U.V.R. Sarma, Sahith Rampelli, Dr. P. Premchand / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 1, January-February 2013, pp.125-131

A Catalog of Architectural Design Patterns for Safety-Critical Real-Time Systems

[1] U.V.R. Sarma, [2] Sahith Rampelli, [3] Dr. P. Premchand
[1] (CVR College of Engineering, Department of CSE, Ibrahimpatan (M), R.R. District, A.P., India
[2] (CVR College of Engineering, Department of CSE, Ibrahimpatan (M), R.R. District, A.P., India
[3] (Osmania University, Department of CSE, Hyderabad, A.P., India

Abstract

Design patterns have been the target of a great deal of research in the last few years. A design pattern is a general solution to a commonly occurring problem[1]. They are composed of three parts: a problem context, a generalized approach to a solution, and a set of consequences. This paper concentrates on developing a catalog for design patterns for safety-critical real-time systems and allows flexibility to choose, search a design pattern and add more design patterns. To support the designers, a tool is developed to suggest the patterns that are appropriate for the software based on its characteristics/design problems. This tool will be able to help generate the code for the suitable design pattern.

Keywords—Design Pattern, Safety-Critical Real-Time Systems, Non-Functional Requirements, Safety-Critical Systems

I. INTRODUCTION

Design pattern, originally proposed by the architect Christopher Alexander, is a universal approach to describe common solutions to widely recurring design problems. The concept of applying of design patterns is popular in Software domain and Hardware domain.

Design pattern aims at supporting designers and system architects in their choice of suitable solutions for commonly recurring design problems. This concept might also be useful to support the design of Reliable Real-Time systems. Our proposed project provides an extended template for an effective design pattern representation for safety-critical real-time systems. This pattern representation includes the traditional pattern concept in combination with an extension describing the implications and side effects with respect to the non-functional requirements such as reliability and safety. Our paper focuses on the representation of reliability patterns based on specific properties of the real-time systems on the construction of a pattern catalog based on the proposed representation by collecting and classifying commonly used software design methods. Also, it is intended to construct the catalog such that an automatic recommendation of suitable design pattern for given real-time software can be achieved. To support the designers, a tool is developed to suggest the pattern(s) that are appropriate for the software based on its characteristics. This tool will be able to help generate the code for the suitable design pattern.

The UML (*Unified Modeling Language*) provides a notation for design patterns but this notation is targeted primarily towards mechanistic design patterns The UML pattern notation is based on the UML class diagram. The elements of most interest are:

- Pattern
- Class
- Object
- Relation
- Association
- Aggregation
- Composition
- Generalization
- Refinement

In this paper, we will not discuss detailed design patterns in order to limit its size.

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.

Our Proposed tool provides solutions for the following six particular Design Problems:

i) Transient Faults
ii) Hardware Random Faults

iii) Hardware Systematic Faults
iv) Safety Monitoring
v) Time Base Faults (Sequence Monitoring)
vi) Software Faults

Let us consider the "Homogeneous Redundancy Pattern" and then generate the template code using our catalog.

A. Homogeneous Redundancy Pattern

Other Names*:* Switch-To-Backup Pattern, Homogeneous Redundancy Pattern, Standby Spare Pattern, Dynamic Redundancy Pattern, Two Channel Redundancy Pattern.

Type: Hardware Pattern

Abstract*:* An obvious approach to solving the problem of things breaking is to provide multiple copies of that thing. In safety and reliability architectures, the fundamental unit is called a *channel.* A channel is a kind of subsystem, or run-time organizational unit, which is end-to-end in its scope, from the monitoring of real world signals to the control of actuators that do the work of the system. The Homogeneous Redundancy Pattern replicates channels with a switch-to-backup policy in the case of an error.

Context: Homogeneous Redundancy Pattern is primarily a pattern to improve reliability by offering multiple channels. These channels can operate in sequence, as in the Switch To Backup Pattern (another name for this pattern), or in parallel, as in the Triple Modular Redundancy Pattern. The pattern improves reliability by addressing random faults (failures). Since the redundancy is homogeneous, by definition any systematic fault in one copy of the system is replicated in its clones, so it provides no protection against systematic faults (errors).

Problem: The problem addressed by the Homogeneous Redundancy Pattern is to provide protection against random faults-that is, failures—in the system execution and to be able to continue to provide functionality in the presence of a failure. The primary channel should continue to run as long as there are no problems. In the case of failure within the channel, the system must be able to detect the fault and switch to the backup channel.

Structure: The checking components implement a switch-to-backup policy by invoking the other channel when an error is detected in the currently operating channel.

Implication: Safety and reliability in the presence of either systematic or random faults.

Implementation*:* The implementation of this pattern is to remove common fault modes, the computing hardware (CPU, memory, etc.) as well as mechanical systems should be replicated. The only special work is the logic to identify the faults and switch to the alternative channel when a fault is detected.

Consequences*:* The Homogeneous Redundancy Pattern has a number of advantages. It is conceptually simple and easy to design. It provides good coverage for random (that is, hardware and transient) faults, although only if the hardware is itself replicated. It is usually a simple matter to get good isolation of faults and to eliminate common mode faults. The pattern applies when random faults occur at a significantly higher rate than systematic faults, such as in rough or arduous physical environments. It also is useful for safety-critical or high-reliability systems that must continue to operate in the presence of faults. An advantage of this pattern is the low R&D cost – since there is only a single channel to design. Its primary disadvantage is the lack of coverage for systematic faults and increased deployment costs over non-redundant systems. The disadvantages of the pattern are primarily the higher recurring cost and a lack of coverage for systematic faults. Because the electronic and mechanical hardware must be duplicated for maximal coverage, each shipping system must bear the cost of additional hardware components. Furthermore, since the channels are clones, any systematic fault in one channel must, by definition, appear in the other. The pattern runs a single channel and switches over to a backup channel only when a fault is detected. This means that the computation step is lost when a fault is detected and either the data is lost or recovery time to redo the computation must be taken into account in time-critical situations.

Related Patterns*:* Protected Single Channel Pattern, Heterogeneous redundancy patterns, Triple Modular Redundancy Pattern (TMR). To eliminate much of the recurring cost of this pattern, we can use the Protected Single Channel Pattern. To add coverage of systematic faults as well, one of the heterogeneous redundancy patterns can be used. The Triple Modular Redundancy Pattern (TMR) also provides reliability in the presence of random faults, just like the Homogeneous Redundancy Pattern, but TMR does not have to restart a computation when a failure is detected. However, TMR is a more expensive design pattern to apply.

Following Figure1 shows details about Homogeneous pattern implementation in UML.

Architectural Design Patterns should be standardized to meet the needs of developers and readers. We take this opportunity to generate class diagram and thereby generate the source code for Architectural Design Patterns.

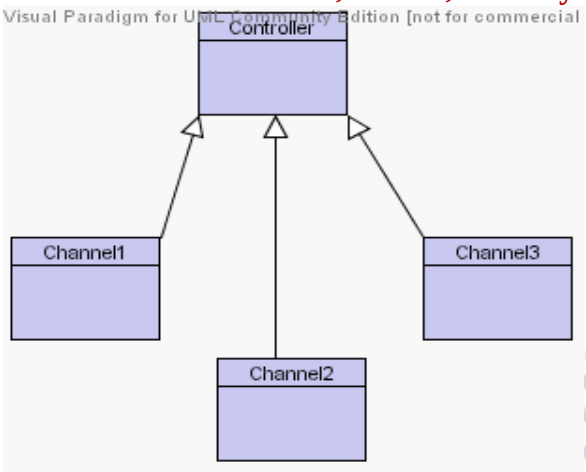

Figure 1: UML - Class Diagram Representation for Homogeneous Redundancy Pattern

II. ARCHITECTURE

This section describes the architecture that will be used to develop the Design Pattern Catalog. To implement the required features for the current catalog, the software is divided into a set of modules.

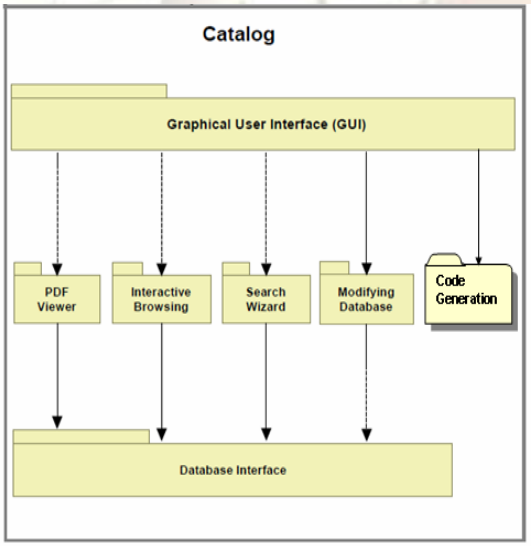

Figure 2: The architecture of the Catalog software

Graphical User Interface (GUI)

The interactive browsing interface provides a graphical display for the pattern structure and the ability to navigate the different components

PDF Viewer

The PDF Viewer provides a list of PDF files for the available patterns. 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.

Interactive Browsing for Design Pattern

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.

Search Wizard:

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.

Modifying Database :

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. These two features offer an easy way to modify and extend the current catalog.

Code Generation:

The construction of a pattern catalog is based on the proposed representation by collecting and classifying commonly used hardware and software design methods. And an automatic recommendation of suitable design methods for a given application. Moreover, it is intended to provide the code for the suitable design pattern for given real-time software

III. IMPLEMENTATION

A. Presentation Layer

The presentation layer allows querying the Database for editing catalog, interacting with the Database locally through a JDBC (Java Database Connectivity) for administrators and end users. It, mainly, provide support assistance to developers and other functionalities dealing with object-oriented side of the framework like UML representation and code generation. The user interface may be a standalone solution. The following Fig 4 shows an interface for the management of the Design Patterns Catalog. Readers can browse the list of formalisms and the corresponding patterns. Users may also add new patterns to the existing Database.

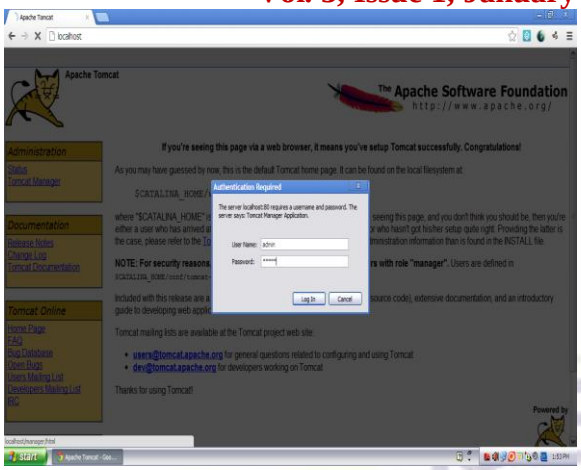

Figure 3: Tomcat 6.0 Admin Page

Figure 3, shows Tomcat Admin page, where our tool is deployed to execute.

The following figure shows the Home page of the catalog.

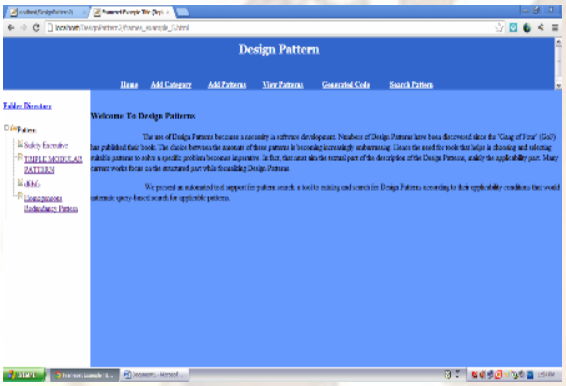

Fig 4: Design Pattern Catalog-Interface

B. Result screens

Our tool is based on the applicability of the description of Design Patterns. We collect a set of criteria from the description of Design Patterns and classified by their applicability. In order to extract Patterns from the Database, we opted for a categorization into a database of pertinent keywords which characterize the statement criteria. 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 enough and enough to cover the most knowledge of Design Patterns. To exploit the platform that we built, the tool provides the end-user with a non-exhaustive 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 wizard is a Java platform which enables the search and the selection of suitable Design Patterns regards to the situations in which to use the desired Design Pattern.

The first constraint involves the selection of keywords that match the scope of the user interest. It's an important phase in which the operation intends the filtering and the refining of the user's ideas in order to reduce the search field and have closely results to the desired Patterns.

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

The following figure 5 is the Design Pattern search wizard interface.

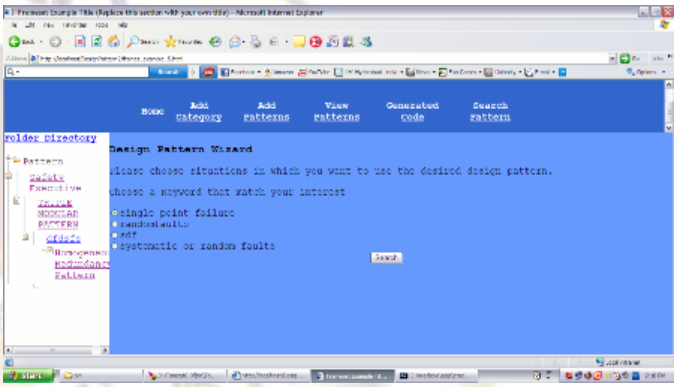

Figure 5: Design Pattern Search Wizard interface

The following figure shows the Add Pattern Page, Where User can add new pattern information, the pattern details are permanently stored in the back-end Database (i.e. Oracle 10g).

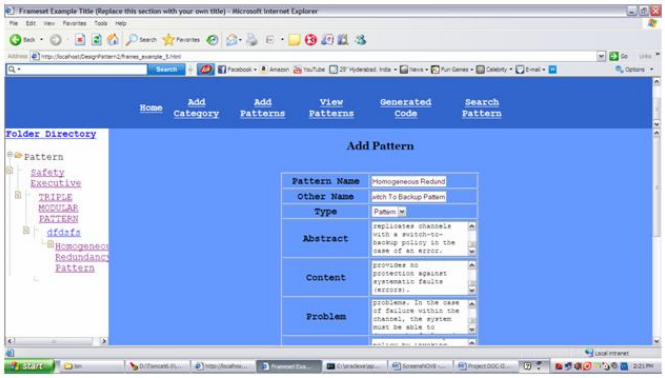

Figure 6: Add Patterns

The Pattern Information can be fetched from the Database by sending an appropriate SQL Query. The Pattern Data can be updated or modified when ever it is required.

The Pattern information can be viewed in two different ways. First, the user can Click on View Patterns link and can find the list of existing patterns. Second, user can browse the pattern by navigating the Tree View Structure. All the pattern are provided in Tree View Format, which is implemented using Java-JTreeView technology.

Figure 8 shows the contents of the pattern. For user's convenience the pattern information can be provided

in PDF format. Our application is rich featured and users can access the patterns very comfortably.

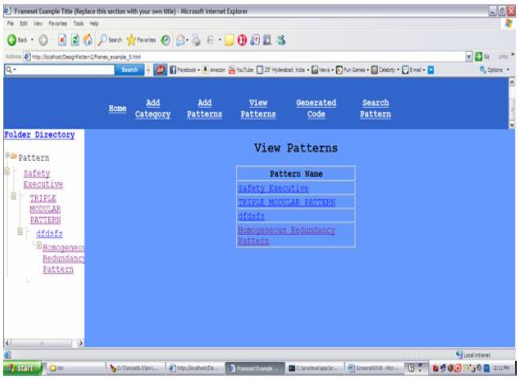

Figure 7 : View Patterns

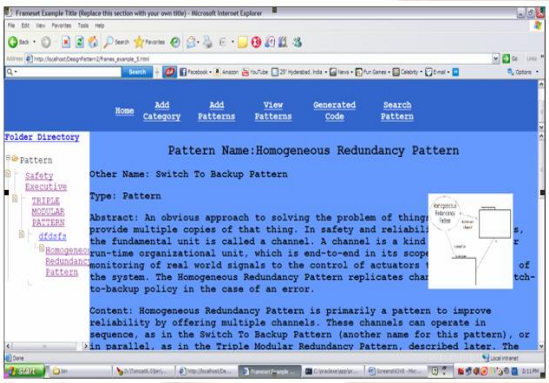

Figure 8. View Pattern Content

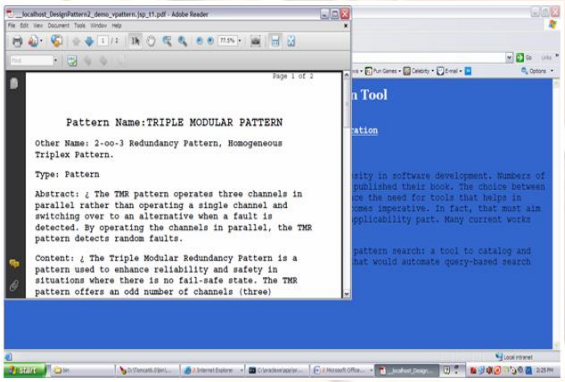

Fig 9: PDF Conversion

Code generation output screens are given below as figure 10 and figure 11. The code is generated using the tool.

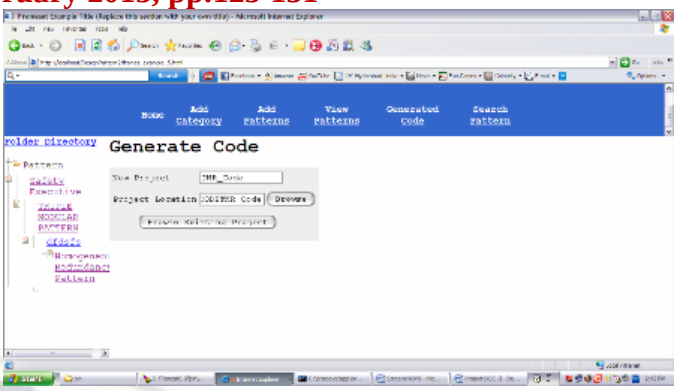

Fig 10: Code Generation

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

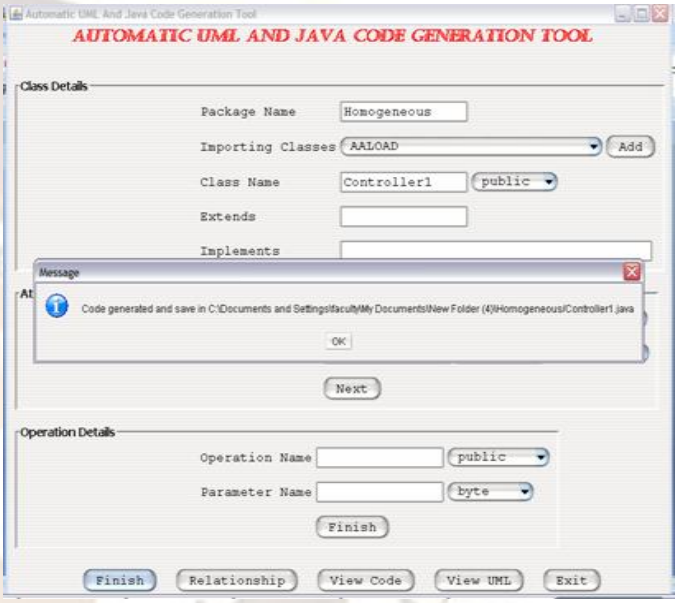

Fig 11: Automatic UML and JAVA Code generation tool.

C. Relationship

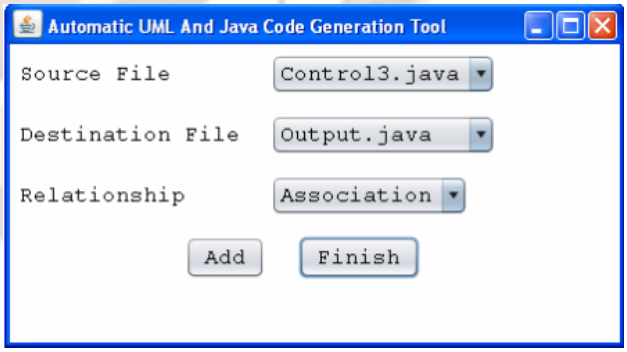

Fig 12: Relationships with in Classes

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

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

IV. CODE GENERATION FOR HOMOGENEOUS REDUNDANCY PATTERN

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.

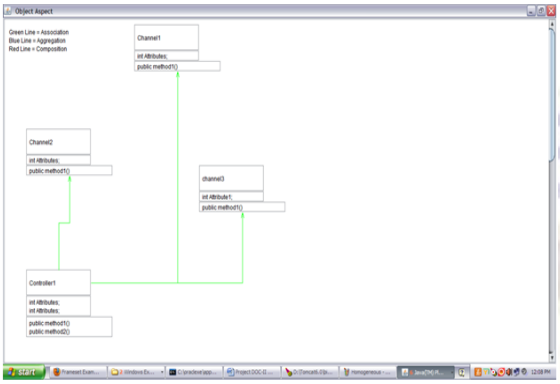

Fig 13: UML – Class Diagram for Homogeneous Redundancy Pattern, Drawn using our proposed tool.

The below figure 15, figure 16, figure 17 and figure 18, provide the Skeleton Code (Code Template) for the Homogeneous Pattern.

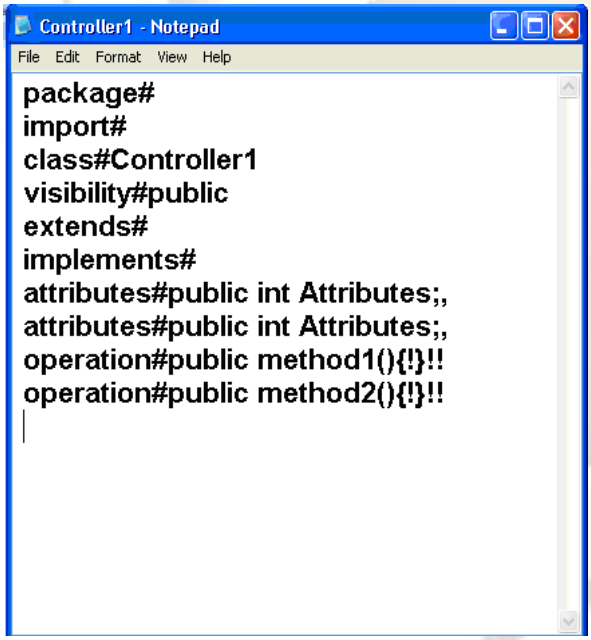

Figure 14: Input text file to Generate Code Template

Figure 14 is a text file, which consists of the meta information of each class. This text file will be submitted to View Code link to generate the code for the Design Pattern.

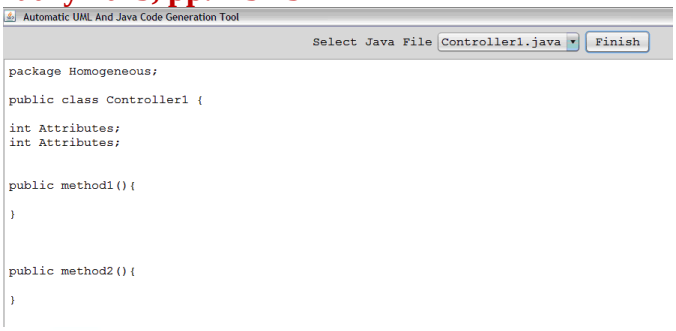

Fig 15: Code Template for Homogeneous Redundancy Pattern

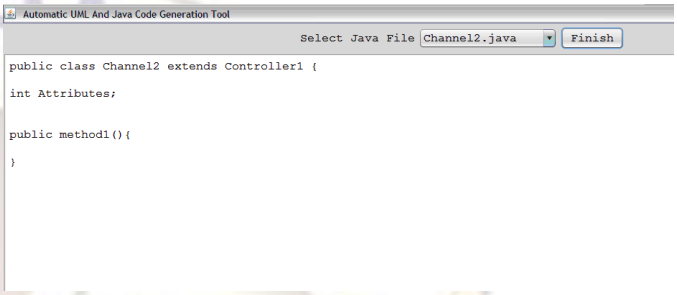

Fig 16: Code Template for Homogeneous Redundancy Pattern

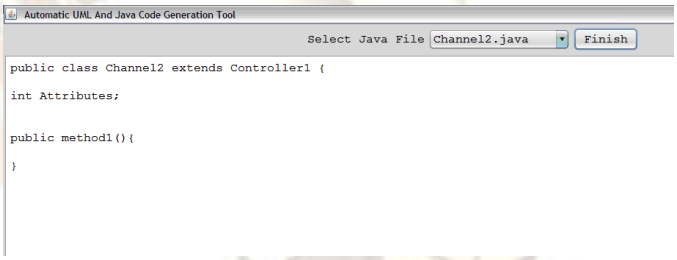

Fig 17: Code Template for Homogeneous Redundancy Pattern

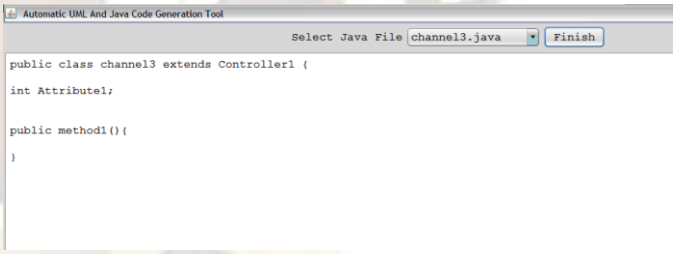

Fig 18: Code Template for Homogeneous Redundancy Pattern

View UML link will provide the UML Diagram.

V. CONCLUSION

We currently constructed a Design Pattern catalog by collecting and classifying commonly used hardware and software design methods. Moreover, automatic recommendation of suitable design methods for a given application is achieved.

In support of the designers, a tool is developed to suggest the patterns that are appropriate for the Real-Time applications based on its Design Problems. This tool will be able to help

the designers by generating the code template for the selected design pattern.

VI. FUTURE ENHANCEMENTS

This Design Pattern catalog can be extended to include more patterns, which addresses the various design problems in the real-time systems.

A Simulation Module extends the capability of our tool to offer desirable comparison and assessment of the Design Patterns.

ACKNOWLEDGMENT

The authors would like to thank CVR College of Engineering, R.R.Dist, for providing its amenities.

REFERENCES

[1] Design Pattern Representation for Safety-Critical Embedded Systems, Ashraf Armoush, Falk Salewski, Stefan Kowalewski, J.Software Engineering & Application,2009, 2:1-12,Published Online April 2009 in SciRes(www.SciRP.org/journal/jsea), Scientific Research Publishing.

[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, University of Minho, published in 2008@IEEE.

[5] Design Patterns: Element of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. Edition published in 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 15 Dec 2003.

[9] Modeling Real-Time applications with Reusable Design Patterns, Saoussen Rekhis, Nadia Bouassida,Rafik Bouaziz MIRACL-ISIMS, Sfax University, BP1088, 3018, Sfax, Tunisia.International Journal of Advanced Science and Technology,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