Concept of Designing Medical Software using Design Pattern

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.

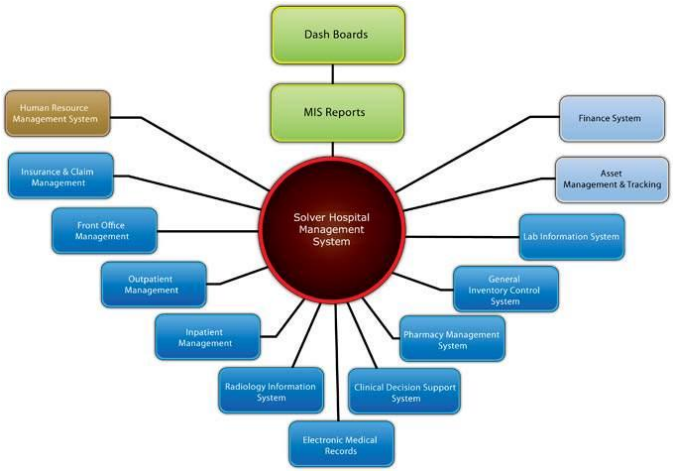
INTRODUCTION

Considering a modern design and development of medical software systems, a concept of design patterns application is proposed and developed to enhance various software design attributes and qualities. Configuration examples are enhanced, reusable answers for the programming issues that we experience each day. A plan example is not a class or a library that we can essentially connect to our framework; it's a great deal more than that. It is a format that must be actualized in the right circumstance. It's not dialect particular either. A decent outline example ought to be coded in most—if not all—dialects, contingent upon the abilities of the coding dialect. Configuration examples are, by guideline, well-thoroughly considered answers for programming issues. Numerous software engineers have experienced these issues before, and have utilized these "arrangements" to comprehend them. In the event that you experience these issues, why reproduce an answer when you can utilize an officially demonstrated arrangement? How about we consider, making effective social insurance programming – from patient administration frameworks to medicinal gadgets, to electronic restorative records – varies considerably from customary programming.

Keywords: Software Design Patterns, Medical Software, Software resuse.

MEDICAL SOFTWARE SYSTEMS

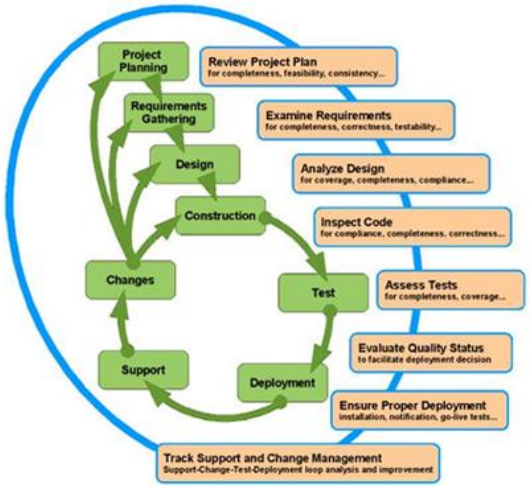
Medicinal services programming requests select space skill, extend practice, and programming engineering shapes. While numerous sellers and specialists would demand that great programming practices and client focused plan principals are worldwide crosswise over spaces, we trust that social insurance tasks are distinctive. Social insurance area ability, joined with programming best practices, can essentially enhance venture's odds of progress. Creating effective medicinal services programming generously varies from customary venture programming in that it requests area particular mastery, extend system, and programming engineering designs. The dangers and expenses related with conveying to advertise an exceedingly secure, adaptable arrangement, firmly coordinated with a scope of clinical clients' work processes, is high regarding the sum and nature of highlight readiness and improvement in a venture. Assist, item conveyance is in no way, shape or form a certification of commercial center achievement. Consider a general Healthcare Management systems with related services.



The goals of such Healthcare Systems includes the following

- Accuracy.
- Reliability.
- Better Patient Data Security.
- Enhancing Patient Safety.
- No Redundancy
- Easy retrieval of Stored Information.
- Improved Patient Services.
- Reduces Paper Work.
- Improving Hospital Inventory Management.
- Optimal Utilization of Resources.

A software development methodology adapted for a medical software development and its related processes is shown:



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

- Handy treatment plans
- Modifiable personal health records
- Opinion and evaluation forms
- Electronic appointment management
- E-billing
- E-consultations

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

- Quick access to medical data
- E-consultations with a doctor
- E-prescribing
- Educational materials in audio, video and text forms
- Notices and alerts
- Newsfeeds
- GPS navigation and maps to provide directions to the nearest clinic

HEALTHCARE SOFTWARE DEVELOPEMENT:

Developer viewpoint of the new healthcare system would be on:

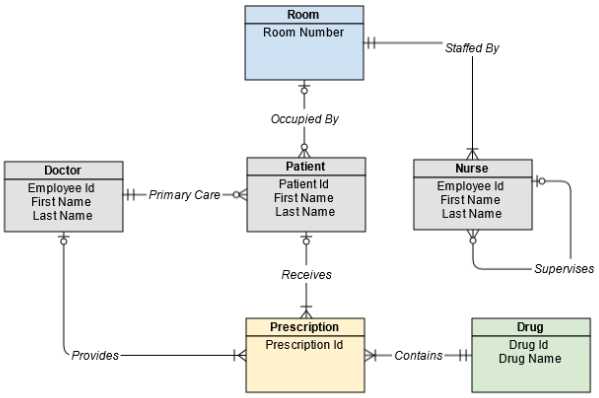
•Task-situated approach. To convey simple to-utilize programming, we give careful consideration to each application's motivation. Having as a main priority the prerequisites of each specific office, we make redid answers for help your staff satisfy assignments all the more productively.

•User-accommodating UI/UX plan. As medicinal services application engineers, we make explained UX designs. On account of a far reaching interface and smooth route, your representatives will play out the required assignments without experiencing numerous screens, which is depleting and tedious.

•Integration with numerous inward frameworks. To convey extra an incentive to your association, we make the versatile application a door to different arrangements in your foundation, including EHR, rehearse administration, booking, income cycle administration and different frameworks.

•Security confirmation. We comprehend the significance of information security in human services portable application improvement. We ensure that lone approved faculty get to clinical data – inside the extent of their power.

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



All things considered, creating programming is extremely troublesome, and creating programming that can be effectively reused is considerably harder. the outlines for areas of programming code ought to be sufficiently general answers for have the capacity to address future issues and prerequisites adaptably while as yet being sufficiently particular with a specific end goal to address the present issue nearby. Developers that are experienced at planning programming frameworks know not to outline their framework utilizing one-off issue arrangements, and rather reuse designs that they have developed acquainted with through earlier use in comparable circumstances and situations and reuse these arrangements as a reason for their new outlines. The essential thing standard of programming designing known as the "Rule of sweeping statement" predicts and energizes this conduct or a certain something, it is totally intriguing to sit in a meeting room with a gathering of developers who have been working all together on a product advancement extend utilizing designs for a couple of months. The rate of data trade is to a great degree high, with a thought said by one developer, and a couple others all the while completing the main software engineer's sentence with a shouted, harmony word like "Scaffold!", and after that one of them jotting lines of code wildly on the whiteboard as the rest gesture in compliment.

The dialect of the programming group utilizing examples is puzzling and enchanted, practically like mantras talked in some guileful dark dialect. Numerous software engineering educators fight with conviction that the instructing of examples and the learning of them speeds the learner's reception of the standards of question arranged programming innovation. It is unquestionable that the learning of examples enhances the software engineers' improvement vocabulary.

Programming configuration designs likewise help in finding proper articles, in deciding the relevant protest granularity and in outlining a product framework that is architected from the beginning to better adjust to change. At the outline level, designs empower huge scale reuse of

programming structures by catching the master information of example based improvement and dispersing it all through the advancement group.

It is for the most part recognized that these are the two most essential advantages: the path in which they frame a vocabulary for articulating outline choices amid the ordinary course of advancement discussions among software engineers. This can likewise become possibly the most important factor amid the nearby programming work of alleged "combine programming", among the individuals who have observed it to be helpful for them.

When you are working with a gathering of software engineers who are either working in sets or as a feature of a gathering utilizing design based improvement, you much of the time hear talk like "I think we require a methodology here", or, from one developer to whatever is left of the gathering, "We should actualize this usefulness as an Observer".

Software engineers' commonality with example based improvement has additionally turned into a sort of contracting shorthand. At whatever point a skilled developer leaves a product improvement group I am driving, and we have to supplant him or her with anther software engineer, I utilize the "Do we require a developer acquainted with configuration designs" address as a line of division for enlisting and employing choices. The appropriate response is *not* dependably to enlist a costly developer personally comfortable with configuration designs, either.

An example is an issue arrangement match that can be connected in a comparative mold in new settings; the example is finished with exhortation on the most proficient method to apply it in the new setting. Note that the formal meaning of an example is not steady in the writing.

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 desibe 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:

All things considered, really, there are a few downsides to the greater part of this discussion of example based programming advancement.

One of the primary disadvantages, and a standout amongst the most imperative thing for specialized venture administrators and business partners and also senior supervisors to remember, is that examples don't prompt direct programming reuse.

Coordinate reuse of areas of programming code is for programming libraries. Examples don't make or advance programming libraries of reuable attachment and-play programming code, but instead prompt reuable outline, designs and methods which can be changed over by PC developers into one of a kind program code.

Despite the fact that the cutesy names of programming configuration examples may persuade that they are additionally simpe to learn, they are most certainly not. It is sufficiently simple to ace some of their names, and to likewise remember their structure outwardly, yet it is not simple to perceive how they can prompt genuine outline arrangements. This can take even exceptionally experienced PC software engineers forever and a day of practice, training and working background.

Coordinating the utilization of programming examples into a real, true advancement association's every day improvement life and normal organization cycle can be an overwhelming errand. The joining, beside the requests the previously mentioned instruction and preparing can go up against an improvement staff bargained of PC developers new to the product configuration designs depicted above, is an extremely work escalated action.

A product advancement group's software engineers may encounter design over-burden, whereby in their unending journey to utilize design based procedures, they have turned into a fixation instead of as a compelling and proficient unfortunate obligation. Aa said above, programming configuration examples are no silver projectile, and don't prompt direct code reuse, but instead give another way to deal with efficiently tackling programming plan issues that are generally and every now and again experienced by programming improvement groups.

CONCLUSION

Softwae deign patterns are the most convinent ways of designing modern healthcare systems in a time critical environment along with changing requirement in near future. 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, "Using Design Patterns to Overcome Image Processing Constraints on FPGAs", Institute of Information Sciences and Technology Massey University, Private Bag 11 222, Palmerston North, New Zealand. 2006.
[2] Gribbon, K. T. and Bailey, D. G., "A Novel Approach to Real-time Bilinear Interpolation," Second IEEE International Workshop on Electronic Design, Test and Applications, Perth, Australia, pp. 126-131, Jan, 2004.
[3] Gribbon, K. T., Johnston, C. T., and Bailey, D. G., "A Realtime FPGA Implementation of a Lens Distortion Correction 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 to Develop Object-Oriented Communication Software Frameworks and Applications", Washington University, St. Louis.
[6] Tom Fischer, John S, Pete S, Chaur G Wu "Professional Design Patterns in VB.NET, Building Adaptable Applications", Wrox Press, 2002. [7]. Gama, Helm, Johnson, Vlissides, Design Patterns Elements of Reusable Object-Oriented Software, Addison Wesley, 1995,B. Cheng – Michigan State University.
[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)
[8] Advances in Medical Image Processing (A special issue on the Workshop in Aachen, germany, March 2010) – Thomas, Til Aach, Katrin, Walter, Torsten and Ingrid
[9] 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
[10] Iconic Fuzzy Sets for MR Image Segmentation – Wido Menhardt, Philips Research Labs, Hamburg, West Germany
[11] Blackboard – Software Architecture in Practice (2nd Edition)- Leb Bass, Paul, Rick Kazman, Pearson education.
[12] A pattern Similarity Scheme for Medical Image Retrieval – Dimitris K Iakovidis, Nikos Pelekis, Evangelos, Ioannis, Haralampos and Yannis Theodoridis