

# A Model for Safety-Critical Real-Time Systems by making use of Architectural Design Patterns

U.V.R. Sarma<sup>1</sup>, Sahith Rampelli<sup>2</sup> and Dr. P. Premchand<sup>3</sup>

<sup>1</sup>CVR College of Engineering, Department of CSE, Ibrahimpatan(M), R.R. District, A.P., India  
Email: sarmauvr@yahoo.co.in

<sup>2</sup> CVR College of Engineering, Department of CSE, Ibrahimpatan(M), R.R. District, A.P., India  
Email: sahith.indian@gmail.com

<sup>3</sup>Osmania University, Department of CSE, Hyderabad, A.P., India  
Email: p.premchand@uceou.edu

**Abstract**—Design Patterns, which give abstract solutions to commonly recurring design problems, have been widely used in the Software and Hardware domain. This paper presents the principles of Architectural & Design patterns for Real-Time Software Systems. For the successful application of design patterns for Safety-Critical Real-Time systems, an integration of a number of architectural design patterns is desirable. For this reason, a pattern catalog is constructed that classifies commonly used Hardware and Software design methods. Moreover, it is intended to construct the catalog such that an automatic recommendation of suitable design patterns for a given software application can be made. The paper focuses on reliability patterns and studies the impact of the patterns on the QoS. To support the designers, a tool is developed to suggest the patterns that are appropriate for the software based on the characteristics of the problem. This tool will be able to help generate the code template for the selected design pattern.

**Index Terms**— Design Pattern, Real-Time Systems, Non-Functional Requirements, Safety-Critical Systems.

## I. INTRODUCTION

Over the last few years, Real-Time systems have been increasingly used in safety-critical applications where failure can have serious problems. Designing of Real-Time systems is a complex process, which requires the assimilation of common design methods both in hardware and software to implement functional and non-functional requirements for these safety-critical applications.

Design patterns provide abstract solutions to commonly recurring design problems in the software and hardware domain. In this paper, the concept of design patterns is adopted in the design of safety-critical real-time system. A catalog of design patterns is constructed to support the design of real-time systems. The proposed catalog contains a set of software and hardware design patterns to cover frequent design problems such as handling of random and systematic faults and safety monitoring. Furthermore, the catalog provides a decision support component that supports the decision process of choosing a suitable pattern for a particular problem based on the available resources and the requirements of the applicable patterns. The Proposed Tool Provides the Code Template for the selected pattern.

Bruce Douglass proposed several design patterns for the Safety-Critical Real-Time applications based on the well-known design methods. The UML (*Unified Modeling Language*) provides a notation for design patterns but this notation is targeted primarily towards mechanistic design patterns. In this proposed paper, we are not discussing the Design patterns in a detailed way in order to limit its size.

## II. DESIGN PATTERN TEMPLATE

In this section, we proposed a Design Pattern Template to represent the well-known design patterns for Safety-Critical Real-Time Software applications. As shown in Figure 1, the upper part of the template includes the traditional representation of a design pattern and also listing of the pattern implications on the non-functional requirements. Moreover, further support is given by stating implementation issues, summarizing the consequences, side effects, related patterns and Code Template as well.

Design Pattern Template	
Pattern Name	A handle or a meaningful name to describe the pattern.
Other Name	The other well-known names for the pattern.
Pattern Type	Classification of Design patterns into either software or hardware
Abstract	Describes the short description of the pattern.
Context	The general situation in which the designer may use this pattern
Problem	This part gives a Summary of the problem which is addressed and solved by this pattern.
Structure	Structure represents a solution to the problem under consideration. It provides the main elements of the pattern, the relation between these elements, and how they cooperate to solve the problem.
Implication	The Consequences on the non-functional requirements.
	Reliability Safety
Implementation	This part gives the aspects, hints, techniques that should be taken into consideration when implementing the pattern.
Consequences	This part includes the side effects and disadvantages that may appear due to the application of this pattern.
Related Patterns	The related design patterns to this pattern, and the possibility to combine related patterns with this pattern.
Pattern Code	Provides Source Code of this pattern

Figure 1: Design Pattern Template

The proposed design template includes a part for pattern implication on the non-functional requirements such as reliability and safety. In order to add these side effects into the pattern concept, we propose an extended template for an effective design pattern representation for Safety-Critical Real-Time applications.

Section III describes the Architecture of the proposed Model. Section IV provides the implementation details. Section V deals with the Code generation part.

Types of Architectural Design Patterns

Hardware Patterns: It includes the patterns that contain explicit hardware redundancy to improve either reliability or safety. This group contains the following 8 patterns:

- Triple Modular Redundancy Pattern.
- Homogeneous Redundancy Pattern.
- Heterogeneous Redundancy Pattern.
- M-Out-Of-N Pattern.
- Monitor-Actuator Pattern.
- Sanity Check Pattern.
- Watchdog Pattern.
- Safety Executive Pattern.

Software Patterns

- N-Version Programming Pattern.
- Recovery Block Pattern.
- Acceptance Voting Pattern.
- N-Self Checking Programming Pattern.
- Recovery Block with Backup Voting Pattern

Triple Modular Redundancy Pattern (TMR)

Let us consider the “Triple Modular Redundancy Pattern” and then generate the code template using our proposed tool.

Type: Hardware

- 1) *Pattern Name:* Triple Modular Redundancy Pattern (TMR)
- 2) *Other Name:* 2-oo-3 Redundancy Pattern, Homogeneous Triplex Pattern.

3) *Abstract:* The TMR pattern operates three channels in parallel rather than operating a single channel and switching over to an alternative when a fault is detected. By operating the channels in parallel, the TMR pattern detects random faults.

The TMR pattern runs the channels in parallel and at the end compares the results of the computational channels together. As long as two channels agree on the output, then any deviating computation of the third channel is discarded. This allows the system to operate in the presence of a fault and continue to provide functionality.

4) *Context:* The TMR pattern offers an odd number channels that are operating in parallel, and this pattern is used

to enhance reliability and safety in real-time applications where there is no fail-safe state.

5) *Problem:* TMR pattern provides protection against random faults.

6) *Implication:* To enhance reliability and safety.

7) *Implementation:* The development of the TMR pattern is common to replicate the hardware and software to avoid common mode faults so that each channel uses its own memory, CPU and so on.

8) *Consequences:* The TMR Pattern can only detect random faults. High recurring cost because the hardware and software in the channels must be replicated. The TMR pattern is a common one in applications where reliability needs are very high and worth the additional cost to replicate the channels.

9) *Related Patterns:* Heterogeneous redundancy, Homogeneous Redundancy Pattern.

Following Figure 2 details about TMR pattern implementation in UML and sample code in Java.

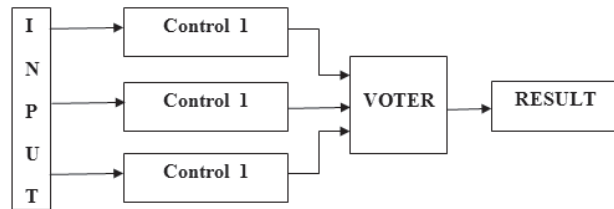


Figure 2: TMR Pattern

The class diagram is given as Figure 3 below.

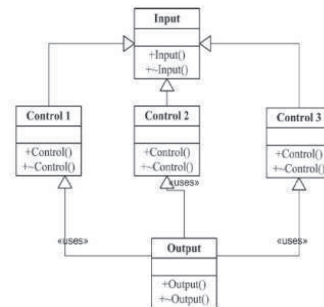


Figure 3: UML - Class Diagram Representation for TMR Pattern

III. ARCHITECTURE

This section describes the architecture that will be used to develop the catalog program. To implement the required features for the current catalog, the software is divided into a set of modules where each module is implemented in a package.

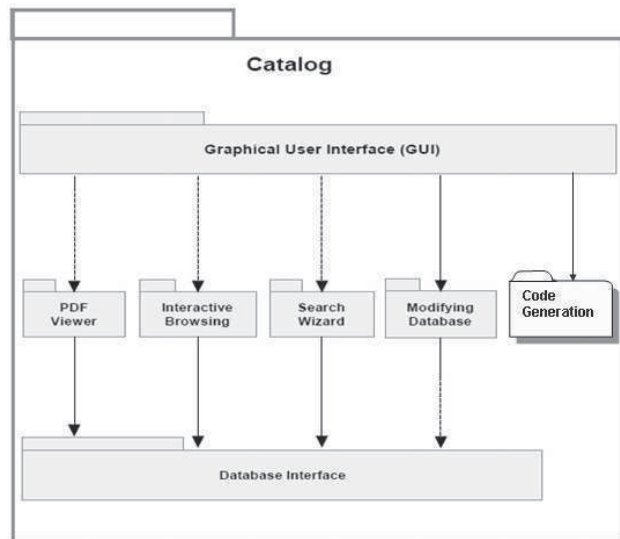


Figure 4: The architecture of the Catalog software

The Graphical User Interface (GUI) represents an intermediate connection between the user and the other modules. It handles the interaction with the user and the application. For example, the interactive browsing interface provides a graphical display for the pattern structure and the ability to navigate the different components, while the PDF Viewer includes a plug-in to display the PDF files.

The PDF Viewer provides a list of PDF files for the available patterns, divided into three groups. Furthermore, it has been implemented using a PDF plug-in, which gives users the ability to browse, save, and print the patterns similar to the original Adobe Acrobat Reader software.

The interactive browsing module provides the second method for pattern presentation. It includes a selective navigation of the contents of the pattern. This feature gives users the ability to select, retrieve, and copy complete information about any part of the selected pattern.

The search wizard module includes the decision support feature provided by this tool. It gives users the ability to find a suitable pattern or a combination of patterns for the desired application by answering questions in an oriented step by step wizard.

This module serves two purposes. First, it allows users to modify the catalog contents, such as the fields of the patterns, solutions, decision points (requirements), problems and decision trees. Second, it provides the functionality to add new elements to the database such as creating a new pattern, a new solution, or a new decision tree. These two features offer an easy way to modify and extend the current catalog.

#### IV. Implementation

##### A. Presentation Layer

The presentation layer allows querying the Database for editing the design patterns and updating the catalog, connecting with the Database locally through JDBC.

Sophisticated interface is provided for the administrators and end user. The proposed application provides assistance to developers and other functionalities dealing with object-oriented framework such as UML representation and code generation for the design patterns. The user interface is a stand-alone application. The following Fig 4 shows an interface for the management of the Design Patterns Catalog. Users can browse the list of patterns. Advanced users may add new patterns.



Figure 5: Interface for Design Pattern Catalog

Here, we believe that modeling of Architectural Design Patterns should be standardized to meet the needs of developers. Our proposed application can generate class diagrams for the specified design pattern and thereby generate the source code for Architectural Design Pattern.

##### B. The Design Pattern Search Wizard and other screens

We collect a set of criteria from the description of Design Patterns and classify by their applicability. In order to extract Patterns from the Database, we opted for a categorization of design patterns. This is mainly restricted to the applicability part of Design Patterns and can easily be extended to cover other literal description parts. The set of criteria and the corresponding keywords database must be thinking out for the related patterns. Our proposed tool provides the end-user with a comprehensive list of keywords, that we have collected, to be used automatically and the tool will help end-users to choose the appropriate Design Patterns.

The Java based search wizard enables the effective search and the selection of suitable Design Patterns with respect to the situations in which to use the desired Design Pattern.

**Filtering:** The first constraint involves the selection of keywords that match the scope of the user query. The search operation intends filtering and refining of the user's ideas in order to reduce the search scope and have closer results to the desired Patterns.

Second constraint, the program will suggest a list of all the situations matching the selected keyword. The user is required to read the criteria and select those that best describe the situations he queries for. By checking appropriate statements the user is ready to generate the suitable Design Patterns.

The following figure 6 is the Design Pattern search wizard interface.

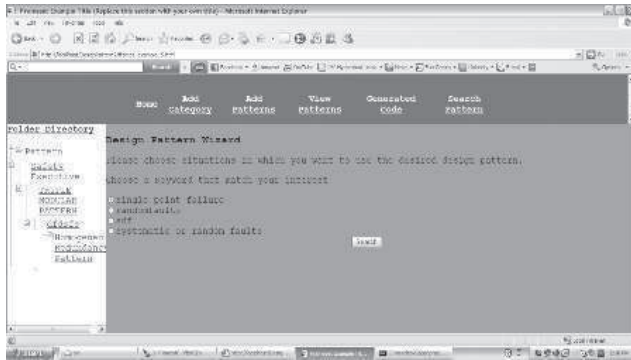


Figure 6: Design Pattern Search Wizard interface

This is how the login page will look like, as shown in figure 7.

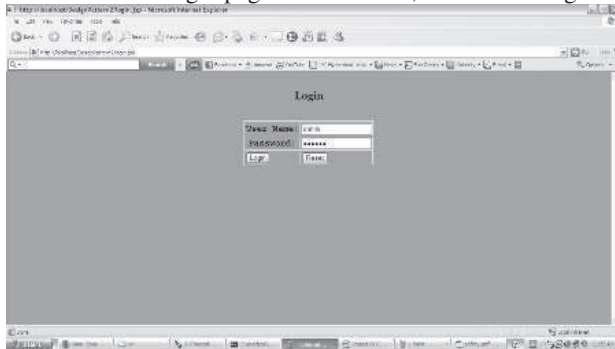


Figure 7: Login Page.

The users can also register; the “PDF conversion” page as given in figure 8.

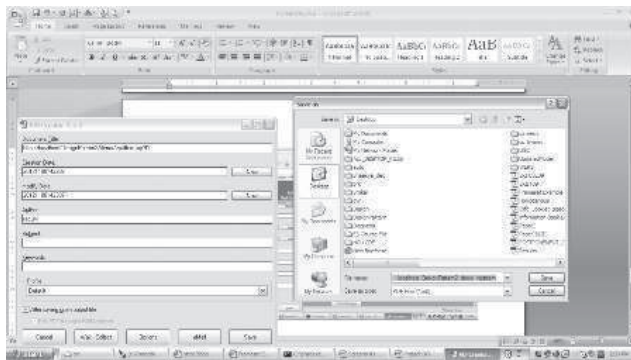


Figure 8: PDF Conversion

The home screen of the Admin is given in figure 9.

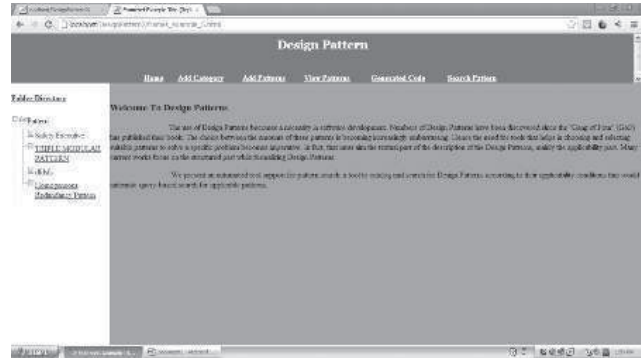


Figure 9: Home Admin

Code generation output screens are given in the following figures 10. The code is generated using the tool.

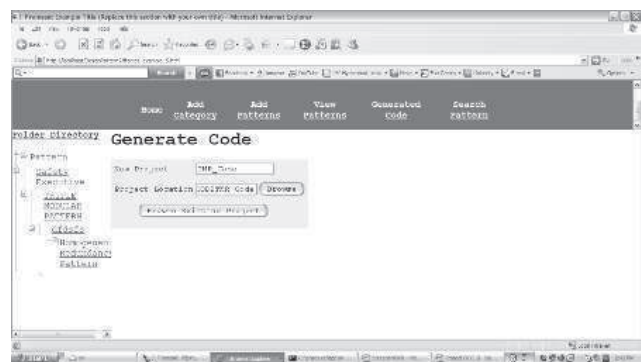


Figure 10: Code Generation

As specified in figure 10, we can give the class details; attribute details and operation details in the tool.

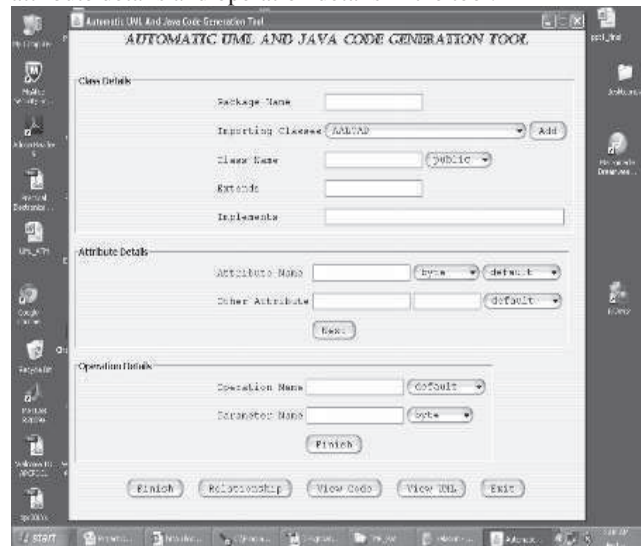


Figure 11: Automatic UML and JAVA Code generation tool.

C. Relationship

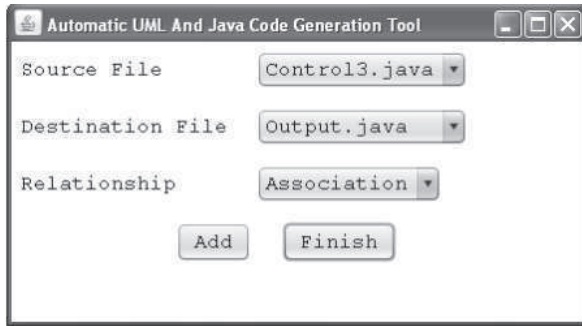


Figure 12: Relationships within Classes

The above Figure 12 allows the user to maintain the relationships with the given classes.

Source Class and Destination class are selected along with the Relationship (i.e., Association, Aggregation and Generalization etc).

V. CODE GENERATION FOR TRIPLE MODULAR REDUNDANCY(TMR) PATTERN

Fig. 13 is the UML diagram for the TMR pattern. In Fig 14 View Code, the Java file is selected to view the Java Code, which consists of Classes, Attributes and Methods. Skeleton Code will be provided to the user and can be customized/enhanced in future as per the requirements.

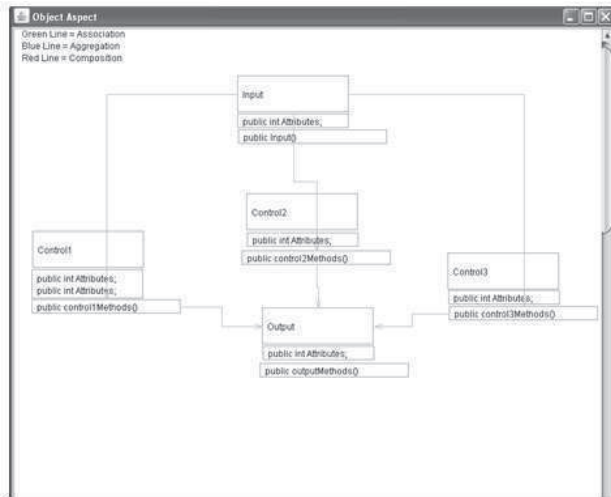


Figure 13: UML diagram for TMR Pattern using Proposed Tool.



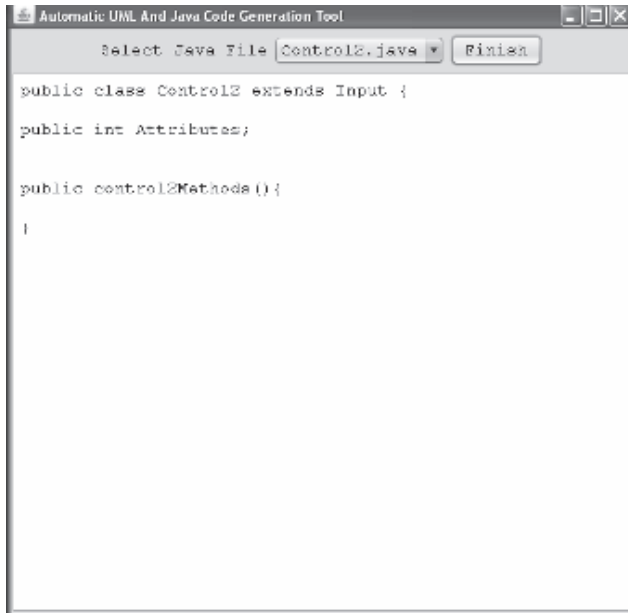
Figure 14: View Pattern Code

The following figures Figure 15, 16, 17 and 18 provide the Skeleton Code (Code Template) for the TMR Pattern.



Figure 15: Code Template for TMR Pattern





```

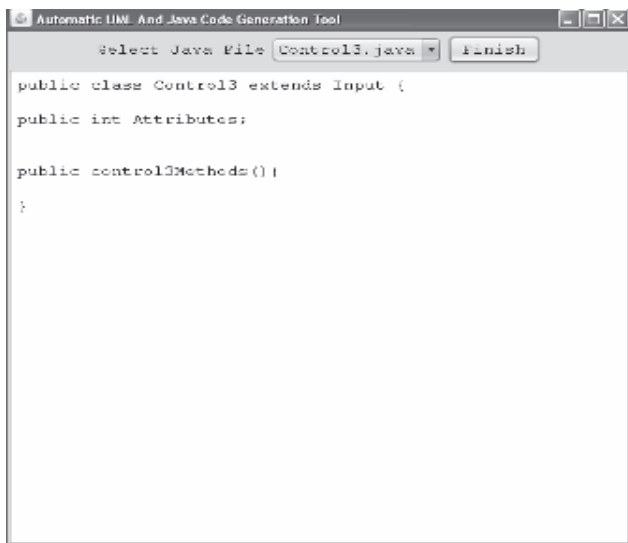
Automatic UML And Java Code Generation Tool
Select Java File Control2.java Finish

public class Control2 extends Input {
public int Attributes;

public control2Methods() {
}
}

```

Figure 16: Code Template for TMR Pattern



```

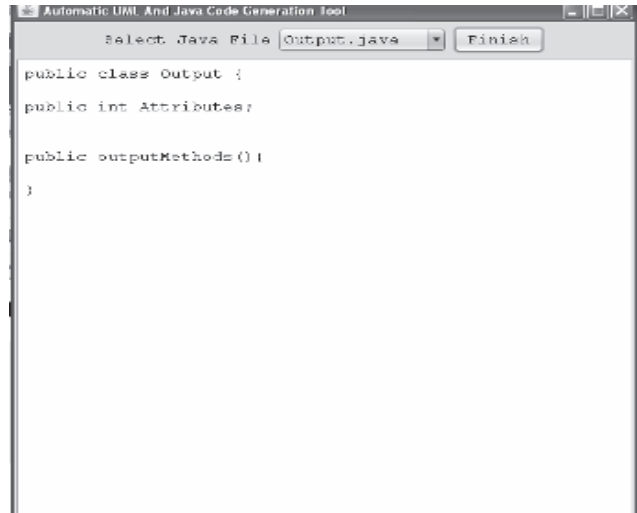
Automatic UML And Java Code Generation Tool
Select Java File Control3.java Finish

public class Control3 extends Input {
public int Attributes;

public control3Methods() {
}
}

```

Figure 17: Code Template for TMR Pattern



```

Automatic UML And Java Code Generation Tool
Select Java File Output.java Finish

public class Output {
public int Attributes;

public outputMethods() {
}
}

```

Figure 18: Code Template for TMR Pattern

View UML link will provide the UML Diagram.

## VI. CONCLUSION

Integration of design patterns is desirable for the successful development of the real-time applications. In order to achieve a successful application, we presently constructed a Design Pattern catalog, where we maintain a collection of design patterns that are commonly used in hardware and software domain. Moreover, the constructed catalog provides an automatic recommendation of suitable design method for a given application.

In order to support the designers, we proposed a tool that suggests a suitable design pattern based on the software characteristics. This tool will be helpful in generating the source code for the suitable design pattern.

## VII. FUTURE ENHANCEMENTS

The work presented in this paper introduces some possible directions for future work. This catalog can be extended to include other design techniques that address the design problems for Safety-Critical Real-Time systems.

The Simulation module can be developed. Therefore, the construction of a comprehensive simulator that provides reliability and safety simulation for all design patterns would be desirable and useful for comparison and assessment.

## ACKNOWLEDGMENT

The authors would like to thank the student Mr. Datta Virivinti of B. Tech. (IT), CVR College of Engineering, for his contribution in developing the interface. The authors would also like to thank the college for providing its amenities.

## REFERENCES

- [1] Design Pattern Representation for Safety-Critical Embedded Systems, Ashraf Armoush, Falk Salewski, Stefan Kowalewski,2009.
- [2] Design Patterns for Safety-Critical Embedded System, Ashraf Armoush, 2010.
- [3] Design patterns to implement safety and Fault Tolerance, Hemangi Gawand, R.S,Mundada, P.Swaminathn, International Journal of Computer Applications(0975 – 8887), Volume 18- No. 2, March 2011.
- [4] Application-Level Fault Tolerance in Real-time Embedded Systems, Francisco Afonso, 2008.
- [5] Design Patterns: Element of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, 2012.
- [6] <http://www.patternrepository.com>
- [7] Real-Time Software Design Patterns, Janusz ZALEWSKI.
- [8] Pattern-Based Architectures Analysis and Design of Embedded Software Product Lines, Public version,EMPRESS,2003.
- [9] Modeling Real-Time applications with Reusable Design Patterns, Saoussen Rekhis, Nadia Bouassida,Rafik Bouaziz MIRACL-ISIMS, Vol. 22, September, 2010.
- [10] A. Armoush, E. Beckschulze, and S. Kowalewski. Safety assessment of design patterns for safety-critical embedded systems. In 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2009). IEEE CS, Aug. 2009