MAT JOURNALS

Journal of Computer Science Engineering and Software Testing
Volume 3 Issue 2

Diagnose: A Novel Software Design Pattern For Modern Healthcare System

Niranjan R Chougala
Research Scholar, Visvesvaraya Technological University, Belagavi, India.

Dr. Shreedhara K.S.
Professor & Chairman DOS in CS & E,
University BDT College of Engineering – Davanagere, India.

Abstract

In this paper, we give an overview of a software design pattern "Diagnose" designed to ease the development of medical or healthcare software applications. The main originality of this library is to provide high level components for the "Diagnose" pattern, as well as a programming pattern that enables communication and interaction between these components. This pattern primarily focused on healthcare applications but can be further extended to match other domain purposes having similar functionality or actions. Here, "Diagnose" design introduced with standard attributes and functionalities. Software design patterns have shown to be an effective & easier way for the developer to reuse the successful design & architecture and make them more accessible to the architects and developers of the new system. "Diagnose" for medical information system which helps medical practitioners to create, operate and analyse data for the developers. Diagnose allows to design and develop 'right' system without compromising reusability and OOP.

Index Terms *– Healthcare software, Software design pattern, Information system, OOP.*

INTRODUCTION

From the software designer's perspective software is thought of in terms of logical parts, procedures and connections for a given service, whereas Software patterns provide us reusable solved solution. Diagnose provide similar features and services to cater the needs of a healthcare developers and practitioners. It is one level higher than framework and libraries which provides class structures, objects to solve certain problems and overloaded procedures to service the domain needs. It's evident that using COTS (Components off-the-shelf) libraries we write some code against their APIs and get our services from others code. But they don't help us to structure our own applications in terms of understanding, more maintainable and flexibility. Design patters helps to achieve these with ease.

The Diagnose Pattern

The prime focus of the Diagnose pattern is to provide design solution for a medical software development process and assist medical practitioners in diagnosing diseases. It helps in writing software better ways by using best practices and methods. The Diagnose Pattern logically viewed as an offshoot of best domain practices and COTS. The Diagnose features not only helps in monitoring medical data, activating appropriate function and other statistical computations but also helps medical practitioners to visualize the medical data/statistics required for better analysis, which in turn supports to make better decisions & comparisons and collaborations. The entire design process has been followed with Object Oriented concepts and design pattern methodology as proposed by Erich Gamma.

Design of the Diagnose pattern

1. Intent [describes the pattern and what it's supposed to do. This is a short overview of the pattern that someone is supposed to be able to read and understand to know if the pattern solves our problem.]
- Diagnose Pattern is mainly focused on storing and diagnosing medical data.
- It helps medical software developers to standardize and adopt various medical data storage nomenclatures.
- It monitors, examines and detects various diagnostic events.

2. Motivation [lists a scenario that describes the problem the pattern is supposed to solve, using a concrete scenario.]
- Diverse way of storing, retrieving, referencing medical data and software designs.

3. Applicability [describes general scenarios where a pattern can be applied.]
- Medical software design & development.
- Generalizing disease diagnosing process.

4. Structure [shows a diagram indicating how the pattern works.]
- Diagnose pattern has a

i. DiagnoseData Object which monitors current conditions of various parameters like Blood Pressure, Haemoglobin, Body Temperature, RBC Count, WBC Count, etc…
ii. Subject Object which manages data for required query / diseases

1. Malaria Object
2. Dengue Object
3. Fever Object
4. Cholera Object etc…

iii. The Subject Object has to subscribe / register with the Subject to receive updates and results.
iv. Similarly, Subject Object can ask for unsubscribe / unregister with the Subject for stop receiving updates and results.
v. One-to-many relations between a set of objects

5. Participants [lists the classes and objects that take part in the pattern, as well as their roles.]
- Medical subject interface will have

i. registerDiagnose
ii. removeDiagnose
iii. notifyDiagnose

- DiagnoseData interface has

i. Update

- Display interface method for showing all required elements.
- go-between, developers can customize this interface to create their own display elements.
- StatDisplay, displays minimum, average and maximum measurements of the medical parameters.

6. Collaborations [indicates how the participants work together.]
- Diagnose has various participants like register, unregister, update, notify etc… which are loosely coupled so that they provide / share date whenever it got changed some time without the need.
- Unrequired data from the participants can be ignored.

7. Consequences [lists both the good and bad effects of the pattern.]
- Loosely coupled

8. Implementation/Sample Code [outlines the techniques used when implementing this pattern and/or sample code.]
- Code enclosed

9. Known Uses [gives some real-world examples where the pattern has been used.]
- Designing and developing medical software
- Analysing various medical data

10. Related Patterns [lists other patterns that might be related to this one.]
- Observer

- MVC

Diagnose general structure and usage:
Diagnose contains a members responsible for identifying a specific disease and its method will be a disease itself with its related attributes and functionalities. New software which wants to integrate this component has to add as an API or interface to a medical application in order to use it as plug-in.

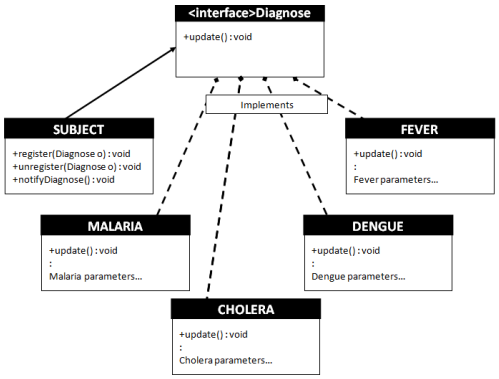

Fig 1 : *Diagnose design structure*

Diagnose has a Subject class to perform register, unregister and notify function to perform basic interface with update() which keeps all members updated. Whichever objects registered with the Diagnose will be notified accordingly with the alert values set. For example, Dengue has specific threshold values to be monitored and this function will be carried out effectively by the DENGUE. Similarly, Fever will get updated values and triggers its threshold values if they are crossed or touches a specific range. The implemented piece of this pattern resides and operates with the other software components to enhance software design pattern approach. Diagnose structure can also be further extended to cater other meaning services for its domain specific or non-domain specific functions. For example, in domain specific function, a new disease can be added to Diagnose with its monitoring parameter in update function and other methods defined separately. Whereas non-domain specific functions like diagnosing a host computer, can be included with its parameters and methods defined accordingly. Thus, Diagnose provides all flexibilities to incorporate various futuristic functionalities and their interfaces.

CONCLUSION

This proposed software design pattern "Diagnose" provides a comprehensive design and implementation guidelines for the medical or healthcare software domain and its related stakeholders. The Diagnose pattern being designed using OO and structurally facilitate to comply with the guidelines of design pattern development. It helps medical system developers to plug-in diseases and their critical parameter, triggers medical practitioners to make better decision. The functionalities of Diagnose can also be altered and used without changing its intent for domain and non-domain purposes. Presently, Diagnose design is focused on healthcare system implementation and interfaces. It can also be extended for any domain with its intent intact.

REFERENCES

1. Interestingness Analysis of Semantic Association Mining in Medical Images - Saritha S. and Santosh Kumar G., ICIP 2011. CCIS 157, K R Venugopal and L M Patnaik (Eds).
2. Reliable Images : the DICOM Standard for Medical Imaging
3. Ant Colony Optimization Algorithm – Nada M. A. AI Salami, UbiCC Journal, Volume 4, Number 3, August 2009
4. Advances in Medical Image Processing (A special issue on the Workshop in Aachen, germany, March 2010) – Thomas, Til Aach, Katrin, Walter, Torsten and Ingrid
5. Kullback-Leibler error Criteria To Reduce the Complexity of NN Applied to traffic sign Recognition – Dr KS

Shreedhar, TY narasimha Reddy – 2nd National Conference on RCCIT'11
6. Iconic Fuzzy Sets for MR Image Segmentation – Wido Menhardt, Philips Research Labs, Hamburg, West Germany
7. Blackboard – Software Architecture in Practice (2nd Edition)- Leb Bass, Paul, Rick Kazman, Pearson education.
8. A pattern Similarity Scheme for Medical Image Retrieval – Dimitris K Iakovidis, Nikos Pelekis, Evangelos, Ioannis, Haralampos and Yannis Theodoridis