ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

An ISO 3297: 2007 Certified Organization *Vol.5, Special Issue 2, April 2017*

An International Conference on Recent Trends in IT Innovations - Tec'afe 2017

Organized by

Dept. of Computer Science, Garden City University, Bangalore-560049, India

Designing Modern Healthcare Software System Using Pattern Approach: A Concept

Niranjan R Chougala, Dr.Shreedhara K.S.

Research Scholar, Visvesvaraya Technological University, Belagavi, India

Professor & Chairman DOS in CS & E, University BDT College of Engineering – Davanagere, India

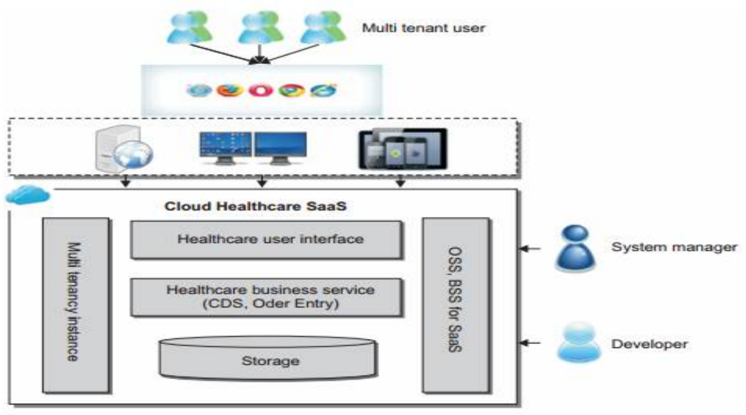
ABSTRACT: Design patterns are optimized, reusable solutions to the programming problems that we encounter every day. A design pattern is not a class or a library that we can simply plug into our system; it's much more than that. It is a template that has to be implemented in the correct situation. It's not language-specific either. A good design pattern should be implementable in most—if not all—languages, depending on the capabilities of the language.

Design patterns are, by principle, well-thought out solutions to programming problems. Many programmers have encountered these problems before, and have used these 'solutions' to remedy them. If you encounter these problems, why recreate a solution when you can use an already proven answer? Let’s consider, Creating successful healthcare software – from patient management systems to medical devices, to electronic medical records – differs substantially from traditional software.

I. INTRODUCTION

Modern Healthcare System:

Healthcare software demands unique domain expertise, project methodology, and software architecture patterns. While many vendors and consultants would insist that good software practices and user-centered design principals are universal across domains, we believe that healthcare projects are different. Healthcare domain expertise, combined with software best practices, can significantly improve project’s chances of success. Developing successful healthcare software substantially differs from traditional enterprise software in that it demands domain specific expertise, project methodology, and software architecture patterns. The risks and costs associated with bringing to market a highly secure, flexible solution, tightly integrated with a range of clinical users’ workflows, is high in terms of the amount and nature of feature preparation and development in a project. Further, product delivery is by no means a guarantee of marketplace success. Consider typical modern healthcare software design;



Patient portals with an intuitive user interface and multiple patient-engagement tools:

- Accessible treatment plans
- Editable personal health records
- Feedback and assessment forms
- Automated appointment scheduling
- Online billing
- E-consultations

Mobile healthcare applications to help patients take even more control over their health trough:

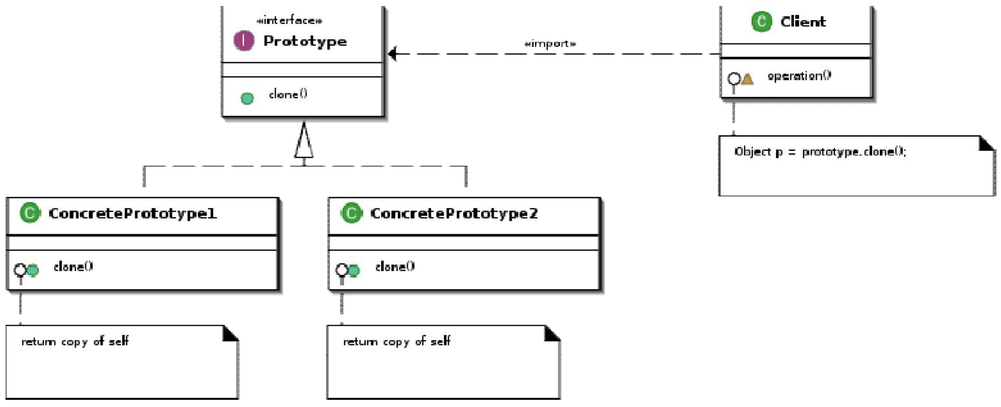
- Quick access to medical data
- Online consultations with a physician
- E-prescribing
- Educational resources in a variety of formats (video, audio and texts)
- Reminders and alerts
- Newsfeeds
- GPS navigation and maps to provide directions to the nearest clinic

A Developer Perspective:

Developer perspective of the new healthcare system would be on:

- Task-oriented approach. To deliver easy-to-use software, we pay a special attention to every app's purpose. Having in mind the requirements of every particular department, we create customized solutions to help your staff fulfil assignments more efficiently
  .
- User-friendly UI/UX design. As healthcare app developers, we create elaborated UX patterns. Thanks to a comprehensive interface and smooth navigation, your employees will perform the needed tasks without going through multiple screens, which is exhausting and time-consuming.
- Integration with multiple internal systems. To bring additional value to your organization, we make the mobile application a gateway to other solutions in your infrastructure, including EHR, practice management, scheduling, revenue cycle management and other systems.
- Security assurance. We understand the importance of data protection in healthcare mobile app development. We make sure that only authorized personnel access clinical information – within the scope of their authority.

Pattern based healthcare software design practice typically shown using the following UML.



The language of the programming team using patterns is mysterious and magical, almost like incantations spoken in some artful black language. Many computer science instructors contend with conviction that the teaching of patterns and the learning of them speeds the learner’s adoption of the principles of object oriented software technology. It is undeniable that the learning of patterns improves the programmers’ development vocabulary.

Software design patterns also help in finding appropriate objects, in determining the apropos object granularity and in designing a software system that is architected from the outset to better adapt to change. At the design level, patterns enable large-scale reuse of software architectures by capturing the expert knowledge of pattern based development and distributing it throughout the development team.

It is generally acknowledged that these are the two most important benefits: the way in which they form a vocabulary for articulating design decisions during the normal course of development conversations among programmers. This can also come into play during the close programming work of so-called “pair programming“, among those who have found it to be useful for them.

When you are working with a group of programmers who are either working in pairs or as part of a group using pattern-based development, you frequently hear talk like “I think we need a strategy here”, or, from one programmer to the rest of the group, “Let’s implement this functionality as an Observer”.

Programmers’ familiarity with pattern-based development has also become a kind of hiring shorthand. Whenever a talented programmer leaves a software development team I am leading, and we need to replace him or her with anther programmer, I use the “Do we need a programmer familiar with design patterns” question as a line of demarcation for recruiting and hiring decisions. The answer is *not* always to hire an expensive programmer intimately familiar with design patterns, either.

A pattern is a problem-solution pair that can be applied in a similar fashion in new contexts; the pattern is complete with advice on how to apply it in the new context. It is important to note that the formal definition of a pattern is not consistent in the literature.

There are three types of patterns:

1. An *architectural* pattern occurs across software subsystems.

2. A *design* pattern occurs within a subsystem but is independent of the language.

3. An *idiom* is a low-level pattern that is programming language-specific.

Each individual pattern is compromised of four elements:

1. A name. Some of the names of the software design patterns can be rather whimsical: “flyweight”, and “singleton”. The whimsy is to serve the purpose of making the patterns memorable to programmers.

2. A problem description. The problem part of the pattern describes the problem and its context, as well as specific design issues such as how to represent algorithms as objects. The problem statement may also speak about when it is best to apply this particular pattern and may also describe class structures that are symptoms of an inflexible software design.

3. A solution to the problem. The solution part of the design pattern does not describe any one particular concrete design or implementation, but only describes the elements that make up the design. The solution only provides a general arrangement of objects and classes which can be used to solve this type of problem.

4. The **consequences** of the solution. This part of the design pattern describes the results and inherent risks and trade-offs associated with applying this particular design pattern. It may include the impact of this design pattern on space and time, programming language and implementation issues, or include notes on software flexibility, system extensibility, and portability. These consequences are critical for evaluating alternative software design patterns.
Be cautious, before using Software design patterns:
Well, actually, there are several drawbacks to all of this talk of pattern-based software development.
One of the main drawbacks, and one of the most important thing for technical project managers and business stakeholders as well as senior managers to keep in mind, is that patterns do not lead to direct software reuse.
Direct reuse of sections of software code is for software libraries. Patterns do not create or promote software libraries of reusable plug-and-play software code, but rather lead to reusable design, architectures and techniques which can be converted by computer programmers into unique program code.

Even though the cutesy names of software design patterns may lead you to believe that they are also simple to learn, they are not. It is easy enough to master some of their names, and to also memorize their structure visually, but it is not very easy to see how they can lead to actual design solutions. This can take even very experienced computer programmers years and years of practice, education and working experience.

Integrating the use of software patterns into an actual, real-world development organization’s daily development life and regular deployment cycle can be a daunting task. The integration, aside from the demands the aforementioned education and training can take on a development staff compromised of computer programmers unfamiliar with the software design patterns described above, is a very labour-intensive activity.

A software development team’s programmers may experience pattern overload, whereby in their unending quest to use pattern-based techniques, they have become an obsession rather than as an effective and efficient means to an end. Aa mentioned above, software design patterns are no silver bullet, and do not lead to direct code reuse, but rather provide another approach to systematically solving software design problems that are commonly and frequently encountered by software development teams.

II. CONCLUSION

The present need for the healthcare software greatly depend on modern needs and rapid application development. Pattern approach surely supports these kinds of requirements and support for the software evolution. It is highly recommended to adopt pattern based software design to make software future compactable.

REFERENCES

[1]. K. T. Gribbon, D. G. Bailey, C. T. Johnston, "UsingDesign Patterns to Overcome Image ProcessingConstraints on FPGAs", Institute of InformationSciences and Technology Massey University, PrivateBag 11 222, Palmerston North, New Zealand. 2006.
[2]. Gribbon, K. T. and Bailey, D. G., “A Novel Approachto Real-time Bilinear Interpolation,” Second IEEEInternational Workshop on Electronic Design, Testand Applications, Perth, Australia, pp. 126-131, Jan,2004.
[3]. Gribbon, K. T., Johnston, C. T., and Bailey, D. G., “ARealtime FPGA Implementation of a Lens DistortionCorrection Algorithm with Bilinear Interpolation,”Proc. Image and Vision Computing New Zealand,Massey University, Palmerston North, New Zealand,pp. 408-413, Nov, 2003.
[4]. Deepak Alur, John Crupi and Dan Malks, “Core J2EE”patterns, Second Edition, 2003.

[5]. Douglas C. Schmidt, "Using Design Patterns toDevelop Object-Oriented Communication SoftwareFrameworks and Applications", WashingtonUniversity, St. Louis.
[6]. Tom Fischer, John S, Pete S, Chaur G Wu"Professional Design Patterns in VB.NET, BuildingAdaptable Applications", Wrox Press, 2002.[7]. Gama, Helm, Johnson, Vlissides, Design PatternsElements of Reusable Object-Oriented Software,Addison Wesley, 1995,B. Cheng – Michigan StateUniversity.
[7]. Niranjan R Chougala&Dr.Shreedhara K.S. Professor & Chairman DOS in CS & E, University BDT College of Engineering – Davanagere, Karnataka, Application building concepts in medical image processing using software design patterns, International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)