Final Inspection for Design Pattern Homologation using a Real Time Embedded Software in a Production Line

Danilo Douradinho Fernandes  
*Instituto Tecnológico de Aeronáutica - ITA*

Denis Ávila Montini  
*Instituto Tecnológico de Aeronáutica - ITA*

Gabriel de Souza Pereira Moreira  
*Instituto Tecnológico de Aeronáutica - ITA*

Felipe Rafael Motta Cardoso  
*Instituto Tecnológico de Aeronáutica - ITA*

Luiz Alberto Vieira Dias  
*Instituto Tecnológico de Aeronáutica - ITA*

Adilson Marques da Cunha  
*Instituto Tecnológico de Aeronáutica - ITA*

Abstract

This paper summarizes a research process applied to Final Inspection (FI) and Causal Analysis for Problems Resolution (CAR) using a Real Time Embedded Software Component in a Production Line. It uses C++ language in an IBM-Rational Rose Real Time (RRRT) environment. It tackles the creation of a consistent homologation process to eliminate components’ errors, imperfections, and defects even before design patterns’ publications in a library. Its major contribution is the construction of an FI artifact for a software component related to CAR. It proposes an industrial model for an FI process of design patterns’ homologation using an Integrated Computer Aided Software Engineering Environment (I-CASE-E). It considers a design pattern that was identified in a project, described, stored in a library, and successfully reused in another project. The use of Rational Unified Process (RUP) has helped teams of undergrad and graduate students from the Brazilian Aeronautics Institute of Technology (Instituto Tecnológico de Aeronáutica - ITA) to develop their components, during the second semester of 2007. The RUP utilization has provided a fertile scenario to create and apply design pattern concepts. At the end, different Computer Software Components (CSC) were homologated with their attributes and generic methods allowing reusability.

Key words: FI, CAR, Test, Design Pattern, and Real Time Embedded Software Component.

1. Introduction

During the second semester of 2007, teams of students from the Brazilian Aeronautics Institute of Technology (Instituto Tecnológico de Aeronáutica - ITA) Software Engineering Research Group (Grupo de Pesquisa de Engenharia de Software - GPES) have used the Technological Development Laboratory of the Casimiro Montenegro Filho Foundation (Fundação Casimiro Montenegro Filho - FCMF) to develop a Real Time Embedded Software (RTES).

The Plan-Do-Check-Analyze (PDCA) process was used to integrate the Final Inspection (FI) and the Causal Analysis for Problems Resolution (CAR). The PDCA process cycle has integrated a reverse engineering and a cognitive model to generate a Non-Conformance Report (NCR) in a Gap Analysis during an FI.

Developed software components on this research have used: the Problem Based Learning (PBL) methodology [1]; the IBM Rational Unified Process (RUP); the Integrated Computer Aided Software Engineering Environment (I-CASE-E) from Rational Rose Real Time (RRRT) [2]; a Design Pattern [3] [4] [5] [6]; an FI [7]; and a Gap Analysis report [7].

This paper focuses on an FI definition for a Design Pattern modeled by an I-CASE-E and applied to CAR, in order to obtain a problem resolution quality report. Quality control activities were derived from RUP producing a process definition for the quality area by developing an FI artifact.

The creation of this artifact has allowed the organization of all Non-Conformance Registry (NCR) corrections found in a specific component. In order to allow the Design Pattern homologation, all NCR had to be fixed.

I-CASE-E tools were used as a significant initial development investment, involving licenses acquisition, equipments, trainings, and qualifications. A new FI artifact was conceptualized and implemented in a real time embedded software process, representing the major contribution.

During the development process, the creation of this FI involved identification, catalogation, registration, design pattern reuse, productivity, and profit estimation.

2. Scenario

This research has applied the PBL methodology aiming to restrict the scope of the design pattern reuse for software development, during a System Development Life Cycle (SDLC) analysis, shown in Figure 1. In this academic environment, the SDLC study of the RTES has adopted the following nomenclature. On the project highest level was the Computer Software System (CSS) comprised of Computer Software Configuration Items (CSCI). Each CSCI was divided into Computer Software Components (CSC). And finally, each CSC was also divided into Computer Software Units (CSU). All of them are shown in Figure 1 as part of a project named...
Unmanned Aircraft Vehicle – Control Station – Ecological Monitoring Satellite (UAV-CS-EMS), used as a CSS case study.

The CSC Data Logger Point (DTL-P) is composed of the following three different CSUs: a CSU Storage (STRG); a CSU Management (MNGT); and a CSU Recovery (RCVR). These three CSUs represent only one of the three CSC branches of the CSCi Data Logger Platform (DLP). Together with the other three CSCi (UAV, CS, and EMS), the CSCi DLP comprises the CSS named UAV-CS-EMS [1].

During the CSC DTL-P analysis a new design pattern, called IO Manager, was identified, described and published in a repository.

This FI artifact was used to verify, validate, and homologate the developed CSC Data Logger Point (DTL-P). In this case, the added value was the FI capability of complementing the CAR to fix each NCR, as shown in Figure 2.

![Figure 1. The UAV-CS-EMS (CSS) Project [1]](image1)

**Figure 1. The UAV-CS-EMS (CSS) Project [1]**

3. **CMMi and CAR used concepts**

This paper focus on the Causal Analysis and Resolution (CAR), a Process Area (PA) of the CMMi quality model [8], developed by the Carnegie Mellon University for systems projects. In order to prevent errors propagations in operation, the CAR effectiveness was tested to neutralize errors and defects since their origins. The application of this scientific instrumentation in the quality area can be explored in different forms. One of them is through a specific activity dataset and information that should be carried through a Gap Analysis [7]. The Gap Analysis represents the basis for detailed solutions report, supporting process reengineering models [7]. The result information must subsidize the considered cognitive model developed as traditional software homologation models using FI. The CAR information came from FI and went to the problem resolution. The FI artifact was used to identify the origin of defects, imperfections, and errors in the system.

This approach was based on phenomenological identification to verify one specific problem. The relationship between FI artifact and CAR made possible to perform the causal and effect analysis.

From the quality control point of view there is also a relationship between problem identification and scientific searching.

The FI artifact represents a mean to find a solution using mathematical and/or statistical CAR methods. The process starting point consists of identifying and implementing the problem solution.

From the systematic use of this FI and CAR artifact, defects, errors, and imperfections can be identified and eliminated from other Projects, Phases, or Tasks. The FI and CAR artifact can become an effective mechanism to disseminate lessons learned.

The application of CAR concepts have been used for a long time in systems. Finished projects using FI and CAR artifacts have provided significant quality and productivity improvement, preventing defects introduction in new products, for example [7].

3.1. **I-CASE-E tools, CMMi-CAR, SDLC, and FI**

IBM-RRRT environment tools use UML notation and can be used to implement Object Orientation (OO) paradigm, to allow components source-code generation in languages, like C++, and to model design patterns. The UML allows a visual language for modeling, construction, and documentation of artifacts for OO complex software systems [5].

Each design pattern needed to be homologated by an FI. As a result of this study, it was defined an information system to add value to VERification (VER) and VALIDation (VAL) processes, as part of CMMi Process Areas (PA). These two areas were considered coherent, consistent and supported by CAR.
FI represented only one stage of VER and VAL processes that was applied to the Test and Homologation SDLc Phase. The resulting FI has achieved successive enhancements until be able to support all VER, VAR, and CAR hypotheses, recognized from the homologation of an integrated component. Therefore, the component was added to a library.

3.2. SDLC applied concepts

Each software house has an SDLC process with phases of multiple steps. It consists of integrated information systems involving important and high performance quality assurance systems. This set of concepts aims to attend customer’s expectations about development cost estimation and it made possible efficient and effective activities as planned [9].

Among the main aspects of an UML and UML-RT tailoring [10], this study focused only in the traditional OO. Therefore, this approach was able to be applied in a real-time embedded system using RRRT. This UML-RT study principle has focused in a set of components construction parts including capsules, connectors and ports to be built, having as result high cohesion and low coupling between system’s elements [6].

In order to provide a disciplinary approach to distribute tasks and responsibilities within organized vision of software construction developed by IBM-Rational [10], RUP has represented in this case a product which functions have described a complete and standardized process of Software Engineering [11].

Relationships between UP disciplines and de UML Models are shown in Figure 3.

![Figure 3. UP and UML diagrams relationships](image)

The RRRT Tool has generated code using UML-RT modeling [5]. Components generated by RRRT have included packages, passive (OO Standard) classes, capsules (active classes with ports and connectors), and protocol classes [2].

Activities focused in meta-models of classes and package diagrams. These meta-models have helped to define tests of component using the UML diagrams like Use Case, Class, Sequence, Structure, and State. The effectiveness of OO was demonstrated by using UML-RT modeling and the generation of high-quality source code.

3.3. RRRT Framework

The software components construction for real time systems was made using a framework. The reuse approach was top-down, allowing selection of generic characteristics, hypothetically adaptable to other components semantically similar.

A framework is a collection of collaborative classes that provide specific functionalities. According to James Booch, frameworks help object orientation reuse, so that developers can customize them for new applications [8].

A design pattern project application using I-CASE-E RRRT tools has aimed to obtain standard component architecture. The component have to be shaped according to UML-RT for an adapted RUP [2] process.

The adapted and suggested model process was achieved through the identification of the most convenient points for the construction and reuse of design patterns. An existing concept of “V model” was used from Manufacture Cells Production Lines [13] and adapted to the SDLC components reuse. Figure 4 shows the component reuse process represented the proposed model, named W-Model.

![Figure 4. The “W”-Model for Component Reuse](image)

The process was created using UP customization [14]. Model identify “one” to “five” phases: Business Requirements, Financial Requirements, Functional Requirements, Non-Functional Requirements, and Detailed Requirements. They are represented by diagrams and, more specifically, by use cases.

The SDLC was composed of twelve Phases. The framework concepts applied for reuse, representing the main contributions of this research, had occurred between Phases five and seven, and is shown in Figure 4. The W-model was organized in three different views: Semantical, Logical, and Physical. Each one based on different concepts.

The Semantical view of the W-model refers to the search of design patterns. The Logical view focuses in the identification and documentation of a group of
requirements found in the five initial steps of the process. It addresses reuse aspects through the generalization of functionalities. The Physical view of the W-model is related to construction, adaptation and test of components in a computer language.

3.4 FI use in the validation of a Design Pattern

The use of specific concepts like CMMi, SDLC, RRRT, Microsoft C++ and CAR have supported the project of the FI software artifact [13] [6].

The application of FI on design patterns allowed organized code generation, homologation and components description recovery aiming to ease the design pattern identify process. From unsatisfactory FI results, a Causal Analysis was inserted in this context, for example, to identify SDLC performance improvements [10].

The FI was developed to be applicable on different projects and included in tests specification procedures aiming to integrate one generic framework in projects of software development.

A component approval demanded the identification of a design pattern homologation procedure. The FI conceptualization was based on a generalization strategy. Its definition has allowed identifying diverse processes and products anomalies. Design pattern homologation was made possible by using FI procedures.

The FI use for homologation is not an innovative activity. However, this work tested the following hypothesis: Can a Causal Analysis be simpler, faster, and easier by using FI?

The study, correction, and integration of an FI to a predefined process have contributed to the tested hypothesis. Data set and information necessary to apply concepts of FI and CAR were fully integrated in only one quality artifact.

The FI was developed to be applicable to different classes of projects. It was considered as a mechanism to be inserted in test specification revision procedure aiming to integrate one generic framework for software development projects.

The proposed approach for FI was used, according to requirements, as shown in Table 1.

Table 1. The FI Use Requirements

<table>
<thead>
<tr>
<th>The FI utilization must be able to provide:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- The use of legal requirements’ documents, as for example: Software Requirement Specification (SRS);</td>
</tr>
<tr>
<td>- The use of models, notations for programs modeling, as for example: Essential or Structured Analysis;</td>
</tr>
<tr>
<td>- The performance of a formal planning for test activities;</td>
</tr>
<tr>
<td>- The definition e a procedure to support revision process of specification and test; and</td>
</tr>
<tr>
<td>- The procedure definitions of non-conformity revisions through a Procedure Analysis.</td>
</tr>
</tbody>
</table>

This was carried out through generic check-points established to allow and integrate given FI information and Causal Analysis in only one artifact.

The FI artifact was used to verify itself, validate and homologate one design pattern in this case study, by using RRRT tools. The FI has allowed tracking design pattern specifications up to software codification.

The FI development and validation were executed at ITA, in the FCMF Laboratory for Technological Development. The Technical Review was performed on the code generated, considering its requirements.

4. The FI Information System

The FI has allowed the organization of activities related to the software quality process, and became useful for VER, VAL, and CAR.

For the FI and CAR concepts integration, the FI was used to support a prototype with process checking and requirement procedures.

The FI artifact was divided in two parts: Inspections; and concatenated Analysis and Solution.

In this work, FI and CMMi concepts was used to propose a solution based on identification, verification, and validation.

The FI artifact directly dealt with different FI concepts in three conceptualizations: Catalog Form; FI Form; and Result.

The CMMi PA named CAR involves concepts such as: Re-Engineering Process; Project Planning; Deming – PDCA Cycle; Causal Analysis Heuristic; Cognitive Model; Cognitive Meta-model; Dashboard; Knowledge Data Base - KDB; Quantitative Control; Cognitive KDB; Re-engineering Control; Qualitative Analysis; and/or Technical Report.

The proposed FI Process for Design Patterns Homologation is shown in Table 2.
### Table 2 – Final Inspection (FI) Process for Design Pattern Homologation.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Actions</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Software Catalog artifact.</td>
<td>1. To use the Catalog Form for data collection as summarized on the technical report of the software artifact.</td>
<td>1. To generate a Catalog Form aiming to make sure that the minimum and necessary information is available to be analyzed.</td>
</tr>
<tr>
<td>2. Formalization of Final Inspection.</td>
<td>1. To follow the SDLC established and for each requirement to perform the following analysis: 1.1 - To identify questions established on the column &quot;Observation Points&quot;; 1.2 - To evaluate information on the column &quot;Objective answer (Yes/No/Not Applied)&quot;; 1.3 - To describe on the column &quot;Detailed Answer (Verification of evidences)&quot;; 1.4 - To define priorities on the column &quot;Severity&quot; (Low, Average, High); 1.5 - The column &quot;State&quot; is automatically generated based on item 1.2 and 1.4; and 1.6 - The column &quot;Phase Remarks&quot; is automatically consolidated from phase results of item 1.5.</td>
<td>1. To define a criterion to demonstrate conformity with the previous model.</td>
</tr>
<tr>
<td>3. The FI Result Publication.</td>
<td>1. To use the resulting FI Reengineering Process to define the improvement plan.</td>
<td>1. To reengineer the panel process to present automatically the consolidated results from final inspection aiming to give information for a PDCA and decide on the improvement of opportunities.</td>
</tr>
</tbody>
</table>

### Causal Analysis for Problems Resolution (CAR)

<table>
<thead>
<tr>
<th>Activities</th>
<th>Actions</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Reengineering Initialization Process.</td>
<td>1. To start the PDCA planning based on FI results; 2. To refine the problem origin using a heuristics definition process; 3. To establish the reengineering principles; 3.1 - To carry out preparation of the “Construct” which is part of the Cognitive Model; 3.2 - To localize the central point of inference and apply reengineering process; 3.3 - To insert in the research database; 4 - To insert data and research database information in the cognitive meta-model; and 4.1. - To verify the cognitive model inference rules.</td>
<td>1. To find vulnerability and to describe process decision in an identified project.</td>
</tr>
<tr>
<td>5. Project Planning.</td>
<td>1. To plan the Research Project according to the characteristics identified in the objective heuristics: 1 – To estimate the necessary hours; 2 – To define the SDLC; 3 – To assign to each phase hours of percentage consumption; and 4 - To define the execution schedule.</td>
<td>1. To define a plan of a specific project according to specific characteristics defined by a heuristic process.</td>
</tr>
<tr>
<td>6. KDB.</td>
<td>1. To insert data into a KDB.</td>
<td>1. To define a database model conceptualization aims to cover the cognitive information model.</td>
</tr>
<tr>
<td>7. Interpretation of Qualitative Analyses.</td>
<td>1. To analyze the qualitative results provided by specialists.</td>
<td>1. To characterize the solution found in order to register the results achieved in the project.</td>
</tr>
</tbody>
</table>
4.1 The FI architecture process definition

The RUP, as a generic process, allows improvement opportunities in procedures for testing accomplishments and design pattern homologations.

During the verification process, it was developed some procedures for the FI artifact with specific characteristics. Figure 5 presents the main customization, resulted from integration concepts of FI and CAR PA.

The RUP tailoring was characterized by the creation of a new standard shown in Figure 5.

The result has represented an original approach of verification and validation design patterns in an integrated mode.

The FI artifact technical review was carried out on the final phase of the SDLC, following the concept of Model Entry Task Verify and eXit (ETVX) [14].

5. Applying FI results in an RRRT design pattern

Joint vulnerabilities and non-conformities were used to identify problems to be solved. Different approaches were identified to be reevaluated in accordance with a Causal Analysis process distinguished by the design pattern homologation process shown in Table 3.

The study of this problem was centered on phase six of the W-model. Table 3 presents steps covered for problem identification and homologation of the design pattern.

The FI artifact use has allowed the identification of design pattern vulnerabilities and their corrections before the final delivery. In this case it allowed the creation of a component library.

In order to perform a design pattern phenomenon study, it was first created an FI process.

Afterwards, a process for CAR was also created, aiming to fulfill the modeling, verification, and validation criteria, including code and product characteristics [2].
Table 3. Causal Analysis Process for Design Pattern Homologation.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Actions</th>
<th>Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business identification.</td>
<td>1. To analyze semantical business equivalences requirements, financials, functional, non-functional, and detailed.</td>
<td>1. To identify, for each requirement type, a matrix is presented listing components that contain at least one of the requirements in one axis and all attributes in the other.</td>
</tr>
<tr>
<td>2. Selection of candidate component in the library.</td>
<td>1. Search in components library to identify requirements. 2. To identify requirements in the traceability matrix.</td>
<td>1. To define components tracing criteria. For each requirement type, it presents a matrix that lists requirements in an axis and the entire item tracked in the other.</td>
</tr>
<tr>
<td>3. Test of components.</td>
<td>1. To do every documentation analysis and test infrastructure.</td>
<td>1. To verify and validate component test project.</td>
</tr>
<tr>
<td>4. Iteration customization planning.</td>
<td>1. To detail customization points for planning. 2. To create requirements traceability tree.</td>
<td>1. To map the traceability tree throughout customization points as a graphical visualization and affected traceability relationships.</td>
</tr>
<tr>
<td>5. Components distribution for customization.</td>
<td>1. To make component documentation of customization points available for groups.</td>
<td>1. To manage requirements using tools to keep repository information.</td>
</tr>
<tr>
<td>6. Customization process.</td>
<td>1. To dimension and execute component customization.</td>
<td>1. To update the software activity estimate project plan which consists on attributes identification that must be registered in products, in management requirements plan. 2. To send component for customization to project reviewer to determine if the project team already has congregated the necessary items for its construction. In this case, the main goal is to determine if the work plan is enough for the production. Review activities remain equal since they are consistent with the pattern process.</td>
</tr>
<tr>
<td>7. Identification of a project standard candidate.</td>
<td>1. To create a new standard to the library proposal.</td>
<td>1. To determine if the elected specification reason is necessary. This project technique group which has produced artifacts needs to congregate all generic modeling characteristics and component architecture, so that a reviewed standard can be presented. 2. To select components in the library for review. The analysis group must inform to the project reviewer when they will be ready to reach targets.</td>
</tr>
<tr>
<td>8. Project standard catalog.</td>
<td>1. To perform project standard catalog process.</td>
<td>1. To elaborate a standard project catalog form describing its motivation, applicability, structure, profits in reuse, and implementation examples. 2. To catalog storage forms into a data library with requirements, information, attributes, and project dependencies.</td>
</tr>
<tr>
<td>9. Project reuses finishing.</td>
<td>1. To revise the documentation.</td>
<td>1. To review, document, and validate previous steps.</td>
</tr>
</tbody>
</table>

6. Conclusion

The major contribution of this research was the Final Inspection (FI) process elaboration and application [7] on components constructed in a production line [15], using a real-time embedded software system. It was applied in the homologation of a design pattern named IO Manager, derived from a Computer Software Component (CSC) called Data Logger.

The reuse of design patterns on a real time embedded system was applied within a Problem Based Learning (PBL) environment [1]. One of the main results of this study was the elaboration of W-process and FI artifact, both comprising a design pattern homologation process, applied to a production line using a real time embedded software.

According to the last UAV-CS-EMS project statistics, 15,096 lines of C++ code were generated by the Rational Rose Real Time (RRRT) CASE tool. It was distributed in 56 files, using visual modeling for components and patterns reusability with a minimum of manual codification. An RRRT framework was used to construct components as planned activity. It was important to apply components reuse and design pattern. Otherwise it could represent higher cost and wasting time during the software development process.

In order to successfully develop the IO Manager design pattern, not only specific knowledge about UML-RT was required, but also several other technological knowledge in a specific chosen programming language.

It was also needed an essential familiarity with the main impacts by using test plan, abstraction capacity, catalog, and design pattern election for a library. The
reuse of the IO Manager design pattern in the W-process has required also the consideration of some technological aspects. UML-RT domain concepts, ports, and static and dynamic classes were fundamental for operation effectiveness.

An FI artifact conceptualization for embedded real time software design pattern homologation using I-CASE-E RRRT tool was essential to improve quality and reusability levels [1]. The consequence of a project with this quality level could represent high profit because of the value added to the software process and product. FI artifact and catalog form creation were fundamental for fulfilling design pattern test, organized in UML-RT and Rational Unified Process (RUP). Technology applications in a real time embedded software production line [15] made possible efficient reuse of the IO Manager design pattern in an I-CASE-E project.

7. Recommendations

In order to continue this research, it is recommended:
1 – The systematic reuse of the Catalog Form concept on larger software scale components;
2 – The reuse of the W-process to support design pattern and Catalog Form in production lines of manufacture cells;
3 – The refinement and review of the W-process to allow the customization for each new production line; and
4 - This model to be applied on software houses and factories to scale up profits in RTES production cells.

Finally, it is suggested that the results of this research be used as an alternative to improve technological implementation aspects using I-CASE-E for design pattern maximization.

8. Acknowledgments

Authors of this paper would like to acknowledge:
1 – ITA for the opportunity of providing the course of CE-235 Real-time Embedded Systems for projects involving PBL; 
2 – GPES for its research environment; 
3 – FCMF for its software engineering infrastructure for research; and 
4 – Tata Consultancy Services (TCS) as well as for its partnerships with productive sector companies.

9. References