Inspeção Final para homologação de *design patterns* em uma linha de produção de software embarcado de tempo real

BSc. Danilo Douradinho Fernandes (ITA) danilodf@gmail.com

MSc. Denis Ávila Montini (ITA) denisavilamontini@yahoo. com.br

BSc. Gabriel de Souza Pereira Moreira (ITA) gabrielspmoreira@yahoo. com.br

PhD. Luiz Alberto Vieira Dias (ITA) vdias@ita.br

Dsc. Adilson Marques da Cunha (ITA) cunha@ita.br

*Resumo: Este artigo apresenta um processo de Inspeção Final de componentes de software desenvolvidos na linguagem C++ em um ambiente IBM-**R**ational **R**ose **R**eal **T**ime (**RRRT**). Nele, propõe-se a criação de um processo de homologação consistente para a eliminação de erros, falhas e defeitos de componentes antes de publicá-lo em uma biblioteca de padrões de projeto (design patterns). A sua principal contribuição encontra-se na construção de um artefato de inspeção final para componentes de software. A proposta de criação de um modelo de processo industrial de inspeção final para homologação de design patterns em um Ambiente Integrado de Engenharia de Software Ajudada por Computador (**I**ntegraded **C**omputer **A**ided **S**oftware **E**ngineering **E**nviroment - **I-CASE-E**) para a definição de um processo industrial visou homologar componentes desenvolvidos para que possam ser arquivados e reaproveitados, de forma sistêmica, por meio de bibliotecas. As diretrizes do Processo Unificado da Rational (**R**ational **U**nified **P**rocess – **RUP**) ajudaram aos alunos de Graduação e Pós-graduação do **I**nstituto **T**ecnológica da **A**eronáutica (**ITA**), no segundo semestre de 2007, propiciando a criação de um cenário fértil para a aplicação dos conceitos de design pattern. Um Componente de Software de Computador (**C**omputer **S**oftware **C**omponent – **CSC**) foi homologado com atributos e métodos genéricos para viabilizar sua reutilização.*

Palavras-chave: Fábrica de Software, Células de Manufatura, Ambiente de Ferramentas I-CASE-E, Design Pattern, Reutilização de Componentes, Inspeção Final e Teste.

1. Introdução

No segundo semestre de 2007, os integrantes do **G**rupo de **P**esquisa em **E**ngenharia de **S**oftware (**GPES**) utilizaram o **LAB**oratório de Desenvolvimento **TEC**nológico da **F**undação **C**asimiro **M**ontenegro **F**ilho (**LAB TEC** da **FCMF**) do **I**nstituto **T**ecnológico de **A**eronáutica (**ITA**) para desenvolver um software embarcado de tempo real, utilizando um ciclo de Planejamento, Realização, Controle e Análise (***P**lan, **D**o, **C**heck, and **A**nalysis* – ***PDCA***) para integrar a Inspeção Final (***F**inal **I**nspetion* - ***FI***). O ciclo PDCA integra a engenharia reversa e o modelo cognitivo, para fechar Relatos de Não-Conformidades (***N**on-**C**onformance **R**eports* – ***NCR***) encontrados no *gap analysis* realizado.

O software desenvolvido utilizou: uma metodologia de Aprendizado Baseado em Problemas (***P**roblem **B**ased **L**earning* - ***PBL***) [CUNHA, 2007]; o processo unificado da IBM Rational (***R**ational **U**nified **P**rocess* - ***RUP**);* o Ambiente Integrado de Engenharia de Software Ajudada por Computador (***I**ntegraded **C**omputer **A**ided **S**oftware **E**ngineering **E**nviroment* - ***I-CASE-E***) - **R**ational **R**ose **R**eal **T**ime (**RRRT**) [RATIONAL, 2003]; um padrão de projeto (*design pattern)* [CHRIS, 1977] [GoF, 1995] [GAMMA, 2005] [SELIC, 2006]; uma *FI,* [TCS-2006]; e um *gap analysis* [TCS-2006].

Este trabalho enfoca a definição de Inspeção Final para *design patterns* criados por ferramentas de *I-CASE-E* aplicadas à Análise Causal e Resolução, (***C**ausal **A**nalysis and*

R**esolution* – ***CAR), objetivando a obtenção de um relatório de qualidade com a resolução do problema.

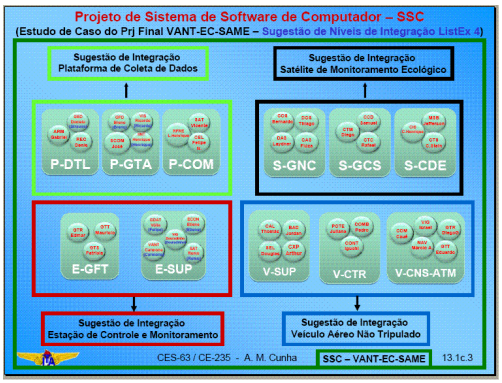
A partir de um processo adaptado do RUP, foram realizadas atividades de controle de qualidade, de definição processo e de desenvolvimento de artefato para a qualidade. Este artefato permitiu a organização, a coleta e a correção das não-conformidades encontradas no componente e corrigiu oportunidades de melhoria para a homologação do *design pattern*. A utilização de ambientes integrados de ferramentas CASE representa significativos investimentos iniciais de desenvolvimento, envolvendo aquisição de licenças, equipamentos, treinamentos e capacitações.

A concepção e a implementação de um novo artefato de *FI* e de um processo de desenvolvimento de software embarcado e de tempo real representaram as principais contribuições deste artigo. Esta *FI* envolveu a identificação, catalogação e reutilização de um padrão de projeto e uma posterior quantificação de ganhos de produtividade e qualidade, obtidos a partir de reutilizações de software, durante o processo de desenvolvimento.

2. Cenário

A pesquisa reportada neste artigo aplicou a metodologia PBL, visando restringir o escopo de reutilização dos padrões de projeto para o desenvolvimento de software, durante a análise do Ciclo de Vida de Desenvolvimento de Sistemas (***S**ystems **D**evelopment **L**ife **C**ycle* - ***SDLC***) mostrado na Figura 01.

Ao se estudar um SDLC voltado para software embarcado de tempo real, em um ambiente acadêmico, adotou-se a seguinte nomenclatura. No mais alto nível, encontra-se o **S**istema de **S**oftware de **C**omputador (**SSC)**, dividido em **I**tens de **C**onfiguração de **S**oftware de **C**omputador (**ICSC**), em **C**omponentes de **S**oftware de **C**omputador (**CSC**), e por último, em **U**nidades de **S**oftware de **C**omputador (**USC)**, todos representados na Figura 01.



LEGENDA:

SSC - **Sistema de Software de Computador.**

CSCI - **Itens de Configuração de Software de Computador.**

CSC - **Componentes de Software de Computador.**

USC - **Unidades de Software de Computador.**

FIGURA 01 – Estratégia para obtenção do Sistema de Software de Computador do Projeto VANT-EC-SAME. Fonte: Cunha (2007).

Neste artigo, aborda-se apenas um CSC, a Plataforma de Registros de Dados (*Plataform Data Logger – P-DTL*), composta por três USCs: a USC **P**-**GED** (**P**lataforma da Unidade de **GE**renciamento de **D**ados); a USC **P**-**ARM** (**P**lataforma da Unidade de **ARM**azenamento de Dados), e USC **P**-**REC** (**P**lataforma da Unidade de **REC**uperação de Dados) do Projeto de um **V**eículo **A**éreo **N**ão **T**ripulado associado a uma **E**stação de **C**ontrole

e a um **S**atélite **A**rtificial para **M**onitoramento **E**cológico (**VANT-EC-SAME**), [CUNHA, 2007].

Após a criação do *design pattern* IO Manager, o passo seguinte foi submetê-lo a um processo de homologação apoiado em um artefato. Este artefato de *FI* serviu para que se pudesse verificar, validar e homologar o componente desenvolvido. A vantagem pesquisada foi à capacidade da *FI* de implementar a Análise Causal para as Não-Conformidades encontradas, durante o processo de inspeção, apresentado na Figura 02.

Inspeção Final:

Ficha Catalográfica	Cadastro do resumo Técnica de um novo componente.
Formulário da Inspeção Final	Formulário de Inspeção Final.
Resultado da Inspeção	Resultado da Inspeção Realizada.

Heurística do projeto de pesquisa da Análise Causal – CAR:

Reengenharia de Processo	Descrição das Técnicas de Reengenharia de Processoe Pesquisa.
Base de Dados - KDB	Base de Dados com os resultados obtidos da Pesquisa.
Plano de Projeto	Planejamento do Projeto de Pesquisa CAR.
Análise Qualitativa	Resumo Qualitativo da Pesquisa.

FIGURA 02 – Sistema ***F****inal* ***I****nspection* – ***FI*** para Sistemas Computadorizados. Fonte: Cunha (2007).

3. Os conceitos de CMMI e CAR utilizados

Este trabalho restringiu-se a um foco na Análise Causal e sua Resolução (***C****ausal* ***A****nalysis and* *R*esolution - **CAR**), uma Área de Processo (***P****rocess* ***A****rea* - **PA**), do modelo de qualidade CMMI [CMMI, 2006] desenvolvido pela *Carnegie Mellon University* para projetos de sistemas. A diretriz de efetividade perseguida pelo CAR consiste em neutralizar os erros e defeitos, logo na sua origem, a fim de evitar a propagação dos erros na operação. A aplicação desta instrumentação científica na área de qualidade pode ser explorada de diversas formas. Através de um conjunto de atividades específicas de coleta de dados e informações, deve-se realizar uma *gap analysis* [TCS-2006].

A *gap analysis* deve constituir-se numa base para a soluções explicadas, de forma detalhada. Ela torna-se fundamental para os modelos de Reengenharia de Processos [TCS-2006]. As suas informações resultantes devem fornecer subsídios para o Modelo Cognitivo proposto, desenvolvido como uma extensão dos tradicionais conceitos de homologação de software, por meio da *FI*.

A análise causal origina-se da *FI* e desdobra-se na Resolução que, por princípio, identifica a origem de defeitos, falhas e erros em um sistema. A abordagem adequada para essas identificações é a constatação fenomenológica, pois é a partir da observação do fenômeno, que se busca o entendimento e a relação estabelecida entre efeitos e causas possíveis.

Do ponto de vista de controle de qualidade, a relação entre a identificação de problemas e as diversas formas de heurísticas cientifica para buscar suas soluções, constituí uma área de estudo embasada em métodos matemáticos e estatísticos [SHEWHART, 1986], para fundamentar a CAR. O ponto de partida consiste na compreensão do processo definido e de como ele é implementado.

A partir da sistemática da *FI* ocorre a utilização da CAR. Defeitos, erros e falhas constituem-se em problemas encontrados em outros Projetos, Fases ou Tarefas do Projeto. Eles são passíveis de análises de suas Causas e de como poderiam ser Resolvidos, tornando-se então um mecanismo para comunicar as lições aprendidas.

A localização e o estudo das relações dos erros e suas origens propostos pela CAR, permitem que se possa entender mais sobre a natureza do problema e permitem o estudo de sua existência. Meios e hipóteses heurísticas de como removê-los, bem como preveni-los. Os tipos de defeitos e outras dificuldades encontradas são analisados para que se possa identificar a existência de tendências e erros sistemáticos.

A aplicação dos conceitos do CAR, a longo prazo em sistemas, acaba proporcionando a melhoria da qualidade e da produtividade em sistemas, prevenindo, por exemplo, a introdução de defeitos em um novo produto. Observa-se que a efetividade de se prevenir defeitos consiste em corrigir o processo e capacitar os colaboradores para evitar a inserção de erros.

3. 1 Conceitos utilizados de Ferramenta I-CASE-E, CMMI - CAR, SDLC, e Inspeção Final

O Ambiente Integrado de Ferramentas CASE da IBM-RRRT, com notação UML, pôde ser utilizado para implementar o paradigma de **O**rientação a **O**bjetos (**OO**), propiciando a geração de componentes em C++ em um projeto de *design pattern.* A utilização da UML propiciou uma linguagem visual para modelagem, construção e documentação de artefatos usuais, em sistemas complexos de software OO [GAMMA, 2005].

Cada *design pattern* precisa ser homologado por uma *FI.* Como resultado da pesquisa, definiu-se um sistema de informações para agregar valor à **VER**rificação (**VER**) e à **VAL**idação (**VAL**), como duas Áreas de Processo (***P**rocess **A**reas* – ***PA*)** do CMMI. Estas duas áreas, coerentes e consistentes entre si, suportam a CAR. A *FI*, representa apenas uma etapa de ambos os processo VER e VAR aplicados à Fase de Teste e Homologação do SDLC.

A *FI* desenvolvida passou por um processo de refinamentos sucessivos até suportar todas as hipóteses de VER, VAR e CAR previstas na homologação de um componente integrado, para poder ser arquivado em uma biblioteca.

2. 2 Os conceitos do SDLC utilizados

Cada Fábrica de software possuí um SDLC composto de Fases com múltiplos passos. Cada SDLC constituí-se de sistemas de informações, integrando-se a sistemas de qualidade de alto nível. A organização deste conjunto de conceitos visa atender ou até mesmo exceder às expectativas dos clientes dentro dos tempos previstos, das estimativas de custo, dos trabalhos efetivos e eficazes, conforme o planejado [CUMMINGS, 2006].

Entre os principais aspectos de adaptação da UML [OMG, 2007] e UML-RT, foram focalizados neste trabalho apenas os conceitos de adaptação da OO tradicional, a fim de que essa abordagem pudesse ser implementada no sistema de Tempo-Real do RRRT. Essa linha de pesquisa da UML-RT focou na definição de um conjunto de construtores que incluíram cápsulas, conectores e portas para construir componentes com alta coesão e baixo acoplamento entre os elementos do sistema [SELIC, 2006].

O RUP, desenvolvido pela IBM-Rational [JACOBSON, 1999], contém um processo completo e padronizado de Engenharia de Software [RATIONAL, 2005], propiciando a atribuição de tarefas e responsabilidades, de forma organizada para a construção de softwares.

A linha base sugerida pelo RUP para artefatos contém documentos de modelagem de sistemas que referenciam a UML e, por extensão, a UML-RT. Para exemplificar a composição formada pelo RUP e a UML, na Figura 03, apresenta-se a relação entre o processo unificado (***U**nified **P**rocess – **UP***) e linguagem UML [SELIC, 2006].

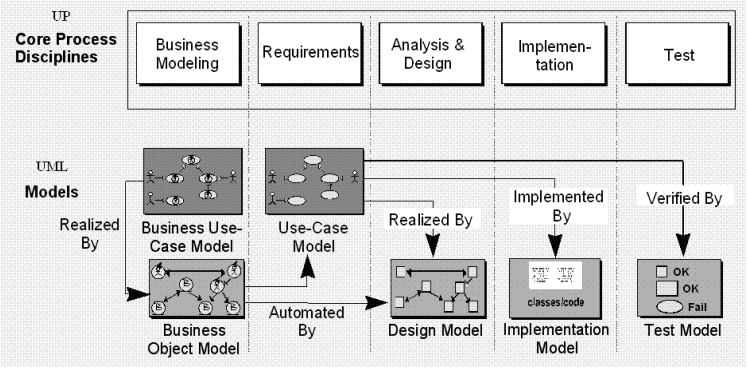


FIGURA 03 – Relacionamento entre o processo UP e os diagramas da UML. Fonte: Cunha (2007).

A combinação da geração de código pelo RRRT, partindo de uma modelagem visual da UML-RT, em um processo UP, já representa uma forma de *framework* de tempo-real. Esta abstração veio de uma modelagem universal com UML-RT, possibilitando a geração de códigos-fonte compatíveis com diferentes linguagens, sistemas operacionais e compiladores [SELIC, 1994].

A ferramenta RRRT gera código por meio da modelagem UML-RT [GAMMA, 2005]. Os componentes gerados pela RRRT incluem Pacotes, Classes Passivas (com Padrões de Classes Orientadas a Objeto), Cápsulas (Classes Ativas com Portas e Conectores) e Classes de Protocolo [RATIONAL, 2003].

O projeto do *design pattern* foi elaborado baseado em um ciclo mínimo do UP. Ele focou nos diagramas de Casos de Uso, de Classes, de Pacote, de Seqüência, de Estruturas e de Estados. A efetividade da OO traduziu-se na modelagem UML-RT para a criação de códigos-fonte em C++, totalmente funcionais por intermédio do ambiente RRRT, propiciando a aplicação de conceitos como o da Reutilização de Componentes.

2. 3. O *Framework* da Ferramenta CASE RRRT

A construção de componentes de software para sistemas de tempo real foi realizada por meio de um *framework*. A abordagem de reutilização foi a *top-down*, permitindo seleções de características genéricas e hipotéticamente adaptáveis à outros componentes semânticamente similares. Um *framework* é uma coleção de classes colaborativas que provêm funcionalidades específicas. Segundo James Booch, a utilização de *frameworks* favorece a reutilização da orientação a objetos [BOOCH, 1999], a fim de que os desenvolvedores possam customizá-los para novas aplicações.

O fundamento empregado na reutilização foi o de que um *framework* precisa ser genérico para um intervalo semanticamente conhecido [GAMMA, 2005]. O efeito da aplicação do conceito de um projeto de *design pattern*, utilizando-se o ambiente integrado de ferramentas CASE RRRT, visou obter um padrão de arquitetura de componentes. Esses componentes tiveram que ser modelados segundo a UML-RT para um processo RUP adaptado [RATIONAL, 2003].

O modelo de processo adaptado e sugerido foi obtido pela observação do ponto mais conveniente do processo para a construção e reutilização dos *design patterns*. Utilizou-se conceitos já existentes como o modelo “V”, oriundo de Linhas de Produção de Células de Manufatura [Montini, 2005], para o SDLC adaptado à reutilização de componentes. O processo pode ser representado pelo modelo “W” proposto, mostrado na Figura 04.

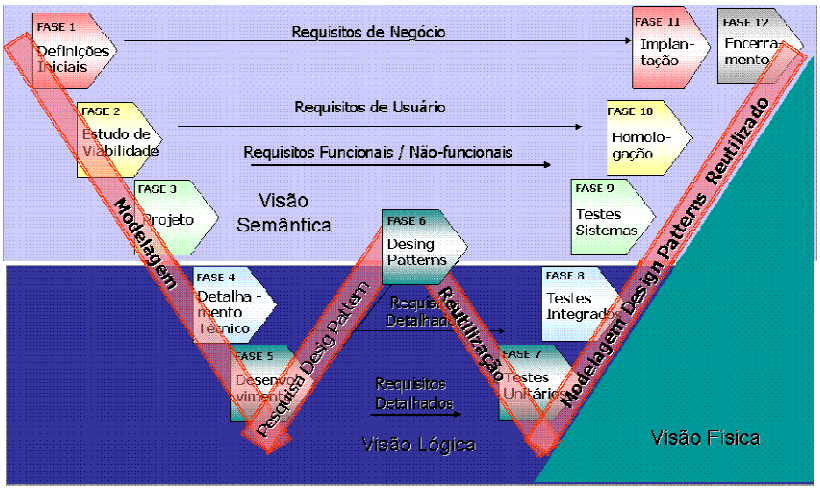


FIGURA 04 – O Metamodelo "W" de Reutilização de Componentes. Fonte: Autores.

O processo foi obtido com a customização do UP [TCS, 2006]. Neste modelo, as fases de "um" à "cinco" identificam os Requisitos de Negócio, Financeiros, Funcionais, Não-Funcionais e Detalhados, representados pelos seus diagramas e, mais especificamente, pelos Casos de Uso do sistema modelado.

O SDLC estudado é constituído de doze Fases. Os conceitos de *framework* aplicados a reutilização ocorreram entre as Fases cinco e sete, conforme mostrado na Figura 03. O modelo "W" foi organizado em três Visões: Semântica, Lógica e Física, todas fundamentadas em diferentes conceitos.

A primeira Visão do Modelo "W" enfoca a semântica e define um processo para a pesquisa dos *design patterns* construídos. A sua reutilização se dá através de um conjunto de procedimentos para definir os pontos específicos de customização do componente analisado. A segunda Visão enfoca a lógica, agrupa e documenta os requisitos decorrente das cinco primeiras Fases do projeto. Ela endereça os aspectos de reutilização e potencialização do uso, por meio de generalizações das necessidades e funcionalidades. A terceira e última Visão do Modelo "W" é a Visão Física que implementa a construção, a adaptação e os testes de componentes em uma linguagem de computação.

2. 4. A utilização do conceito de FI para a validação de *Design Pattern*

A utilização de conceitos específicos como CMMI, SDLC, RRRT, Microsoft C++ e CAR fundamentam o projeto do artefato de Inspeção Final de componente de software [BOOCH, 1999] [BUDD, 2002] [SELIC, 2006].

O uso da *FI* aplicada a *design patterns* permitiu uma forma disciplinada de geração de códigos, homologação e recuperação de histórico de componentes com um conjunto de informações que visou facilitar sua recuperação, durante os processo de pesquisa [BUDD, 2002]. A partir de resultados insatisfatórios da *FI*, uma Análise Causal foi inserida nesse contexto, como por exemplo, para se identificar um ponto de melhoria de performance do próprio SDLC [OMG, 2007] [TCS, 2006]. A *FI* foi desenvolvida para ser aplicável a diferentes classes de projetos e pôde ser considerada um artefato inserido no procedimento de revisão de especificação de testes [TCS, 2007], visando integrar um *framework* genérico em projetos de desenvolvimento.

A aprovação de um componente exigiu a identificação de um procedimento para a homologação do *design pattern*. A concepção da *FI* baseou-se em uma estratégia de generalização. A sua definição foi completada de forma a permitir localizar diversas

anomalias de programas e produtos. A homologação do *design pattern* só foi possível por meio do uso dos procedimentos que acompanham a *FI*.

A utilização da *FI* para homologação não é uma atividade inovadora. Porém, a hipótese testada foi: Quanto mais rápido e fácil de se utilizar a *FI*, mais simples será a Análise Causal. O estudo, correção e integração de uma *FI* agregada a um processo pré-definido contribuiu com a hipótese testada. O conjunto de dados e informações necessárias para aplicar os conceitos da *FI* e CAR foi o mesmo, devidamente integrado em um único artefato de qualidade.

A FI foi desenvolvida para ser aplicável a uma grande quantidade de classes de projetos diferentes e pôde ser considerado como sendo um artefato a ser inserido no procedimento da revisão da especificação de testes [TCS, 2007] visando integrar um *framework* genérico para projetos de desenvolvimento.

A *FI* serviu para se verificar, validar e homologar um *design pattern* neste estudo de caso utilizando a ferramenta CASE RRRT. A *FI* permitiu rastrear a especificação do *design pattern* até os códigos desenvolvidos [BUDD, 2002]. Isso foi realizado em pontos de checagem genéricos estabelecidos para permitir e registro de ocorrências e integrar dados e informações da *FI* e da Análise Causal em um único artefato. A abordagem proposta para a utilização da *FI*, deu-se pela definição dos seguintes pré-requisitos:

Quadro 01 Fundamentos para a utilização do FI

- Utilização de um documento Formal de Requisitos, Exemplo: ***S**oftware **R**equirement **S**pecification* - **SRS**;
- Utilização de Modelos, Notações para a Modelagem dos programas. Exemplo: Análise Essencial ou Estruturada;
- Reallizar um Planejamento Formal para a atividade de Teste;
- Definir um procedimento para apoiar o processo de revisão da especificação e teste;
- Definição dos procedimentos para a revisão das não-conformidades através de um procedimento de análise.

Fonte: Autores.

Os fundamentos sugeridos para a *FI* foram obtidos após as experiências no Laboratório de Desenvolvimento Tecnológico da FCMF no ITA. A revisão técnica encontrou-se relacionada com o código gerado, contendo os seus respectivos requisitos.

3. O Protótipo do Sistema de Informação do FI

A *FI* propiciou a organização das atividades de qualidade, tornando-se uma ferramenta útil, no contexto de VER, VAL e CAR. Para a integração dos conceitos de FI e CAR utilizou-se um sistema de informações amparado por um protótipo contendo os pontos de checagem do processo, a partir procedimentos pré-estabelecidos. O artefato foi divido em duas vertentes. A primeira consistiu de uma inspeção, e a segunda, concatenou a Análise e a Solução. Os conceitos da *FI* neste trabalho aderiram ao CMMi, e a buscaram uma solução fundamentada para a identificação, verificação e validação.

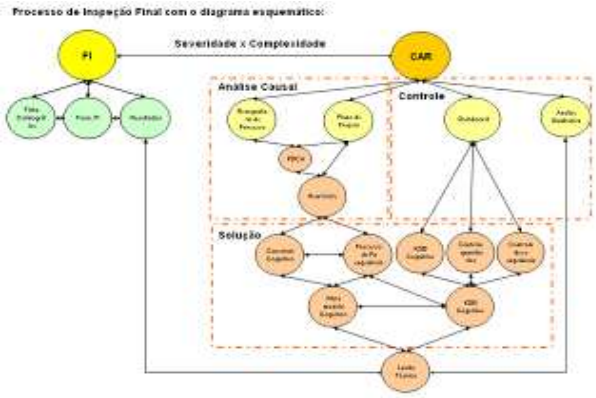


Figura 05 - O organograma do FI. Fonte: Autores.

A primeira dimensão tratou diretamente dos conceitos da *FI* em três diferente concepções: a Ficha Catalográfica; o Formulário da *FI*; e o Resultado. A segunda dimensão CAR envolveu os conceitos: da Reengenharia de Processo; do Plano de Projeto; do ciclo de Deming – PDCA; da Heurística da Análise Causal; do Modelo Cognitivo; do Processo de Reengenharia; do Metamodelo Cognitivo; do *Dashboard;* do ***Knowledge Data Base(KDB)***; do Controle Quantitativo; do KDB Cognitivo; do Controle da Reengenharia; da Análise Qualitativa; e do Laudo Técnico.

O resultado da integração desses conceitos foi organizado em torno de um processo de *FI* e dos procedimentos decorrentes para a efetivação destes conceitos. O organograma do processo é mostrado na Figura 05 com os processos *FI* e CAR devidamente relacionados e hierarquizados.

3. 1. O processo de definição de um Protótipo de *FI*

O RUP, como um processo genérico, propicia oportunidades de melhorias nos procedimentos para a realização de teste e homologação de *design patterns*. Na *FI* definida foi constituído um processo genérico por não se encontrarem estabelecidos *a priori* os papéis e responsabilidades de atividades especialistas. Durante um proceso de verificação para validação tiveram que ser desenvolvidos procedimentos para a *FI* com características específicas. O Quadro 02 apresenta os principais resultados da customização do procedimento para a integração da *FI* com a CAR.

Quadro 01 – Processo de Inspeção Final – FI para a homologação de *design pattern*.

Procedimento	**Inspeção Final – FI**	
Atividades	**Ações**	**Objetivos**
1. Catalogação dos artefatos de software.	1. Utilizar a Ficha Catalográfica para a coleta de dados com o resumo técnico do artefato de software processado.	1. A Ficha Catalográfica visa asegurar que as informações mínimas necessárias e suficientes sobre o artefato de software realizado esteje disponível para ser análisado.
2. Formalização da Inspeção Final.	1 - Seguir o SDLC estabelecido, e para cada requerimento realizar as seguinte análises: 1. 1. Identificar as requisições das perguntas estabelecidos na coluna de "Pontos de Observação"; 1. 2 - Avaliar o encontrado na coluna "Resposta Objetiva (Sim/ Não / NA)"; 1. 3 - Descrever o encontrado na coluna "Resposta Detalhada (Verificação das vidências)"; 1. 4 - Definir na coluna "Severidade" (Baixa, Média, Alta) em relação a prioridade do encontrado; 1. 5 - A coluna "Estado" é uma coluna com resposta automática com base nos itens 2 e 4; 1. 6 - A coluna "Indicador da Fase" é uma coluna automática com os resultados consolidados da Fase, referente ao Item 5.	1. Utilização de um conjunto definado de critérios para a qualificação da conformidade das evidências em relação ao modelo definido.
3. Publicação do Resultado do FI.	1 - Utilizar o Painel da Reengenharia de Processo resultante do FI para a definição do plano de ação necessário para a resolução da oportunidade de melhoria encontrada.	1. O Painel da Reengenharia de Processo apresenta o resultado do consolidado automaticamente da inspeção final visando forncer subsídios para o estabelecimento de um PDCA para resolver os oportnidades de melhoria.
Procedimento	**Heurística do projeto de pesquisa da Análise Causal – CAR**	
Atividades	**Ações**	**Objetivos**
4. Inicialização da Reengenharia de Processo.	1. Iniciar o planejamento do PDCA com base no resultado do FI; 2. Refiniar a origem do Problema com a definição de um processo heurístico; 3. Estabelecer os fundamento da Reengenharia de Processo; 3. 1 - Realizar a composição do Construct do Modelo Cognitivo estudado; 3. 2 - Localizar o Ponto central de inferência e aplicar a Reengenharia de Processo; 3. 3 - Popular a Base de dados da Pesquisa; 4 - Popular o Metamodelo Cognitivo com os dados e informações da Base de dados de pesquisa; 4. 1 Verificar as regras de inferência do Modelo cognitivo.	1. Encontrar a vulnerabilidade e descrever o procesos de resolução do projeto identificado.

5. Planejamento do Projeto.	1. Realizar o planejamento do Projeto de Pesquisa de acordo com as características identificadas no processo de heurística do objetivo: 1 - Estimar a quantidade de horas necessárias; 2 - Definir o SDLC; 3 - Atribuir o percentual de consumo de horas da fase; 4 - Definir o cronograma de execução.	1. Definir um plano de projeto específico para o projeto de pesquisa de acordo com as características definidas pelo processo heurístico.
6. Base de Dados - KDB.	1. Popular a base de dados KDD do projeto.	1. A conepção deste modelo de banco de dados visa cobrir as necessidades de informação do modelo cognitivo.
7. Entrepretação das Análises Qualitativas.	1. Análiser Qualitativamente os resultados apurados pelos especialistas.	1. Qualificar a solução encontrada para cadastro histórico.

Fonte: Autores.

A adaptação do RUP caracterizou a criação de um novo padrão de processo (*pattern process)* mostrado no Quadro 02. O resultado representou uma forma original e adaptativa de se resolver a implementação da verificação para a validação para *design patterns* de forma integrada. A implementação da Revisão Técnica conhecida como Inspeção Final (***Final Inspection* - *FI***), foi realizada ao final de cada Fase do SDLC do Projeto, seguindo-se o conceito do Modelo ***Entry Task Verify and eXit* (ETVX)** [RADICE, 1988].

4. Os resultados da aplicação do FI em um d*esign pattern* RRRT

As vulnerabilidades e Não-Conformidades encontradas serviram para se identificar os problemas a serem resolvidos. As proposições identificadas precisaram ser reavaliadas, de acordo com um processo de Análise Causal particularizado para *design pattern.* Neste estudo, o problema foi centralizado na Fase seis do Modelo "W". O Quadro 03 apresenta os passos percorridos para a identificação do problema e homologação do *design pattern.*

Quadro 02 – Processo da Análise Causal da homologação de *design pattern*.

Processo da Análise Causal		
Atividades	**Ações**	**Objetivos**
1.Identificação do negócio.	1. Analisar as equivalências semânticas dos requisitos Negócios, Financeiros, Funcionais, Não Funcionais e Detalhados. 2. Criar as Matrizes de Atributos de Requisitos.	1. Para cada tipo de requisito, apresenta-se uma matriz que lista os componentes que contenham pelo menos um dos requisitos em um eixo e todos os atributos no outro eixo.
2.Seleção de componentes candidatos na biblioteca.	1. Pesquisar na biblioteca de componentes os requisitos identificados. 2. Análisar as Matrizes de Rastreabilidade de Requisitos.	1. Definição dos critérios de escolha dos componentes rastreados. Para cada tipo de requisito, apresenta uma matriz que lista os requisitos em um eixo e todos os itens rastreados no outro eixo.
3.Teste de componentes.	1. Análise da documentação e infra-estrutura de teste.	1. Verificação e Validação do Projeto de Teste do componente.
4.Planejamento da customização e iteração.	1. Detalhar os pontos de customização para o planejamento. 2. Árvore de Rastreabilidade de Requisitos.	1. Mapeamento da árvore de rastreabilidade, através uma visualização gráfica, dos pontos de customização e os relacionamentos da rastreabilidade afetados.
5.Distribuição de componentes para customização.	1. Disponibilizar para a equipe da documentação do componente e os pontos de customização.	1. Utilização de uma ferramenta de gerenciamento de requisitos para manter as informações de repositório.
6.Processo de Customização.	1. Dimensionar e executar e customização do componente.	1. Atualização do Plano de Projeto – A atividade de estimativa de software consiste na identificação dos atributos que devem ser re-documentados nos produtos, no Plano de Gerenciamento de Requisitos. 2. Envio o componente para a Customização: O "Revisor do Projeto" determina se a equipe de projeto já reuniu os itens necessários para a construção. A principal meta é determinar se o plano de trabalho é suficiente para a produção. Os outros passos das atividades de revisão permanecem iguais, visto que são condizentes com o *pattern process*.
7.Identificação de um padrão de projeto candidato.	1. Proposta de um novo padrão à biblioteca.	1. Especificação do motivo da Seleção: A equipe técnica do projeto que produziu os artefatos reúne todos as características genéricas da modelagem e arquitetura do componente para que um padrão possa ser apresentado aos revisores. 2. Seleção d os componentes na biblioteca para a revisão: A equipe deve informar ao "Revisor do Projeto" quando eles estarão prontos

		para a revisão e quais são os alvos da revisão.
8.Catalogação do padrão de projeto.	1. Realizar o processo de catálogo de padrão de projeto.	1. Elaboração da ficha catalográfica do padrão de projeto, no qual seja descrita sua motivação, aplicabilidade, estrutura, ganhos na reutilização e exemplos de implementação. 2. Armazenamento da ficha catalográfica em uma biblioteca de dados e informações sobre requisitos, atributos e dependências de projeto.
9.Finalização do Projeto de reutilização.	1. Revisar a documentação.	1. Revisão e validação os passos e documentações anteriores.

Fonte: Autores.

A utilização do artefato de *FI* formalmente constituído, permitiu a identificação das vulnerabilidades do *design pattern* construído e suas correções antes da entrega ao destinatário final. Neste caso, uma biblioteca de componentes. No estudo do fenômeno da realização deste tipo de *FI* para *design patterns*, primeiramente, criou-se um processo para a *FI*. Em seguida, criou-se um processo para a CAR, visando a verificação e validação dos critérios de modelagem, dos códigos e das características do produtos [RATIONAL, 2003].

5. Conclusão

A real contribuição foi obtida com base na elaboração e aplicação de um conceito de um artefato e um processo especialista de Inspeção Final (***F**inal **I**nspetion* - ***FI***) [TCS-2006] em uma linha de produção de componentes [MONTINI, 2005] para sistemas de software embarcado e de tempo real para a homologação de Componentes de Software de Computador, CSC – um *design pattern* denominado CSC: IO Manager.

A reutilização de padrões de projeto para desenvolvimento de software embarcado de tempo real utilizou o Aprendizado Baseada em Problemas (***P**roblem **B**ased **L**earning* - ***PBL***) [CUNHA, 2007] para a definição da linha pesquisada. O resultado da pesquisa foi a elaboração dos Processos “W” e o FI. Ambos os processos compõem a homologação de *design patterns* para uma linha de produção software embarcado de tempo real.

Segundo o último levantamento do projeto VANT-EC-SAME, foram geradas pela ferramenta *CASE RRRT* 15. 096 linhas de código em C++, distribuídos em 56 arquivos, com a utilização de modelagem visual, evidenciando o reuso de programas com um mínimo de codificação manual. A utilização de um *framework* de RRRT para construção de componentes deve ser uma atividade planejada e os seus pré-requisitos conhecidos, definidos e respeitados. A despreocupação com os pré-requisitos de reutilização de componentes genéricos e *design pattern* podem representar desperdícios de custo e tempo, ao longo dos projetos.

A proposição de *design pattern* necessitou de conhecimentos específicos não só na modelagem *UML-RT*. O conhecimento técnico na linguagem de programação escolhida foi fundamental, pois o desconhecimento impactou no plano de teste, na capacidade de abstração, na catalogação e na seleção do *design pattern* na biblioteca. A reutilização do *design pattern* IO Manager, no Modelo de Processo “W”, necessitou que os aspectos técnicos fossem respeitados. O domínio dos conceitos de UML-RT como portas, classes estáticas e dinâminas foram fundamentais para a efetividade da operação.

A criação de um modelo de FI, para homologação dos *design patterns* de software embarcados de tempo real para a ferramenta *I-CASE-E RRRT* é fundamental para proporcionar um grau de qualidade e reutilização que justifique o custo de desenvolvimnento [CUNHA, 2007]. A consequência de se obter um projeto com esse tipo de qualidade é o alto custo que este tipo de processo construtivo agrega ao produto.

O uso de conceitos como a criação de uma da ficha catalográfica foi fundamental para o arquivamento e teste do design pattern e para a organização da UML-RT e do ***R**ational **U**nified **P**rocess – (**RUP**)*. Os ganhos obtidos foram demonstrados no estudo de caso. O uso

de diversas tecnologias aninhadas em uma Linha de Produção de sftware embarcado de tempo real [MONTINI, 2005] viabilizou economicamente a reutilização do IO Manager no projeto *I-CASE-E*.

6. Recomendações e sugestões

Esta pesquisa desenvolveu alternativas para melhorar aspectos de implementação tecnológica *I-CASE-E* em Fábricas de Software para maximização de uso de *design pattern*. Recomenda-se a reutilização sistemática do conceito de Ficha-Catalográfica, em larga escala de componentes software. A extensão desses processos e métodos propostos ficam a critério de grupos de pesquisa interessados no conceito de reutilização de software.

Sugere-se a utilização dos conceitos de Modelo W, *design pattern* e Ficha-Catalográfica em Linhas de Produção de Células de Manufatura [MONTINI, 2005] [MONTINI, 2005a] especializadas em ferramentas *I-CASE-E*. O modelo "W" deve ser revisitado, refinado e customizado para cada nova Linha de Produção. Sugere-se ainda que, a aplicação deste Modelo em Fábricas de Software [MONTINI, 2006a] [MONTINI, 2006b] [MONTINI, 2006c] [MONTINI, 2006d] [MONTINI, 2007] [MONTINI, 2008] vise ganhos de escala na produção de software embarcados e de tempo real.

7. Agradecimentos

A manutenção de uma infra-estrutura para pesquisa em Engenharia de Software é um ponto crucial. Institutos e Grupos de Pesquisas como o ITA e o GPES vêm contribuindo para o avanço de trabalhos especializados. O acompanhamento das tendências atuais, por meio da aplicação de conceitos de software embarcado de tempo real, representa alternativas para o preparo e a implementação de novas tecnologias. Projetos pedagógicos envolvendo PBL, bem como parcerias com empresas do setor de produtivo, como a TCS – Tata Consultancy Services, mostrados neste artigo, já vêm representando relevantes trocas de experiências profissionais e acadêmicas no mundo moderno.

8. Referências

[BOOCH,1999] Booch, G. , "Object-oriented analysis and design with applications", Benjamin/Cummings, 1999.

[BUDD, 2002] Budd, T. , "An Introduction to Object Oriented Programming", Addison Wesley 3rd Ed, 2002.

[CHRIS, 1977] Christopher, Alexander. A Pattern Language. Estados Unidos da América: Oxford University Press, 1977.

[CMMI, 2006] CMMI® for Development (CMMI-DEV) Version 1. 2, CMU/SEI-2006-TR-008, Carnegie Mellon University. Disponível no site: http://www. sei. cmu. edu/publicatons/documents/06. reports/06tr008. html, Acessado em Agosto de 2006.

[CUNHA, 2007] Cunha, A. , "CE-235 Real-time Embedded Systems Lecture Notes", Brazilian Aeronautics Institute of Technology – ITA. Disponível em : http://www. ita. br/~cunha, Acesso em: Dezembro de 2007.

[CUMMINGS, 2006] Cummings, Haag, McCubbrey, Pinsonneault, Donovan, (2006), Sistemas de Informação de Administração para a Informação Envelhecem, Toronto, McGraw-colina Ryerson

[JACOBSON, 1999] Jacobson, I. , and Rumbaugh, J. , "The Unified Software Development Process", Addison Wesley Logman, 1999.

[GAMMA, 2005] Gamma, E. , Helm, R. , Johnson, R. , Vlissides J, "Design Patterns", Addison-Wesley, 2005.

[GoF, 1995] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. 1. ed. Estados Unidos da América: Addison-Wesley, 1995. ISBN 0201633612.

[MONTINI, 2005] Montini, Denis Ávila, Modelo de indicadores de risco para o orçamento de componentes de software para célula de manufatura. / Denis Ávila Montini. 360p. Dissertação (Mestrado) em Engenharia de Produção – Universidade Paulista - (2005).

[MONTINI, 2005a] Montini, Denis Ávila; Spinola, Mauro de Mesquita; Sacomano, José Benedito; Nascimento, Marcos Ribeiro Do; BATTAGLIA, Danilo. Aplicação do Modelo PSP Manual e amparado por ferramenta CASE em um estudo de caso de Fábrica de Software Brasileira. Simpósio, Hotel Serra Azul - Gramado, RS, 2005.

[MONTINI, 2006a] Montini, Denis Ávila; Albuquerque, Antonio Roberto Pereira Leite de; Nascimento, Marcos Ribeiro Do. Strategy for the use of the model of personal software process for the establishment of measurement and analysis for obtain CMMI level 2 in a study of case of a brazilian software factory. In: ASEE - 5th ASEE Global Colloquium on Engineering Education, Rio de Janeiro, 2006.

[MONTINI, 2006b] Montini, B Denis Ávila; Spinola, Mauro de Mesquita; Sacomano, José Benedito; Nascimento, Marcos Ribeiro do; Battaglia, Danilo. Application of model PSP manual and supported by tool in a study of case of brazilian plant of software. Revista produção on line, Florianópolis - SC - Brasil, 2006.

[MONTINI, 2006c] Montini, Denis Ávila; Sekhar, Chandra; Negry, Tatiana Tavares; Nascimento, Marcos; Battaglia, Danilo. Strategy for the use of the model of personal software process for the establishment of measurement and analysis for obtain CMMI level 2 in a study of case of a brazilian software factory. In: Montivideu: TACTICS Iberoamerica 2006.

[MONTINI, 2007] Montini, Denis Ávila; Moreira, Gabriel de Souza; Vieira, Luiz Alberto; Battaglia, Danilo; Gnatiuc, Carlos Eduardo; Cunha, Adilson Marques da; In: TACTICS Iberoamerica 2007. Estudo de caso de uma estratégia de integração de middleware para um serviço SOA de gerenciamento e controle de fábrica de software: TCS – Tata Consultancy Services – Intranet website de Base de Conhecimento Corportativa KnowMax: TACTICS Iberoamerica 2007, Brasil, São Paulo, SP 25-26/Outubro/2007.

[MONTINI, 2008] Montini, Denis Ávila; Moreira, Gabriel de Souza; Vieira, Luiz Alberto; Battaglia, Danilo; Gnatiuc, Carlos Eduardo; Cunha, Adilson Marques da; In: Global TACTiCS 4th Global Conference, Study of case of a strategy of middleware integration for SOA service of administration and control of factory of software: TCS – Tata Consultancy Services – Intranet website Corporative Knowledge Data Base KnowMax: Global TACTiCS 4th Global Conference, Índia, Kerala, Thiruvananthapuram. 16-19/janeiro/2008.

[OMG, 2007] OMG, “UML”. Dispoível em: http://www. omg. org/gettingstarted/what_is_uml. htm. Acesso em: Dezembro 2007.

[RADICE, 1988] Radice E Philips, 1988, Special Report CMU/SEI-94-SR-3 - Exemplo de representação de processo em ETVX – Disponível no sítio WEB em 17/06/2007: www. seicmu. edu/pub/ docments /94. reports/pdf/sr03. 94. pdf

[RATIONAL, 2003] Rational Software Corporation, “DEV470 - Mastering Rational Rose Real Time - Student Material”, Volume 1, 2003. Disponível em: http://www-128. ibm. com/com/developerworks/ rational/library/content/03July/1000/1155/_umlmodeling. pdf. Acesso em: Dezembro 2007.

[RATIONAL, 2005] Rational Software Corporation, “Rational Unified Process”. Disponível em: http://www-128. ibm. com/developerworks/ rational/library/may05/brown/index. html. Acesso em: Dezembro 2007.

[SELIC, 1994] Seliec, B. , Gullekson, G. , and Ward, P. , “Real-Time Object-Oriented Modeling”, John Wiley & Sons, 1994.

[SELIC, 2006] Selic, B. , and J. Rumbaugh, “Using UML for Modeling Complex Real-Time Systems”. Disponível em: http://www-128. ibm. Acesso em: Outubro de 2006.

[SHEWHART, 1986] Shewhart, Walter A (1939). Statistical Method from the Viewpoint of Quality Control. Dover Publications Dezembro 1, 1996.

[TCS, 2006] TCS, Tata Consultancy Services, “Operational Process for Iterative Development (Unified Process) Version 1. 0”, Integrated Quality Management System IQMS, Air India Building Mumbai, India Outubro de 2006.