

MÉTODO iDAV: ANÁLISE DE UM MODELO DE SIMULAÇÃO

João Paulo Barbieri

Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas - *Campus Machado*
Rod. Machado, Paraguaçu, km 3 - Bairro Santo Antônio - Machado/MG - CEP 37.750-000
joao.barbieri@ifsuldeminas.edu.br

Alexandre Ferreira de Pinho

Universidade Federal de Itajubá - *Campus Itajubá*
Av. BPS, 1303 - Bairro Pinheirinho - Itajubá/MG - CEP 37.500-903
pinho@unifei.edu.br

José Arnaldo Barra Montevechi

Universidade Federal de Itajubá - *Campus Itajubá*
Av. BPS, 1303 - Bairro Pinheirinho - Itajubá/MG - CEP 37.500-903
montevechi@unifei.edu.br

RESUMO

Impulsionadas por inovações tecnológicas, transformações significativas ocorreram nos sistemas de produção nos últimos séculos. Considerando as últimas décadas, diversas pesquisas nasceram em torno do termo Indústria 4.0 que possui nove pilares, sendo um deles a Simulação. Em razão do desenvolvimento tecnológico juntamente com a emergência do conceito de Gêmeo Digital, pesquisadores e analistas da área de simulação encontram-se diante de um desafio: produzir modelos computacionais com o máximo de variáveis presentes em um sistema real. Esses modelos computacionais podem ser analisados e aprimorados a partir de uma ótica estrutural. Assim, este artigo apresenta o Método iDAV. Busca-se, por meio desse método, mensurar a complexidade estrutural de modelos de simulação e definir parâmetros metodológicos que irão auxiliar no processo de análise desses modelos. Os detalhes de cada fase e o resultado final da aplicação do Método iDAV serão apresentados e discutidos nesse artigo.

PALAVRAS CHAVE. Modelo de Simulação, Método iDAV, Complexidade.

ABSTRACT

Driven by technological innovations, significant transformations have occurred in production systems in recent centuries. Considering the last decades, several researches were born around the term Industry 4.0, which has nine pillars, one of which is Simulation. Due to technological development along with the emergence of the Digital Twin concept, researchers and analysts in the area of simulation find themselves facing a challenge: producing computational models with the maximum number of variables present in a real system. These computational models can be analyzed and improved from a structural perspective. Therefore, this paper presents the iDAV Method. The aim is to, through this method, measure the structural complexity of simulation models and define methodological parameters that will assist in the process of analyzing these models. The details of each phase and the final result of the iDAV Method will be presented and discussed in this paper.

KEYWORDS. Simulation Model. iDAV Method. Complexity.

SIM - Simulação

1. Introdução

Impulsionadas por inovações tecnológicas, transformações significativas ocorreram nos sistemas de produção nos últimos séculos. Com recursos e esforços, o desenvolvimento de pesquisas continua a acontecer para que avanços e melhorias de nossos atuais arranjos produtivos se concretizem no tempo presente. Assim sendo, nas últimas décadas, diversas pesquisas nasceram em torno do termo Indústria 4.0. Esse termo surge apoiado na integração de nove pilares tecnológicos, a saber: Robôs Inteligentes, Manufatura Aditiva, **Simulação**, Integração Horizontal e Vertical, Internet das Coisas, *Big Data*, Computação em Nuvem, Segurança Cibernética e Realidade Aumentada. No pilar **Simulação**, foco deste artigo, há diversos desafios que precisam ser encarados pelos pesquisadores.

Em um projeto de simulação, é consenso na comunidade científica que um modelo computacional é uma abstração da realidade e, portanto, é inviável inserir todas as variáveis que compõe um sistema real. Contudo, em razão do desenvolvimento tecnológico juntamente com a emergência do conceito de Gêmeo Digital, que são modelos computacionais que têm elevado grau de integração com sistemas reais por meio de diversos dispositivos tecnológicos [Tao e Zhang 2017], pesquisadores e analistas da área de simulação encontram-se diante de um desafio: produzir modelos computacionais com o máximo de variáveis presentes em um sistema real. Esses modelos computacionais, por possuírem elevado nível de detalhamento, podem ser analisados e aprimorados a partir de uma ótica estrutural. Assim, este artigo apresenta o Método iDAV.

Busca-se, por meio desse método, mensurar a complexidade estrutural dos modelos de simulação e definir parâmetros metodológicos que irão auxiliar no processo de mensuração desses modelos. O termo iDAV é o acrônimo das palavras *identificação*, *Definição*, *Análise* e *Verificação* [Barbieri et al. 2023]. Essas palavras representam fases do processo metodológico de mensuração da complexidade estrutural dos modelos computacionais.

Nesse artigo será relatado os detalhes de cada fase e o resultado final da aplicação do Método iDAV em um modelo de simulação. É importante afirmar que o Método iDAV pode ser aplicado simultaneamente ou após a construção do modelo de simulação. Contudo, é recomendado que a aplicação desse método seja realizada de modo simultâneo, porque, assim ocorrendo, os possíveis erros de levantamento de componentes e conexões poderão ser dirimidos.

2. Complexidade de Modelos Computacionais

Os sistemas desenvolvidos pela humanidade estão crescendo continuamente e, intuitivamente, esse fato aumenta a complexidade de tais sistemas. Os exemplos são muitos, como é o caso de empresas de manufatura, logística, saúde e finanças que buscam atender às necessidades da demanda crescente por meio de sistemas de engenharia que combinam o mundo digital com o real. Portanto, não é inesperado o aumento do tamanho e da complexidade dos sistemas [Sarjoughian 2017] em nossa sociedade.

Aplica-se o conceito sistema a qualquer "todo" no qual exista "componentes" em interação [Bertalanffy 1968]. A união de componentes "isolados", as relações entre os componentes e a estrutura de camadas são características fundamentais de um sistema [Forrest 2018]. Sistema, do ponto de vista da Teoria Geral dos Sistemas, é um conjunto de componentes conectados que, dependendo do nível de análise, cada componente pode ser tratado como um sistema composto por outros componentes conectados, e assim por diante, dando a ideia de camadas. Nesse artigo, o termo sistema significa concentração de componentes interdependentes, ligados por conexões em um ambiente de simulação. O conceito de complexidade está intimamente vinculado ao conceito de sistema.

A complexidade é frequentemente interpretada como uma forma de hierarquia em que um sistema complexo é composto por subsistemas inter-relacionados que, por sua vez, possuem seus próprios subsistemas [Sarjoughian 2017]. Esse pensamento sobre um dado sistema vai até chegar no nível mais baixo no qual existem componentes elementares.

O que determina a simplicidade ou a complexidade de uma estrutura é o modo crítico como ela é descrita [Simon 1962]. Na área de simulação, a complexidade está associada ao nível

de detalhe dos modelos computacionais. O nível de detalhe de um modelo computacional refere-se mais ao sistema que esse modelo representa do que o próprio modelo. Em outras palavras, dependendo do escopo adotado pelo projeto de simulação, a adoção de um nível de detalhe está relacionada à avaliação de inclusão de elementos e de suas conexões que são observáveis no sistema em estudo [Brooks e Tobias 1996]. A escolha do nível de detalhe influencia na complexidade do modelo.

Etimologicamente, a palavra complexidade vem do termo latim *complexus* que significa entrelaçado. Para existir complexidade é necessário que haja componentes e conexões entre eles. Intuitivamente, o sistema aparenta ser mais complexo quanto maior for o número de componentes e de conexões entre eles [Bertuglia e Vaio 2005]. Assim, a complexidade de um modelo depende do número de componentes e de variáveis presentes no mesmo [Zeigler et al. 2000].

As propriedades estruturais de um modelo de simulação envolvem componentes, conexões, estruturas lógicas, fluxo de informações, linhas de código, requisitos de *hardware* e *software*. Nesse artigo, a complexidade de um modelo computacional é entendida como um reflexo da quantidade de componentes, do número de conexões entre os componentes e a essência dessas conexões.

3. Método iDAV

O Método iDAV, apresentado nesse artigo, busca mensurar a complexidade estrutural dos modelos de simulação e definir parâmetros metodológicos que irão auxiliar no processo de análise desses modelos. O nome iDAV é o acrônimo dos termos *identificação*, *Definição*, *Análise* e *Verificação* [Barbieri et al. 2023]. Esses termos representam as fases do processo metodológico de mensuração da complexidade estrutural dos modelos computacionais.

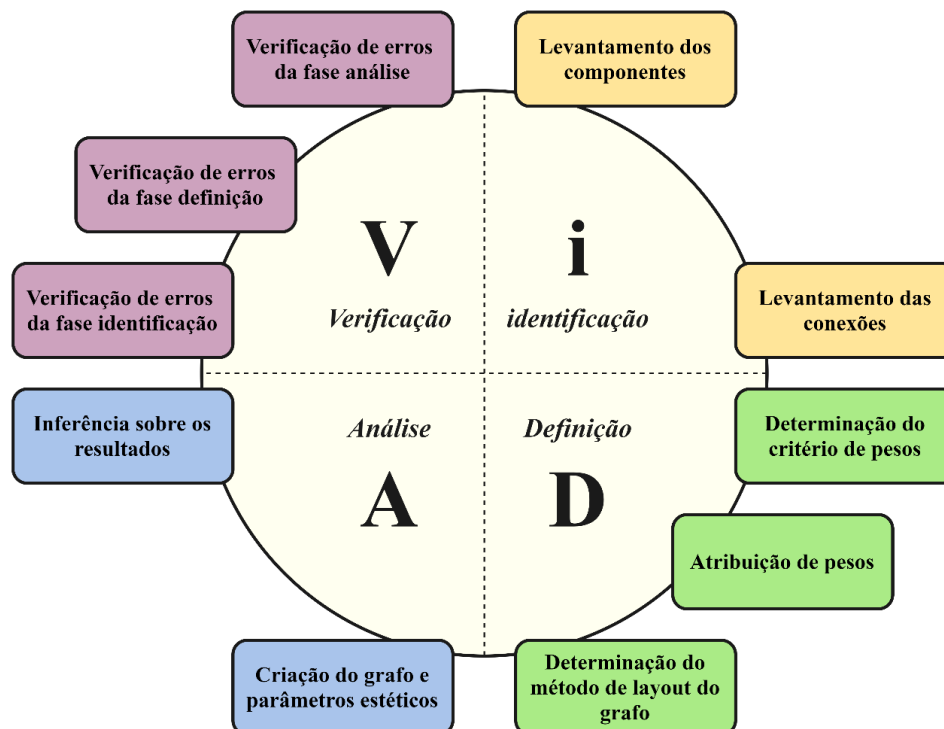


Figura 3.1 - Método iDAV
Fonte: Elaborado pelos próprios autores

Conforme a Figura 3.1, o Método iDAV possui característica cíclica. Ele composto por quatro fases e o processo de mensuração da complexidade de modelos computacionais inicia-se na fase de *identificação* – esse termo começa com letra minúscula para indicar o início do processo. Na sequência, são colocadas em prática as fases *Definição*, *Análise* e *Verificação*.

A fase *identificação* é composta por duas ações que são (i) o **Levantamento dos componentes** e (ii) o **Levantamento das conexões**. Nessa fase, os componentes existentes no modelo computacional são identificados e registrados. O mesmo é feito com as conexões desses componentes, porém é fundamental documentar a relação entre cada componente. O resultado dessa fase se materializa em uma matriz que pode ser construída por meio de planilhas eletrônicas.

Na fase *Definição* existem três etapas que são (i) **Determinação do critério de pesos**, (ii) **Atribuição de pesos** e (iii) **Determinação do método de layout do grafo**. Na etapa (i) é determinado o critério de pesos a ser adotado em cada conexão existente entre os componentes. Em seguida, os pesos são atribuídos na etapa (ii) dentro da matriz. Na etapa (iii) é definido o método de *layout* do grafo. Nesse ponto é importante ressaltar que o Método iDAV contém conceitos oriundos da Teoria dos Grafos. Assim, para ajudar na definição do *layout* do grafo, recomenda-se os trabalhos de [Godehardt 1988], [Di Battista et al. 1994], [Herman et al. 2000], [Kaufmann e Wagner 2001], [Koren 2005], [Wallis 2007], [Jacomy et al. 2014] e [Tamassia 2014].

Na fase *Análise* estão as etapas (i) **Criação do grafo e parâmetros estéticos** e (ii) **Inferência sobre os resultados**. Existem ferramentas que auxiliam na criação e na definição de parâmetros estéticos dos grafos. Uma dessas ferramentas é o *software* Gephi® – *software* de código aberto utilizado para análise e visualização de redes. O Gephi® foi escolhido por ser um *software* livre e por possuir mecanismos que facilitam a comunicação com planilhas eletrônicas. Com o grafo criado, o processo de inferência sobre os resultados se inicia. Essa inferência é feita com conceitos e métodos estatísticos.

A fase de *Verificação* é a última do Método iDAV. Essa fase é composta por três atividades que são (i) **Verificação de erros da fase identificação**, (ii) **Verificação de erros da fase definição** e (iii) **Verificação de erros da fase análise**. Tendo em mente o aprimoramento contínuo do processo de mensuração da complexidade dos modelos computacionais, a fase de *Verificação* foi dividida em três etapas de modo a determinar o foco da ação de melhoria. Assim, nessa fase, o especialista verifica a existência de erros que podem ter ocorrido nas fases anteriores.

4. Aplicação do Método iDAV

O Método iDAV será aplicado em um modelo computacional hipotético. Esse modelo baseia-se em um sistema de manufatura com *layout* em linha. A escolha desse tipo de sistema está relacionada com o conhecimento que os pesquisadores possuem sobre as particularidades gerais desse tipo de sistema.

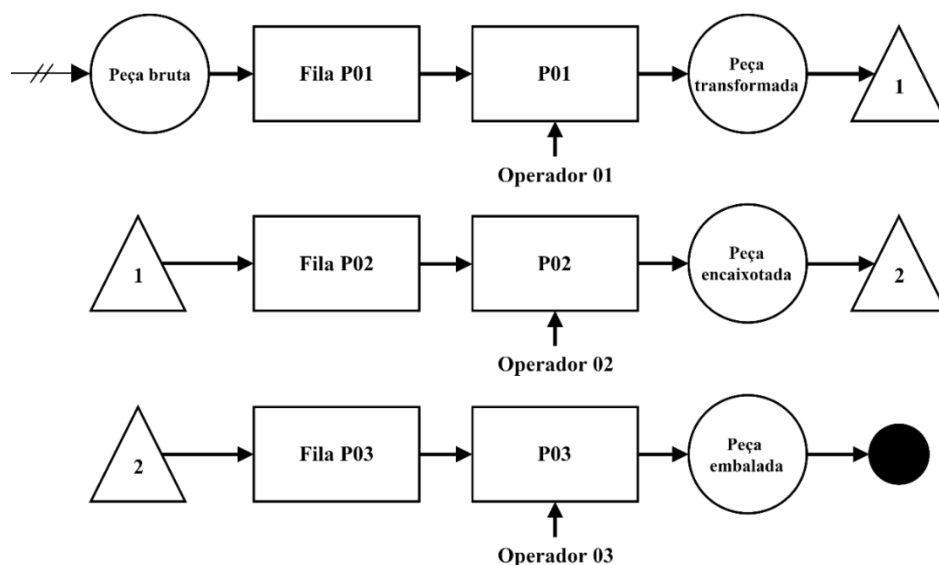


Figura 4.1 - Modelo conceitual do sistema de manufatura

Fonte: Elaborado pelos próprios autores

Fazendo uso do IDEF-SIM (*Integrated Definition Methods – Simulation*), técnica de modelagem desenvolvida por [Leal 2008], o modelo conceitual do sistema de manufatura em linha foi criado, Figura 4.1. O modelo conceitual do sistema hipotético mostra três processos que comumente ocorre em sistemas de manufatura. Esses processos estão em linha e cada um deles possui uma fila. Há, em cada processo, um operador. O Operador 1 recebe a peça bruta e inicia o processo de transformação dessa peça. O Operador 2 recebe a peça transformada e a coloca dentro de uma caixa. O Operador 3 recebe a caixa e a embala para expedição.

A partir do modelo conceitual, o modelo computacional foi desenvolvido. Adotou-se a abordagem Simulação a Eventos Discretos (SED). O *software* utilizado para o desenvolvimento do modelo computacional foi AnyLogic®. O Método iDAV foi sendo aplicado a medida em que o modelo computacional foi sendo criado. Cabe ressaltar que o Método iDAV pode ser aplicado simultaneamente ou após a construção do modelo computacional. Nas seções seguintes, cada fase do Método iDAV será exposta.

4.1 identificação

Para mensurar a complexidade dos modelos computacionais na fase de **identificação**, as seguintes etapas foram executadas: (i) **levantamento dos componentes** e o (ii) **levantamento das conexões**. Na etapa (i) foram contabilizados 31 componentes do modelo computacional. Entende-se por componente todo elemento inserido pelo especialista no ambiente de simulação. Na etapa (ii) foram contabilizadas 36 conexões no modelo computacional em estudo.

Todos esses componentes e suas conexões foram registrados em forma de matriz dentro de uma planilha eletrônica, conforme mostra a Figura 4.2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1																					
2																					
3																					
4																					
5																					
6																					
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					

Figura 4.2 - Matriz de identificação de componentes e conexões

Fonte: Elaborado pelos próprios autores

As conexões existentes dentro de um modelo possuem naturezas distintas. Nota-se, a partir da Figura 4.2, que existem conexões entre componentes nas quais não há a necessidade de escrever linhas de código. Assim, o número 1 (um) representa uma conexão simples (*drag-and-drop*). Por outro lado, quando não se consegue efetuar a ação *drag-and-drop*, o especialista usa o recurso de programação computacional para conectar os componentes dentro do modelo. A próxima seção tratará com mais detalhes sobre esse assunto.

4.2 Definição

Nessa seção, serão tratadas as etapas: (i) **determinação do critério de pesos**, (ii) **atribuição de pesos** e (iii) **determinação do método de layout do grafo**. Durante o procedimento de contagem das conexões existentes entre os componentes de um dos modelos computacionais,

percebeu-se que a praticidade em utilizar a ação *drag-and-drop* em comparação à construção de linhas de códigos era maior. Todavia, essa praticidade é relativa e depende do nível de conhecimento e de experiência do especialista em simulação. Assim, atribuir peso às conexões se baseando no conhecimento e na experiência do especialista traz resultados tendenciosos. Com intuito de eliminar essa tendência, os pesos foram atribuídos com base na Equação 4.1.

$$v(G) = e - n + (2 \cdot p) \quad (4.1)$$

A Equação 4.1 foi desenvolvida por [McCabe 1976] e esse artigo faz uso dela para atribuir pesos às conexões entre os componentes dos modelos computacionais. Essa equação tem o nome de Complexidade Ciclomática (CC) e ela é uma das métricas mais utilizada na mensuração da complexidade [Polančič e Cegnar 2017]. A CC é uma ferramenta que auxilia na medição da complexidade da estrutura de fluxo de controle de um sistema. Em termos práticos, conforme [McCabe 1976], a CC mede o poder de tomada de decisão lógica de um sistema. Na Equação 4.1, o termo $v(G)$ é o número ciclomático e significa quantas decisões lógicas são tomadas no sistema; o termo n representa o número de vértices do sistema; o termo e significa a quantidade de arestas do sistema; o termo p representa a quantidade de blocos de códigos conectados do sistema. Por exemplo, considere a seguinte conjunto de linhas de código:

```
182 | Value_T_Op01 = value;
183 |
184 | if (Value_T_Op01 == 0){
185 |
186 |   Estados_Op01 = "Descansando";
187 |   Horario_Op01 = Op01.getHourOfDay();
188 |
189 | }else if (Value_T_Op01 == 2){
190 |
191 |   Estados_Op01 = "Almocando";
192 |   Horario_Op01 = Op01.getHourOfDay();
193 |
194 | }else{
195 |
196 |   Estados_Op01 = "Trabalhando";
197 |   Horario_Op01 = Op01.getHourOfDay();
198 |
199 | }
```

Esse conjunto de linhas de código representa a conexão do componente **Transicao_1** com os componentes **Value_T_Op01**, **Estados_Op01**, **Horario_Op01** e **Op01**. A Figura 4.3 mostra o grafo desse conjunto de linhas de código.

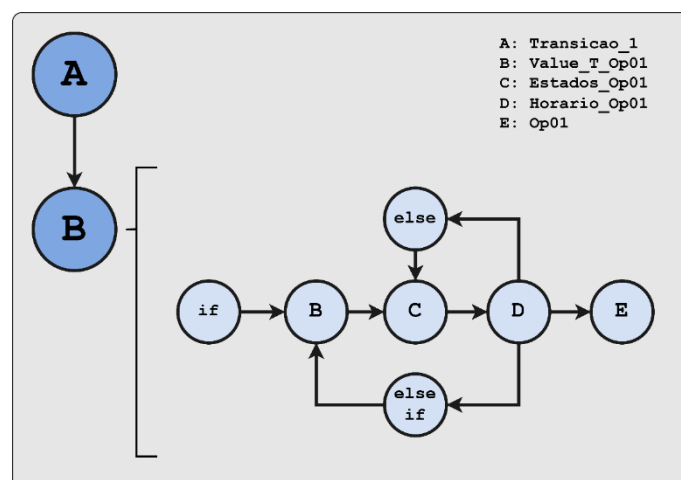


Figura 4.3 - Grafo das linhas de código 182 a 199
Fonte: Elaborado pelos próprios autores

Com base na Figura 4.3, pode-se aplicar a Equação 4.1. O resultado é:

$$v(G) = e - n + (2 \cdot p) = 9 - 9 + (2 \cdot 2) = 4 \quad (4.2)$$

Com a Figura 4.3 e a Equação 4.2 fica claro a existência de dois blocos de código. Sendo a interseção, a variável **Value_T_Op01** coleta o valor do calendário **Transicao_1**; esse é o primeiro bloco. O valor desse bloco é testado pela lógica *if – else if – else* que verifica qual o valor da variável **Value_T_Op01**, e determina as ações para as variáveis **Estados_Op01** e **Horario_Op01**; esse é o segundo bloco. Portanto, de modo simplificado, a função **p** da Equação 4.1 representa o número de sub-rotinas existentes em um conjunto de códigos. Cabe salientar que a adoção de sub-rotinas é uma decisão técnica feita pelo especialista em simulação.

A aplicação da CC ocorreu para todas as linhas de código existentes no modelo computacional em estudo. Nesse modelo computacional, as conexões do tipo *drag-and-drop* receberam peso igual a 1, como mostra a Figura 4.2, seguindo a lógica da ferramenta CC.

Após as etapas (i) e (ii) da fase **Definição**, iniciou-se a etapa (iii) que trata da determinação do método de *layout* do grafo. Com base nos trabalhos de [Godehardt 1988], [Di Battista et al. 1994], [Herman et al. 2000], [Kaufmann e Wagner 2001], [Koren 2005], [Wallis 2007], [Jacomy et al. 2014] e [Tamassia 2014], os principais métodos de *layout* de grafos são: *layout em árvore*, *layout em camadas*, *layout circular*, *layout baseado em arco*, *layout de dominância*, *layout espectral*, *layout baseado em força* e *layout ortogonal*. Nesse artigo, foi adotado o *layout baseado em força*. A escolha teve como base a ênfase gráfica dada por esse *layout* que mostra o grau de importância de cada componente dentro do sistema estudado. Assim, esse *layout* vai auxiliar no processo de análise estrutural do modelo computacional. Na próxima seção, serão descritas as etapas da fase **Análise**.

4.3 Análise

A fase **Análise** possui duas etapas: (i) **criação do grafo e parâmetros estéticos** e (ii) **inferência sobre os resultados**. Assim, iniciou-se o processo de criação do grafo. Representando o modelo computacional, o grafo foi construído a partir de três dados: componentes, conexões e natureza das conexões. Para construir os grafos, três procedimentos foram necessários. O primeiro foi a criação de uma matriz representando o número de componentes, as conexões entre esses componentes e a natureza dessas conexões. O segundo procedimento foi a transferência da matriz de cada modelo computacional para o *software* Gephi® – *software* de código aberto utilizado para análise e visualização de redes. O terceiro foi o procedimento estético de configuração das cores do grafo.

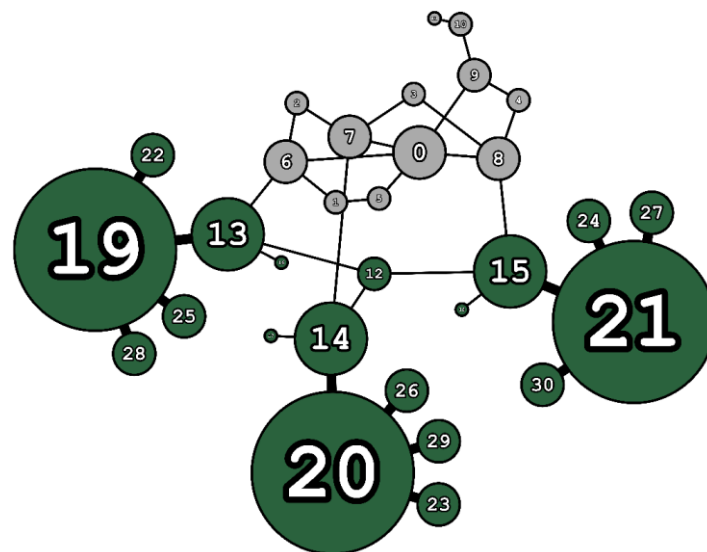


Figura 4.4 - Grafo de estrutura do modelo computacional
Fonte: Elaborado pelos próprios autores

Conforme Figura 4.4, a ênfase do grafo de estrutura está na estrutura do modelo computacional na qual é possível separá-la em dois setores: Fluxo de Processo (FP) e Comportamento da Entidade (CE). O setor FP e CE estão destacados, por essa ordem, em cinza e verde. O FP representa os postos de trabalho em que os operadores estão instalados. O CE são os operadores e suas rotinas de trabalho, considerando entrada e saída.

Visualmente, existem componentes cujo diâmetro é maior. O diâmetro de um dado componente é alterado conforme a quantidade de conexões nele presente e os pesos atribuídos a essas conexões. Neste ponto, ao representar um modelo computacional através de grafos, é possível identificar quais são os componentes e as conexões essenciais dentro do modelo. Por exemplo, na Figura 4.4 destacam-se os componentes 19, 20 e 21. Esses componentes são os eventos relacionados as rotinas de trabalho de cada operador, ou seja, entradas e saídas do posto de trabalho.

Assim, a partir do grafo que retrata o modelo computacional, o especialista toma ciência sobre a estrutura de seu modelo computacional. Com essa ciência é possível analisar a relevância dos componentes e conexões, e aprimorar pontos críticos do modelo computacional.

4.4 Verificação

A fase **Verificação** contém três etapas: (i) **verificação de erros na fase identificação**, (ii) **verificação de erros na fase definição** e (iii) **verificação de erros na fase análise**. O Método iDAV funciona de maneira cíclica, visando o aprimoramento contínuo do processo de mensuração da complexidade dos modelos computacionais. Dessa forma, houve três ciclos de verificação. Esses ciclos, além de eliminar erros encontrados nas fases anteriores, ajudaram a aprimorar os modelos computacionais de modo a torná-los mais enxutos em termos de linhas de códigos e de componentes.

Na próxima seção, alguns testes de sensibilidade foram feitos para identificar os efeitos visuais, tanto nos grafos de estrutura quanto nos de essência, quando se acrescenta elementos gráficos (3D) e quando se acrescenta mais postos de trabalho na linha de produção.

5. Testes de sensibilidade

Um dos objetivos dessa seção é reaplicar o Método iDAV e analisar os efeitos gerados pela adição de elementos 3D. Em projetos de simulação, a adição de elementos 3D é interessante, pois melhora a comunicação com os *stakeholders*. Outro objetivo dessa seção é analisar os efeitos gerados pelo aumento dos postos de trabalho. Até o momento, um sistema de manufatura com *layout* em linha foi utilizado e esse sistema possui três postos de trabalho. O que ocorre se esse sistema triplicar?

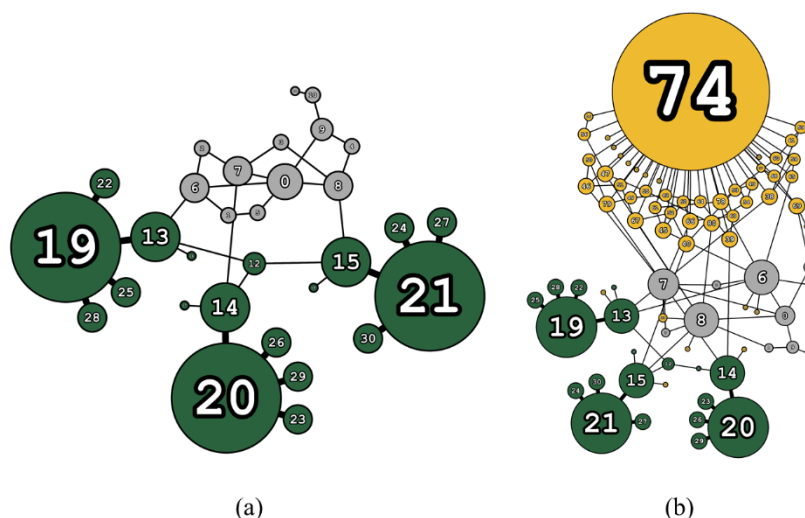


Figura 5.1 - Grafos de estrutura do modelo computacional: adição dos elementos 3D
Fonte: Elaborado pelos próprios autores

No total, foram adicionados 51 elementos 3D. Nota-se na Figura 5.1 que o elemento 74 se destaca. Tal destaque acontece em razão da concentração de conexões. No total, são 42 conexões simples e isso explica o tamanho de seu diâmetro. O elemento 74 retrata o componente *window3d* e é dentro desse componente que os elementos 3D estão. Cabe esclarecer que esses elementos 3D retratam graficamente a planta produtiva com seus postos de trabalho, operadores e a transformação da matéria-prima em produto acabado.

O processo de adição dos elementos 3D consome uma quantidade considerável de tempo de modelagem, mas esse processo gerou modelos computacionais mais descritivos. A questão visual é importante, pois ajuda até mesmo no processo de verificação. No caso desse artigo, ao adicionar os elementos 3D, toda a estrutura lógica dos componentes ficou explícita no momento da simulação.

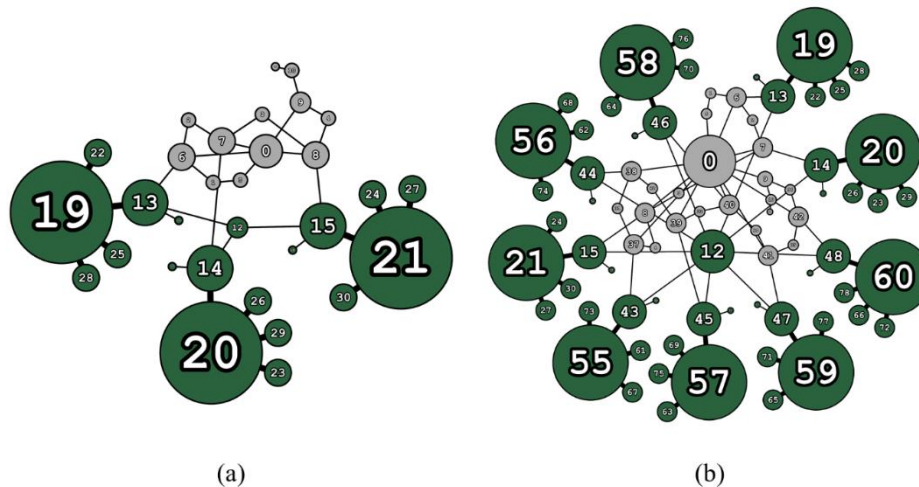


Figura 5.2 - Grafos de estrutura do modelo computacional: adição dos elementos 3D
Fonte: Elaborado pelos próprios autores

Ao triplicar os postos de trabalho, passando de 3 postos para 9, os grafos de estrutura ganham uma forma estrelada. O setor Fluxo de Processo (FP) encontra-se no centro dos grafos de estrutura e o setor Comportamento da Entidade (CE) está nas extremidades, facilitando a análise sobre os componentes mais influentes dentro dos modelos computacionais.

O Método iDAV, a partir dos testes de sensibilidade, é aplicável em contextos em que o projeto de simulação aumenta o nível de detalhamento e escopo. Isso permite ao especialista em simulação o uso do Método iDAV independente das alterações que o projeto de simulação possa sofrer.

6. Conclusão

Um modelo computacional é uma abstração da realidade e a adição de todas as variáveis que regem o sistema real é um desafio. Esse desafio aumenta em razão do desenvolvimento tecnológico juntamente com a emergência do conceito de Gêmeo Digital. Portanto, é tendência de que os projetos de simulação tenham escopo e nível de detalhamento elevados. O Método iDAV, nesse contexto, surge para auxiliar os especialistas em simulação a analisar estruturalmente seus modelos computacionais, identificando os componentes e conexões essenciais dos modelos. Além disso, o Método iDAV também ajuda a tornar os modelos computacionais mais enxutos evitando redundâncias em termos de programação e modelagem. Como mostrado nos testes de sensibilidade, o Método iDAV acompanha a evolução dos modelos computacionais no decorrer do projeto de simulação. Por fim, recomenda-se que o Método iDAV seja aplicado de modo simultâneo à construção do modelo computacional, porque, assim ocorrendo, os possíveis erros de levantamento de componentes e conexões poderão ser dirimidos. O ideal, e aqui fica uma sugestão para uma futura pesquisa, seria que todo *software* de simulação possuisse o Método iDAV

integrado em suas ferramentas de análise. Desse modo, o especialista em simulação poderia visualizar a construção de seus modelos computacionais e aprimorá-los utilizando apenas um *software*.

Agradecimentos

Os autores desse artigo gostariam de expressar gratidão ao Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - IFSULDEMINAS, à Universidade Federal de Itajubá - UNIFEI, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais - FAPEMIG pelo apoio concedido a essa pesquisa científica.

Referências

Barbieri, J. P., Pinho, A. F. e Montevechi, J. A. B. (2023). Autonomous Entities: a hybrid model and its effects. *International Journal of Simulation Modelling*, v.22, n.2, p.187-198.

Bertalanffy, L. (1968). *General Systems Theory*. New York: Braziller.

Bertuglia, C. S. e Vaio, F. (2005). *Nonlinearity, Chaos, and Complexity: The Dynamics of Natural and Social Systems*. 1. ed. New York: Oxford University Press.

Brooks, R. J. e Tobias, A. M. (1996). Choosing the Best Model: Level of Detail, Complexity, and Model Performance. *Mathematical and Computer Modelling*, v.24, n.4, p.1-14.

Di Battista, G., Eades, P., Tamassia, R. e Tollis, I. G. (1994). Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, v.4, p.235-282.

Forrest, J. Y. (2018). *General Systems Theory: Foundation, Intuition and Applications in Business Decision Making*. Switzerland: Springer.

Godehardt, E. (1988). *Graphs as structural models: the application of graphs and multigraphs in cluster analysis*. 4 ed. Wiesbaden: Vieweg.

Herman, I., Melançon, G. e Marshall, M. S. (2000). Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, v.6, n.1, p.24-43.

Jacomy, M., Venturini, T., Heymann, S. e Bastian, M. (2014). ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLOS ONE*, v.9, n.6, pp.1-12.

Kaufmann, M. e Wagner, D. (2001). *Drawing Graphs: Methods and Models*. Berlin: Springer.

Koren, Y. (2005). Drawing Graphs by Eigenvectors: Theory and Practice. *Computers and Mathematics with Applications*, v.49, p.1867-1888.

Leal, F. (2008). Análise do efeito interativo de falhas em processos de manufatura através de projeto de experimentos simulados. 2008. 237 p. *Tese (Doutorado em Engenharia Mecânica)* – Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, Guaratinguetá, SP.

McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, v.2, n.4, p.308-320.

Polančič, G. e Cegnar, B. (2017). Complexity metrics for process models – A systematic literature review. *Computer Standards & Interfaces*, v.51, pp.104-117.

Sarjoughian, H. S. (2017). Restraining complexity and scale traits for component-based simulation models. In: *Winter Simulation Conference, Proceedings...* Las Vegas, NV, USA.

Simon, H. A. (1962). The Architecture of Complexity. *Proceedings of the American Philosophical Society*, v.106, n.6, p.467-482.

Tamassia, R. (2014). *Handbook of Graph Drawing and Visualization*. Boca Raton: CRC Press.

Tao, F. e Zhang, M. (2017). Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. *IEEE Access*, v. 5, p. 20418-20427.

Zeigler, B. P., Praehofer, H. e Kim, T. G. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2. ed. San Diego: Academic Press.

Wallis, W. D. (2007). *A beginner's guide to graph theory*. 2 ed. Boston: Birkhäuser.