Desenvolvimento de uma Ferramenta de Software para Estudos dos Efeitos da Modernização e Ampliação da UHE de Tucuruí

Í. F. Di Paolo[1], T. D. Cordeiro[1], J. A. S. Sena[5], M. C. P. Fonseca[1], W.Barra Jr. [1], J. A. L. Barreiros [1], O. F. Silva[2], É. S. Rodrigues[3], A. A. B. Pardauil[4], D. C. Moreira[4]

***Resumo*– O desenvolvimento de *software* para simulação dinâmica de sistemas elétricos de potência exige um estudo amplo e complexo de diversas áreas da engenharia de *software* para sua modelagem e implementação. Este artigo apresenta o os resultados obtidos no desenvolvimento de uma ferramenta de *software* para simulação dinâmica de sistemas elétricos de potência, com base em modelagem orientada a objeto de geradores, turbinas, rede elétrica e sistemas de controle de unidades hidrogeradoras. O trabalho propõe um padrão de projeto orientado a objetos para mapeamento entre diagrama de blocos e o paradigma orientado a objetos baseado no *Factory Method* em conjunto com o *Facade* (*design pattern* da GoF), permitindo assim maior flexibilidade aos projetistas e acelerando o desenvolvimento de aplicativos mapeando a dinâmica dos blocos em classes especializadas. Para avaliar o desempenho do simulador desenvolvido, com relação a reprodução fidedigna de fenômenos dinâmicos observáveis na UHE de Tucuruí, foram efetuados estudos de comparação entre as respostas dinâmicas fornecidas pelo simulador com as respostas dinâmicas coletadas em testes de campo realizados na UHE de Tucuruí. Os resultados mostram o bom desempenho da ferramenta desenvolvida.**

***Palavras-chave* — Sistemas de potência, simulação dinâmica, padrões de projeto, *factory method*, *facade*, técnicas orientadas a objetos.**

I. Introdução

A atividade de simular Sistemas Elétricos de Potência (SEP) é essencial para que o engenheiro possa avaliar e planejar atividades de expansão de operação e o comportamento do sistema elétrico diante de eventos como perdas de linhas de transmissão, variações de carga, curtos-circuitos, dentre tantos outros.

Em um ambiente virtual podem-se construir modelos físicos que representem um sistema real com a diversidade de detalhes demandados pelo estudo a ser realizado no sistema. Este artigo discute a estratégia de padrão de projeto *Factory Method* em conjunto com o *Facade* para a modelagem de sistemas físicos que possam ser representados em diagramas de blocos. Assim, faz-se um mapeamento entre os paradigmas de modelagem em blocos e orientado a objetos e posteriormente usa-se a característica de encapsulamento da orientação a objetos para abstrair a dinâmica interna de sistemas complexos.

O artigo está organizado da seguinte forma: na 2ª seção é apresentada uma visão geral dos padrões de projeto orientado a objetos e sua importância na construção de *frameworks*. Na 3ª seção, os padrões *Factory Method* e *Facade* são apresentados com maiores detalhes, e suas aplicações na implementação de um *framework* de sistemas de potência é detalhada. A 4ª seção apresenta um estudo de caso com os resultados do *framework* construído com o uso destes conceitos. A 5ª seção apresenta os resultados obtidos e a 6ª seção a conclusão do estudo.

II. Padrões de Projetos de Frameworks

A. Visão geral

Um *framework* é um conjunto de classes cooperativas que implementam mecanismos que são essenciais para um domínio específico de problemas. Um programador pode criar funcionalidades novas no domínio do problema que está trabalhando estendendo as classes do *framework*. Já um *framework de aplicação* é um conjunto de classes que implementa serviços comuns a certo tipo de aplicação [8].

Este artigo relata os avanços obtidos na pesquisa do *framework* BCSSD (Biblioteca de Classes para Simulação de Sistemas Dinâmicos) apresentado em [14] e ampliado em [4] e [5] utilizando estratégias de padrões de projeto GoF (apresentadas em [6] e [7]).

Um padrão de projeto (ou *design pattern*) é uma regra geral para a modelagem e implementação de projetos orientados a objetos [8]. Padrões de projeto também podem ser definidos como itens de uma espécie de catálogo de problemas recorrentes que podem ser resolvidos de maneira sistematizada.

Pela grande quantidade existente de padrões de projeto, este catálogo pode ser organizado de diversas formas. As formas mais convencionais de organização são pelo seu escopo e pelos propósitos gerais [7].

Organização pelo escopo significa dizer se o padrão é aplicado primariamente a classes ou a objetos. E pelos propósitos gerais, os padrões podem ser organizados como:

- Criacionais: envolvem a criação de instâncias de objetos, os quais fornecem alguma maneira de desconectar o cliente (responsável pelas requisições) dos objetos a partir dos quais serão geradas as instâncias.
- Estruturais: permitem a organização de classes ou objetos em estruturas maiores.

[1] UFPA - Programa de Pós-Graduação em Engenharia Elétrica da UFPA
[2] UFPA - Faculdade de Engenharia Elétrica da UFPA
[3] UFPA - Faculdade de Engenharia de Computação da UFPA
[4] Eletronorte – Regional de Produção Hidráulica
[5] Eletronorte – Centro de Tecnologia

- Comportamentais: preocupam-se com a forma como as classes e objetos interagem e com a distribuição de responsabilidades.

B. Factory Method

O padrão de projeto *Factory Method* possui escopo nas classes, não nos objetos, e é criacional. Ou seja, o cliente responsável por requisitar uma de suas instâncias não está interessado em detalhes da dinâmica interna das implementações, e sim em características gerais.

Como está centrado nas classes, o *Factory Method* define uma interface para criar um objeto, mas permite decidir qual classe instanciar, permitindo a uma classe deferir a instanciação para subclasses [6]. É, basicamente, uma "fábrica de métodos", cujos algoritmos específicos podem ser encapsulados em classes especializadas, enquanto uma classe maior agrega funcionalidades gerais.

No mapeamento entre os diagramas de blocos e a orientação a objetos proposto no presente trabalho, a classe que agrega as funcionalidades gerais é a própria representação do bloco, enquanto as classes especializadas são as que representam os elementos físicos do sistema de potência.

O Diagrama de Classes que representa a estrutura para aplicação do *Factory Method* é apresentado no diagrama na Figura 1. A classe abstrata Produto define a interface de objetos para a criação da fábrica de objetos. *ProdutoConcreto* implementa a interface ou classe abstrata *Produto*. *Criador* é a classe abstrata em que a fábrica de métodos é declarada com os métodos principais que serão usados pela *ProdutoConcreto*. E, *CriadorConcreto* se sobrepõe à fábrica de métodos *Criador* para retornar à *ProdutoConcreto*. *Criador* depende de suas subclasses para definir a fábrica de métodos para retornar uma instância adequada de um *ProdutoConcreto*.

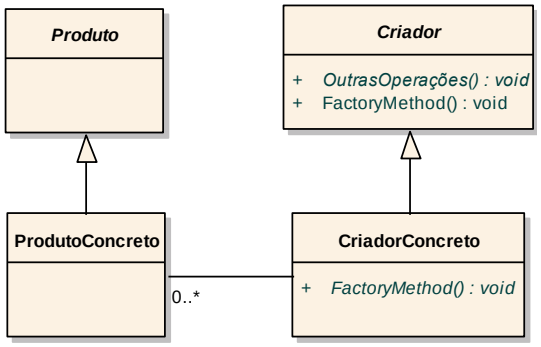

Figura 1. Diagrama de Classes para implementação do Padrão de Projeto *Fatory Method*. Fonte: Adaptado de [7], p. 108.

C. Facade

O padrão de projeto *Facade* possui escopo nos objetos, não nas classes, e é estrutural. Ou seja, permite a organização das classes ou objetos em estruturas maiores, facilitando o acesso de classes clientes a uma diversidade de funcionalidades em diferentes classes.

"O padrão *Facade* fornece uma interface unificada para um conjunto de interfaces em um subsistema. A fachada define uma interface de nível mais alto que facilita a utilização do subsistema" [6]. A Figura 2 apresenta o problema da falta de uso do Facade, em que se tem classes clientes que utilizam serviços de diversas classes. Dependendo da quantidade de serviços e clientes disponíveis, pode-se tornar muito trabalhoso para o programador realizar manutenções ou até expandir sistemas baseados nesta arquitetura.

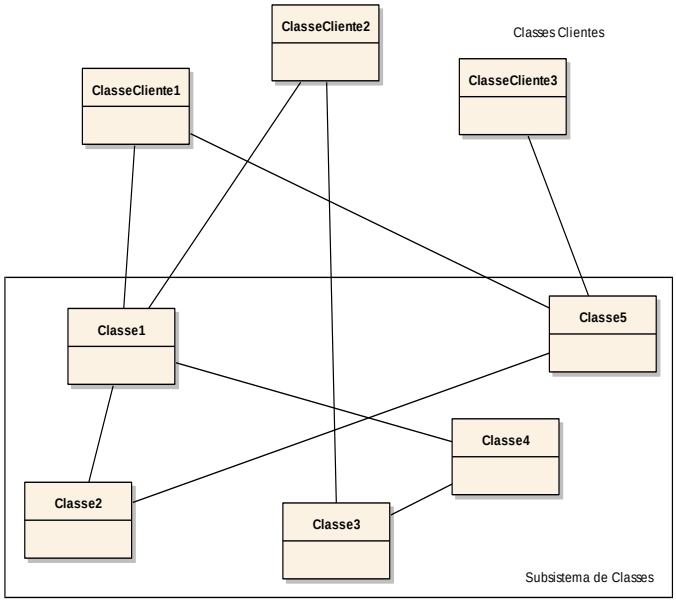

Figura 2. Diagrama de Classes enfocando o problema da falta do Padrão *Facade*. Fonte: Adaptado de [7], p. 185.

A Figura 3 já apresenta o enfoque com a presença de uma classe de "fachada" para servir de interface entre os clientes e as diversas classes que fornecem os serviços, abstraindo das classes clientes os diversos formatos de implementação de suas funcionalidades. A classe *Facade* sabe quais as responsabilidades de cada subsistema, e é responsável por enviar as requisições de cada cliente ao respectivo subsistema objetos. As classes do subsistema são as que implementam cada funcionalidade, não tendo conhecimento da fachada, não guardando nenhuma referência dela ou de algum cliente [7].

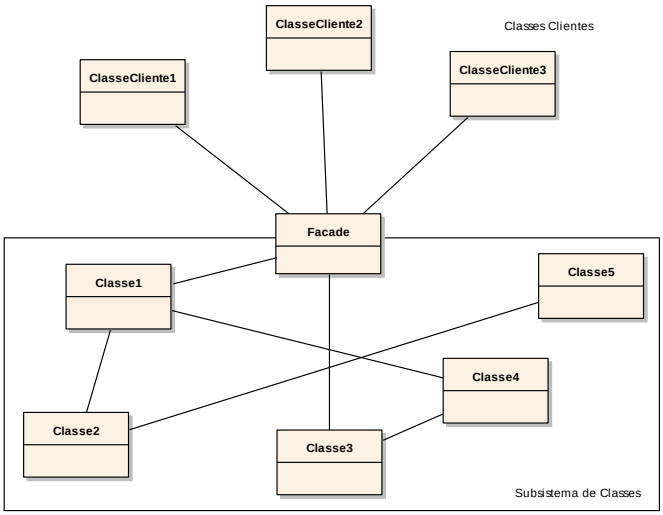

Figura 3. : Diagrama de Classes para implementação do Padrão de Projeto *Facade*. Fonte: Adaptado de [7], p. 185.

III. Aplicando o Padrão de Projeto *Factory Method* em Sistemas Elétricos de Potência

A. Visão geral do mapeamento entre paradigmas de modelagem

O mapeamento entre paradigmas de modelagem é muito comum em sistemas de engenharia. Entre os paradigmas de modelagem e programação, por exemplo, existe o procedimental, estrutural, orientado a objetos, orientado a aspectos e o relacional.

A própria matemática faz uso deste artifício para simplificar a complexidade de cálculos para se chegar a uma solução ou melhorar a análise dos resultados de um problema, como convertendo sistemas no domínio do tempo para a frequência [12].

O fato é que esse mapeamento é sempre necessário porque, para um determinado contexto, um paradigma pode ser mais apropriado que outro. As razões são várias, como a popularidade de um paradigma, sua simplicidade de implementação, entre outros.

Em sistemas elétricos de potência e em estudos de controle, é comum a representação de sistemas dinâmicos em diagramas de blocos, com a dinâmica representada tanto no domínio do tempo quanto no da frequência ([12] e [9]). Para a implementação em linguagens de programação modernas (C++, Java, C#, entre outras), o paradigma mais utilizado é o de orientação a objetos ([2], [14], [11], [4], [5]).

B. Mapeamento entre Diagramas de Blocos e a Orientação a Objetos

Diagramas de blocos são representações gráficas que permitem representar as relações entre cada elemento de um sistema [12]. Pode-se construir o diagrama global do sistema conectando-se os diversos módulos conforme suas relações; e a possibilidade de ser utilizado pelos engenheiros para representar um número muito grande de sistemas.

Algumas ferramentas de desenvolvimento já possuem um ambiente para programação baseado em diagramas de blocos, como o Simulink (Matlab, 2008). Porém, quando há a necessidade de se programar em ambientes orientado a objetos, esbarra-se sempre na escolha da metodologia a ser utilizada: como parametrizar os parâmetros dos blocos? Como aplicar a integração numérica? Como representar os sistemas dinâmicos?

Neste contexto, a BCSSD ([14], [4] e [5]) aplica o padrão *Factory Method* para deixar o programador livre para modelar qualquer bloco que queira, simplesmente escrevendo nos códigos-fontes as equações dinâmicas que caracterizam o bloco, deixando encapsulados todos os demais "comportamentos" esperados para um bloco. Assim, pode-se representar, através de uma linguagem de programação orientada a objetos, qualquer sistema que possa ser representado em diagrama de blocos.

O diagrama de classes na Figura 4 apresenta a estrutura de aplicação do padrão *Factory Method* na BCSSD, com as devidas associações apresentadas no diagrama na Figura 1, para efetuar o mapeamento. Todo bloco possui uma estrutura básica codificada na classe *Bloco*, cujos elementos do sistema de potência são codificados herdando estas características básicas e implementando suas funcionalidades específicas.

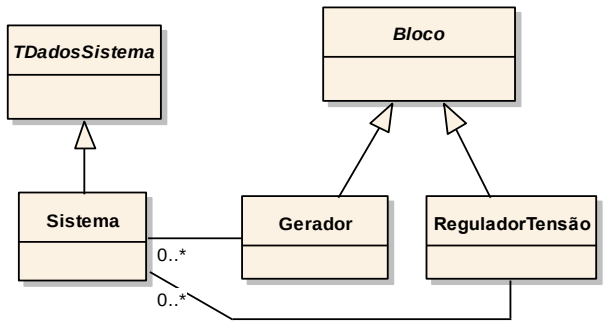

Figura 4. Arquitetura do Padrão de Projeto *Factory Method* para mapeamento entre diagrama de blocos e orientação a objetos.

O diagrama na Figura 5 apresenta alguns detalhes das classes responsáveis por armazenar a estrutura do diagrama de blocos e executar a simulação. *TDadosSistema* é uma classe abstrata que contêm os parâmetros do sistema. A BCSSD foi construída em C++, logo, suas coleções são representadas por ponteiros, como nos atributos *Geradores* e *ReguladoresTensão*. Todos os elementos do sistema são representados por atributos. Além deles, demais estruturas utilizadas para execução do fluxo de carga e da própria simulação também são declaradas, como a matriz admitância de rede (*Y*) e a sua representação reduzida (*Yred*). Os métodos que implementam as faltas no sistema também podem ser implementados aqui, como *SimularCurtoCircuito(...)*, usado para simular curtos-circuitos trifásicos e *SimularDegrauReferencia(...)* utilizado para simular um degrau na referência dos reguladores de tensão. A classe *Sistema* é uma classe concreta que herda estas características, possui efetivamente a representação do sistema orientado a objetos devidamente mapeado através dos diagramas de blocos, e executa as simulações de acordo com os parâmetros informados.

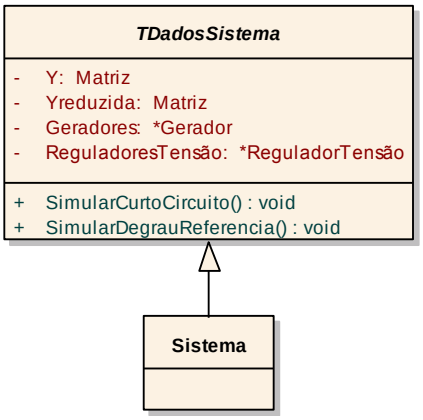

Figura 5. Detalhes das classes principais que armazenam a estrutura do diagrama de blocos para executar as simulações.

A "fábrica de métodos" é, então, apresentada com alguns detalhes no diagrama na Figura 6. A classe *Bloco* possui os atributos como os pontos de entrada e saída dos dados, e os

métodos abstratos para implementar suas funcionalidades, como obter os valores de entrada e de saída dos respectivos pontos, efetuar as conexões com outros blocos, conectando entradas de um bloco na saída de outro, por exemplo. Além disso, os métodos para o cálculo das derivadas dos blocos e das saídas das variáveis de estado são declarados, mas implementados nas classes especializadas.

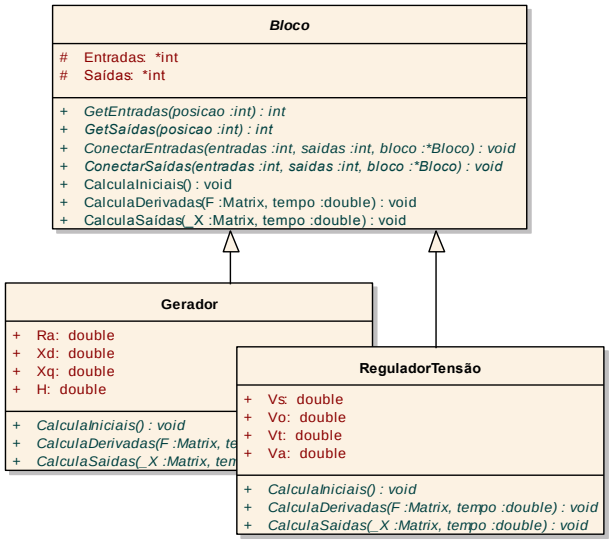

Figura 6. Detalhes das classes que mapeiam o diagrama de blocos para sistemas de potência.

As classes especializadas, que herdam da classe *Bloco*, são as que representam os elementos do sistema de potência. O diagrama na Figura 6 mostra duas dessas classes com alguns detalhes: *Gerador* e *ReguladorTensão*. Nessas classes são declarados itens específicos de cada bloco. Geradores, por exemplo, têm resistência de armadura (*Ra*), reatâncias de eixos (*Xd* e *Xq*), momento de inércia (*H*), entre outros. Uma maneira simples de representar Reguladores de Tensão é através de um bloco de primeira ordem, com um ganho (K_A) e uma constante de tempo (T_A).

São nessas classes especializadas que os métodos *CalculaDerivadas(...)*, utilizado para calcular as variáveis de estado em cada passo de integração, *CalculaIniciais(...)*, utilizado para calcular as condições iniciais de operação, e *CalculaSaídas(...)*, utilizado para atualizar o vetor de saídas, são implementados, concluindo a estrutura do padrão de projeto e o mapeamento entre diagrama de blocos e a orientação a objetos.

C. Uso do Padrão de Projeto Facade no encapsulamento de subsistemas

Para a realização da simulação dinâmica de um SEP, são realizadas as atividades apresentadas na Figura 7. A simulação é iniciada com o cálculo do fluxo de carga para definir a condição inicial de operação do sistema. Com a convergência dos cálculos, são executados os cálculos das condições iniciais de todos os blocos dinâmicos, ou no caso da estratégia proposta, de todas as classes concretas que implementam a dinâmica do sistema. A cada passo de integração, os métodos de cálculos de derivadas são executados e, em seguida, são calculadas as integrais de cada variável de estado com os métodos de integração numérica implementados. Por fim, são calculadas as saídas de cada bloco. Este processo de repete até que seja alcançado o tempo total da simulação.

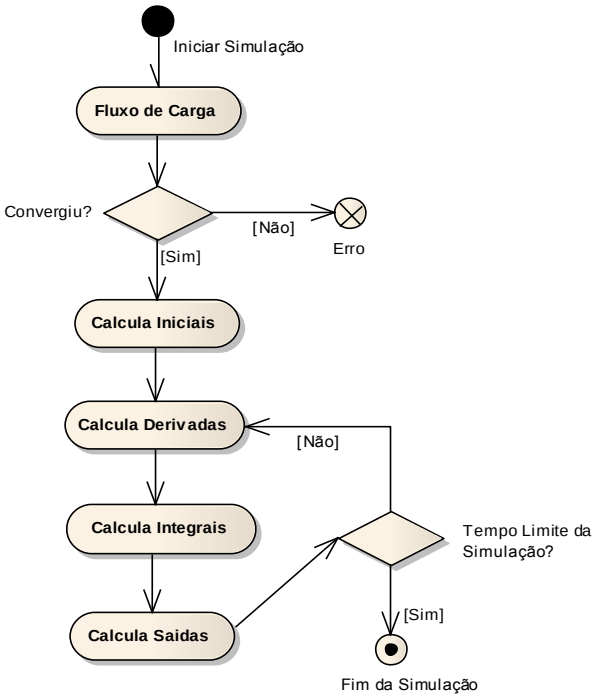

Figura 7. Diagrama de Atividades para Simulação de Sistemas Elétricos de Potência. Fonte: [4].

A classe que gerencia a integração numérica é apresentada em detalhes em [4] e [5], utilizando o padrão de projeto *Builder*, e a fachada, ou a implementação do padrão *Facade*, está justamente na abstração de diversos subsistemas de um SEP que pode ser modelado e implementado como um bloco único. Neste contexto é apresentada a estrutura da Figura 8, cuja classe *TIntegrador* é a classe cliente que é responsável por gerenciar a simulação e realizar os cálculos a cada passo de integração. É ela quem manda a ordem para a classe Sistema, que além de ser uma especialização da classe *Bloco*, serve de fachada para todos os blocos dinâmicos instanciados como objetos do SEP, e assim todos os cálculos são realizados, considerando cada dinâmica implementada.

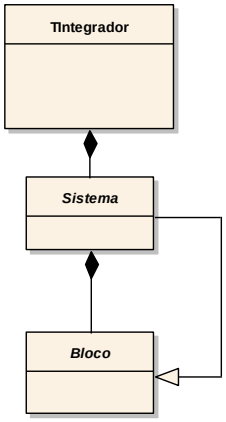

Figura 8. Diagrama de Classes para implementação do padrão de projeto *Facade*. Fonte: [4] e [5].

Nesta estratégia, diversas implementações da classe *Sistema* podem coexistir, contendo diversas implementações da classe *Bloco*. Um detalhe importante é que as implementações dos métodos de cálculo de derivadas e de saídas são as mesmas para todas as classes concretas que implementam blocos dinâmicos. Porém, o cálculo das condições iniciais, não, pois cada bloco dinâmico necessita de diferentes atributos do sistema para suas implementações, fazendo com que o uso do padrão *Facade* também facilite a manutenção e criação de novas classes concretas, ou novos blocos dinâmicos.

Outra vantagem do padrão *Facade* é a possibilidade de acoplar componentes de interfaces gráficas, visualizando em diversos níveis os sistemas e subsistemas de um determinado cenário, similar ao ambiente do Simulink [10].

IV. Implementação e Resultados do Factory Method e Facade em Sistemas Elétricos de Potência

A. Implementação dos métodos das classes especializadas

Os métodos das classes especializadas são os que efetivamente deverão ser implementados para referenciar um bloco no sistema de potência. Cada bloco do diagrama de blocos tem uma classe especializada associada, e assim o mapeamento é efetuado.

Para implementar um gerador, por exemplo, o método *CalculaDerivadas(...)* implementa as equações de estado, cujo modelo 4 é apresentado no sistema de equações (1) [2] onde $\omega_0=2\pi f_0$, ω está em *rad/s*, δ em *rad* e as demais variáveis estão em *p.u.* (por unidade).

$$\begin{cases} \dot{\omega} = \frac{\omega_0}{2H}\left[P_m - P_e - D_a(\omega - \omega_0)\right] \\ \dot{\delta} = \omega - \omega_0 \\ \dot{E_q}' = \frac{1}{T_{d0}'}\left[E_f + (X_d - X_d')\bar{I}_d - E_q'\right] \\ \dot{E_q}'' = \frac{1}{T_{d0}''}\left[E_q' + (X_d' - X_d'')\bar{I}_d - E_q''\right] \\ \dot{E_d}'' = \frac{1}{T_{d0}''}\left[(X_q - X_q'')\bar{I}_q - E_d''\right] \end{cases} \quad (1)$$

O método *CalculaSaidas(...)* é implementado através do sistema de equações (2) [2].

$$\begin{cases} V_q = E_q'' - R_a I_q + X_d'' I_d \\ V_d = E_d'' - R_a I_d + X_q'' I_q \\ P_t = V_d I_d + V_q I_q \\ Q_t = -V_q I_d + V_d I_q \\ V = \sqrt{V_d^2 + V_q^2} \end{cases} \quad (2)$$

O regulador automático de tensão apresentado neste trabalho segue o modelo apresentado no diagrama na Figura 9 (adaptado de [2]), em que V_s é a tensão de referência, V_o é um outro sinal que pode ser a saída do estabilizador de sistema de potência, por exemplo; V_a é a tensão de campo e V_t é a tensão terminal.

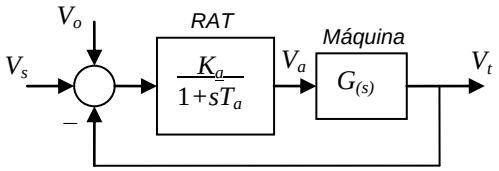

Figura 9. Estrutura do regulador automático de tensão de primeira ordem. Fonte: Adaptado de [2].

Assim, na classe *ReguladorTensão(...)*, que implementa um regulador automático de tensão (RAT) de primeira ordem, a implementação do método *CalculaDerivadas(...)*, que é no domínio do tempo, atualiza a saída do RAT é dado através da equação (3).

$$\dot{V}_a = \frac{K_a(V_o + V_s - V_t) - V_a}{T_a} \quad (3)$$

E a implementação do método *CalculaIniciais(...)* é apresentada na equação (4).

$$V_{ref} = V_s = \frac{V_a}{K_a} - V_o + V_t \quad (4)$$

Todas as demais classes concretas, que representam os blocos do diagrama de blocos, devem ser implementadas dessa maneira, assim como os demais controladores sejam eles PIDs convencionais, adaptativos ou digitais, cargas, etc, independentemente do método de integração, que está encapsulado em uma outra estrutura de classes, utilizando outro padrão de projeto que foge ao escopo deste trabalho.

B. Simulando um Sistema de Potência

Para apresentar o resultado da implementação do padrão de projeto *Factory Method* na simulação de um sistema elétrico de potência do tipo multimáquinas, o sistema utilizado é apresentado no diagrama unifilar na Figura 10. Este sistema está representado com 11 barras e 4 geradores, compondo duas áreas de geração interligadas através de uma linha de transmissão [9].

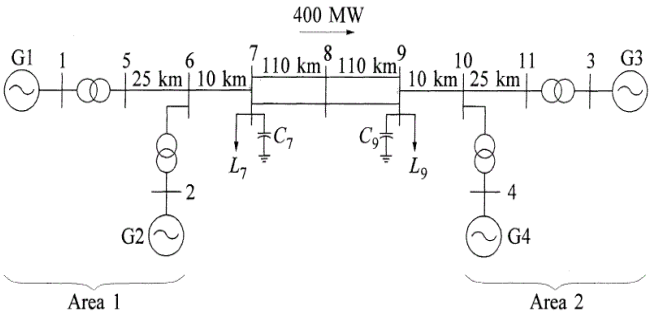

Figura 10. Sistema elétrico de potência usado para simulação. Fonte: [9].

Para efeito de comparação, são apresentadas também simulações realizadas em [1] e [10] para o mesmo sistema.

O gráfico na Figura 11 apresenta as curvas da potência

ativa em todos os geradores através da simulação de um curto-circuito na barra 1, ocorrido em *1s*, com duração de *10ms*. Pode-se observar que após a falta existe uma manifestação de modos de oscilação eletromecânicos no sistema. Esta oscilação apresenta baixo amortecimento, demonstrando a necessidade de estabilizadores de sistemas de potência para melhorar o desempenho da planta.

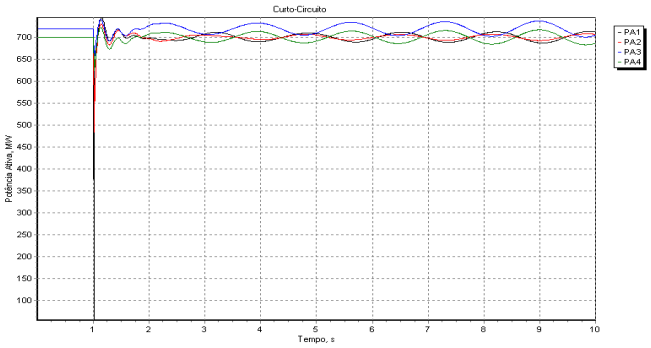

Figura 11. Potência ativa nos terminais de todos os geradores após aplicação de um curto-circuito trifásico na barra 1.

O gráfico na Figura 12 apresenta a mesma simulação realizada utilizando-se o Matlab [10], obtendo-se resultado equivalente.

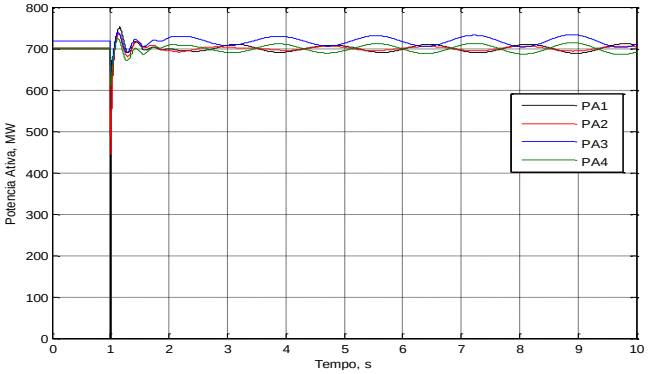

Figura 12. Potência ativa nos terminais de todos os geradores após aplicação de um curto-circuito trifásico na barra 1, utilizando-se o Matlab.

O gráfico na Figura 13 apresenta a curva da potência ativa no gerador 1 através da simulação de um degrau de *5%* na referência do RAT, ocorrido em *1s*. Pode-se observar a presença de um modo eletromecânico de maior frequência que está, aparentemente, bem amortecido, enquanto que após o instante *t=4s* é possível observar o efeito de outro modo eletromecânico, aparentemente instável. Este modo instável possui frequência mais baixa, com período de aproximadamente *1,8s*.

O gráfico na Figura 14 apresenta o resultado de simulação nas mesmas condições utilizando-se o [1], obtendo-se, também, resultado equivalente.

Estes resultados estão plenamente de acordo com os resultados descritos na literatura [9] e concordam com os resultados obtidos utilizando-se ferramentas comerciais para estudos de sistemas elétricos de potência como [1] e [10].

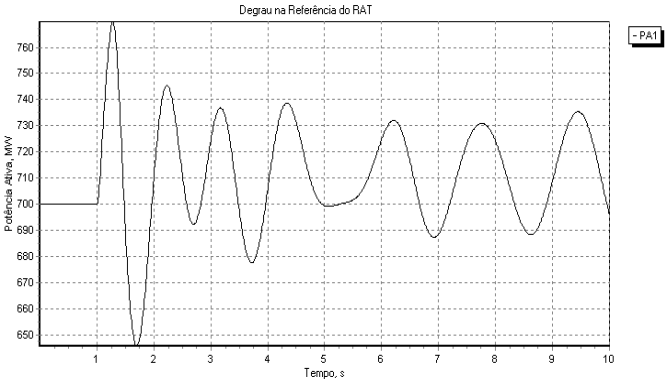

Figura 13. Potência ativa nos terminais de do gerador 1 após aplicação de um degrau de 5% na referência do RAT.

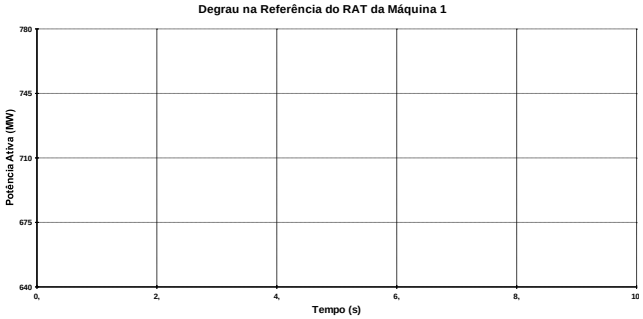

Figura 14. Potência ativa nos terminais de do gerador 1 após aplicação de um degrau de 5% na referência do seu RAT, utilizando-se o ANATEM.

V. Resultados de Validação do Simulador em Comparação com Medidas Dados Reais Coletados na UHE de Tucuruí

A. *Aplicativo Desenvolvido*

Utilizando-se o *software* simulador desenvolvido neste trabalho, denominado *DynaPower*, foi desenvolvido uma ferramenta de software para investigação de fenômenos dinâmicos observáveis em uma grande usina de geração do sistema interligado brasileiro, a Usina Hidroelétrica de Tucuruí. Essa usina é composta por 23 unidades hidrogeradoras, sendo 12 unidades geradoras de 350 MVA e 11 unidades geradoras de 375 MVA, totalizando uma capacidade máxima de geração de 8,325 GVA. A ferramenta de software desenvolvida visa realizar simulações dinâmicas em sistemas elétricos de potência.

No aplicativo desenvolvido, as unidades geradoras da Usina de Tucuruí foram representadas em elevado nível de detalhamento. As máquinas síncronas foram representadas por modelos dinâmicos não-lineares, de 5ª ordem [2]. As turbinas Francis foram representadas através de modelos dinâmicos não-lineares supondo-se que a água fosse incompreensível e o condutor forçado fosse inelástico. Além disso, foram representados também todos os blocos dinâmicos associados aos controladores de cada unidade geradora, incluindo: regulador automático de tensão, regulador de velocidade e estabilizador de sistemas de potência.

A Figura 15 apresenta o diagrama unifilar simplificado das unidades hidrogeradoras interligadas ao barramento infinito, representado por uma grande máquina síncrona que consome a maior parte da energia gerada pela Usina de Tu-

curuí. A interface elétrica de conexão entre a Usina de Tucuruí e o sistema externo, é feita através da barra 36 (ver Figura 15), que representa a subestação da referida usina. As máquinas que vão de G1 a G12 são as máquinas de 350 MVA, e as que vão de G13 a G23 são as máquinas de 375 MVA.

A base comum utilizada no estudo foi de 350 MVA de potência e 13,8 kV de tensão. Os ramos que vão de 24 a 34 são linhas de transmissão curtas, onde todas possuem o mesmo valor de susceptância shunt ($3,6\times10^{-3}$ p.u.) e de impedância série ($3,612\times10^{-5}+j4,5682\times10^{-4}$ p.u. Os ramos que vão de 1 a 23 representam os transformadores das máquinas geradoras as quais possuem reatância série de valor 0,165 p.u. O ramo 35 representa a impedância do equivalente de Thevenin do sistema externo. A barra 24 é o barramento infinito, que também é a referência angular do sistema. As barras que vão de 25 a 35 alimentam pequenas cargas de 1 MW, simbolizando o consumo interno da usina hidroelétrica.

Para validar o simulador dinâmico foram realizados testes de campo na Usina Hidroelétrica de Tucuruí, sendo os ensaios realizados na unidade geradora 8. Os dados coletados foram utilizados para avaliar o desempenho do simulador desenvolvido.

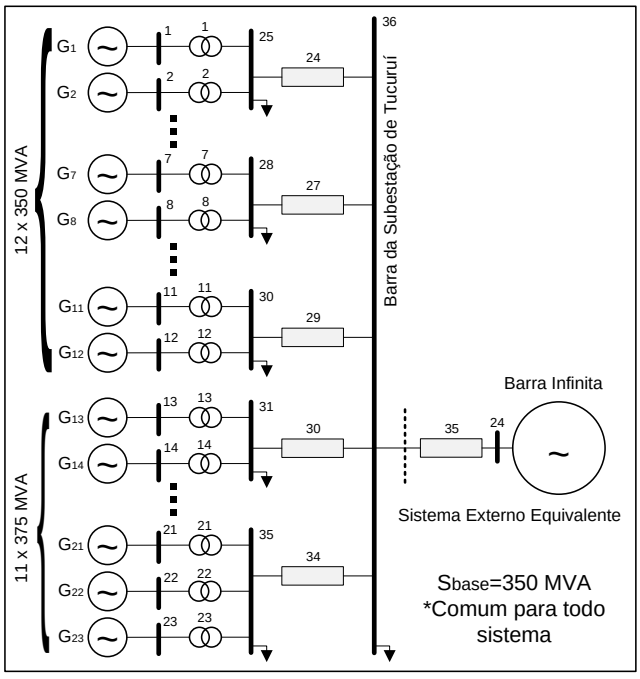

Figura 15: Diagrama unifilar do sistema de potência usado para simulações.

Para avaliar a capacidade do simulador desenvolvido neste trabalho, em relação à capacidade de representação adequada de alguns dos fenômenos dinâmicos observáveis em sistemas elétricos de potência reais, os resultados de simulação computacionais foram comparados com as respostas dinâmicas da planta obtidas em testes de campo realizados na Usina Hidroelétrica de Tucuruí.

O ensaio descrito nesta seção foi realizado no ano de 2009 na unidade geradora de número 8. A potência base da respectiva unidade geradora é de 350 MVA. Na Figura 16 apresentam-se tanto a curva de resposta da potência ativa, medida durante um ensaio realizado em campo pela equipe da Eletronorte juntamente com pesquisadores da UFPA, quanto à curva de resposta da potencia ativa que foi obtida através da simulação do caso, utilizando-se o *software* simulador desenvolvido no projeto (*DynaPower*, em cor vermelha). O teste foi realizado na UGH8, que estava gerando uma potencia ativa de 0,733 p.u. (ou 256,5 MW). Aplicou-se um degrau de +2,9% em 35 s na referência do RV. Analisando-se as curvas da Figura 10, fica evidente o excelente desempenho obtido pelo *software DynaPower* com relação à reprodução confiável dos fenômenos dinâmicos eletromecânicos nas unidades geradoras de Tucuruí.

Pode-se observar que o simulador desenvolvido foi bastante efetivo, fornecendo uma resposta muito similar a resposta real da planta, reproduzindo bem os fenômenos dinâmicos dominantes na planta real.

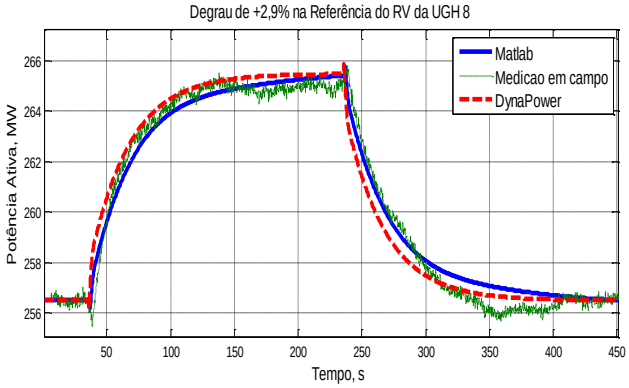

Figura 16. Degrau de +2,9% aplicado na referência do RV da UGH 8 da UHE de Tucuruí. Fonte: [4].

A Figura 17 apresenta-se o resultado de ensaios em campo realizados em 2005, na UGH 1 de UHE de Tucuruí [13]. Neste teste foi aplicado um degrau de +4%, com a máquina operando a 240 MW. Observa-se, novamente, o baixo amortecimento das oscilações com a presença do ESP.

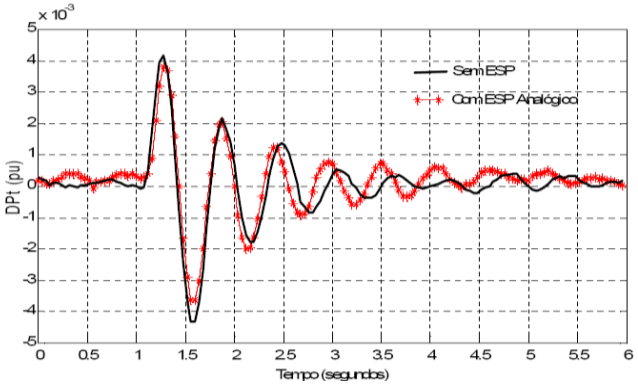

Figura 17: Degrau de +4% aplicado na referência do RAT da UGH 1. Fonte: [13].

Na Figura 18 apresenta-se a mesma situação simulada no software DynaPower, notando-se perfeitamente um comportamento similar ao medido da medição em campo.

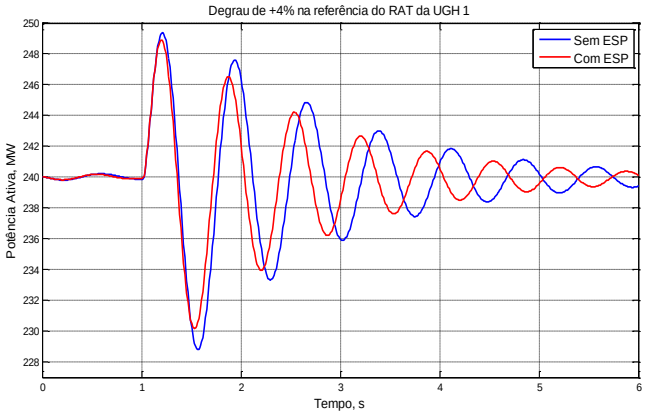

Figura 18: Degrau de +4% aplicado referência do RAT da UGH 1, utilizando o *DynaPower*. Fonte: [4].

VI. Conclusão

O mapeamento entre paradigmas de modelagem é comum em diversas áreas do conhecimento. Especificamente entre os diagramas de blocos e a orientação a objetos é extremamente relevante devido às tecnologias como C++, Java e .NET serem orientadas a objeto, com suas devidas peculiaridades, e na representação de sistemas elétricos de potência e sistemas dinâmicos ser mais utilizado o diagrama de blocos.

O padrão de projeto *Factory Method*, mostrou-se ser extremamente importante neste mapeamento, permitindo a representação de qualquer sistema dinâmico que possa ser representado através de diagrama de blocos.

Além disso, a estrutura do padrão também permite uma fácil implementação da simulação de faltas em sistemas de potência, como curto-circuito e o degrau na referência do RAT, apresentados neste trabalho, como também a variação de carga, degrau na potência mecânica, saída de uma linha de transmissão, entre outras.

O *Factory Method* permite encapsular a dinâmica do sistema em seus respectivos blocos e ter uma estrutura maior responsável por guardar as informações das ligações entre eles e efetuar a simulação de forma eficaz e flexível para qualquer desenvolvedor, que pode representar qualquer bloco dinâmico utilizando uma linguagem orientada a objetos.

Diversas implementações da classe *Sistema* podem coexistir, contendo diversas implementações da classe *Bloco*. Um detalhe importante é que as implementações dos métodos de cálculo de derivadas e de saídas são as mesmas para todas as classes concretas que implementam blocos dinâmicos. Porém, o cálculo das condições iniciais, não, pois cada bloco dinâmico necessita de diferentes atributos do sistema para suas implementações, fazendo com que o uso do padrão *Facade* também facilite a manutenção e criação de novas classes concretas, ou novos blocos dinâmicos.

Outra vantagem do padrão *Facade* é a possibilidade de acoplar componentes de interfaces gráficas, visualizando em diversos níveis os sistemas e subsistemas de um determinado cenário, similar a como é feito no ambiente do Simulink [10].

Os resultados obtidos comparando-se as respostas dinâmicas obtidas com o uso do simulador com dados reais obtidos em testes de campo demonstraram o excelente desempenho do simulador em relação a reprodução de fenômenos eletromecânicos observáveis em campo.

VII. Agradecimentos

Os autores agradecem o apoio recebido das Centrais Elétricas do Norte do Brasil S/A (Eletronorte), através do projeto 63819; a equipe da Usina Hidrelétrica de Tucuruí, em particular aos colaboradores Engenheiros José Jânio de Lana e Ricardo Campos; e aos colaboradores da UFPA, Engenheiros Fabrício Gonzales Nogueira e Marcus Ciro Martins Gomes.

VIII. Referências

[1] ANATEM – Análise de Transitórios Eletromecânicos, Version 10.0.2 (2007). Rio de Janeiro: Centro de Pesquisas de Energia Elétrica – CEPEL.

[2] Arrillaga, J.; Watson, N. R. (2001). Computer Modelling of Electrical Power Systems. 2 ed. Estados Unidos: Wiley.

[3] Booch, G; Rumbaugh, J.; Jacobson, I. (2000). UML: guia do usuário. Rio de Janeiro: Campus.

[4] Di Paolo, Í. F. (2009). Aplicação de técnicas de padrões de projeto orientados a objeto na construção de framework para modelagem e simulação dinâmica de geradores síncronos. Dissertação (Mestrado em Engenharia Elétrica) – PPGEE/UFPA.

[5] Di Paolo, Í. F. et. al. (2010). Creational Object-Oriented Design Pattern Applied to the Development of Software Tools for Electric Power Systems Dynamic Simulations. IEEE Latin America Transactions, n. 3, vol. 8, p. 287-295.

[6] Freeman, Er.; Freeman, El. (2007). Use a Cabeça! Padrões de Projetos (Design Patterns). 2 ed. Rio de Janeiro: Alta Books.

[7] Gamma, E. et al (2007). Design Patterns: elements of reusable object-oriented software. 35 ed. Massachusetts, Estados Unidos: Addison Wesley.

[8] Horstmann, C. (2007). Padrões e Projetos Orientados a Objetos. 2 ed. Porto Alegre: Bookman.

[9] Kundur, P. (1994). Power System Stability and Control, Estados Unidos: McGraw Hill.

[10] Matlab, Version 7.6.0 R2008a (2008). Estados Unidos: The Math Works.

[11] Matos, A. V. (2002). UML: prático e descomplicado, São Paulo: Érica.

[12] Ogata, K. (2003). Engenharia de Controle Moderno. Rio de Janeiro: Prentice Hall do Brasil.

[13] Risuenho, Jorge Ricardo Ribeiro. Desenvolvimento de um estabilizador digital de sistemas de potência para testes em unidades geradoras da UHE de Tucuruí. Dissertação (Mestrado em Engenharia Elétrica) – PPGEE/UFPA, Belém, 2005.

[14] Sena, J. A. S. et al (2005). Estratégias de padrões de projeto do tipo bridge para o desenvolvimento de softwares para simulação de sistemas dinâmicos de grande porte, Sixth Latin-American Congress on Electricity Generation and Transmission.