Abordagem Sistemática para Projeto de Sistemas de ManufaturaFlexível baseada em Técnicas Formais

Systematic approach flexible manufacturing systems design based on formal techniques

Ermes Ferreira Costa Neto[1] orcid.org/0000-0003-0666-840X

Pedro Manuel González del Foyo[2] orcid.org/0000-0002-3151-7761

André Murilo de Almeida Pinto[3] orcid.org/0000-0002-5101-0685

[1] Coordenação de Engenharia Mecânica, Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, PE, Brasil.
[2]Departamento de Engenharia Mecânica, Centro de Tecnologia e Geociências, Universidade Federal de Pernambuco, Recife, PE, Brasil.
[3]Departamento de Engenharia Mecânica, Faculdade Gama, Universidade de Brasília, Gama, DF, Brasil.

E-mail do autor principal: Ermes Ferreira Costa Neto ermescosta@poli.br

RESUMO

Atualmente, as empresas de manufatura precisam de mais flexibilidade para permanecer competitivas no mercado. Novas técnicas para projetar e implementar sistemas de manufatura flexível em menos tempo são pontos cruciais para aumentar a produtividade em tais indústrias. O objetivo deste artigo é apresentar uma abordagem sistemática por meio do uso de diversas técnicas formais, a fim de automatizar processos de fabricação de forma rápida, evitando a migração de erros entre as fases do projeto. A chave para alcançar esse desempenho é usar formalismo equivalente em todas as fases do projeto, incluindo verificação e implementação. A abordagem foi aplicada à automação de uma célula de manufatura robotizada didática, tipo de sistema de manufatura flexível, e os resultados mostram que ela pode ser útil em aplicações reais da indústria, reduzindo o custo e diminuindo o tempo de desenvolvimento no processo de projeto.

PALAVRAS-CHAVE*:* Sistemas de Manufatura Flexível; Teoria de Controle Supervisório; Projeto Modular

ABSTRACT

Nowadays, manufacturing companies need more flexibility to remain competitive in the market. New techniques to design and implement manufacturing processes in less time are crucial point to increase productivity in such industries. The aim of this paper is to present a systematic approach through several formal techniques to automate manufacturing processes in a rapid way avoiding error migration between design phases. The key to achieve such performance is to use equivalent based formalism through all design phases, including verification and implementation. The approach was applied to the automation of a robotic manufacturing cell and the results show that it could be useful in industry real applications, reducing cost and decreasing development time in design process.

KEY-WORDS: *Flexible Manufacturing Systems; Supervisory Control Theory; Modular Design;*

DOI: 10.25286/repa.v4i3.993

1 INTRODUÇÃO

As empresas de manufatura, para se manterem competitivas no mercado atual, precisam ser cada vez mais flexíveis em seus sistemas de produção **[1]**. Para atender tais necessidades, os tradicionais sistemas deprodução em série estão sendo substituídos por modernos sistemas automáticos, compostos por múltiplos subsistemas, que podem produzir qualquer produto em diferentes combinações e programações **[2]**. Esses novos sistemas automáticos são chamados de *Flexible Manufacturing Systems* (FMS).

No entanto, muitas dessas empresas encontram dificuldades na implantação de FMS, na maioria das vezes, devido ao custo financeiro e a complexidade técnica do projeto **[3]**. Para essa última, destaca-se a etapa de projeto do controle supervisório, que busca a coordenação dos diversos subsistemas de forma que atendam a uma série de tarefas individuais e conjuntas, garantindo o bom funcionamento global da FMS **[4]**.

No apoio ao projeto do supervisório de FMS, diversas abordagens formais têm sido propostas, dentre as quais se incluem a Teoria de Controle Supervisório (TCS) **[5]**, Redes de Petri **[6]**, Cadeias de Markov **[7]** e Teoria das Filas **[8]**. A maior parte dessas abordagens limita-se à análise de soluções propostas, que são geradas com base na experiência e inspiração do projetista **[4]**. Em contrapartida, a TCS utiliza a modelagem da planta e das especificações para permitir a síntese automática de supervisórios ótimos **[5]**.

Para a geração do supervisório ótimo pela TCS, é necessário a realização de cálculos polinomiais, sendo que, quanto maior o número de estados dos modelos da planta e especificações, maior será a exigência computacional. Como o projeto de FMS envolve uma grande quantidade de subsistemas e especificações, a abordagem de controle supervisório modular local **[9]** é bastante vantajosa no sentido de promover maior flexibilidade, menor exigência de processamento computacional e segurança na aplicação. Ao invés de se projetar um único supervisor monolítico que satisfaça todas as especificações, procura-se construir um supervisório para cada especificação **[5]**. Neste caso, deseja-se que os supervisores resultantes sejam modulares, isto é, que a ação conjunta dos supervisores modulares locais (SML) tenha o mesmo desempenho que a do supervisor monolítico **[4]**.

Porém, antes da realização da síntese dos SML, é necessário a definição e representação, como modelo funcional independente - módulo, de cada subsistema e especificações da FMS, com seus respectivos eventos e restrições.No momento da abstração para a definição desses modelos, são tomadas as principais decisões **[2]**. Quanto melhor o planejamento, o gerenciamento dos riscos, a comunicação eficiente e a clareza dos requisitos e escopo do projeto, menor serão os problemas na implantação, a qualidade do projeto ou o custo de correções **[10]**.

A este respeito, este artigo propõe uma abordagem sistemática para projeto de sistemas de manufatura, baseada em técnicas formais equivalentes, com uso do: modelo espiral de engenharia de requisitos **[11]**; método de *Integration Definition for Function Modeling* (IDEF0) **[12]**; a Teoria de Controle Supervisório **[5]**; além da abordagem de Controle Supervisório Modular Local **[9]**; verificação de modelos e supervisórios; e implementação no *State Diagram editor* da ferramenta LabVIEW™.

O uso do formalismo tem por objetivo proporcionar ganhos ordenados e consistentes nas transições entre fases do desenvolvimento do projeto, de tal modo, reduzir a possibilidade de erro e facilitar a implementação de melhorias. Além disso, existe ainda a possibilidade de utilizar essa abordagempara demonstração da sua aplicabilidade, em sistemas de natureza híbrida, nos quais a operação pode ser definida abstratamente como um Sistema a Eventos Discretos (SED) e suas tarefas representadas com seus comportamentos encapsulados e transições ocorrendo devido ao comportamento de variáveis de natureza discreta destes.

Essa sistemática foi aplicada no projeto de um sistema para a movimentação de quatro peças, por meio do uso de manipulador robótico didático, modelo 5150A, do fabricante LabVolt, além do uso de sistema de visão computacional. Os resultados obtidos foram satisfatórios. Por fim, este documento foi organizado na seguinte sequência: a seção 2 é apresentada a abordagem sistemática, a qual é aplicada no determinado projeto conforme seção 3; os resultados são mostrados na seção 4 e conclusões são apresentadas na seção 5.

2 ABORDAGEM SISTEMÁTICA

A proposta de abordagem sistemática apresenta 9 etapas, como mostrado na Figura 1. Inicia com a definição do documento de requisitos (Etapa 1) e dos módulos para cada função independente (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, após aprovação na verificação, 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 - autômatos e SML implementados no ambiente LabVIEW™, com seus respectivos hardware e software (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 do FMS. O quadro 1 apresenta um resumo do método, com o detalhamento das entradas, saídas, atividades e técnicas.

Quadro 1: Resumo da abordagem sistemática (continua).

Etapa	Entrada	Saída	Técnica formal
1. Definir o documento de requisitos	Necessidades do usuário	Documento de requisitos	Modelo espiral de engenharia de requisitos **[11]**
2. Definir e criar os módulos	Documento de requisitos	Módulos assíncronos	Método IDEF0 **[12]**
3. Modelar os autômatos e restrições	Módulos assíncronos; Documento de requisitos	Modelos dos autômatos e das especificações	TCS **[5]**
4. Gerar os modelos do SML	Modelos dos autômatos e das especificações	Modelos do SML	Controle supervisório modular local **[9]**
5. Verificar os modelos dos autômatos e SML	Modelos dos autômatos e das especificações; Modelos do SML; Documento de requisitos	Resultado da verificação de modelos	TCS **[5]** Controle supervisório modular local **[9]**

Quadro 1: Resumo da abordagem sistemática (continuação).

Etapa	Entrada	Saída	Técnica formal
6. Converter e simular os modelos no ambiente LabVIEW™	Modelos dos autômatos e das especificações; Modelos do SML	Algoritmo no ambiente LabVIEW™	*StateDiagram* editor da ferramenta LabVIEW™
7. Integrar os programas computacionais aos modelos no ambiente LabVIEW™	Algoritmo no ambiente LabVIEW™	Algoritmo com integração no ambiente LabVIEW™	Ferramenta LabVIEW™
8. Criar IHM de programação no nível de tarefas no ambiente LabVIEW™	Algoritmo com integração no ambiente LabVIEW™	Algoritmo com integração e IHM no ambiente LabVIEW™	Ferramenta LabVIEW™
9. Testar a operação	Algoritmo com integração e IHM no ambiente LabVIEW™	Resultado do teste de operação	Ferramenta LabVIEW™

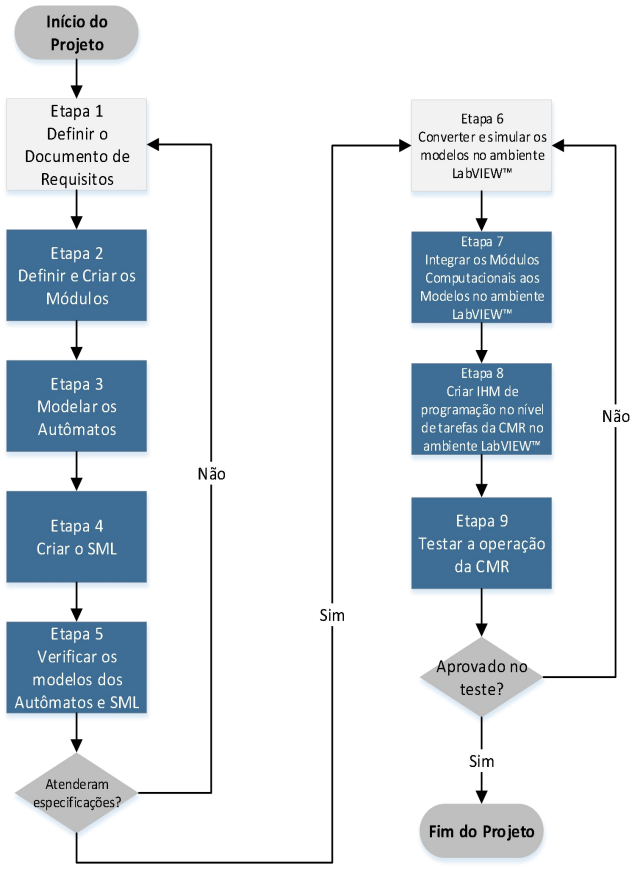

Figura 1: Fluxograma do método proposto 2.
Fonte: Autor.

DOI: 10.25286/repa.v4i3.993

2.1 Etapa 1: Definir o documento de requisitos

Nesta etapa é criado o documento de requisitos que informará a descrição do sistema e as especificações definidas pelo cliente. Será adotado o modelo espiral de geração de documentação de requisitos **[11, 13]**. A seguir é apresentado o resumo das tarefas desta etapa.

- **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, elaboração de lista de possíveis requisitos;
- **Realizar análise e negociação**: Os requisitos levantados na tarefa 1 são analisados e detalhados e daí são definidos os que serão utilizados;
- **Realizar a especificação**: É gerado o documento de requisitos, que deve ter as seguintes seções:
  - Seção 1 (Introdução): Propósito do documento, as definições 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, apresentando as descrições, entradas, saídas e recursos necessários;
  - 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, e tratamento de falhas;
  - Seção 5 (Descrição da interface com o usuário): Apresenta desenhos, figuras ou rascunhos de telas do sistema.
- **Realizar a validação**: Validação (e revisão) do documento de requisitos para garantia de consistência e integridade;
- **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.

2.2 Etapa 2: Definir e criar os módulos

A identificação dos módulos conceituais é realizada nesta etapa, no qual são modelados graficamente utilizando a método IDEF0 **[12]**, por meio do documento de requisitos elaborado na etapa 1. A saída desta etapa, será o diagrama IDEF0, tabela com cada módulo conceitual, que representará os subsistemas, partes integrantes do sistema de manufatura, e suas respectivas variáveis de entradas, saídas, recursos e controle. Para tal, devem-se realizar as tarefas descritas a seguir:

- Identificar e nomear as diversas partes independentes (em relação à funcionalidade), para atendimento do documento de requisitos;
- Representar cada sistema independente (agora, chamado de módulo) de forma gráfica como uma caixa e colocar a descrição de acordo com sua função, utilizando um verbo ou frase verbal e um código de identificação. Sugere o uso dos códigos: $M_i, i = 1, \dots, n$;
- Identificar para cada módulo suas necessidades de sinais e dados de entrada e seus resultados de saída, além dos sinais de controle e os recursos físicos e computacionais necessários para a execução da sua funcionalidade;
- Definir 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 do sistema de manufatura;
- 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;
- 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.

2.3 Etapa 3: Modelar os autômatos e restrições

O objetivo desta etapa é criar os modelos de autômatos de cada módulo conceitual descrito na etapa anterior e das especificações (são as restrições do sistema de manufatura) levantadas na Etapa 1. Para tal, devem-se realizar as tarefas descritas a seguir de acordo com a TCS **[5]**.

- Construir o modelo básico de autômato assíncrono ($G_i, i = 1, \dots, n$) para cada módulo ($M_i, i = 1, \dots, n$) descrito na etapa 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;
- Criar o modelo de autômato para os eventos não controlados;
- Identificar no documento de requisitos as restrições do processo, ou seja, as restrições de coordenação a serem impostas ao sistema;
- Construir o modelo básico do autômato para cada restrição identificada ($R_{gen,i}, i = 1, \dots, n$).

2.4 Etapa 4: Gerar os modelos do 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) **[9]**. É recomendado o uso da ferramenta computacional *Grail* **[14]**. Devem-se realizar as tarefas descritas a seguir.

- Identificar o conjunto de autômatos criados na etapa 3 que representam subsistemas (conjunto de partes) 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 fatia do comportamento global do processo;
- Gerar a planta local $G_{loc,i}, i = 1, \dots, n$ pela composição síncrona do conjunto dos modelos dos autômatos que representam cada subsistema;
- 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;
- 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 $L_m(E_{loc,i}) \subset L_m(G_{loc,i}), i = 1, \dots, n$;
- 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;
- Calcular a componente trim $E_{loc\_trim,i}, i = 1, \dots, n$ para cada especificação local sem estados proibidos ($E_{loc\_sp,i}, i = 1, \dots, n$);
- Calcular a máxima linguagem controlável $S_{loc,i} = supC(G_{loc,i}, E_{loc\_trim,i}), i = 1, \dots, n$.
- 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.

2.5 Etapa 5: Verificar os modelos de autômatos e SML

A verificação dos modelos dos autômatos e SML são realizadas por meio das ferramentas computacionais *Grail* **[14]** e *Uppaal* **[15]**. O *Uppaal* é uma ferramenta de *Model Checking* para sistemas de tempo real. Devem-se realizar as tarefas descritas a seguir. Caso não seja atendida quaisquer das tarefas descritas, retornar para a Etapa 2.

- 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*);
- 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*);

DOI: 10.25286/repa.v4i3.993

- Desenhar os modelos dos autômatos ($G_i, i = 1, \ldots, n$) e dos supervisórios reduzidos locais do SML ($S_{red,i}, i = 1, \ldots, n$) (Utilizar a ferramenta computacional *Uppaal*);
- Eliminar os eventos que geram não coordenação (são os eventos repetidos que estão em *loop* no estado inicial) dos supervisórios reduzidos locais do SML ($S_{red,i}, i = 1, \ldots, n$);
- Verificar o conjunto de modelos dos autômatos e SML a existência de *deadlock* (utilizar a verificação de modelos do *Uppaal*);
- Escrever as especificações do documento de requisitos (etapa 1) em linguagem CTL (utilizar a verificação de modelos do *Uppaal*);
- Verificar o atendimento de cada especificação pelos modelos de autômatos e SML (utilizar a verificação de modelos do *Uppaal*).

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

Os modelos dos autômatos ($G_i, i = 1, \ldots, n$) e SML ($S_{red,i}, i = 1, \ldots, n$) são convertidos para o ambiente computacional LabVIEW™ utilizando a ferramenta *State Diagram Editor*. Para tal, devem-se realizar as tarefas descritas a seguir.

- Criar um *State Diagram*(conforme Figura 2) para cada autômato e SML, desenhando seu respectivo modelo no *State Diagram editor*;
- Criar *leds* indicadores e ligar (utilizar a variável local) aos eventos que indicam sinais internos de todos os *State Diagram*, conforme Figura 2 para cada modelo de autômato e SML;

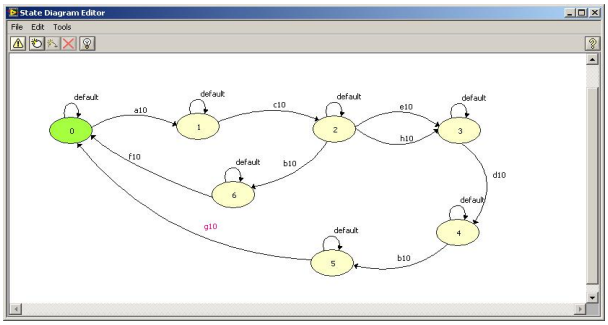

Figura 2: Exemplo de *State Diagram*.
Fonte: Autor.

Criar o algoritmo de comparação, como indicado pela seta na Figura 3 para todos os eventos dos *StateDiagram* de cada modelo de autômato e SML;

- Criar uma variável local para cada evento interno não controlado do modelo de autômato e SML conforme Figura 3;
- Fazer temporizador dentro do *StateDiagram* conforme Figura 3 para cada modelo de autômato e SML;

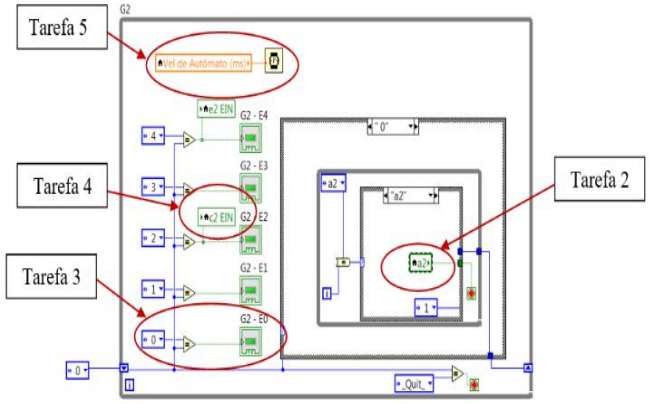

Figura 3: Criação do modelo do autômato.
Fonte: Autor.

- Realizar simulação com variações aleatórias de parâmetros de entrada para verificação de conflitos e erros de programação e o atendimento as especificações do documento de requisitos (Etapa 1), em particular, a Seção que trata do tratamento de falhas.

2.7 Etapa 7: Integrar os programas computacionais aos modelos no ambiente LabVIEW™

Nesta etapa ocorrerá a integração entre os diversos programas computacionaisdo sistema de manufatura ao programa de controle supervisório no ambiente LabVIEW™ desenvolvido nesta abordagem. O programa computacional executará o módulo conceitual definido na etapa 2. Para isso, devem-se realizar as tarefas descritas a seguir.

- Elaborar o programa computacional para cada módulo conceitual no ambiente LabVIEW™, atendendo as necessidades das entradas, saídas, controle e recursos descritos na etapa 2;
- Criar cada módulo conforme apresentado na Figura 4 e Figura 5 no ambiente LabVIEW™, todas as instruções apresentadas nas figuras devem ser implementadas para o correto

funcionamento na integração com *StateDiagram*;

- Implementar o algoritmo no local indicado na Figura 4 como (tarefa 3). Neste local deve ser escrito o programa computacional.

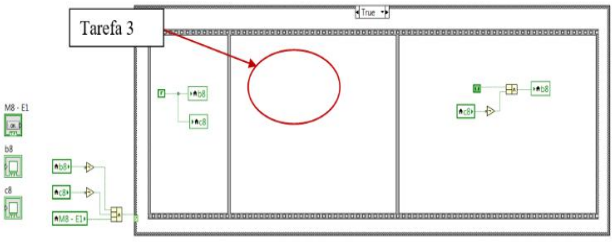

Figura 4: Módulo conceitual estado *true* implementado no LabVIEW™.
Fonte: Autor.

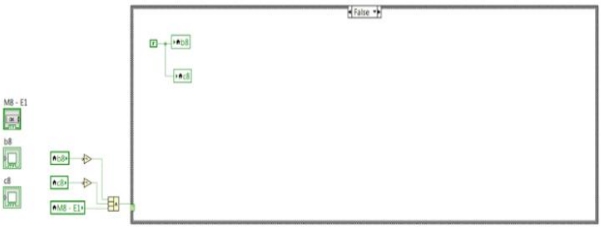

Figura 5: Módulo conceitual estado *false* implementado no LabVIEW™.
Fonte: Autor.

- Testar a execução de cada programa computacional desenvolvido;
- 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 6.

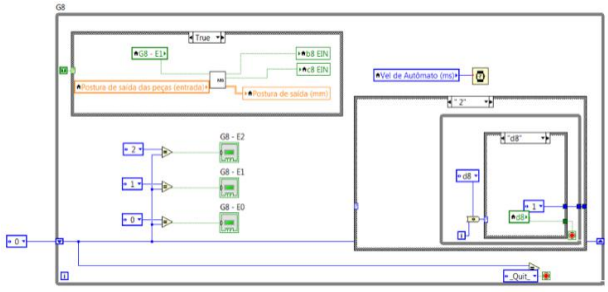

Figura 6: Módulo computacional integrado no ambiente LabVIEW™.
Fonte: Autor.

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

2.8 Etapa 8: Criar IHM de programação no nível de tarefas do FMS no ambiente LabVIEW™

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

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

2.9 Etapa 9: Testar a operação

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

- Identificar todas as ligações físicas entre o computador e os equipamentos;
- Identificar o correto funcionamento da comunicação entre os diversos hardwares e os softwares;
- Realizar as condições iniciais;
- Implementar a programação no nível de tarefas para execução;
- Executar a programação e observar o andamento das tarefas;
- Levantar condições de falhas descritos no documento de requisitos elaborados na Etapa 1;
- Testar a operação com as condições de falhas;

Obtendo sucesso em todas as tarefas propostas nesta etapa, o sistema de manufatura estará simulado, testado e aprovado.

3 ESTUDO DE CASO

A abordagem proposta foi aplicada no projeto de uma célula de manufatura robotizada, tipo de FMS, para seleção e movimentação de quatro diferentes peças, que permite a programação no nível de tarefas. Os recursos físicos e computacionais do FMS, de acordo como quadro 2, foram integrados em uma única plataforma e suas tarefas gerenciadas por um sistema supervisório.

DOI: 10.25286/repa.v4i3.993

Quadro 2: Recursos Físicos do FMS do estudo de caso.

Recurso físico	Detalhamento	Aplicação no FMS
1	Robô industrial	Realizar a manipulação das peças.
2	Sistema de visão computacional	Realizar a identificação as 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.

O desenho esquemático do FMS é apresentado na Figura 7.

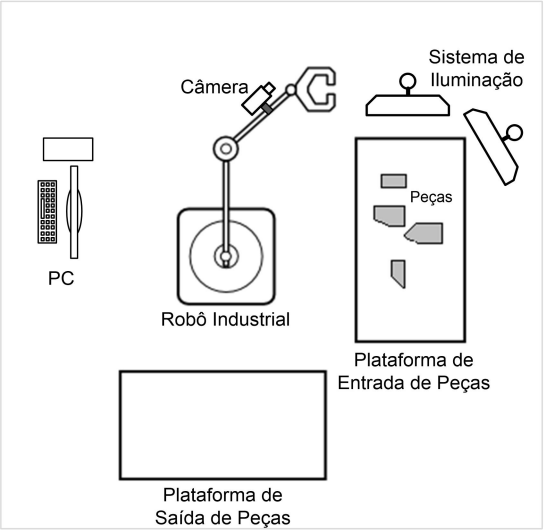

Figura 7. Desenho esquemático do FMS.
Fonte: Autor.

O processo do FMS, 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. Os usuários são operadores de máquinas em ambiente industrial.

O Robô Industrial do FMS 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 8apresenta uma imagem deste robô.

Figura 8. ROBOT 5150-A do fabricante Lab-Volt.
Fonte: Autor.

Esses robôs são encontrados nos laboratórios de universidades e utilizados em diversas aplicações, entre as quais, movimentação de peças. As principais características são apresentadas no quadro 3.

Quadro 3: Principais características.

Item	Descrição	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.

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. As peças são apresentadas na Figura 9.

Figura 9: Peças para movimentação pelo FMS.
Fonte: Autor.

A Figura 10 apresenta a IHM desenvolvida com abstração para programação no nível de tarefas do FMS e o teste de conectividade do sistema.

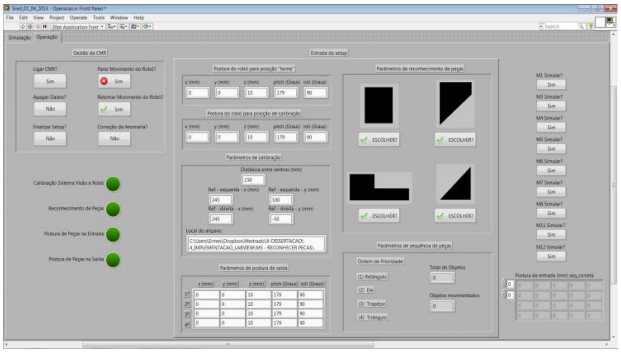

Figura 10. Ambiente para programação da FMS.
Fonte: Autor.

Para testar a abordagem proposta, cinco simulações no FMS foram realizadas, 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 do FMS para seleção e movimentação de duas peças (triângulo e éle);
4. Procedimento de operação do FMS 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.

O FMSrealizou, com sucesso,de forma automática, todas as cinco simulações. O sistema, também, permitiu a indicação visual das fases do processo e, caso houvesse, a existência de falhas. A Figura 11 ilustra o FMS.

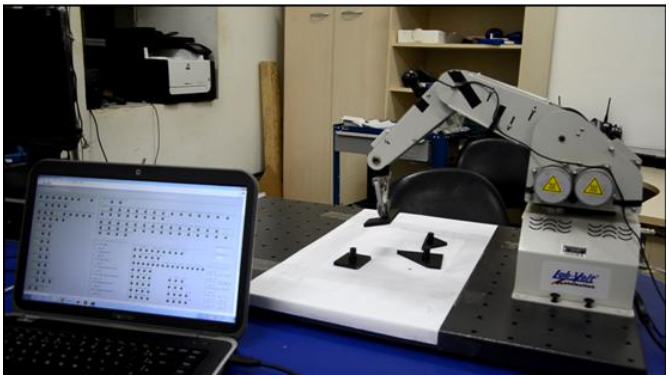

Figura 11: FMS.
Fonte: Autor.

DOI: 10.25286/repa.v4i3.993

Quadro 4. 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 *StateDiagram* 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 (StateDiagram)
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

4 RESULTADOS

A aplicação da abordagem sistemática demonstrou a viabilidade no projeto de um FMS. Foi possível, identificar as delimitações entre as diversas etapas da proposta, representada pela comparação entre os elementos básicos de autômatos com todas as técnicas utilizadas, conforme quadro 4, o que facilitou a identificação de erros e proporcionou a implantação de melhorias, ainda na fase de projeto.

Quadro 5. 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 StateDiagram 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)
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	-	-	-	-

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 da proposta. As relações entre as informações do documento de requisitos e símbolos gráficos das diversas técnicas utilizadas são apresentadas no quadro 5. 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.

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 TCS e SML possibilitou a geração do correto sequenciamento das atividades do FMS pelo emprego de formalismo matemático, a necessidade de não concorrência entre eventos e o caráter modular do sistema.

A verificação de modelos de autômatos e dos supervisórios no *Grail*, proporcionou a descoberta e correção de diversos erros ainda na etapa de projeto do estudo de caso,como exemplos, a existência de *deadlock*, devido a ação bloqueante dos SML nos modelos dos autômatos, eliminados com a melhoria dos autômatos, em particular, do modelo de autômato do robô industrial didático; e a existência de conflitos entre os eventos dos supervisórios reduzidos locais do SML, ou seja, bloqueantes entre si, corrigidos com a alteração conceitual dos subsistemas do FMS 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, responsável pela ação de ligar e desligar a CMR e gerir o processo,tal autômato está presente 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 estudo de caso. 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. Esse problema foi equacionado com a adição de temporizador, valor próximo de 1ms, que permitiu o atraso nas mudanças de eventos entre os diversos *state diagram*.

Neste estudo de caso foram realizados testes de operação do sistema supervisório e da interação com os módulos computacionais, por ação externa de um operador humano na interface homem máquina (IHM), simulando mudanças aleatórias de parâmetros de entrada para identificação de erros. O ambiente simulado do IHM permitiu o acompanhamento do avanço na execução dos eventos do sistema.

Durante a etapa de projeto, com o uso das verificações e simulações, 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 no FMS 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, já que toda a arquitetura foi desenvolvida com software de de uso no chão de fábrica, por exemplo, o ambiente LabVIEW™, reduzindo o custo e diminuindo o tempo de desenvolvimento no processo de concepção conforme motivação deste trabalho. Além disso, os resultados sugerem a possibilidade de utilizar essa abordagem, para demonstração da sua aplicabilidade, em sistemas de natureza híbrida, que tenham condições de serem modularizados e cujas atividades sejam sequenciadas.

5 CONCLUSÕES

Este trabalho introduz um método com a utilização do formalismo único, baseado em autômato, no projeto modular de célula de manufatura robotizada. 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 e aumenta a confiabilidade do desenvolvimento.

A utilização do conceito de projeto modular permite 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 reutilizado em outros projetos. Além de facilitar a abstração e concepção do sistema supervisório conforme o conceito de SML da TCS, e permitir a verificação no atendimento ao comportamento não

DOI: 10.25286/repa.v4i3.993

bloqueante e minimamente restritivo, conforme desejado.

Ademais, aabordagem proposta é aplicável em projetos industriais, mediante uso de plataformas como o LabVIEW™, quando outras abordagens formais têm se mostrado inviáveis em razão do uso da abordagem monolítica para síntese do supervisório único. Como também, demostrou o potencial de aplicabilidade em sistemas de natureza híbrida, por exemplo, em sistemas de controle de tráfego em rodovias, eletrônica de potência, controle de processos químicos e aplicações automotivas.

REFERÊNCIAS

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

[2] COSTA NETO, E. F. **Desenvolvimento de método para projeto modular de célula de manufatura robotizada com programação no nível de tarefas**. 2013. 157 f. Dissertação (mestrado) – Universidade Federal de Pernambuco, Recife, 2013.

[3] ERBE, H.–H. Low Cost Intelligent Automation in Manufacturing. *In:* TRIENNIAL IFAC WORLD CONGRESS, 15., 2002, Barcelona. **Proceeding** [...] Barcelona: 2002.

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

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

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

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

[8] KLEINROCK, L. **Queueing Systems. Volume I: Theory**. Toronto: Wiley-Interscience, 1975. 448 p.

[9] 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**, 2002. 103-110 p.

[10] Project Management Institute. **A Guide to the Project Management Body of Knowledge (PMBOK Guide)**. Newtown Square: PMI, 2013.

[11] BOEHM, B. A Spiral Model of Software Development and Enhancement. *In:* **IEEE Computer**, v. 21, p. 61-72, 5 May 1988.

[12] FIPS PUB 183. **Integration definition for function modeling (IDEF0). Software Standard. Modelling techniques**. FIPS PUB 183, Computer Systems Laboratory National Institute of Standards and Technology, Gaithersburg, 1993.

[13] KOTONYA, G.; SOMMERVILLE, I. **Requirements engineering**: process and techniques. Chichester: Wiley Publishing, 1998. 294 p.

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

[15] BEHRMANN, G.; DAVID, A.; LARSEN, K. G. A tutorial on uppaal: formal methods for the design of real-time systems. *In:* INTERNATIONAL SCHOOL ON FORMAL METHODS FOR THE DESIGN OF COMPUTER, COMMUNICATION, AND SOFTWARE SYSTEMS, 4., 2004, Bertinoro. **Proceeding** […] Bertinoro, Italy: SFM, 2004. 200-236 p.