UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Ermes Ferreira Costa Neto

DESENVOLVIMENTO DE MÉTODO PARA PROJETO MODULAR DE CÉLULA DE MANUFATURA ROBOTIZADA COM PROGRAMAÇÃO NO NÍVEL DE TAREFAS

Recife, PE
2013

Ermes Ferreira Costa Neto

DESENVOLVIMENTO DE MÉTODO PARA PROJETO MODULAR DE CÉLULA DE MANUFATURA ROBOTIZADA COM PROGRAMAÇÃO NO NÍVEL DE TAREFAS

Dissertação apresentada objetivando a obtenção do grau de Mestre em Engenharia Mecânica na Universidade Federal de Pernambuco no Programa de Pós-Graduação em Engenharia Mecânica.

Orientador: Prof. Dr. André Murilo de Almeida Pinto

Coorientador: Prof. Dr. Pedro Manoel González del Foyo

Recife, PE
2013

Catalogação na fonte
Bibliotecário Marcos Aurélio Soares da Silva, CRB-4 / 1175

C837d Costa Neto, Ermes Ferreira.
Desenvolvimento de método para projeto modular de célula de manufatura robotizada com programação no nível de tarefas / Ermes Ferreira Costa Neto. - Recife: O Autor, 2013.
156 folhas, il., gráfs., tabs.

Orientador: Profº Dr.º André Murilo de Almeida Pinto.
Coorientador: Prof.º Dr.º Pedro Manoel González Del Foyo.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Mecânica, 2013.
Inclui Referências e Apêndices.

1. Engenharia Mecânica. 2.Sistema Modular. 3.Autônatos. 4.Célula de Manufatura Robotizada. I. Pinto, André Murilo de Almeida (Orientador). II. Título.

621 CDD (22. ed.) UFPE
BCTG/2013-219

“DESENVOLVIMENTO DE MÉTODO PARA PROJETO MODULAR DE CÉLULA DE MANUFATURA ROBOTIZADA COM PROGRAMAÇÃO NO NÍVEL DE TAREFAS”

ERMES FERREIRA COSTA NETO

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO TÍTULO DE MESTRE EM ENGENHARIA MECÂNICA

ÁREA DE CONCENTRAÇÃO: ENGENHARIA DE MATERIAIS E FABRICAÇÃO
APROVADA EM SUA FORMA FINAL PELO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA/CTG/EEP/UFPE

_____
Prof. Dr. ANDRÉ MURILO DE ALMEIDA PINTO
ORIENTADOR/PRESIDENTE

_____
Prof. Dr. PEDRO MANOEL GONZÁLEZ DEL FOYO
CO-ORIENTADOR

_____
Prof. Dr. CEZAR HENRIQUE GONZALEZ
VICE-COORDENADOR DO PROGRAMA

BANCA EXAMINADORA:

_____
Prof. Dr. ANDRÉ MURILO DE ALMEIDA PINTO (UFPE)

_____
Prof. Dr. PEDRO MANOEL GONZÁLEZ DEL FOYO (UFPE)

_____
Prof. Dr. TIAGO LEITE ROLIM (UFPE)

_____
Prof. Dr. JOÃO PAULO CERQUINHO CAJUEIRO (UFPE)

Esse trabalho é dedicado a todas as pessoas que lutam pelo desenvolvimento do Brasil, igualdade, justiça social, melhoria da qualidade de vida e da educação do povo Brasileiro.

Agradecimentos

Ao meu orientador, Professor Dr. André Murilo, os maiores e mais sinceros agradecimentos pela confiança, amizade, conhecimento e motivação, pois sempre me orientou de forma exemplar e dedicada durante todo meu mestrado.

Ao Professor Dr. Pedro Manoel, pela orientação, direcionamento e todo apoio durante a realização deste trabalho.

Aos demais professores do Programa de Pós-graduação em Engenharia Mecânica (PPGEM) da UFPE.

Aos professores do ITA, pelos ensinamentos transmitidos, no início do mestrado. Meu agradecimento especial aos Professores Dr. Jefferson de Oliveira Gomes, Ph.D. Luís Gonzaga Trabasso e Dr. Emília Villani.

Aos professores da UPE, pelo apoio e ensinamentos. Meu agradecimento especial aos Professores Dr. Eduardo César de Miranda Loureiro, Ph.D. Alcides Codiceira Neto e Dr. Sérgio Peres Ramos da Silva.

Ao Paulo Roberto Sales Lages, pelo apoio, ajuda e compreensão.

Aos meus pais e minha irmã, que sempre me deram todo o apoio necessário e imprescindível para seguir este trabalho.

À minha esposa Rafaela Campos, pela compreensão, paciência, carinho e ajuda, pois foram indispensáveis para a conclusão desta minha etapa.

Ao meu filho Miguel Ferreira, pela atenção e tempo que não pude oferecê-lo devido a minha ausência.

Aos colegas de PPGEM da UFPE, pela amizade e descontração e todos aqueles que, mesmo não tenha sido aqui citado nominalmente, colaboraram, direta ou indiretamente, de maneira não menos importante, na produção deste trabalho.

"Algo é só impossível até que alguém duvide e acabe provando ao contrário"

(Albert Einstein)

Resumo

As mudanças atuais nas técnicas de produção impõem a necessidade de flexibilidade dos sistemas de manufatura, que permitam aumento da produção e variedade de produtos, sem perda de qualidade. Um ambiente favorável à aplicação de célula de manufatura robotizada (CMR). Entretanto, as empresas estão encontrando dificuldades no seu projeto e operação devido ao alto grau complexidade que exige um grande esforço financeiro e intelectual, no caso particular da CMR, o uso de robôs ainda exige a necessidade de exatidão e capacidade de repetitividade nos movimentos. O objetivo deste trabalho é apresentar uma abordagem sistemática para o projeto de sistema modular por meio do uso de diferentes técnicas, utilizando um formalismo básico, baseado em autômato. Para automatizar os processos de fabricação pela aplicação de CMR de uma forma rápida, evitando a migração de erro entre as fases de projeto e que permita a programação no nível de tarefas. Serão utilizados as técnicas de modelo espiral de engenharia de requisitos, diagrama IDEF0, teoria de controle supervisório (TCS) e sistema supervisório modular local (SML), verificação de modelos e a ferramenta *state diagram* do ambiente computacional *LabVIEW™*, que é utilizado como plataforma de programação, integração das diversas arquiteturas e geração de uma interface de operação. Este método foi aplicado no projeto de uma CMR para seleção e movimentação de quatro diferentes peças. Os resultados mostram sua utilidade em aplicações industriais reais, reduzindo o custo e diminuindo o tempo de desenvolvimento no processo de concepção.

Palavras Chave: Projeto de Sistema Modular; Autômatos; Célula de Manufatura Robotizada.

Abstract

Current changes in production techniques impose the need for flexibility in manufacturing systems, to increase production and product variety, without loss of quality. An environment favorable for the implementation of a robotic manufacturing cell (RMC). However, companies are failing difficulties in their design and operation through their degree complexity that require a large financial outlay and intellectual, in the particular case of the RMC, the use of robot still requires the need for high accuracy and repeatability in movement ability. The objective of this work is to present a systematic approach to modular system design through the use of different techniques, using a basic formalism, based on finite automata, to automate manufacturing processes by applying RMC quickly avoiding migration error between project phases, allowing the task-level programming.. We use the following techniques spiral model of requirements engineering, IDEF0 diagram, supervisory control theory and local modular supervisory system, model verification and the state diagram tool of the computing environment programming platform LabVIEW ™, integration of various architectures and generation of a user interface. This method was applied in the design of a RMC for selection and handling of four different parts. The results show its usefulness real in industrial applications, reducing the cost and decreasing the time of development in the design process

Keywords: System Modular Design; Finite automata; Robotic Cell Manufacturing.

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

AL	*Assembly Language*
CCD	C*harge-Coupled Device*
CMOS	*Complementary Metal Oxide Semiconductor*
CMR	Célula de Manufatura Robotizada
D-H	*Denavit–Hartenberg*
DOF	*Depth of Field*
FMS	*Flexible Manufacturing System*
FOV	*Field Of Vision*
GDL	Graus de Liberdade
GE	*General Electrics*
IBM	*International Business Machines*
IDEF0	*Integrated Computer Aided Manufacturing Definition Methodology*
IEEE	*Institute of Electrical and Electronics Engineers*
IHM	Interface Homem Máquina
LED	*Light-Emitting Diode*
NASA	*National Aeronautics and Space Administration*
NI	*National Instruments*
P	Prismática
R	Revolução
Re	Resolução
RMC	*robotic manufacturing cell*
RSP	Representação de Sistema Produto
SCARA	*Selective Compliance Arm for Robotic Assembly*
SED	Sistemas a eventos discretos
SFM	Sistema Flexível de Manufatura
SML	Sistema Supervisório Modular Local
SRI	*Stanford Research Institute*
SS	*Sensor Size*
TCP	*Tool Center Point*
TCS	Teoria de Controle Supervisório
USB	*Universal Serial Bus*
VAL	*Victor Assembly Language*
VIDICON	*VIdeo Cameras using tubes*
WAVE	*A Model-Based Language for Manipulator Control*
WD	*Working Distance*

Sumário

1 Introdução

Este capítulo apresenta a caracterização do problema em estudo com uma breve consideração sobre as vantagens da automação flexível nas indústrias de manufatura e as dificuldades durante o projeto e implantação. Também apresenta o objetivo geral e os específicos e a organização deste trabalho.

1.1 Caracterização do Problema em Estudo

Antes da década 60, o mercado mundial era caracterizado por um crescente aumento quantitativo de produção, onde tudo que era produzido era vendido, sem a necessidade de preços competitivos. Os anos que se seguiram, a globalização e a evolução tecnológica puseram uma mudança no mercado, iniciando uma fase de exigências por melhores preços, quantidade e qualidade. Já no final dos anos 70, surgiu uma nova exigência do mercado consumidor, o menor tempo no lançamento de novos produtos. As empresas de manufatura tiveram que modernizar as linhas de produção para reduzir o tempo despendido nas alterações para a fabricação de novos produtos (SANTOS, 2008).

A partir dos anos 90, o mercado assumiu uma tendência de crescimento do consumo, coma exigência de segmentação e de diferenciação de produtos. As empresas, para continuarem competitivas, precisam de inovação no produto, ou seja, habilidade para renovar rapidamente o produto com qualidade e baixo custo. É justamente o período em que se encontra o atual mercado, onde é fundamental a manutenção do ciclo de inovação de produtos para o sucesso, caso contrário o fracasso é inevitável (JUNQUEIRA, 2001).

As empresas de manufatura para manterem competitivas no mercado atual, precisam de sistemas de produção de alta flexibilidade, tanto no que se refere ao volume quanto à diversidade de produtos, de forma efetiva, confiável e a um baixo custo (PARACENCIO, 2009). Os sistemas de produção tradicionais baseados em fabricação por bateladas, ou lotes, estão sendo substituídos por sistemas modernos que podem produzir qualquer produto em diferentes combinações e horários, sem exigir a necessidade de fabricação por bateladas, ou lotes. Os sistemas de produção modernos são também chamados

de sistemas flexíveis de manufatura (SFM). O SFM produz uma variedade de produtos modificando a sua configuração de acordo com o planejamento da produção. Esta flexibilidade permite uma alocação mais rápida dos recursos, mas incrementa a complexidade de controle do sistema (NAKAMOTO, 2001).

Entretanto, muitas empresas de manufatura estão encontrando dificuldades na implantação do SFM. A complexidade deste sistema exige um grande esforço financeiro e intelectual no seu projeto e implantação. Além disso, a combinação de perdas em consequência: das aproximações numéricas; do tempo ocioso de equipamentos durante o *setup* para alterações de funcionalidade e do alto custo da equipe de programação e manutenção especializada, restringem os benefícios econômicos esperados (ERBE, 2002). Para o caso particular de Célula de Manufatura Robotizada (CMR), um tipo de SFM que possui robôs e sistemas inteligentes, pode-se acrescentar como dificuldade a necessidade de exatidão e repetitividade de posicionamento e nos movimentos exigidos nas operações de manipulação de peças, montagem, pintura, ou soldagem (DUELEN; SCHÖER, 1991).

No contexto nacional, as empresas despendem muitos recursos após a implantação da CMR. Até 90% destes recursos são gastos na reconfiguração (programação) após o *startup* inicial, dedicados, quase exclusivamente, à detecção e correção de falhas e realização de melhorias (MORAES; CASTRUCCI, 2007).

Desta forma, torna-se evidente a necessidade de apresentar uma solução para o projeto de SFM, no caso particular de CMR. Sendo assim, a proposta deste trabalho é modularizar o sistema a ser projetado, utilizando um formalismo básico de Sistemas a Eventos Discretos (SED) baseado em autômatos, em todas as etapas deste projeto. O uso do mesmo formalismo proporciona uma sistemática e um ganho de consistência nas transições entre as etapas, o que diminui a possibilidade de erro e facilita a implantação de melhorias.

Serão utilizados para cada etapa uma técnica que permita o uso do formalismo básico, como modelo espiral de engenharia de requisitos para a geração das especificações funcionais do sistema para células de manufatura robotizada com programação no nível de tarefas, diagrama IDEF0 para a definição dos módulos, teoria de controle supervisório para criação dos modelos dos autômatos e síntese do sistema supervisório modular local, verificação de modelos para averiguar o atendimento as especificações funcionais e implementação em ambiente *LabVIEW™* que permite uma programação gráfica e uso de máquina de estados.

Outro ponto importante consiste em apresentar uma solução que permita a reconfiguração da funcionalidade da CMR de forma fácil, rápida e com menor exigência de inteligência humana. A programação no nível de tarefas é realizada em interface amigável e de fácil interpretação, que possui recursos computacionais para a implementação dos algoritmos e comunicação direta com os sensores e atuadores do sistema. Neste trabalho, será proposto a programação no nível de tarefas para a CMR, conforme é utilizado na programação de robôs industriais. Este nível maior de abstração será utilizado para construção dos módulos no projeto do sistema modular, onde a parte contínua será encapsulada e a parte discreta tratada como eventos, conforme formalismo básico de SED.

A utilização de formalismo único, baseado em autômatos, no projeto de CMR é uma linha de pesquisa relativamente nova se considerarmos os estudos realizados a respeito dos já tradicionais processos de projetos de sistemas complexos.

A hipótese central deste trabalho é que a utilização de formalismo básico de SED, baseado em autômato, possa proporcionar a diminuição das dificuldades na sua implantação e no *setup* deste sistema. Outra razão é a incipiente ou inexistente disponibilidade de métodos ou procedimentos sistemáticos com uso de diferentes técnicas para o projeto deste sistema, que orientem as atividades de modo menos complexo e mais integrado. Como também, que auxiliem na minimização dos potenciais riscos nos estágios iniciais, momento que são tomadas as mais estratégicas e influentes decisões sobre o sistema e também sobre as sequencias de tarefas a serem aplicados neste sistema.

O método proposto neste trabalho foi utilizado no projeto modular de uma célula de manufatura robotizada para a realização da seleção e movimentação de 4 (quatro) diferentes peças, utilizando um robô antropomórfico, fabricante LabVolt e modelo 5150A, e visão computacional, a reconfiguração da funcionalidade pelo usuário foi realizada com abstração de acordo com a programação no nível de tarefas.

As conclusões apresentadas neste trabalho provam a utilidade do método proposto para o projeto de sistema modular de CMR. Ainda existe a possibilidade do uso deste método em outros sistemas de natureza híbrida, no qual o funcionamento pode ser definido de forma abstrata como um SED e suas tarefas representadas com o seu comportamento contínuo em módulos encapsulados e as transições ocorrendo devido o comportamento das variáveis de natureza discreta destes módulos.

1.2 Objetivo

O objetivo geral deste trabalho é apresentar uma proposta de um método para projeto modular com programação no nível de tarefas fundamento em técnicas que permitam o uso de formalismo único, baseado em autômatos, para diminuir a possibilidade de erro, facilitar a implantação de melhorias, ainda na fase de projeto e permitir o uso do conceito de automação flexível. Neste foco, optou-se por desenvolver uma abordagem mais específica do projeto de célula de manufatura robotizada, tipo de sistema flexível de manufatura, que possui um caráter modular.

Para a consecução do objetivo geral são propostos os seguintes objetivos específicos:

1. Levantar as necessidades e comportamentos desejados;
2. Identificar as partes do sistema;
3. Criar módulo para cada parte do sistema;
4. Realizar a integração conceitual dos módulos;
5. Elaborar os modelos de autômatos para cada módulo e restrição;
6. Elaborar o sistema supervisório modular local;
7. Implementar os autômatos, restrições e sistema supervisório em ambiente computacional *LabVIEW*™;
8. Realizar a integração em ambiente computacional *LabVIEW*™ dos modelos dos autômatos e supervisório, módulos computacionais e *hardware*;
9. Criar e implementar uma interface de programação no nível de tarefas em ambiente computacional *LabVIEW*™;
10. Realizar verificação na CMR por meio de simulação de falhas.

1.3 Organização do Trabalho

O texto está organizado em seis capítulos sumarizados como segue. O capítulo 1 apresenta a introdução, o objetivo, e organização deste trabalho. O capítulo 2 apresenta uma revisão bibliográfica sobre projeto modular, levantamento de requisitos, diagramas IDEF0,

sistemas a eventos discretos, automação industrial, sistemas flexíveis de manufatura e célula de manufatura robotizada. O método proposto neste trabalho é apresentado no capítulo 3, cuja aplicação no estudo de caso é detalhada no capítulo 4. Finalmente o capítulo 5 apresenta as conclusões e recomendações de futuros trabalhos.

2 Revisão Bibliográfica

Este capítulo apresenta uma introdução à automação, robótica e sistema de visão computacional. Em seguida, serão abordados os sistemas flexíveis de manufatura, célula de manufatura robotizada e programação no nível de tarefas. São apresentados alguns tópicos relacionados ao projeto de sistema modular e modelo espiral para levantamento de requisitos de projeto. Como também a criação de módulos conceituais utilizando diagramas IDEF0 e o conceito de sistemas a eventos discretos, teoria de controle supervisório com a abordagem para projeto de sistema supervisório modular local e verificação de modelos.

2.1 Automação Industrial

Os primeiros conceitos de automação industrial surgiram no período da revolução industrial por meio da mecanização, que consistia basicamente na substituição da força humana ou animal por outras formas de acionamento como, por exemplo, vapor (FREITAS, 2004). Atualmente, a automação industrial visa não somente obter economia de custos de mão de obra, mas o aumento da qualidade dos produtos, diminuição de tempo de produção, maior volume de produção, redução de custos e eliminação de riscos a que os operadores são expostos durante a produção manual, tais como ambientes inseguros, insalubres e atividades repetitivas (BARROS, 2006).

Assim, a automação industrial é a tecnologia que se ocupa da utilização de sistemas mecânicos, elétricos e computacionais para a realização de uma tarefa, ou processo com o mínimo possível de interferência humana (GROOVER, 2001) (PAZOS, 2002). A automação é classificada em três tipos básicos (GROOVER, 2001) (OLIVEIRA, 2003):

1. **Automação fixa**: Baseada na disposição física dos equipamentos do sistema de produção sem capacidade de alterar a sequência de tarefas. É capaz de fabricar um único tipo de produto. Características: Alto volume de produção; Alta eficiência e produtividade; Nenhuma flexibilidade de alteração de produto; Produção contínua.

2. **Automação programável**: Baseada na disposição física dos equipamentos do sistema de produção com capacidade de alterar a sequência de tarefas. É capaz de produzir diferentes tipos de produtos por bateladas ou lotes. A reconfiguração é realizada por meio de intervenção humana para realizar a programação, ajustes ou trocas de partes do sistema, que consome um tempo relativamente longo. Características: Baixo volume de produção; Produção por batelada, ou lotes.
3. **Automação flexível**: É uma extensão da automação programável, com a diferença que praticamente nenhum tempo é perdido durante a reconfiguração, pois os ajustes ou trocas de partes do sistema é realizado de forma autônoma. Por conseguinte, o sistema pode produzir qualquer produto em diferentes combinações e horários, sem exigir a necessidade de fabricação por bateladas, ou lotes. Características: Custo de mão de obra especializada para projeto, instalação e manutenção; Médio volume de produção; Produção contínua e capacidade de alteração de produto a qualquer momento.

A Figura 1 apresenta uma comparação entre os três tipos de automação em relação a quantidade de produção e variedade de produtos. Observa-se na figura para quantidades baixas de produção, ou lançamento de novos produtos, a produção manual é competitiva com a automação programável.

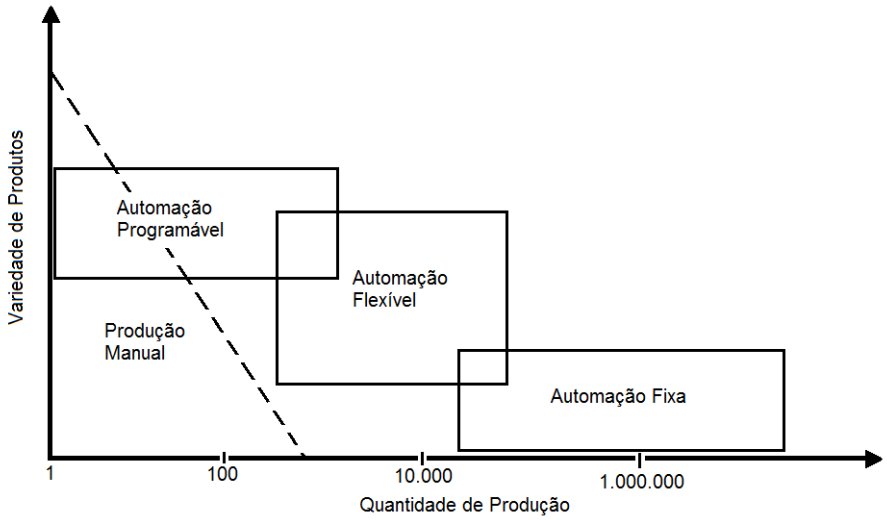


Figura 1. Os três tipos de automação comparados em relação variedade de produtos a quantidade de produção (em unidade de produto) (GROOVER, 2001).

2.1.1 Automação Flexível

No final dos anos 60 iniciou-se o uso da automação flexível em sistemas para a realização de operações de usinagem. Já no final dos anos 70 houve a difusão deste tipo de automação com o uso de robôs industriais, de sistemas de visão computacional e de sistemas flexíveis de manufatura, principalmente nos países desenvolvidos (GROOVE, 2001). Os principais fatores do crescimento desta tecnologia foram (TCHIJOV, 1989):

1. Mudança do paradigma de produção contínua e em grande quantidade para produção flexível em lotes;
2. Aumento da importância da qualidade nos itens produzidos aliada a produção de peças de maior complexidade, exigindo a substituição do controle humano pelo controle do computador;
3. Resistência social dos trabalhadores dos países desenvolvidos em relação a tarefas monótonas, trabalhos repetitivos, típicos da produção em massa. Isto gerou, em países como o Japão, um aumento considerável dos salários para trabalhos com pouca qualificação;
4. Diminuição do custo dos equipamentos para a realização da automação e aumento do custo da mão de obra.

2.2 Robótica

A robótica é a ciência ou o estudo da tecnologia associado com o projeto, fabricação, teoria e aplicação de robôs. Sendo que, o robô é um dispositivo automático utilizado para a realização de tarefas e possui conexões de realimentação (*feedback*) entre seus sensores, atuadores e o ambiente. Os robôs podem ter nenhuma, parcial, ou total ação de controle humano (FU; GONZALEZ; LEE, 1987). A evolução tecnológica e o surgimento de novas necessidades sociais têm ajudado no desenvolvimento de robôs para novas aplicações. Atualmente, os robôs são classificados em dois tipos básicos (WORLD ROBOTICS, 2012):

1. **Robôs de serviço**: São os robôs utilizados para substituir atividades humanas em ambientes não industriais. Podem ser robôs de serviço de uso profissional,

para aplicação na defesa, agricultura, logística, medicina, construção, plataforma móvel, limpeza, inspeção, subaquático, resgate e segurança, outros e os robôs de serviço de uso doméstico ou pessoal para aplicação de trabalho doméstico, entretenimento e lazer, outros.

2. **Robôs industriais**: São os robôs utilizados para substituir atividades humanas em ambientes industriais. São também chamados de manipuladores robóticos e realizam as atividades de movimentação de peças, soldagem, pintura, montagem, entre outras.

A Figura 2 e a Figura 3 ilustram, respectivamente, os robôs de serviço e os robôs industriais.

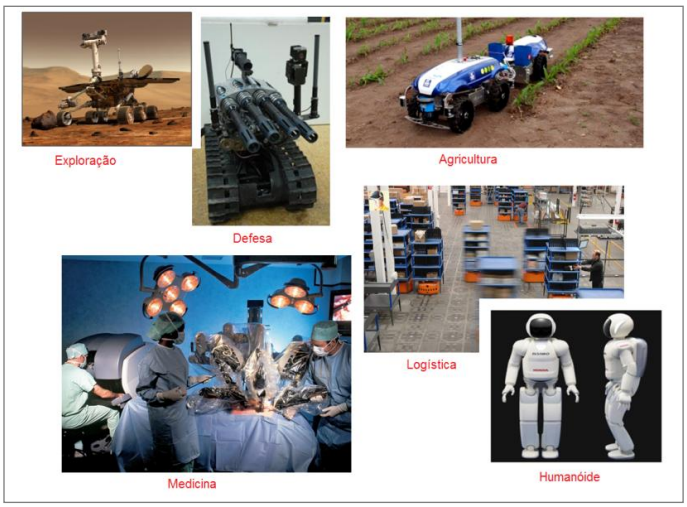


Figura 2. Robôs de Serviço.

Figura 3. Robôs industriais.

2.2.1 Robôs Industriais

Os robôs industriais são máquinas multifuncionais e reprogramáveis que podem executar tarefas normalmente associadas a seres humanos, possuindo também a capacidade de identificar alterações nas condições e restrições de tarefas e do ambiente, decidir as ações a serem realizadas e planejar a sua execução (PIRES, 2002). Outra definição é a de máquina manipuladora, com vários graus de liberdade, controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial (ANSI/RIA 10218-1, 2007). Atualmente, existem várias motivações para o uso de robôs nos processos de produção industrial, sendo (PAZOS, 2002) (AGUIAR, 2009):

1. **Redução de custo na produção:** O custo de um robô industrial amortizado ao longo da vida útil é menor que o custo do trabalho de um operário, incluindo encargos trabalhistas e diversos benefícios que aumentam o valor de homem-hora de trabalho.
2. **Melhoria da produtividade:** Os robôs industriais trabalham mais rápido que operários mantendo a exatidão, a repetitividade, a qualidade e menor perda por produtos defeituosos.
3. **Operação em ambientes hostis ou com materiais perigosos**: Podem operar em ambientes insalubres e classificados.
4. **Facilidade de integração**: Capaz de ser integrado em tempo real com sistemas de informação.

As principais características dos robôs industriais são: configuração física (partes do robô); graus de liberdade (GDL); espaço de trabalho; resolução; exatidão; repetitividade; carga e tamanho; tipo de articulação. O robô industrial é constituído pela sua configuração física que é uma cadeia serial de membros rígidos (**elos**) conectados por meio de articulações (**juntas**) e movimentados por ação de atuadores, sensores e sistema de controle computadorizado. O último elo possui o órgão terminal, ou *Tool Center Point* (TCP), local que são acoplados dispositivos (*end-effector*) para realização de tarefas. A Figura 4 ilustra algumas partes da configuração física.

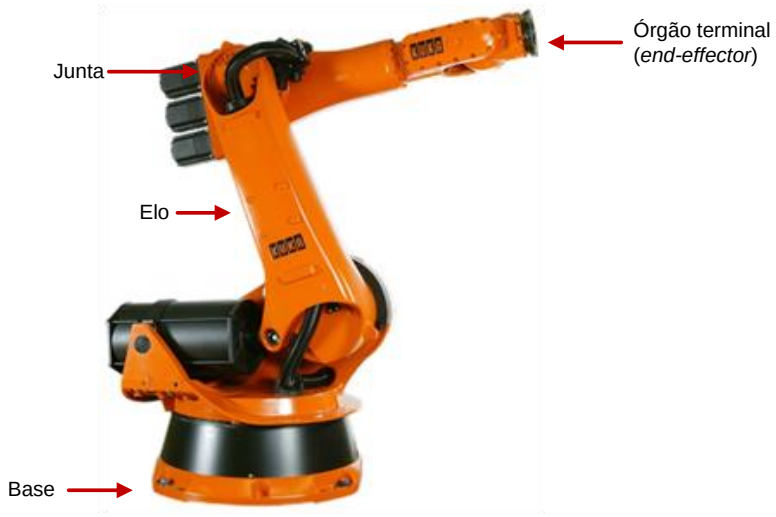


Figura 4. Partes do robô industrial (KUKA KR 210).

As juntas podem ser do tipo: **Prismática** ou Translação ou deslizante (P): movimento linear entre dois elos; **Revolução** ou rotativas (R): movimentos de rotação entre dois elos; **Bola e encaixe**: movimentos em torno dos três eixos cartesianos (x,y,z) entre dois elos. A Figura 5 apresenta os três tipos de juntas.

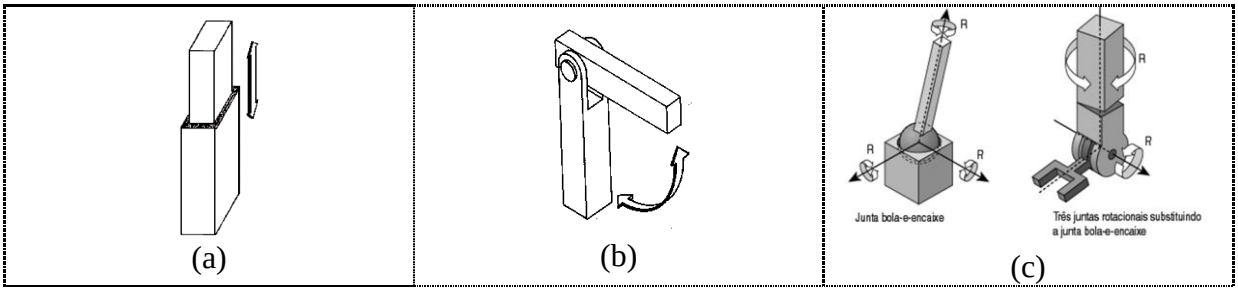


Figura 5. Tipos de juntas: (a) Prismática, (b) revolução e (c) bola e encaixe (ROSARIO, 2005).

Os robôs industriais são classificados em cinco classes ou combinação dos tipos de juntas (ROSARIO, 2005): Robô de coordenadas cartesianas (com três juntas tipo PPP); Robô de coordenadas cilíndricas (RPP); Robô de coordenadas polares ou esféricas (RRP); Robô SCARA, do inglês *Selective Compliance Arm for Robotic Assembly* (RRP) e Robô de coordenadas de revolução ou articulado (RRR), chamado de antropomórfico. A Figura 6 ilustra um robô industrial, tipo antropomórfico, KUKA KR 210 e suas possibilidades de movimento.

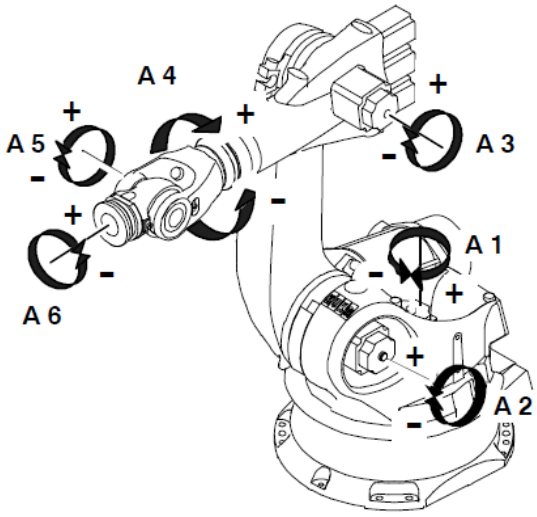


Figura 6. GDL do robô industrial (KUKA KR 210).

O sistema de coordenadas ortogonal de um robô industrial é representado pelas informações da posição (translação) nos eixos x, y e z (*x*,*y*,*z*), e da orientação, rotação no eixo x, y e z (*α*,*β*,*γ*) pelos valores dos ângulos *α* (ou a) de rolamento (*roll*), *β* (ou b) de arfagem (*pitch*) e *γ* (ou c) de guinada (*yaw*) em torno de cada eixo. Este par de informações (posição e orientação) é chamado de postura, ou *frame*, e é ilustrado na Figura 7.

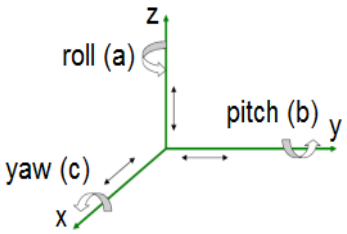


Figura 7. Ilustração do sistema de coordenadas ortogonal (ROSARIO, 2005).

Os sistemas coordenados descritos a seguir são de uso comum para se escrever movimentos de robôs industriais, o que facilita uma linguagem padrão (ADADE FILHO, 2001):

1. **Sistema coordenado da base {B}**: está localizado na base do robô industrial. Também é denotado por {0}, ou frame do elo 0;
2. **Sistema coordenado do pulso {P}**: está afixado no último elo de posicionamento do órgão terminal do robô industrial;
3. **Sistema coordenado do órgão terminal {M}**: está localizado no último elo do robô industrial. Também é denotado por {N}, ou frame do elo N. Em

algumas situações {M} também pode ter sua origem colocada no pulso do robô industrial.

4. **Sistema coordenado da ferramenta {F}**: está afixado no final de alguma ferramenta que o robô industrial utiliza. Quando o órgão terminal é uma garra e a garra estiver vazia, {F} é usualmente localizado com a sua origem entre as pontas dos "dedos" da garra, podendo coincidir com o frame {M}.

A cinemática do robô industrial é apresentada no Apêndice A.

2.3 Sistema de Visão Computacional

O sistema de visão computacional é capaz de capturar, tratar e processar a imagem desejada (objeto) e é amplamente utilizado na automação flexível. É formado por uma câmera, sistema de iluminação e unidade de processamento (COSTA NETO et al, 2008). A Figura 8 apresenta as ligações entre estes dispositivos.

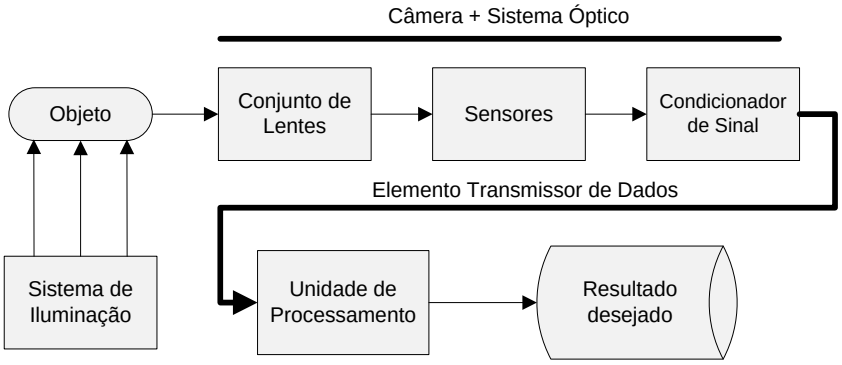


Figura 8. Estrutura de componentes do sistema de visão computacional (COSTA NETO et al, 2008).

O sistema de visão computacional descreve uma imagem como uma função p(x,y) formada por uma matriz de M linhas e N colunas, onde cada elemento de imagem é chamado de pixel (*picture element*) e contem um valor discreto associado a coordenadas de posição (x,y), que representa um determinado nível de intensidade luminosa ou de cor percebida pelo sensor de uma câmera. As imagens podem ser classificadas como binárias (*black & white*), em níveis de cinza (*gray scale*), coloridas indexadas (*indexed*) ou RGB (*truecolor*) (FU; GONZALEZ; LEE, 1987) (GONZALEZ; WOODS, 2007). Os componentes do sistema de visão computacional são apresentados no Apêndice B.

2.4 Sistema Flexível de Manufatura

O sistema flexível de manufatura (SFM ou FMS, do inglês *Flexible Manufacturing System*) é uma combinação de equipamentos autônomos (como exemplo, robôs industriais) e sistemas computacionais independentes (módulos) (como exemplo, sistema de visão computacional) integrados entre si, atuando de forma planejada, sequenciada e coordenada na produção de bens de serviço e consumo. O SFM oferece um desempenho de alta produtividade, com capacidade de respostas de forma rápida e econômica a mudanças no ambiente operacional (BARROS, 2006) e são caracterizados por um baixo custo de produção, alta qualidade, estoque mínimo e flexibilidade para alteração do produto final (NAKASHIMA; GUPTA, 2003).

A flexibilidade proporciona a capacidade de tomada de decisão sobre os recursos e a sequência de tarefas na produção de bens e respostas rápidas a mudanças imprevistas e/ou instabilidades causadas pelo ambiente e/ou sistemas, evitando assim possíveis paradas ou quebras de máquinas com o objetivo de aumentar ao máximo a utilização das máquinas.

O SFM possui algumas particularidades que o diferencia dos demais sistemas de manufatura, tais como (BARROS, 2006):

1. Flexibilidade de máquina: capacidade de mudar rapidamente da produção de um tipo de peça para outro.
2. Flexibilidade de processo: capacidade dos sistemas de variar os passos necessários para completar uma tarefa.
3. Flexibilidade de produto: capacidade para mudar rápida e economicamente de um produto para outro.
4. Flexibilidade de roteiro: capacidade do sistema de mudar a sequência de visitação de máquinas (no caso da quebra de uma delas) e continuar produzindo peças. Esta capacidade é devida à existência de diversos roteiros de produção ou ao fato de que uma operação pode ser realizada por mais de uma máquina.
5. Flexibilidade de Volume: capacidade de operar economicamente em diversos volumes de produção.
6. Flexibilidade de Operação: capacidade para mudar a ordem das operações de produção na fabricação de um mesmo produto.

Quanto maior for à flexibilidade do SFM, maior a complexidade para integração, a supervisão e o controle, ou seja, a elaboração do projeto, especificação técnica, planejamento e gerenciamento da montagem (SANTOS, 2008).

O SFM utiliza um conjunto de estações de trabalho flexíveis e quando estas estações possuem robôs industriais, é chamada de célula de manufatura robotizada (CMR). (BARROS, 2006). A Figura 9 ilustra uma CMR, que apresenta dois robôs industriais interconectados por meio de um sistema de armazenagem e transporte automático com sistema de supervisão para a decisão de cada momento (quando) o que deve ser feito (o que) e sobre que máquina (onde).

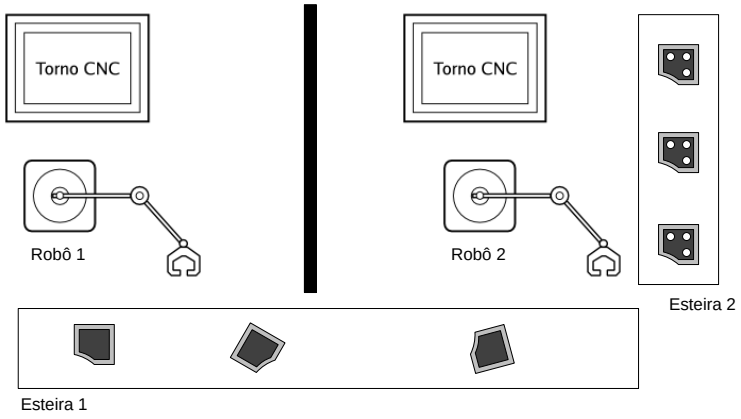


Figura 9. Ilustração do CMR.

2.4.1 Célula de Manufatura Robotizada

A Célula de Manufatura Robotizada é um tipo de SFM, por isso, também é chamado de célula flexível de manufatura robotizada. É um sistema complexo integrado, multifuncional e reprogramável, que gerencia a execução das diversas tarefas, por meio de regras de coordenação e sequenciamento (ROSARIO, 2010). A CMR deve ter a capacidade de identificar as restrições oriundas das tarefas e/ou do ambiente e de realizar ações para a continuidade da operação.

2.4.1.1 Programação de Célula de Manufatura Robotizada

A atividade de programação (ou ensino) da CMR é uma intervenção tanto no meio físico (dispositivos de controle e interface homem-máquina), quanto no meio lógico (sistemas supervisórios e aplicativos de software dedicados) para ajustes e definições de execução das diversas tarefas necessárias para a realização funcional desejada. Normalmente, a complexidade da programação da CMR está diretamente relacionada à capacidade de obtenção correta das trajetórias e do posicionamento dos robôs industriais que compõem o sistema.

No que se refere à programação de robôs industriais, existem duas abordagens comumente utilizadas: a programação por aprendizagem (programação *on-line*) e a programação por meio de algoritmos computacionais (programação *off-line*) (ADADE FILHO, 2001), que subdividem em quatros tipos, mostrados na Tabela 1.

Tabela 1. Tipos, Métodos e Características de programação de robôs industriais.

Tipo	Método	Características
Programação manual	• Programação *on-line*; • A programação é feita realizando a sequência de execução das tarefas que definem o comportamento da CMR e gravando os pontos finais de cada movimento.	• Movimento aleatório, apenas respeitando os pontos finais; • Tarefas repetitivas (Ex.: pintura, limpeza, soldagem, etc.); • Necessidade de parada do processo produtivo para execução da programação; • Dificuldade de incorporar mecanismos de tomada de decisão (instrumentação, sistemas inteligentes, etc.);
Programação ponto a ponto	• Programação *on-line*; • A programação é feita realizando a sequência de execução das tarefas que definem o comportamento da CMR operando de acordo com o desejado, gravando alguns pontos durante a trajetória.	• Movimento linear (ou circular, caso seja solicitado); • Tarefas repetitivas (Ex.: pintura, limpeza, soldagem, etc.); • Necessidade de parada do processo produtivo para execução da programação; • Dificuldade de incorporar mecanismos de tomada de decisão (instrumentação, sistemas inteligentes, etc.). •

Programação pela geração de algoritmo computacional	• Programação *off-line*; • A programação é realizada por meio de criação de algoritmo computacional que definem o comportamento desejado pela CMR.	• Sem necessidade de parada do processo produtivo; • Tarefas repetitivas, ou não; • Incorpora mecanismos de tomada de decisão (instrumentação, sistemas inteligentes, etc.); • Utilização de sistemas supervisórios e de controle; • Facilidade de detalhamento do fluxo lógico utilizado no programa;
Programação Simulada e geração de algoritmo computacional	• Programação *off-line*; • A programação é realizada por meio de simulação em ambiente computacional da sequência de execução das tarefas que definem o comportamento da CMR e, posterior, geração de algoritmo computacional.	• Sem necessidade de parada do processo produtivo; • Tarefas repetitivas; • Dificuldade de incorporar mecanismos de tomada de decisão (instrumentação, sistemas inteligentes, etc);

A programação *off-line* consiste na utilização de uma linguagem de programação para definição de um programa correspondente ao comportamento desejado do robô (ADADE FILHO, 2001). Esta abordagem é cada vez mais utilizada para a geração do programa dos robôs industriais na CMR. Porém, possui algumas limitações que dificultam a geração do programa, sendo:

1. **Desvios incondicionais (*go to*)**: dificuldade no controle da execução do programa e no uso de critérios de programação estruturada;
2. **Sem portabilidade de código fonte:** cada fabricante de robô possui uma linguagem própria de programação;
3. **Linguagens de baixo nível**: aumento de complexidade na geração do programa;

Quanto menor a complexidade para a geração do programa, menor a possibilidade de erros na programação da CMR. Assim, a programação no nível de tarefas é o que proporciona a melhor intervenção humana com menor possibilidade de erros (GINI, 1987).

2.4.1.1.1 Programação no Nível de Tarefas de CMR

Dois trabalhos (GINI, 1987) (HOLMBOM et al., 1993) classificam a complexidade de programação em níveis e apresentam uma solução para este problema.

O primeiro trabalho (GINI, 1987) propõe quatro níveis de complexidade, sendo: de junta, de manipulador, de objeto e de tarefas. Já o segundo trabalho (HOLMBOM et al., 1993) sugere três: físico, de controle e de tarefas. Nestas propostas, o primeiro e o último nível, respectivamente, representam o maior e a menor exigência de inteligência humana na programação. Para os dois trabalhos, o nível de tarefas é o que proporciona a melhor intervenção humana com menor possibilidade de erros.

A programação no nível de tarefas é realizada em interface amigável e de fácil interpretação, que possui recursos computacionais para a implementação dos algoritmos e comunicação direta com os sensores e atuadores do sistema (GINI, 1987). Cada instrução no nível de tarefas representa um conjunto de comandos de instruções no nível de junta (GINI, 1987), ou físico (HOLMBOM et al., 1993), que são em linguagem de máquina.

Também são utilizados recursos computacionais que torna abstrata e fácil à atividade de programação. Estes recursos possuem interface direta com os sensores e acionamentos da CMR. Os comandos neste nível incluem várias instruções de fácil interpretação que representam um conjunto de comandos (algoritmo computacional) de baixo nível de programação.

Um exemplo de instrução do nível de tarefas seria: Montar “produto A”. Depois de digitada, a sequência para a montagem do “produto A” é realizada, obedecendo todas as regras e especificações de entrada do operador e de acordo com o sistema supervisório e de controle. É também necessário apresentar uma solução que permita a reconfiguração do sistema de forma fácil e rápida, pois o uso de robôs industriais na automação flexível aumenta a complexidade de reconfiguração e torna o sistema susceptível a erros, devido à exigência de intervenção humana e a complexidade na programação de percursos e sequências de tarefas (GINI, 1987).

2.5 Projeto de Sistema Modular

De um modo geral, um sistema é uma parte limitada do Universo, delimitada por fronteiras. A interação do sistema com o mundo externo ocorre por meio de envio e recepção de estímulos, denominados eventos. Além disso, eventos internos à fronteira podem ocorrer, causando reorganização do sistema. Em qualquer caso, essas mudanças são caracterizadas 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 (CURY, 2001).

Já a modularidade é a atividade de dividir o sistema em unidades (módulos). Estas unidades representam funções que possuem elevado grau de independência e fronteiras bem definidas. Elas podem conter uma vasta gama de conteúdos de valor agregado e complexidade que vão desde simples execuções de funções aritméticas de soma e subtrações, até complexas funções e interações de equipamentos para o controle dos movimentos de um manipulador industrial (BALDWIN; CLARK, 2000).

O projeto de sistema modular é uma estratégia para a construção de processos ou produtos complexos a partir de módulos que podem ser desenvolvidos individualmente, mas que possibilitem uma integração entre eles. Os módulos são definidos de acordo com os requisitos de projeto, tais como: arquitetura do sistema, interfaces externas e internas do sistema, necessidades operacionais, de qualidade e de segurança, entre outros. A definição da fronteira entre os módulos deve ser exata, sem ambiguidade e completa, senão o conceito de projeto de sistema modular perde seus efeitos.

2.6 Modelo Espiral para Levantamento de Documento de Requisitos

A construção de um SFM exige, na etapa inicial, a definição documentada das propriedades ou comportamentos desejados (requisito) do sistema. Esta é uma tarefa investigativa, criativa e contínua, pois no início do desenvolvimento, ninguém realmente sabe o que o SFM deve fazer ao ficar pronto, inclusive o cliente. É, portanto, essencial que o

método utilizado permita expressar sem ambiguidades os requisitos, bem como validar e verificar suas consistências.

Neste contexto, é apresentado um método para levantamento da documentação de requisitos, chamado de modelo espiral (KOTONYA; SOMMERVILLE, 1998) que está ilustrado na Figura 10. Consiste de um processo iterativo, sistemático e disciplinado de levantamento de necessidades e de funcionalidades de um sistema, com a finalidade da geração de documentação de especificação dos requisitos. Este método possui quatro etapas: elicitação; análise; especificação e validação dos requisitos. (KOTONYA; SOMMERVILLE, 1998).

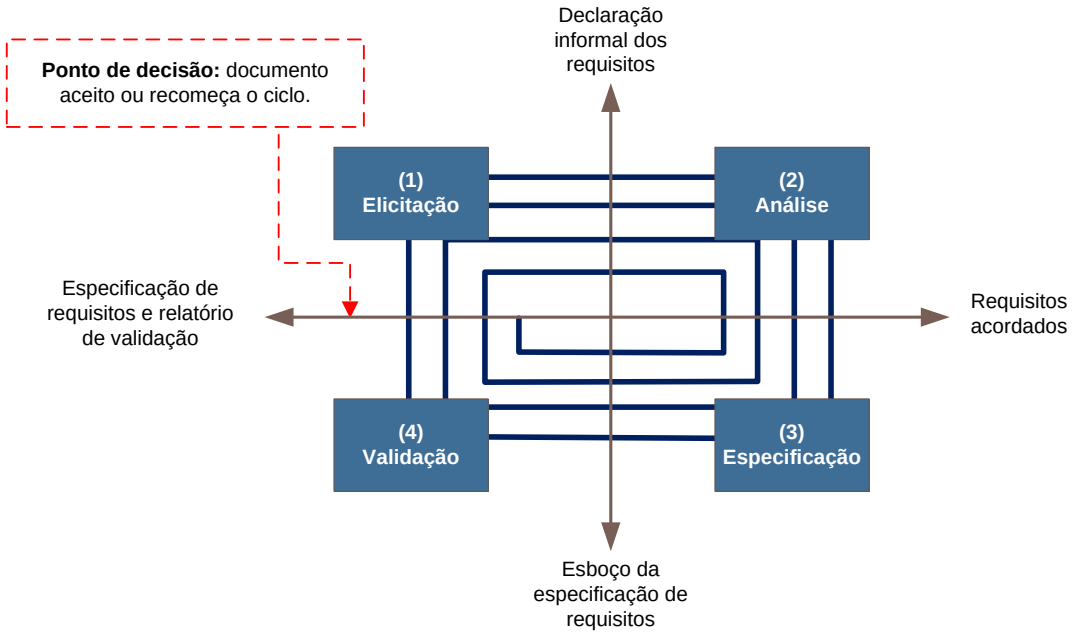


Figura 10. Modelo espiral para geração de documento de requisitos (KOTONYA; SOMMERVILLE, 1998).

2.6.1 Elicitação

Nesta etapa, busca-se obter informações sobre as necessidades, comportamento ou características que o usuário deseja de um módulo ou sistema como um todo. Os usuários e desenvolvedores trabalham em conjunto para definir o problema a ser solucionado, enfocando principalmente os serviços que o sistema deve oferecer.

Porém, é comum os usuários não saberem exatamente o que realmente querem da SFM, isso pode fazer com que os requisitos definidos inicialmente não reflitam o

comportamento real desejado. Deste modo, não basta apenas realizar entrevistas ao usuário, mas também, é exigido uma análise cuidadosa do domínio da aplicação, da empresa e dos processos de negócios (BARROS, 2006).

Desta forma, a elicitação de requisitos, realizada de forma efetiva, deve abordar quatro dimensões: Entendimento do domínio da aplicação; Entendimento do problema; Entendimento do negócio e Entendimento das necessidades e das restrições dos *"stakeholders*" (é uma referência ampla para cliente, comprador, operador, ou usuário) - na qual se dá o entendimento detalhado: das necessidades de apoio a serem providas pelo sistema à realização do trabalho e aos interesses de cada um dos *"stakeholders"*; dos processos de trabalho a serem apoiados pelo sistema; e do papel de eventuais sistemas existentes na execução e condução dos processos de trabalho (KOTONYA; SOMMERVILLE, 1998).

Há diversas técnicas para elicitação dos requisitos (MARTINS, 2001) e pode-se usar uma ou mais de uma técnica para elicitação de requisitos em SFM. A Tabela 2 apresenta um resumo de algumas destas técnicas.

Tabela 2. Resumo de algumas técnicas de elicitação de requisitos.

Técnica	Descrição	Vantagem	Desvantagem
Introspecção	A introspecção é o ato de "imaginar" como deveria ser um sistema para atender a um determinado problema. Ela leva em consideração um ponto de vista particular, por exemplo, o ponto de vista do desenvolvedor do sistema, que procura imaginar que tipo de sistemas ele desejaria ter para realizar as tarefas que o usuário real irá executar.	Permite a elaboração de uma concepção de sistema rapidamente.	Está fortemente vinculada a experiência do desenvolvedor em relação ao problema a ser resolvido.
Observação	O principio básico desta técnica é obter informações a partir da observação de tarefas executadas pelo usuário (ou por um grupo de usuários), com um mínimo a de interferência da parte do desenvolvedor.	Os requisitos podem ser obtidos sem interferência de comunicação, uma vez que estarão sendo percebidos pelo observador no ambiente do próprio usuário. A observação ocorre no cotidiano do usuário, onde as situações de trabalho (rotinas) que envolvem as pessoas que vão fazer uso do sistema emergem naturalmente.	O período de observação para cobrir as principais rotinas de trabalho do usuário pode ser longo, uma vez que muitas situações importantes podem ocorrer apenas ocasionalmente.
Questionário	O questionário é um instrumento que pode ser útil na elicitação de requisitos quando se deseja obter informações de um grande número de pessoas. Outro aspecto interessante do uso de questionários, é que eles podem ser preparados de tal forma que suas respostas possam ser facilmente tabuladas, possibilitando uma análise estatística dos cabos obtidos.	Pode trazer informações pertinentes ao sistema estudado vinda de um grande número de pessoas envolvidas no mesmo. Facilita uma análise estatística das informações obtidas.	O grau flexibilidade que o questionário dá para as respostas dos envolvidos muitas vezes é pequeno, tendendo a ser rígido, dificultando a obtenção de informações mais subjetivas. Muitas vezes as pessoas não se sentem à vontade de colocar suas opiniões por escrito.
Entrevista	A entrevista é uma técnica muito comum utilizada em processos de elicitação de requisitos. Existem basicamente dois tipos de entrevista: entrevista fechada e entrevista aberta. Estes dois tipos de entrevista, normalmente, ocorrem de forma individual e privada.	É flexível na obtenção das informações, sendo adequada para se obter informações de caráter subjetivo. Aproxima o engenheiro de requisitos do usuário do sistema, fazendo com que o usuário se sinta participativo no processo de	Pode demandar muito tempo se muitas pessoas precisarem ser entrevistadas, uma vez que a entrevista normalmente ocorre de forma individualmente ocorre de forma individual, com cada usuário do sistema. A tabulação das informações

		desenvolvimento do sistema.	obtidas a partir das entrevistas, principalmente na de tipo aberto, costuma ser trabalhosa.
Análise de Protocolo	A análise de protocolo é uma técnica de elicitação de requisitos que pode ser aplicada a partir da narração de uma tarefa realizada pelo usuário. Enquanto o usuário está executando a tarefa ele vai aflando o que está fazendo. Os proponentes desta técnica acreditam que este processo pode ser considerado com uma verbalização direta de processos cognitivos específicos.	Duas formas de comunicar informações sobre o sistema ocorrem simultaneamente, o fazer e o falar o que está fazendo. Requisitos podem ser capturados e ajustados comparando as duas formas de comunicação. O fazer pode completar o falar e vice-versa.	O usuário pode ser sentir inibido de falar enquanto faz, podendo também se desconcentrar no que está fazendo enquanto tenta explicar verbalmente.
Prototipação	Construção de um protótipo do sistema a ser implementado. O protótipo é frequentemente usado para elicitar e validar requisitos do sistema. Um requisito essencial para um protótipo, é que deve ser possível desenvolvê-lo de forma rápida, tal que ele possa ser utilizado durante o processo de Engenharia de Requisitos. O protótipo não é o sistema real, mas apenas uma simulação do sistema que será construído de fato.	O protótipo oferece uma ajuda efetiva na comunicação entre usuário e engenheiro de requisitos, devido a seu forte apelo visual.	Existe o risco de se usar a estrutura do protótipo como base de implementação da primeira versão do sistema.
Casos de Uso	A utilização de casos de uso para elicitação de requisitos do sistema vem recebendo interesse crescente da comunidade de Engenharia de Requisitos, e sendo bem aceita por muitos metodologistas. Esta abordagem enfatiza a importância da captura dos requisitos do sistema do ponto de vista dos atores que interagem com ele.	Uma vez identificados os casos de uso, os requisitos do sistema estão praticamente definidos. Oferece uma estrutura de organização para o processo de elicitação e análise dos requisitos. Os requisitos podem ser apresentados para o usuário por meio de diagramas de fácil entendimento, o que facilita o processo de comunicação entre o usuário e engenheiro de requisitos.	Cada caso de uso precisa ser descrito detalhadamente em separado, uma vez que sua representação pictórica com o diagrama de casos de uso oferece poucas informações sobre os requisitos.

2.6.2 Análise e Negociação

A etapa de Análise e Negociação destina-se a resolver problemas relacionados à declaração informal dos requisitos obtidos na etapa anterior. Na maioria, são ocasionados: pelas dificuldades na representação do conhecimento obtido de mais de um *"stakeholder"* (NADDEO, 2002); por inevitáveis conflitos entre as diferentes origens dos requisitos; por informação incompleta; e pelo alto custo no uso do requisito na construção do sistema.

Neste momento, os requisitos são detalhados e diferentes *"stakeholders"* negociam para decidir quais serão aceitos, sendo gerados os requisitos dos usuários. Há sempre alguma flexibilidade nesta etapa, no qual é acordado o conjunto de requisitos (KOTONYA; SOMMERVILLE, 1998). Existem três elementos fundamentais sobre os requisitos, que devem ser verificados na análise auxiliando na identificação de problemas (BARROS, 2006):

1. Verificação de Necessidade: a necessidade de um requisito obtido deve ser analisada. Alguns requisitos podem ser propostos, mas não contribuem com as metas de negócio da organização ou para o problema específico a ser endereçado pelo sistema.
2. Verificação de consistência e completitude: consistência significa que nenhum requisito deve ser contraditório, e completitude significa que nenhum serviço ou restrição que sejam necessários ao sistema tenha sido omitido.
3. Verificação de Possibilidade: os requisitos devem ser verificados no sentido de assegurar que eles são factíveis de serem implementados no sistema a ser desenvolvido tanto em termos de orçamento como de tempo.

2.6.3 Especificação

Uma vez identificados e negociados, os requisitos devem ser documentados para utilização no projeto de sistema modular. Nesta etapa, os requisitos do usuário são refinados de acordo com as características exigidas para o sistema, incluindo equipamentos e programas computacionais. Sendo detalhados em termos de engenharia (requisitos de sistema).

Estes requisitos do sistema devem ser apresentados como:

1. Requisitos Funcionais: Representam algo que o sistema deve fazer, ou seja, uma função esperada do sistema que agregue valor aos seus usuários. São definidos pelos eventos internos e externos críticos do sistema e pelas informações que o cliente deseja obter do sistema, ou seja, as respostas fundamentais do sistema.
2. Requisitos não Funcionais: Representam as definições sobre as propriedades e restrições do sistema, ou seja, indicam como os requisitos funcionais devem ser alcançados, tais como exigências de qualidade, como desempenho e robustez, ou controle de custo do projeto, até limitações de uso de uma ou outra tecnologia e restrições de projeto. A Figura 11 apresenta os fatores que podem definir um requisito não funcional.

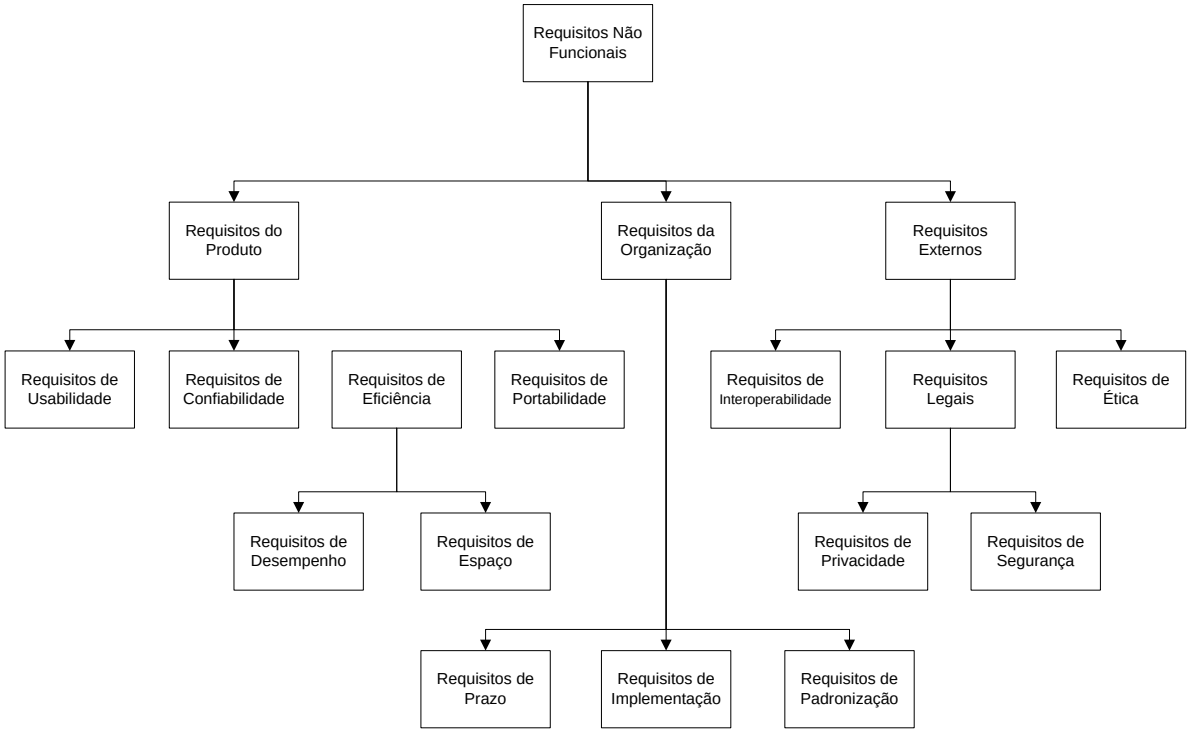


Figura 11. Tipos de requisitos não funcionais

Existem outros padrões para especificação de requisitos, como os propostos pelo IEEE, sendo eles: (IEEE STD 830, 1998) que descreve as etapas recomendadas para a especificação de requisitos de *software*, é baseado em um modelo onde o resultado do processo de especificação de requisitos é claro e o documento de especificação é completo; (IEEE STD 1233, 1998) tem o propósito de prover orientação para a captura de "bons" requisitos, de forma clara e bem estruturada e de como organizá-los.

O documento de requisitos gerado nesta etapa possui alguns benefícios, sendo o veículo básico de comunicação entre desenvolvedores e usuários sobre o que deve ser construído, registra os resultados da análise do problema (obtido por meio da elicitação e análise dos requisitos), define quais propriedades o sistema deve ter e quais são as restrições impostas em seu projeto e implementação, é a base para estimativas de custo e cronograma, é a base para o desenvolvimento do plano de teste do sistema, oferece uma definição padrão de comportamento esperado pelos profissionais envolvidos na manutenção do sistema e é utilizado para registrar mudanças na engenharia do sistema. (FAULK, 2007). Este documento de requisitos deve estar numa linguagem de fácil compreensão para os diferentes *"stakeholders"*.

O documento de requisitos deverá apresentar seção 1 (Introdução): Visão geral do documento e convenções, termos e abreviações com a identificação dos requisitos e

prioridades dos requisitos, seção 2 (Descrição geral do sistema): Apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e descrevendo seus usuários, seção 3 (Requisitos funcionais): Especifica todos os requisitos funcionais do sistema, descrevendo os fluxos de eventos, prioridades, atores, entradas, saídas e recursos necessários, seção 4 (Requisitos não funcionais): Especificam todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição. Adequação a padrões e descrição de *hardware* e *software* e Seção 5 (Descrição da interface com o usuário): Apresenta desenhos, figuras ou rascunhos de telas do sistema.

2.6.4 Validação

Nesta etapa são realizadas diversas revisões que tem a finalidade de detectar problemas no documento de requisitos e garantir a consistência e integridade, antes que estes sejam utilizados como base no desenvolvimento do sistema. A validação procura certificar que os requisitos acordados e especificados representam uma descrição aceitável do sistema a ser construído (KOTONYA; SOMMERVILLE, 1998). A revisão é a principal técnica utilizada na validação de requisitos (NADDEO, 2002).

As revisões consistem de reuniões estruturadas, nas quais um grupo de pessoas (usuários e desenvolvedores), prévia e cuidadosamente escolhidas, discute os problemas, omissões e ambiguidades encontradas, após lerem e analisarem o documento de requisitos e chegam a uma solução de consenso em relação às ações a serem adotadas para corrigi-los (NADDEO, 2002).

Os principais problemas descobertos durante a validação dos requisitos são de não atendimento a padrões de qualidade, requisitos descritos de forma ineficiente, levando a ambiguidade, erros na modelagem do problema ou sistema e requisitos conflitantes não identificados durante a etapa de análise (BARROS, 2006). Alguns problemas são resolvidos com a correção do documento de requisitos, no entanto, outros problemas podem levar a uma nova execução do modelo espiral, levando a uma nova elicitação, análise, negociação e especificação.

2.7 Diagrama IDEF0 para Definição de Módulos Conceituais

As definições das fronteiras entre os módulos que compõem um SFM, devido à complexidade, podem oferecer dificuldades, tendo como consequência a construção de modelos de reduzida representatividade. É necessária a utilização de técnicas que apoiem o modelador na abstração do sistema e definição dos módulos permitindo a descrição detalhada do funcionamento do sistema e favorecendo o processo de criação do modelo conceitual de cada parte (RYAN; HEAVEY, 2006) (ZACHMAN, 1987) (CASSANDRAS, 2008).

Neste aspecto, vários pesquisadores propõem a integração utilizando o diagrama IDEF0 (*Integrated Computer Aided Manufacturing DEFinition Methodology*) para a modelagem de módulos. Os diagramas IDEF correspondem a um conjunto de técnicas processuais utilizadas para auxiliar e padronizar a integração das informações disponíveis em um sistema, por meio de uma linguagem de modelagem gráfica (sintaxe e semântica). Esta medida propiciou o aumento, sistemático, da capacidade de produção, sobretudo no estabelecimento de controles, documentação e melhoria no desempenho das organizações (FIPS PUBS, 1993) (MENZEL; MAYER, 1998). As principais aplicações: Modelagem de tarefas para uso em Redes Petri utilizando o conjunto de abordagens do IDEF0 (BOSILJ-VUKSIC; GIAGLIS; HLUPIC, 2000). Criação de método para representação de informações *Fuzzy* utilizando abordagens de extensões do IDEF1X (MA; ZHANG; MA, 2002). Elaboração de modelos conceituais em simulações computacionais com o uso das técnicas de IDEF0 (LEAL et al., 2007).

Entre os diferentes diagramas IDEF, o diagrama IDEF0 é a versão amplamente utilizada para modelagem de uma variedade de sistemas (HERNANDEZ-MATIAS et al., 2006). Esta aplicação é bastante eficaz no auxílio da modelagem conceitual e, por se tratar de uma linguagem gráfica e textual, facilita a visualização das conectividades existentes entre as diversas funções de um sistema.

Desta forma, para o sistema modular em fase de projeto, o diagrama IDEF0 pode ser utilizado na definição e criação dos módulos, relacionando cada função do sistema a um módulo assíncrono do processo. O módulo criado é representado por uma “caixa-preta” e com setas que representam entradas e saídas de sinais, como apresentada na Figura 12. Esta caixa possui um nome (descrição funcional do módulo, por meio de verbo ou expressão verbal)

centralizado e um número (código sequencial de identificação) posicionado no canto inferior direito.

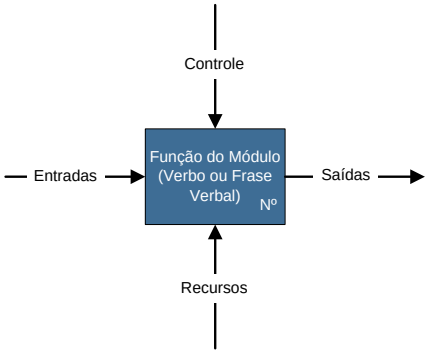


Figura 12. Representação Gráfica de um módulo utilizando diagrama IDEF0.

Os módulos são interligados por setas e este conjunto é chamado de diagrama ou representação gráfica do diagrama IDEF0 conforme Figura 13. As setas não representam o fluxo ou sequência como uma modelagem convencional de fluxo de processo, mas sim fluxo de dados ou sinais de funções a serem desempenhadas pelos módulos do sistema em representação.

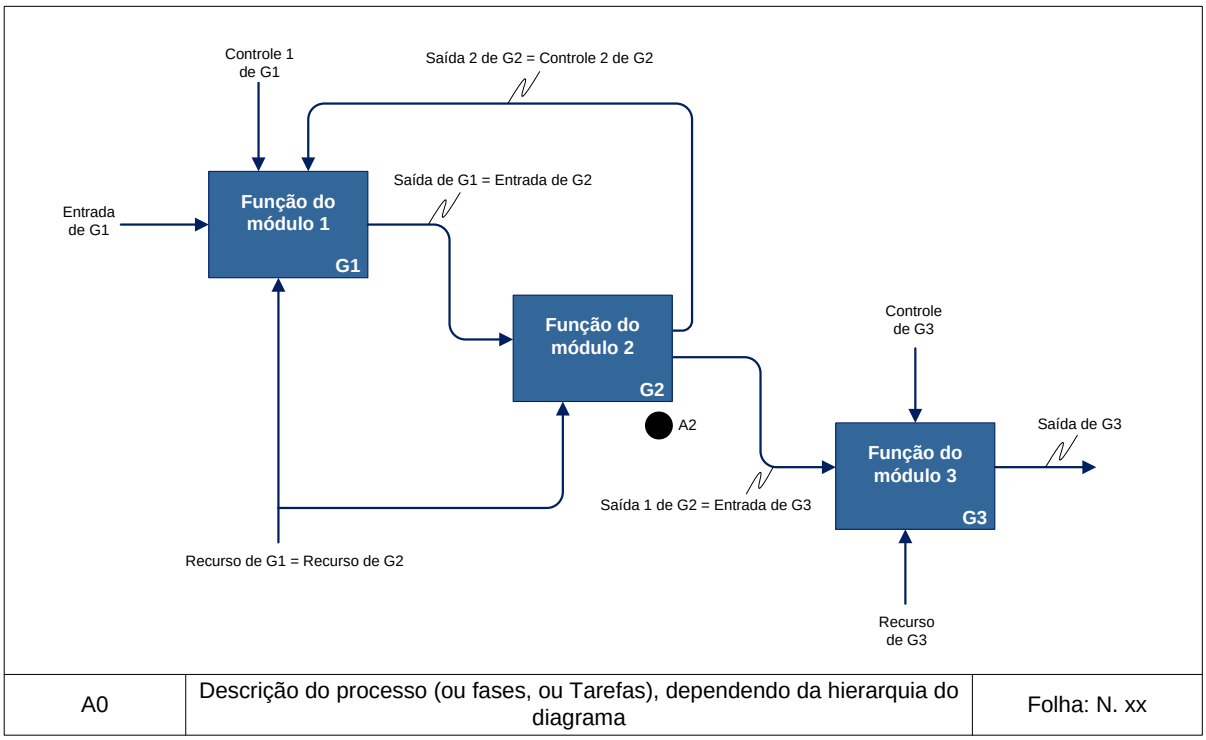


Figura 13. Exemplo de uma representação gráfica do diagrama IDEF0.

O sentido e o local em que as setas tocam as caixas representam para o lado esquerdo as entradas de dados ou sinais para execução da função do módulo. Já para o lado direito as saídas de dados e sinais resultantes da execução da função do módulo, o lado superior a entrada de sinais para o controle das condições requeridas de saída e lado inferior a entrada dos recursos que suportam a execução da função. O primeiro diagrama deve ter o identificador A-0 e apenas uma função (caixa) que identifica o processo. Em seguida, este diagrama é decomposto para a modelagem de todos os níveis do processo. Para cada diagrama são utilizados de três a seis módulos interligados. As funções devem ser dispostas na diagonal.

A modelagem utilizando o diagrama IDEF0 possui entre suas principais características a não representação do tempo, não tem preocupação em evidenciar a sequência das funções e sim a dependências entre elas. Existe ainda uma representação específica para tomada de decisão e não existe uma preocupação de ter uma estrutura hierárquica de processo.

2.8 Sistema a Eventos Discretos

O Sistema a Eventos Discretos (SED) são sequências ordenadas de eventos que levem à realização de determinados objetivos, sem admitir a ocorrência simultânea de dois ou mais eventos distintos (CASSANDRAS, 2008). Um sistema pode ser caracterizado pela natureza de suas variáveis de estado. As variáveis discretas inserem-se num conjunto enumerável de valores (ex.: números inteiros, variáveis booleanas). As variáveis contínuas têm seu valor definido dentro do conjunto de números reais e podem assumir infinitos valores durante um determinado intervalo de tempo (CASSANDRAS, 2008) (ALLA; DAVID, 1998 apud AGUIAR, 2009).

Os sistemas de variáveis contínuas dirigidos pelo tempo são descritos por equações diferenciais. Estes, em geral, mudam de estado continuamente, tendo o seu comportamento descrito por uma função que relaciona o estado (variável dependente) ao tempo (variável independente) (CASSANDRAS, 2008). Ao contrário, os sistemas de variáveis discretas evoluem de acordo com a ocorrência abrupta de eventos físicos, em intervalos de tempo em geral irregulares e desconhecidos. Diz-se ainda que, entre a ocorrência

de dois eventos consecutivos, o sistema permanece num determinado estado. A ocorrência de um evento causa então uma transição ou mudança de estado no sistema (CURY, 2001). O SED pode permanecer um tempo arbitrariamente longo em um mesmo estado, sendo que sua trajetória pode ser dada por uma lista de eventos, incluindo-se eventualmente os instantes de tempo em que tais eventos ocorrem. A quantidade de informação necessária depende dos objetivos da aplicação (CASSANDRAS, 2008).

Neste sentido, o SED é utilizado para o controle supervisório do sequenciamento e coordenação das tarefas da SFM. Até o momento, foram desenvolvidos vários modelos para SED, sem que nenhum tivesse se afirmado como universal. Esses modelos refletem diferentes objetivos na análise dos sistemas em estudo (como exemplo, SFM). Os principais modelos de SED são Redes de Petri Controladas (MURATA, 1989), Cadeias de Markov (ÇINLAR, 1975), Teoria das Filas (KLEINROCK, 1975) e a Teoria do Controle Supervisório (TCS) (RAMADGE; WONHAM, 1989) baseado nos Autômatos (CARROLL; LONG, 1989).

Em todas as abordagens, a geração do supervisório é pelo método da tentativa e erro, experiência e inspiração do projetista, com exceção para dois modelos, que apresentam característica particular de serem dotados de procedimento de síntese (criação automática por formalismo matemático) de sistemas supervisórios (ou, controladores). São as abordagens de TCS e as Redes de Petri Controladas, que utilizam modelos matemáticos das partes do SFM (robôs, sistema de visão computacional, outros) e suas especificações (requisitos do processo) para a geração do sistema supervisório.

2.8.1 Teoria de Controle Supervisório

Teoria de Controle Supervisório (TCS) é uma abordagem que permite modelagem gráfica do comportamento do sistema, enquanto, simultaneamente, introduz regras formais matemáticas que o definem. TCS é um dos métodos mais utilizados na modelagem de sistemas dinâmicos sequenciais, devido as suas características: simplicidade, poder de representação compreendendo concorrência, sincronização e compartilhamento de recursos, capacidade de análise matemática e aplicação de ferramentas de *software*.

Esta abordagem permite a construção de controles supervisórios que modificam o comportamento em malha aberta do sistema, por meio da eliminação de sequências

indesejadas de eventos, restringindo o seu comportamento à especificação desejada (RAMADGE; WONHAM, 1989). Também, atribui um maior grau de liberdade ao sistema controlado e impõe que o supervisório obtido sempre atenda as especificações de controle. Assim, quando as partes do sistema controlado são modificadas ou os objetivos de controle alterados, este modelo permite agilidade na elaboração do novo supervisório.

2.8.1.1 Definições Básicas

Na TCS o conjunto de todos os eventos possíveis do sistema é chamado de alfabeto (Σ). Já o comportamento deste sistema é modelado como uma sequência de eventos (*palavras*), no qual o conjunto destas palavras representa a linguagem (L), sendo representado por um modelo matemático de uma máquina de estados finitos, quando regulares, é chamada de autômato (CURY, 2001) (CASSANDRAS, 2008). Os eventos (σ) são classificados por (CASSANDRAS, 2008):

1. Eventos controláveis ($\sigma \in \Sigma_c$): Podem ser desabilitados pelo sistema supervisório;
2. Eventos não controláveis ($\sigma \in \Sigma_u$): Não podem ser desabilitados pelo sistema supervisório.

O autômato (G) é caracterizado por duas linguagens:

1. Comportamento fechado $L(G)$: Representam as sequências possíveis de eventos para o sistema;
2. Comportamento marcado $Lm(G)$: Representam as sequências possíveis de eventos que levam a finalização de uma tarefa.

O autômato pode ser representado por grafos direcionados, em que os nós representam os estados e os arcos rotulados representam os eventos. Os estados marcados são representados por círculos concêntricos e o estado inicial por um arco de entrada. Arcos com um traço representam eventos controláveis. A Figura 14 ilustra um autômato.

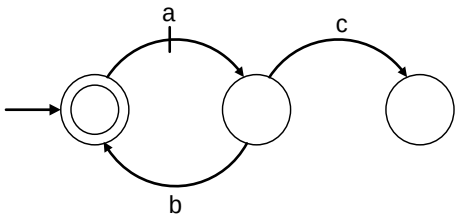


Figura 14. Representação gráfica de um autômato

2.8.1.2 Controle Supervisório Monolítico

O sistema a ser controlado, em geral, é um conjunto de módulos arranjados segundo uma distribuição conhecida. Estes módulos, vistos isoladamente, têm cada um, um comportamento básico original. Quando atuando em conjunto com os demais módulos, cada módulo deve ser restringido de forma a cumprir com a função coordenada a ser executada pelo sistema global. A composição dos comportamentos de cada módulo isolado pode então ser identificada com a planta G, com comportamentos gerados e marcados, respectivamente $L(G)$ e $Lm(G)$ (CURY, 2001).

Assume-se que G é modelado por um autômato. A notação G então será usada indistintamente para se referenciar à planta. O conjunto de restrições de coordenação define uma especificação E a ser obedecida. As linguagens $L(G)$ e $Lm(G)$ contem cadeias indesejáveis de eventos por violarem alguma condição que se deseja impor ao sistema. Pode ser o caso de haver estados proibidos em G por provocarem bloqueio ou por serem inadmissíveis, por exemplo, no caso de uma colisão de um robô com a mesa utilizada para montagem de componentes, de uma célula de manufatura robotizada.

De modo a fazer com que os módulos atuem de forma coordenada, introduz-se um agente de controle denominado supervisor, denotado por S. Considera-se que o supervisor S interage com a planta G, numa estrutura de malha fechada, em que S observa os eventos ocorridos em G e define que eventos, dentre os requisitos do sistema, são fisicamente possíveis de ocorrerem no estado atual (CURY, 2001) (CASSANDRAS, 2008).

Mais precisamente, S tem uma ação desabilitadora de eventos e, neste sentido diz-se que S é um controle de natureza permissiva. O conjunto de eventos habilitados num dado instante pelo supervisor define a entrada que afeta a planta G. A entrada é atualizada a cada nova ocorrência de evento observado em G (CURY, 2001). A planta G também é afetada

pelos eventos de natureza não controlável, ou seja, os eventos que S não tem ação desabilitadora.

Os cálculos para a síntese deste supervisório monolítico (S) exigem esforço polinomial, que crescem exponencialmente com o aumento dos estados, das especificações e da planta. Este é o principal fator que limita a aplicação desta técnica. Diversos autores apresentam propostas para superar as dificuldades computacionais. Dentre as quais, podem-se destacar a técnica de modularidade (RAMADGE; WONHAM, 1989) e de simetria (EYZELL; CURY, 1998). A técnica da modularidade, ou controle supervisório modular, adotada neste trabalho, apresenta-se como melhor solução para sistemas que envolvem uma grande quantidade de especificações.

2.8.1.3 Controle Supervisório Modular

De acordo com a TCS, a especificação global utilizada na construção do supervisório é a composição de todas as especificações locais. Porém, existe outra abordagem chamada de Controle Supervisório Modular, em vez de projetar um único supervisório (ou supervisório monolítico) para satisfazer todas as especificações, é projetado um supervisório (modular) para cada especificação local, de forma que, atuando em conjunto, os supervisores modulares satisfaçam todas as especificações (RAMADGE; WONHAM, 1989).

Esta nova abordagem permite desenvolver soluções simples para sistemas complexos, que facilmente podem ser alteradas. Em contrapartida, por possuírem ações de supervisão descentralizadas, podem permitir conflitos na ação de controle, caso não sejam verificadas as condições de modularidade. O exame da condição de modularidade é feito verificando se o conjunto de linguagens é não conflitante de acordo com a equação (1).

$$\bigcap_{i=1}^{n} \overline{Li} = \overline{\bigcap_{i=1}^{n} Li} \qquad (1)$$

Em que,

Li: É a linguagem de $i = 1, \ldots, n$ e n representa a última linguagem.

Isso quer dizer que, sempre que uma sequência de eventos repetitivos (também chamado de prefixo) for aceito por todo o conjunto de linguagens, a linguagem deve aceitar uma palavra contendo esse prefixo. (RAMADGE; WONHAM, 1989).

2.8.1.4 Sistema Supervisório Modular Local

A abordagem de controle de Sistema Supervisório Modular Local (SML) permite explorar a modularidade das especificações e a estrutura naturalmente descentralizada dos sistemas de manufatura automatizados, de forma a proporcionar um menor esforço computacional no processo de síntese e de implementação do supervisório (QUEIROZ; CURY, 2002b). No projeto de sistemas de maior complexidade, a modelagem das diversas partes envolvidas é geralmente um passo intermediário na representação do comportamento do sistema, especialmente quando é considerada a estrutura descentralizada do SFM. Estas partes são geralmente modeladas pela composição de módulos de menor porte ($G_i, i = 1, \ldots, n$), gerando subsistemas completamente assíncronos. Cada subsistema é denominado de Representação de Sistema Produto (RSP) ($G_{loc,i}, i = 1, \ldots, n$) (QUEIROZ; CURY ,2002a). Em seguida, é necessário obter as especificações locais $E_{loc,i} = G_{loc,i} || R_{loc,i}\ , i = 1, \ldots, n$ pela composição síncrona (que possui eventos comuns) das restrições locais ($R_{loc,i}, i = 1, \ldots, n$) e sua respectiva planta local ($G_{loc,i}, i = 1, \ldots, n$) para que a equação mostrada abaixo seja válida.

$$L_m(E_{loc,i}) \subset L_m(G_{loc,i}), i = 1, \ldots, n \quad (2)$$

Deve-se eliminar, caso houver, os estados proibidos (bloqueantes) de $E_{loc\_sp,i}, i = 1, \ldots, n$ e calcular a especificação local não bloqueante, chamado de componente *trim* $E_{loc\_trim,i}, i = 1, \ldots, n$. Para finalizar, é calculado a máxima linguagem controlável $S_{loc,i} = SupC(E_{loc,i}, G_{loc,i}), i = 1, \ldots, n$ pelo processo de síntese para cada linguagem das especificações locais $E_{loc,i}, i = 1, \ldots, n$. A ação de supervisão é atribuída a cada estado de $S_{loc,i}, i = 1, \ldots, n$, ou seja, desabilitando os eventos que não podem ocorrer nas RSP.

A máxima linguagem controlável $S_{loc,i} = supC\left(G_{loc,i}, E_{loc\_trim,i}\right), i = 1, \dots, n$ é a linguagem que mais se aproxima da especificação local não bloqueante ótima ($E_{loc\_trim,i}, i = 1, \dots, n$) e que não possua cadeia de eventos indesejáveis. Ou seja, a síntese de cada supervisor modular local que realiza, para cada planta local ($G_{loc,i}, i = 1, \dots, n$), a especificação local não bloqueante ótima ($E_{loc\_trim,i}, i = 1, \dots, n$) no sentido de menos restritivo possível e atenda a condição de controlabilidade.

O SML controla apenas o comportamento dos subsistemas afetados. Assim, pode-se afirmar que esta abordagem induz uma estrutura de controle descentralizada que surge naturalmente do processo de síntese. Para examinar a existência de conflito entre os supervisórios modulares deve-se utilizar a equação (3) (QUEIROZ; CURY, 2002a).

$$\|_{i=1}^{n}\overline{L}_i = \overline{\|_{\iota=1}^{n}L_\iota} = \left\|_{i=1}^{n}\overline{SupC\left(E_{loc,\iota}, G_{loc,\iota}\right)}_i = \overline{\left\|_{\iota=1}^{n}SupC\left(E_{loc,\iota}, G_{loc,\iota}\right)\right.}_\iota \quad (3)$$

2.8.1.5 Resolução de Conflitos

Quando existe conflito entre os supervisórios modulares locais, uma alternativa evidente é a aplicação do supervisório monolítico para o menor subconjunto de especificações conflitantes (QUEIROZ; CURY, 2002b). Entretanto, outras abordagens mais elaboradas são encontradas na literatura, como evitar conflitos com introdução de interface entre supervisórios conflitantes e a planta (WONG et al., 1995), a atribuição de prioridades a supervisores individuais (CHEN; FORTUNE; LIN, 1995), coordenação hierárquica (WONG; WONHAM, 1998), ou redução de conflitos (VAZ; WONHAM, 1986).

2.8.1.6 Verificação Formal do SML

A verificação formal do SML refere-se ao problema de garantir o atendimento de uma determinada especificação pelo conjunto de autômatos e seus supervisórios modulares locais (CLARKE; GRUMBERG; PELED, 1999). Existem quatro abordagens fundamentais

no campo da verificação formal: Verificação de satisfabilidade, verificação de modelos, simulação e prova de teoremas. Entre estas, a verificação de modelos tem sido a mais utilizada na literatura acadêmica da área de verificação formal (GONZALEZ DEL FOYO, 2009).

2.8.1.6.1 Verificação de Modelos

A verificação de modelos (*model checking*) é uma técnica automática para verificação formal de sistemas concorrentes de estados finitos (CLARKE; GRUMBERG; PELED, 1999). É utilizada para verificar se o sistema supervisório e os modelos em autômatos temporizados satisfazem cada especificação, representadas por lógicas temporais (GONZALEZ DEL FOYO, 2009).

Na década de 80, dois trabalhos (QUIELLE; SIFAKIS, 1982) (CLARKE; EMERSON; SISTLA, 1986) apresentaram a técnica de verificação de modelos. Esta técnica permite eliminar a ocorrência de erros na atividade de verificação, por ser totalmente automática, por permitir o uso de várias lógicas temporais e não temporais para a representação das especificações a serem verificadas, por gerar contraexemplo[1], por permitir a verificação de partes do sistema durante o projeto de sistemas complexos, por ter algoritmos menos complexos e maior velocidade de verificação comparada a outras técnicas, como verificação de satisfabilidade, ou prova de teoremas (CLARKE; GRUMBERG; PELED, 1999) (GONZALEZ DEL FOYO, 2009). Porém também possui restrições, tais como ter única aplicação em sistemas de estados finitos e o aumento de estados durante a verificação (CLARKE; GRUMBERG; PELED, 1999) (SILVA, 2008). Estas restrições têm sido a força motriz para a continuidade das pesquisas no desenvolvimento e melhoria desta técnica.

O algoritmo desta técnica recebe as entradas, que são o modelo do sistema, as especificações e propriedades, em seguida, verifica o atendimento das especificações nos possíveis caminhos do sistema e emite o resultado de sim (atende), ou não (atende) e seu contraexemplo. A Figura 15 apresenta o esquema que representa o processo da verificação de modelo.

[1] Os contraexemplos são sequências de eventos que descrevem o não atendimento da especificação.

Figura 15. Processo de verificação de modelo.

As especificações são classificadas pelas propriedades de segurança (*safety*), que determina que algo negativo nunca vai acontecer, de evolução (*liveness*), que determina dentro de determinadas condições algo inevitavelmente vai acontecer, de ausência *deadlock*, que determina o sistema não entra numa situação da qual não consiga sair e de ausência de *livelock*, que determina um conjunto de estados não criam um componente de estados acessíveis entre si, sem transição de saída e sem estados marcados.

2.8.1.6.2 Lógica Temporal

A lógica temporal é um formalismo usado para descrever as especificações desejadas do sistema utilizando operadores temporais, quantificadores de caminhos, combinados aos operadores de lógica booleana. Como exemplo, no futuro uma dada proposição será verdadeira, ou um determinado estado nunca será alcançado, ou uma situação pode ser atingida. Existem diversas lógicas temporais, sendo as mais comuns a CTL*, CTL, LTL, ACTL, ATL e TCTL (CASSANDRAS, 2008). Os operadores temporais são tipos de mnemônicas utilizados para representar na especificação que um estado será alcançado no "futuro" (*in the future*), ou "sempre" (*globally*), ou "na próxima vez" (*next time*), ou "até outro estado ser alcançado" (*until*), ou "enquanto o estado for alcançado" (*release*). Já os quantificadores de caminhos (*path quantifiers*) são tipos de mnemônicas utilizados na representação da especificação e são colocados para representar "todo caminho" (*always*), ou "existe pelo menos um caminho que uma determinada expressão formal é alcançada" (*exist*).

3 Método Proposto

Este capítulo apresenta proposta de um método para projeto modular de célula de manufatura robotizada com programação no nível de tarefas baseado em técnicas formais. Este método considera-se que existe, no mínimo, um robô industrial e um sistema de visão computacional que possui uma comunicação direta com uma plataforma computacional que gere suas ações e resultados.

O método propõe o uso de técnicas como modelo espiral da engenharia de requisitos, diagramas IDEF0, a criação de um sistema supervisório modular local capaz de centralizar a coordenação da sequência de operações da CMR e tratar falhas e exceções. Além de uma interface de integração e operação que possibilita a programação no nível de tarefas no ambiente computacional *LabVIEW*™. Este método será utilizado no estudo de caso apresentado no Capítulo 4.

Este método propõe modularizar o sistema a ser projetado, utilizando um formalismo único, baseado em autômatos de Sistemas a Eventos Discretos (SED), em todas as etapas deste projeto, para proporcionar uma sistemática e um ganho de consistência nas transições entre as etapas, diminuir a possibilidade de erro e facilitar a implantação de melhorias. Também consiste em apresentar uma solução que permita a reconfiguração da funcionalidade da CMR de forma fácil, rápida e com menor exigência de inteligência humana, por meio da programação no nível de tarefas.

3.1 Organização do Método Proposto

O método está organizado em 9 etapas, como mostrado na Figura 16. Inicia com a definição do documento de requisitos da CMR (Etapa 1) e dos módulos para cada função independente da CMR (Etapa 2). Para cada módulo criado é modelado um autômato (Etapa 3) e gerado o modelo do SML (Etapa 4). Os modelos dos autômatos e SML são verificados utilizando o formalismo matemático e simulação (Etapa 5).

Os modelos dos autômatos e SML são utilizados para serem implementados no ambiente *LabVIEW*™ e é realizada a simulação de cenários de falhas (Etapa 6). A partir daí,

ocorre à integração dos módulos, com seus respectivos *hardware* e *software*, com os modelos dos autômatos e SML implementados no ambiente *LabVIEW*™ (Etapa 7) e a interface homem-máquina é criada utilizando o conceito de abstração da programação no nível de tarefas (Etapa 8). A última etapa (Etapa 9) é realizado o teste de operação da CMR.

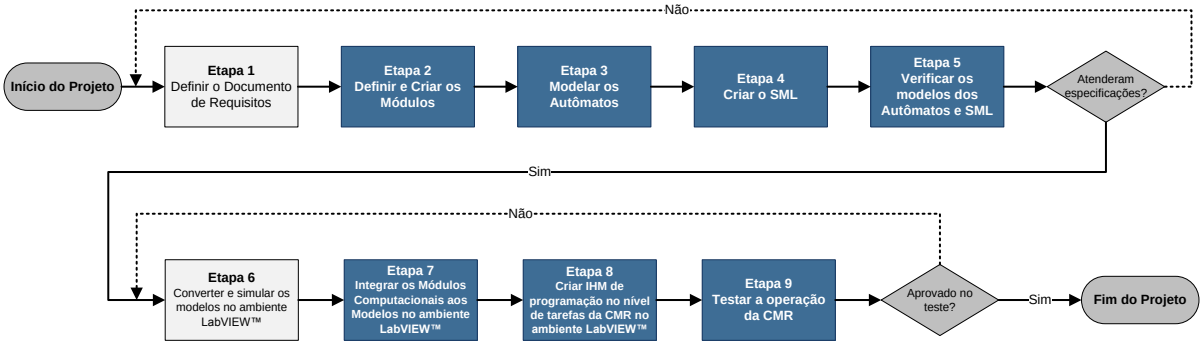


Figura 16. Fluxograma do método proposto.

A Tabela 3 apresenta um resumo do método, com o detalhamento de entrada, saída, atividade e técnica.

Tabela 3. Resumo do método proposto.

ETAPA	ENTRADA	SAÍDA	ATIVIDADE	TÉCNICA FORMAL
ETAPA 1 Definir o documento de requisitos	Necessidades do usuário	Documento de requisitos	Identificação da necessidade do usuário	Método Espiral de Engenharia de Requisitos
ETAPA 2 Definir e criar os módulos	Documento de requisitos	Módulos Assíncronos Conceituais	Definição dos módulos	Diagrama IDEF0
ETAPA 3 Modelar os autômatos	Módulos assíncronos; Documentação de requisitos	Modelos dos autômatos e das especificações	Geração dos modelos dos autômatos e das especificações	TCS
ETAPA 4 Criar o SML	Modelos dos autômatos e das especificações	Modelos do SML	Geração dos modelos do SML	TCS e SML
ETAPA 5 Verificar os modelos de autômatos e SML	Modelos autômatos e SML; Documentos de requisitos	Resultado da verificação de modelos	Verificação e melhoria dos modelos dos autômatos e SML	Verificação de Modelos
ETAPA 6 Converter e simular os modelos no ambiente *LabVIEW*™	Modelos autômatos e SML	Algoritmo no ambiente *LabVIEW*™ dos Modelos	Geração e simulação dos Modelos em Ambiente *LabVIEW*™	*LabVIEW*™

ETAPA 7 Integrar os módulos computacionais aos modelos no ambiente *LabVIEW*™	Algoritmo no ambiente *LabVIEW*™ dos Modelos	Integração no algoritmo no ambiente *LabVIEW*™ dos modelos e módulos	Integração dos módulos dos diversos dispositivos ao algoritmo no ambiente *LabVIEW*™ dos Modelos	*LabVIEW*™
ETAPA 8 Criar IHM de programação no nível de tarefas da CMR no ambiente *LabVIEW*™	Algoritmo no ambiente *LabVIEW*™ dos modelos e módulos	Ambiente de programação no nível de tarefas	Criação de IHM com abstração de programação no nível de tarefas da CMR	*LabVIEW*™
ETAPA 9 Testar a operação da CMR	Algoritmo no ambiente *LabVIEW*™ dos modelos e módulos e programação no nível de tarefas	Resultado do teste de operação da CMR	Verificação de erros e análise de falhas	*LabVIEW*™

3.1.1 Etapa 1: Definir o documento de requisitos

Nesta etapa ocorre a geração do documento de requisitos para o projeto e construção do sistema que informará a descrição do sistema desejado pelo cliente e as especificações. Será adotado o modelo espiral de geração de documentação de requisitos (KOTONYA; SOMMERVILLE, 1998) apresentado no capítulo 2. A seguir é apresentado o resumo das tarefas desta etapa.

1. Realizar a elicitação: Levantamento de informações para entendimento: (a) da aplicação do sistema; (b) dos processos independentes envolvidos na execução da aplicação do sistema e (c) das necessidades e das restrições do cliente. No final, geração de lista de possíveis requisitos;
2. Realizar análise e negociação: Os requisitos levantados na tarefa 1 são analisados e detalhados e daí são definidos os que serão utilizados;
3. Realizar a especificação: É gerado o documento de requisitos, que deve ter as seguintes seções:
    3.1. Seção 1 (Introdução): Propósito do documento, as definições e prioridades dos requisitos;

3.2. Seção 2 (Descrição geral do sistema): Apresenta uma visão geral do sistema, caracterizando qual é o seu escopo e descrevendo seus usuários.

3.3. Seção 3 (Requisitos funcionais): Especifica todos os requisitos funcionais do sistema, apresentando as descrições, entradas, saídas e recursos necessários;

3.4. Seção 4 (Requisitos não funcionais): Especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, desempenho, segurança, descrição de *hardware* e *software*;

3.5. Seção 5 (Descrição da interface com o usuário): Apresenta desenhos, figuras ou rascunhos de telas do sistema.

4. Realizar a validação: Validação (e revisão) do documento de requisitos para garantia de consistência e integridade;
5. Realizar tomada de decisão: (a) finalizar, caso não seja identificado nenhum problema, ou (b) retornar a tarefa 1 e realizar nova execução do modelo espiral, caso seja constatada alguma anormalidade.

3.1.2 Etapa 2: Definir e criar os módulos

A identificação dos módulos conceituais é realizada nesta etapa, no qual os módulos são modelados graficamente utilizando a técnica de diagrama IDEF0 (FIPS PUBS 183, 1993), a partir do documento de requisitos na etapa 1. A saída desta etapa, será o diagrama IDEF0, tabela com cada módulo levantado, e suas respectivas variáveis de entradas, saídas, recursos e controle. Para tal, devem-se realizar as tarefas descritas a seguir.

1. Identificar os sistemas independentes (em relação à funcionalidade) e necessários, para atendimento dos requisitos da CMR;
2. Renomear cada conjunto físico-computacional como módulo computacional;
3. Representar graficamente cada módulo como uma caixa e colocar a descrição de acordo com sua função, utilizando um verbo ou frase verbal e um código de identificação. Este método sugere o uso de $M_i, i = 1, \dots, n$.

4. Identificar para cada módulo suas necessidades de sinais e dados de entrada e seus resultados de saída, os sinais de controle conforme diagrama IDEF0;
5. Identificar o tipo de variável para cada entrada, saída e controle, classificando-as em variável discreta, ou contínua. Como também, o tipo de manipulação (ou estimulo) da variável, sendo interno, externo, ou informado pelo usuário da CMR;
6. Interligar na representação gráfica do diagrama IDEF0 as entradas e saídas dos módulos, quanto às necessidades, sendo que as entradas entram pelo lado esquerdo da caixa e as saídas geradas saem pelo lado direito;
7. Identificar para cada módulo os recursos físicos e computacionais necessários para a execução da funcionalidade do módulo;
8. Interligar na representação gráfica do diagrama IDEF0 para cada módulo os controles e recursos identificados, sendo que os sinais de controle entram pelo lado superior e o recurso pelo lado inferior da caixa.

3.1.3 Etapa 3: Modelar os autômatos

O objetivo desta etapa é criar os modelos de autômatos para cada módulo conceitual levantado na etapa anterior e para as especificações apresentadas no documento de requisitos da Etapa 1. Para tal, devem-se realizar as tarefas descritas a seguir.

1. Construir o modelo básico do autômato assíncrono ($G_i, i = 1, \dots, n$) para cada módulo ($M_i, i = 1, \dots, n$) descrito na etapa 2;
2. Identificar os eventos controlados e não controlados do processo. Os módulos criados na etapa 2 possuem variáveis discretas e contínuas, o modelo do autômato será apenas responsável pela representação do comportamento das variáveis discretas, que serão representados como eventos utilizando a seguinte regra: (a) variável discreta de entrada do módulo gerada pelo usuário ou variável discreta de saída do módulo estimulada pelo sistema computacional serão eventos não controlados e (b) variável discreta de entrada do módulo estimulada pelo sistema computacional serão eventos controlados;

3. Criar um modelo de autômato para os eventos não controlados;
4. Identificar no documento de requisitos as restrições do processo, ou seja, as restrições de coordenação a serem impostas ao sistema;
5. Construir o modelo básico do autômato para cada restrição ($R_{gen,i}, i = 1, \dots, n$).

3.1.4 Etapa 4: Criar o SML

Nesta etapa será criado o sistema supervisório. Para a síntese dos modelos dos autômatos e especificações, foi utilizada a abordagem de sistema supervisório modular local (SML) (QUEIROZ; CURY, 2002b). É recomendado o uso da ferramenta computacional *Grail* (RAYMOND; WOOD, 1995). O apêndice C apresenta uma introdução para o uso desta ferramenta. Devem-se realizar as tarefas descritas a seguir.

1. Identificar o conjunto de modelos dos autômatos da etapa 3 que representam subsistemas com auxílio do documento de requisitos da etapa 1. Este conjunto representa uma fase do processo, ou conjunto de equipamentos do sistema, que concebem uma parte do comportamento global do processo.
2. Gerar a planta local $G_{loc,i}, i = 1, \dots, n$ pela composição síncrona do conjunto dos modelos dos autômatos que representam cada subsistema;
3. Construir as restrições locais $R_{loc,i}, i = 1, \dots, n$ pela composição síncrona das restrições genéricas para cada subsistema da tarefa 2 desta etapa;
4. Obter as especificações locais $E_{loc,i} = G_{loc,i} || R_{loc,i}, i = 1, \dots, n$ pela composição síncrona das restrições locais ($R_{loc,i}, i = 1, \dots, n$) e sua respectiva planta local ($G_{loc,i}, i = 1, \dots, n$), ou seja, que possuem eventos comuns e considerando o comportamento da planta, tal que $L_m(E_{loc,i}) \subset L_m(G_{loc,i}), , i = 1, \dots, n$;
5. Eliminar, caso houver, os estados proibidos de $E_{loc\_sp,i}, i = 1, \dots, n$ para cada especificação local ($E_{loc,i}, i = 1, \dots, n$) da tarefa 4;
6. Calcular a componente trim $E_{loc\_trim,i}, i = 1, \dots, n$ para cada especificação local sem estados proibidos ($E_{loc\_sp,i}, i = 1, \dots, n$);

7. Calcular a máxima linguagem controlável $S_{loc,i} = supC(G_{loc,i}, E_{loc\_trim,i}), i = 1, \dots, n.$
8. Obter o supervisório reduzido $S_{red,i}, i = 1, \dots, n$ para cada $S_{loc,i}, i = 1, \dots, n.$ Desta forma será obtido o SML.

3.1.5 Etapa 5: Verificar os modelos de autômatos e SML

A verificação de modelos dos autômatos e SML são realizadas por meio das ferramentas computacionais *Grail* (RAYMOND; WOOD, 1995) e *Uppaal* (BEHRMANN; DAVID; LARSEN, 2004). O *Uppaal* é uma ferramenta de *Model Checking* para sistemas de tempo real. O apêndice D apresenta uma introdução para o uso do *Uppaal*. Devem-se realizar as tarefas descritas a seguir.

1. Verificar cada um dos modelos dos autômatos ($G_i, i = 1, \dots, n$) e SML ($S_{red,i}, i = 1, \dots, n$) é não bloqueante, ou seja, sem a existência de *deadlock* e/ou *livelock*, (utilizar a ferramenta computacional *Grail*). Caso seja bloqueante, retornar para a Etapa 2 deste método;
2. Verificar o conjunto dos supervisórios reduzidos locais do SML ($S_{red,i}, i = 1, \dots, n$) são não conflitantes (ou seja, não bloqueantes entre si) (utilizar a ferramenta computacional *Grail*);
3. Desenhar os modelos dos autômatos ($G_i, i = 1, \dots, n$) e dos supervisórios reduzidos locais do SML ($S_{red,i}, i = 1, \dots, n$) (Utilizar a ferramenta computacional *Uppaal*);
4. Eliminar os eventos que geram não coordenação[2] dos supervisórios reduzidos locais do SML ($S_{red,i}, i = 1, \dots, n$);
5. Verificar o conjunto de modelos dos autômatos e SML a existência de *deadlock* (utilizar a verificação de modelos do *Uppaal*). Caso exista, retornar para a Etapa 2 deste método;

[2] Eventos que geram não coordenação: são os eventos repetidos que estão em loop no estado inicial de cada supervisório reduzido local do SML.

6. Escrever as especificações do documento de requisitos (etapa 1) em linguagem CTL (utilizar a verificação de modelos do *Uppaal*);
7. Verificar o atendimento de cada especificação pelos modelos de autômatos e SML (utilizar a verificação de modelos do *Uppaal*). Caso não atenda, retornar para a Etapa 2 deste método.

3.1.6 Etapa 6: Converter e simular os modelos no ambiente *LabVIEW™*

Os modelos dos autômatos $(G_i, i = 1, \dots, n)$ e SML $(S_{red,i}, i = 1, \dots, n)$ são convertidos para o ambiente computacional *LabVIEW™* utilizando a ferramenta *State Diagram*. O apêndice E apresenta uma introdução ao uso da ferramenta *State Diagram* do *LabVIEW™*. Para tal, devem-se realizar as tarefas descritas a seguir:

1. Criar um *State Diagram* para cada autômato e SML, desenhando seu respectivo modelo no *State Diagram editor*;

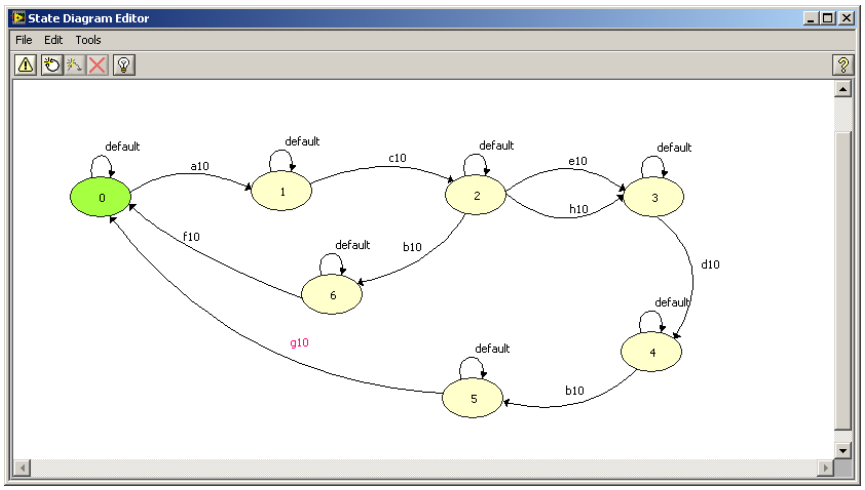


Figura 17. *State Diagram Editor* do *LabVIEW™*.

2. Criar *leds* indicadores e ligar (utilizar a variável local) aos eventos que indicam sinais internos de todos os *State Diagram*, conforme Figura 17 para cada modelo de autômato e SML;
3. Criar o algoritmo de comparação, como indicado pela seta na Figura 18 para todos os eventos dos *State Diagram* de cada modelo de autômato e SML;

4. Criar uma variável local para cada evento interno não controlado do modelo de autômato e SML conforme Figura 18;
5. Fazer temporizador dentro do *State Diagram* conforme Figura 18 para cada modelo de autômato e SML;

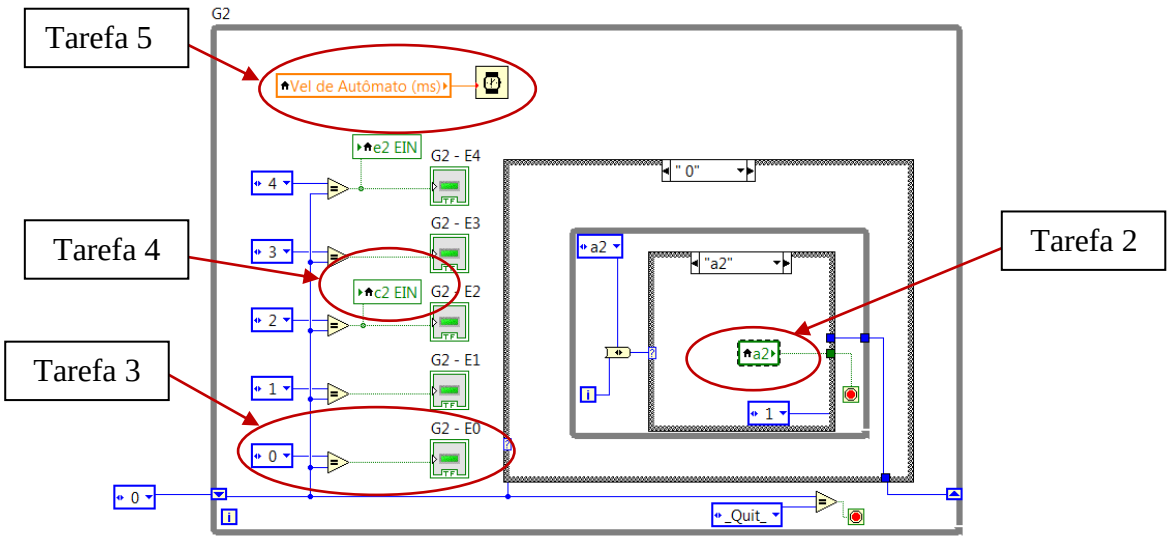


Figura 18. Criação do modelo do autômato.

6. Realizar simulação para verificação de conflitos e erros de programação;
7. Levantar os cenários que representam as variações de parâmetros de entradas;
8. Executar na simulação os diversos cenários;
9. Simular e verificar o atendimento as especificações do documento de requisitos (Etapa 1).
10. Levantar e detalhar múltiplos cenários que representam as variações de parâmetros de entradas
11. Executar os múltiplos cenários;
12. Verificar as soluções de falhas apresentadas no documento de requisitos (etapa 1) e validar resultados.

3.1.7 Etapa 7: Integrar os módulos computacionais aos modelos no ambiente LabVIEW™

Nesta etapa ocorrerá a integração entre os diversos módulos da CMR ao programa de controle supervisório no ambiente *LabVIEW™*. O módulo é o programa computacional que executará o módulo conceitual definido na etapa 2. Para isso, devem-se realizar as tarefas descritas a seguir:

1. Elaborar o programa computacional para cada módulo conceitual no ambiente *LabVIEW™*, atendendo as necessidades das entradas, saídas, controle e recursos descritos na etapa 2.
2. Criar cada módulo conforme apresentado na Figura 19 e Figura 20 no ambiente *LabVIEW™*, todas as instruções apresentadas nas figuras devem ser implementadas para o correto funcionamento na integração com *State Diagram*.
3. Implementar o algoritmo no local indicado na Figura 19 como (Etapa 7.3). Neste local deve ser escrito o programa computacional

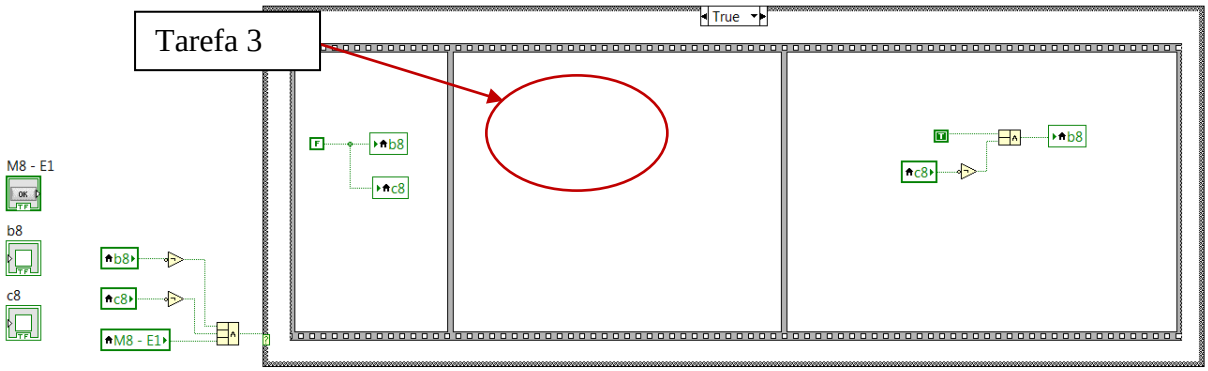


Figura 19. Módulo conceitual estado *true* implementado no *LabVIEW™*.

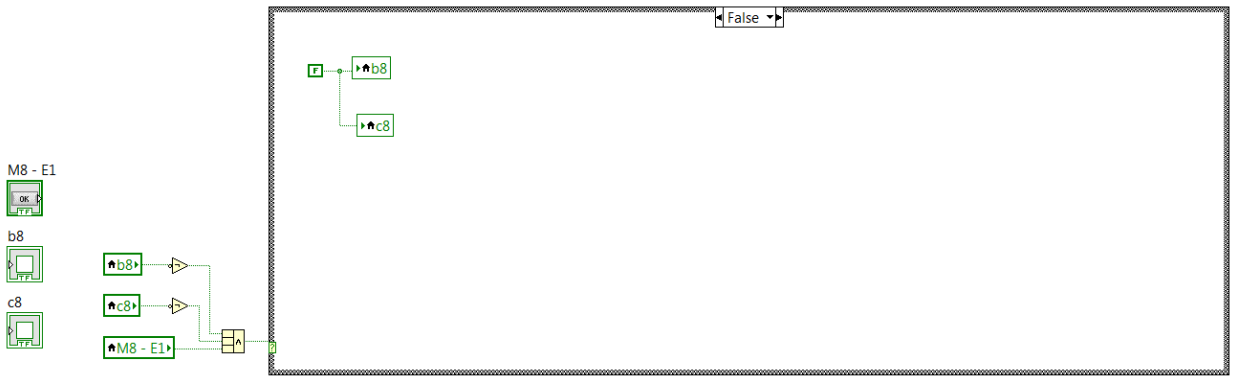


Figura 20. Módulo conceitual estado *false* implementado no *LabVIEW™*.

4. Testar a execução de cada programa computacional desenvolvido;
5. Ligar todas as entradas, saídas, controles e recursos (caso seja *hardware*) de cada programa computacional criado às suas respectivas variáveis relacionadas aos modelos dos autômatos criados na etapa 6 conforme Figura 21;

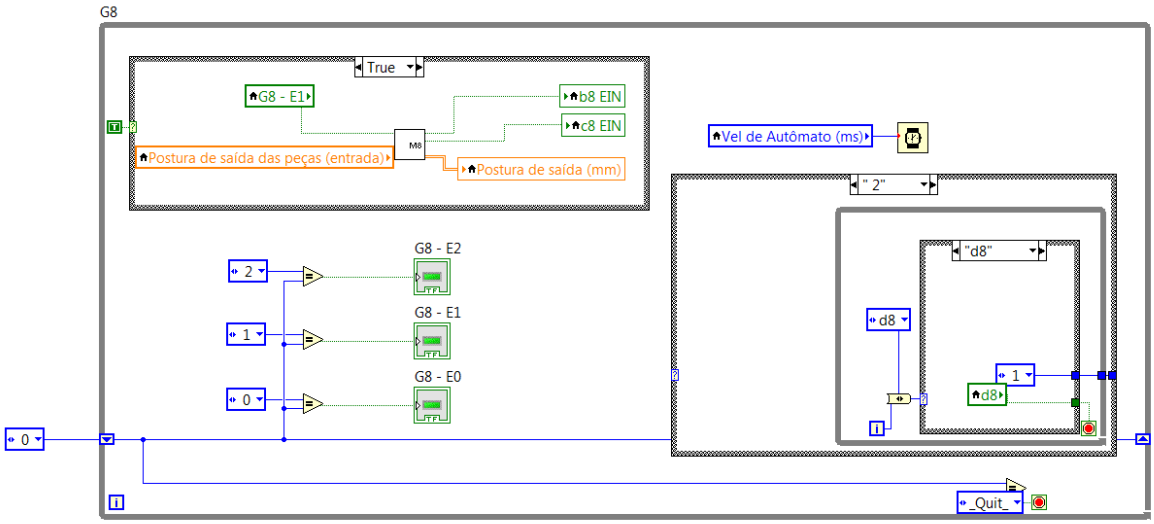


Figura 21. Módulo computacional integrado no ambiente *LabVIEW™*.

6. Integrar e testar as conexões entre o *hardware* e o *software* implementado. Corrigir em caso de erro.

3.1.8 Etapa 8: Criar IHM de programação no nível de tarefas da CMR no ambiente LabVIEW™

Nesta etapa será criada a interface homem máquina (IHM) com abstração para programação no nível de tarefas da CMR. Devem-se adotar as seguintes tarefas:

1. Realizar levantamento de termos para representação das tarefas para serem usadas na programação no nível de tarefas;
2. Criar o ambiente para programação no nível de tarefas no ambiente *LabVIEW*™;
3. Testar a programação no nível de tarefas.

3.1.9 Etapa 9: Testar a operação da CMR

Nesta etapa é realizado o teste de operação da CMR para verificação de defeitos e erros. Devem-se adotar as seguintes tarefas.

1. Identificar todas as ligações físicas entre o computador e os equipamentos da CMR;
2. Identificar o correto funcionamento da comunicação entre os diversos *hardwares* e os *softwares* da CMR;
3. Realizar as condições iniciais da CMR;
4. Realizar a programação no nível de tarefas para execução da CMR;
5. Executar a programação e observar o andamento das tarefas na CMR;
6. Levantar condições de falhas descritos no documento de requisitos elaborados na Etapa 1;
7. Testar a CMR com as condições de falhas;

Obtendo sucesso em todas as tarefas propostas nesta etapa, a CMR estará simulada e aprovada.

3.2 Conclusões do Método Proposto

O método apresentado busca sistematizar o projeto de CMR utilizando 9 etapas com o uso de diversas técnicas. Em algumas etapas (1 até 5) e (6 e 9) é realizado uma averiguação do atendimento as especificações do sistema, caso não atenda, o método propõe um retorno para uma etapa pontual, isto reduz o tempo nas correções de erros no final do projeto.

4 Aplicação do Método

Este capítulo apresenta a aplicação do método proposto no Capítulo 3 para projeto modular de célula de manufatura robotizada com programação no nível de tarefas baseado em técnicas formais. O processo da CMR será de seleção e movimentação de peças. Serão mostrados os recursos físicos, a descrição do processo da CMR, a aplicação de cada etapa do método e os resultados obtidos.

4.1.1 Etapa 1: Definir o documento de requisitos

Esta seção apresenta o documento de requisitos, gerado pela técnica de modelo espiral de engenharia de requisitos (KOTONYA; SOMMERVILLE, 1998) apresentado no capítulo 02. O documento de requisitos possui as seguintes seções:

Seção 1. Introdução

Seção 1.1. Propósito do Documento

Este documento contém a especificação de requisitos para a célula de manufatura robotizada de seleção e movimentação de quatro diferentes peças que permita a programação no nível de tarefas.

Seção 1.2. Abreviações e Prioridades dos Requisitos

Os requisitos são apresentados como: RF para requisitos funcionais e NF para requisitos não funcionais, seguindo de uma numeração sequencial. Já as prioridades dos requisitos serão denominadas de essencial, de importante e de desejável. Em que, o essencial é o requisito sem o qual o sistema não entra em funcionamento; importante é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória; e desejável é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele.

Seção 2. Descrição Geral do Sistema

A Célula de Manufatura Robotizada realiza o processo de seleção e movimentação de quatro diferentes peças e permite a programação no nível de tarefas. Os

recursos físicos e computacionais da CMR são integrados em uma única plataforma e suas tarefas gerenciadas por um sistema supervisório. A CMR deve realizar automaticamente a movimentação de peças solicitadas pelo usuário, possibilitar a programação no nível de tarefas da CMR, disponibilizar indicação visual de falha do sistema e projetar a solução para atender ao conceito de projeto modular. O processo está dividido em três fases distintas: Fase 1: a rotina de inicialização; Fase 2: modo de seleção e determinação de sequência de movimentação de peças e Fase 3: modo de operação de movimentação das peças. A Figura 22 ilustra as fases deste processo. Os usuários são operadores de máquinas em ambiente industrial.

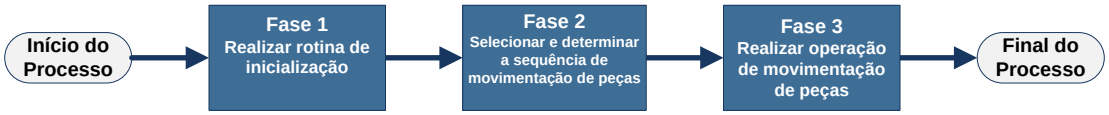


Figura 22. Fases do Processo.

A Fase 1: Modo de seleção e determinação de sequência movimentação de peças é apresentada no fluxo de tarefas da Figura 23 e detalhado logo a seguir.

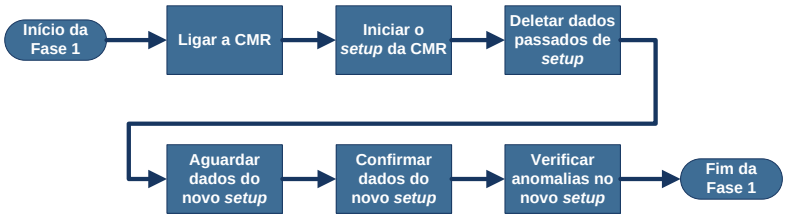


Figura 23. Fase 1 - Rotina de inicialização.

(a) Ligar a CMR;

(b) Remover do sistema supervisório arquivos antigos de dados referentes a processos executados anteriormente;

(c) Executar o *setup* da CMR, no qual o usuário informa a (b.1) postura do robô para posição de segurança, também chamada de "HOME", (b.2) postura do robô para realização da calibração, (b.3) parâmetros de calibração[3] e ajuste entre o sistema de visão computacional e as coordenadas espaciais do robô,

[3] Parâmetros de calibração: É o conjunto de informações necessário para a correta calibração da localização do robô em relação aos dados extraídos das imagens capturadas pelo sistema de visão.

(b.4) parâmetros de reconhecimento das peças, (b.5) parâmetros de sequência de movimentação e (b.6) postura de saída das peças selecionadas;

(d) Carregar os novos dados de entrada e (d) verificar anomalias.

A Fase 2: Modo de seleção e determinação da sequência de movimentação é apresentada no fluxo de tarefas da Figura 24 e detalhado logo a seguir.

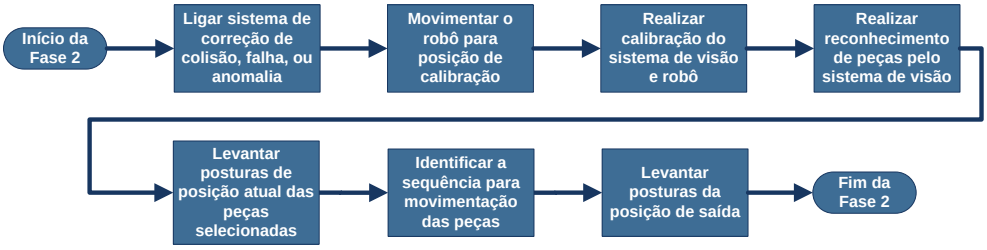


Figura 24. Fase 2 - Modo de seleção e determinação da sequência de movimentação.

(a) Ligar sistema de correção de possível colisão e falha no robô, ou anomalia no processamento dos módulos computacionais;

(b) Enviar comando para a movimentação do robô para a postura de calibração e de reconhecimento de peças para a seleção na plataforma de entradas de peças;

(c) Realizar calibração e ajuste entre o sistema de visão computacional e as coordenadas espaciais do robô;

(d) Realizar levantamento da postura atual das peças selecionadas na plataforma de entrada de peças;

(e) Identificar a sequência de movimentação das peças e (g) Identificar as posturas de saída das peças selecionadas.

A Fase 3: Modo de movimentação de peças é apresentada no fluxo de tarefas da Figura 25 e detalhado logo a seguir.

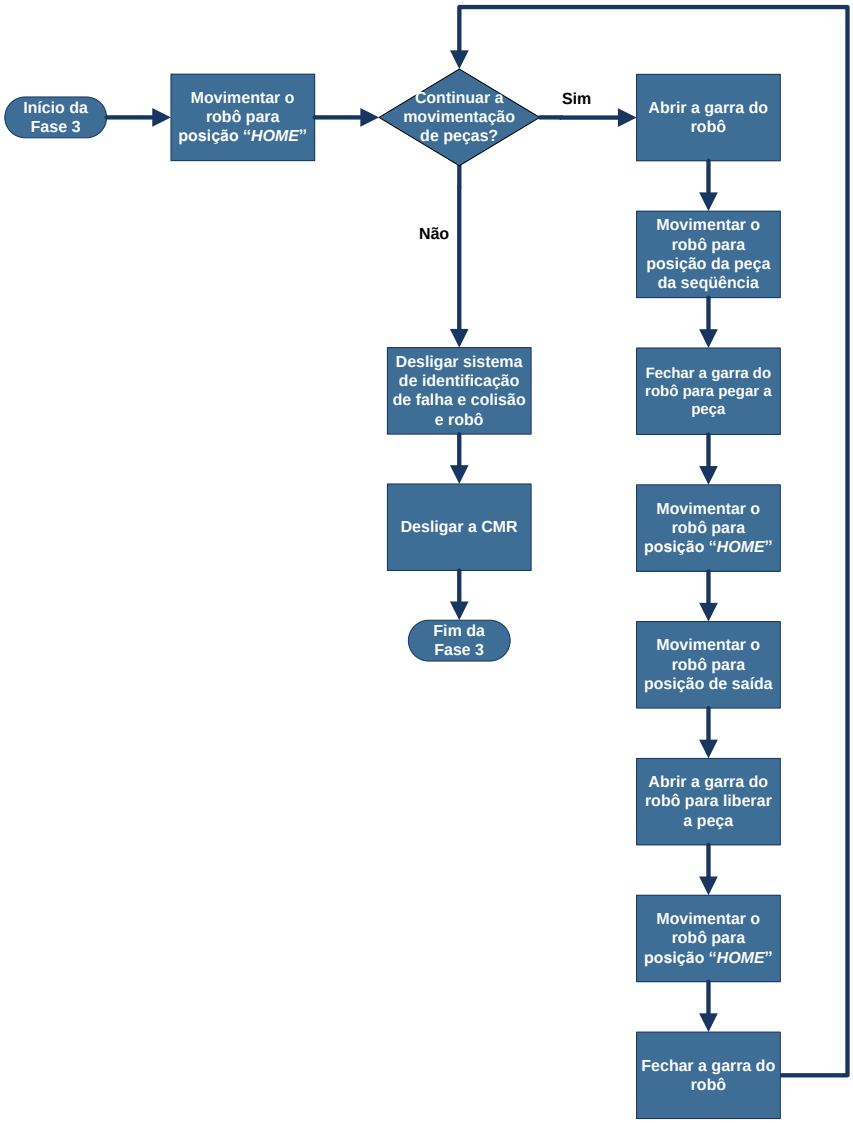


Figura 25. Fase 3 - Modo de operação da montagem.

(a) Enviar comando de movimentação do robô industrial para a postura de segurança "HOME";
(b) Abrir a garra do robô;
(c) Enviar comando de movimentação do robô industrial para a postura da primeira peça da sequência de movimentação;
(d) Fechar a garra do robô industrial para transporte da peça;
(e) Enviar comando de movimentação do robô industrial para a postura de segurança "HOME";
(f) Enviar comando de movimentação do robô industrial para a postura de saída de peças;
(g) Abrir a garra para liberar a peça;

(h) Enviar comando de movimentação do robô industrial para a postura de segurança “HOME”;

(i) Fechar garra do robô.

As tarefas entre (b) e (i) são repetidas até a movimentação completa das peças selecionadas para saída da plataforma de entrada. Após a finalização da movimentação deve-se:

(j) Desligar o sistema de identificação de falha e colisão;

(k) Desligar a CMR.

Seção 3. Requisitos Funcionais

A Tabela 4 apresenta os requisitos funcionais do sistema desejado.

Tabela 4. Requisitos funcionais.

Requisito Funcional (RF)	Descrição do Requisito Funcional	Tipo de Informação	Código do RF	Descrição da Informação
RF001	O usuário deve ser capaz de ligar a CMR e o sistema deve desligar após a execução automática das três fases, realizando gerenciamento das três fases do processo e, quando necessário, aguardando a entrada de dados do sistema. Prioridade: Essencial.	**Entradas**	RF001e1	Ligar a CMR por iniciativa do usuário.
			RF001e2	Finalizar a execução da Fase 1.
			RF001e3	Finalizar a execução da Fase 2.
			RF001e4	Finalizar a execução da Fase 3.
		Saídas	RF001s1	Iniciar a execução da Fase 1.
			RF001s2	Iniciar a execução da Fase 2.
			RF001s3	Continuar a execução da Fase 2.
			RF001s4	Iniciar a execução da Fase 3.
			RF001s5	Desligar a CMR.
		Recursos	RF001r1	Aplicativo de software no ambiente *LabVIEW*™
			RF001r2	Usuário
RF002	O usuário deve ter condições de realizar a entrada de dados iniciais para a realização automática das fases da CMR. Prioridade: Essencial.	**Entradas**	RF002e1	Iniciar o *setup*.
			RF002e2	Apagar os dados anteriores do *setup*.
			RF002e3	Finalizar a entrada de dados do *setup* por iniciativa do usuário.
			RF002e4	Parâmetros da CMR.
		Saídas	RF002s1	Informar dos antigos dados estão apagados e iniciar a entrada de novos dados do *setup*.
			RF002s2	Informar a finalização da entrada de dados do *setup* por iniciativa do usuário.
			RF002s3	Postura do robô para posição de segurança “HOME”.

			RF002s4	Postura do robô para posição de calibração.
			RF002s5	Parâmetros de calibração e ajuste entre o sistema de visão computacional e as coordenadas espaciais do robô.
			RF002s6	Parâmetros para reconhecimento de peças.
			RF002s7	Parâmetros de sequência de movimentação.
			RF002s8	Postura de saída das peças selecionadas.
		Recursos	RF002r1	Aplicativo de software no ambiente *LabVIEW*™
			RF002r2	Usuário
RF003	O sistema deve realizar os movimentos do robô de forma automática. Prioridade: Essencial.	**Entradas**	RF003e1	Habilitar o módulo.
			RF003e2	Desabilitar o módulo na Fase 2.
			RF003e3	Desabilitar o módulo na Fase 3.
			RF003e4	Movimentar o robô para postura de calibração.
			RF003e5	Movimentar o robô para postura de segurança “HOME”.
			RF003e6	Movimentar o robô para a postura da peça.
			RF003e7	Informar a correção da possível colisão.
			RF003e8	Parar a movimentação do robô por iniciativa do usuário.
			RF003e9	Retomar a movimentação do robô por iniciativa do usuário.
			RF003e10	Informar a correção da falha da execução do movimento do robô.
			RF003e11	Postura do robô para posição de segurança “HOME”.
			RF003e12	Postura do robô para posição de calibração.
			RF003e13	Postura do robô para postura de movimentação de peças da entrada.
			RF003e14	Postura do robô para postura de movimentação de peças da saída.
		Saídas	RF003s1	Indicar a chegada à postura de calibração.
			RF003s2	Indicar a chegada à postura de segurança “HOME”.
			RF003s3	Indicar a chegada à postura de peça para movimentação.
			RF003s4	Indicar possível colisão.
			RF003s5	Indicar falha da execução do movimento do robô.
		Recursos	RF003r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF003r2	Robô Industrial – ROBOT 5150A.
RF004	O sistema deve ter	**Entradas**	RF004e1	Executar o módulo.

	capacidade de realizar calibração e ajuste utilizando o sistema de visão computacional para a correta postura do robô. Prioridade: Essencial.		RF004e2	Reiniciar a execução do módulo (após correção de anomalia).
			RF004e3	Parâmetros de setup de calibração.
		Saídas	RF004s1	Indicar o final da execução do módulo.
			RF004s2	Indicar a existência de anomalia.
			RF004s3	Parâmetros de calibração do robô e sistema de visão.
		Recursos	RF004r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF004r2	Sistema de visão computacional.
			RF004r3	Robô Industrial.
RF005	O sistema deve ser capaz de realizar o reconhecimento dos quatro diferentes tipos de peças utilizando o sistema de visão computacional. Prioridade: Essencial.	**Entradas**	RF005e1	Executar o módulo.
			RF005e2	Reiniciar a execução do módulo (após correção de anomalia).
			RF005e3	Parâmetros para reconhecimento de peças.
		Saídas	RF005s1	Indicar o final da execução do módulo.
			RF005s2	Indicar a existência de anomalia.
			RF005s3	Imagem das peças de entradas selecionadas.
		Recursos	RF005r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF005r2	Sistema de visão.
RF006	O sistema deve ser capaz de determinar a postura de cada peça identificada pelo reconhecimento de peça e selecionada para a movimentação. Prioridade: Essencial.	**Entradas**	RF006e1	Executar o módulo.
			RF006e2	Reiniciar a execução do módulo (após correção de anomalia).
			RF006e3	Parâmetros de calibração do robô e sistema de visão.
			RF006e4	Imagem das peças de entradas selecionadas.
		Saídas	RF006s1	Indicar o final da execução do módulo.
			RF006s2	Indicar a existência de anomalia.
			RF006s3	Conjunto de postura das peças de entradas selecionadas.
		Recursos	RF006r1	Aplicativo de software no ambiente *LabVIEW*™.
RF007	O sistema deve definir a sequencia de movimentação de peças selecionadas da posição de entrada para a posição de saída. Prioridade: Essencial.	**Entradas**	RF007e1	Executar o módulo.
			RF007e2	Reiniciar a execução do módulo (após correção de anomalia).
			RF007e3	Parâmetros de sequência de movimentação.
		Saídas	RF007s1	Indicar o final da execução do módulo.
			RF007s2	Indicar a existência de anomalia.
			RF007s3	Parâmetros da sequência de movimentação.
		Recursos	RF007r1	Aplicativo de software no ambiente *LabVIEW*™.

RF008	O sistema deve ser capaz de definir a postura de saída na plataforma de saída para cada peça selecionada. Prioridade: Essencial.	**Entradas**	RF008e1	Executar o módulo.
			RF008e2	Reiniciar a execução do módulo (após correção de anomalia).
			RF008e3	Parâmetros de saída das peças selecionadas.
		Saídas	RF008s1	Indicar o final da execução do módulo.
			RF008s2	Indicar a existência de anomalia.
			RF008s3	Conjunto de postura da saída das peças.
		Recursos	RF008r1	Aplicativo de software no ambiente *LabVIEW*™.
RF009	O sistema deve proporcionar a correção de possível colisão, falha do robô, ou anomalia na fase 2 do processo da CMR. Prioridade: Essencial.	**Entradas**	RF009e1	Informar a existência de colisão, ou falha, ou anomalia.
			RF009e2	Indicar a correção por iniciativa do usuário.
		Saídas	RF009s1	Indicar a correção.
		Recursos	RF009r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF009r2	Usuário.
RF010	O sistema deve proporcionar a correção de possível colisão, falha do robô, ou anomalia na fase 3 do processo da CMR. Prioridade: Essencial.	**Entradas**	RF010e1	Informar a existência de colisão, ou falha, ou anomalia.
			RF010e2	Indicar a correção por iniciativa do usuário.
		Saídas	RF010s1	Indicar a correção.
		Recursos	RF010r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF010r1	Usuário.
RF011	O sistema deve realizar a abertura e fechamento da garra do robô de forma automática. Prioridade: Essencial.	**Entradas**	RF011e1	Sinal para habilitar o módulo da ferramenta do robô.
			RF011e2	Sinal para desabilitar o módulo da ferramenta do robô.
		Saídas	RF011s1	Sinal de ferramenta do robô ligada.
			RF011s2	Sinal de ferramenta do robô desligada.
		Recursos	RF011r1	Aplicativo de software no ambiente *LabVIEW*™.
			RF011r2	Garra do robô.
RF012	O sistema deve ser capaz de gerir a movimentação das peças selecionadas para a movimentação. Prioridade: Essencial.	**Entradas**	RF012e1	Habilitar a gestão da movimentação de peças.
			RF012e2	Informar o final da movimentação de peça.
			RF012e3	Parâmetros da sequência de movimentação.
		Saídas	RF012s1	Iniciar/continuar movimentação de peças.
			RF012s2	Encerrar a movimentação de peças.
		Recursos	RF012r1	Aplicativo de software no ambiente *LabVIEW*™.

Seção 4. Requisitos Não Funcionais

São apresentados os requisitos não funcionais.

NF001. Usabilidade

Descrição: O sistema deve ter uma interface amigável, simular o sistema antes da operação e supervisão das operações em tempo real. Deve permitir fácil programação e alto nível de abstração durante a reconfiguração.

NF002. Desempenho

Descrição: O sistema deve realizar as ações em tempo real.

NF003. Segurança

Descrição: O sistema deve garantir a segurança e integridade dos equipamentos, verificar a existência de colisão e garantir a isolação da área para segurança do operador, caso a área seja violada, desligar a CMR.

NF004. Tratamento de Falhas

Descrição: O sistema deve informar ao usuário da CMR e aguardar confirmação de solução para a: (a) Existência de anomalia ou possível colisão na execução do movimento do robô; (b) Informar ao usuário da CMR e aguardar confirmação de solução para a: (b.1) Existência de anomalia na execução da calibração e ajuste do robô e sistema de visão; (b.2) Existência de anomalia na execução do reconhecimento de peças pelo sistema de visão; (b.3) Existência de anomalia na execução da determinação das posturas das peças selecionadas na entrada para serem movimentadas; (b.4) Existência de anomalia na execução da determinação do sequenciamento da movimentação das peças; e (b.5) Existência de anomalia na execução da determinação das posturas de saída das peças movimentadas; (c) Interromper o movimento do robô devido a existência de uma possível colisão, ou anomalia, ou por iniciativa do usuário.

NF004. *Software*

Descrição: O sistema deve ser programado em ambiente *LabVIEW*™.

NF005. *Hardware*

Descrição: A CMR utilizada neste estudo de caso possui os recursos físicos descritos na Tabela 5. A Figura 26 apresenta o desenho esquemático desta CMR.

Tabela 5. Recursos Físicos da CMR do estudo de caso.

Recurso Físico	Detalhamento do Recurso Físico	Aplicação na CMR
1	Robô Industrial	Realizar a manipulação das peças
2	Sistema de Visão Computacional	Realizar a identificação das peças
3	Ferramenta do Robô (garra)	Realizar a fixação das peças para manipulação pelo robô industrial
4	Sistema de Segurança (simulado)	Realizar a verificação de possível colisão
5	Plataformas de Entrada e Saída de Peças	Entrar e sair com as peças
6	Peças	Objetos para movimentação

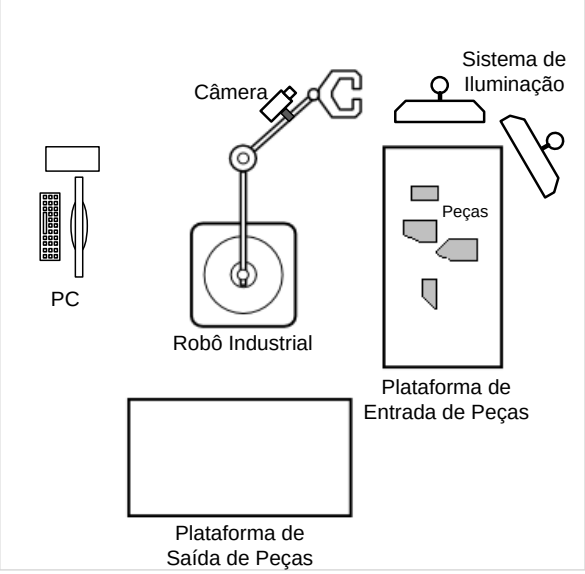


Figura 26. Desenho esquemático da CMR da aplicação do método.

A ferramenta do robô é uma garra de atuação elétrica, responsável pela fixação para movimentação da peça no robô. Possui dois estados: (1) sistema ligado – peça fixada no robô e (2) sistema desligado – peça livre da fixação no robô. O sistema de segurança é responsável pela verificação de possível colisão. Quando ligado, sensores realizam a varredura no espaço de trabalho do robô a procura de possível colisão. Este sistema foi simulado neste trabalho. As plataformas de entrada e saída de peças são mesas de dimensões de 1000 x 1200 x 750 mm. O manipulador robótico e o sistema de visão são detalhados nas próximas seções. As peças são apresentadas na Figura 27.

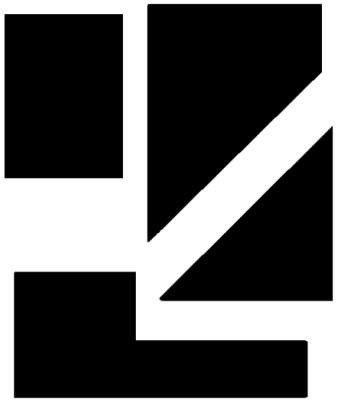


Figura 27. Peças para movimentação.

O Robô Industrial da CMR é o ROBOT 5150-A, robô educacional, que possui cinco graus de liberdade (GDL) e é do tipo antropomórfico com juntas de revoluções (RRR). Este robô foi criado e produzido pela empresa *Lab*-Volt. A Figura 28 apresenta uma imagem deste robô.

Figura 28. ROBOT 5150-A do fabricante *Lab-Volt*

São encontrados nos laboratórios de universidades e utilizados em diversas aplicações, entre as quais, movimentação de peças. As principais características são apresentadas na Tabela 6.

Tabela 6. Principais características.

Item	Descrição do Item	Característica Técnica
1	Graus de Liberdade	5 GDL
2	Motor	Motor de passo DC
3	Número de passos por revolução	200
4	Repetitividade de posição	± 3,2 mm
5	Velocidade da ferramenta	3,3 ft/s máxima
6	Massa	9,3 kg
7	Capacidade estática de carga	0,44kg
8	Alcance máximo	432 mm
9	Transmissão	Engrenagens e correias dentadas
10	Tensão de entrada	13,8 VDC
11	*End-Effector (órgão Terminal)*	Garras
12	Envelope de Trabalho	
12.1	Base	338°
12.2	Articulação do ombro	181°
12.3	Articulação do cotovelo	198°
12.4	*Pitch*	185°
12.5	*Roll*	360°

O sistema de visão computacional, utiliza uma *webcam* para aquisição de imagem e um computador pessoal com o *software* instalado da plataforma *LabVIEW™* que realiza o tratamento de imagens e o reconhecimento de objetos (neste caso, peças). A técnica adotada neste trabalho é de iluminação direcional, que provê uma iluminação em direção fixa e com pouca dispersão com uso de refletores, o qual reflete sobre o objeto realçando características especiais deste e, assim evitando regiões de sombras.

Seção 5. Descrição da Interface com o Usuário

A interface com o usuário deve permitir a programação no nível de tarefas.

4.1.2 Etapa 2: Definir e criar os módulos

Esta seção apresenta a modelagem gráfica e conceitual dos módulos utilizando a técnica IDEF0. A Tabela 7 apresenta os sistemas independentes, em relação à funcionalidade, identificados pelo documento de requisitos da Etapa 1, o qual é renomeado em módulos.

Tabela 7. Módulos Conceituais e suas fases de aplicação.

Módulo	Função do Módulo	Fase da aplicação do módulo
M1	Ligar e desligar a CMR e Gerir as três Fases do Processo	Fase 1, 2 e 3
M2	Realizar *setup* pelo usuário	Fase 1
M3	Movimentar o robô	Fase 2 e 3
M4	Calibrar e ajustar robô ao sistema de visão	Fase 2
M5	Reconhecer peças pelo sistema de visão	Fase 2
M6	Determinar as posturas das peças selecionadas na entrada para serem movimentadas	Fase 2
M7	Determinar a sequência de movimentação	Fase 2
M8	Determinar as posturas da saída das peças	Fase 2
M9	Corrigir possível colisão, falha do robô, ou anomalia na Fase 2	Fase 2
M10	Corrigir possível colisão, ou falha do robô na Fase 3	Fase 3
M11	Abrir e Fechar a garra do robô	Fase 3
M12	Gerir a movimentação das peças	Fase 3

A Figura 29 apresenta a arquitetura adotada para esta CMR.

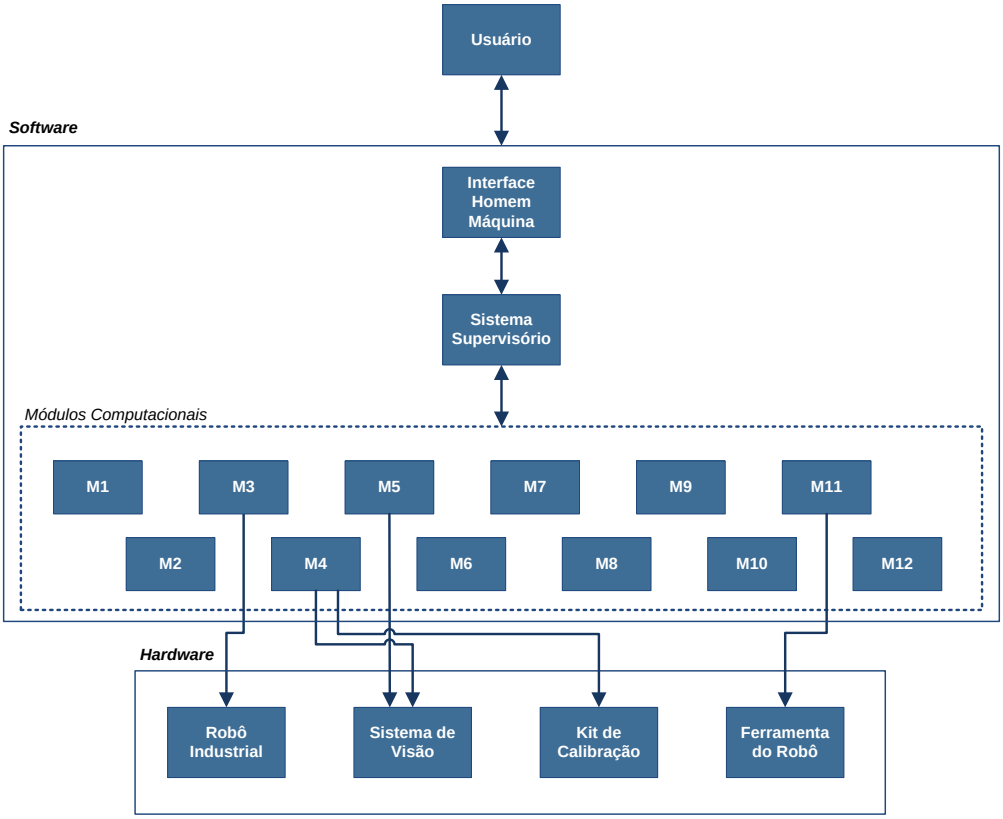


Figura 29. Arquitetura da CMR.

Foi elaborada a representação gráfica do IDEF0 para cada módulo, apresentadas entre a Figura 30 e Figura 36.

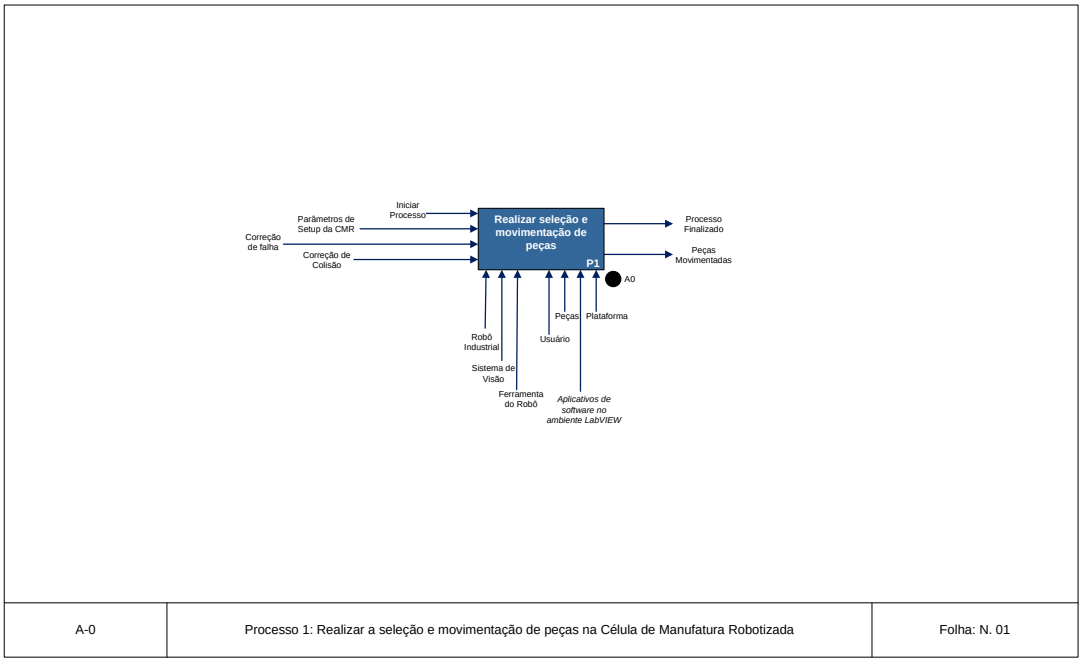
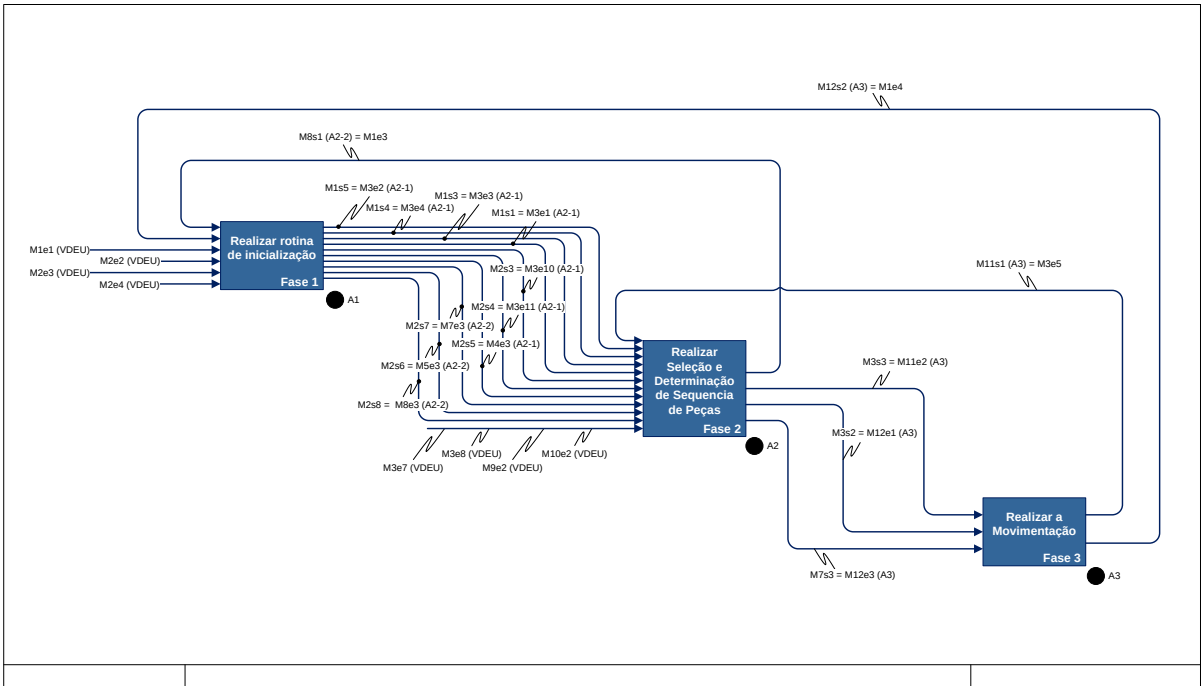
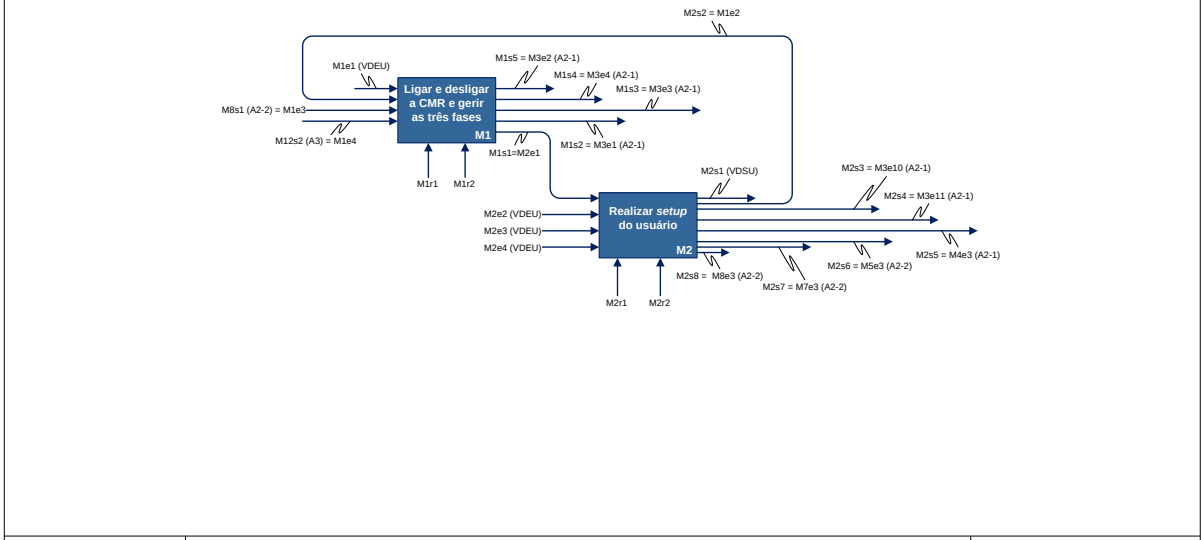


Figura 30. Diagrama IDEF0 do processo: A-0



A0	Processo 1: Realizar a seleção e movimentação de peças na Célula de Manufatura Robotizada	Folha: N. 02

Figura 31. Diagrama IDEF0 das fases do processo: A0



A1	Fase 1: Rotina de Inicialização	Folha: N. 03

Figura 32. Diagrama IDEF0 da fase 1: A1

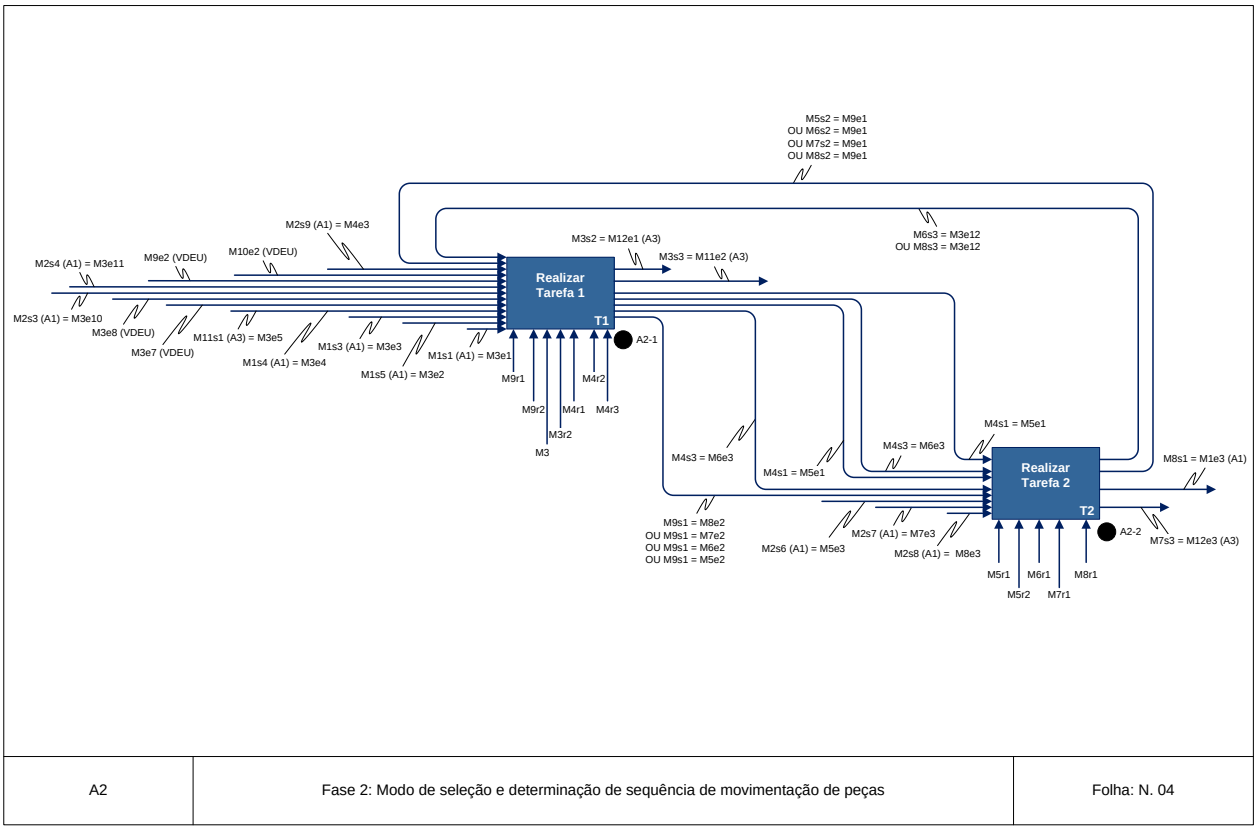


Figura 33. Diagrama IDEF0 da fase 2: A2

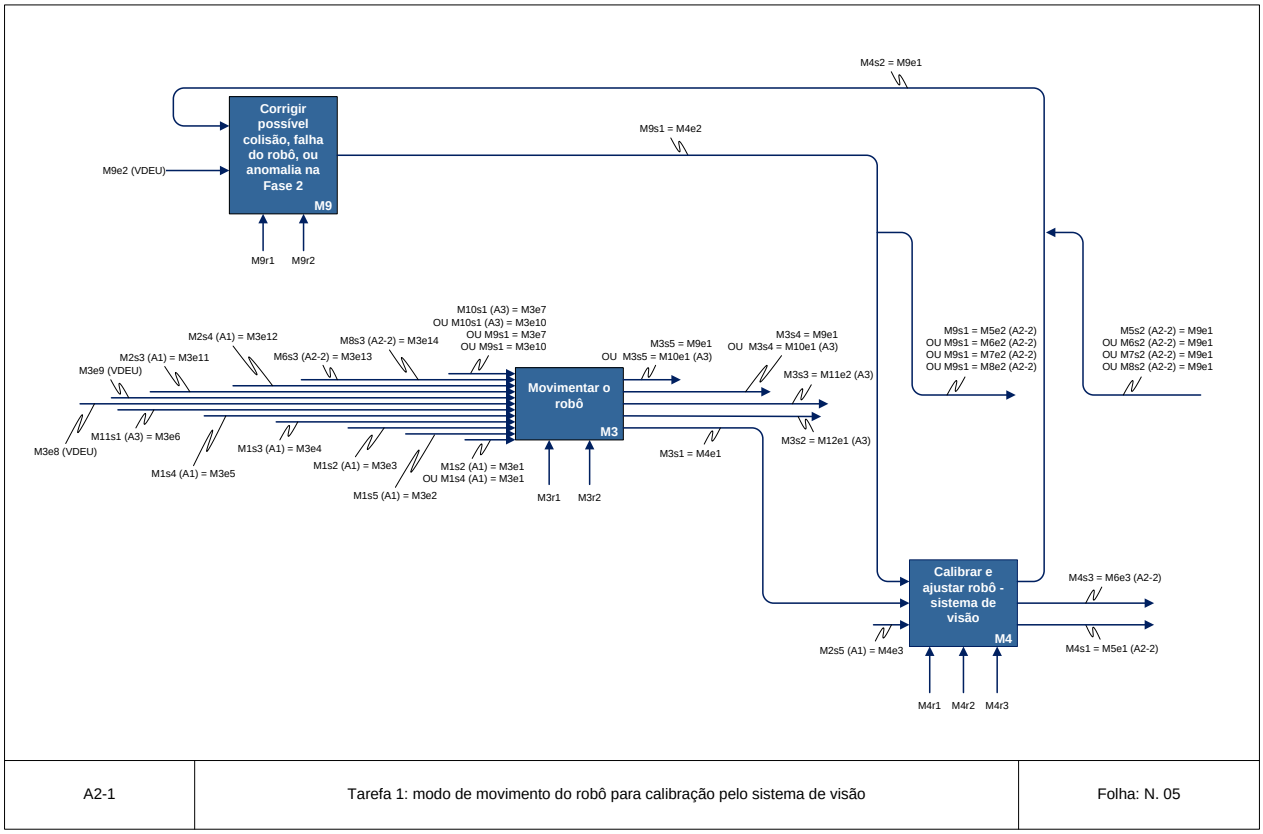
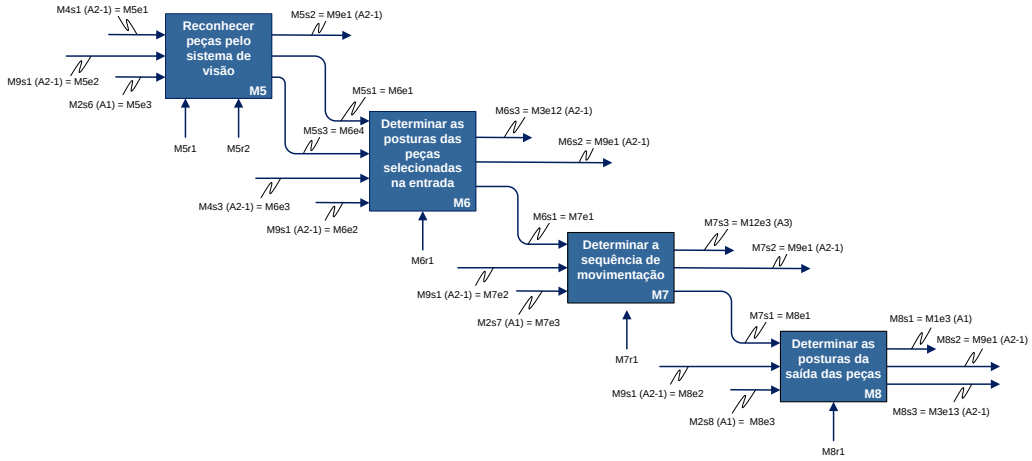
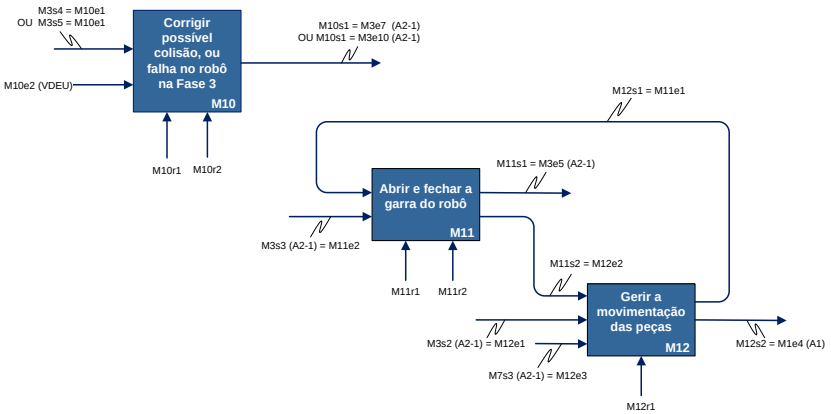


Figura 34. Diagrama IDEF0 da tarefa 1 da fase 2: A2-1.



A2-2	Tarefa 2: Modo de Determinação das posições das peças na entrada e saída e sequencia de movimentação.	Folha: N. 06

Figura 35. Diagrama IDEF0 da tarefa 2 da fase 2: A2-2.



A3	Fase 3: Modo de Movimentação de peças	Folha: N. 07

Figura 36. Diagrama IDEF0 da fase 3: A3

A Tabela 8 apresenta o detalhamento das informações de cada módulo. Lembrando que as setas não representam o fluxo ou sequência como uma modelagem convencional de fluxo de processo.

Tabela 8. Variáveis da modelagem conceitual do IDEF0

Módulo (M_i)	Função do Módulo	Tipo de Informação	Código da Informação	Tipo de Variável[4]	Descrição da Variável
M1	Ligar e desligar a CMR e Gerir as três Fases do Processo	**Entradas**	M1e1	VDEU	Ligar a CMR por iniciativa do usuário.
			M1e2	VDES	Finalizar a execução da Fase 1.
			M1e3	VDES	Finalizar a execução da Fase 2.
			M1e4	VDES	Finalizar a execução da Fase 3.
		Saídas	M1s1	VDSS	Iniciar a execução da Fase 1.
			M1s2	VDSS	Iniciar a execução da Fase 2.
			M1s3	VDSS	Continuar a execução da Fase 2.
			M1s4	VDSS	Iniciar a execução da Fase 3.
			M1s5	VDSS	Desligar a CMR.
		Controle	-	-	Não possui
		Recursos	M1r1	-	Aplicativo de software no ambiente *LabVIEW*™
			M1r2	-	Usuário
M2	Realizar *setup* pelo usuário	**Entradas**	M2e1	VDES	Iniciar o *setup.*
			M2e2	VDEU	Apagar os dados anteriores do *setup*.
			M2e3	VDEU	Finalizar a entrada de dados do *setup* por iniciativa do usuário.
			M2e4	VCEU	Parâmetros da CMR.
		Saídas	M2s1	VDSU	Informar dos antigos dados estão apagados e iniciar a entrada de novos dados do *setup*.
			M2s2	VDSS	Informar a finalização da entrada de dados do *setup* por iniciativa do usuário.
			M2s3	VCSS	Postura do robô para posição de segurança “HOME”.
			M2s4	VCSS	Postura do robô para posição de calibração.

[4] As legendas para tipo de variável do IDEF0:
VDES: Variável discreta de entrada gerada pelo sistema computacional;
VCES: Variável contínua de entrada gerada pelo sistema computacional;
VDEU: Variável discreta de entrada gerada pelo usuário;
VDSS: Variável discreta de saída gerada pelo sistema computacional;
VCSS: Variável contínua de saída gerada pelo sistema computacional;
VDSU: Variável discreta de saída para usuário;

			M2s5	VCSS	Parâmetros de calibração e ajuste entre o sistema de visão computacional e as coordenadas espaciais do robô.
			M2s6	VCSS	Parâmetros para reconhecimento de peças.
			M2s7	VCSS	Parâmetros de sequência de movimentação.
			M2s8	VCSS	Postura de saída das peças selecionadas.
		Controle	-	-	Não possui
		Recursos	M2r1	-	Aplicativo de software no ambiente *LabVIEW*™
			M2r2	-	Usuário
M3	Movimentar o robô.	**Entradas**	M3e1	VDES	Habilitar o módulo.
			M3e2	VDES	Desabilitar o módulo na Fase 2.
			M3e3	VDES	Desabilitar o módulo na Fase 3.
			M3e4	VDES	Movimentar o robô para postura de calibração.
			M3e5	VDES	Movimentar o robô para postura de segurança “HOME”.
			M3e6	VDES	Movimentar o robô para a postura da peça.
			M3e7	VDES	Informar a correção da possível colisão.
			M3e8	VDEU	Parar a movimentação do robô por iniciativa do usuário.
			M3e9	VDEU	Retomar a movimentação do robô por iniciativa do usuário.
			M3e10	VDES	Informar a correção da falha da execução do movimento do robô.
			M3e11	VCES	Postura do robô para posição de segurança “HOME”.
			M3e12	VCES	Postura do robô para posição de calibração.
			M3e13	VCES	Postura do robô para postura de movimentação de peças da entrada.
			M3e14	VCES	Postura do robô para postura de movimentação de peças da saída.
		Saídas	M3s1	VDSS	Indicar a chegada à postura de calibração.
			M3s2	VDSS	Indicar a chegada à postura de segurança “HOME”.
			M3s3	VDSS	Indicar a chegada à postura de peça para movimentação.
			M3s4	VDSS	Indicar possível colisão.
			M3s5	VDSS	Indicar falha da execução do movimento do robô.
		Controle	-	-	Não possui
		Recursos	M3r1	-	Aplicativo de software no ambiente *LabVIEW*™.
			M3r2	-	Robô Industrial – ROBOT 5150A.

M4	Calibrar e ajustar o robô – sistema de visão	**Entradas**	M4e1	VDES	Executar o módulo.
			M4e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
			M4e3	VCES	Parâmetros de setup de calibração.
		Saídas	M4s1	VDSS	Indicar o final da execução do módulo.
			M4s2	VDSS	Indicar a existência de anomalia.
			M4s3	VCSS	Parâmetros de calibração do robô e sistema de visão.
		Controle	-	-	Não possui
		Recursos	M4r1	-	Aplicativo de software no ambiente *LabVIEW*™.
			M4r2	-	Sistema de visão computacional.
			M4r3	-	Robô Industrial.
M5	Reconhecer peças pelo sistema de visão	**Entradas**	M5e1	VDES	Executar o módulo.
			M5e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
			M5e3	VCES	Parâmetros para reconhecimento de peças.
		Saídas	M5s1	VDSS	Indicar o final da execução do módulo.
			M5s2	VDSS	Indicar a existência de anomalia.
			M5s3	VCSS	Imagem das peças de entradas selecionadas.
		Controle	-	-	Não possui
		Recursos	M5r1	-	Aplicativo de software no ambiente *LabVIEW*™.
			M5r2		Sistema de visão.
M6	Determinar as posturas das peças selecionadas na entrada	**Entradas**	M6e1	VDES	Executar o módulo.
			M6e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
			M6e3	VCES	Parâmetros de calibração do robô e sistema de visão.
			M6e4	VCES	Imagem das peças de entradas selecionadas.
		Saídas	M6s1	VDSS	Indicar o final da execução do módulo.
			M6s2	VDSS	Indicar a existência de anomalia.
			M6s3	VCSS	Conjunto de postura das peças de entradas selecionadas.
		Controle	-	-	Não possui
		Recursos	M6r1	-	Aplicativo de software no ambiente *LabVIEW*™.
M7	Determinar a sequência de movimentação	**Entradas**	M7e1	VDES	Executar o módulo.
			M7e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
			M7e3	VCES	Parâmetros de sequência de movimentação.

		Saídas	M7s1	VDSS	Indicar o final da execução do módulo.
			M7s2	VDSS	Indicar a existência de anomalia.
			M7s3	VCSS	Parâmetros da sequência de movimentação.
		Controle	-	-	Não possui
		Recursos	M7r1	-	Aplicativo de software no ambiente *LabVIEW*™.
M8	Determinar as posturas da saída das peças	**Entradas**	M8e1	VDES	Executar o módulo.
			M8e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
			M8e3	VCES	Parâmetros de saída das peças selecionadas.
		Saídas	M8s1	VDSS	Indicar o final da execução do módulo.
			M8s2	VDSS	Indicar a existência de anomalia.
			M8s3	VCSS	Conjunto de postura da saída das peças.
		Controle	-	-	Não possui
		Recursos	M8r1	-	Aplicativo de software no ambiente *LabVIEW*™.
M9	Corrigir possível colisão, falha do robô, ou anomalia na Fase 2	**Entradas**	M9e1	VDES	Informar a existência de colisão, falha, ou anomalia.
			M9e2	VDEU	Indicar a correção por iniciativa do usuário.
		Saídas	M9s1	VDSS	Indicar a correção.
		Controle	-	-	Não possui
		Recursos	M9r1	-	Aplicativo de software no ambiente *LabVIEW*™.
			M9r2	-	Usuário.
M10	Corrigir possível colisão, ou falha no robô na Fase 3	**Entradas**	M10e1	VDES	Informar a existência de colisão, ou falha, ou anomalia.
			M10e2	VDEU	Indicar a correção por iniciativa do usuário.
		Saídas	M10s1	VDSS	Indicar a correção.
		Controle	-	-	Não possui
		Recursos	M10r1	-	Aplicativo de software no ambiente *LabVIEW*™.
			M10r1	-	Usuário.
M11	Abrir e Fechar a garra do robô	**Entradas**	M11e1	VDES	Sinal para habilitar o módulo da ferramenta do robô.
			M11e2	VDES	Sinal para desabilitar o módulo da ferramenta do robô.
		Saídas	M11s1	VDSS	Sinal de ferramenta do robô ligada.
			M11s2	VDSS	Sinal de ferramenta do robô desligada.
		Controle	-	-	Não possui

		Recursos	M11r1	-	Aplicativo de software no ambiente *LabVIEW™*.
			M11r2	-	Garra do robô.
M12	Gerir a movimentação das peças	**Entradas**	M12e1	VDES	Habilitar a gestão da movimentação de peças.
			M12e2	VDES	Informar o final da movimentação de peça.
			M12e3	VCES	Parâmetros da sequência de movimentação.
		Saídas	M12s1	VDSS	Iniciar/continuar movimentação de peças.
			M12s2	VDSS	Encerrar a movimentação de peças.
		Controle	-	-	Não possui
		Recursos	M12r1	-	Aplicativo de software no ambiente *LabVIEW™*.

4.1.3 Etapa 3: Modelar os autômatos

Nesta seção são modelados os autômatos para cada um dos módulos criados na etapa 2 e os autômatos para atendimento das especificações informadas no documento de requisitos elaborado na etapa 1.

Para cada módulo concebido na etapa anterior é criado um modelo de autômato. Os módulos criados possuem variáveis discretas e contínuas, assim o autômato será responsável pela representação do comportamento das variáveis discretas.

Tabela 9. Descrição dos eventos de cada autômato.

Autômato (G_i)	**Descrição do Autômato**	**Evento (Σ)**	**Tipo de Evento[5]**	**Variável IDEF0**	**Tipo de Variável do IDEF0**	**Descrição do Evento**
G1	Ligar e desligar a CMR e Gerir as três Fases do Processo	a1	EEN	M1e1	VDEU	Ligar a CMR por iniciativa do usuário.
		b1	EIN	M1s1	VDSS	Iniciar a execução da Fase 1.
		c1	EIC	M1e2	VDES	Finalizar a execução da Fase 1.
		d1	EIN	M1s2	VDSS	Iniciar a execução da Fase 2.
		e1	EIN	M1s3	VDSS	Continuar a execução da Fase 2.

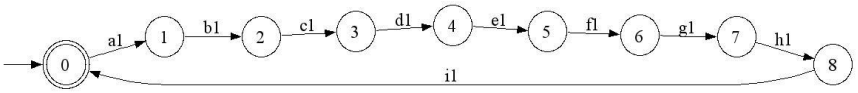
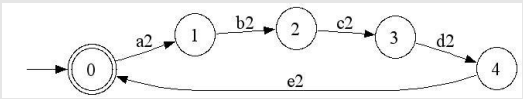
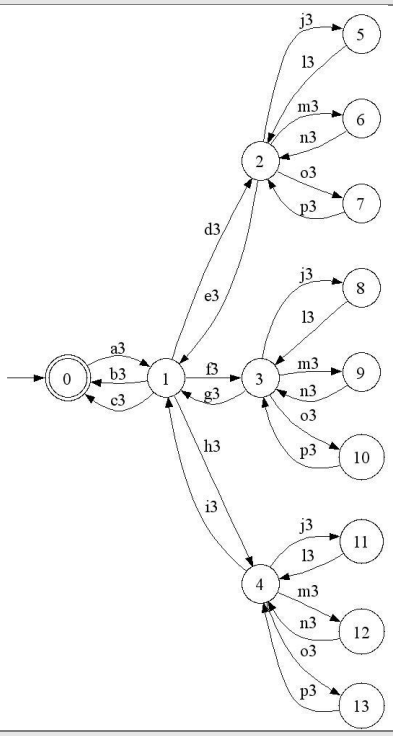
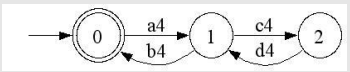
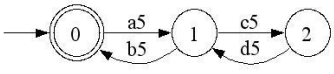
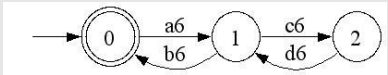
[5] As legendas para cada tipo de evento:
EIC: Evento interno controlado;
EIN: Evento interno não controlado;
EEN: Evento externo não controlado.

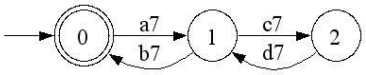
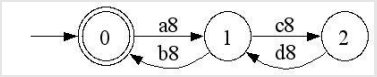
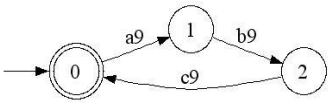
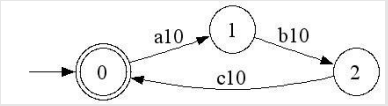
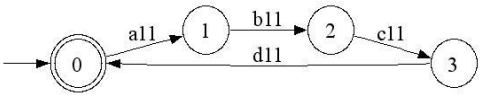
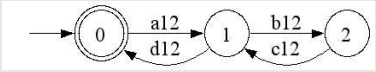
		f1	EIC	M1e3	VDES	Finalizar a execução da Fase 2.
		g1	EIN	M1s4	VDSS	Iniciar a execução da Fase 3.
		h1	EIC	M1e4	VDES	Finalizar a execução da Fase 3.
		i1	EIN	M1s5	VDSS	Desligar a CMR.
G2	Realizar *setup* pelo usuário	a2	EIC	M2e1	VDES	Iniciar o *setup.*
		b2	EEN	M2e2	VDEU	Apagar os dados anteriores do *setup*.
		c2	EIN	M2s1	VDSU	Informar dos antigos dados estão apagados e iniciar a entrada de novos dados do *setup*.
		d2	EEN	M2e3	VDEU	Finalizar a entrada de dados do *setup* por iniciativa do usuário.
		e2	EIN	M2s2	VDSS	Informar a finalização da entrada de dados do *setup* por iniciativa do usuário.
G3	Movimentar o robô	a3	EIC	M3e1	VDES	Habilitar o módulo.
		b3	EIC	M3e2	VDES	Desabilitar o módulo na Fase 2.
		c3	EIC	M3e3	VDES	Desabilitar o módulo na Fase 3.
		d3	EIC	M3e4	VDES	Movimentar o robô para de calibração.
		e3	EIN	M3s1	VDSS	Indicar a chegada à postura de calibração.
		f3	EIC	M3e5	VDES	Movimentar o robô para postura de segurança “HOME”.
		g3	EIN	M3s2	VDSS	Indicar a chegada à postura de segurança “HOME”.
		h3	EIC	M3e6	VDES	Movimentar o robô para a postura da peça.
		i3	EIN	M3s3	VDSS	Indicar a chegada à postura de peça para movimentação.
		j3	EIN	M3s4	VDSS	Indicar possível colisão.
		l3	EIC	M3e7	VDES	Informar a correção da possível colisão.
		m3	EEN	M3e8	VDEU	Parar a movimentação do robô por iniciativa do usuário.
		n3	EEN	M3e9	VDEU	Retomar a movimentação do robô por iniciativa do usuário.
		o3	EIN	M3s5	VDSS	Indicar falha da execução do movimento do robô.
		p3	EIC	M3e10	VDES	Informar a correção da falha da execução do movimento do robô.
G4	Calibrar e ajustar robô – sistema de visão	a4	EIC	M4e1	VDES	Executar o módulo.
		b4	EIN	M4s1	VDSS	Indicar o final da execução do módulo.
		c4	EIN	M4s2	VDSS	Indicar a existência de anomalia.
		d4	EIC	M4e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).

G5	Reconhecer peças pelo sistema de visão	a5	EIC	M5e1	VDES	Executar o módulo.
		b5	EIN	M5s1	VDSS	Indicar o final da execução do módulo.
		c5	EIN	M5s2	VDSS	Indicar a existência de anomalia.
		d5	EIC	M5e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
G6	Determinar as posturas das peças selecionadas na entrada	a6	EIC	M6e1	VDES	Executar o módulo.
		b6	EIN	M6s1	VDSS	Indicar o final da execução do módulo.
		c6	EIN	M6s2	VDSS	Indicar a existência de anomalia.
		d6	EIC	M6e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
G7	Determinar a sequência de movimentação	a7	EIC	M7e1	VDES	Executar o módulo.
		b7	EIN	M7s1	VDSS	Indicar o final da execução do módulo.
		c7	EIN	M7s2	VDSS	Indicar a existência de anomalia.
		d7	EIC	M7e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
G8	Determinar as posturas da saída das peças	a8	EIC	M8e1	VDES	Executar o módulo.
		b8	EIN	M8s1	VDSS	Indicar o final da execução do módulo.
		c8	EIN	M8s2	VDSS	Indicar a existência de anomalia.
		d8	EIC	M8e2	VDES	Reiniciar a execução do módulo (após correção de anomalia).
G9	Corrigir possível colisão, ou falha do robô, ou anomalia na Fase 2	a9	EIC	M9e1	VDES	Informar a existência de colisão, ou falha, ou anomalia.
		b9	EEN	M9e2	VDEU	Indicar a correção por iniciativa do usuário.
		c9	EIN	M9s1	VDSS	Indicar a correção.
G10	Corrigir possível colisão e falha no robô na Fase 3	a10	EIC	M10e1	VDES	Informar a existência de colisão, falha, ou anomalia.
		b10	EEN	M10e2	VDEU	Indicar a correção por iniciativa do usuário.
		c10	EIN	M10s1	VDSS	Indicar a correção.
G11	Abrir e Fechar a garra do robô	a11	EIC	M11e1	VDES	Abrir a garra do robô.
		b11	EIN	M11s1	VDSS	Indicar a abertura da garra do robô.
		c11	EIC	M11e2	VDES	Fechar a garra do robô.
		d11	EIN	M11s2	VDSS	Indicar o fechamento da garra do robô.
G12	Gerir a movimentação das peças	a12	EIC	M12e1	VDES	Habilitar a gestão da movimentação de peças.
		b12	EIN	M12s1	VDSS	Iniciar/continuar movimentação de peças.
		c12	EIC	M12e2	VDES	Informar o final da movimentação de peça.
		d12	EIN	M12s2	VDSS	Encerrar a movimentação de peças.

O diagrama de cada autômato é mostrado na Tabela 10.

Tabela 10. Diagrama de cada autômato modelado para o processo.

Autômato (Gi)	Descrição do Autômato	Diagrama do modelo do Autômato
G1	Ligar e desligar a CMR e Gerir as três Fases do Processo	
G2	Realizar *setup* pelo usuário	
G3	Movimentar o robô	
G4	Calibrar e ajustar robô – sistema de visão	
G5	Reconhecer peças pelo sistema de visão	
G6	Determinar as posturas das peças selecionadas na entrada	

G7	Determinar a sequência de movimentação	
G8	Determinar as posturas da saída das peças	
G9	Corrigir possível colisão, falha do robô, ou anomalia na Fase 2	
G10	Corrigir possível colisão, ou falha no robô na Fase 3	
G11	Abrir e Fechar a garra do robô	
G12	Gerir a movimentação das peças	

Os modelos dos autômatos apresentados possuem os seguintes eventos não controláveis: a1, b1, d1, e1, g1, i1, b2, c2, d2, e3, g3, i3, j3, m3, n3, o3, b4, c4, b5, c5, b6, c6, b7, c7, b8, c8, b9, c9, b10, c10, b11, d11, b12 e d12, ou seja, o sistema supervisório não pode desabilitá-los. A partir do documento de requisitos foram criados cinco restrições genéricas ($R_{gen,i}, i = 1, \dots, 5$) que são mostrados na Figura 37 e sendo cada um:

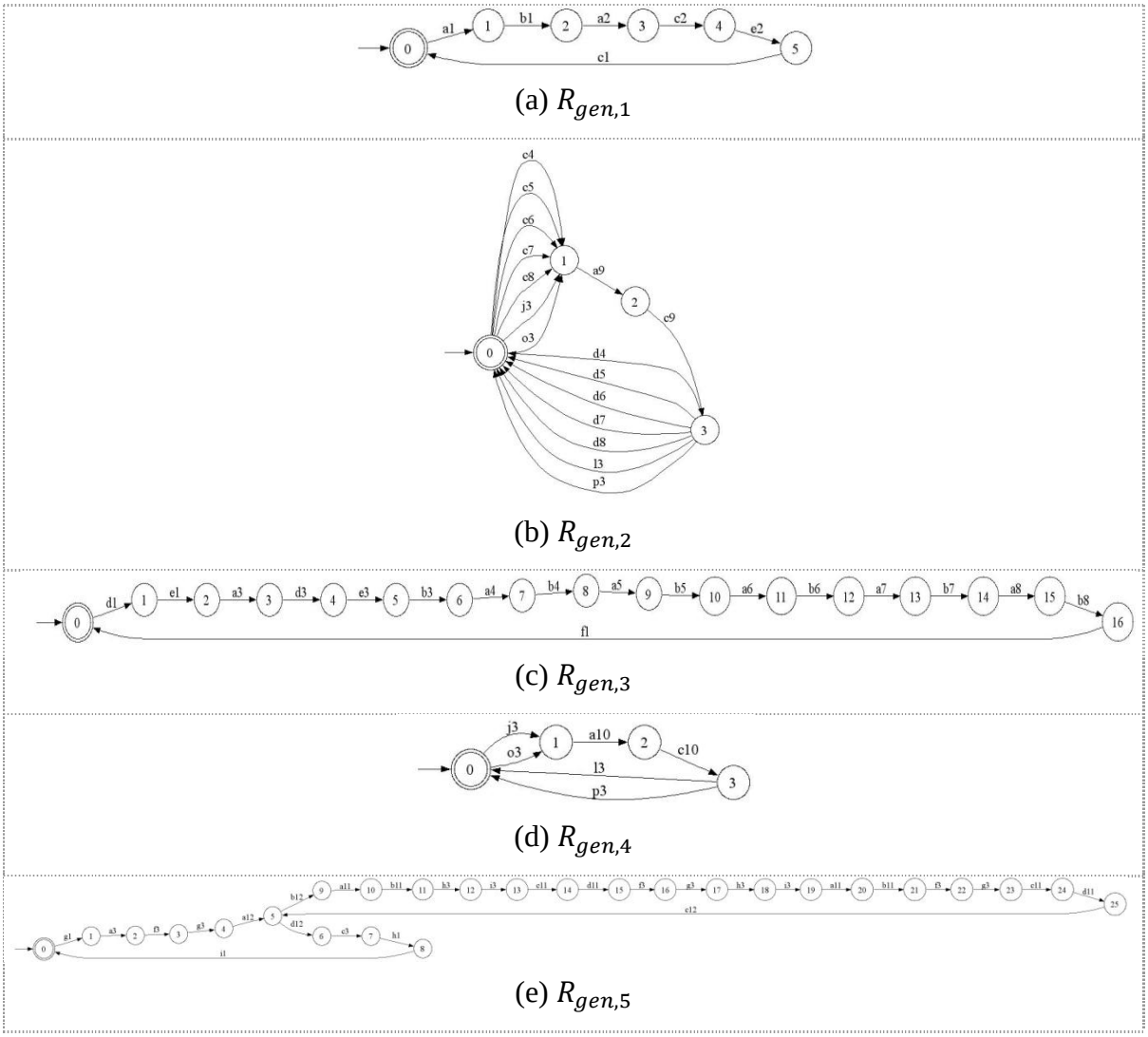
$R_{gen,1}$: Modo de Inicialização da CMR;

$R_{gen,2}$: Gestão da correção de possível colisão, falha no robô e anomalia;

$R_{gen,3}$: Modo de Seleção e Determinação de Sequência para Movimentação de Peças;

$R_{gen,4}$: Gestão da correção de possível colisão, falha no robô;

$R_{gen,5}$: Modo de Execução da Movimentação de Peças.



(a) $R_{gen,1}$

(b) $R_{gen,2}$

(c) $R_{gen,3}$

(d) $R_{gen,4}$

(e) $R_{gen,5}$

Figura 37. Autômatos das restrições.

4.1.4 Etapa 4: Criar o SML

Nesta seção, será construído o sistema supervisório modular local (SML) para atuar na CMR. Para tal, são construídos três restrições locais $R_{loc,i}, i = 1, \dots, 3$ pela composição síncrona das restrições genéricas $R_{gen,1}, i = 1, \dots, 5$ que possuem eventos comuns, sendo:

$$R_{loc,1} = R_{gen,1}$$

$$R_{loc,2} = R_{gen,2} || R_{gen,3}$$

$$R_{loc,3} = R_{gen,4} || R_{gen,5}$$

A partir da composição entre os modelos dos autômatos criados na Etapa 3 são criados três plantas locais $G_{loc,i}, i = 1, \dots, 3$ para cada restrição local $R_{loc,i}, i = 1, \dots, 3$. A Tabela 11 apresenta a relação entre as plantas locais criadas e suas respectivas restrições locais.

Tabela 11. Relação entre as plantas locais e as respectivas restrições locais.

Planta Local	Restrição Local
$G_{loc,1} = G1 \| \| G2$	$R_{loc,1}$
$G_{loc,2} = G1 \| \| G3 \| \| G4 \| \| G5 \| \| G6 \| \| G7 \| \| G8 \| \| G9$	$R_{loc,2}$
$G_{loc,3} = G1 \| \| G3 \| \| G10 \| \| G11 \| \| G12$	$R_{loc,3}$

Em seguida, são criadas as especificações locais $(E_{loc,i}, i = 1, \dots, 3)$ pela composição da planta local $(G_{loc,i}, i = 1, \dots, 3)$ com a sua respectiva restrição local $(R_{loc,i}, i = 1, \dots, 3)$. Os estados proibidos acessados pelos eventos indesejáveis foram eliminados, conforme a Tabela 12, gerando as especificações locais sem os estados proibidos $(E_{\mathrm{loc\_sp,i}}, \mathrm{i} = 1, \dots, 3)$.

Tabela 12. Relação entre a especificação local e seus eventos indesejáveis.

Especificação Local	Eventos Indesejáveis
$E_{loc,1} = G_{loc,1} \| \| R_{loc,1}$	Nenhum.
$E_{loc,2} = G_{loc,2} \| \| R_{loc,2}$	f3, g3 e c3.
$E_{loc,3} = G_{loc,3} \| \| R_{loc,3}$	d3 e b3.

Para cada especificação local $(E_{loc\_sp,i}, i = 1, \dots, 3)$ é calculado a componente trim $(E_{loc\_trim,i}, i = 1, \dots, 3)$, eliminando os estados bloqueantes. Em seguida, é calculado a linguagem controlável $(S_{loc,i} = supC(G_{loc,i}, E_{loc\_trim,i}), i = 1, \dots, 3)$, que é a síntese de cada supervisor modular local que realiza, para cada planta local $(G_{loc,i}, i = 1, \dots, 3)$, a especificação local não bloqueante ótima $(E_{loc\_trim,i}, i = 1, \dots, 3)$, sendo menos restritivo possível e atendendo a condição de controlabilidade. Para finalizar a Etapa 4, são obtidos os supervisores locais reduzidos $(S_{red,i}, i = 1, \dots, 3)$, apresentados na Figura 38.

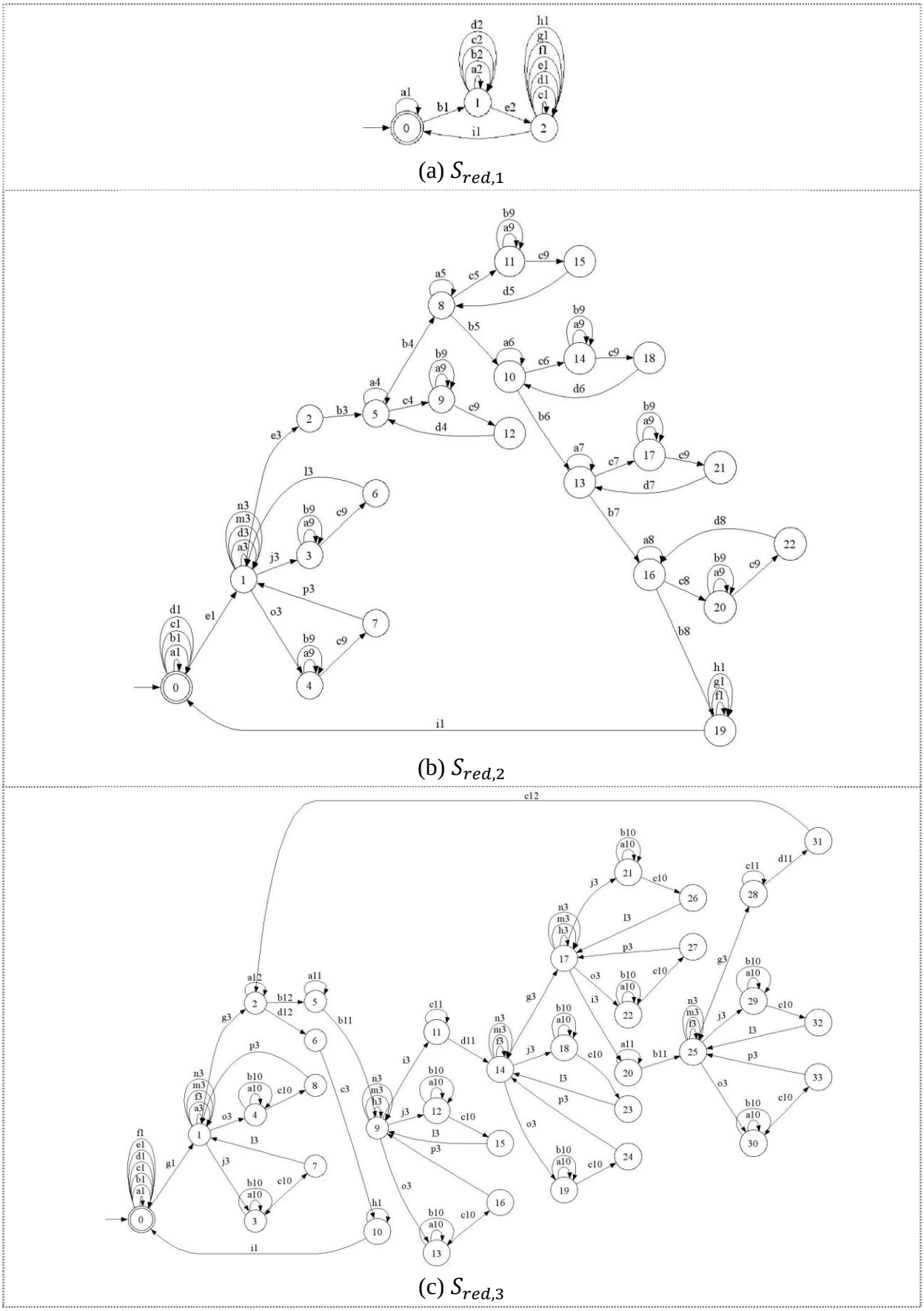


Figura 38. Supervisores modulares locais reduzidos.

4.1.5 Etapa 5: Verificar os modelos de autômatos e SML

Nesta seção, Os modelos dos autômatos ($G_i, i = 1,\ldots,3$) e os SML ($S_{red,i}, i = 1,\ldots,3$) são verificados quanto a serem não bloqueantes (ou seja, coacessíveis e acessíveis) utilizando a ferramenta computacional *Grail*, conforme Figura 39.

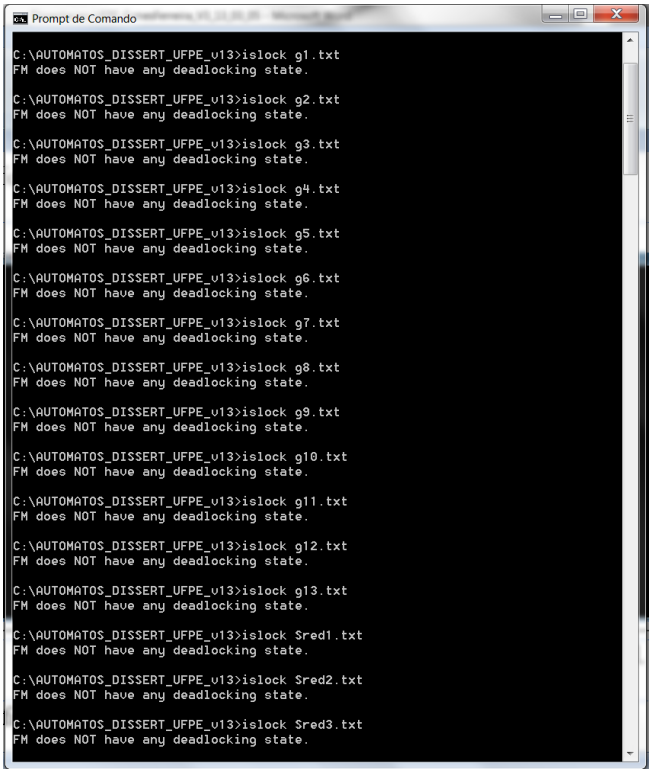


Figura 39. Verificação de não bloqueante dos Autômatos e Supervisórios reduzidos

Também, os supervisórios reduzidos locais do SML ($S_{red,i}, i = 1,\ldots,3$) são verificados quanto a serem não conflitantes entre si utilizando a ferramenta computacional *Grail*, conforme Figura 40.

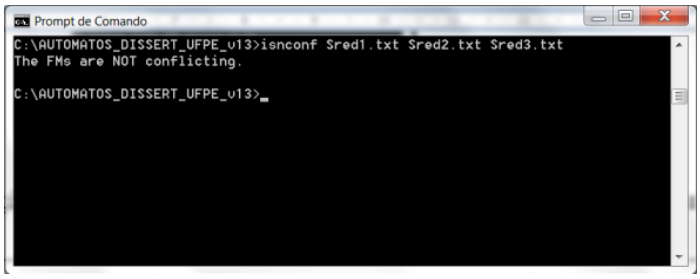
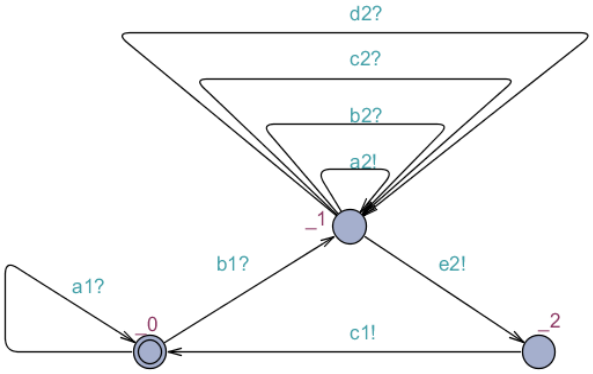
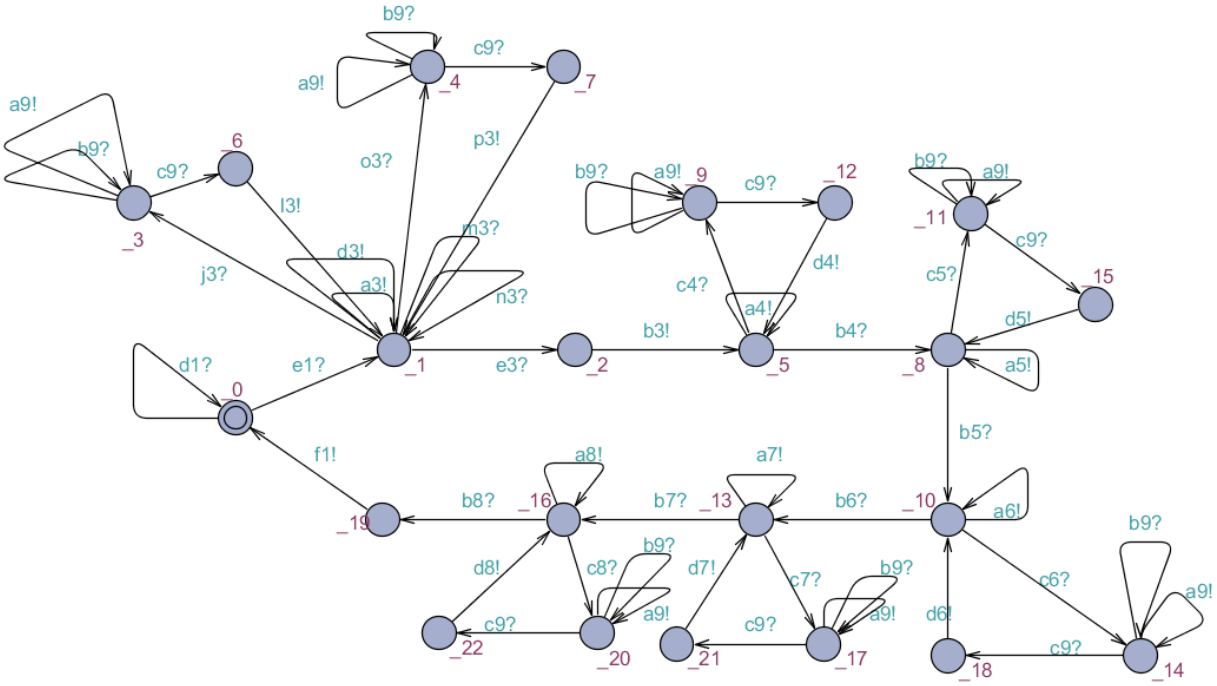


Figura 40. Verificação do não conflito entre os supervisórios reduzidos locais.

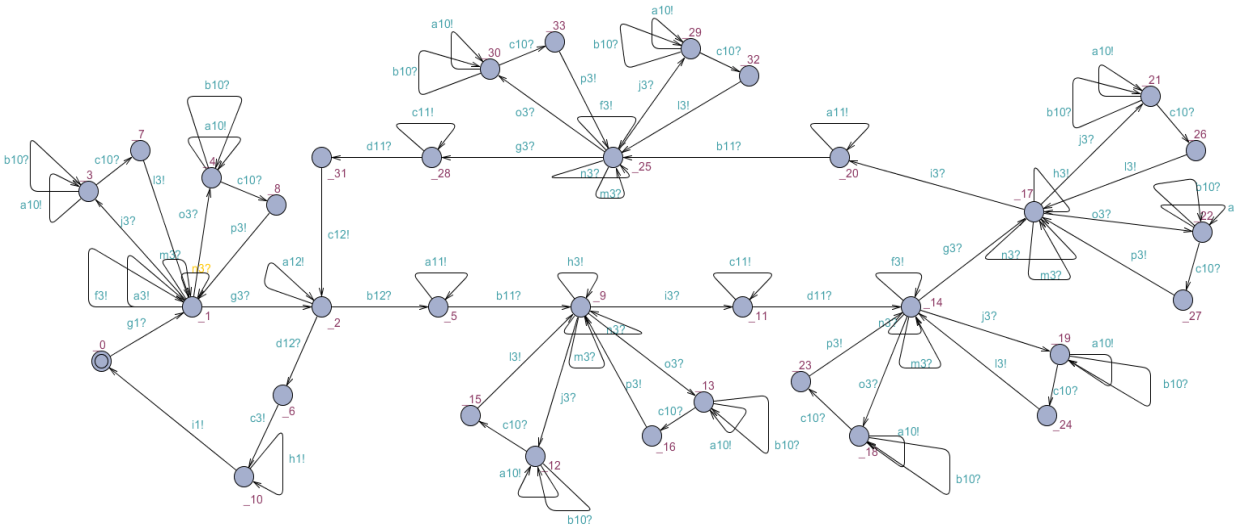
Os modelos de autômatos e SML são desenhados na ferramenta computacional *Uppaal*. Foram eliminados os eventos que geravam não coordenação do supervisório reduzido conforme Figura 41. Os eventos eliminados não alteram a estrutura de supervisão, foram retirados apenas os eventos do modelo de autômato $G1$.



(a) $S_{red,1}$



(b) $S_{red,2}$



(c) $S_{red,3}$

Figura 41. Supervisórios reduzidos locais com eventos eliminados.

Em seguida, é realizada a simulação no *Uppaal* conforme Figura 42.

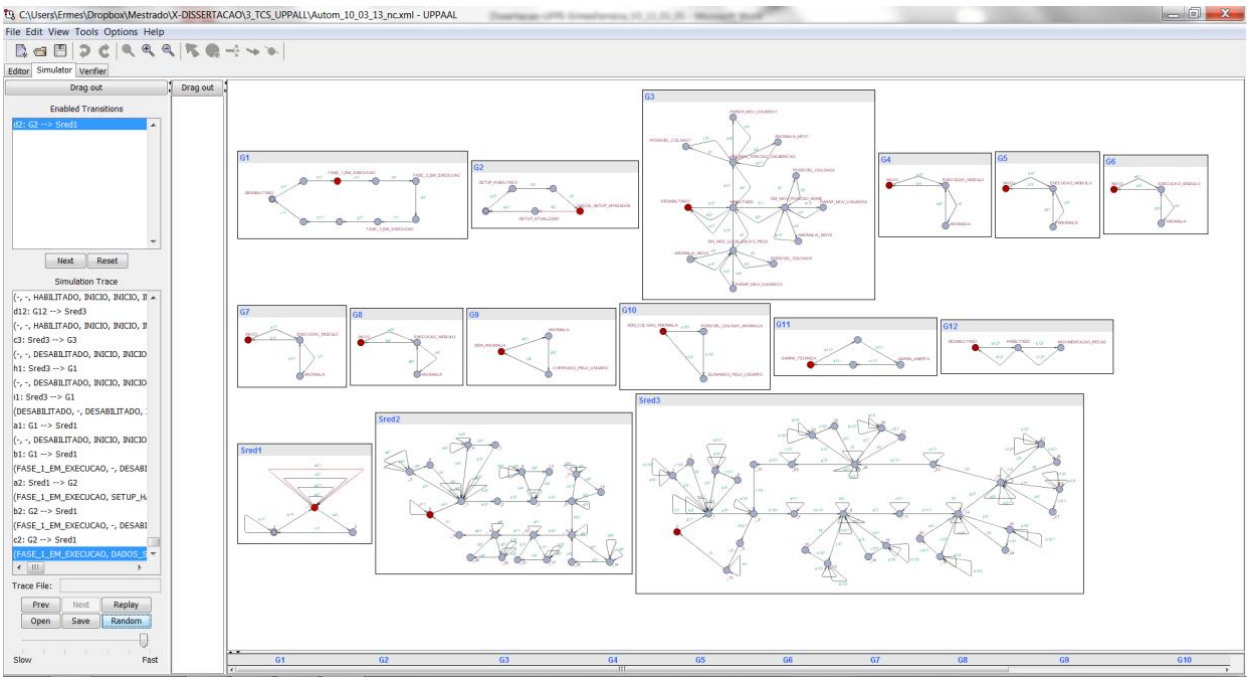


Figura 42. Simulação dos autômatos e supervisórios.

A verificação de modelo é realizada para examinar a não existência de *deadlock* e o atendimento aos requisitos do sistema supervisório do requisito não funcional NF004. A seguir são listadas em linguagem de verificação do Uppaal três especificações:

1. *A[] (G10.POSSIVEL_COLISAO_ANOMALIA or G10.ELIMINADO_PELO_USUARIO) imply (G3.ANOMALIA_MOV2 or G3.ANOMALIA_MOV3 or G3.POSSIVEL_COLISAO2 or G3.POSSIVEL_COLISAO3):* Sempre que existir uma anomalia na execução do movimento do robô, ou uma possível colisão do robô e somente retorne a normalidade após correção do usuário durante a fase 3.
2. *A[] (G3.ANOMALIA_MOV1 or G3.ANOMALIA_MOV2 or G3.ANOMALIA_MOV3 or G3.POSSIVEL_COLISAO1 or G3.POSSIVEL_COLISAO2 or G3.POSSIVEL_COLISAO3 or G3.PARAR_MOV_USUARIO1 or G3.PARAR_MOV_USUARIO2 or G3.PARAR_MOV_USUARIO3) imply not (G3.EM_MOV_POSICAO_CALIBRACAO or G3.EM_MOV_POSICAO_HOME or G3.EM_MOV_LOCALIZACAO_PECA)*: Interromper o movimento do robô devido a existência de uma possível colisão, ou anomalia, ou por iniciativa do usuário.
3. *A[] (G9.ANOMALIA or G9.CORRIGIDO_PELO_USUARIO) imply (G3.ANOMALIA_MOV1 or G3.POSSIVEL_COLISAO1 or G4.ANOMALIA or G5.ANOMALIA or G6.ANOMALIA or G7.ANOMALIA or G8.ANOMALIA):* Sempre que existir uma anomalia na calibração e ajuste do robô e sistema de visão, ou do reconhecimento de peças pelo sistema de visão só retorne a normalidade após correção do usuário, ou da determinação das posturas das peças selecionadas na entrada para serem movimentadas, e possível colisão ou falha no movimento do robô durante a fase 2, somente retorne a normalidade após correção do usuário, ou da determinação do sequênciamento da movimentação das peças, ou da determinação das posturas de saída das peças movimentadas, somente retorne a normalidade após correção do usuário.

A Figura 43 apresenta o resultado da verificação de modelo.

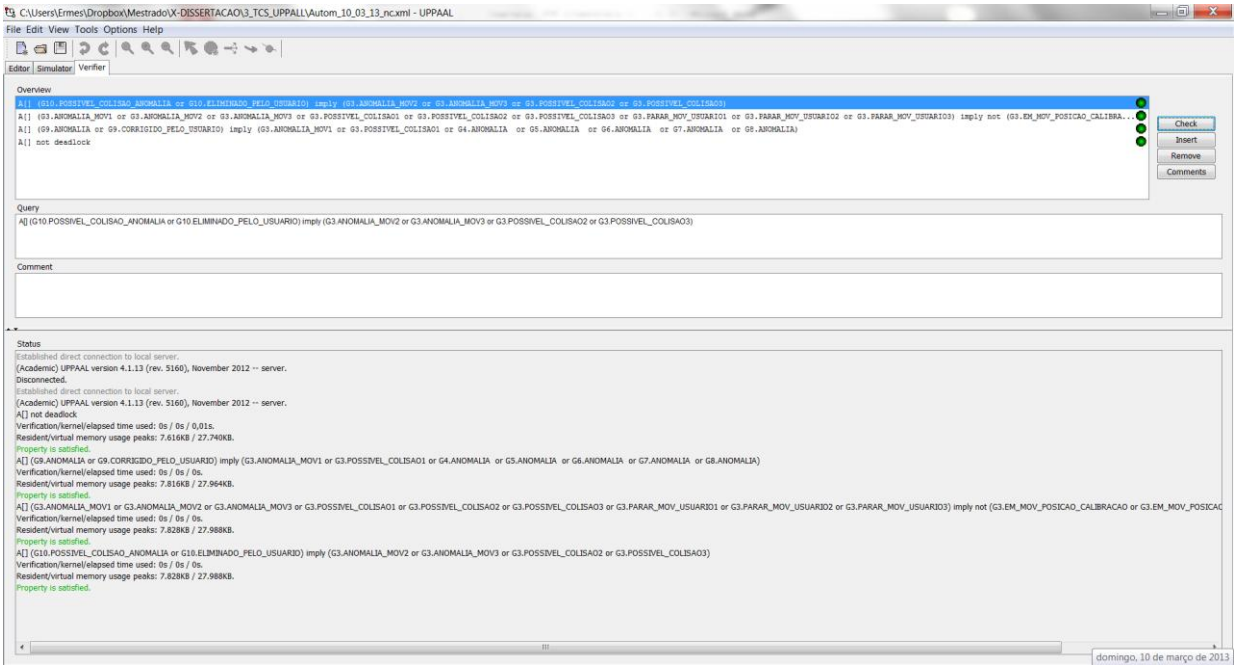


Figura 43. Verificação de modelos utilizando o *UPPAAL*.

4.1.6 Etapa 6: Converter e simular os modelos no ambiente *LabVIEW™*

Esta seção apresenta a modelagem gráfica dos autômatos e supervisório reduzidos modificados da Etapa 5 no ambiente *LabVIEW™* utilizando o *toolbox State Diagram*. As Figura 44 e Figura 45 exemplificam as conversões.

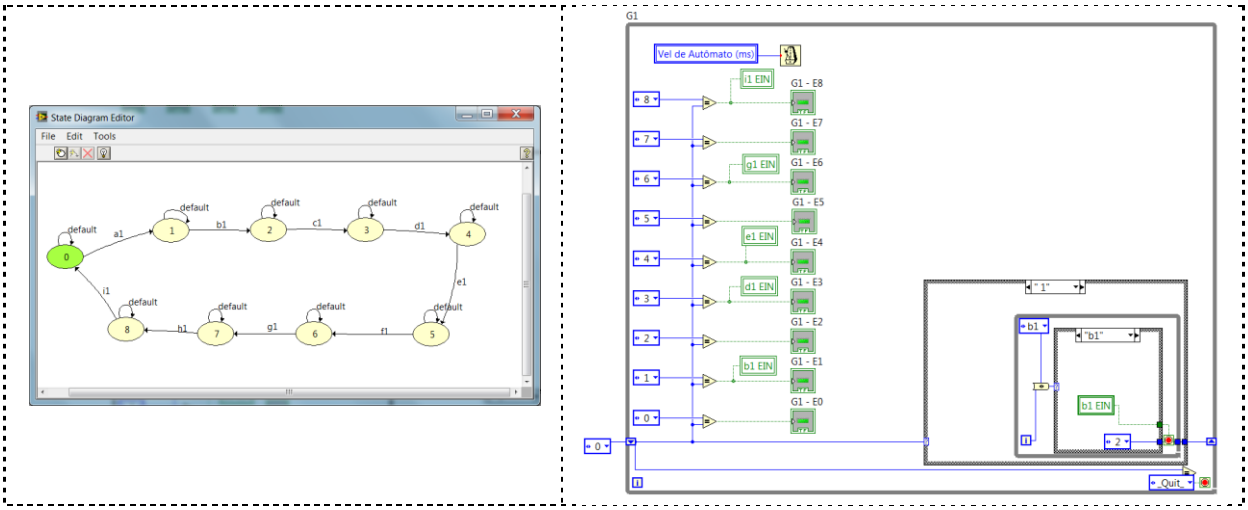


Figura 44. Conversão do autômato *G1*.

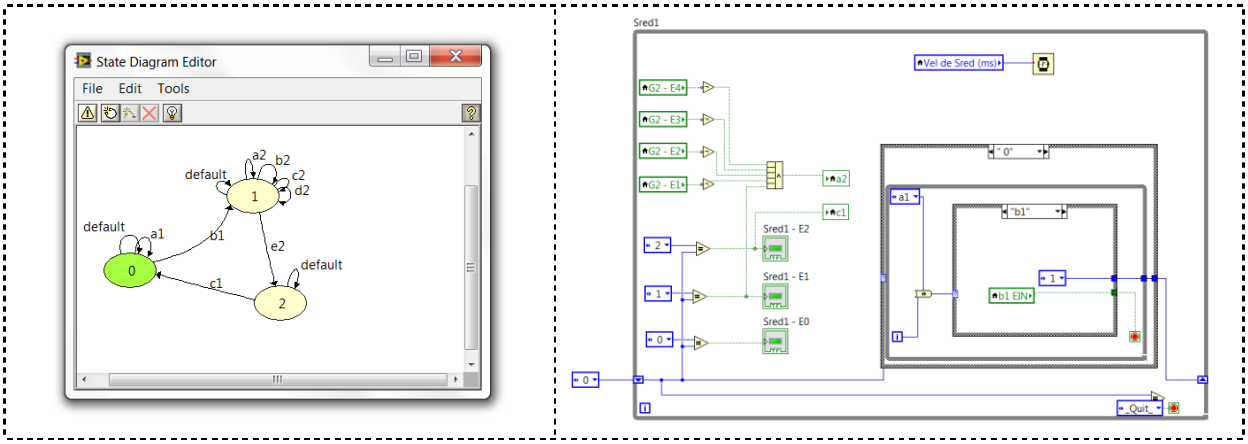


Figura 45. Conversão do $S_{red,1}$.

A Figura 46 ilustra o painel frontal (*Front panel*) do SML implementado na plataforma *LabVIEW™*. São realizadas simulações neste programa para verificar o bom funcionamento do sistema.

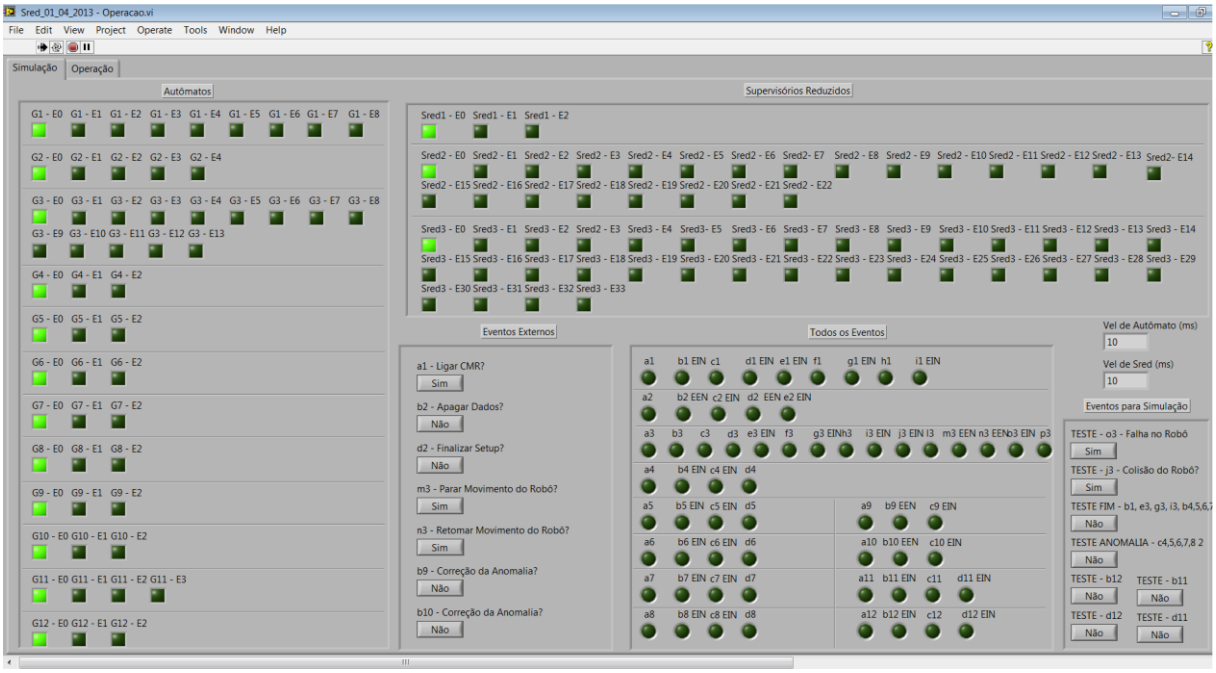


Figura 46. *Front panel* do SML na plataforma *LabVIEW™*.

4.1.7 Etapa 7: Integrar os módulos computacionais aos modelos no ambiente *LabVIEW™*

Esta seção apresenta a criação dos programas e integração dos módulos computacionais (*hardware* e *software*) aos modelos dos autômatos e SML. Foram criados os módulos computacionais para a execução das funções descritas na Tabela 13 e a programação de cada um é apresentada no Apêndice F.

Tabela 13. Módulos Computacionais desenvolvidos.

Módulo	Função do Módulo	Fase da aplicação do módulo
M3	Movimentar o robô	Fase 2 e 3
M4	Calibrar e ajustar robô ao sistema de visão	Fase 2
M5	Reconhecer peças pelo sistema de visão	Fase 2
M6	Determinar as posturas das peças selecionadas na entrada para serem movimentadas	Fase 2
M7	Determinar a sequência de movimentação	Fase 2
M8	Determinar as posturas da saída das peças	Fase 2
M11	Abrir e Fechar a garra do robô	Fase 3
M12	Gerir a movimentação das peças	Fase 3

A Figura 47 ilustra a integração entre o M3, módulo computacional de movimentação do robô e o G3, seu respectivo autômato.

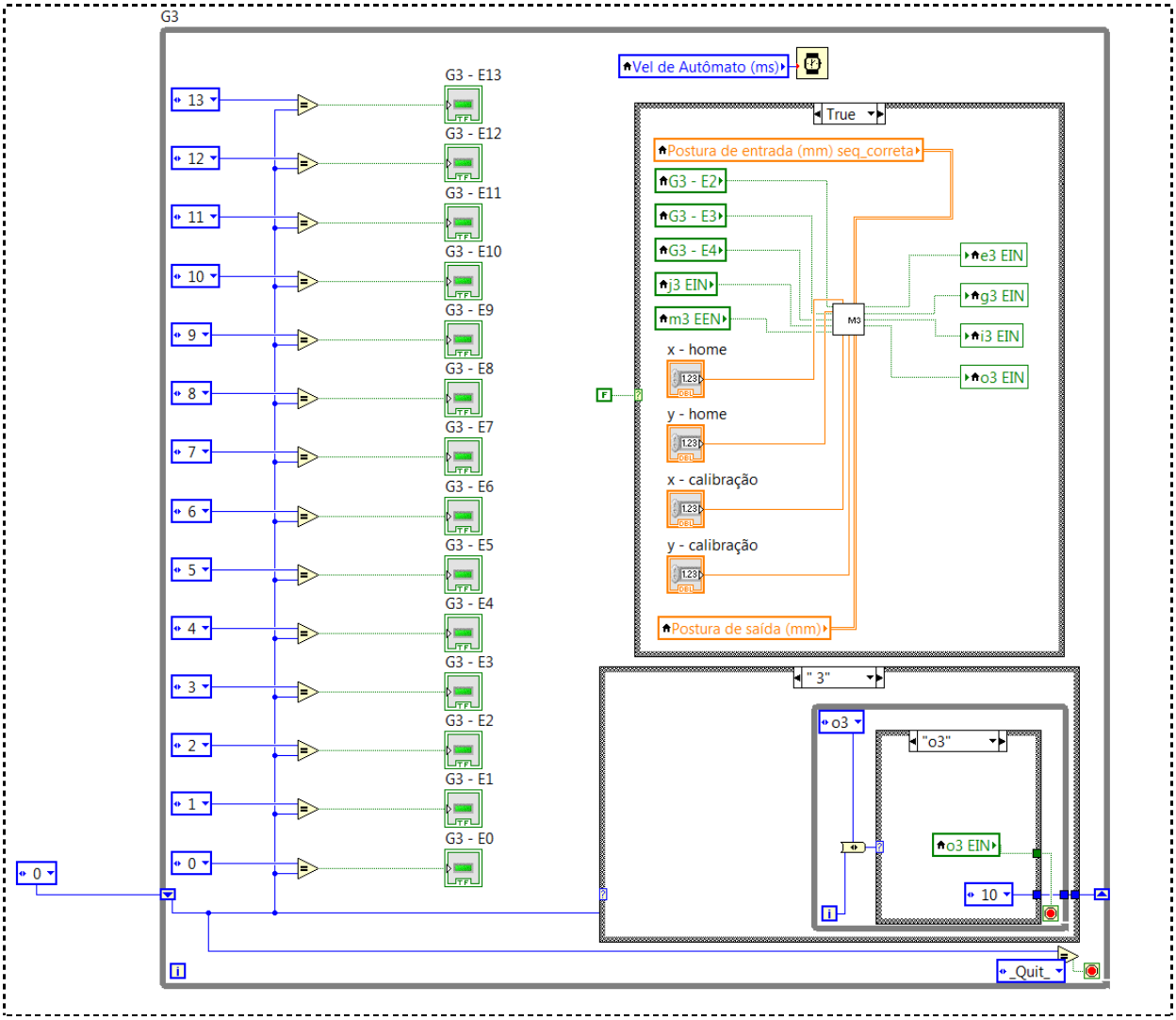


Figura 47. Integração do M3 com o Autômato G3.

4.1.8 Etapa 8: Criar IHM de programação no nível de tarefas da CMR no ambiente *LabVIEW™*

Nesta etapa será criada a interface homem máquina (IHM) com abstração para programação no nível de tarefas da CMR e o teste de conectividade no sistema. O desenvolvimento do ambiente para a programação no nível de tarefas da CMR é apresentado na Figura 48.

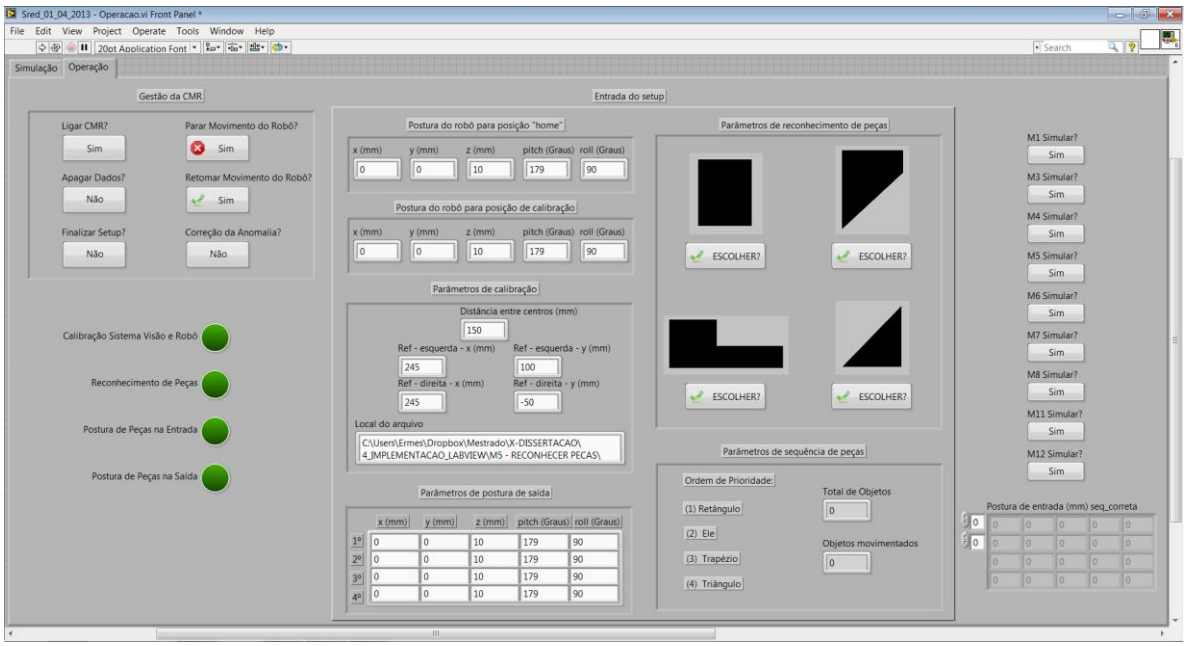


Figura 48. Ambiente para programação nível de tarefas da CMR.

4.1.9 Etapa 9: Testar a operação da CMR

Nesta seção, para testar o processo de criação de módulos, modelagem dos autômatos e SML e criação dos programas no ambiente *LabVIEW™*, alguns experimentos foram realizados na CMR para o processo de seleção e movimentação de peças. Cinco simulações foram realizadas com sucesso, sendo estas:

1. Procedimento de calibração do sistema de visão e robô, reconhecimento de peças e determinação de postura;
2. Procedimento de verificação do processo conforme documento de requisitos;
3. Procedimento de operação da CMR para seleção e movimentação de duas peças (triângulo e éle);
4. Procedimento de operação da CMR para seleção e movimentação de quatro peças (triângulo, éle, retângulo e trapézio);
5. Procedimento de solução para anomalia, possível colisão, ou parada do robô por iniciativa do usuário.

A Figura 49 ilustra a CMR experimental montada.

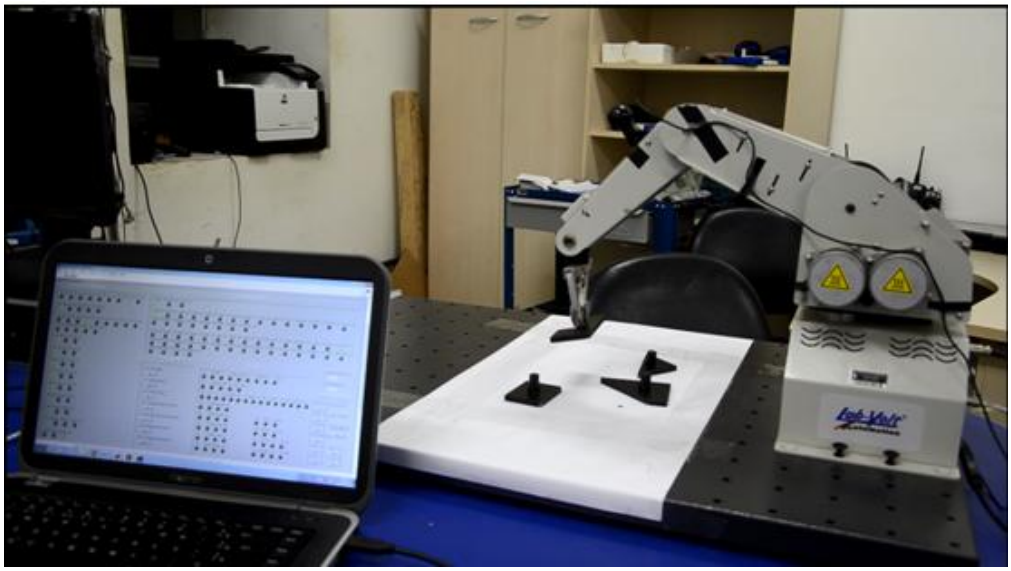


Figura 49. CMR experimental.

4.2 Resultados Obtidos

Os principais resultados obtidos com a aplicação do método proposto ao estudo de caso foi o estabelecimento de uma sistematização para o uso de diversas técnicas para as atividades de projeto. As etapas ficaram bem definidas durante a elaboração do projeto, que facilitou a identificação de erros e proporcionou a implantação de melhorias, pois foi possível utilizando o único formalismo, baseado em autômatos, acompanhar todas as etapas de desenvolvimento. A Tabela 14 apresenta a relação entre os elementos básicos de autômatos com todas as técnicas utilizadas no método.

Tabela 14. Comparação entre as técnicas utilizadas no método.

Elementos Básicos de Autômatos	**Modelo Espiral de Engenharia de Requisitos**	**Diagrama IDEF0**	**Teoria de Controle Supervisório**	**Sistema Supervisório Modular Local**	**Verificação de Modelos**	**Ferramenta da *State Diagram* do ambiente *LabVIEW*™**
Processo	Sim (Descrição Geral do Processo)	Sim (conexões de modelos em redes)	Sim (modelo das restrições)	Sim (modelo das especificações)	Sim (sistema)	Sim (programa)
Atividade	Sim (requisitos funcionais)	Sim (caixa)	Sim (modelos dos autômatos)	Sim (modelos dos supervisórios)	Sim (modelos dos supervisórios)	Sim (*State Diagram*)
Estados	Não	Não	Sim	Sim	Sim	Sim

Eventos	Sim (entradas e saídas)	Sim (setas da lateral de entradas e saídas)	Sim (sinais discretos)	Sim (sinais discretos)	Sim (sinais discretos)	Sim (sinais discretos)
Recursos	Sim (recursos)	Sim (setas de baixo)	Sim	Sim	Sim	Sim
Início/Fim	Não	Não	Sim	Sim	Sim	Sim

O uso do documento de requisitos proporcionou o acompanhamento em todas as etapas de projeto pelo cliente e guiou as alterações identificadas no decorrer da aplicação do método. As relações entre as informações do documento de requisitos e símbolos gráficos do autômato são apresentadas na Tabela 15. É observada a existência de símbolos gráficos comuns nas conversões entre as técnicas. Como exemplo, as setas de entradas do diagrama IDEF0 que representam sinais discretos gerados pelo sistema podem ser traduzidas em eventos para os modelos de autômatos da teoria de controle supervisório.

Tabela 15. Tradução dos símbolos gráficos entre as técnicas utilizadas no método.

Modelo Espiral de Engenharia de Requisitos	**Diagrama IDEF0**	**Teoria de Controle Supervisório**	**Sistema Supervisório Modular Local**	**Verificação de Modelos**	**Ferramenta da *State Diagram* do ambiente *LabVIEW™***
Descrição geral do processo	Relação entre as caixas	Conjunto de estados e transições (modelo das restrições)	Conjunto de estados e transições (modelo das especificações)	Conjunto de estados e transições (modelo das especificações)	Conjunto de estados e transições (modelo das especificações)
Requisito Funcional (RF)	Caixa	Conjunto de estados e transições (modelo dos autômatos)	Conjunto de estados e transições (modelo dos autômatos)	Conjunto de estados e transições (modelo dos autômatos)	Conjunto de estados e transições (modelo dos autômatos)
-	-	-	Conjunto de estados e transições (modelo dos supervisórios)	Conjunto de estados e transições (modelo dos supervisórios)	Conjunto de estados e transições (modelo dos supervisórios)
Entrada do RF	Seta da lateral esquerda	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto
Saída do RF	Seta da lateral direita	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto	Transição de eventos do sinal discreto
Recursos do RF	Seta de baixo	-	-	-	-
Descrição do RF	Seta de cima	-	-	-	-

Já o uso de diagrama IDEF0 proporcionou a ligação entre os requisitos funcionais do documento de requisitos e o levantamento das variáveis de entrada, saída, recursos e controle e seus respectivos tipos de sinais. Outra constatação é a relação entre os eventos dos autômatos finitos e as entradas e saídas no IDEF0 relatado no método proposto. Como também, o uso da TCR e SML possibilitou a geração do correto sequenciamento das atividades da CMR pelo emprego de formalismo matemático, a necessidade de não concorrência entre eventos e caráter modular do sistema.

A verificação de modelos de autômatos e dos supervisórios proporcionou a descoberta e correção de diversos erros ainda na etapa de projeto, o que reforçou os ganhos sugeridos do uso deste método. Neste estudo de caso a verificação de modelos foi responsável pela modificação de diversos modelos de autômatos, como exemplo, o modelo do robô industrial e a alteração conceitual dos subsistemas da CMR estabelecidos pelas especificações locais. Durante a utilização do aplicativo *uppaal* na verificação de modelos com o uso da SML foi observado e corrigido o problema de concorrência entre os eventos do autômato G1, presentes em todos supervisórios locais.

A ferramenta *state diagram* do ambiente *LabVIEW*™ e todas as intervenções necessárias para atendimentos aos conceitos de TCS e SML possibilitou a implementação e simulação em plataforma de uso industrial do sistema em desenvolvimento. Também foi capaz de realizar as integrações com as partes do sistema, como por exemplo, o robô e o sistema de visão. Além disso, a programação gráfica é intuitiva e fácil de ser implementada no ambiente *LabVIEW*™. Porém, apresentou problemas de execução em tempo real. Foi observado que um número grande (maior que 50) de estados em um ou diversos *state diagram* em um único programa geravam anomalias de execução. Ou seja, o sistema em alguns testes executava corretamente e em outros não.

A realização de testes de operação simulados proporcionou a identificação de erros e possíveis melhorias ainda na etapa de projeto. Neste estudo de caso foram realizados testes de operação do sistema supervisório e da interação com os módulos computacionais. O ambiente simulado do programa permitiu o acompanhamento do avanço na execução dos eventos do sistema.

Durante a etapa de projeto, foram identificados erros no atendimento ao documento de requisitos, que levou a realizar alterações, como a modificação do diagrama IDEF0 e modelos de autômatos. Todas as alterações foram realizadas com nenhum impacto a estrutura do projeto e de forma clara e rápida. Como também, os testes de operação na CMR

proporcionaram a averiguação da abstração na interface homem máquina e os resultados mostraram o atendimento ao conceito de programação no nível de tarefas.

Estes resultados mostram que o método proposto pode ser útil em aplicações industriais reais, reduzindo o custo e diminuindo o tempo de desenvolvimento no processo de concepção conforme motivação deste trabalho. Além disso, os resultados demonstram que o método proposto pode ser utilizado em outros sistemas de natureza híbrida, que tenham condições de serem modularizados e cujas atividades sejam sequenciadas.

5 Considerações Finais

Neste capítulo são detalhadas as principais conclusões obtidas e apresentadas algumas propostas para trabalhos futuros.

5.1 Conclusões

Este trabalho introduz um método com a utilização do formalismo único, baseado em autômato, no projeto modular de célula de manufatura robotizada permitindo a abstração da programação no nível de tarefas. Este método permite a sistematização nas transições das diferentes etapas do projeto, proporcionando uma padronização e diminuindo a possibilidade de erros. Também facilita o acompanhamento da evolução do projeto, aumentando a confiabilidade do desenvolvimento, reduz custo e diminui o tempo de desenvolvimento no processo de concepção.

Propõe-se a utilização do conceito de projeto modular, permitindo que o sistema seja dividido em partes menores, ou módulos, cada um dos quais podendo ser desenvolvido, testado e concluído de forma independente, e depois incorporado no sistema. Cada módulo pode ser substituído por outro no sistema, ou utilizado em outros projetos. Além de facilitar a abstração e concepção do sistema supervisório conforme o conceito de SML da TCS.

Possibilita-se a utilização do método em outros tipos de sistemas de natureza híbrida. Por exemplo, em aplicações automotivas, como sistemas de controle de tráfego em rodovias, na eletrônica de potência e controle de processos químicos em geral. Os sistemas híbridos considerados possuem comportamentos dinâmicos de variáveis contínuas e de variáveis discretas interagindo entre si, sendo que o comportamento discreto é definido por eventos gerados quando variáveis do espaço de estados contínuo (ou funções destas) alcançam certos patamares, forçando então transições no estado discreto. O comportamento contínuo, por sua vez, é determinado por uma condição discreta, função do estado discreto atual do sistema. Assim, para cada módulo o comportamento contínuo é encapsulado e o comportamento discreto é representado por um modelo de autômato.

Propõe-se ainda uma abordagem modular para a síntese de supervisores para células de manufatura robotizada, devido a utilização do conceito de projeto modular. Foi utilizado o *Grail* e o *Uppaal* para verificação de modelos o atendimento ao comportamento não bloqueante e minimamente restritivo, conforme desejado. Esta abordagem permite a implementação do sistema em ambiente industrial pelo uso da plataforma *LabVIEW™*, o que seria inviável pelo uso da abordagem monolítica para síntese do supervisório único.

De forma bastante sucinta, pode-se dizer que as principais contribuições deste trabalho são:

- O estabelecimento de um método para sistematização e a utilização de um único formalismo, baseado em autômatos, para o projeto de célula de manufatura robotizada;
- A utilização do conceito de projeto modular na concepção de célula de manufatura robotizada;
- A possibilidade do uso do método apresentado para o projeto de sistemas híbridos e de implementação em ambiente industrial.

5.2 Propostas de Trabalhos Futuros

Os seguintes trabalhos são propostos como atividades futuras, dando continuidade à pesquisa realizada:

1. Procedimento de calibração metrológica do Robô Educacional ROBOT 5150A da *Lab-Volt*;
2. Utilização do *toolbox Statechart* do ambiente *LabVIEW™* para implementação do sistema da CMR e implementação do SML;
3. Desenvolvimento de algoritmo para programação no nível de tarefas do robô educacional ROBOT 5150A da *Lab-Volt*;
4. Implementação de sistema de calibração da interface robô industrial e CAD para geração da programação *off-line*;
5. Desenvolvimento de algoritmo para geração de sequência ótima de movimentação de robô industrial.

6 Referências

ADADE FILHO, A. **Fundamentos de robótica**: cinemática, dinâmica e controle de manipuladores robóticos. 2 ed. São José dos Campos: ITA, 2001. 354 p.

AGUIAR, A. J. C. **Integração de Rede de Petri e Simulação Gráfica para Verificação de Células Robóticas Colaborativa**. 2009. 151 p. Dissertação (Mestrado) – Instituto Tecnológico de Aeronáutica, São José dos Campos, 2009.

ALLA, H.; DAVID, R. ***Continuous and hydrid petri nets***. *In: Journal of Circuits, Systems and Computers*, 1998. V.8, n.1, p. 159-188.

AUGUSTEIJN, M. F.; CLEMENS, L. E. ***A neural-network approach to the detection of texture boundaries, Engineering Applications of Artificial Intelligence***, v. 9, n. 1, p. 75-81, 1996.

BALDWIN, C. Y; CLARK, C. B. ***Design Rules****: The Power of Modularity*. Cambridge: The MIT Press, 2000. 1v.

BARROS, M. R. A. **Estudo da automação de células de manufatura para montagens e soldagem industrial de carrocerias automotivas**. 2006. 133 p. Dissertação (Mestrado) – Universidade de São Paulo, São Paulo, 2006.

BEHRMANN, G.; DAVID, A.; LARSEN, K. G. ***A tutorial on uppaal: formal methods for the design of real-time systems***. *In: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems,* 2004. 200-236 p.

BONNER, S.; SHIN, K. G. ***A Comparative study of robot languages***, *IEEE Computer*. *Upper Saddler River*. NJ.1982. v.18, n.18, 82-96 p.

BOSILJ-VUKSIC, V.; GIAGLIS, M. G.; HLUPIC, V. ***IDEF Diagrams and Petri Nets for Business Process Modeling: Suitability, Efficacy, and Complementary Use***. *In: 2000 Winter Simulation Conference,* 2000

CARROLL, J.; LONG, D. ***Theory of Finite Automata with an Introduction to Formal Languages***. *Englewood Cliffs: Prentice Hall,* 1989.

CARVALHO, J. S.; SOUSA, J. M. **Tutorial de *Uppaal***. Universidade da Beira Interior, 2009.

CASSANDRAS, C. G.; LAFORTUNE, S. ***Introduction to Discrete Event Systems***. 2 ed. EUA: *Springer*, 2008.

CHEN, Y. L.; FORTUNE, S.; LIN, F. ***Modular supervisory control with priorities for discrete event systems.*** *In: 34th IEEE Conference on Decision and Control*, 1995. 409-415 p.

ÇINLAR, E. ***Introduction to Stochastic Processes***. Englewood Cliffs: Prentice Hall, 1975.

COSTA NETO, E. F. et al. **Aplicação do método de visão computacional para a medição bidimensional de furos em fuselagem aeronáutica**. V Congresso Nacional de Engenharia Mecânica. Salvador, 2008.

CLARKE, E. M.; GRUMBERG, O.; PELED, D. A. ***Model Checking***. *1 ed. Cambridge: The MIT Press*, 1999.

CLARKE, E.; EMERSON, E. A.; SISTLA, A. P. ***Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications***. *ACM Transactions on Programming Languages and Systems*, 1986. 8 v. 2 n. 244-263 p.

CRAIG, J. J. ***Introduction to robotics mechanics and control***. 3 ed. *New Jersey: Pearson Prentice hall*, 2005. 406 p.

CURY, J. E. R. **Teoria de Controle Supervisório de Sistemas a Eventos Discretos**. In: V Simpósio Brasileiro de Automação Inteligente, Gramado, 2001.

DAVIES, E. R. ***Machine Vision - Theory Algorithms Practicalities***. 3 ed. *Morgan Kaufmann*, 2005. 935 p.

DUELEN, G.; SCHÖER, K. ***Robot calibration: method and results***. *Robotics & Computer Integrated Manufacturing, Great Britain*, 1991. 8 v. 4 n.

ERBE, H.–H. ***Low Cost Intelligent Automation in Manufacturing***. *In: 15th Triennial IFAC World Congress*, Barcelona, 2002.

ERHARDT-FERRON. ***Theory and Applications of Digital Image Processing***. 1 ed. *University of Applied Sciences Offenburg*, 2000.

EYZELL, J.M.; CURY, J.E.R. ***Exploiting Symmetry in the Synthesis of Supervisors for Discrete Event Systems***. *In: the American Control Conference, Philadelphia*, 1998.

FAULK, S. R. ***Software Requirements****: A Tutorial, in Software Requirements Engineering*, 2 ed. IEEE CS Press, p. 128-149, 1997.

FEDERAL INFORMATION PROCESSING STARDARDS. ***FIPS PUBS 183: Integration definition for function modeling (IDEF0)***. *National Institute of Standards and Technology, Computer Systems Laboratory, Gaithersburg*, 1993.

FREITAS, J. C. A. **Concepção, projeto e implementação de células automatizadas utilizando conceitos de programação off-line de robôs**. 2004. 151 p. Dissertação (Mestrado) – Universidade Estadual de Campinas, Campinas, 2004.

FU, K. S; GONZALEZ, R. C.; LEE, C. S. G. ***Robotics****: Control, Sensing, Vision and Intelligence*. 1 ed. EUA: McGraw-Hill, 1987. 680 p.

GARCIA, E. et al. ***The evolution of robotics research: from industrial robotics to field and service robotics***. *In: IEEE Robotics & Automation Magazine*, 2007. v.14, n.1.

GARCIA, T. R.; CURY, J. E. R. ***Grail*** **para controle supervisório de sistemas a eventos discretos**. 1 ed. Santa Catarina: Universidade Federal de Santa Catarina – UFSC, 2006. 13 p.

GINI, M. ***The future of robot programming***. *Robotica*, Cambridge, v.5, p. 235-246, 1987.

GONZALEZ, R. C.; WOODS, R. E. ***Digital Image Processing***. 3 ed. New Jersey: Pearson Prentice Hall, 2007.

GONZALEZ DEL FOYO, P. M. **Verificação Formal de Sistemas Discretos Distribuídos**. 2009. 160 p. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo. São Paulo, 2009

GROOVER, M. P. ***Automation, Production Systems, and Computer Integrated Manufacturing***. 2 ed. New Jersey: Prentice Hall, 2001.

HERNANDEZ-MATIAS, J.C. et al. ***An Integrated Modelling Framework To Support Manufacturing System Diagnosis For Continuous Improvement***. *Robotics And Computer-Integrated Manufacturing*, 2006.

HOLMBOM, P. et al. ***A model for the execution of task level specifications for intelligent and flexible manufacturing systems***. *In: Proceedings Euromicro 93 Open System Design: Hardware, Software and Applications*, 1993. 38 v. 255-263 p.

INSTITUTE OF ELETRICAL AND ELECTRONICS ENGINEERS. ***IEEE Std 1233: Guide for Developing Systems Requirements Specifications***. *New York*, 1998.

INSTITUTE OF ELETRICAL AND ELECTRONICS ENGINEERS**.** ***IEEE Std 830: Recommended Practice for Software Requirements Specifications***. *New York*, 1998.

JÄHNE, B.; HAUSSECKER, H.; GEISSLER, P. ***Handbook of Computer Vision and Applications - Sensors and Imaging***. 1 ed. San Diego: Academic Press, 1999. 1 v.

JAIN, R.; KASTURI, R.; SCHUNCK, B. G. ***Machine Vision***. 1 ed. [S.l.]: McGraw Hill, 1995.

JUNQUEIRA, F. **Modelagem de Sistemas Flexíveis de Movimentação de Materiais através de Redes de Petri Interpretadas**. Tese de Mestrado, USP, 2001.

KLEINROCK, L. ***Queueing Systems. Volume I: Theory***. 1 ed. [S.l.]: Wiley-Interscience, 1975. p. 448.

KOTONYA, G.; SOMMERVILLE, I. ***Requirements engineering: process and techniques***. *Wiley*, 1998. 294 p.

LEAL, F. et al. **Elaboração de modelos conceituais em simulação computacional através de adaptações na técnica IDEF0: uma aplicação prática**. In: XXVII Encontro Nacional de Engenharia de Produção, Foz do Iguaçu, 2007.

MA, Z.; ZHANG, W.; MA, W. ***Extending Idef1x to Model Fuzzy Data***. *Journal of Intelligent Manufacturing*, 2002. 13 v. 295–307 p.

MARTINS, L. E. G. **Uma metodologia de elicitação de requisitos de software baseada na teoria da atividade**. 2001. 182 p.Tese (Doutorado) – Universidade de Campinas, Campinas, 2001.

MENZEL, C. P.; MAYER, R. J. ***The IDEF family of languages***. 2 ed. Berlin: Springer, 2006.

MORAES, C. C.; CASTRUCCI, P. L. **Engenharia de Automação Industrial**. 2 ed. São Paulo: Livros Técnicos e Científicos (LTC), 2007. 360 p.

MURATA, T. ***Petri Nets: Properties, Analysis and Applications***. *In: IEEE Special Issue on Discrete Event Dynamic Systems*, 1989. v. 77, n. 4, 541-580 p.

NADDEO, P. S. **Uma Taxonomia na Área de Engenharia de Requisitos**, 2002. 92 p. Dissertação (Mestrado) – Instituto de Matemática e Estatística da Universidade de São Paulo. São Paulo, 2002.

NAKAMOTO, F.Y., **Regras de Controle para Alocação de Recursos em Sistemas Produtivos com Processos Concorrentes**, SBAI, 2001.

NAKASHIMA, K.; GUPTA, S. M. ***Performance Evalution of a Supplier Management System with Stochastic Variability***. *Institute Journal Manufacturing Technology and Management*, 2003. 5 v. 1-2 n.

OLIVEIRA, A. D. G. **Sistemas de automação: Robôs**, 2003. 60 p. Trabalho de Conclusão de Curso (Especialização em Mecatrônica) – Universidade São Judas Tadeu, São Paulo, 2003.

ORTH, A. **Desenvolvimento de um Sistema de Visão para medir o Desgaste de Flanco de Ferramentas de Cort**e, 2001. 130 p. Dissertação (mestrado) – Universidade Federal de Santa Catarina, Florianópolis, 2001.

PARACENCIO, L. G. M. **Proposta de metodologias para integração de células de manufatura**. 2009. 155 p. Tese (doutorado) – Universidade Estadual de Campinas, Campinas, 2009.

PAZOS, F. **Automação de sistemas e robótica**. Rio de Janeiro: Axcel Books do Brasil, 2002. 377 p.

PRATT, W. K. ***Digital Image Processing: PIKS Inside***. 4 ed. California: John Wiley and Sons, 2007.

PIRES, J. N. **Robótica das Máquinas Gregas à Moderna Robótica Industrial**. Jornal Público, Coimbra, 1 e 8 de junho de 2002. Caderno de computadores. Disponível em: <*http://robotics. dem.uc.pt/norberto/nova/pdfs/gregosxxi.pdf*>. Acesso em 08 jan. 2012.

QUEIROZ, M. H.; CURY, J. E. R**. Controle Supervisório Modular de Sistemas de Manufatura**. In: Revista Controle & Automação, SBA, 2002a. 13 v. 115-125 p.

QUEIROZ, M. H.; CURY, J. E. R. ***Synthesis and implementation of local modular supervisory control for a manufacturing cell, Discrete Event Systems: Analysis and Control***. In: *Kluwer Academic Publishers*, 2002b. 103-110 p.

RAMADGE, P.J.; WONHAM, W. M. ***The control of discrete event systems***. *In: IEEE Special Issue on Discrete Event Dynamic Systems*, 1989. v. 77, p. 81-98.

RAYMOND, D.; WOOD, D. ***Grail: A C++ library for automata and expressions***. *In: Journal of Symbolic Computation*, 1995. v. 11 p. 341-350.

ROBOTIC INDUSTRIES ASSOCIATION. **ANSI/RIA/ISO 10218-1:** ***Robots for Industrial Environment - Safety Requirements*** *- Part 1 – Robot*. Michingan, USA, 2007.

ROSARIO, J. M. **Princípios de mecatrônica**. 1 ed. São Paulo: Pearson Prentice Hall, 2005. v. 1, p. 360.

ROSARIO, J. M. **Robótica Industrial I: Modelagem, Utilização e Programação**, 1 ed. São Paulo: Editora Baraúna, 2010. p. 494.

RYAN, J.; HEAVEY, C. ***Process modeling for simulation***. *In: Computers in Industry (Elsevier),* 2006. v. 57, p. 437-450,

SANTOS, R. S. B. **Modelagem e análise de performance de sistemas flexíveis de manufatura baseado em redes de petri temporizadas: estudo de caso na indústria automobilística,** 2008. 115 p. Dissertação (Mestrado) – Universidade de São Paulo, São Paulo, 2008.

SILVA, A. M. **Aplicação de Verificação de Modelos a Programas de CLP: Explorando a Temporização**, 2008. 121 p. Dissertação (mestrado) - Instituto Militar de Engenharia, Rio de Janeiro, 2008.

STEMMER, M. R. et al. **Apostila de Sistemas de Visão**. Universidade Federal de Santa Catarina. Florianópolis, 2005.

TCHIJOV, I. CIM ***Introduction: Some Socioeconomic Aspects***. *Technological Forecasting and Social Change*, 1989. n. 35, p. 261-275.

VAZ, A. F.; WONHAM, W. M. ***On supervisor reduction in discrete-event systems***. *In: International Journal of Control*, 1986. v. 44, n. 2, p. 475-491.

WONG, K.C. et al. ***Conflict resolution in modular control with application to feature interaction***. *In: 34th IEEE Conference on Decision and Control*, 1995. p. 416-421.

WONG, K.C.; WONHAM, W.M. ***Modular Control and Coordination of Discrete-Event Systems***. *In: Discrete Event Dynamic Systems*, 1998. v. 8, n. 3, p. 241-273.

WORLD ROBOTICS. ***Executive Summary***, 2012.

ZACHMAN, J. A. ***A framework for information systems architecture***. *In: IBM Systems Journal*, 1987. v. 26. n. 3, p. 276-292.

Apêndice A – Cinemática do Robô Industrial

A cinemática é a ciência do movimento que o trata sem atentar para as forças que o ocasionam. O objetivo da cinemática em robôs industriais é apresentar uma solução para o problema da determinação da postura do órgão terminal, ou a determinação dos ângulos de cada junta (CRAIG, 2005). A Figura 50 apresenta a problemática da cinemática.

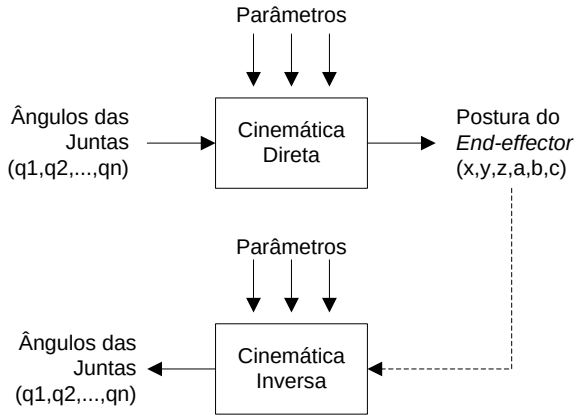


Figura 50. Problemática da cinemática (FU; GONZALEZ; LEE, 1987).

O robô industrial é basicamente uma série consecutiva de corpos rígidos (elos) ligados por junta, cada elo possui um *frame*. O movimento do órgão terminal (ou *end-effector*) do robô industrial é causado pelos movimentos dos conjuntos de elos localizados entre a base e o próprio órgão terminal, este conjunto, também é chamado de cadeia cinemática (FU; GONZALEZ; LEE, 1987).

A notação *Denavit-Hartenberg* (D-H) é um método sistemático para descrever a cadeia cinemática do robô industrial, utiliza um número mínimo dos parâmetros para cada par de juntas (rotativas ou prismáticas) sucessivas e o relacionamento cinemático entre um par de elos adjacentes. Esta notação só deve ser utilizada no manipulador serial (cadeia aberta) que possui juntas com um único tipo de movimento e quando os elos forem corpos rígidos (ADADE FILHO, 2001). A Figura 51 representa os parâmetros D-H e o algoritmo é apresentado a seguir (FU; GONZALEZ; LEE, 1987) (CRAIG, 2005):

1. Enumerar as juntas de 1 até *n* a partir da base até o *end-effector* (j_1 até j_n). Sendo *n* o número de GDL;

2. Estabelecer o *frame* da base (x_0, y_0, z_0). Definir o eixo z_o coincidente ao eixo de movimento (rotação ou deslizante) da junta j_1, colocar a origem O_0 ao longo do eixo z_0. Definir x_0 de arbitrariamente como normal z_0. Estabelecer y_0 pela regra da mão direita[6].
3. Determinar o *frame* x_i, y_i, z_i para cada elo $i = 1, \ldots, n$. Definir o eixo z_i coincidente ao eixo de movimento (rotação ou deslizante) da junta j_{i+1}, colocar a origem O_i na intersecção do eixo z_{i-1} e z_i, ou na interseção da normal comum[7] entre os eixos z_{i-1} e z_i no eixo z_i. Definir o eixo x_i ao longo da normal comum entre os eixos z_{i-1} e z_i na origem O_i com sentido da junta j_i para junta j_{i+1}. Se os eixos z_{i-1} e z_i se cruzam, então o eixo x_i é a normal comum. Estabelecer y_i pela regra da mão. O *frame* do elo *i* está na junta j_{i+1}, portanto, ao final do elo i. Repetir até o frame do órgão terminal (ou *end-effector*) (x_n, y_n, z_n);
4. Determinar os quatro parâmetros de D-H:
    a. θ_i (ângulo da junta): O ângulo de rotação partindo do eixo x_{i-1} até o eixo x_i em torno do eixo z_{i-1} (utilizar a regra da mão direita);
    b. d_i (distância entre juntas): A distância partindo da origem O_{i-1} até a interseção entre o eixo z_{i-1} com o eixo x_i ao longo do eixo z_{i-1};
    c. a_i (distância - *off-set* - entre elos): A distância partindo da interseção do eixo z_{i-1} com o eixo x_i até a origem O_i ao longo do eixo x_i (ou a menor distância entre os eixos z_{i-1} e z_i);
    d. α_i (ângulo de torção entre elos): O ângulo de rotação partindo do eixo z_{i-1} até o eixo z_i em torno do eixo x_i (utilizar a regra da mão direita);

[6] Regra da mão direita é a representação do sistema de coordenadas relacionando os dedos da mão direita, o qual o dedo indicador representa o eixo z, o dedão o eixo y e a palma da mão o eixo x.

[7] Normal comum entre duas linhas é a linha que determina a menor distância entre estas duas linhas (CRAIG, 2005).

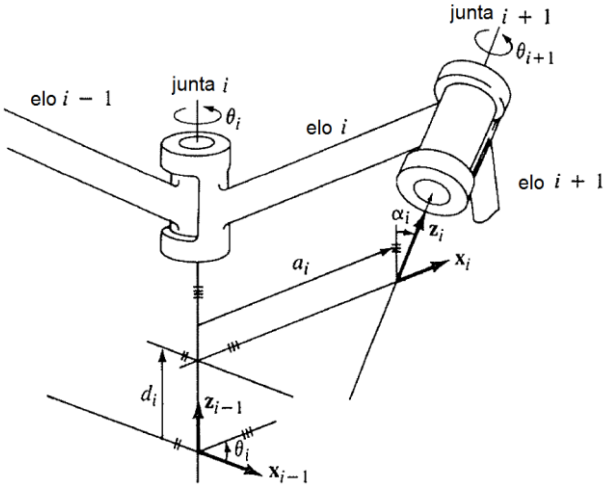
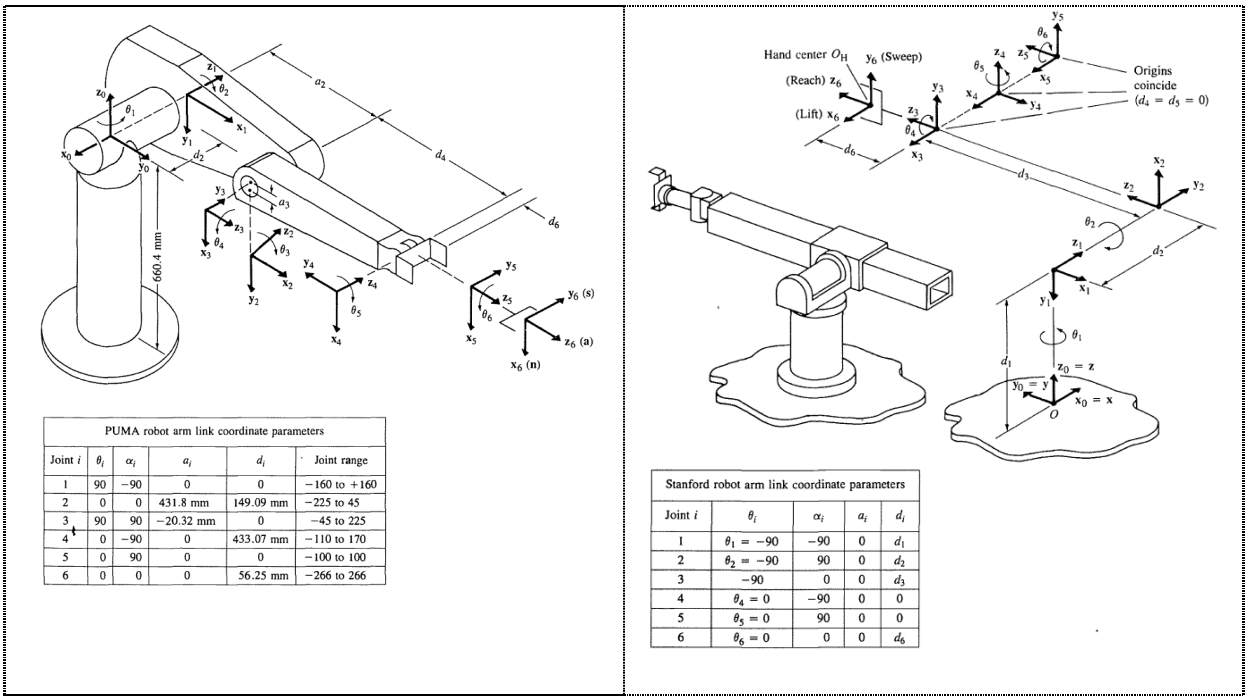


Figura 51. Sistema de coordenadas dos elos e seus parâmetros D-H (FU; GONZALEZ; LEE, 1987).

Para uma junta de rotação, os parâmetros d_i, a_i e α_i são constantes e o θ_i é variável. O parâmetro θ_i representa o valor na rotação da junta entre o elo i e elo $i-1$. Para uma junta deslizante, os parâmetros θ_i , a_i e α_i são constantes e o d_i, é variável. O parâmetro d_i representa o valor do deslocamento linear da junta. A Figura 52 apresenta o levantamento dos parâmetros de D-H do robô PUMA 560 e o robô STANFORD.



PUMA robot arm link coordinate parameters

Joint i	θ_i	α_i	a_i	d_i	Joint range
1	90	−90	0	0	−160 to +160
2	0	0	431.8 mm	149.09 mm	−225 to 45
3	90	90	−20.32 mm	0	−45 to 225
4	0	−90	0	433.07 mm	−110 to 170
5	0	90	0	0	−100 to 100
6	0	0	0	56.25 mm	−266 to 266

Stanford robot arm link coordinate parameters

Joint i	θ_i	α_i	a_i	d_i
1	$\theta_1 = -90$	−90	0	d_1
2	$\theta_2 = -90$	90	0	d_2
3	−90	0	0	d_3
4	$\theta_4 = 0$	−90	0	0
5	$\theta_5 = 0$	90	0	0
6	$\theta_6 = 0$	0	0	d_6

Figura 52. Exemplos de levantamento de parâmetros de D-H (FU; GONZALEZ; LEE, 1987).

Uma vez que os sistemas de coordenadas D-H tenham sido estabelecidos, uma matriz de transformação homogênea composta pode ser criada relacionando i-ésimo ao (i-1)-

ésimo *frame* aplicando as transformações sucessivamente apresentadas a seguir (ROSARIO, 2010):

1. Rotação no eixo z_{i-1} de um ângulo θ_i para alinhar o eixo x_{i-1} com o eixo x_i (o eixo x_{i-1} é paralelo ao eixo x_i e aponta para o mesmo sentido);
2. Translação uma distância de d_i ao longo do eixo z_{i-1} para colocar os eixos x_{i-1} e x_i na coincidência;
3. Translação ao longo do eixo x_i uma distância de a_i para trazer as duas origens e o eixo x_i a coincidência;
4. Rotação do eixo x_i um ângulo de α_i para trazer os O_i e O_{i-1} a coincidência.

O produto matricial das quatro transformações (matrizes homogêneas elementares) produz uma matriz de transformação homogênea composta A_i^{i-1} conhecida como matriz de transformação de D-H para *frame* adjacentes, i e $i-1$ (referência):

Para junta de rotação, obtém a equação (4):

$$A_i^{i-1} = ROT(z_{i-1}, \theta_i)TRANS(z_{i-1}, d_i)TRANS(x_i, a_i)ROT(x_i, \alpha_i) \tag{4}$$

$$A_i^{i-1} = \begin{vmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{5}$$

$$A_i^{i-1} = \begin{vmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{6}$$

Para junta deslizante, obtém a equação (7):

$$A_i^{i-1} = ROT(z_{i-1}, \theta_i)TRANS(z_{i-1}, d_i)ROT(x_i, \alpha_i) \tag{7}$$

$$A_i^{i-1} = \begin{vmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & 0 \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{8}$$

A descrição cinemática completa de uma cadeia articulada pode ser obtida a partir do produto matricial entre as diversas matrizes de transformações homogêneas compostas. Usando a matriz A_i^{i-1}, podemos estabelecer a relação entre o *frame* da *end-effector* e o *frame* da base para um robô industrial de n. GDL conforme apresentado na equação (9):

$$T_n^o = A_1^0 A_2^1 \dots A_{n-1}^{n-2} A_n^{n-1} \tag{9}$$

Simplificando A_n^{n-1} por A_n obtém a equação (10):

$$T_n^o = A_1 A_2 \dots A_{n-1} A_n \tag{10}$$

A matriz de transformação homogênea pode ser representada pela equação (11):

$$T_n^o = \begin{vmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} R_n^0 & P_n^0 \\ 0 & 1 \end{vmatrix} \tag{11}$$

A matriz T_n^o estabelece a matriz de orientação R_n^0 e o vetor posição P_n^0. A matriz de rotação descreve os ângulos de rotação dos eixos do *frame* do *end-effector* em relação ao *frame* da base (referência) do robô industrial. Nas aplicações industriais são utilizados os Ângulos de *Euler* para descrição da orientação de um corpo rígido no espaço (ROSARIO, 2010).

Os Ângulos de *Euler* consistem em três ângulos de rotação, que levam o objeto de sua orientação original de referência até a orientação desejada. As rotações são efetuadas consecutivamente no sistema de coordenadas (*frame*) do corpo, que assim vai assumindo uma nova situação a cada rotação (ADADE FILHO, 2001). A seguir são apresentados os três conjuntos de Ângulos de *Euler* possíveis, conforme sequência de rotação (FU; GONZALEZ; LEE, 1987) (ADADE FILHO, 2001) (CRAIG, 2005).

Ângulos de *Euler* ZXZ: Inicia-se com o *frame* do órgão terminal (ou *end-effector)* {N}, coincidente com o *frame* da base {0}. Efetuam-se então as rotações, em torno dos eixos do sistema móvel {N}, conforme Equação (12):

$$R^0_{N\,ZXZ} = ROT(z_n, \phi)ROT(x_n\,, \theta)ROT(z_n, \psi) \tag{12}$$

$$R^0_{N\,ZXZ} = \begin{vmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{13}$$

$$R^0_{N\,ZXZ} = \begin{vmatrix} \cos\phi\cos\psi - \sin\phi\cos\theta\sin\psi & -\cos\phi\sin\psi - \sin\phi\cos\theta\cos\psi & +\sin\phi\sin\theta \\ \sin\phi\cos\psi + \cos\phi\cos\theta\sin\psi & -\sin\phi\sin\psi + \cos\phi\cos\theta\cos\psi & -\cos\phi\sin\theta \\ \sin\theta\sin\psi & \sin\theta\cos\psi & \cos\theta \end{vmatrix} \tag{14}$$

Solução do problema inverso pode ser obtida como:

$$\phi = atan2\big(a_x, -a_y\big)^{8} \tag{15}$$

$$\theta = atan2\Big(\sqrt{({n_z}^2 + {s_z}^2)}, a_z\Big) = atan2\Big(\big(-a_y\cos\phi + a_x\sin\phi\big), a_z\Big) = atan2\Big(\big(n_y\cos\psi - s_y\sin\psi\big), (n_x\cos\psi - s_x\sin\psi)\Big) \tag{16}$$

$$\psi = atan2(n_z, s_z) \tag{17}$$

Ângulos de *Euler* ZYX (*roll*, *pitch* e *yaw*): Inicia-se com o *frame* do órgão terminal (ou *end-effector)* {N}, coincidente com o *frame* da base {0}. Efetuam-se então as rotações, em torno dos eixos do sistema móvel {N}, conforme Equação (18):

$$R^0_{N\,ZYX} = ROT(z_N, \alpha)ROT(y_N, \beta)ROT(x_N, \gamma) \tag{18}$$

$$R^0_{N\,ZYX} = \begin{vmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{vmatrix} \tag{19}$$

[8] $atan2$ é a função que calcula a tangente inversa de duas variáveis (argumentos), porém os sinais de ambos os argumentos são usados para determinar o quadrante do resultado. Retorna o resultado em radianos entre $-\pi$ e π (inclusive).

$$R^0_{N\,ZYX} = \begin{vmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{vmatrix} \quad (20)$$

Solução do problema inverso pode ser obtida como:

$$\alpha = atan2(n_y, n_x) \quad (21)$$

$$\beta = atan2\left(-n_z, \sqrt{({n_x}^2 + {n_y}^2)}\right) = atan2\left(-n_z, (n_x\cos\alpha + n_y\sin\alpha)\right) \quad (22)$$

$$\gamma = atan2(s_z, a_z) = atan2\left((a_x\sin\alpha - a_y\cos\alpha), (-s_x\sin\alpha + s_y\cos\alpha)\right) \quad (23)$$

Ângulos de *Euler* ZYZ: Inicia-se com o *frame* do órgão terminal (ou *end-effector)* {N}, coincidente com o *frame* da base {0}. Efetuam-se então as rotações, em torno dos eixos do sistema móvel {N}, conforme Equação (24):

$$R^0_{N\,ZYZ} = ROT(z_N, \alpha)ROT(y_N, \beta)ROT(z_N, \gamma) \quad (24)$$

$$R^0_{N\,ZYZ} = \begin{vmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{vmatrix} \begin{vmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad (25)$$

$$R^0_{N\,ZYZ} = \begin{vmatrix} +\cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma & -\cos\alpha\cos\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta \\ +\sin\alpha\cos\beta\cos\gamma + \cos\alpha\sin\gamma & -\sin\alpha\cos\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta \\ -\sin\beta\cos\gamma & \sin\beta\sin\gamma & \cos\beta \end{vmatrix} \quad (26)$$

Solução do problema inverso pode ser obtida como:

$$\alpha = atan2(a_y, a_x) \quad (27)$$

$$\beta = atan2\left(\sqrt{({n_z}^2 + {s_z}^2)}, a_z\right) \quad (28)$$

$$\gamma = atan2(s_z, -n_z) \quad (29)$$

A Matriz de transformada Homogênea T_n^o é uma solução para a problemática da cinemática direta apresentada na Figura 50. Já a cinemática inversa para existir solução deverá atender aos critérios de (ADADE FILHO, 2001): Existência: A postura desejada deve situar-se dentro do espaço de trabalho do robô industrial; Multiplicidade: Para a existência de mais de um conjunto de variáveis de juntas que proporcionam a mesma postura do órgão terminal, deve-se escolher a que possibilite o menor movimento nas juntas, minimiza a energia cinética do movimento e evite colisões; Método de solução: Requer resolver um conjunto de equações não lineares, para os quais inexistem métodos de soluções gerais, cada sistema de equações deve ser considerado um caso particular.

A literatura (FU; GONZALEZ; LEE, 1987) (ADADE FILHO, 2001) (CRAIG, 2005) (ROSARIO, 2010) sugere dois métodos de soluções: **Método de cálculos iterativo**: baseado em soluções numéricas iterativas, geralmente consomem muito tempo, sendo impraticável para aplicações de tempo real; **Método Analítico**: baseado em soluções geométricas, procura-se decompor a geometria espacial do mecanismo em vários problemas de geometria plana, utilizando-se então de seus recursos, determinam-se os ângulos das juntas, ou baseado em soluções algébricas, procura-se criar um sistema de equações matriciais não lineares, utilizando a Transformada Homogênea T_n^o e procedimento a seguir:

1. Identificar o número de GDL do robô industrial. Para o exemplo, assumir-se cinco GDL;
2. Calcular a Matriz Homogênea utilizando a Notação D-H. Para o exemplo, temos a equação (30);

$$T_5^0 = A_1 A_2 A_3 A_4 A_5 \tag{30}$$

3. Realizar as multiplicações sucessivas da inversa da matriz $A_i^{i-1}, i = 1\ até\ n$. Para o exemplo, obtêm-se quatro equações matriciais, representadas pelas equações (31) até (34);

$$T_5^1 = (A_1)^{-1} T_5^0 \tag{31}$$

$$T_5^2 = (A_2)^{-1}(A_1)^{-1} T_5^0 \tag{32}$$

$$T_5^3 = (A_3)^{-1}(A_2)^{-1}(A_1)^{-1} T_5^0 \tag{33}$$

$$T_5^4 = (A_4)^{-1}(A_3)^{-1}(A_2)^{-1}(A_1)^{-1}T_5^0 \quad (34)$$

4. Inspecionar, na sequência, cada conjunto de equações, elemento por elemento, para encontrar a equação que seja resolvida facilmente para se obter a variável de junta anterior. Achando-a, usá-la para achar equações que determinem a variável de junta posterior, de um elo que seria o "primeiro" da cadeia;
5. Prosseguir passo a passo (de (31) a (34)) até que todas as variáveis de juntas sejam determinadas.

Apêndice B – Componentes do Sistema de Visão Computacional

Os componentes do sistema de visão computacional são apresentados a seguir.

C.1 Câmera

A câmera é um dispositivo que converte energia eletromagnética luminosa refletida de um objeto físico em um sinal elétrico condicionado. É composta por duas partes básicas (COSTA NETO et al, 2008):

1. **Sensor**: É uma matriz de elementos foto sensíveis, que recebem e mapeiam o nível de intensidade luminosa, transformando em sinal elétrico analógico proporcional (JÄHNE; HAUSSECKER; GEISSLER, 1999). São classificados de acordo com o tipo de varredura realizado e conforme sua sensibilidade, podendo ser: *VIdeo Cameras using tubes* (VIDICON), C*harge-Coupled Device* (CCD) e *Complementary Metal Oxide Semicondutor* (CMOS). O sinal analógico é digitalizado, o que o torna caracterizado pela robustez, altas taxas de transferência de dados, imunidade a ruídos eletromagnéticos, suporte a aquisição de imagens com alta resolução (*megapixel*) e maiores níveis de contraste de cores (sendo câmeras coloridas) ou tons de cinza (sendo câmeras monocromáticas).
2. **Sistema Óptico**: Regula os feixes de luz refletidos pelo objeto físico para uma intensidade adequada para sensibilizar o sensor de maneira que a imagem capturada fique nítida.

C.1.1 Parâmetros da Câmera

Os parâmetros da câmera, ou parâmetros fundamentais, são as informações necessárias para a correta especificação do sistema de visão computacional. Sendo (JAIN; KASTURI; SCHUNCK, 1995) (STEMMER et al, 2005):

1. **Campo de Visão (*Field of Vision* – FOV)**: representa a área visível do objeto em estudo que incide sobre o sensor, ou seja, a porção do objeto que preenche e sensibiliza a área do sensor.
2. **Distância de Trabalho (*Working Distance* –WD)**: representa a distância da parte frontal das lentes até a superfície do objeto. Trata-se normalmente de uma faixa de valores (máximo e mínimo).

3. **Resolução (*Resolution* – Re)**: representa a menor porção do objeto em estudo que pode ser distinguida pelo sistema. É normalmente visualizada em pares de linha, ou em número de pixels, e também é bem conhecida pela expressão “resolução espacial”.
4. **Profundidade de Campo (*Depth of Field* – DOF)**: representa a maior distância (em termos de profundidade no campo de visão) que pode ser mantida em foco no objeto em estudo para uma determinada distância de trabalho. Também pode ser vista como a quantidade de movimento permitida ao objeto que ainda conserve foco na área ou superfície inspecionada.
5. **Tamanho do Sensor (*Sensor Size* – SS)**: representa o tamanho da área ativa do sensor, especificada em sua dimensão horizontal. Tamanhos comuns são de 1/4, 1/3, 1/2, 2/3 e 1 polegada.

Figura 53. Parâmetros fundamentais para definição de um sistema óptico (STEMMER et al, 2005).

C.2 Sistema de Iluminação

O sistema de iluminação é responsável pela iluminação homogênea e constante ao longo do tempo que incide na superfície do objeto de interesse. Este sistema realça características superficiais do objeto, tais como geometria, estrutura, cor, transparência e refletância, viabilizando a etapa de processamento de imagem (JÄHNE; HAUSSECKER; GEISSLER, 1999). (ERHARDT-FERRON, 2000). As principais técnicas de iluminação usadas em sistema de visão computacional são (JÄHNE; HAUSSECKER; GEISSLER, 1999) (ORTH, 2001) (DAVIES, 2005) (STEMMER et al, 2005) (PRATT, 2007):

1. **Direcional**: consiste na emissão de um feixe luminoso (unilateral ou bilateral) em direção fixa e com pouca dispersão, o qual reflete sobre o objeto realçando características especiais deste. Vantagens: forte, realçando características superficiais do objeto; Desvantagens: pode gerar sombras ou

regiões de muito brilho; Fontes de Iluminação: fibras ópticas, LEDs, lâmpadas incandescentes.

2. **Multidirecional**: consiste na emissão de diversos feixes luminosos provindos de diferentes e equidistantes pontos em relação ao centro do objeto, gerando uma iluminação mais uniforme de toda sua superfície. Vantagens: forte e uniforme, reduzindo sombras; Desvantagens: podem gerar regiões de muito brilho, custos mais elevados; Fontes de Iluminação: fibras ópticas, LEDs.
3. **Anel**: tem o formato de um anel luminoso, que é fixado junto à lente da câmera, fornecendo iluminação paralela ao eixo óptico. Vantagens: forte e uniforme, reduzindo sombras; Desvantagens: pode gerar um anel de muito brilho sobre superfícies reflexivas; Fontes de Iluminação: fibras ópticas, LEDs, lâmpadas fluorescentes.
4. **Campo Escuro**: tipo especial de iluminação direcional, onde a luz é emitida em diversas direções, porém, sempre perpendicular ao eixo óptico. Torna-se muito útil para inspeção de objetos com superfícies em alto relevo ou irregulares. Vantagens: propicia ótimo contraste para superfícies transparentes ou em alto relevo; Desvantagens: gera pouco contraste em objetos opacos, e sua instalação é mais complicada, devido à proximidade dos objetos; Fontes de Iluminação: fibras ópticas, LEDs, lasers.
5. **Luz de Fundo**: consiste em posicionar a fonte luminosa atrás do objeto, emitindo luz na direção do eixo óptico. Útil para aplicações em metrologia e análise de objetos translúcidos. Vantagens: propicia ótimo contraste para detecção de bordas e análise de objetos translúcidos; Desvantagens: elimina os detalhes superficiais do objeto, por não serem iluminados; Fontes de Iluminação: fibras ópticas, LEDs, lâmpadas fluorescentes.
6. **Cúpula**: consiste em emitir feixes luminosos provenientes da base de uma cúpula semiesférica, que se propagam e refletem em diferentes pontos ao longo da cúpula, concêntrica com o objeto. Provê iluminação uniforme e difusa sobre toda a superfície do objeto. Vantagens: reduz sombras e inibe áreas de muito brilho em superfícies reflexivas. Desvantagens: instalação é mais complicada, devido à proximidade do objeto, custos mais elevados; Fontes de Iluminação: fibras ópticas, LEDs, lâmpadas fluorescentes.

7. **Difusa**: consiste em projetar iluminação perpendicularmente ao eixo óptico através de um difusor. A luz dispersa que passa pelo difusor atinge um divisor de feixes (*beamsplitter*) e é refletida para a superfície do objeto. Os feixes que provêm dos objetos paralelos ao eixo óptico atravessam o divisor de feixes e sensibilizam o sensor. Vantagens: reduz sombras e brilho, apresenta iluminação uniforme das superfícies planas; Desvantagens: limitada a pequenas distâncias de trabalho, produz pouca intensidade luminosa, além de ser de difícil instalação e ter custos mais elevados; Fontes de Iluminação: fibras ópticas, LEDs, com difusores.
8. **Dia Nublado**: semelhante à iluminação em cúpula, esta técnica também emite feixes luminosos provenientes da base da cúpula e ainda por uma fonte superior através de um divisor de feixes. Os raios luminosos propagam-se e refletem em diferentes pontos ao longo desta, produzindo um efeito luminoso semelhante ao comportamento de um dia nublado, com iluminação uniforme em todas as direções que incidem sobre o objeto. Vantagens: propicia iluminação uniforme, reduz sombras e inibe áreas de muito brilho em superfícies complexas e reflexivas; Desvantagens: instalação é mais complicada, devido à proximidade do objeto, custos mais elevados; Fontes de Iluminação: fibras ópticas, LEDs, lâmpadas fluorescentes.
9. **Estruturada**: produz uma linha luminosa sobre a superfície do objeto, possibilitando uma análise das características geométricas da superfície do objeto ao movimentar esta linha sobre o mesmo. Muito usada em aplicações em 3D. Vantagens: destaca informações de relevo e geometria das superfícies; Desvantagens: requer fontes potentes de iluminação, de difícil calibração e complexo sistema de movimentação do feixe luminoso; Fontes de Iluminação: fibras ópticas, LEDs, lasers.

A Figura 54 apresenta o desenho esquemático de montagem de cada técnica de iluminação citada.

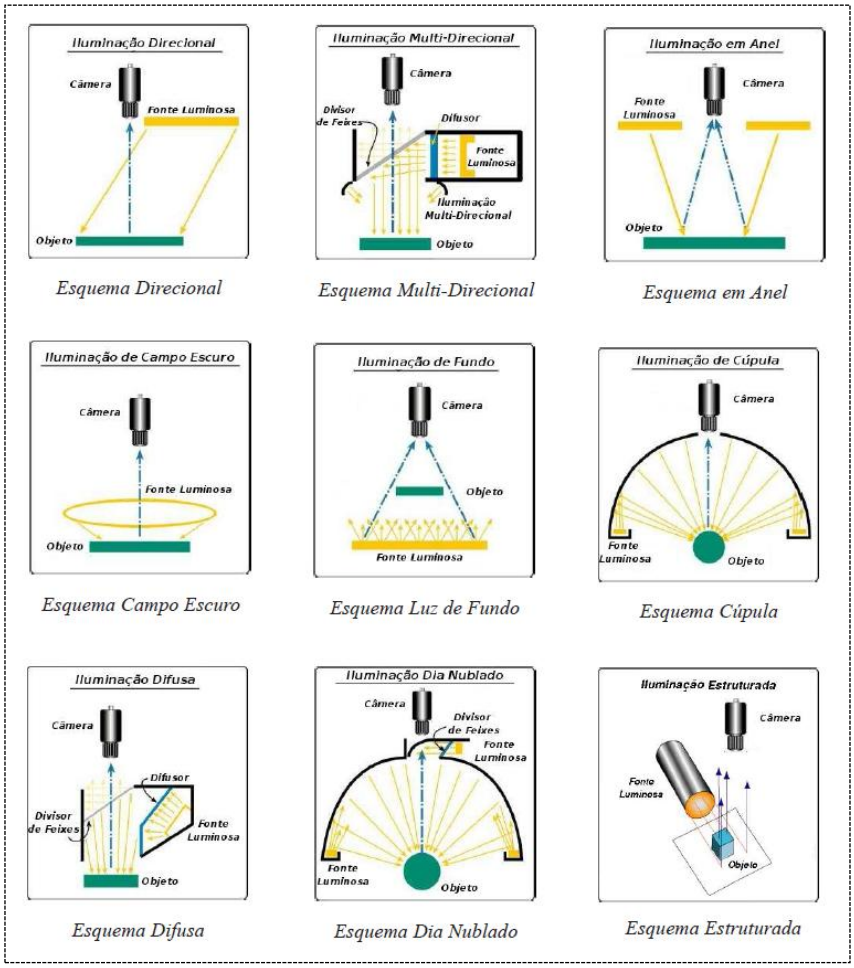


Figura 54. Desenho esquemático de cada técnica de iluminação. (STEMMER et al, 2005)

C.3 Unidade de Processamento

A unidade de processamento realiza a aquisição, tratamento e processamento da imagem para o processo da automação flexível. O *hardware* é normalmente um computador pessoal que possui entrada USB e um *software* é interface computacional capaz de realizar a aquisição, tratamento e o processo desejado. Neste trabalho será abordado a plataforma *LabVIEW*™.

C3.1 Aquisição de Imagem

A aquisição de imagem consiste em utilizar uma câmera para capturar uma imagem real e transformar em uma imagem digital constituída por uma tabela de valores discretos inteiros (*pixels*) para ser processada por um computador (PRATT, 2007). O algoritmo na plataforma *LabVIEW*™ para a aquisição de imagem é apresentado na Figura 55 e descrito a seguir:

1. Ligar o sistema de aquisição de imagem;
2. Criar temporariamente espaço em memória para alocação;

3. Abrir a seção de aquisição de imagem via USB;
4. Inicializar laço de aquisição continuada de imagem via USB, realizando a aquisição quando o laço for finalizado;
5. Fechar a seção de aquisição de imagem via USB;
6. Salvar a imagem no formato .jpg.

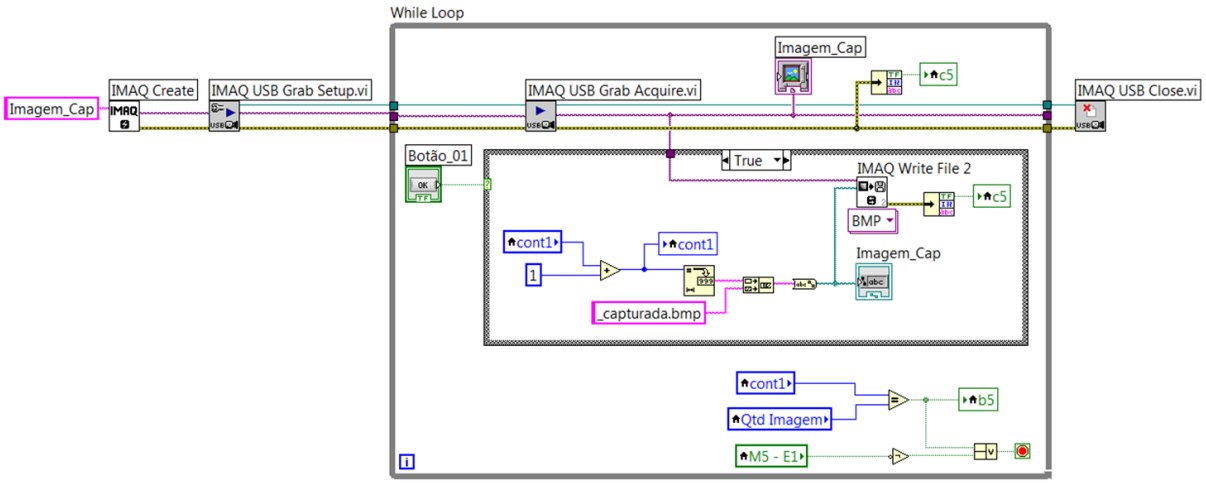


Figura 55. Diagrama de bloco de aquisição de imagem.

C3.2 Tratamento de Imagem

O tratamento de imagem consiste em processar a imagem digital para realçar características desejadas, ou corrigir defeitos e imperfeições geradas durante a aquisição da imagem causada por características físicas do sistema, condições deficientes de iluminação, baixa resolução da câmera e outros (STEMMER et al, 2005) (PRATT, 2007). Entre as operações básicas, baseando-se em operações de processamento pontual, as mais comumente empregadas são (ERHARDT-FERRON, 2000) (STEMMER et al, 2005):

1. **Identidade**: Obtêm uma cópia idêntica, ponto a ponto da imagem original.
2. **Negativa**: Invertem a intensidade luminosa de todos os pontos da imagem, resultando em uma nova imagem. Esta operação é adequada para destacar pequenos detalhes brancos ou cinza sobre grandes áreas negras.
3. **Logarítmica**: Aplicam uma transformação logarítmica à cada ponto de uma imagem. Esta operação mapeia uma faixa de baixos valores de intensidade luminosa para faixas maiores, tornando visíveis partes da imagem que se encontravam muito escuras.
4. **Potência**: aplicam uma transformação de potência à cada ponto de uma imagem. Funciona de forma semelhante à logarítmica, realçando pontos de baixa intensidade luminosa.

As operações de filtragem são utilizadas para realçar características específicas das imagens. Dividem-se entre as operações de suavização (*smoothing*), para atenuar o nível de ruído nas imagens, removendo pequenos detalhes e suavizando contornos, e enfatização (*sharpening*), para destacar detalhes da imagem, principalmente cantos e arestas. As principais operações de filtragem são (JAIN; KASTURI; SCHUNCK, 1995) (STEMMER et al, 2005) (PRATT, 2007):

1. **Suavização Linear**: gerar novas imagens, por meio de cálculo do novo valor de intensidade de um dado ponto a partir de uma média linear dos valores de intensidade dos pontos vizinhos.
2. **Estatísticos Não Lineares**: gerar novas imagens com supressão de modelos de ruídos comportados, a partir da análise estatística do valor de intensidade luminosa de seus pontos vizinhos, que são ordenados conforme uma regra específica para posteriormente substituir o valor de intensidade do ponto central de acordo com a classificação da ordenação.
3. **Gradiente**: Baseia na aplicação de derivadas a toda a área da imagem. Desta forma, a derivação dos sinais de intensidade luminosa da imagem resulta no realce das regiões de transição abrupta desta intensidade, ou seja, um destaque especial para os cantos e bordas de objetos na imagem, eliminando muito da informação constante na imagem.
4. **Laplaciano**: similar aos filtros gradientes, estes também se baseiam na aplicação de derivadas às imagens para realçar cantos e arestas de objetos. Porém estes filtros aplicam a segunda derivada sobre as imagens, ao invés da primeira.

As operações estatísticas são utilizadas para modificar a imagem. As principais são (ERHARDT-FERRON, 2000) (STEMMER et al, 2005) (PRATT, 2007):

1. **Histograma**: É a representação gráfica de uma imagem. O eixo horizontal representa todos os níveis de intensidade luminosa possíveis do pixel e o vertical o número de pixel para cada nível de intensidade luminosa.
2. **Equalização**: É uma redistribuição dos níveis de intensidade luminosa de cada pixel para atingir uma maior uniformidade de uma, ou varias faixas no histograma da imagem.

As operações de morfologia matemática são utilizadas para alterar características da imagem utilizando conceitos de conjuntos e operadores lógicos. Segue uma descrição das operações (GONZALEZ; WOODS, 2002) (STEMMER et al, 2005):

1. **Dilatação**: é a aplicação de um elemento estruturante de forma concêntrica sobre um conjunto definido de pontos (brancos ou pretos) em uma imagem, de maneira que o elemento estruturante adicione informação sobre a vizinhança destes pontos. Esta operação é utilizada principalmente para preencher intervalos e lacunas indesejáveis na imagem.
2. **Erosão**: É o inverso da dilatação. A aplicação do elemento retira informação (gerando erosão nas áreas percorridas). Esta operação é utilizada principalmente para eliminar detalhes irrelevantes, como ruídos, e abrir intervalos ou lacunas em regiões de conexão indesejada.
3. **Abertura**: O operador de abertura aplica uma erosão seguida de uma de dilatação na imagem. Esta sequência de operações visa eliminar pequenos ruídos na imagem e abrir lacunas em regiões de fraca conexão entre objetos, através da erosão, e posteriormente tenta restaurar as dimensões reais de objetos da imagem através da dilatação. Os ruídos e fracas conexões eliminados com a erosão não retornam à imagem após a dilatação.
4. **Fechamento**: O operador aplica uma dilatação seguida de uma erosão. Esta sequência de operações visa restaurar conexões fracas entre objetos da imagem.

As técnicas de segmentação tem o objetivo de dividir a imagem utilizam conceitos de morfologia matemática, porém com maior tempo de processamento. São baseados em duas propriedades: a descontinuidade, que divide a imagem em regiões de acordo com as mudanças de nível de intensidade luminosa e a similaridade, que divide a imagem em regiões de acordo com algum padrão. Segue a descrição das principais operações desta técnica (GONZALEZ; WOODS, 2007) (STEMMER et al, 2005) (PRATT, 2007):

1. **Detecção de descontinuidades**: Procura regiões de transição abrupta do nível de intensidade luminosa dos pontos da imagem para realizar as divisões. Dentre as descontinuidades mais comuns encontram-se pontos, linhas e bordas. Para a detecção de pontos e linhas utilizam-se os operadores de *laplace* e para a detecção de bordas utilizam-se os algoritmos de *Roberts, Prewitt e Sobel* com operadores gradientes.

2. **Detecção de limiares (*thresholding*)**: Procura agrupar os diferentes objetos e regiões da imagem conforme a similaridade de nível de intensidade luminosa, em seguida ocorre a operação de limiarização simples, no qual um valor de nível de intensidade luminosa é escolhido como ponto de corte para a binarização da imagem.

A Figura 56 apresenta a implementação para o tratamento de imagem na plataforma *LabVIEW*™ e a Figura 57 na ferramenta *NI Vision Assistent*. Este algoritmo é descrito a seguir:

1. Ler o arquivo de imagem em formato .jpg;
2. Aplicar o algoritmo de filtro denominado suavização linear;
3. Aplicar o algoritmo de morfologia matemática denominada abertura;
4. Aplicar o algoritmo de *Thresholding* para binarização da imagem, os experimentos mostraram que os intervalos entre 75 e 255 deveriam ser ajustados para 255 (branco) e o restante para 0 (preto);
5. Aplicar o algoritmo de equalização;
6. Aplicar o algoritmo de morfologia matemática denominada remoção de pequenos objetos;
7. Aplicar o algoritmo de filtro denominado Laplaciano;
8. Transformar a imagem em uma matriz de duas dimensões.

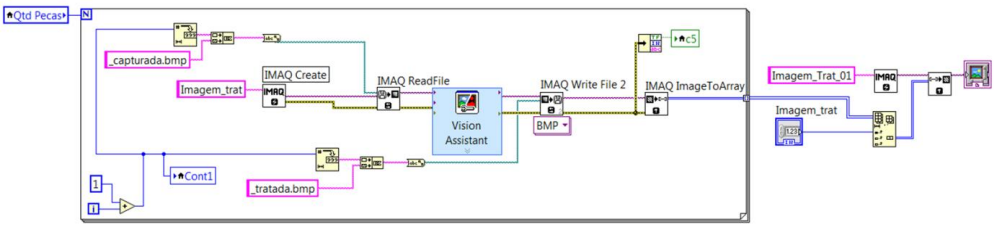


Figura 56. Diagrama de bloco de tratamento de imagem.

Figura 57. Algoritmo de tratamento de imagem no *NI Vision Assistant*.

C3.3 Reconhecimento de Padrão

É o processamento da imagem para reconhecimento e levantamento de características de padrões (ou modelos de objetos). As abordagens de reconhecimento estão

divididas em duas categorias para o reconhecimento em imagem de representação bidimensional (FU; GONZALEZ; LEE, 1987) (JAIN; KASTURI; SCHUNCK, 1995) (GONZALEZ; WOODS, 2007):

1. **Método teórico de decisão**: baseados em descrições quantitativas. A imagem do padrão é representada por uma função e é realizada uma busca de propriedades semelhantes a função na imagem de reconhecimento. Ignorando as relações geométricas que podem ser inerentes na forma de um padrão.
2. **Método estrutural**: baseados em descrições simbólicas e suas relações. É realizado decomposição da imagem padrão em partes primitivas. O comprimento, direção e ordem conhecida de cada parte, orienta a busca do padrão na imagem de reconhecimento. *Shape Matching*, *String Matching*, *Match geometric e Match Patterns*, entre outras, são técnicas deste método.

A Figura 58 apresenta a implementação para o tratamento de imagem na plataforma *LabVIEW*™ e a Figura 59 na ferramenta *NI Vision Assistent*. Este algoritmo é descrito a seguir:

1. Ler o arquivo de imagem de reconhecimento em formato .jpg;
2. Ler o arquivo de imagem do padrão em formato .jpg;
3. Aplicar o algoritmo de *Shape Matching* para identificação do ponto de centroide p(x,y) em *pixel* de cada imagem padrão na imagem de reconhecimento.

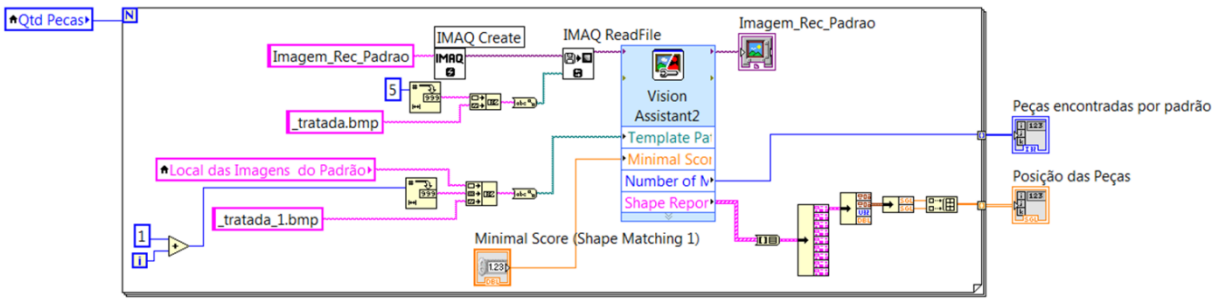


Figura 58. Diagrama de bloco de reconhecimento de imagem.

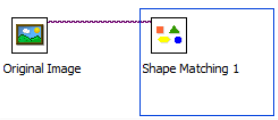


Figura 59. Algoritmo de reconhecimento de imagem no *NI Vision Assistant*.

Apêndice C – Introdução a Ferramenta Computacional *Grail*

Texto adaptado de (GARCIA; CURY, 2006):

O *Grail* é uma ferramenta de computação simbólica para pesquisa e aplicações que envolvem autômatos de estados finitos e linguagens regulares. Foi desenvolvida sob o paradigma da orientação a objeto e implementada em C++. O *Grail* é visto como uma biblioteca de funções (os filtros), acessíveis em linha de comando, que operam sobre máquinas de estados finitos – MF (autômatos de estados finitos e linguagens regulares).

No *Grail* o formato de especificação de uma MF consiste de uma lista de instruções que pode ser armazenada em um arquivo ASCII (utiliza-se a ferramenta *bloco de notas* do ambiente *Windows*®). Cada instrução é uma tripla composta por um estado origem, uma etiqueta de transição e um estado destino. A MF da Figura 60 apresenta um exemplo de especificação no *Grail*:

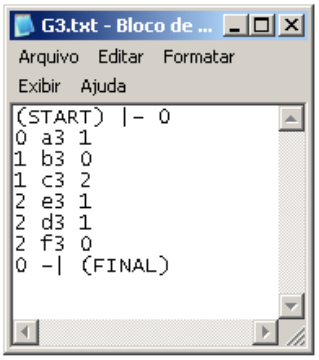


Figura 60. Exemplo de especificação no *Grail*

Os estados no *Grail* são representados por números naturais. Os estados iniciais e finais são indicados por pseudos etiquetas, que são os símbolos especiais |- e -|, (START) e (FINAL) são pseudos estados. O alfabeto é dado pelos símbolos que aparecem nas etiquetas de transição, com exceção das pseudos etiquetas.

Uma máquina de estados finitos pode ser não determinista, no sentido de haver transições com a mesma etiqueta saindo de um mesmo estado e entrando em estados diferentes.

Os filtros do *Grail* são acessíveis em linha de comando. Utiliza-se qualquer janela de terminal do sistema (*prompt* de comando MS-DOS) para acessá-los. Os objetos a serem

utilizados como parâmetros podem ser direcionados diretamente a partir do teclado ou redirecionados a partir de arquivos.

A Tabela 16 apresenta os filtros do *Grail* utilizados nas etapas 2, 3 e 4 do método proposto, que são utilizados para síntese do SML e verificação de propriedades relacionadas ao controle supervisório.

Tabela 16. Filtros utilizados na etapa de síntese.

Filtro	Sintaxe	Descrição
fmsync	fmsync MF1 MF2 > MF3	Calcula a composição síncrona de duas Máquinas Estados Finitos (MF).
fmtrim	fmtrim MF1 > MF2	Encontra a componente trim (MF2) de uma MF1.
fmsupc	fmsupc MF1 MF2 MF3 > MF4	Computa a máxima linguagem controlável (MF4) (Supervisório), em que MF1 é a planta, MF2 a composição da planta com suas especificações (componente trim) e MF3 os eventos não controláveis.
fmsupred	fmsupred MF1 MF2 > MF3	Calcula a redução de supervisórios, em que MF1 é a planta e MF2 é o supervisor (VAZ; WONHAM, 1986).
fmrenum	fmrenum MF1 > MF2	Renumera os estados MF1.
islock	islock MF1	Verifica se uma MF é não bloqueante.
iscreach	iscreach MF1	Verifica se uma MF é coacessível.
isreach	isreach MF1	Verifica se uma MF é acessível.
isnconf	isnconf MF1 MF2 ... MFn	Verifica se as MF’s são não conflitantes.

Apêndice D – Introdução a Ferramenta Computacional *Uppaal*

Texto adaptado de (CARVALHO; SOUSA, 2009):

A ferramenta computacional *Uppaal* de *Model Checking* de sistemas de tempo real. Foi desenvolvida pelos pesquisadores das Universidades de Uppsala e de Aalborg. Sua primeira versão foi lançada em 1995. Ela permite a edição (modelagem), simulação e verificação de autômatos em tempo real utilizando uma interface gráfica amigável conforme Figura 61.

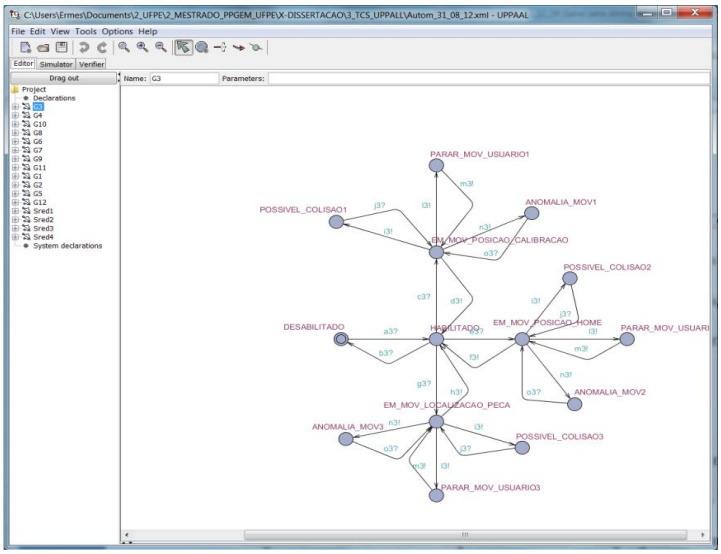


Figura 61. Interface gráfica do *Editor* do *Uppaal*.

A edição é o ambiente no *Uppaal* que é representado os autômatos e supervisórios, que pode ser feita de duas formas distintas, textualmente ou graficamente. Ambas as formas possuem vantagens e desvantagens. Por exemplo, a especificação gráfica torna-se bastante intuitiva, porém alto grau de dificuldade para automação. Ao contrário, a especificação textual permite ser automatizada, mas requer um profundo conhecimento da linguagem.

A especificação gráfica consiste na produção do modelo por intermédio do separador *editor* disponível na interface gráfica do *Uppaal*. Na zona da esquerda consta uma estrutura de opções que permite mudar entre as diferentes partes descritivas do modelo. A opção *Declarations* contém a especificação das variáveis globais do modelo (variáveis inteiras, canais de sincronização, relógios e constantes). A opção $G3$ representa um autômato temporizado do modelo que por sua vez pode ter declarações locais. Por fim a opção *System*

declarations contém a declaração dos processos com a devida atribuição de parâmetros. Os eventos são chamados de *Edges* e possui os parâmetros apresentados na Tabela 17 e os estados são chamados de *Location* e possui os parâmetros apresentados na Tabela 18.

Tabela 17. Parâmetros de *Edges.*

Parâmetro	Descrição do Parâmetro
Select	Permite atribuir um valor a uma variável temporária (apenas válida para essa transição), dado um intervalo, de forma não determinista.
Guard	Serve para ativar a transição sobre uma dada condição.
Sync	Realizar sincronização entre *Edges*. Existem duas etiquetas: A ! para envio (não controlado) e a ? para recepção (controlado).
Update	Utilizado para atualizar variáveis globais ou locais.

Tabela 18. Parâmetros de *Location.*

Parâmetro	Descrição do Parâmetro
Name	Nome do estado que serve de identificador para a linguagem de especificação do modelo e para a linguagem de especificação de propriedades.
Invariant	São condições que podem ou não existir num estado.
Initial/ Urgent/ Committed	Indicam estado inicial (*Initial*), estado prioritário (*Urgent*), estado não priorizado (*Committed*).

A Figura 62 ilustra as janelas do *Edge* e *Location*.

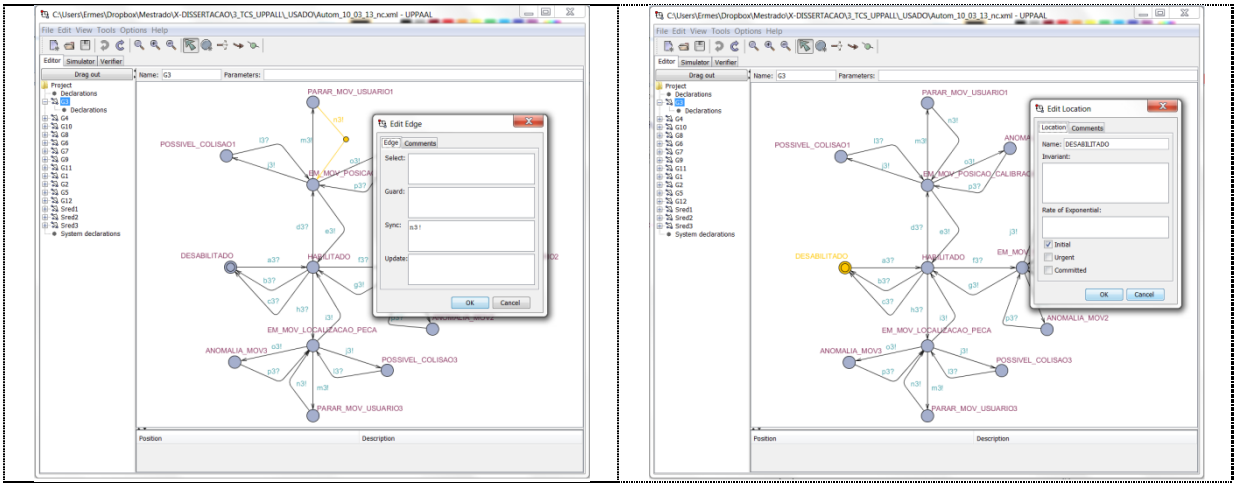


Figura 62. Janelas do *Edge* e *Location.*

A simulação é o ambiente para a avaliação intuitiva e eficiente da construção dos modelos. Dependendo da especificidade dos modelos é possível recorrer à simulação para realizar ajustes e correções. A interface do simulador do Uppaal permite três modos de funcionamento distintos. No primeiro o utilizador executa a simulação passo a passo sendo livre de escolher a transição que pretende fazer. No segundo o utilizador executa a simulação de forma automática e aleatória podendo controlar a velocidade de simulação. No terceiro o utilizador pode percorrer execuções anteriormente realizadas. A Figura 63 apresenta o *Simulador*.

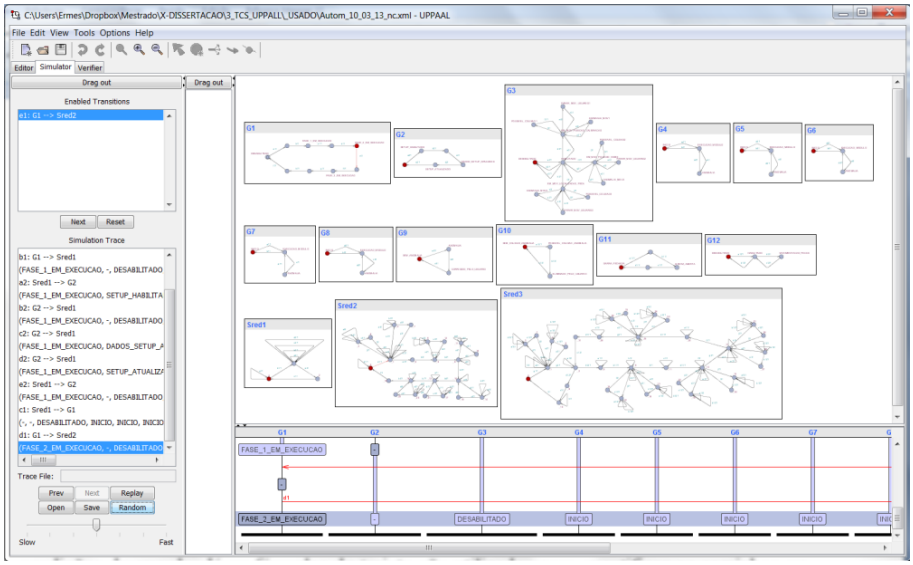


Figura 63. Interface gráfica do *Simulador* do *Uppaal*.

A *Verifier* é o ambiente para o processamento automatizado por algoritmos que permitem confirmação de dadas propriedades desejado ao sistema, apresentado na Figura 64. O *Uppaal* utiliza uma versão simplificada de um tipo de lógica temporal, a CTL (*computacion tree logic*) para a escrita das expressões de verificação. A CTL usa fórmulas de estado e de trajeto por meio de operadores lógicos, temporais e quantificadores de caminho para descrever as linguagens do sistema e os requisitos de especificação durante a verificação.

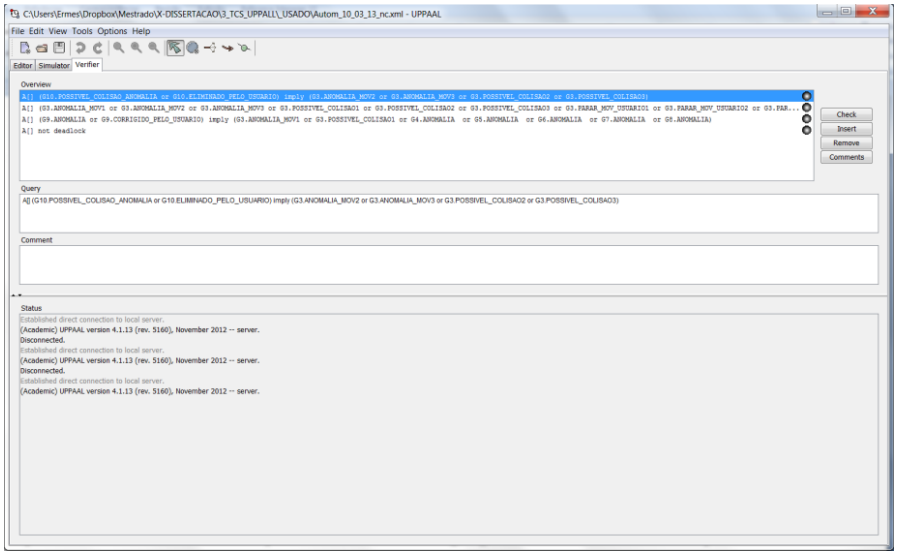


Figura 64. Interface gráfica do *Verifier* do *Uppaal*.

Fórmulas de estado são utilizadas para verificar um estado individual sem interferência do comportamento do sistema. A sintaxe desta fórmula é um conjunto de condições de guarda. Já as fórmulas de trajeto são utilizadas para quantificar os caminhos ou comportamento do sistema, são classificadas em *reachability* (acessibilidade), *safety* (segurança) e *liveness* (evolução).

Tabela 19. Resumo de Sintaxe do *Uppaal*.

Fórmula	Sintaxe	Descrição da Sintaxe
Estado	*α*	Estado ou expressão (estados e operadores lógicos). Exemplo: *G3.HABILITADO* e *(G4.ANOMALIA or G5.ANOMALIA)*.
Trajeto	*A[] not deadlock*	Todos os estados são alcançáveis.
Trajeto	*E<> α*	Existe pelo menos um caminho partindo do estado inicial, tal que α é eventualmente satisfeito.
Trajeto	*A<> α*	Para todo o caminho partindo do estado inicial, tal que α é eventualmente satisfeito.
Trajeto	*E[] α*	Existe pelo menos um caminho partindo do estado inicial, tal que α é satisfeito.
Trajeto	*A[] α*	Para todo o caminho partindo do estado inicial, tal que α é satisfeito.
Trajeto	*α -->β*	Para todo *α* satisfeito, *β* é eventualmente satisfeito.

Apêndice E – Introdução ao Ambiente *LabVIEW*™

O *LabVIEW*™ (*Laboratory Virtual Instrument Engineering Workbench*) utiliza uma linguagem de programação chamada G. Essa linguagem possui diretivas, como PASCAL e C, porém ao invés de utilizar comandos na forma de texto para gerar as linhas de código, usa uma linguagem de programação gráfica. Ou seja, o programa é feito na forma de um diagrama de blocos. Por utilizar uma estrutura de programação orientada pelo fluxo de dados e hierárquica, o *LabVIEW*™ torna simples a implementação de sistemas complexos que englobam aquisição e manipulação de dados, ou ainda o controle de equipamentos por meio do computador. Além disso, o *LabVIEW*™ inclui diversas bibliotecas compostas por componentes, contendo funções para aplicações específicas (algoritmos de análise estatística, aquisição e tratamento de imagens, etc.).

Qualquer programa feito em *LabVIEW*™ é chamado de instrumento virtual (*VI* – *Virtual Instrument*) já que sua aparência e operação assemelham-se às de instrumentos reais. Um *VI*, assim como um programa usual, é composto por um conjunto de instruções que fazem a manipulação e fluxo dos dados, e por uma interface com o usuário, é possível encontrar as entradas e saídas necessárias. Basicamente, pode-se identificar em um *VI* duas partes que o compõem:

Painel frontal: constitui a interface com o usuário, apresentando de forma visual todos os controles, gráficos e indicadores que formam uma tela, que simula o painel físico de um instrumento. Este pode ser formado por botões, *leds*, *knobs* e indicadores que permitem a interação por meio do *mouse* ou do teclado do computador.

Diagrama de blocos: é a estrutura do programa propriamente dita que contém o código fonte construído de forma gráfica.

Pode-se ainda encapsular um *VI* inteiro (isto é, diagrama de blocos + painel frontal) em um módulo reutilizável dentro de um outro *VI*. Esse módulo encapsulado constituirá um *subVI*. A Figura 65 ilustra o ambiente *LabVIEW*™ apresentando o painel frontal e o diagrama de blocos.

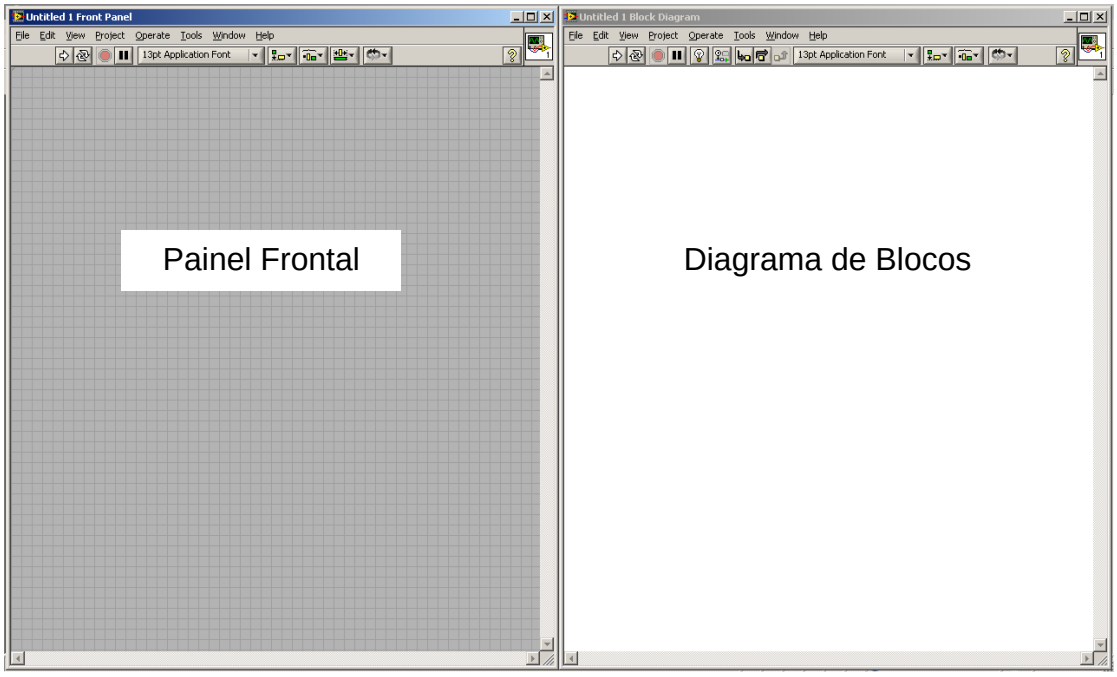


Figura 65. Ambiente *LabVIEW™*.

Na parte superior do painel frontal há uma barra de ferramentas como apresentada na Figura 66.

Figura 66. Barra de Ferramentas.

Em que,

 Roda o *VI*, uma vez.

 Roda o *VI*, continuadamente, isto é, quando chega ao final, volta novamente ao início e recomeça.

 Aborta a execução do *VI*.

 Pausa / Continua a execução do *VI*.

Os componentes de um *VI* genérico são:

Ícone / conector: define as entradas e saídas do *VI* acessíveis à conexão quando este é utilizado como um *subVI* dentro de um outro *VI* (é análogo às definições de entrada e saída para se usar uma rotina como função dentro de um programa numa linguagem convencional). O ícone é a definição da aparência gráfica que se deseja que este *VI* tenha no diagrama de blocos quando usado como um *subVI*.

Paleta de Função e Controle: As paletas de função e controle contém sub-paletas de objetos que podem ser utilizadas para criar um *VI*. A paleta de controle para indicar indicadores e controles, apresentada na Figura 67, enquanto A paleta de função serve para montar o diagrama de bloco, mostrada na Figura 68.

Controles e Indicadores: No painel frontal as entradas e saídas do *VI* são representadas respectivamente por controles e indicadores, que visualmente podem ser apresentados seja como *knobs*, botões, indicadores digitais, *leds*, ou outros formatos e tipos.

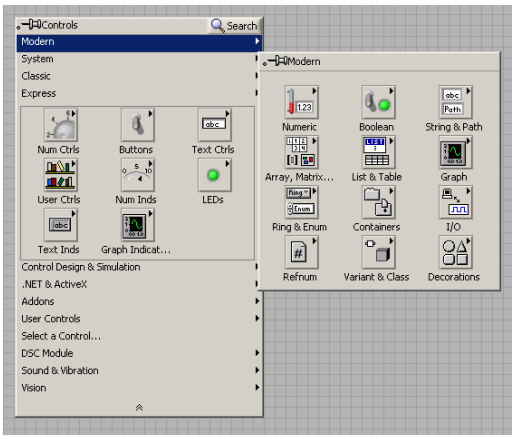


Figura 67. Paleta de Controle.

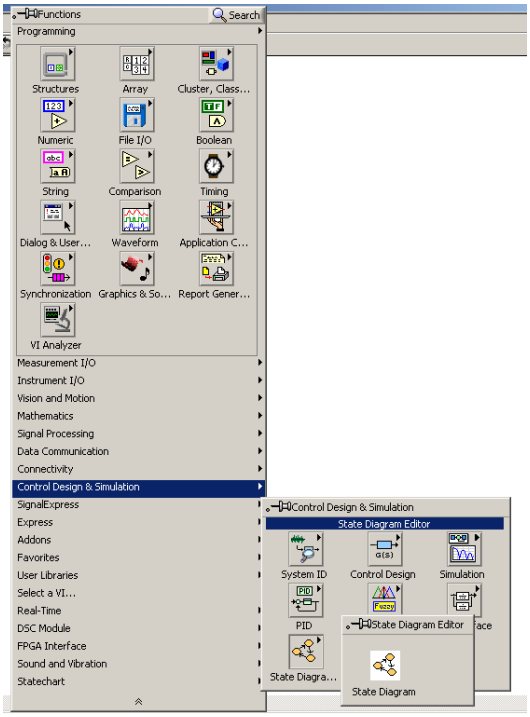


Figura 68. Paleta de Funções.

Os componentes utilizados na Programação são:

Estruturas e Gráficos: Assim como em outras linguagens nos programas contém laços, no *LabVIEW™* há determinadas estruturas que representam graficamente laços, como o for e o *while* no diagrama de blocos. Além disso, é possível usar estruturas de decisão, como o *case* ou ainda estruturas específicas do *LabVIEW™*, como o *sequence* e a *formula node*.

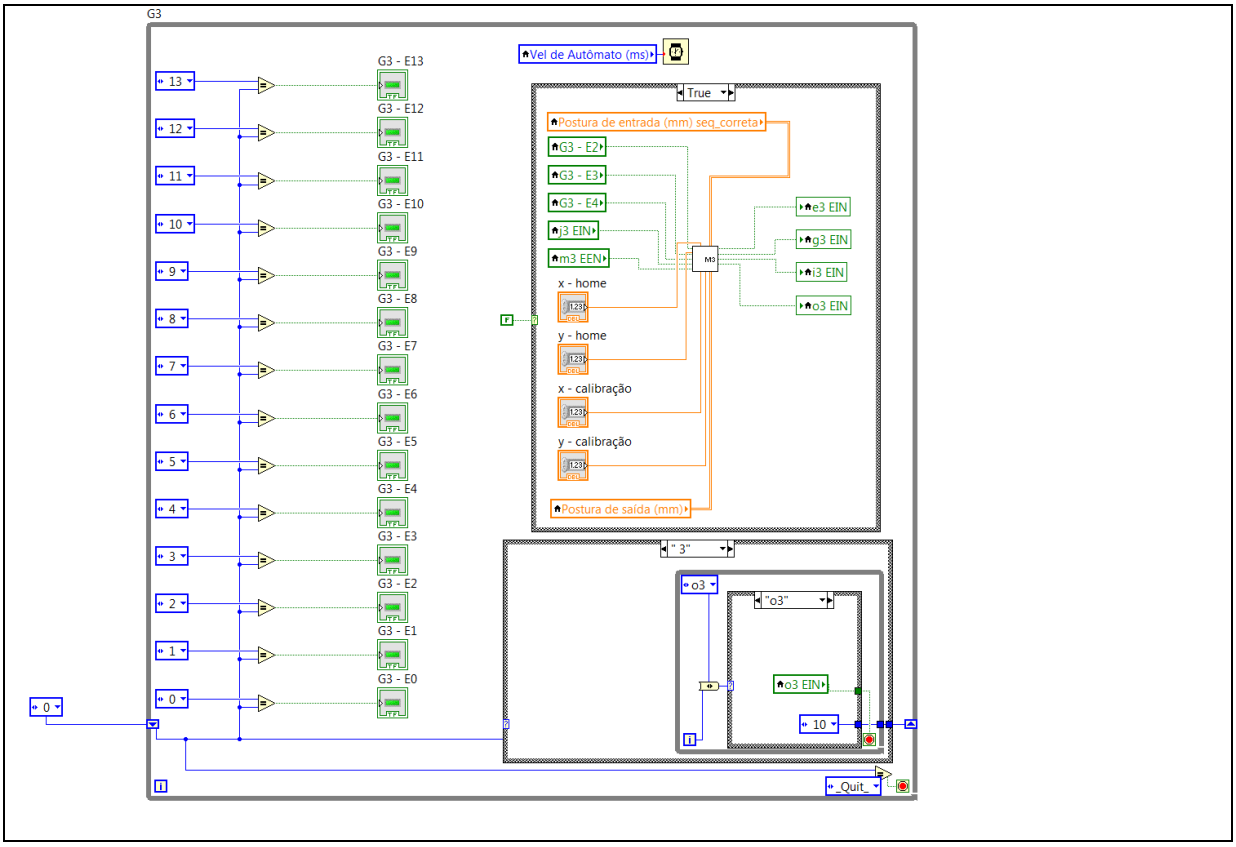
Ligações: Transfere os dados entre objetos do diagrama de bloco através de ligações. São semelhantes às variáveis em linguagens de programação baseadas em texto. Cada ligação tem uma única fonte de dados, mas podem ser ligadas a vários *VI´s* e funções que fazem a leitura dos dados. As ligações são de cores, estilos e espessuras diferentes, dependendo de seus tipos de dados.

Blocos: Processam os dados enviados pelas ligações. O *LabVIEW™* possui diversos blocos utilizados para operações matemáticas, lógicas, processamento de imagens e sinais, operações de sequênciamento e laços de repetições, temporizados e contadores, entre outros.

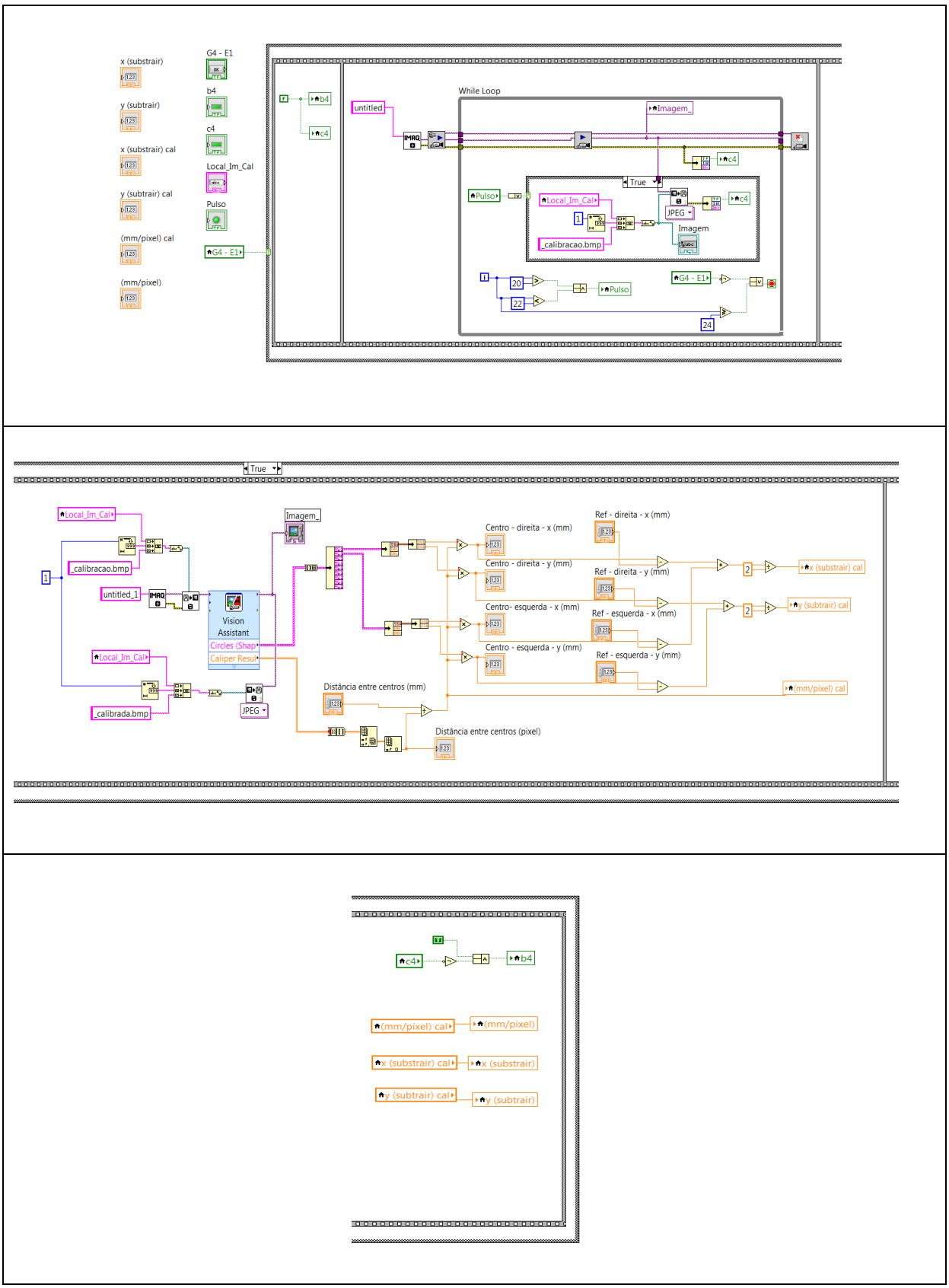
Apêndice F – Módulos Computacionais

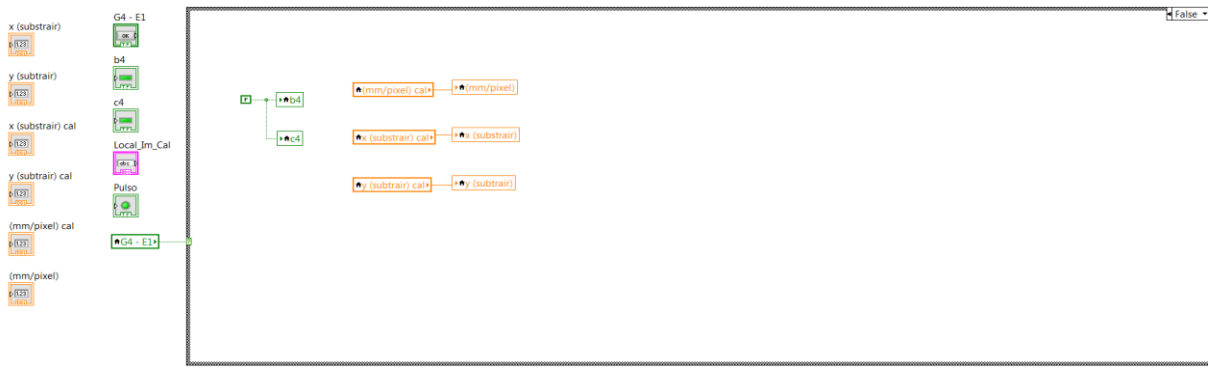
É apresentado neste apêndice todos os programa desenvolvidos no ambiente LabVIEW™ no estudo de caso conforme Tabela 13.

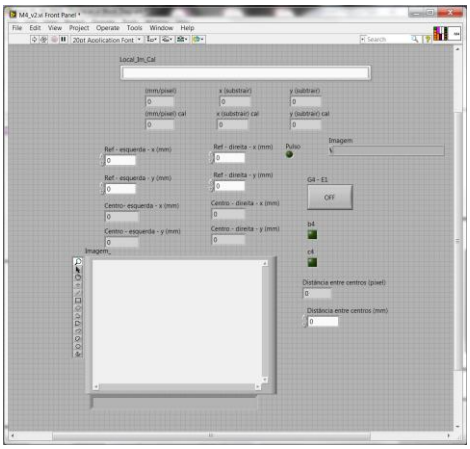
M3 – Movimentar o robô

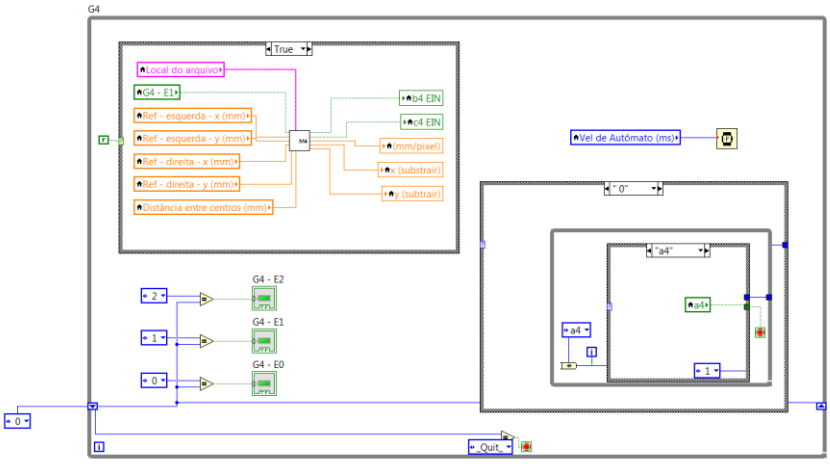


M4 – Calibrar e ajustar robô ao sistema de visão

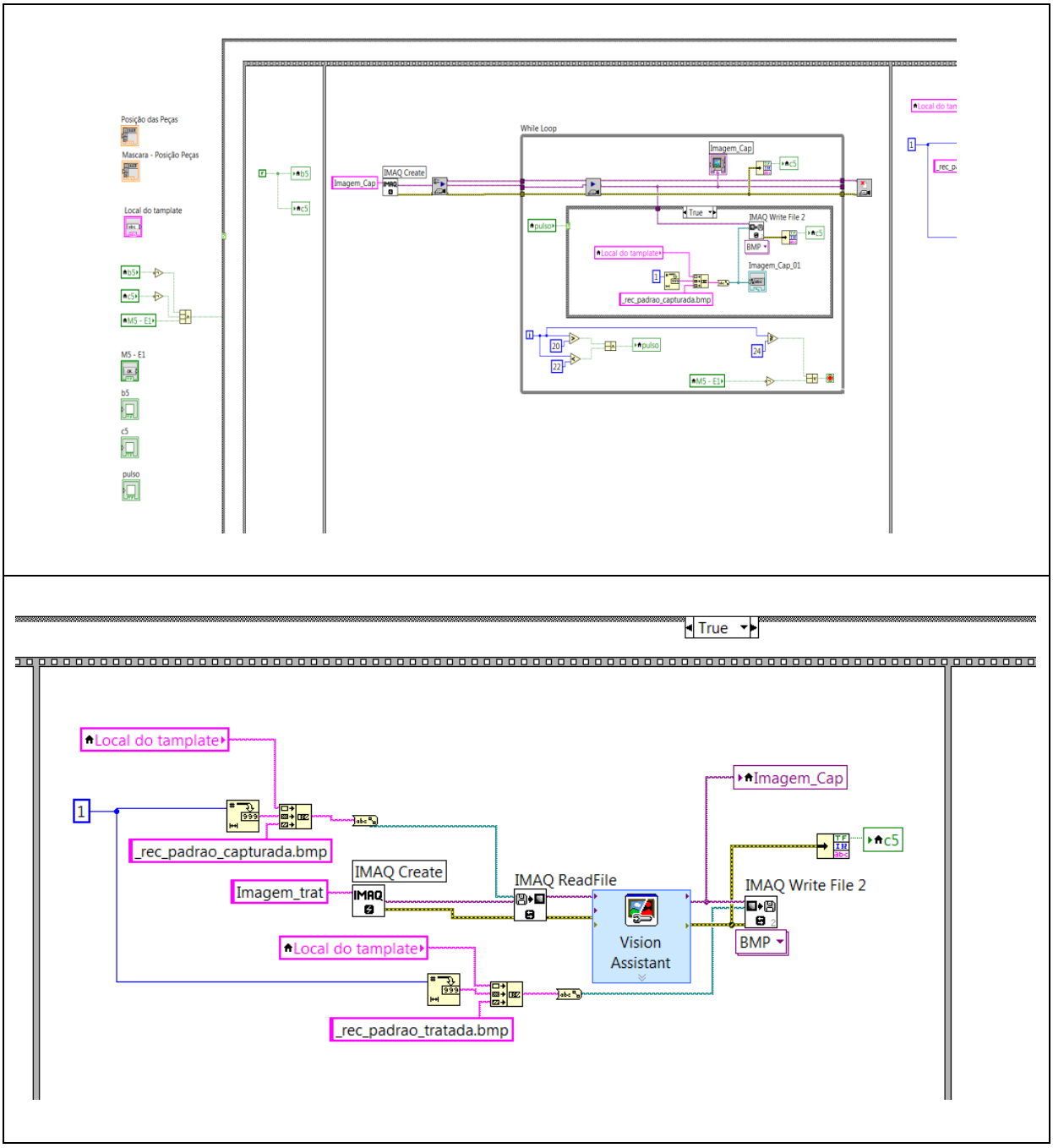


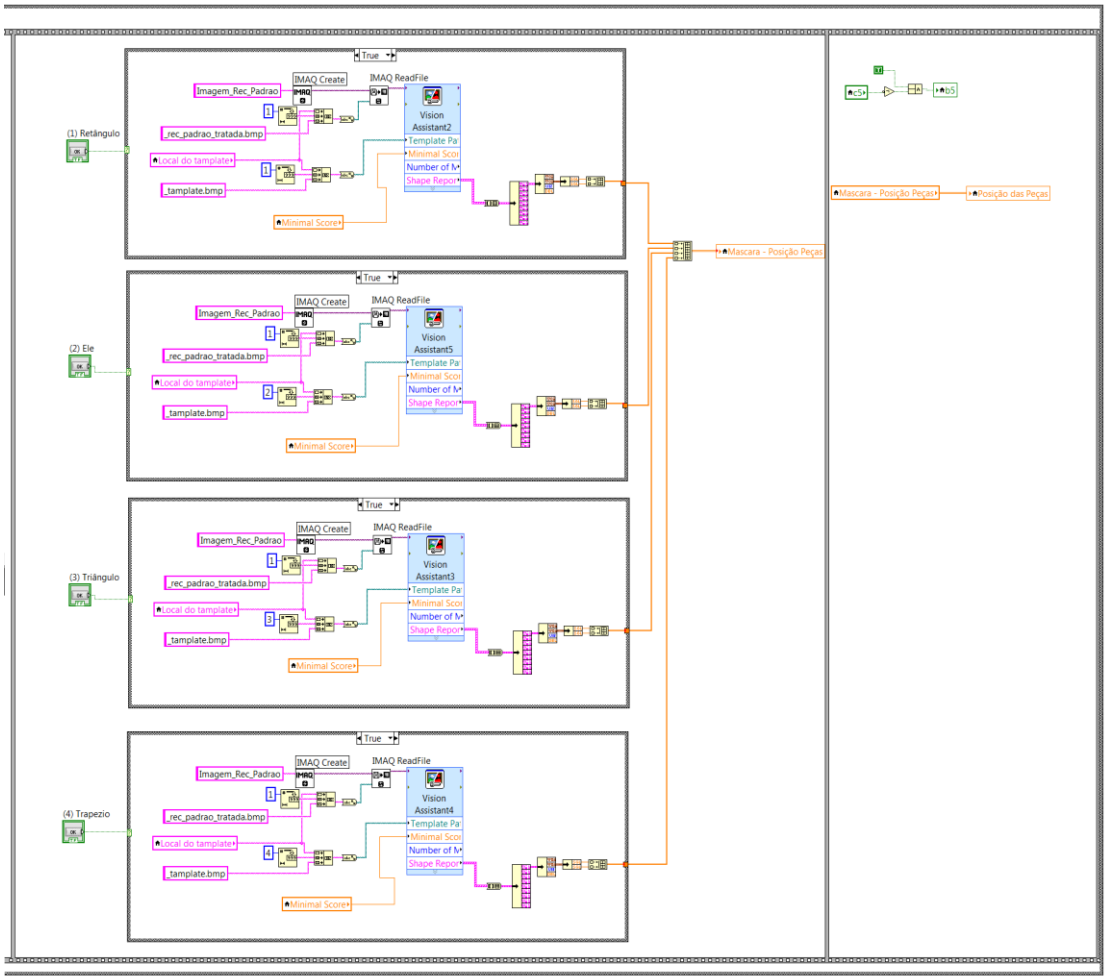

x (substrair)
y (subtrair)
x (substrair) cal
y (subtrair) cal
(mm/pixel) cal
(mm/pixel)
G4 - E1
b4
c4
Local_Im_Cal
Pulso
G4 - E1
b4
c4
(mm/pixel) cal
(mm/pixel)
x (substrair) cal
x (substrair)
y (subtrair) cal
y (subtrair)
False

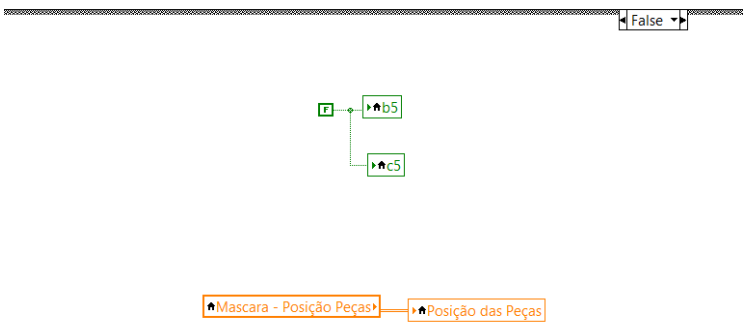


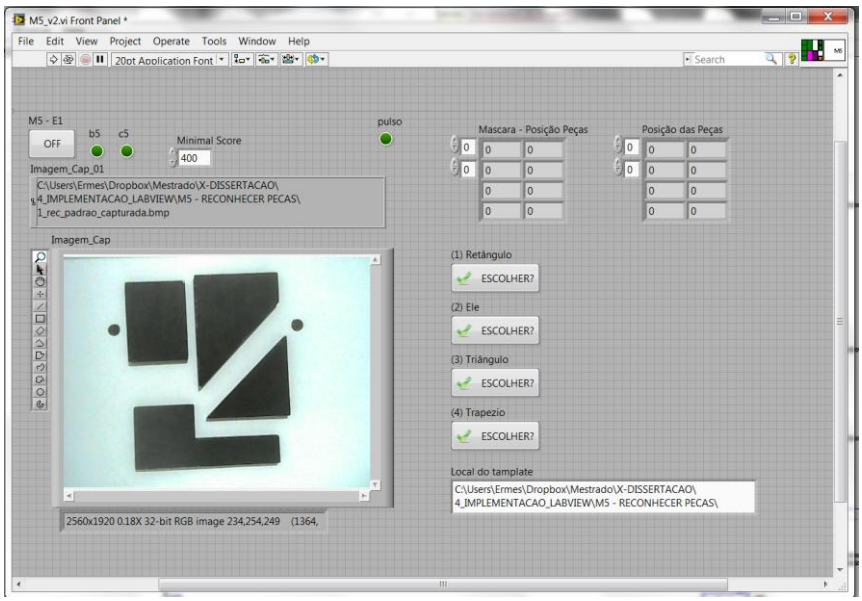

G4
True
Local do arquivo
G4 - E1
Ref - esquerda - x (mm)
Ref - esquerda - y (mm)
Ref - direita - x (mm)
Ref - direita - y (mm)
Distância entre centros (mm)
b4 EIN
c4 EIN
(mm/pixel)
x (substrair)
y (subtrair)
Vel de Autômato (ms)
"0"
"a4"
a4
1
G4 - E2
G4 - E1
G4 - E0
2
1
0
0
Quit

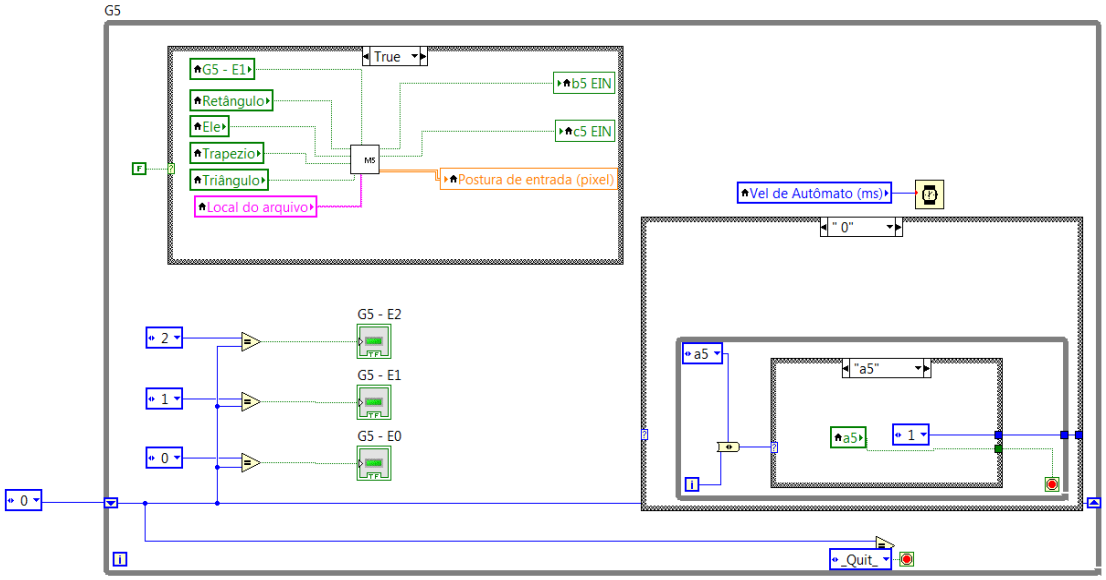
M5 – Reconhecer peças pelo sistema de visão



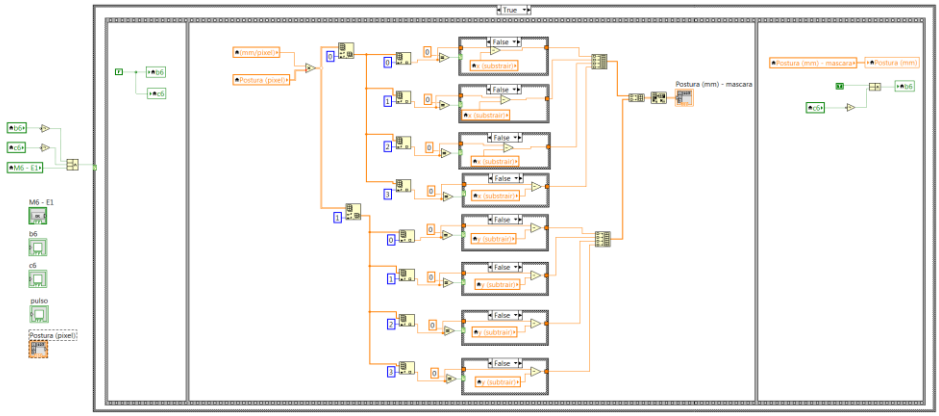

True
IMAQ Create
IMAQ ReadFile
Imagem_Rec_Padrao
_rec_padrao_tratada.bmp
Local do tamplate
_tamplate.bmp
Minimal Score
Vision Assistant2
Vision Assistant5
Vision Assistant3
Vision Assistant4
(1) Retângulo
(2) Ele
(3) Triângulo
(4) Trapezio
Mascara - Posição Peças
Posição das Peças
b5
c5

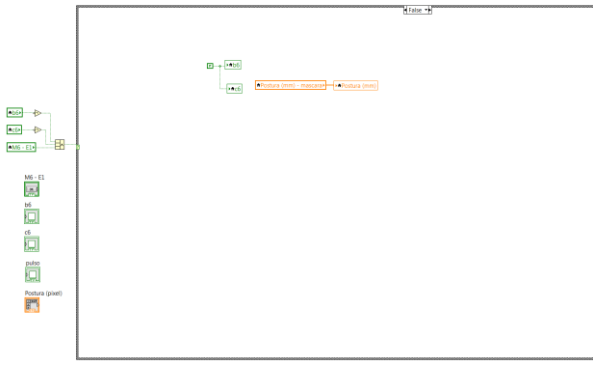

False
b5
c5
Mascara - Posição Peças
Posição das Peças

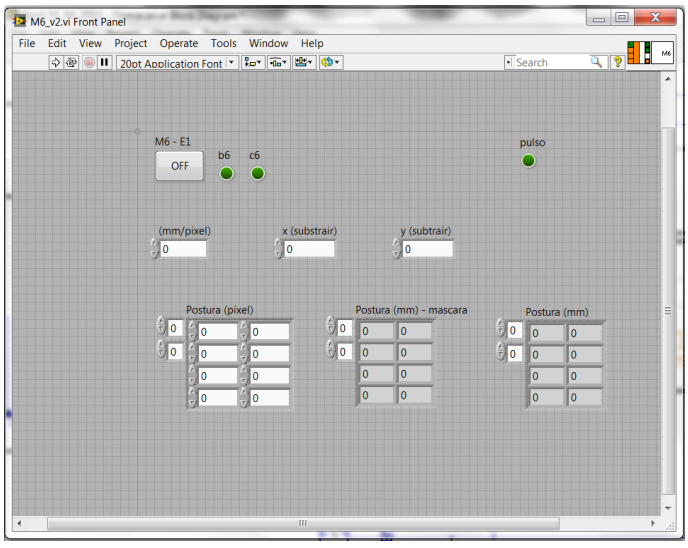

M5_v2.vi Front Panel *
File Edit View Project Operate Tools Window Help
M5 - E1
OFF
b5
c5
Minimal Score
400
pulso
Imagem_Cap_01
C:\Users\Ermes\Dropbox\Mestrado\X-DISSERTACAO\4_IMPLEMENTACAO_LABVIEW\M5 - RECONHECER PECAS\1_rec_padrao_capturada.bmp
Imagem_Cap
2560x1920 0.18X 32-bit RGB image 234,254,249 (1364,
Mascara - Posição Peças
Posição das Peças
(1) Retângulo
ESCOLHER?
(2) Ele
ESCOLHER?
(3) Triângulo
ESCOLHER?
(4) Trapezio
ESCOLHER?
Local do tamplate
C:\Users\Ermes\Dropbox\Mestrado\X-DISSERTACAO\4_IMPLEMENTACAO_LABVIEW\M5 - RECONHECER PECAS\

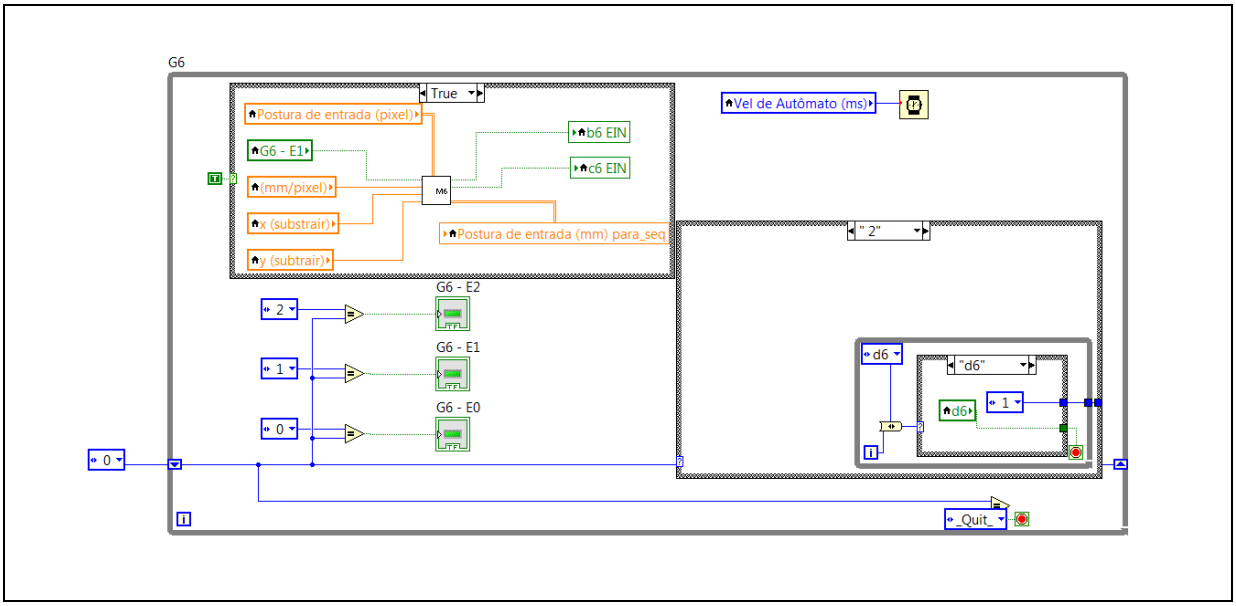

G5
True
G5 - E1
Retângulo
Ele
Trapezio
Triângulo
Local do arquivo
b5 EIN
c5 EIN
Postura de entrada (pixel)
Vel de Autômato (ms)
" 0"
G5 - E2
G5 - E1
G5 - E0
2
1
0
a5
"a5"
Quit

M6- Determinar as posturas das peças selecionadas na entrada para serem movimentadas

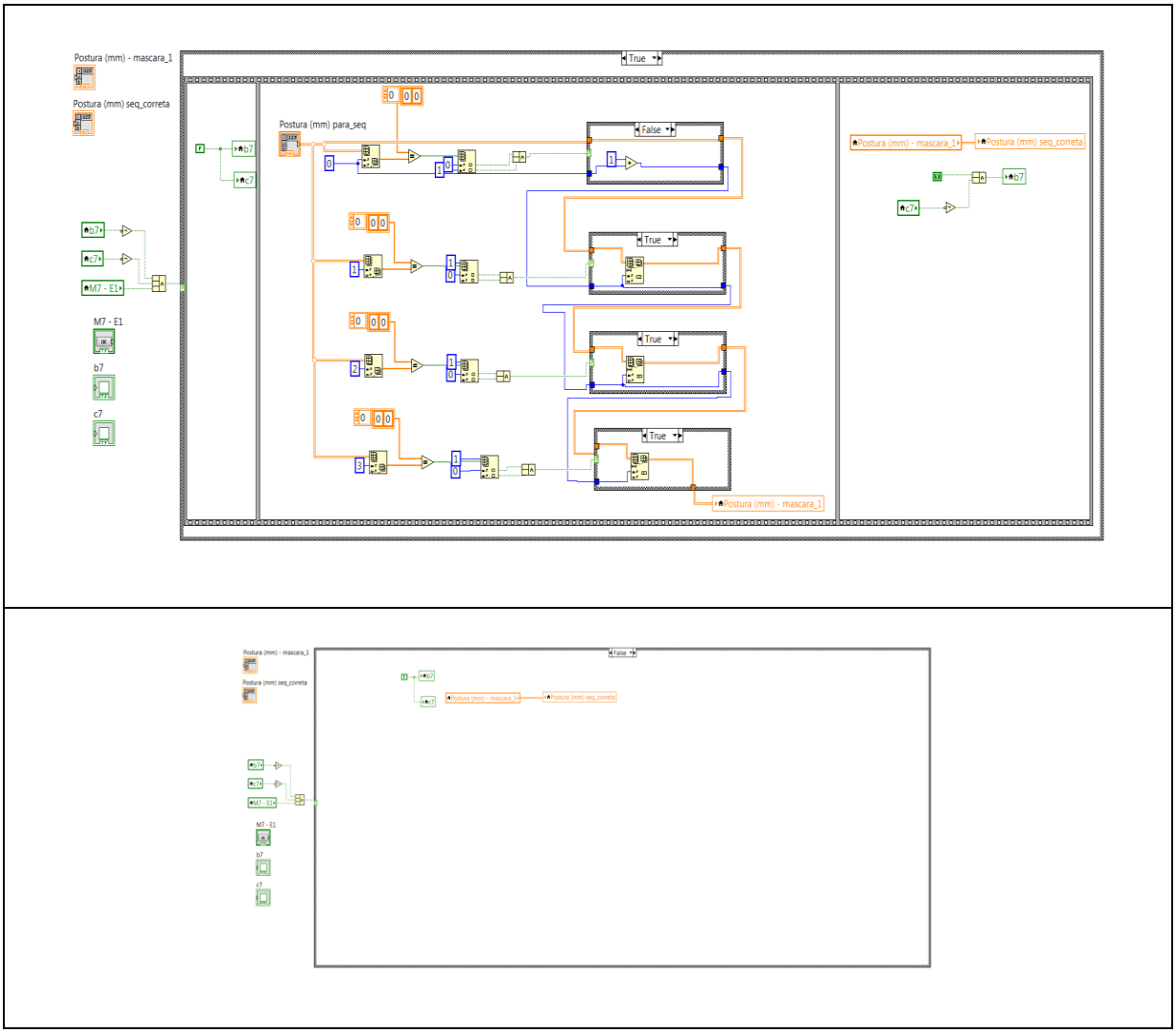


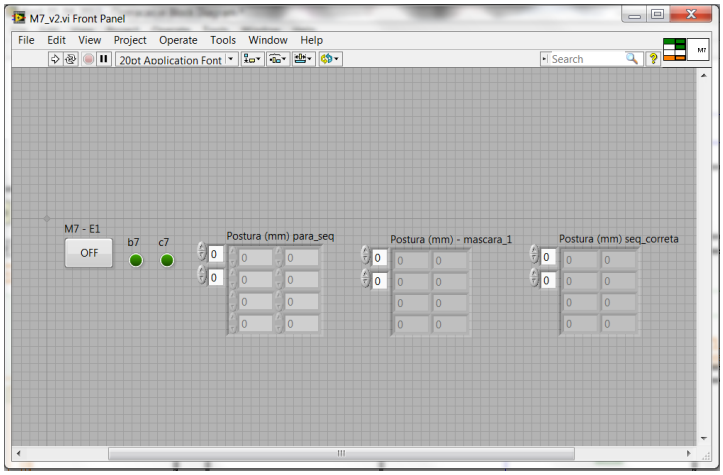


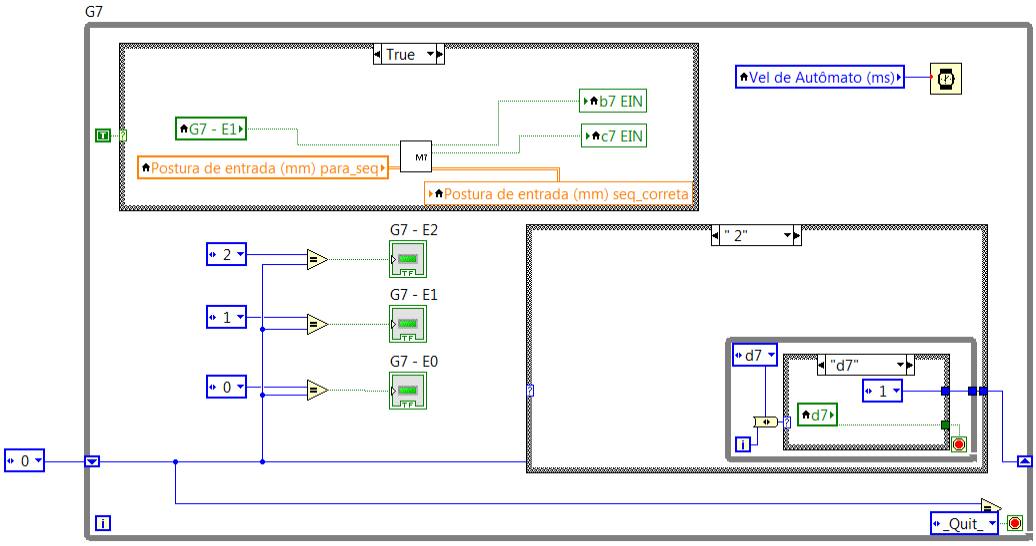




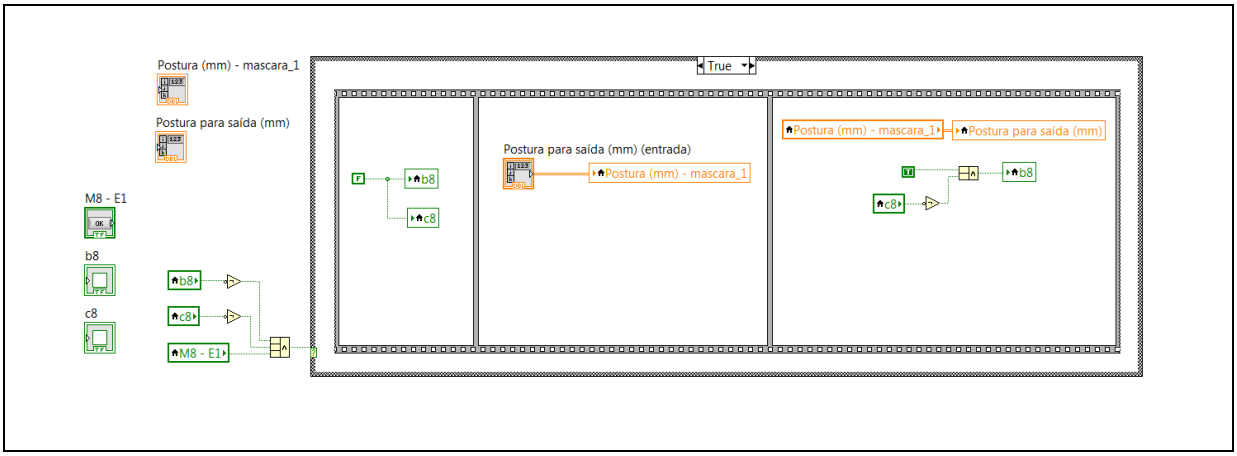
M7 – Determinar a sequência de movimentação

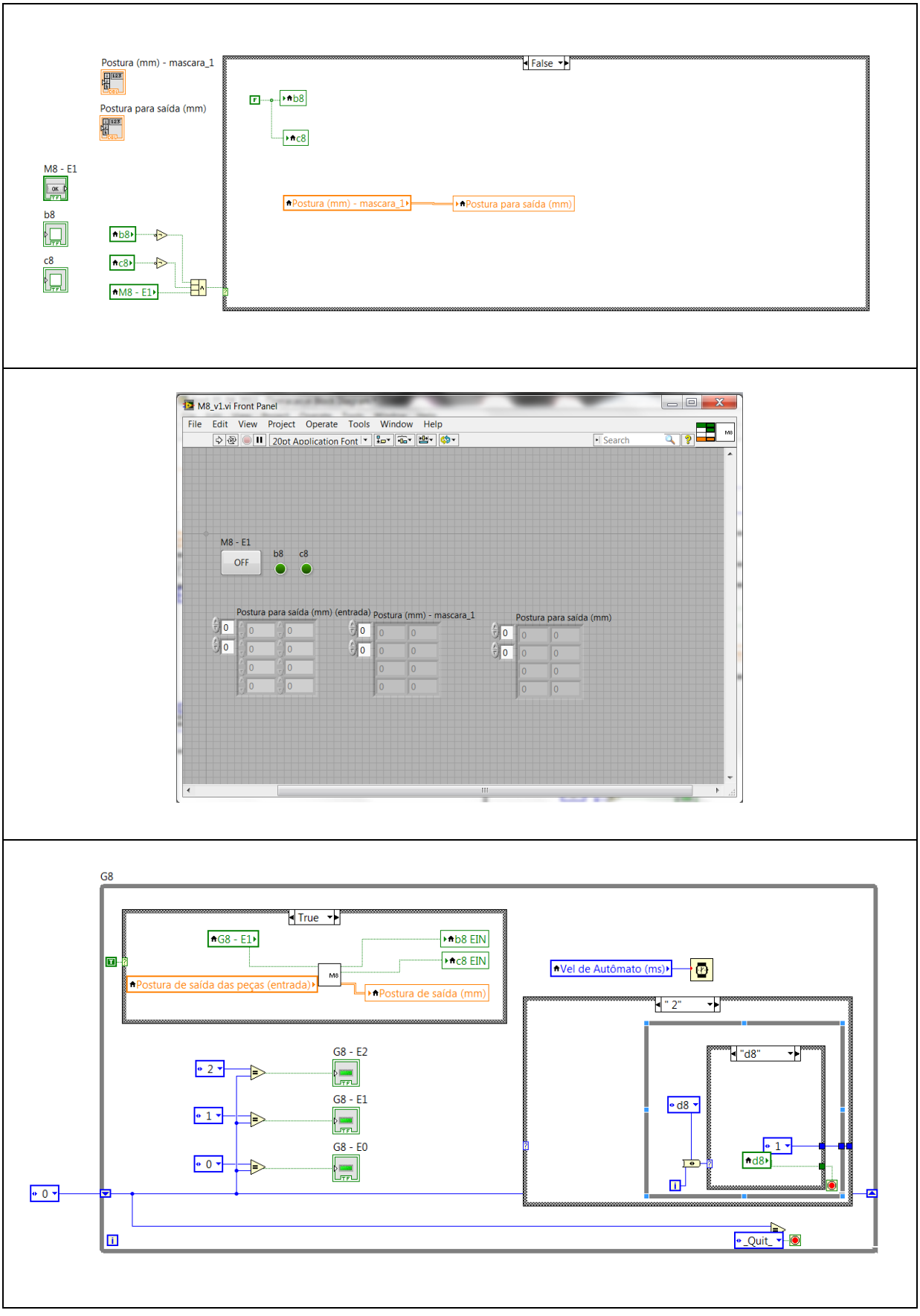




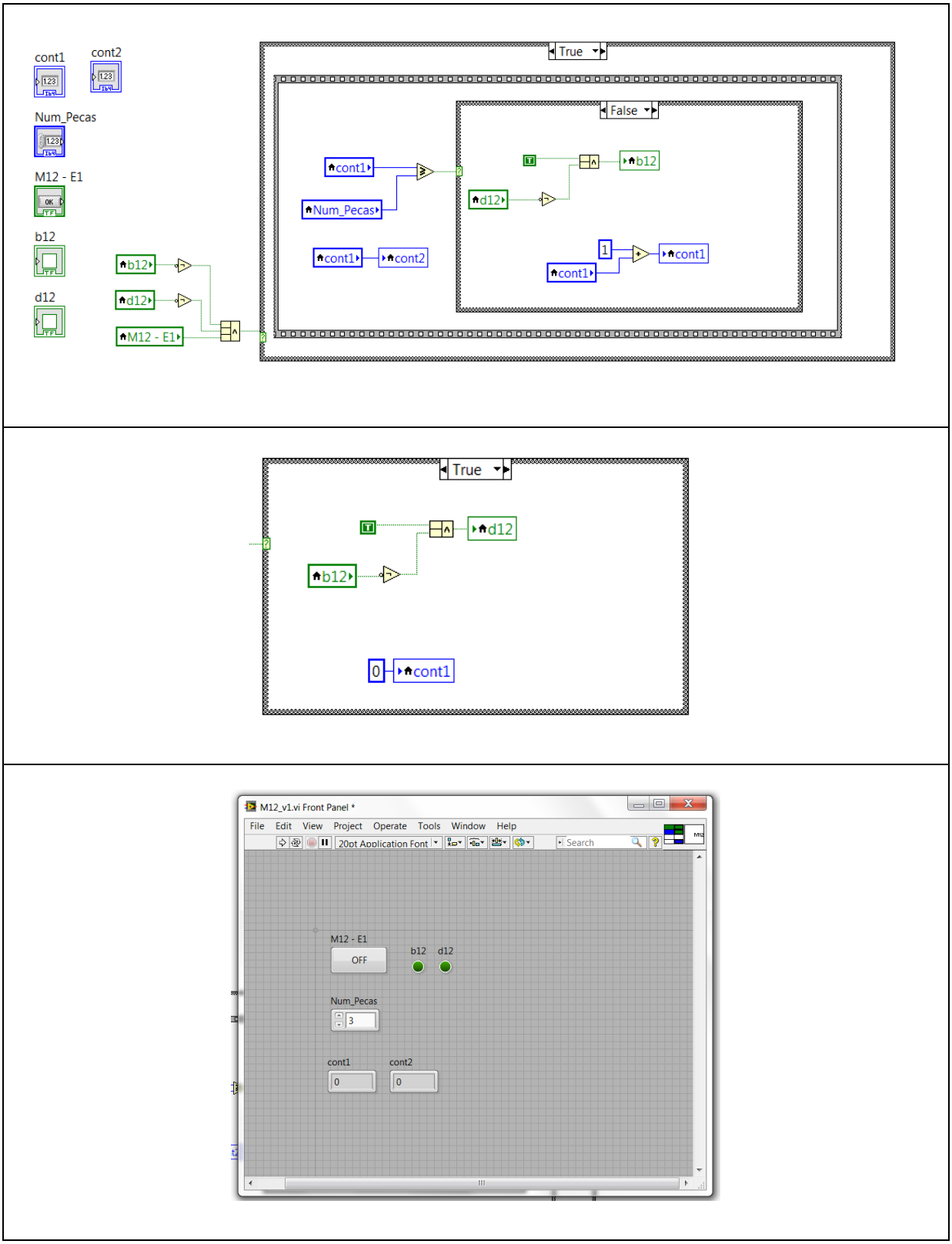


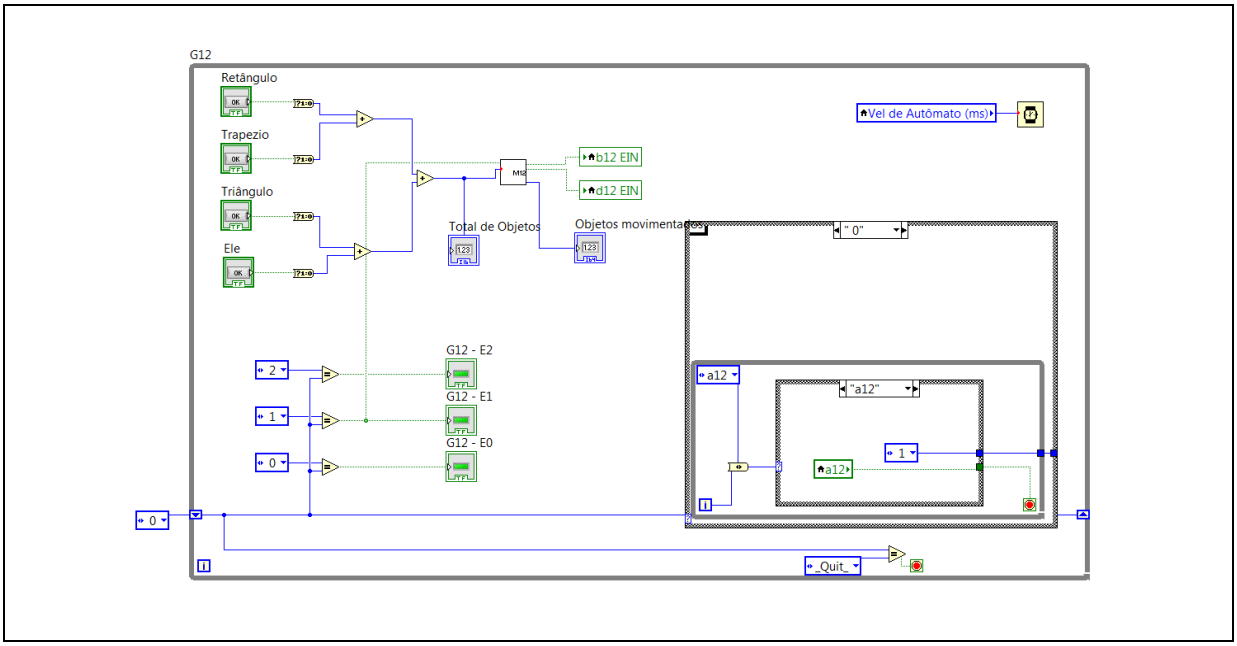
M8 – Determinar as posturas da saída das peças




Postura (mm) - mascara_1
Postura para saída (mm)
M8 - E1
b8
c8
False
b8
c8
Postura (mm) - mascara_1
Postura para saída (mm)
M8_v1.vi Front Panel
File Edit View Project Operate Tools Window Help
20pt Application Font
Search
M8 - E1
OFF
b8
c8
Postura para saída (mm) (entrada)
Postura (mm) - mascara_1
Postura para saída (mm)
G8
True
G8 - E1
b8 EIN
c8 EIN
Postura de saída das peças (entrada)
M8
Postura de saída (mm)
Vel de Autômato (ms)
" 2"
"d8"
d8
1
d8
G8 - E2
G8 - E1
G8 - E0
2
1
0
0
Quit

M12 – Gerir a movimentação das peças




G12
Retângulo
Trapezio
Triângulo
Ele
Total de Objetos
Objetos movimentados
b12 EIN
d12 EIN
Vel de Autômato (ms)
G12 - E2
G12 - E1
G12 - E0
" 0"
a12
"a12"
1
2
0
Quit

Apêndice G – Modelagem Cinemática Direta do *Robot* 5150A

A Figura 69 apresenta o sistema de coordenadas de cada junta do robô educacional *Robot* 5150A fabricante *Lab-Volt*.

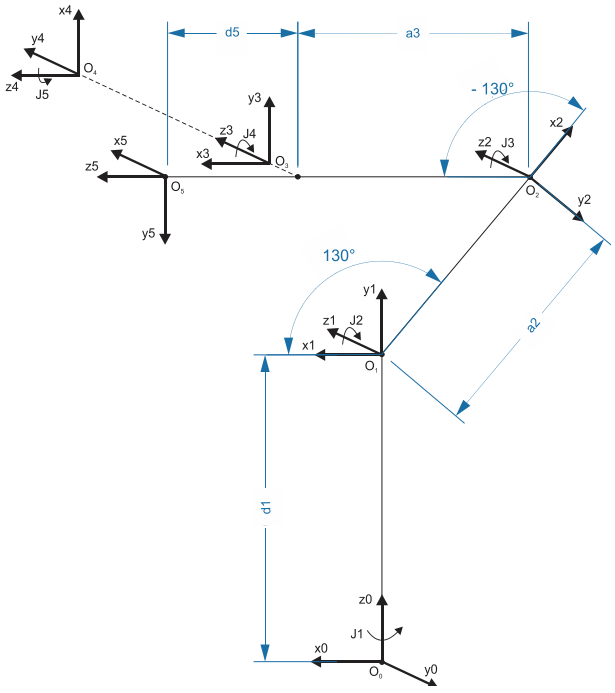


Figura 69. Sistema de Coordenadas de cada junta do *Robot* 5150A.

Os parâmetros *Denavit-Hartenberg* do *Robot* 5150A é apresentado na Tabela 20. O motor de passo da junta $J3$ está invertido no robô, ou seja, quando avança os passos, diminui o grau, conforme conversão.

Tabela 20. Parâmetros D-H do *Robot* 5150A.

Junta (Ji)	Descrição da Junta	θ_i	d_i	a_i	α_i	Faixa da Junta	Passos/ grau
J1	*Base*	*0°*	*d1 = 255,5 mm*	*0*	*90°*	*-185° até 153°*	*18*
J2	*Should*	*130°*	*0*	*a2 = 190,5 mm*	*0°*	*-32° até 149°*	*24*
J3	*Elbow*	*-130°*	*0*	*a3 = 190,5 mm*	*0°*	*-147° até 51°*	*24*
J4	*Pitch*	*90°*	*0*	*0*	*90°*	*20° até 130°*	*18*
J5	*Roll*	*90°*	*d5 = 115 mm*	*0*	*0°*	*-360° até 360°*	*36*

A cinemática direta é obtida pela matriz de transformação homogênea (T_5^o) da equação (35):

$$T_5^o = \begin{vmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad (35)$$

Em que,

$$n_x = \cos(q1)\cos(q2+q3+q4)\cos(q5) + \sin(q1)\sin(q5) \quad (36)$$

$$s_x = \cos(q5)\sin(q1) - \cos(q1)\cos(q2+q3+q4)\sin(q5) \quad (37)$$

$$a_x = \cos(q1)\sin(q2+q3+q4) \quad (38)$$

$$p_x = \cos(q1)\left(\sin(q2)\,(115\cos(q3)\cos(q4) + \sin(q3)\left(-190.5 - 115\sin(q4)\right)\right) + \cos(q2)\left(190.5 + 115\cos(q4)\sin(q3) + \cos(q3)\,(190.5 + 115\sin(q4))\right) \quad (39)$$

$$n_y = \cos(q2+q3+q4)\cos(q5)\sin(q1) - \cos(q1)\sin(q5) \quad (40)$$

$$s_y = -\cos(q1)\cos(q5) - \cos(q2+q3+q4)\sin(q1)\sin(q5) \quad (41)$$

$$a_y = \sin(q1)\sin(q2+q3+q4) \quad (42)$$

$$p_y = \sin(q1)\sin(q2)\left(115\cos(q3)\cos(q4) + \sin(q3)\,(-190.5 - 115\sin(q4))\right) + \cos(q2)\sin(q1)\left(190.5 + 115\cos(q4)\sin(q3) + \cos(q3)\,(190.5 + 115\sin(q4))\right) \quad (43)$$

$$n_z = \cos(q5)\sin(q2+q3+q4) \quad (44)$$

$$s_z = -\sin(q2+q3+q4)\sin(q5) \quad (45)$$

$$a_z = -\cos(q2+q3+q4) \quad (46)$$

$$p_z = 255.5 + \sin(q2)\left(190.5 + 115\cos(q4)\sin(q3) + \cos(q3)\,(190.5 + 115\sin(q4))\right) + \cos(q2) * \left(-115\cos(q3)\cos(q4) + \sin(q3)\,(190.5 + 115\sin(q4))\right) \quad (47)$$