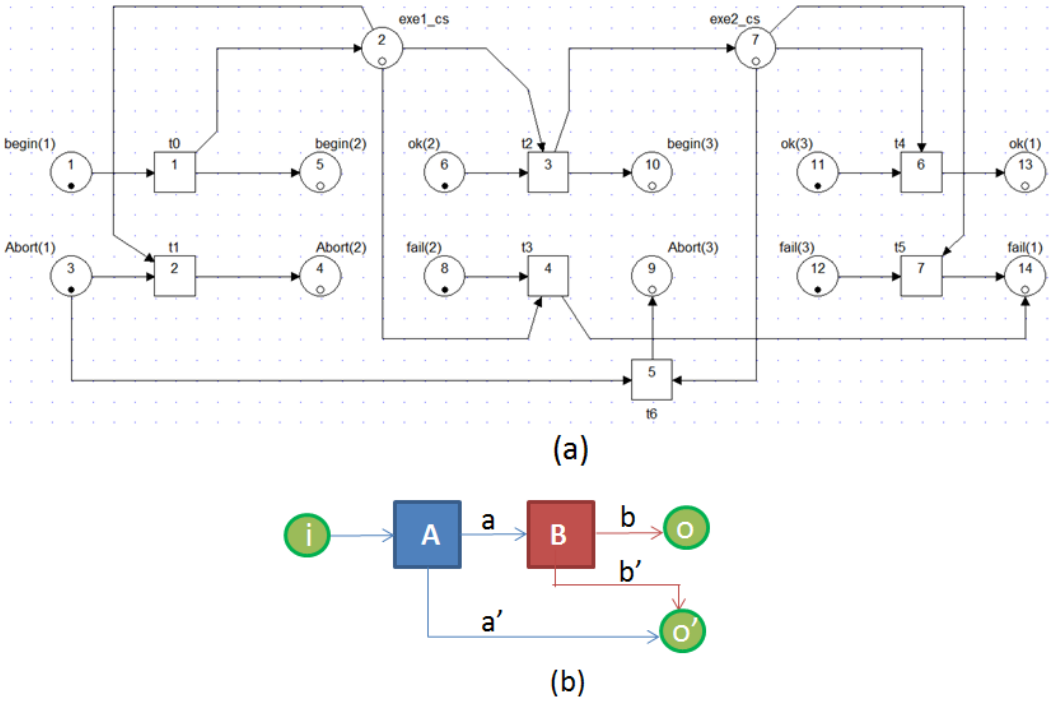


Figura 5.11: (a)Modelo de estrutura da PNBB da Figura 5.1 em sequencia com a ação primitiva 5.10. (b) Modelo esquemático da função.

tamento dessa rede é o mesmo apresentado pela rede da Figura 5.1, isso significa que as propriedades (alcançabilidade, vivacidade entre outras) são idênticas. Essa é a grande vantagem de se usar a representação em PNBB, pois as características das estruturas agregadas permanecem inalteradas, transferindo as propriedades das PNBBs fundamentais das redes menores para as redes maiores.

5.3 Sequências

A Figura 5.11(a) mostra uma *estrutura de controle* compostas de duas redes PBNNs conectadas em sequência. A Figura 5.11 (b) mostra o esquema, cujo os blocos A e B compõe as duas PBNNs. A saída de cada bloco usa os símbolos (a e b) e (a' e b'), para expressar *ok* e *fail*, dos blocos A e B respectivamente. Essa configuração é marcada por um acoplamento em *série* (saída do bloco A é a entrada do bloco B), sendo a entrada do sistema indicada por *i*. A saída *o* é a sem falhas na rede, já a saída *o'* é a saída com falhas.

A estrutura da Figura 5.11(a) tem as mesmas interfaces da Figura 5.10, porém de forma triplicada, sendo:

$$I_1 = \{begin_1, abort_1, ok_1, fail_1\} \ ,$$
$$I_2 = \{begin_2, abort_2, ok_2, fail_2\} \ ,$$
$$I_3 = \{begin_3, abort_3, ok_3, fail_3\} \ .$$

Percebe-se que as interfaces não são exclusivas para interligar uma ou mais redes externas (como I_1), elas também criam conexões entre as *estruturas internas* da própria rede (I_2 e I_3). Essa *estrutura de controle* contém uma ou mais *interfaces internas* permitindo a formação de uma rede a partir dos blocos PBNNs. Já a *interface externa* é responsável em conectar o bloco principal da PNBB a uma estrutura externa hierarquicamente superior. Portanto, para a construção da *estrutura de controle* entre duas ou mais redes, deve-se seguir o exemplo apresentado a seguir:

Dada a rede $N_A = \{P_A, T_A, A_A\}$, sendo: P_A o conjunto de lugares, T_A o conjunto de transições e A_A o conjunto de arcos. Se ao menos uma interface I_j cujo $j \in N\,\{0\}$, está em j, o conjunto de todos os $P \in N_A$. Dado o segundo conjunto de interface I_k, sendo $N_B = \{P_B, T_B, A_B, I_k\}$ pode-se aplicar a composição $(N_A, I_k) \otimes (N_B, I_k)$ se somente se:

- $I_j \equiv I_i \rightarrow$ locais de entrada e saída devem ser os mesmos
- $(P_A/I_i) \cap (P_B/I_j) = \oslash \rightarrow$ O resto da rede em ambos os conjuntos tem que ser diferentes, para isto é adicionado um prefixo em cada lugar.

- $T_A \cap T_B = \oslash \rightarrow$ As transições têm que ser diferentes, para isto um prefixo em cada transição é adicionado.
- $A_A \cap A_B = \oslash \rightarrow$ Os arcos têm que ser diferentes em ambas as redes.
- O resultado, depois da composição será: $(N_A, I_k) \otimes (N_B, I_k) = \{T_{\mu AB}, P_{\mu AB}, A_{\mu AB}\}$, onde $P_{\mu AB}$ contém $(P_A/I_i) \cap (P_B/I_j) \cup P'$ (conjunto da fusão das *interfaces* I_i e I_k). Para cada $p_j \in I_J$ e $p_i \in I_k$ onde $p_j \equiv p_k$ um simples lugar p_jk é adicionado ao conjunto $P_{\mu AB}$. O conjunto $T_{\mu AB}$ contém $T_A \cup T_B$, e o conjunto $A_{\mu AB}$ contém $A_A \cup A_B$.

Aplicando a composição nas duas redes PNBBs da Figura 5.11 temos o conjunto formado por $N_{\mu CS} = \{P_{\mu CS}, T_{\mu CS}, A_{\mu CS}\}$, sendo:

- $P_{\mu CS} = \{begin_1, abort_1, ok_1, fail_1, begin_2, abort_2, ok_2, fail_2, begin_3, abort_3, ok_3, fail_3, exe_{1cs}, exe_{2cs}\}$,
- $T_{\mu CS} = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6\}$
- $A_{\mu CS} = \{begin_1 \rightarrow T_0$, $abort_1 \rightarrow T_5$, $abort_1 \rightarrow T_6$, $T_0 \rightarrow begin_2$, $T_0 \rightarrow exe_{2cs}$, $T_5 \rightarrow abort_2$, $exe_{2cs} \rightarrow T_1$,$exe_{2cs} \rightarrow T_2$, $T_6 \rightarrow abort_3$, $ok_2 \rightarrow T_1$, $fail_2 \rightarrow T_2$, $T_1 \rightarrow begin_3$, $T_2 \rightarrow fail_1$, $exe_{3cs} \rightarrow T_3$, $exe_{3cs} \rightarrow T_4$, $ok_3 \rightarrow T_3$, $fail_3 \rightarrow T_4$, $T_3 \rightarrow ok_1$, $T_4 \rightarrow fail_1\}$;

Ambas PNBBs são compostas por: $N_{\mu PNBB1} = \{P\mu PNBB1,\ T\mu PNBB1,\ A\mu PNBB1\}$ e $N_{\mu PNBB2} = \{P\mu PNBB2,\ T\mu PNBB2,\ A\mu PNBB2\}$, sendo:

- $N_{\mu PNBB1} = \{begin_1, abort_1, ok_1, fail_1, exe_0\}$;
- $T_{\mu PNBB1} = \{T_0, T_1, T_2, T_3\}$;
- $A_{\mu PNBB1} = \{begin_1 \rightarrow T_0$, $abort_1 \rightarrow T_1$, $T_0 \rightarrow exe_0$, $exe_0 \rightarrow T_1$, $exe_0 \rightarrow T_2$, $exe_0 \rightarrow T_3$, $T_2 \rightarrow ok_1$, $T_3 \rightarrow fail_1\}$;
- $N_{\mu PNBB2} = \{begin_1, abort_1, ok_1, fail_1, exe_0\}$;
- $T_{\mu PNBB1} = \{T_0, T_1, T_2, T_3\}$;
- $A_{\mu PNBB1} = \{begin_1 \rightarrow T_0$, $abort_1 \rightarrow T_1$, $T_0 \rightarrow exe_0$, $exe_0 \rightarrow T_1$, $exe_0 \rightarrow T_2$, $exe_0 \rightarrow T_3$, $T_2 \rightarrow ok_1$, $T_3 \rightarrow fail_1\}$;

A *interface* $I_{\mu CS2} = \{begin_2, abort_2, ok_2, fail_2\}$ surge depois da composição entre $N_{\mu PNBB1}$ e $I_{\mu PNBB1} = \{begin_1, abort_1, ok_1, fail_1, exe_0\}$ e a *interface* $I_{\mu CS3} = \{begin_3, abort_3, ok_3, fail_3\}$ é o resultado das composições entre $N_{\mu PNBB2}$ e $I_{\mu PNBB1} = \{begin_1, abort_1, ok_1, fail_1, exe_0\}$.

5.3.1 Validação da PNBB

As propriedades de alcançabilidade, vivacidade entre outras são transferidas das redes menores para as redes maiores (PALOMERAS et al., 2009),. Em outras palavras: não é necessário estender as análises para a rede resultante, todas as características são herdadas. A PNBB é um conjunto fechado e não perde suas propriedades. Pode-se observar tal propriedade na Figura 5.11, por exemplo, onde se utiliza a integração entre duas PNBBs. A preservação da características das PNBBs são garantidas quando:

- caso o evento "abortar" seja enviado para a PNBB durante o funcionamento da *ação primitiva*, as PBNNs internas são desativadas em cascata removendo todas as fichas dos lugares, permanecendo a rede com a configuração inicial;
- quando a *ação primitiva* for completada, somente uma ficha estará marcada com o lugar *ok*;
- Se houver falha durante a execução da *ação primitiva* desta PNBB, uma ficha marcará o lugar *fail*, indicando que PNBB que detectou o evento;

Uma condição indispensável para a utilização deste modelo de *estrutura de controle* é garantir que as *tarefas* de uma PNBBs possam ser habilitadas e abortadas por uma rede (PNBB) superior. Entretanto a *ação primitiva* tem que estar iniciada. Podemos verificar essa condição na Figura 5.11 e nas demais estruturas. Em todas as configurações, somente é possível abortar a respectiva rede, se somente se, a ação estiver em execução. Neste caso, percebe-se que, para a rede para o aborto da rede T_5 tem que ser disparado e o lugar *exe* tem que estar marcado.

A Figura 5.11 (a) mostra uma estrutura de controle compostas por duas redes PBNNs conectadas em paralelo. Para sintetizar, a Figura 5.11 (b) expressa sua representação esquemática, sendo os blocos A e B compostos pelas duas PBNNs. A saída de cada bloco usa os símbolos (a e b) e (a' e b'), para indicar as saídas *ok* e *fail* respectivamente. Para a saída ser validada, tanto os lugares de *fail* quanto no lugar *ok*, a marcação do estado final tem que ser idêntica em ambos os blocos. Em outras palavras: a saída só é validada, se os blocos A e B tiverem exatamente os mesmos estados em sua saída: ok ou fail.

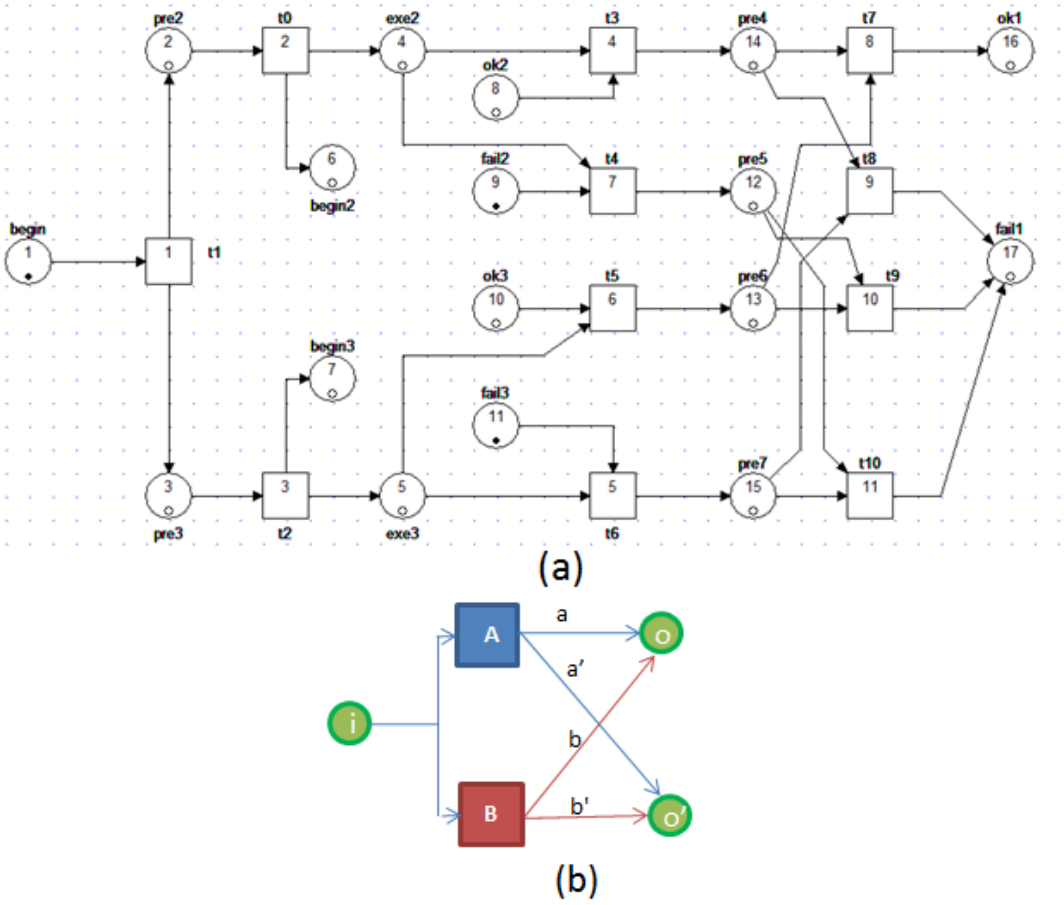


Figura 5.12: (a) Modelo uma estrutura paralela de duas PNBB da Figura 5.1 em sequencia com a ação primitiva 5.10. (b) Modelo esquemático da função.

5.3.2 Outras Estruturas

As combinações de duas ou mais PNBB resulta em vários outros modelos estruturais que permitem elaborar os mais diversos tipos de controle para os SCMs. A seguir será mostrado alguma dessas estruturas:

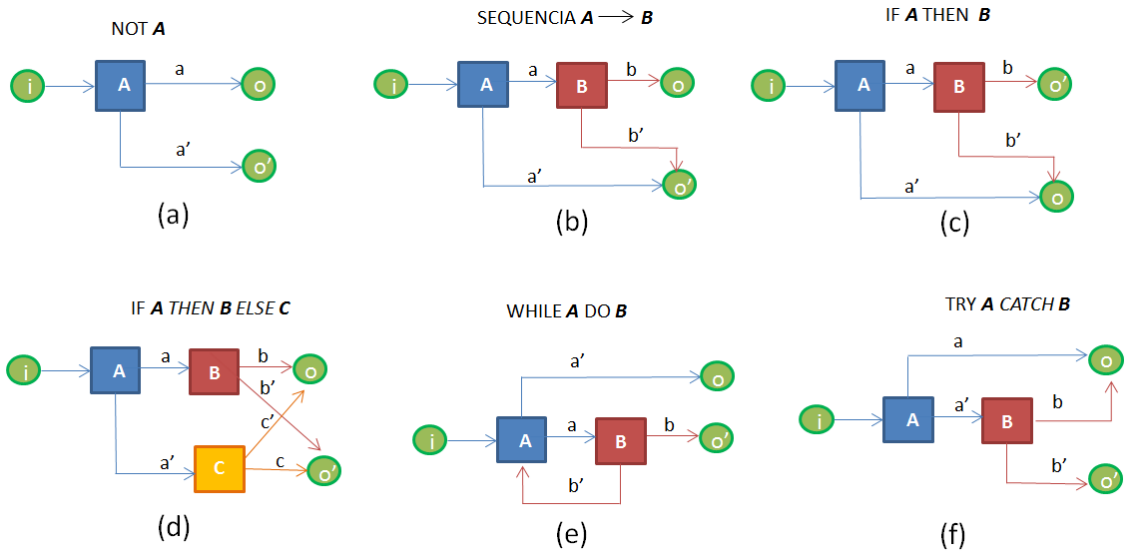


Figura 5.13: Modelos das estruturas sequencias (não paralelas:) (a) not, (b) sequência, (c) if then, (d) if then else, (e) while do e (f) try catch

- (a) NOT: Utilizada para "negar" uma saída de um sistema. Caso o sistema gere uma falha, a saída *ok* é marcada e caso a estrutura seja válida, a saída *fail* é marcada.

- (b) SEQUÊNCIA: Utilizada para criar um sistema em "série", ou seja, saída de uma estrutura é a entrada da próxima PNBB. O estado final *ok* é marcado apenas quando ambas PNBBs apresentarem as mesmas saídas. Se uma das PNBBs que constituem a estrutura falhar, o estado final *fail* é marcado.

- (c) IF-THEN: Executa a PNBB A dentro da rotina if e se a saída válida for marcada, a rede executa PNBB B. Em caso de falha nas saídas de A ou B, o lugar *ok* é marcado. Para qualquer outra sequência o estado *fail* é marcado.

- (d) IF-THEN-ELSE: Executa a PNBB A dentro da rotina if e se a saída válida de A for marcada o sistema executa PNBB B. Caso a saída de B for válida o lugar *ok* é marcado. Caso contrário, o lugar *fail* é marcado. Em situações de falha de saída do bloco A a rotina *then* executa C, marcando o lugar *ok* ou *fail*, em caso de saídas válidas ou inválidas respectivamente.

- (e) WHILE-DO: Esse bloco executa a PNBB A , e caso a saída for válida, executa PNBB B. Enquanto a saída de B não for válida essa rotina não é

concluída. Essa instrução é finalizada quando a saída do bloco B for marcado em *ok*.

- (f) TRY-CATCH: Essa função é similar ao bloco IF $NOT A$ THEN B relatado na Figura 5.13 (a), onde a saída da PNBB A válida, marca diretamente o lugar *ok*. Caso o saída do bloco A não for válida, o PNBB executa a sequência B e se a resposta for válida, a saída *ok* também é marcada. Porém se a saída do bloco B não for válida, a saída *fail* é marcada.

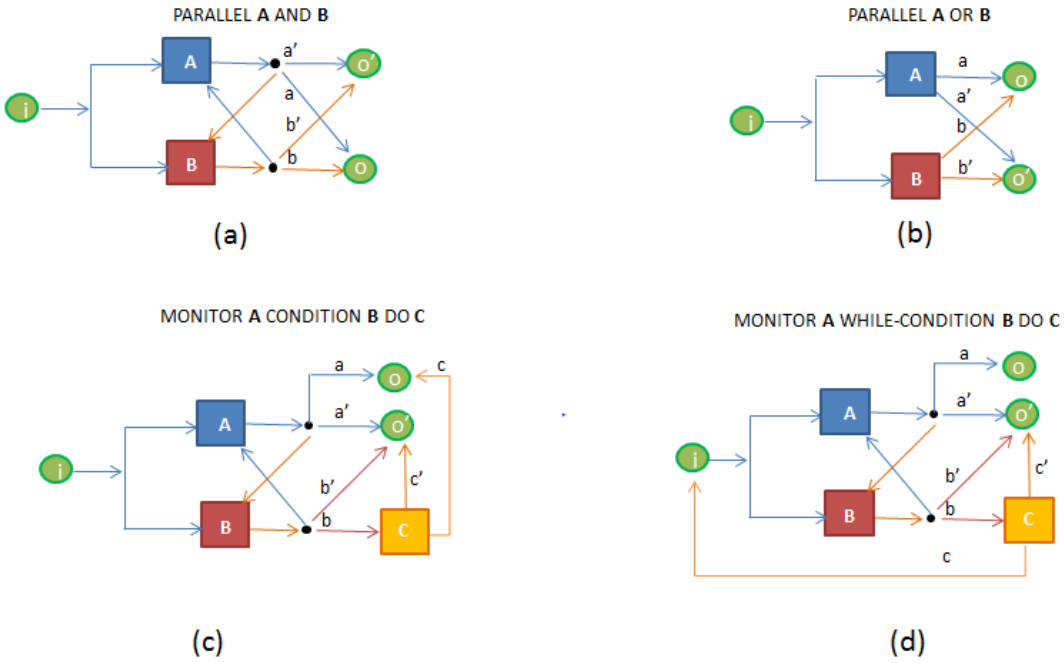


Figura 5.14: Modelos das estruturas paralelas: (a) parallel-and, (b) parallel-or (c) monitor-condition-do (d) monitor-while-condition-do

- (a) PARALLEL-AND: Executa dois blocos de PNBB A e B de forma paralela, onde as saídas de ambos os blocos são independentes. O PNBB que atingir o valor válido e marcar o lugar *ok*, toda a estrutura finaliza com uma *ok*. Caso contrário, se o lugar *fail* for marcado é necessário que ambos os lugares das PNBBs A e B sejam marcados.

- (b) PARALLEL-OR: Utilizado para modelar um sistema, cujo o estado de conclusão do bloco de qualquer estrutura da PNBB finaliza o bloco, tanto para saídas válidas *ok* ou quanto para saídas inválidas *fail*.

- (c) MONITOR-CONDITION-DO: Monitora as condições do sistema, verificando as redes A e B. Se a rede que iniciou seu funcionamento primeiro terminar por último, o sinal de saída é abortado em *fail*. Caso contrário, o monitoramento da rede habilita a execução do bloco C.

- (d) MONITOR-CONDION-DO-WHILE: Funcionamento idêntico ao bloco acima, porém, se saída de C se for válida, a rede executa a estrutura novamente.

5.4 Considerações finais

Neste presente capítulo é possível verificar que as redes de Petri agrupadas através de regras pré-definidas de forma coordenada tornam uma poderosa ferramenta de programação e de supervisão de dados. Esse formalismo modela as *ações primitivas* em redes PNBBs de forma sequencial, descrevendo quaisquer tipo de missões para veículos submarinos em sua camada deliberativa. No estudo de (PALOMERAS et al., 2009), por tratar-se de uma aplicação real e com um grau mais elevado de complexibilidade, usou-se uma linguagem alternativa para executar o plano de missão e compilá-lo automaticamente para as redes de Petri. Essa solução ajuda a detectar possíveis problemas de inconsistências e otimiza o diagnóstico das propriedades da rede resultante. Porém, como o estudo desta dissertação é feito em ambiente virtual e totalmente controlado, é desnecessário criar essa linguagem de programação. A formatação da PNBB pode ser facilmente elaborada seguindo as recomendações exigidas pelos axiomas da tanto dos teoria dos sistemas à eventos discretos quanto da teoria das PNBBs descritas neste capítulo.

A implementação de lógicas sequenciais, paralelas e condicionais amplia bastante uso das redes PNBBs dentro da proposta de elaboração de trajetórias de uma arquitetura de robótica móvel. No próximo capítulo será abordado todos os detalhes das configuração, execução e os testes de validação dessa estrutura SCM para um AUV em ambiente virtual.

-

Capítulo 6

Inspeção de Dutos Submarinos

6.1 Introdução

Apesar de sua grande importância *off-shore*, os dutos marinhos representam também um grande risco à produção e ao meio ambiente. Vazamentos de óleo e explosões de tubulações de gás são alguns tipos de acidente que podem ocorrer em casos de falhas estruturais. Além dos riscos ambientais, acidentes desta proporção, trazem grandes prejuízos financeiros às companhias petrolíferas. A punição aos responsáveis é elevada tanto pelo aspecto financeiro quanto pelos impactos negativos gerados por transtornos ambientais (P HOPKINS, 1995).

Para evitar eventuais acidentes e manter a operação segura é necessário realizar uma rotina manutenção preventiva. As inspeções são feitas através dos *Pipeway Inspections Gate* (PIG) que são ferramentas instrumentadas (equipadas com ultrasom, sensores infra-vermelho, entre outros) que percorrem a tubulação junto com o fluido de processo (gás ou óleo), fazendo a aquisição de dados para análise estrutural do equipamento (B. et al., 2005).

Quando não é possível realizar as inspeções internas devido as limitações geométricas da linha, a inspeção externa é o recurso adotado e apresenta bons resultados. Os ROVs, equipados com câmeras de alta definição, é a ferramenta utilizada para gerar as imagens externas do duto. Em certas momentos imagens pontuais são feitas, já em outros casos, a solução é a inspeção completa da linha. O foco é sempre obter imagens que garantam uma boa análise da integridade física do duto. Apesar disso, os altos custos operacionais, justificados pela logística onerosa do ROV, criam desvantagens pelo uso desse método.

O objetivo de construir uma ferramenta que seja capaz de operar sem estar conectado fisicamente a uma base de operação é o melhor caminho para permitir que as inspeções visuais dos tudo de óleo e gás. Motivado por essa demanda, o foco principal deste capítulo é apresentar um modelo de sistema de controle de missões

para AUVs, empregado em inspeções e monitoramento de tubulações de óleo e gás. Será definido um SCM teórico em PNBB capaz de realizar missões de forma totalmente autônoma em condições de mar bem próximas das reais. Apesar tratar-se de um simulador de AUV, por questões óbvias, o modelo apresentado limita-se a abordar a construção do SCM. Estudos de quaisquer algoritmos para localização e rastreamento de dutos e cabos, é um objeto secundário nesta pesquisa. Há no ambiente virtual, formas para avaliar a posição do veículo sem a necessidade de simular a instrumentação. Apesar disso, caso haja interesse em aprofundar nas técnicas de localização e rastreamento de dutos, pode-se verificar mais detalhes em (PALOMERAS et al., 2012), (EVANS et al., 2003), (ASAKAWA et al., 2000), (BALASURIYA; URA, 2002) onde são destacadas aplicações reais através de técnicas magnéticas, visuais e sonares.

6.2 Planejamento

Segundo (TURNER, 1995) a diferenças entre plano e planejamento está na eficiência. Em situações de emergência o planejamento se torna indispensável através de um conjunto ações (administração recursos e criação cenários) que visa preservar a integridade do veículo e o histórico dos dados coletados.

Para missões em ambientes desconhecidos ou de longa duração, o planejamento automático é vantajoso. Ele permite o replanejamento da trajetória (adicionando, deletando ou modificando os itens na lista do plano de missão), mesmo durante sua execução. Entretanto, o desenvolvimento desta esta técnica é um trabalho desafiador. Sua estrutura cresce exponencialmente com o número de alternativas oferecidas. Essa técnica é bastante estudada em (TURNER, 1995) e (RAJAN et al., 2007).

Outro método de planejamento existente passa todo por uma refinada pesquisa, onde o programador é responsável por todo planejamento da missão, criando uma listagem com todos os possíveis eventos que potencialmente poderão ocorrer. O DSLs e os *scripts* são linguagens utilizadas para elaborar esse tipo de missão.

Não é o foco desta tese abordar técnicas de planejamento automático. Assim, modelos simples são adotados para mostrar como se faz a interface entre o SCM em um modelo virtual através do formalismo das PNBB descritos em (PALOMERAS et al., 2012). A estrutura resultante feita neste trabalho pode ser vista na Figura 6.1.

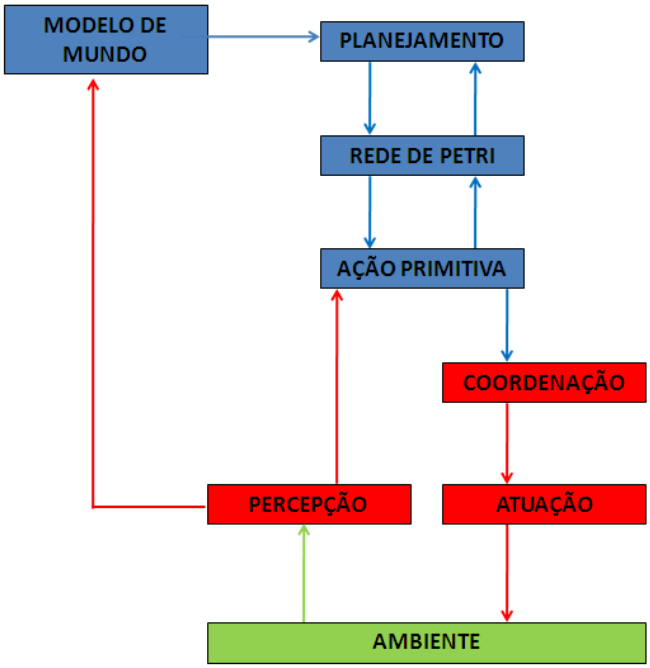


Figura 6.1: Fluxo de dados realizado dentro da estrutura para a realização do planejamento desta missão

6.2.1 Simulador

Esta dissertação utiliza o simulador de veículos submarinos implementado em Matlab/Simulink (GOULART, Junho 2007) baseado nas equações dinâmicas amplamente estudadas por (FOSSEN, 1994). A dinâmica de movimento do AUV é idêntica ao comportamento durante uma operação real, pois foram incluídos modelos contendo algumas propriedades físicas, tais como: forças inercias, forças hidrodinâmicas, forças gravitacionais etc. Além do modo 3D de visualização da missão o software também disponibiliza: vista lateral (Figura 6.2), vista embarcada (Figura 6.3) e vista superior (Figura 6.4). O duto de inspeção tem cerca de 100 metros de comprimento e 2 metros de diâmetro, desenvolvido dentro do ambiente da realidade virtual disponível no Simulink.

Netlab

O sistema de controle de missões é representado por um modelo em redes de Petri que gerencia os eventos à medida que as ocorrências no sistema são atualizadas. A evolução do sistema é definida pelas mudanças de estados do projeto implementado. O Netlab foi a ferramenta utilizada para desenvolver a rede de Petri nesta dissertação. Esse software possui uma interface para Windows que permite ao usuário desenvolver projetos e simulações gráfica em Redes de Petri de forma simples e intuitiva. Outra vantagem é que o Netlab tem uma interface com Matlab/Simulink

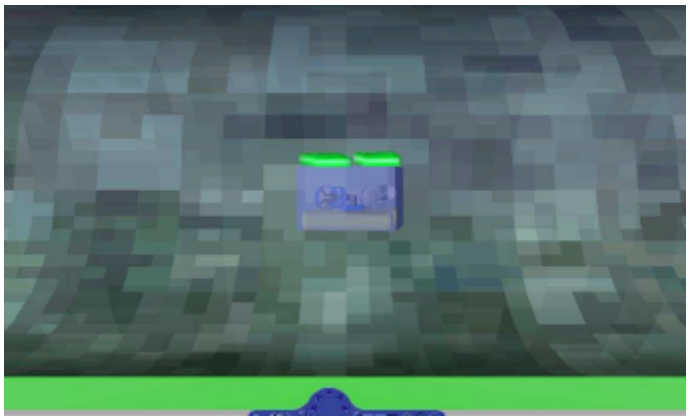


Figura 6.2: Vista lateral do simulador

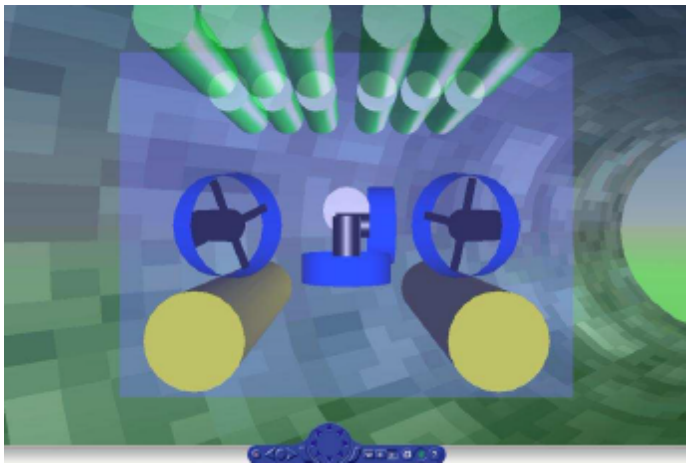


Figura 6.3: Vista embarcada do simulador

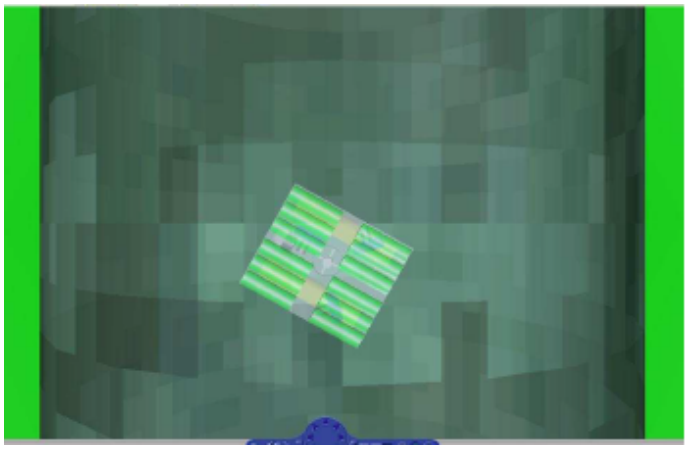


Figura 6.4: Vista superior do simulador

possibilitando a importação dos arquivos criados no Netlab para dentro do Simulink em blocos estruturados.

Nessa relação entre Netlab e Simulink as redes de Petri podem ser estendido com entradas e saídas de lugares. Se um local for definido como uma entrada, sua marcação é determinada pelo Simulink e se ele for definido como uma saída, a marcação é transferido para Simulink. A comunicação entre ambos os software é em tempo real, permitindo a criação de uma série de testes para validar a rede implementada. As Figuras 6.5 e 6.6 representam os elementos no Netlab e no Simulink respectivamente.

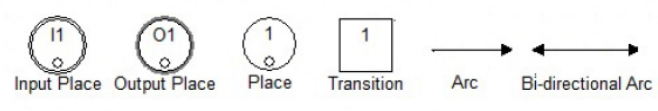


Figura 6.5: Biblioteca dos elementos do Netlab

Figura 6.6: Biblioteca dos elementos de interface entre Netlab e Simulink

Simulink

O ambiente de programação do Simulink é predominantemente construído por diagramas de blocos que descrevem as funções do sistema. Toda a implementação, modelos matemáticos e interfaces podem ser vistas com mais detalhe em (GOULART, Junho 2007). Mudanças estruturais significativa entre o trabalho de (GOULART, Junho 2007) e essa dissertação é mostrada nas Figuras 6.7 e 6.8. Pode-se observar que o Joystick foi removido (Figura 6.8) e em seu lugar foi inserido blocos referentes as RP responsáveis pelo planejamento das trajetórias. Nas próximas seções serão discutidos isoladamente todas essas estruturas, apresentando os resultados obtidos do sistema implementado.

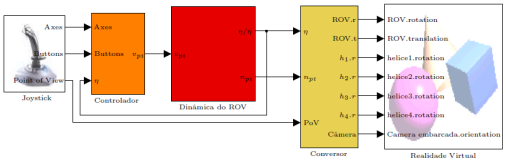


Figura 6.7: Figura do arranjo dos blocos do simulador do trabalho (GOULART, Junho 2007)

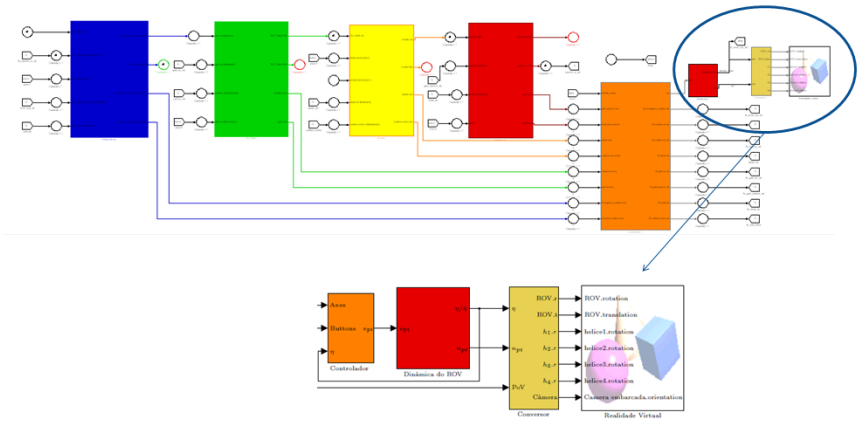


Figura 6.8: Figura do novo arranjo dos blocos do simulador de SCM para este trabalho, comparando a estrutura antes e depois das modificações

6.2.2 Estruturas

O plano de missão descrito sobre linguagem formal pelas redes de Petri exerce uma relação lógica, possibilitando a criação de rotinas e sob-rotinas de programação. Neste contexto, quatro estruturas elementares foram desenvolvidas para descrever o planejamento. O arranjo dessas estruturas é construído de acordo com as regras da PNBBs respeitando os critérios de hierarquia descritas no capitulo anterior. Portanto, as estruturas básicas escolhidas para compor o SCM são: *monitor_condition_do*, *try_catch*, *monitor_condition_while_do* e *parallel_or*.

monitor_condition_do

É a estrutura principal que inicia a missão e ao mesmo tempo monitora a presença de alarmes. Caso haja a detecção de algum, ela imediatamente desmobiliza todas as outras estruturas inferiores e inicia o plano de emergência trazendo o AUV para a superfície. Na Figura 6.8 pode-se verificar essa estrutura modelada em Simulink no bloco de cor azul, e na Figura 6.9 pode-se observar o modelamento desta estrutura em redes de Petri no Netlab.

try_catch

As correntezas marinhas são fenômenos naturais decorrentes das iteração entre forças gravitacionais dos astros, consequências das ações dos ventos e causada pela variação da temperatura dos oceanos. A presença delas dificulta o deslocamento dos AUVs durante o deslocamento, impondo grandes desafios aos métodos de navegação. As correntezas são caracterizadas por ter elevados módulos de forças na superfícies e baixos módulos de forças nas regiões mais profundas. A rotina *try_catch* é usada para a minimizar esse efeito. Quando for habilitada pela rotina superior *monitor_condition_do*, o bloco *try* busca seu objetivo (deslocar até o ponto

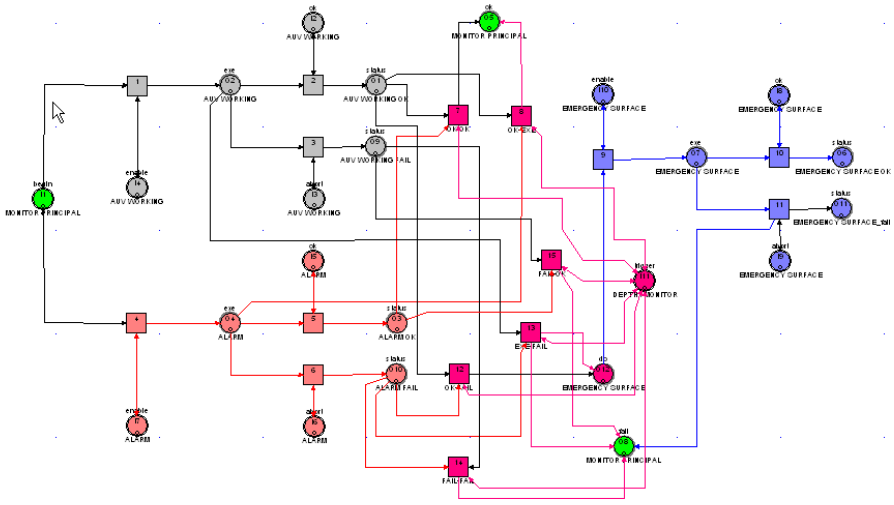


Figura 6.9: Estrutura da programação *monitor_condition_do* modelada em redes de Petri no Netlab

desejado na direção do eixo x). Esse deslocamento é temporizado, ou seja, existe um determinado tempo para cumprir essa tarefa (o comportamento desta trajetória pode ser visto na Figura 6.15). Alcançando esse ponto dentro do tempo estabelecido, o bloco é finalizado e a estrutura inferior é habilitada sequencialmente. A presença de correnteza neste cenário é caracterizada pelo não cumprimento desta rotina dentro do tempo pré-estabelecido. Assim, o bloco *catch* é habilitado e cancela o movimento na direção do eixo x indo ao ponto pré-programado (no eixo z). Esse comportamento impede que haja um consumo excessivo de baterias evitando que comprometa o andamento da missão.

Na Figura 6.8 pode-se verificar essa estrutura modelada no Simulink no bloco de cor verde, e na Figura 6.9 pode-se observar o modelamento desta estrutura em redes de Petri no Netlab.

monitor_condition_while_do

Tem o funcionamento idêntico ao bloco *monitor_condition_do* adicionando a sub-rotina de *loop* que permite a recuperação de dados. Esse bloco é habilitado e permite ao AUV ir até o ponto pré-determinado. Ao longo do deslocamento, o bloco monitora os dados dos instrumentos de navegação. Se as posições estiverem fora dos limites estabelecidos, o AUV retorna a posição configurada e retoma a trajetória. Esse bloco é constituído por uma rotina *while* e pode entrar em *loop* quantas vezes for necessário até atingir a profundidade programada, sem que nenhum erro de posição seja gerado.

Na Figura 6.8 pode-se verificar a estrutura modelada no Simulink no bloco de cor amarela, e na Figura 6.9 pode-se observar o modelamento da estrutura em redes

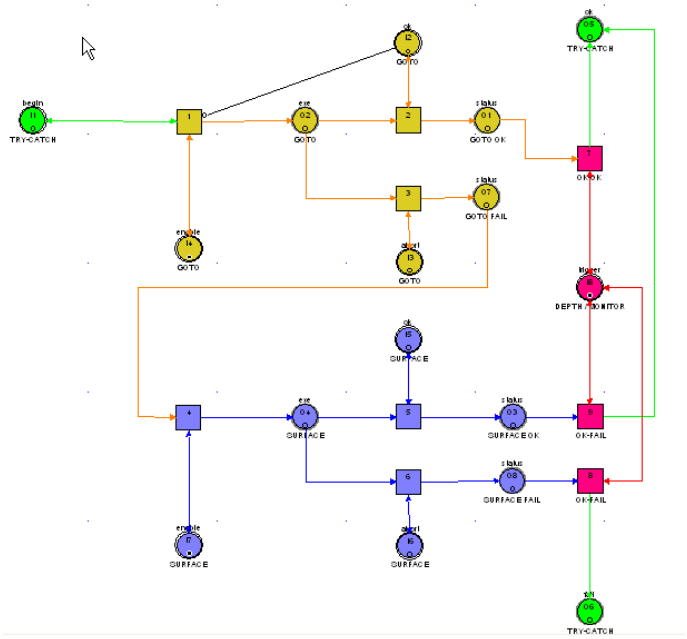


Figura 6.10: Estrutura da programação *try_catch* modelada em redes de Petri no Netlab

de Petri no Netlab.

parallel_or

A estrutura *parallel_or* propicia ao veículo entrar em modo inspeção, baseado em se posicionar no ângulo ψ, capturar as imagens e seguir o duto. Essas ações são feitas de maneira coordenada e paralela. Após o término desta rotina o AUV retorna a posição inicial.

Na Figura 6.8 pode-se verificar essa estrutura modelada no Simulink no bloco de cor vermelha e na Figura 6.12 pode-se observar o modelamento desta estrutura em redes de Petri no Netlab.

Estrutura completa

As quatro rotinas descritas anteriormente estão interligadas para compor o SCM. O resultado destas conexões podem ser vistas na Figura 6.13 a seguir.

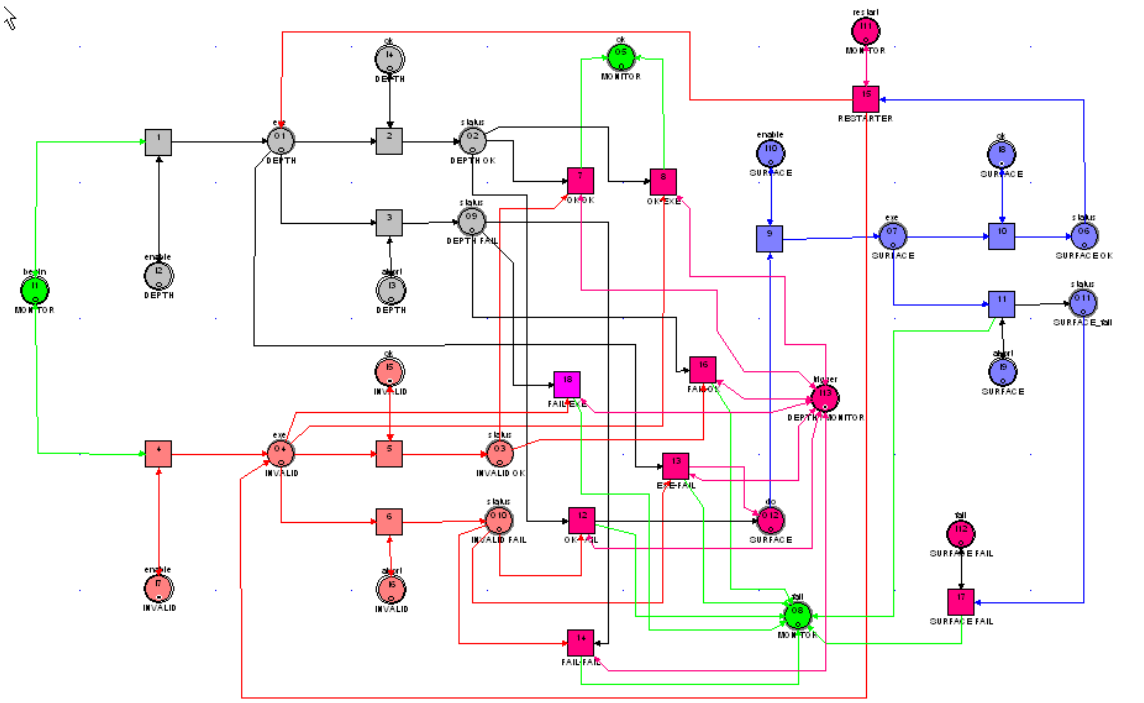


Figura 6.11: Estrutura da programação *monitor_condition_while_do* modelada em redes de Petri no Netlab

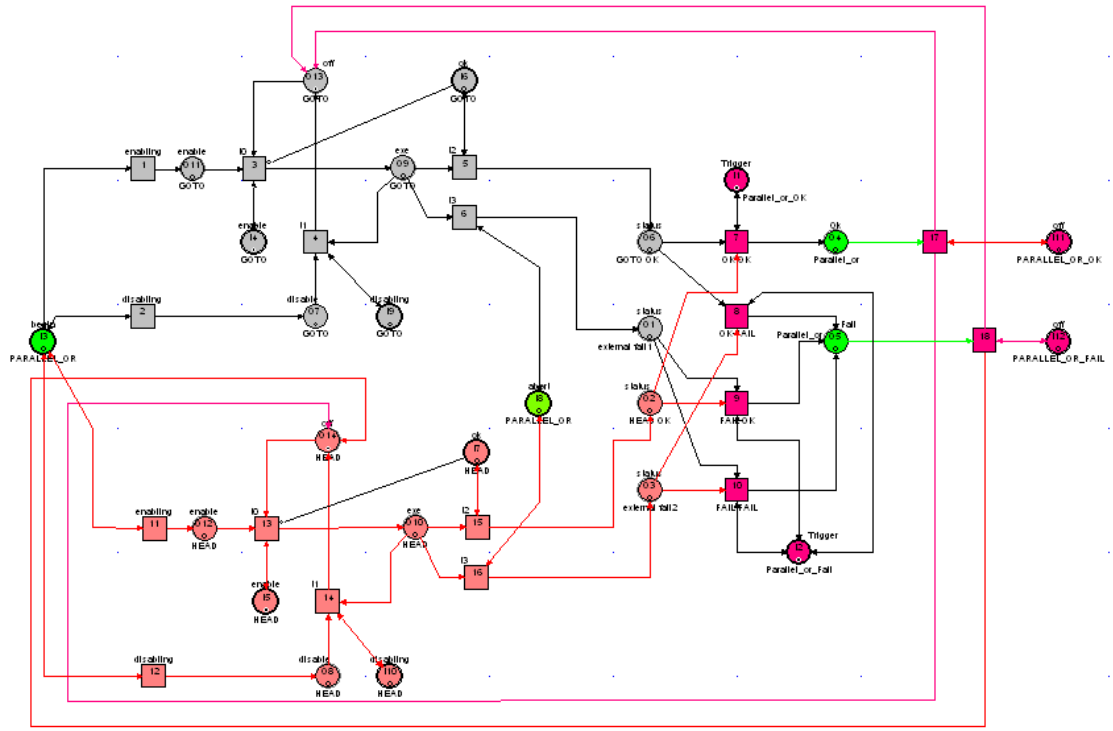


Figura 6.12: Estrutura da programação *parallel_or* modelada em redes de Petri no Netlab

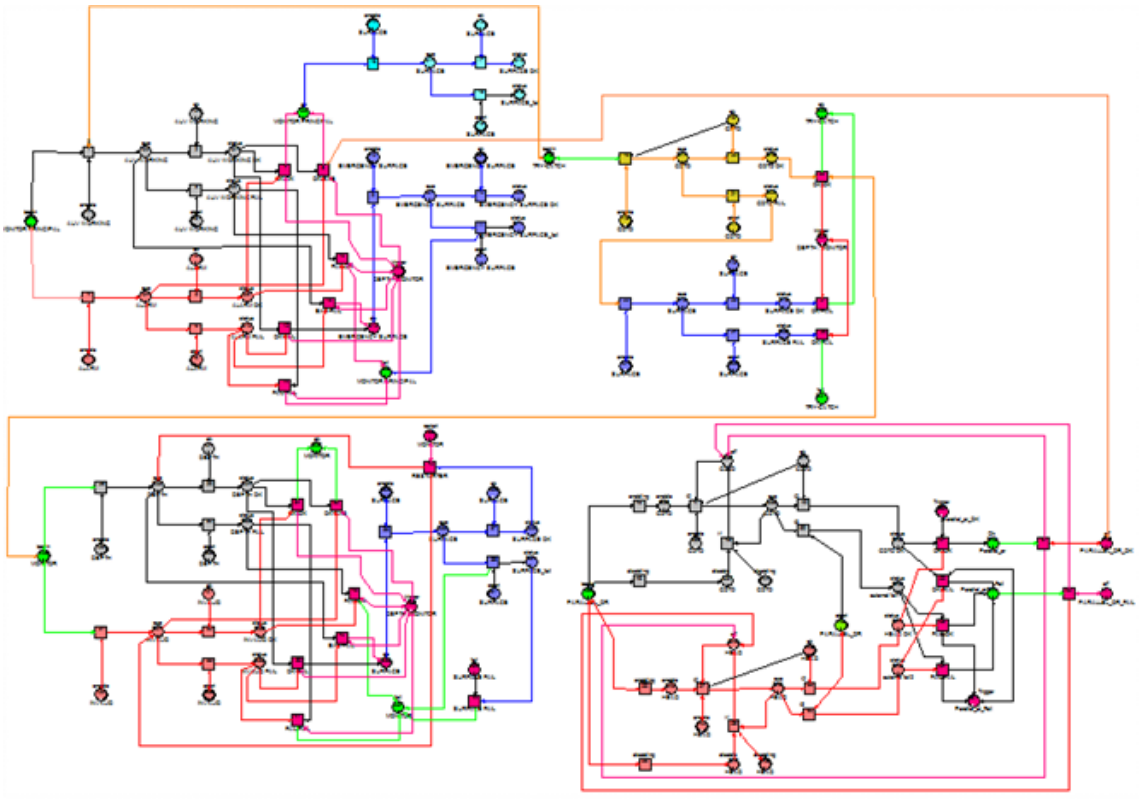


Figura 6.13: Modelo completo das quatros estruturas modeladas em redes de Petri no Netlab

6.3 Organização da Missão

Essa seção descreve as principais *ações primitivas* desenvolvidas no SCM, com a finalidade de planejar uma missão virtual de inspeção em dutos submarinos. Cada uma dessas ações são executadas sequencialmente, obedecendo à subordinação dentro de um modelo pré-definido descrito no capítulo anterior. A construção e a estrutura de todas elas, também apresentada no Capítulo 4, seguem o padrão das PNBBs.

As *ações primitivas* são divididas em dois grupos. O primeiro é caracterizado pelo conjunto de instruções que podem variar o movimento do veículo ao longo da missão, sendo elas:

- *heading*: estabelece o ângulo desejado na direção x mantendo essa orientação durante o seu deslocamento. Um controle de PID é utilizado minimizar o erro causado pelas interferências externas.
- *depth*: estabelece a profundidade desejada e um controle de PID é utilizado manter o AUV neste posicionamento.
- *surface*: o comando leva o AUV a superfície quando a missão é finalizada.

- *emergencySurface*: leva o AUV para superfície quando for diagnosticado alguma situação emergencial que coloque em risco a operação ou a integridade do veículo.

- *goto*: é a instrução utilizada para levar o AUV ao ponto desejado. A navegação nesta condição é feita em duas dimensões (x) e (y).

Já o segundo grupo de *ações primitivas* é caracterizado pelo conjunto de instruções que não interferem no movimento do veículo, mas são importantes para habilitar/desabilitar alarmes, eventos ou processos. Nesta lista estão:

- *initializeVehicle*: executa uma varredura de todos os sensores e atuadores. Se todos eles estiverem funcionando perfeitamente a missão prossegue. Caso contrário, há o aborto e um estado de falha é gerado.

- *stopVehicle*: finaliza a aquisição de dados e para o AUV.

- *alarm*: verifica o estado de todos os elementos do AUV, bem como: autonomia da bateria, temperatura e umidade dos vasos eletrônicos, corrente de alimentação, etc. Um alarme é gerado caso um desses elementos estejam fora dos valores especificados.

A Figura 6.14 indica como é estruturado essas *ações primitivas* instruções dentro da semântica de programação.

6.3.1 Descrição da Missão

Para a construção do SCM desta dissertação, o paradigma adotado foi baseado nas técnicas *off-bording*, ou seja, as etapas das missões são pré-definidas e o sucesso depende da capacidade do programador, usando o formalismo das redes de Petri.

Por tanto, a missão virtual é iniciada na posição $(0, 0, 0, 0)$ referente as orientações de x, y, z, ψ, no ambiente de realidade virtual do Simulink (GOULART, Junho 2007). Pela configuração do cenário virtual, representado na Figura 6.15, o veículo já se encontra submersos em água e pronto para iniciar. Através do comando "iniciar missão", a rotina principal *Monitor_Condition_Do* é habilitada e imediatamente inicia e a rotina hierarquicamente inferior. Na sequência, a *ação primitiva goto* movimenta o veículo na posição x, dentro do bloco *try_catch*, até o valor pré-programado. Durante essa ação, as posição y e ψ, são monitoradas e devem permanecer dentro de limites estabelecidos. Toda o planejamento é executado dentro da estrutura *try*, e caso os eixos de y e ψ alcance valores inesperados nesta sub-rotina, os parâmetros pré-programados dentro do bloco *catch* são assumidos,

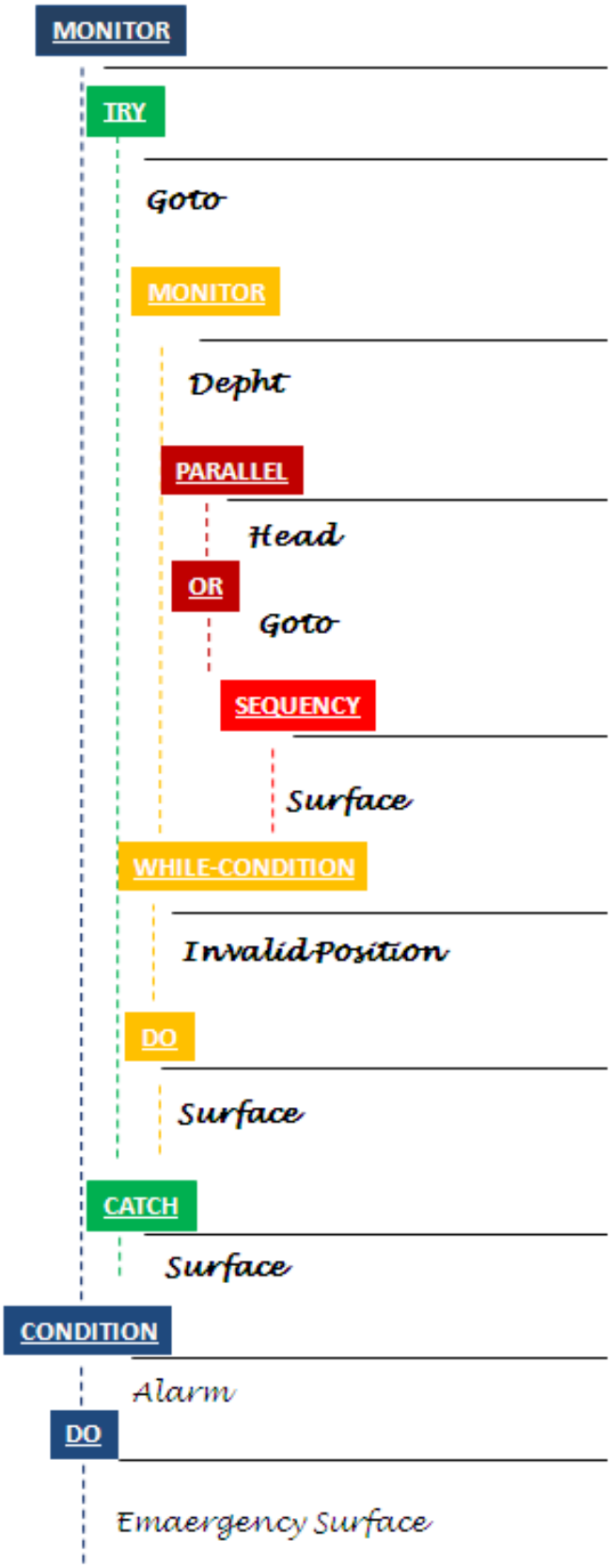


Figura 6.14: Estrutura da programação do PNBB

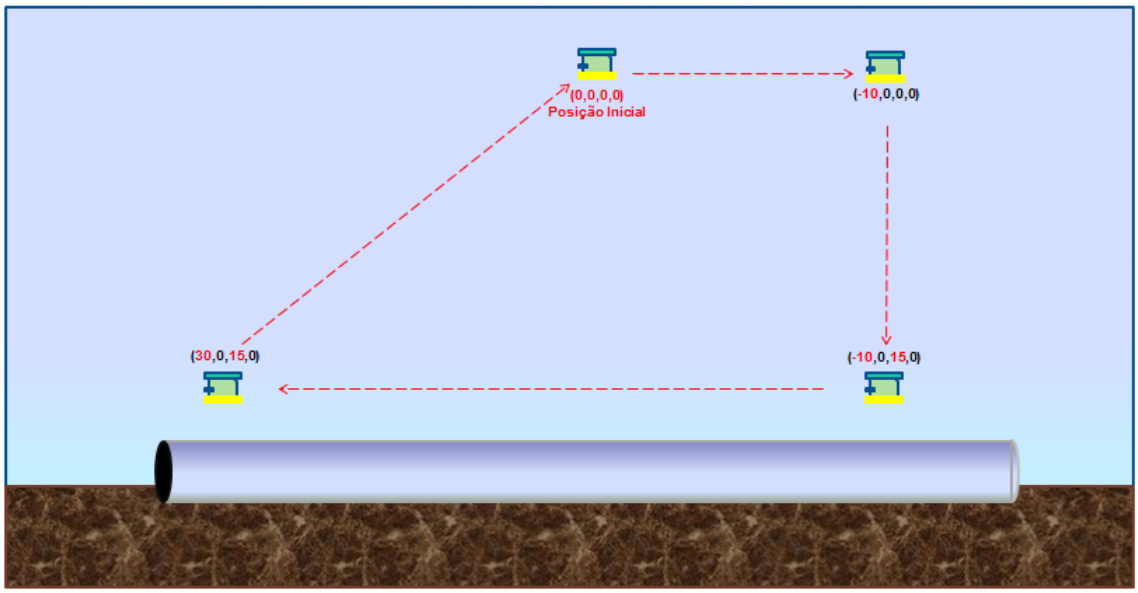


Figura 6.15: *Layout* da missão no ambiente virtual

levando o AUV imediatamente à próxima rotina. A estrutura seguinte, a *Monitor_Condition_While_Do*, realiza as *ação primitiva*, *depth*, verificando as referências de posições. Se as posições estiverem fora dos limites estabelecidos, o AUV retorna a posição final da sub-rotina anterior e retoma a trajetória. Esse bloco atua a rotina *while*, e pode entrar em *loop* quantas vezes for necessário até atingir a profundidade estabelecida, sem que nenhum erro de posição seja gerado. A estrutura seguinte, *parallel_or*, permite que as *ações primitivas*, *goto*, *head*, *takeImage* posicione o AUV e execute a inspeção de forma sequencial e paralela. Logo depois do duto inspecionado, a missão primitiva *Surface* é habilitada, levando AUV para a posição inicial.

Durante toda a execução das sub-rotinas a estrutura *Monitor_Condition_Do*, monitora a presença de alarmes no sistema, e caso haja alguma ocorrência, o AUV retorna à superfície ativando a *ação primitiva EmergencySurface*. O diagrama que detalha toda esta missão pode ser visto na Figura 6.22

6.3.2 Simulações

Foram realizados baterias de simulações que possibilitaram a representação do movimento do veículo em diversas situações, com o objetivo de validar o SCM estruturado em PNBB. Os resultados obtidos estão apresentados a seguir. A linha vermelha indica o *setpoint* e a linha azul indica o sinal real. Pode-se observar que o AUV parte do ponto x, y, z, ψ $(0, 0, 0, 0)$ e no final da missão retorna para a mesma posição, ou seja $(0, 0, 0, 0)$, Figura 6.15. O deslocamento foi limitado apenas nas extensões dos eixos x e z. Os *pre-sets* escolhidos em x (em metros) são: posição inicial em 0, o primeiro *pre-set* em -10, o segundo *pre-set* em 20 e finalmente o terceiro *pre-set* retornando a posição 0. Os *pre-sets* escolhidos em z (em metros) são: posição ini-

cial em 0, primeiro *pre-set* em 15, segundo *pre-set* -1 e o terceiro *pre-set* em 0. A representação gráfica de toda a trajetória é mostrada na Figura 6.26.

Missão Completa Sem Ocorrências

Na Figura 6.17 mostra o deslocamento do AUV para inspeção do duto submarino sem ocorrências durante seu deslocamento. A Figura 6.17 (a) representa o deslocamento do veículo em função do tempo no eixo x, a Figura 6.17 (b) representa o deslocamento do veículo em função do tempo no eixo z e Figura 6.17 (c) representa a variação de eixo ψ em função do tempo. O primeiro trajeto é o deslocamento de -10 na direção x, que ocorre dentro da rotina *Try_Catch*, na sequencia há o movimento em z de 15 na rotina *Monitor_Condition_While_Do*. Esse *pre-set* de 15 em z, permanece até o fim da estrutura *parallel_or*, onde há também o deslocamento longitudinal no eixo x de 20 (nesse momento é que se realiza a inspeção através de câmeras). Por fim, o AUV retorna a posição original $(0, 0, 0, 0)$ concluindo a missão. O diagrama esquemático desta missão também pode se visto na Figura 6.16.

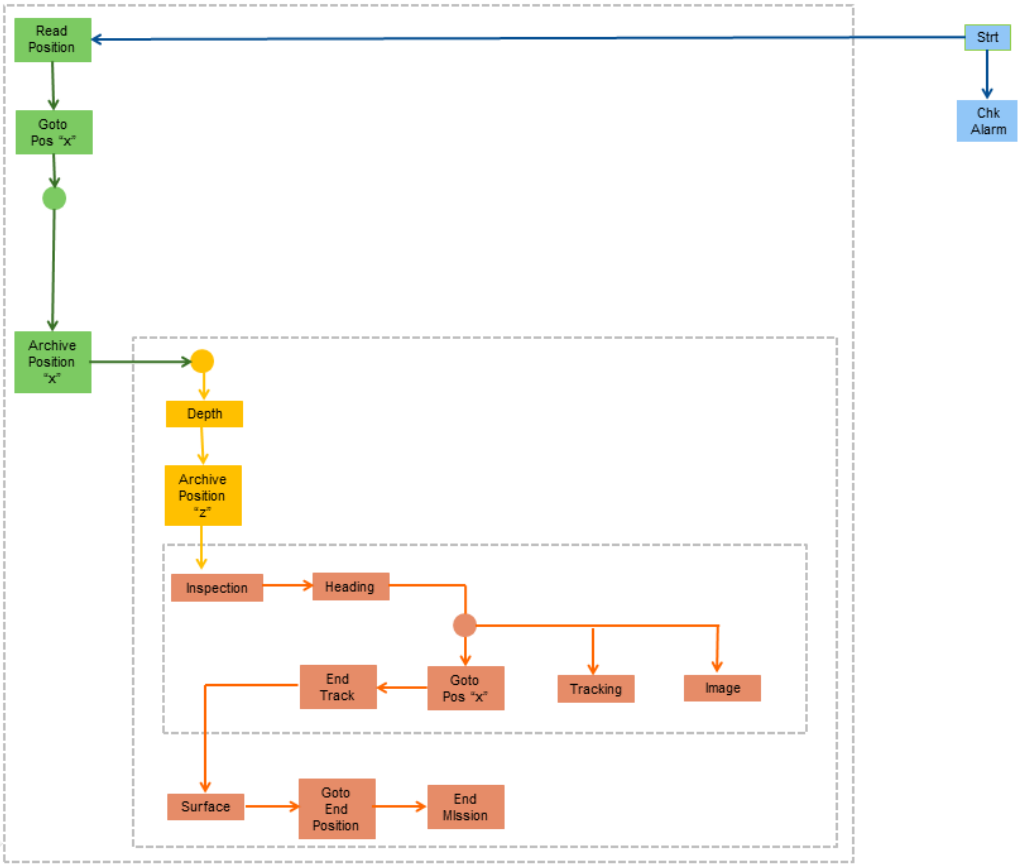


Figura 6.16: Diagrama de uma missão completa sem anomalias

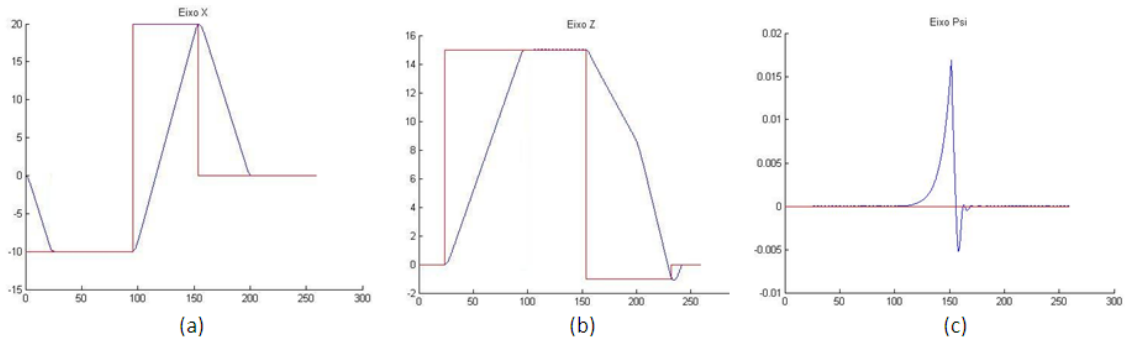


Figura 6.17: Figura referente a uma missão completa sem anomalias

Missão Completa Com Ocorrência na Estrutura *Try_Catch*

Na Figura 6.19 mostra o deslocamento do AUV para inspeção do duto submarino com uma ocorrência na estrutura *Try_Catch*. A *ação primitiva goto* move o AUV na direção x, mas neste instante a estrutura detecta um sinal de correnteza excessiva. Portanto há um aborto na ação *goto* e uma habilitação de outra ação na mesma estrutura *surface*, fazendo com que o veículo retorne até uma posição conhecida antes de passar para a próxima estrutura. Esse fato pode ser observado no instante 25 segundos nos eixos x e z nas Figuras 6.19 (a) e (b) respectivamente. A partir deste momento, toda a missão ocorre dentro da normalidade já descrita na seção Missão Completa Sem Ocorrências. O diagrama esquemático desta missão também pode se visto na Figura 6.18.

Missão Completa Com Ocorrência na Estrutura *Monitor_Condition_Do*

Na Figura 6.21 mostra o deslocamento do AUV para inspeção do duto submarino com uma ocorrência na estrutura *Monitor_Condition_Do*. A evolução da missão ocorre como descrito na seção Missão Completa Sem Ocorrências até a PN habilitar a estrutura *Monitor_Condition_Do*. Neste momento a *ação primitiva depth* é monitorada por outra *ação primitiva*, *InvalidPosition*, que ao detectar um erro no sistema aproximadamente no intervalo de tempo $17s$, provoca o retorno do AUV até a posição programada (-10,0,0,0). A partir deste ponto, a missão é restabelecida até seu cumprimento por completo. Nas Figuras 6.21 (a), (b) e (c) pode-se verificar o comportamento do AUV durante todo esse deslocamento. O diagrama esquemático desta missão também pode se visto na Figura 6.20.

Missão Interrompida Com Ocorrência Alarme da Estrutura

Outro cenário analisado foi situações de alarmes motivando pelas emergências. Por exemplo, alta temperatura nos vasos da eletrônica embarcada, mau funcionamento

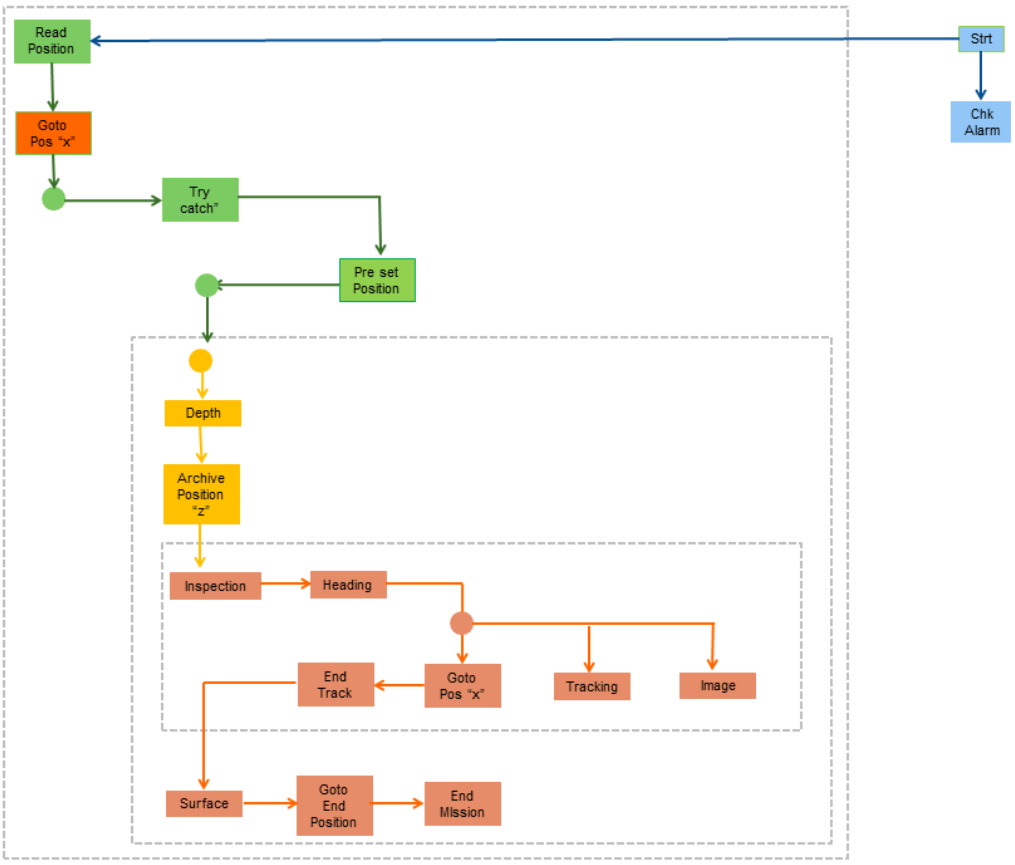


Figura 6.18: Diagrama referente a uma missão completa com ocorrência na estrutura *Try_Catch*

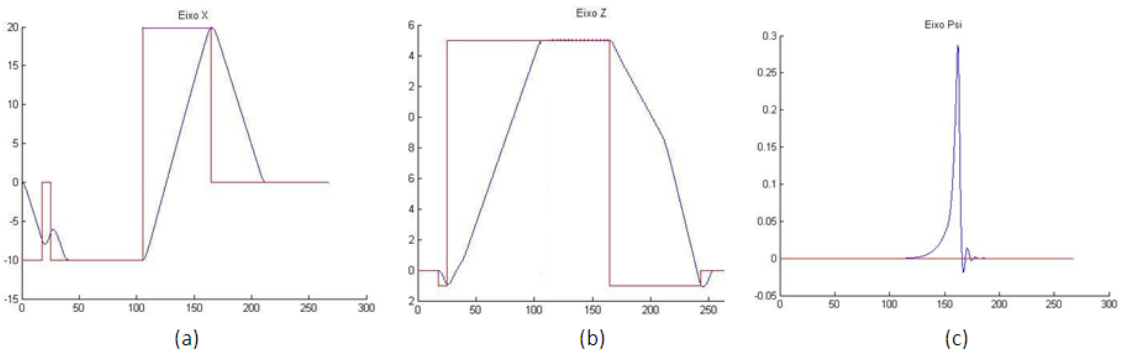


Figura 6.19: Figura referente a uma missão completa com ocorrência na estrutura *Try_Catch*

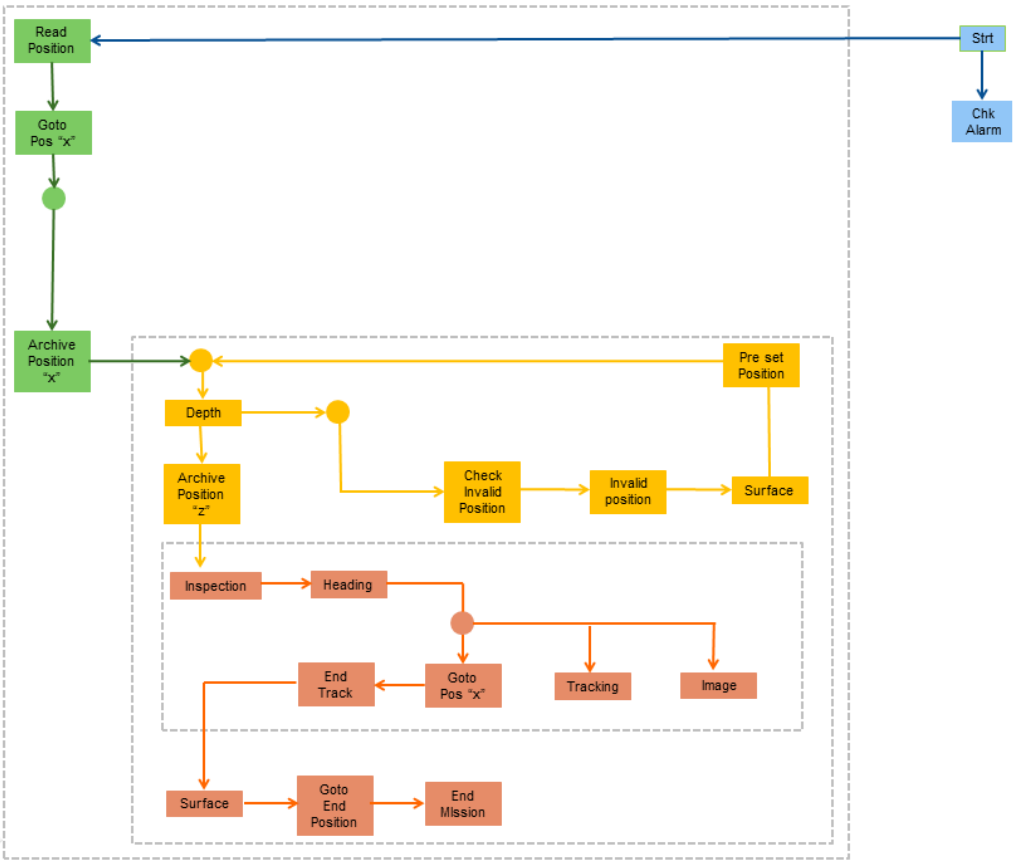


Figura 6.20: Diagrama referente a uma missão completa com ocorrência na estrutura *Monitor_Condition_Do*

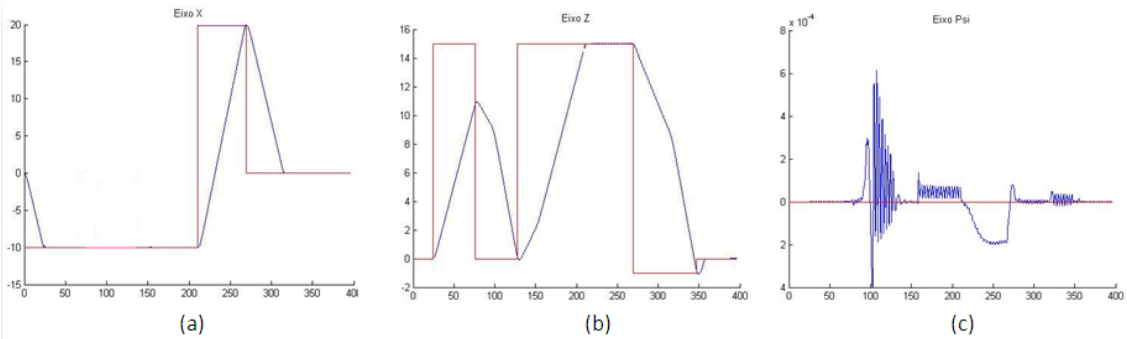


Figura 6.21: Figura referente a uma missão completa com ocorrência na estrutura *Monitor_Condition_Do*

de algum dispositivo, nível baixo de carga acumulado nas baterias entre outros são circunstâncias consideradas graves e podem comprometer a continuidade da missão. O aborto da missão é imediatamente habilitado pela *ação primitiva emergencySurface*, que leva o veículo à posição (0,0,0,0) e é prioritária sobre quaisquer outras *ações primitivas* que estejam em execução no momento. As Figuras 6.23, 6.24 e 6.25 esboçam esse comportamento. O diagrama esquemático desta missão também pode se visto na Figura 6.22.

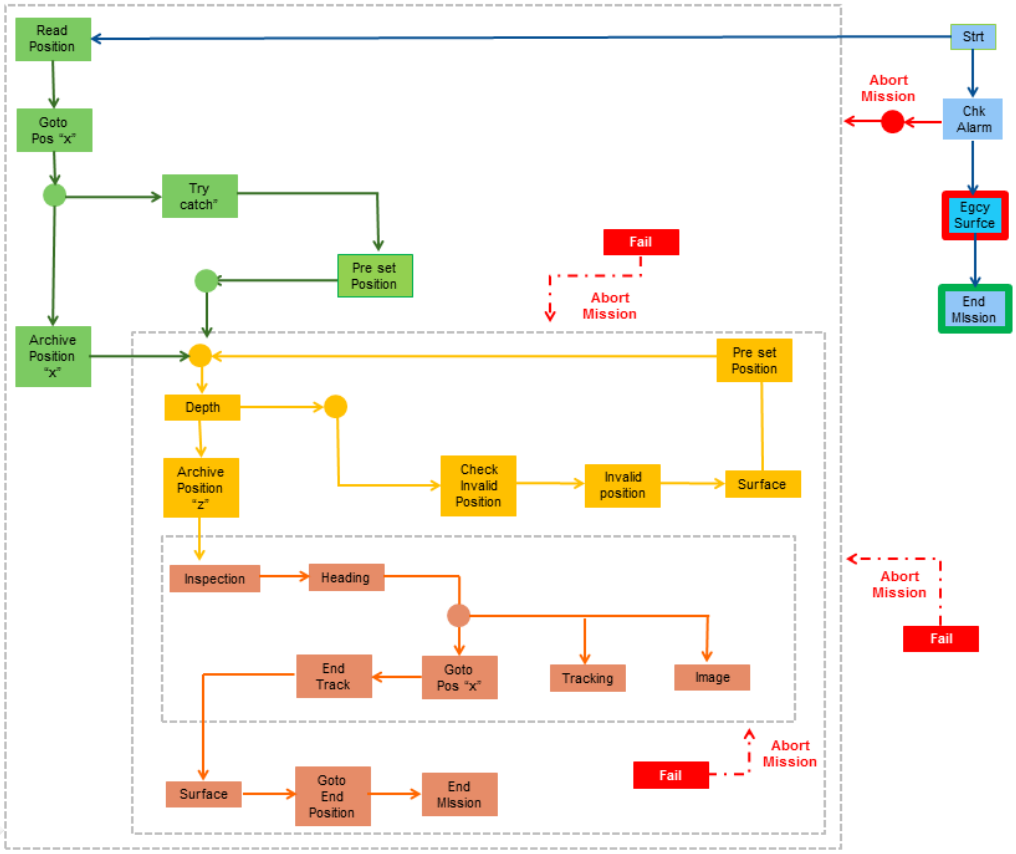


Figura 6.22: Diagrama referente a uma missão completa com ocorrência na estrutura

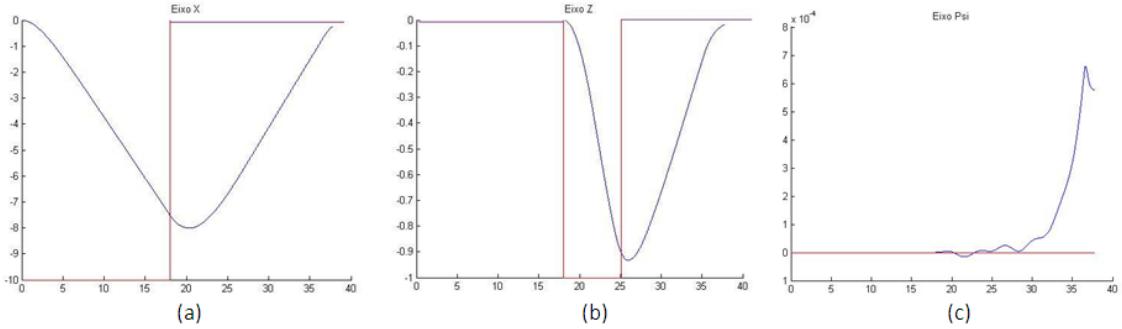


Figura 6.23: Figura de uma missão interrompida por ocorrência alarme na estrutura *Try_Catch*

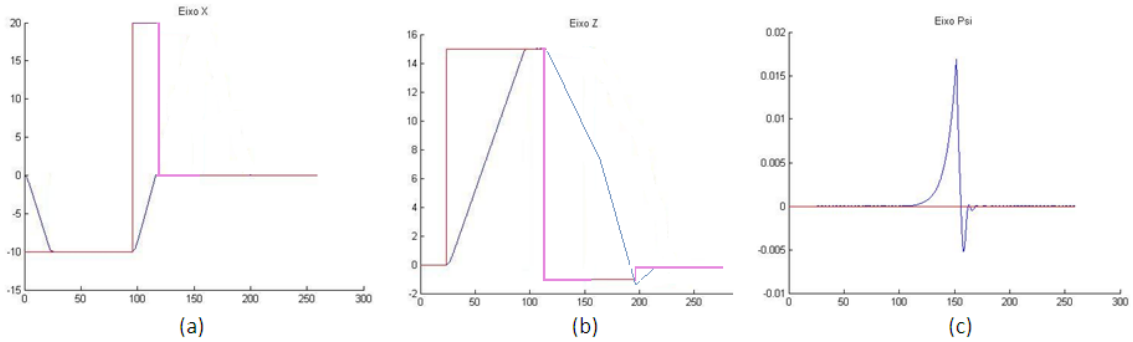


Figura 6.24: Figura de uma missão interrompida por ocorrência alarme na estrutura *Monitor_Condition_Do*

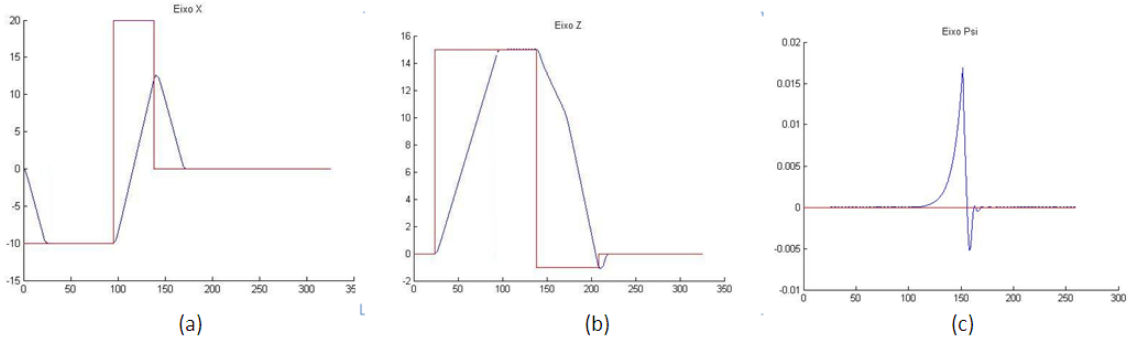


Figura 6.25: Figura de uma missão interrompida por ocorrência alarme na estrutura *Parallel*

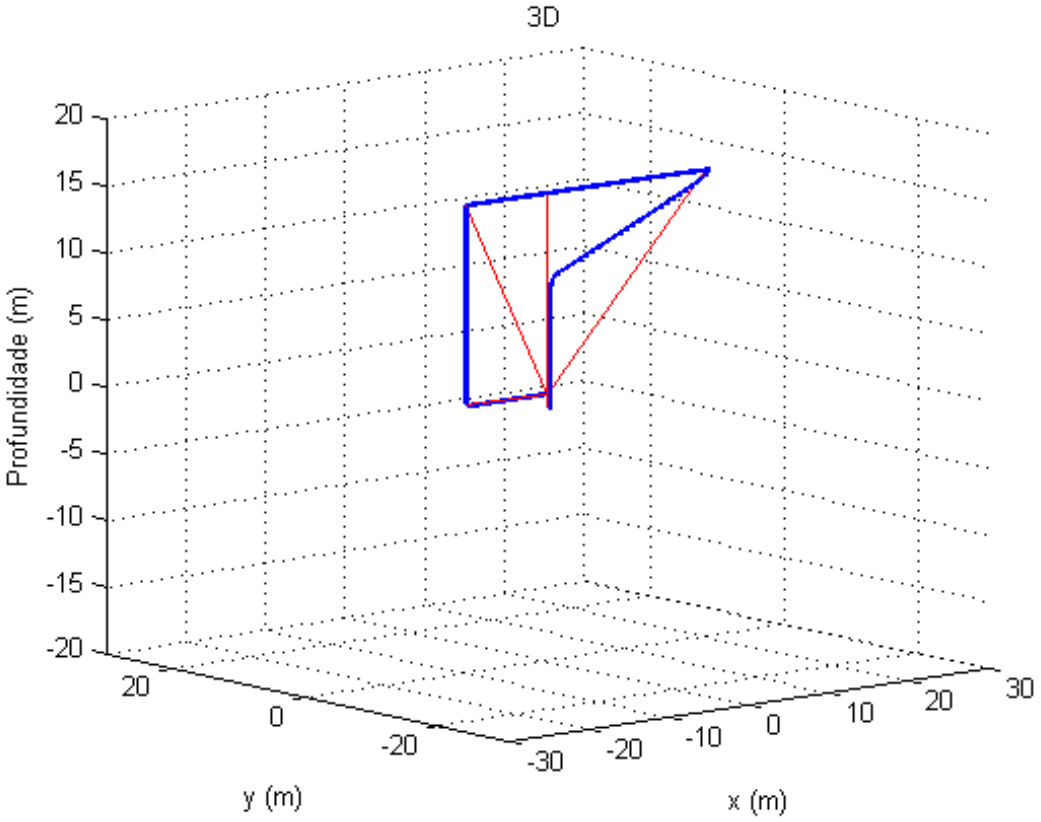


Figura 6.26: Representação 3D de uma missão completa

-

Capítulo 7

Conclusão

Neste capítulo são apresentadas as conclusões gerais obtidas na pesquisa realizada durante o desenvolvimento do Sistema de Controle de Missão para AUVs em realidade virtual. Entretanto, alguns tópicos abordados nesta pesquisa podem ser estudados com mais detalhes em buscando o aperfeiçoamento em trabalhos futuros.

7.1 Resumo

O projeto proposto aborda a construção de um SCM dedicado à plataforma virtual de um AUV, desenvolvida em Matlab. O Capítulo 2 descreve a história e a evolução dos submarinos autônomos apresentando uma breve descrição das principais partes que compõem os AUVs, ressaltando as peculiaridades dos modelos existentes. O Capítulo 3 faz uma revisão da arquitetura de controle e dos SCM mais populares aplicados aos AUVs. O Capítulo 4 aborda a teoria dos SEDs, cujo um dos modelos mais populares que são as Redes de Petri, modela sistemas discretos através dos eventos por regras de transições bem definidas. O Capítulo 5 propõe uma missão baseada nos PNBBs utilizando as *ações primitivas* para gerar o fluxo de informações. Esse modelo apresenta propriedades analíticas que permitem a verificação de algumas características da rede, tais como: alçabilidade, limitada e vivacidade. Outra grande vantagem é a possibilidade da criação de diferentes arranjos de programação, tais como: estruturas sequenciais e paralelas; técnicas condicionais e incondicionais; rotinas de tratamentos de exceção (*Try_Catch*) e rotinas de *loop* (*Monitor_Condition_While_Do*). No Capitulo 6 é feita a simulação virtual pelo pelo do Simulink em conjunto com a toolbox Netlab. O sistema é reproduzido em redes de Petri e tem como função inspecionar dutos submarinos. Os resultados obtidos foram satisfatórios, provando que a versatilidade do modelo PNBB proposto é uma poderosa ferramenta para elaborar planos de missão em redes de Petri para os SCM.

7.2 Conclusões gerais

As funcionalidades das principais *ações primitivas* foram testadas no Matlab/Simulink. O ambiente criado dentro da realidade virtual possibilita a reprodução de alguns eventos recorrentes dentro de uma missão, tais como: erros de comportamento, falhas de tarefas e alarmes. Essas opções permitem estabelecer uma comparação entre as respostas obtidas com as esperadas de um SCM.

Uma missão não é somente bem sucedida quando todos os objetivos são conquistados, mas quando a integridade do veículo e dos dados são preservados. Os resultados dos gráficos corroboram com essa afirmação, sendo que, ao final em todos os cenários simulados a posição de partida foi igual à posição chegada, ou seja (0,0,0,0).

Esse trabalho certifica que é possível elaborar um plano de missão pelas redes de Petri e implementá-las no ROV LUMA. A linguagem formal em alto nível teve um bom desempenho quando submetido as mais variadas situações decorrentes de missão de inspeção de dutos submarinos. Tratamento de interferências de sinais de navegação, rotinas em *loop* e algoritmos de detecção obtiveram resposta satisfatória, comprovando que elas podem ser muito úteis em uma plataforma física. Por outro lado, a elaboração de um SCM para um AUV real em Redes de Petri requer outro software dedicado. O Netlab se mostrou aquém das expectativas, não suportando o número superior a 100 estados. Foi provado que, redes que possuem em sua estrutura de programação o número elementos acima do estabelecido, não apresentam rendimento satisfatório.

7.2.1 Principais aspectos negativos durante a execução do projeto

Limitação de estados da rede no *Netlab*

A *toolbox*, que permite integrar as funcionalidades das redes de Petri no *Simulink*, tem o número de estados limitados em 100 elementos. Esse número baixo de estados dificulta a criação de sistemas mais complexos.

Limitação no número de trajetória

Devido ao problema apresentado no item anterior, a simulação ficou limitada em apenas uma trajetória. A ideia inicial do trabalho era elaborar no mínimo duas diferentes rotas de inspeção, justamente para promover a flexibilidade e a facilidade do uso das PNBBs para esse tipo de situação.

Recursos do Netlab

O Netlab é um software experimental que modela sistemas em redes de Petri. Seu arquivo executável permite a implementação de projetos diretamente do sistema. Entretanto, para que este software funcione em conjunto com o Simulink, a versão do sistema operacional Windows XP tem que estar instalado na máquina de desenvolvimento. No caso desta dissertação foi utilizada uma máquina virtual com o sistema operacional Windows XP para o funcionamento do Netlab e Simulink. Como resultado, as simulações ficaram extremamente lentas, dificultando muito a realização da bateria de testes.

7.2.2 Principais aspectos positivos encontrados durante a execução do projeto

Arranjos sequenciais e paralelos tiveram um ótimo desempenho. Isto certifica o uso dessas estruturas *loop* (*Monitor_Condition_While_Do*), estruturas tipo supervisão (*Monitor_Condition_Do*) e tratamentos de exceção como *Try_Catch*. Todas elas se comportam de acordo a expectativa desejada para a modelagem de um SCM.

Para garantir que funcionamento do sistema de navegação do veículo funcione com um alto grau de confiabilidade, as simulações provaram que é viável vincular o erro de posição a um sinal de alarme. Assim, quando o respectivo alarme é acionado, o AUV retorna a posição de memória evitando o aborto da missão por baixa qualidade de sinal dos instrumentos de navegação. Outra importante consideração observada foi o comportamento da *ação primitiva emergencySurface*. Em situações de emergência ela habilita, de forma atípica, a configuração de potência máxima (em uma situação real, pode-se trabalhar dentro faixa do fator de serviço dos motores) dos *thrusters* do AUV, deixando o veículo ir superfície em tempo reduzido.

A versatilidade encontrada neste modelo das redes de Petri confirma que esse sistema possui uma boa adaptação as mais variadas rotinas de programação, de forma simples e confiável. Em uma análise conclusiva, pode-se dizer que a rede de Petri organizadas em PNBB são uma poderosa ferramenta para construção de sistemas de missão de controle para veículos submarinos.

7.3 Contribuições da tese

Esse trabalho contribui com a pesquisa de modelos de SCM, excepcionalmente para veículos submarinos autônomos, sobre o ponto de vista do usuário e do programador. Várias missões podem ser adaptadas através uma linguagem simples e flexível. Durante o desenvolvimento, algumas contribuições foram atingidas, entre elas estão:

1. *Missão de controle definidas por redes de Petri*: a proposta descreve um modelo de estrutura que permite elaborar um SCM a partir de pequenas redes de Petri. Com o agrupamento destes pequenos modelos é possível descrever, de maneira simples, grandes e variadas estruturas em PNBBs. Além de facilitar o entendimento, essa organização possibilita determinar as propriedades do sistema, tais como: alçabilidade e vivacidade.

2. *Adaptações e flexibilidade de programação*: pela característica da linguagem gráfica, torna-se fácil a adaptação do SCM às várias missões com diferentes objetivos, de acordo com a proposta inicial da tese. Ademais, a estrutura modular permite modificar e programar novos sistemas sem muitas dificuldades, comparada a outras linguagens de programação existentes.

3. *Rede determinística*: embora não sendo necessário a implementação de algorítimos heurísticos utilizados nas técnicas de superposição, o trabalho mostrou que pode-se modelar o mundo real em redes determinística sem limitar as aplicação. E dentro deste conceito as redes de Petri obtiveram um bom desempenho. Também, de acordo com as fonte pesquisadas, modelar sistemas heurístico reais em rede de Petri é uma interessante alternativa e apresenta ótimos resultados práticos.

4. *Formalismo*: esse estudo permitiu introduzir conceitos básicos para a utilização do formalismo de linguagens para aplicação nos sistemas de controle de missões de AUVs. Mostrou que é possível a programação de sistemas em alto-nível utilizando esse formalismo.

7.4 Futuros trabalhos

Este trabalho, não somente estimulou o desenvolvimento de um SCM a partir de um modelo estruturado em redes de Petri, mas também abriu novos campos que servirão como proposta de futuras pesquisas dentro da área de desenvolvimento sistemas de controle de missão. Algumas sugestões estão listadas a seguir:

1. *Desenvolvimento de métodos para localização de objetos*: apesar da pesquisa descrever uma missão que localiza e segue dutos submarinos, esse objetivo ficou em segundo plano. Criar um algoritmo exclusivo para localização de dutos e objetos em ambientes submarinos e de baixa visibilidade, através técnicas de processamento de imagem (BALASURIYA; URA, 2002), pode tornar esse simulador bastante interessante.

2. *Aplicar técnicas deliberativas*: promover estudos que possibilitem a utilização desta interessante técnica para criar planejamentos e trajetórias em ambientes

desconhecidos. Na abordagem - *off-board* - descrita nessa tese, o programador é responsável por todo o planejamento da missão. Para missões de longo alcance e duração, o cenário aplicado neste trabalho é inviável. Estudos de técnicas deliberativas são importantes neste caso. Se a missão falhar, um replanejamento automático é executado de forma deliberativa, sem qualquer intervenção humana, possibilitando que o AUV complete seus objetivos. Esses recursos de planejamento são bastante explorados em (TURNER, 1995).

3. *Desenvolver um sistema especialista híbrido*: adicionar um controle híbrido, capaz de combinar as partes discreta dos SEDs com elemantos do controle contínuo. No caso dos SCM, um sistema especialista poderia ser composto pelas redes de Petri juntamente com as lógicas Fuzzy ou por qualquer outra ferramenta que seja capaz de realizar as tarefas e modelar sistemas não-determinísticos.

4. *Aplicação no Luma*: desenvolver um sistema para codificar toda a linguagem das redes de Petri em código C, por exemplo, para implementação em controladores. Desta forma, todo o projeto utilizaria as características das redes de Petri para verificação das propriedades do sistema, e o programa compilaria as missões em uma linguagem compatível às utilizadas pelos hardwares mais populares existentes no mercado, usando essa método diretamente no ROV LUMA.

-

Referências Bibliográficas

ALAMI, R.; CHATILA, R.; FLEURY, S.; GHALLAB, M.; INGRAND, F. An Architecture for Autonomy. **INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH**, v.17, p.315–337, 1998.

ALLEN, B.; STOKEY, R.; AUSTIN, T.; FORRESTER, N.; GOLDSBOROUGH, R.; PURCELL, M.; ALT, C. von. REMUS: a small, low cost auv; system description, field trials and performance results. In: OCEANS '97. MTS/IEEE CONFERENCE PROCEEDINGS, 1997. 1997. v.2, p.994–1000 vol.2.

ALLMENDINGER, E. (1990). Submersible Vehicle Systems Design. , 1990. In: JERSEY CITY, NJ: SOCIETY OF NAVAL ARCHITECTS AND MARINGE ENGINEERS, 1990. 1990.

ARBIB, M. A. **Schema theory. The Encyclopedia of Artificial Intelligence.** : Wiley-Interscience., 1992.

ARKIN, R. Motor schema based navigation for a mobile robot: an approach to programming by behavior. In: ROBOTICS AND AUTOMATION. PROCEEDINGS. 1987 IEEE INTERNATIONAL CONFERENCE ON, 1987. 1987. v.4, p.264–271.

ARKIN, R.; BALCH, T. AuRA: principles and practice in review. **Journal of Experimantal and Thoerical Artificail Intelligence**, v.9, p.175–189, 1997.

ASAKAWA, K.; KOJIMA, J.; KATO, Y.; MATSUMOTO, S.; KATO, N. Autonomous underwater vehicle AQUA EXPLORER 2 for inspection of underwater cables. In: UNDERWATER TECHNOLOGY, 2000. UT 00. PROCEEDINGS OF THE 2000 INTERNATIONAL SYMPOSIUM ON, 2000. 2000. p.242–247.

AUGUSTO, S. R. **Uma Plataforma Móvel Para Estudos de Autonomia**. 2007. Tese (Doutorado em Ciência da Computação) — Escola politecnica da Universidade de São Paulo, Engenharia de Sistemas.

B., R.; S., M. V. G.; H., M.; G., M. Review of Sensor Technologies for In-line Inspection of Natural Gas Pipelines. **Sandia National Laboratories**, v.7, p.1–10, 2005.

BALASURIYA, A.; URA, T. Vision-based underwater cable detection and following using AUVs. In: OCEANS '02 MTS/IEEE, 2002. 2002. v.3, p.1582–1587 vol.3.

BALLARD, R. **The Discovery of the Titanic**. New York, USA: Madison Press Books, 1987.

BARBIER, M.; J.F.GABARD; BERTHOLOM, A.; DUPAS, Y. An Onboard Software Decisional Architecture for Rapid Environmental Assessment Missions. In: IFAC WORLD CONGRESS, 18., 2011, Milano, Italy. 2011. p.1797–11802.

BARROUIL, C.; LEMAIRE, J. **Advanced Real-time Mission Management for an AUV**. 2005.

BASSI, D. Multilevel control structure for autonomous robotic aircraft operation. **Space Exploration Technologies**, v.6960, p.1–10, Apr 2008.

BLIDBERG, D. R. The Development of Autonomous Underwater Vehicles AUV a Brief Summary. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, ICRA 2001, 2001, Seoul, Korea. **Proceedings** 2001. p.1–12.

BLIDBERG, R.; ALI, H. B.; CHECHIN, S. AUV Design and Experience. **Institute of Marine Technology**, Vladivostok,Russia, 1995.

BOLTRYK, P.; HILL, M.; KEARY, A.; PHILLIPS, B.; ROBINSON, H.; WHITE, P. An ultrasonic transducer array for velocity measurement in underwater vehicles. **Ultrasonics**, v.42, n.19, p.473 – 478, 2004.

BOUSSINOT, F.; SIMONE, R. de. The ESTEREL Language. **Real Time Programming IEEE**, v.79, p.1293–13047, 1991.

BRADLEY, A.; FEEZOR, M.; SINGH, H.; SORRELL, F. Y. Power systems for autonomous underwater vehicles. **Oceanic Engineering, IEEE Journal of**, v.26, n.4, p.526–538, 2001.

BROOKS, R. A. ARobust Layered for a Control System for a Mobile Robot. **Robot and Auto 2**, v.3, p.14–23, 1986.

BROOKS, R. A Robot That Walks: emergent behaviours from a carefully evolved network. **Neural Computation**, v.2, p.253–262, 1989.

CACCIA, M.; COLETTA, P.; BRUZZONE, G.; VERUGGIO, G. Notes on the execution control of robotic tasks: a petri net-based approach. In: CONSIGLIO NAZIONALE DELLE RICERCHE – ISTITUTO AUTOMAZIONE NAVALE, 2013, Genova, Italy. 2013.

CAMBONE, S. A.; KRIEG, K.; PACE, P.; II, L. W. **Unmanned Aircraft Systems Roadmap**. Washington D.C, USA: Office of the Secretary of Defese, 2005. Technical Report.

SPRINGER (Ed.). **Introduction to Discrete Event Systems**. : Springer, 2008.

CAVALLO, E.; MICHELINI, R. A robotic equipment for the guidance of a vectored thrustor AUV. **35th International Symposium on Robotics ISR**, v.1, p.1–6, 23-26 March 2004.

CHANG, Z. hu; BIAN, X.-Q.; SHI, X. cheng. Autonomous underwater vehicle: petri net based hybrid control of mission and motion. In: MACHINE LEARNING AND CYBERNETICS, 2004. PROCEEDINGS OF 2004 INTERNATIONAL CONFERENCE ON, 2004. 2004. v.2, p.1113–1118 vol.2.

CONNELL, J. H. SSS: a hybrid architecture applied to robot navigation. In: ROBOTICS AND AUTOMATION, 1992. PROCEEDINGS., 1992 IEEE INTERNATIONAL CONFERENCE ON, 1992. 1992. p.2719–2724 vol.3.

CRUZEN, C. A.; THOMPSON, J. T. **Advancing Autonomous Operations Technologies for NASA Missions**. Huntsville, AL: National Aeronautics and Space Administration (NASA), 2012. Draft 10.

CURTIN, T.; BELLINGHAM, J. Autonomous ocean-sampling networks. **Oceanic Engineering**, v.26, p.421–423, 2001.

CUTIPA-LUQUE, J. C. **Controle Robusto Multivariável para um Veículo Submersivel Autônomo**. 2007. Dissertação (Mestrado em Ciência da Computação) — Escola Politécnicaa da Universidade de São Paulo Departamento de Engenharia Mecânica.

CUTIPA-LUQUE, J. C. **Identificação e Controle de um Veículo Submersível Autônomo Sub-atuado**. 2012. Doutorado — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecânica, São Paulo.

DAMN. DAMN: a distributed architecture for mobile navigation - thesis summary. In: JOURNAL OF EXPERIMENTAL AND THEORETICAL ARTIFICIAL INTELLIGENCE, 1995. AAAI Press, 1995. p.339–360.

DESHPANDE, A.; SOUSA, J. Borges de. Real-time multi-agent coordination using DIADEM: applications to automobile and submarine control. In: SYSTEMS, MAN, AND CYBERNETICS, 1997. COMPUTATIONAL CYBERNETICS AND SIMULATION., 1997 IEEE INTERNATIONAL CONFERENCE ON, 1997. 1997. v.2, p.1769–1774 vol.2.

ENTREPRISE, J. G. R. **Development and Utilization of Robotic and Unmanned Ground Vehicles**. Washington D.C, USA: Office of the Under Secretary of Defense, Acquisition, Technology and Logistics, Portfolio Systems Acquisition, Land Warfare and Munitions, Joint Ground Robotics Enterprise, 2006. Report to Congress.

EVANS, J.; PATRON, P.; PRIVAT, B.; JOHNSON, N.; CAPUS, C. AUTOTRACKER: autonomous inspection 2014; capabilities and lessons learned in offshore operations. In: OCEANS 2009, MTS/IEEE BILOXI - MARINE TECHNOLOGY FOR OUR FUTURE: GLOBAL AND LOCAL CHALLENGES, 2009. 2009. p.1–7.

EVANS, J.; PETILLOT, Y.; REDMOND, P.; WILSON, M.; LANE, D. AUTOTRACKER: auv embedded control architecture for autonomous pipeline and cable tracking. In: OCEANS 2003. PROCEEDINGS, 2003. 2003. v.5, p.2651–2658 Vol.5.

FIKES, R. E.; NHSSON, N. J. STRIPS: a new approach to the application of .theorem proving to problem solving. In: IJCAI, IMPERIAL COLLEGE, 2., 1971, London, England. 1971. p.189 – 201.

FIRBY, R. J. **Adaptive Execution in Complexy Dynamics Worlds**. 1989. Tese (Doutorado em Ciência da Computação) — Yale University.

FOSSEN, T. **Guidance and control of ocean vehicles**. : Wiley, 1994.

GAVIA, T. **Teledyne Gavia**: autonomous underwater vehicles. Acessado em http://www.gavia.is/, Disponivel Online.

GHALLAB, M.; NAU, D.; TRAVERS, P. **Automated Planning**: theory & practice (the morgan kaufmann series in artificial intelligence). : Morgan Kaufmann, 2004.

GOULART, C. **Modelagem Simulação e Controle de um Veiculo Submarinode Operacao Remota**. Junho 2007. Tese (Doutorado em Ciência da Computação) — COPPE–UFRJ.

HABERBUSCH, M. S.; STOCHL, R. J.; NGUYEN, C. T.; CULLER, A.; WAINRIGHT, J.; MORAN, M. E. Rechargeable cryogenic reactant storage and delivery system for fuel cell powered underwater vehicles. In: AUTONOMOUS UNDERWATER VEHICLES, 2002. PROCEEDINGS OF THE 2002 WORKSHOP ON, 2002. 2002. p.103–109.

HERMAN, M.; HONG, T.-H.; SWETZ, S.; OSKARD, D.; ROSOL, M. Planning and world modeling for autonomous undersea vehicles. In: INTELLIGENT CONTROL, 1988. PROCEEDINGS., IEEE INTERNATIONAL SYMPOSIUM ON, 1988. 1988. p.370–375.

IORDACHE, M.; MOODY, J.; ANTSAKLIS, P. Synthesis of deadlock prevention supervisors using Petri nets. **Robotics and Automation, IEEE Transactions on**, v.18, n.1, p.59–68, Feb 2002.

JALBERT, J.; BAKER, J.; DUCHESNEY, J.; PIETRYKA, P.; DALTON, W.; BLIDBERG, D.; CHAPPELL, S.; NITZEL, R.; HOLAPPA, K. A solar-powered autonomous underwater vehicle. In: OCEANS 2003. PROCEEDINGS, 2003. 2003. v.2, p.1132–1140 Vol.2.

JALVING, B. The NDRE-AUV flight control system. **Oceanic Engineering, IEEE Journal of**, v.19, n.4, p.497–501, 1994.

JOHNS, K.; TAYLOR, T. **Professional Microsoft Robotics Developer Studio**. : Wrox, 2008.

JUNIOR, V. G. **Arquitetura Híbrida para Robôs Móveis Baseada em Funções de Navegação Humana**. 2006. Tese (Doutorado em Ciência da Computação) — Escola Politécnica da Universidade de São Paulo.

KAMINER, I.; PASCOAL, A. M.; KHARGONEKARG, P. P.; COLEMAN, E. E. A Velocity Algorithm for the Implementation of Gain-Scheduled Controllers. **Automatica**, v.31, p.1185–1191, 1995.

KAO, M.; WEITZEL, G.; ZHENG, X.; BLACK, M. A simple approach to planning and executing complex AUV missions. In: AUTONOMOUS UNDERWATER VEHICLE TECHNOLOGY, 1992. AUV '92., PROCEEDINGS OF THE 1992 SYMPOSIUM ON, 1992. 1992. p.95–102.

KENCONLEY. **ROS.org**: smach. Acessado em http://wiki.ros.org/smach, Disponivel Online.

KONDOA H. & URA, T. Navigation of an AUV for investigation of underwater structures. **Control Engineering Practice**, v.12, p.1551–1559, 2004.

KOSECKA. Application of discrete events systems for modeling and controlling robotic agents. In: ROBOTICS AND AUTOMATION, 1994. PROCEEDINGS., 1994 IEEE INTERNATIONAL CONFERENCE ON, 1994. 1994. p.2557–2562 vol.3.

LEE, P.-M.; JUN, B.-H.; KIM, K.; LEE, J.; AOKI, T.; HYAKUDOME, T. Simulation of an Inertial Acoustic Navigation System With Range Aiding for an Autonomous Underwater Vehicle. **Oceanic Engineering, IEEE Journal of**, v.32, n.2, p.327–345, 2007.

LEONARD, J. J.; BENNETT, A. A.; SMITH, C. M.; JACOB, H.; FEDER, S. Autonomous Underwater Vehicle Navigation. In: MIT MARINE ROBOTICS LABORATORY TECHNICAL MEMORANDUM, 1998, Cambrigde, MA. 1998.

MAES, P. How To Do the Right Thing. **Connection Science Journal**, v.1, p.291–323, 1989.

MAJUMDER, S.; SCHEDING, S.; DURRANT-WHYTE, H. F. Multisensor data fusion for underwater navigation. **Robotics and Autonomous Systems**, v.35, n.2, p.97 – 108, 2001.

MARCO, D.; HEALEY, A.; MCGHEE, R. Autonomous Underwater Vehicles: hybrid control of mission and motion. **Autonomous Robots**, v.3, p.169–186, 1996.

MAURYA, P.; DESA, E.; PASCOAL, A.; BARROS, E.; NAVELKAR, G.; MADHAN, R.; MASCARENHAS, A.; PRABHUDESAI, S.; AFZULPURKAR, S.; GOUVEIA, A.; S.NAROJI; L.SEBASTIAO. Control of the Maya AUV in the vertical and horizontal planes: theory and practical results. In: IFAC CONFERENCE ON MANOEUVRING AND CONTROL OF MARINE CRAFT, 7., 2007. 2007.

MEDEIROS, A. A. D. d. **Journal of the Brazilian Computer Society, special issue on Robotics**. : JBCS, 1998. v.4, n.3.

MODARRESS, D.; SVITEK, P.; MODARRESS, K.; WILSON, D. Micro-Optical Sensors for Underwater Velocity Measurement. In: UNDERWATER TECHNOLOGY AND WORKSHOP ON SCIENTIFIC USE OF SUBMARINE CABLES AND RELATED TECHNOLOGIES, 2007. SYMPOSIUM ON, 2007. 2007. p.235–239.

MONTEEN, B.; WARNER, P.; RYLE, J. Cal poly autonomous underwater vehicle. **California Polytechnic State University**, v.1, p.20–28, 2000.

MURATA, T. Petri nets: properties, analysis and applications. **Proceedings of the IEEE**, v.77, n.4, p.541–580, Apr 1989.

MURPHY, R. R. **Introduction to AI Robotic**. Cambridge, Massachusetts: The MIT Press, 2000.

NAVAL OPERATIONS, C. on Autonomous Vehicles in Support of. **Autonomus Vehicles is Support of Naval Operations**. Washington, DC, USA: The National Academies Press, 2005.

OLIVEIRA, P.; PASCOAL, A.; SILVA, V.; SILVESTRE, C. Design, development, and testing at sea of the mission control system for the MARIUS autonomous underwater vehicle. In: OCEANS '96. MTS/IEEE. PROSPECTS FOR THE 21ST CENTURY. CONFERENCE PROCEEDINGS, 1996. 1996. v.1, p.401–406 vol.1.

OLIVEIRA, P.; PASCOAL, A.; SILVA, V.; SILVESTRE, C. Mission Control of the MARIUS AUV: system design, implementation, and sea trials. In: OCEANS '98. MTS/IEEE, 1998. 1998.

P HOPKINS, A. P. **Pipeline Internal Inspection**. : Penspen Integrity, 1995.

PALOMERAS, N.; EL-FAKDI, A.; CARRERAS, M.; RIDAO, P. COLA2: a control architecture for auvs. **Oceanic Engineering, IEEE Journal of**, v.37, n.4, p.695–716, 2012.

PALOMERAS, N.; RIDAO, P.; CARRERAS, M.; SILVESTRE, C. J. F. Using petri nets to specify and execute missions for autonomous underwater vehicles. In: IROS, 2009. IEEE, 2009. p.4439–4444.

ROBERTS, G. N.; SUTTON, R. (Ed.). **Advances in Unmanned Marine Vehicle**. Lisboa Portugal: IEEE, 2006. p.335–386.

PASTER, D. Importance of Hydrodynamic Considerations for Underwater Vehicle Design. In: OCEANS '86, 1986. 1986. p.1413–1422.

PEREIRA, F. L. **Sistemas e Veículos Autonomos - Aplicação Defesa**. 2005. Tese (Doutorado em Ciência da Computação) — Instituto de Defesa Nacional.

PIRJANIAN, P. Multiple Objective Behavior-Based Control. In: JRAS99, 1999. 1999.

RAJAN, K.; MCGANN, C.; PY, F.; THOMAS, H. Robust Mission Planning Using Deliberative Autonomy For Autonomous Underwater Vehicles. **ICRA Workshop on Robotics in challenging and hazardous environments**, v.2, p.21–25, 2007.

RIBAS, D.; RIDAO, P.; NEIRA, J. **Underwater SLAM for Structured Environments Using an Imaging Sonar**. : Springer, 2010. 1-119p. (Springer Tracts in Advanced Robotics, v.65).

ROCHA, R. P. **Estado da Arte da Robotica Movel em Portugal**. Lisboa, Portugal: Departamento de Egenharia Electrotécnica da Universidade de Coimbra (FCTUC), 2001. Technical Report.

ROSENSCHEIN S., K. The synthesis of digital machines with provable epistemic properties. In: CONFERENCE ON THEORETICAL ASPECTS OF REASONING ABOUT KNOWLEDGE, 1986. **Proceedings** Morgan Kaufmann Publishers Inc, 1986.

ROSS, C. T. A conceptual design of an underwater vehicle. **Ocean Engineering**, v.33, n.16, p.2087 – 2104, 2006.

SAFFIOTTI, A.; KONOLIGE, K.; RUSPINI, E. H. A Multivalued Logic Approach to Integrating Planning and Control. **Artificial Intelligence**, v.76, p.481–526, 1995.

SANTéRIO, F. C. **Controle Baseado em Comportamento de Robôs Móveis Auntônomos com Sensores Ópticos e Ultrassonicos**. 2010. Dissertação (Mestrado em Ciência da Computação) — PUC-Rio.

SILVA, J.; MARTINS, A.; PEREIRA, F. A Reconfigurable Mission Control System for Underwater Vehicles. **In OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Centurys**, v.3, p.1088–1092, 1999.

SIMMONS, R.; APFELBAUM, D. A Task Description Language for Robot Control. **International Conference Intelligent Robots and Systems**, v.3, p.1931–1937, 1998.

STACHIW, J. Acrylic plastic as structural material for underwater vehicles. In: INTERNATIONAL SYMPOSIUM ON UNDERWATER TECHNOLOGY, 2004. 2004. v.1, p.289–296.

STUTTERS, L.; LIU, H.; TILTMAN, C.; BROWN, D. Navigation Technologies for Autonomous Underwater Vehicles. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, v.38, n.4, p.581–589, 2008.

TAKAGAWA, S. Feasibility Study on DMFC Power Source for Underwater Vehicles. In: UNDERWATER TECHNOLOGY AND WORKSHOP ON SCIENTIFIC USE OF SUBMARINE CABLES AND RELATED TECHNOLOGIES, 2007. SYMPOSIUM ON, 2007. 2007. p.326–330.

TANGIRALA S. & DZIELSKI, J. A variable buoyancy control system for a large AUV. **IEEE Journal of Oceanic Engineering**, v.32, p.762–771, 2007.

CAMBRIDGE, T. M. P. (Ed.). **Probabilistic Robotics**. : Autonomous Underwater Vehicle, 2005.

TOAL, D. J.; FLANAGAN, C.; LYONS, W. B.; NOLAN, S.; LEWIS, E. Proximal object and hazard detection for autonomous underwater vehicle with optical fibre sensors. **Robotics and Autonomous Systems**, v.53, n.34, p.214 – 229, 2005.

TURNER, R. Orca: intelligent adaptive reasoning for autonomous underwater vehicle control. In: FLAIRS-95 INTERNATIONAL WORKSHOP ON INTELLIGENT ADAPTIVE SYSTEMS (IAS-95), 1995, Melbourne Beach, Florida. **Proceedings** 1995. p.52–62.

VALAVANIS, K.; GRACANIN, D.; MATIJASEVIC, M.; KOLLURU, R.; DEMETRIOU, G. Control architectures for autonomous underwater vehicles. **Control Systems, IEEE**, v.17, n.6, p.48–64, 1997.

WASSERMAN, K.; MATHIEU, J.; WOLF, M.; HATHI, A.; FRIED, S.; BAKER, A. Dynamic buoyancy control of an ROV using a variable ballast tank. **OCEANS**, v.SP, p.2888–2893, 2003.

WERNLI, R. L. **Low Cost AUV For Military Applications**: is the technology ready. San Diego, CA: Space and Naval Warfare Systems Center, 2001. Report.

WILLCOX, S.; J., V.; GRIEVE, R.; RISH, J. The bluefin BPAUV: an organic widearea bottom mapping and mine-hunting vehicle. In: UUST 01, 2001. **Proceedings** 2001.

WILLIAMS, S. B.; NEWMAN, P.; GAMINI; DISSANAYAKE; DISSANAYAKE, G.; ROSENBLATT, J.; DURRANT-WHYTE, H. A decoupled, distributed AUV control architecture. In: INTERNATIONAL SYMPOSIUM ON ROBOTICS, 31., 2000. **Proceedings** 2000.

WINCHESTER, C.; GOVAR, J.; BANNER, J.; SQUIRES, T.; SMITH, P. A survey of available underwater electric propulsion technologies and implications for platform system safety. In: AUTONOMOUS UNDERWATER VEHICLES, 2002. PROCEEDINGS OF THE 2002 WORKSHOP ON, 2002. 2002. p.129–135.

YOERGER, D.; JAKUBA, M.; BRADLEY, A.; BINGHAM, B. Techniques for Deep Sea Near Bottom Survey Using an Autonomous Underwater Vehicle. In: THRUN, S.; BROOKS, R.; DURRANT-WHYTE, H. (Ed.). **Robotics Research**. : Springer Berlin Heidelberg, 2007. p.416–429. (Springer Tracts in Advanced Robotics, v.28).