International Journal of Computer Science and Business Informatics

IJCSBI.ORG

Software Architectural Pattern to Improve the Performance and Reliability of a Business Application using the Model View Controller

G. Manjula

Assistant Professor,
Research Scholar, Dept. of Computer Science& Engineering
Shirdi Sai Engineering College, VTU. Bangalore, India.

Dr. G. Mahadevan

Principal, Annai College of Engineering, T.N.
Guide, Dept. of Computer Science& Engineering
Visvesvaraya Technological University, Belgaum, Karnataka, India.

ABSTRACT

In recent time, several new methods have been developed at a rapid pace. Some of the advancements in continuous years, new methods have been developed at a rapid pace. Some of the advancements in continuous optimization methods have been focused on comparison and contrasting nature of Evolutionary Algorithms and Gradient based methods. As a matter of fact, an Evolutionary algorithm is one of the best methods available for derivative – free optimization on higher dimensional problems. This approach will surely make difference in the existing system, whereas the measuring metrics software platform varies in each application. Our approach applies to software architectures modelled with the Palladio Component Model. It supports quantitative performance, reliability, and cost prediction and can be extended to other quantitative quality criteria of software architectures. By adding a new component model in between the each system is more effective in measuring and easily suitable in any business application. In Software life cycle, the two key activities involved are Requirements Engineering and software architecting researchers are emphasing on mapping and transformation of requirements to software architecture, but the lack of effective solution is still prevalent.

Keywords

Evolutionary Algorithm, PCM, Software Architecture, MVC.

1. INTRODUCTION

Palladio component model is a model which acts like Meta model for the designed application, where we can measure the performance, cost and reliability of a system .PCM is one of the high level design structure where

software process can interface with the frame work. The software process for measuring the performance, reliability and cost we can use the software MAT Lab. The mat lab is one of the tools we can implement in the business application and measuring metrics. PCM can be used many ways. Prediction methods for performance and reliability of general software systems are still limited and rarely used in industry Component developers who produce components that are assembled by software architects and deployed by system allocators. The diverse information needed for the prediction of extra-functional properties is thus spread among these developer roles. PCM can also be used based on the different data set, where the behavioral skills of a data are data integrity. The each behavior of a data can be put into sequence diagram and traced out. But some of the features are dependent then different methodology are used some of the methodology are Parametric dependencies, Branch conditions, Loop iterations, Parametric resource demand. Some properties of patterns for software Architecture are.

1. The existing Patterns document is well structured and designed so that based on the business application it makes easy to adopt practically.
2. Each pattern will be suitable for the application either it is complex or easy, if it is so we can also make new pattern by using heterogeneous software architectures.
3. Patterns are the best methodology to apply any business application to measure the metrics of a system.

The following listing helps to classify the Palladio Component Model (PCM) which is underlying the Palladio approach. In case you are preparing taxonomy or try to identify whether specific features are supported by the PCM, this page assists our work.

Supported quality dimensions

- Performance
- Reliability
- Costs
- Maintainability

Requirements engineering and software architecting are two important activities in software life cycle. Requirements engineering is concerned with purposes and responsibilities of a system. It aims for a correct, consistent and unambiguous requirements specification, which will become the baseline for subsequent development, validation and system evolution. In contrast, software architecting is concerned with the shape of the solution space. It aims at making the architecture of a system explicit and provides a blueprint for the succeeding development activities. It is obvious that there exist quite different perspectives in user (or customer) requirements and software architecture (SA).

In this method, the concept of using feature model for requirements engineering was introduced. As a main activity in domain modeling, feature analysis is intended to capture the end-user's (and customer's) understanding of the general capabilities of applications in a domain requirements engineering and software architecting

Some of the IEEE standards are

1. To solve a problem a constraint is needed so that achieving an objective of a problem is considerable.
2. The constraint or condition must possess the quality of interaction with the system component model to satisfy the standard specification or other formally imposed document.
3. A document must reflect the above mentioned stages, and then we can ensure that the architecture has achieved the objective of a problem.

SA has become an important research field for software engineering community. There exists a consensus that for any large software system, critical situation a high level of computational elements are needed to design. Because critical situation leads to complexity while other models are process flow in the modified or newly added model controller Palladio components [6] [7].

1.1 Method

The transformation rule is applied to the UML, where each state of a system is ready to accept the query provided by the metrics system. The system will also available in java JSP pages, but the retrieval operation from each page will continuously affect the system. The process flow diagram mentioned in the Palladio component model [11]. The context for the method consists of a requirements specification that is taken as an input to the method and an architectural design generated as output. User interface are prone to change requests. An MVP is a basic platform to extend the performance of a designed system called M-ACCURATE approach. This approach is newly involved in the problem context while processing the sequence diagram. In Sequence diagram each model behaviour can be measured and identification of the weakness of data, to overcome we insert one more new model at the initial stage called ACCURATE model. This way we can improve the performance, cost and reliability of a system compared to the existing methodology.

1.2 M-Accurate Approach

In this section we present the M-ACCURATE approach. The name M-ACCURATE comes from an acronym for' Model A Configurable Code generator Unified with Requirements Analysis Techniques'. As it implies, requirements play an important role in both PIM modelling and platform decision. The key idea is to capture functional and non-functional

requirements into separate artifacts, a PIM and a platform configuration respectively, and join them at the downstream of the development.
Implementation constructs UML models are meant for both logical analysis and physical implementation. Certain constructs represent implementation items. A component model is a basic or the initial level, replacement can be done whenever the demand or the constraint is not fulfilling. In other words, replaceable of a model can be done to improve the performance of a system. It is intended to be easily substitutable for other components that meet the same specification. A node is a run-time computing resource that defines a location. It can hold components and objects. The deployment view describes the configuration of nodes in a running system and the arrangement of components and objects on them, including possible migration of contents among nodes.

In Model organization, Computers can deal with large flat models, but humans cannot. In a large system, the modelling information must be divided into coherent pieces so that teams can work on different parts concurrently. Even on a smaller system, human understanding requires the organization of model content into packages of modest size. Packages are general-purpose hierarchical organizational units of UML models. They can be used for storage, access control, configuration management, and constructing libraries that contain reusable model fragments. A dependency between packages summarizes the dependencies among the package contents. A dependency among packages can be imposed by the overall system architecture. Then the contents of the packages must conform to the package dependencies and to the imposed system architecture.
The model view controller relationship is the best architecture pattern for the designed business application, but the behavior of a model in the OMT diagram of an application is data dependent.to over come from this insert one more new model called M-ACCURE model between the model view controller.as figure 1 shows the interface between mode and view model, the Controller is at the backend of the system, because the controller is coordinating all the models.

1.2.1 model

The model component encapsulates core data functionality. The model is independent of specific output representation or input behavior. When the figure 3 is designed there are basically six models which are designed. They are Model, view, controller, concrete model, concrete view and concrete controller. Each model inherits the sub model called CONCRETE which can be taken as presenter in the figure 1.The technic behind this is the each sub model will be treated as the ACCURATE model such that we can improve the performance, cost and reliability of a system.

1.2.2 View

View components display information to the user. A view obtains the data from the model. There can be multiple views of the model. Different views present the information of the model in different ways. Each view defines an update procedure that is activated by the change propagation mechanism.

View model inherited CONCRETE view model in figure3 which updates the data information between model and controller.

2.2.3 Presenter

The concept represented by bottom-level approach represented by pie charts in application result. Here the result will be simulated output. Presentation model fulfill two different roles Composition and Coordination. Presenter model always defines a structure for interactive systems in the form of a hierarchy of other cooperating models such as Model and view, but in this approach there are two new sub models are also added to overcome with the communication failure. The communication between human and computer will always taking care by the presenter model.

2. SEQUENCE DIAGRAM

The following scenarios depict the dynamic behavior of MVC. For simplicity only one view-controller pair is shown in the diagram.

1. The model instance is created, which then initializes its internal data structures.
2. A view object is created .This takes a reference to the model as a parameter for its initialization.
3. The view subscribes to the change –propagation mechanism of the model by calling the attach procedure. The mechanism is presentation model where it inherits the CONCRETE-model, view and controller.
4. The concrete models continue initialization by creating its controller. It passes references both to the model and view.
5. After initialization in each model, the application begins to process events.

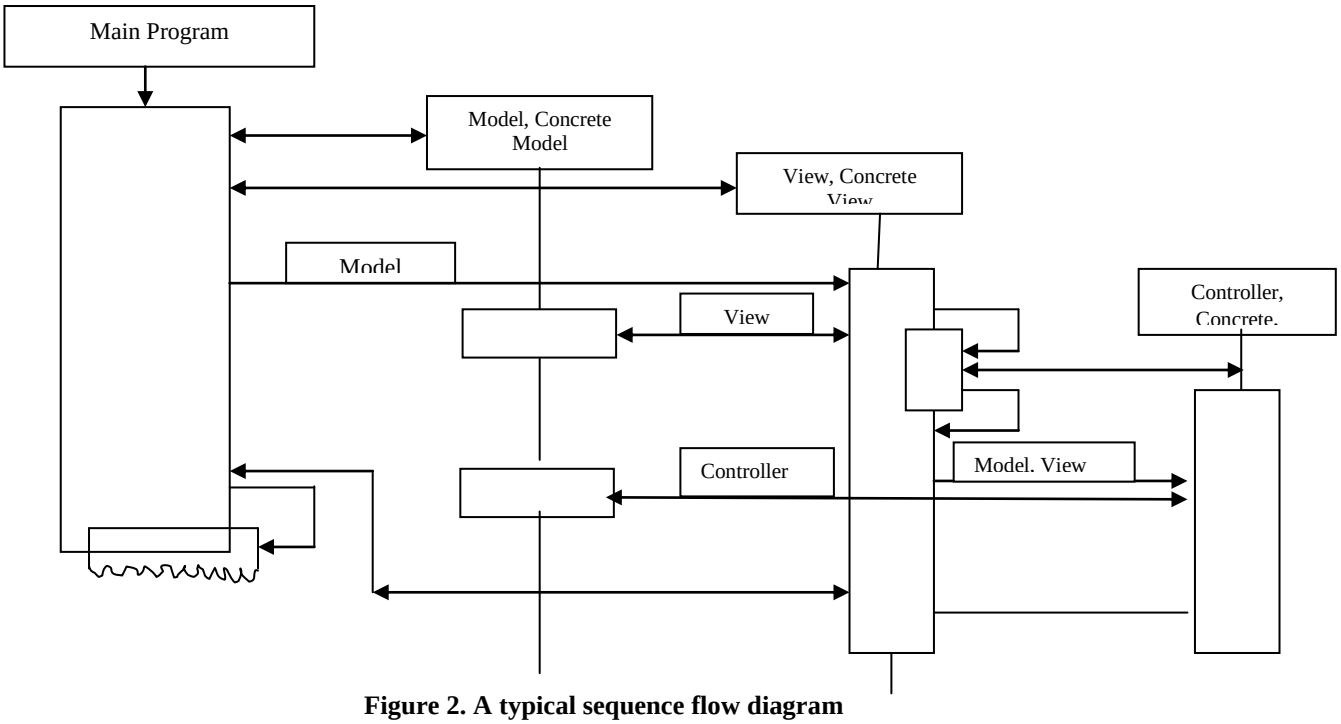

Figure 2. A typical sequence flow diagram

3. IMPLEMENTATION SCENARIO-BASED EVALUATION

It should be noted that it is necessary to use a different scenario set for evaluation than for architecture design. The set used for design is generally supported by the architecture. However, while developing the scenarios, it is not necessary to develop two sets. The two sets could be generated later by randomly dividing the developed set of scenarios. Figure 3 shows a typical Object Management Group (OMG) diagram with asynchronous messages In addition, depending on the system, it might be necessary to develop new scenarios for evaluation purposes if the design is iterated a number of times. In our experience, scenario-based assessment is particularly useful for development Maintainability can be expressed very naturally through change scenarios.

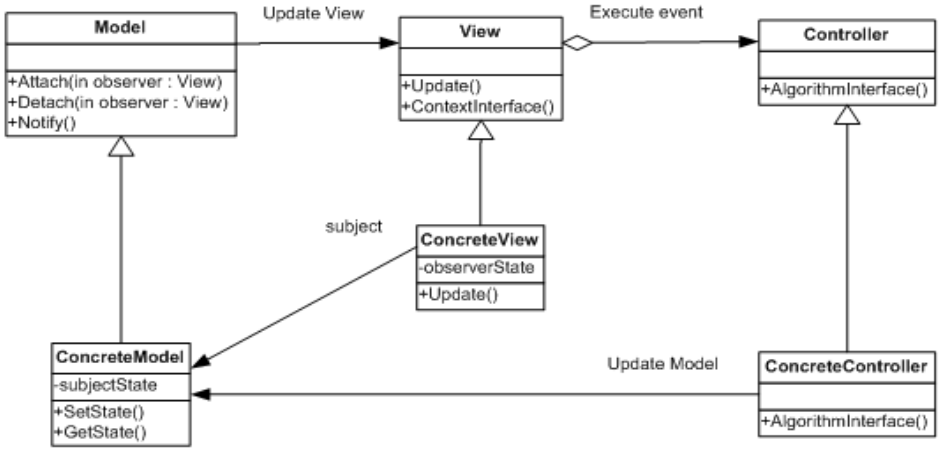

Figure 3. OMG Diagram

4. SIMULATION RESULT

The result of a simulation run contains response time distributions of each executed service. The simulation resolves resource contentions for the service centers either by a FIFO or processor sharing scheduling policy. Further scheduling policies including more realistic schedulers of today's operation systems and multi-core handling will be implemented in the future. The simulation can therefore predict the performance for more complex scenarios than the analytical solver finally; the real system implemented using the code skeletons.

The simulation result will reflect the modification in the measurements taken in the existing model states as each modified or added new model controller will automatically reflect the result and its merits and demerits are better than the existing output [12]. Offers the real performance - no modeling is necessary here. However, in addition to the effort already needed for the prototype to setup and measure the performance figures, the time to implement the system has to be added. This approach of gaining performance values is only applicable in late life-cycles of the software system. When performance problems are discovered after the system has been implemented, a redesign and reimplementation are costly the main components of the architecture are implemented and other components are simulated resulting in an executable system. The context, in which the system is supposed to execute in, could also be simulated at a suitable abstraction level. This implementation can then be used for simulating application behaviour under various circumstances.

```
//code segment highlighted here....
//MODEL main
main(){
Table View *v1=new tableView(&m);
public abstract class Model
   v1->initialize();
BarChartView *v2=new BarChartVeiw(&m);
 V2->initialize();
//now start event processing...
Class Model{
//...continued
 public:
                  void attach(concretemodel *cm){ registry.add(s);}
                  void detach(concretemodel *cm){ registre.remove(s);}
 protected:
                  virtual void notify();
 private:
              Set<Observer*>registry;
};

//model view
 class view :public Observer{
public:
 veiw(model *m):M1model(M1),M1Controller(0)
  {M1model->attach(this);}
  virtual ~View(){M1model->detach(this);}
  virtual void update(){this->draw();}
virtual void initialize();
virtual void draw();
//... to be continued below
//model controlller
 class Controller:public Observer {
 public:
  virtual void ConcreteEvent(Event *){ }
  Controller(veiw *v):V1view){
```

```
   M1model=M1view->getModel();
   M1model=attach(this);
}
Virtual ~controller(){M1model->detach(this);}
Virtual void update{}
```

```
 protected:
                  Model  *M1model;
                  Veiwn   *M1view;
};

 class tableveiw:public veiw{
public:
                  Tableveiw{model * M1): veiw(M1) { }
                  virtual void draw();
                  virtual Controller *M1Controller()
                              {return new TableController(this);}
};
```

5. LIMITATIONS AND FUTURE WORK

The following three limitations will surely improve the software architecture pattern metrics, the business application methodology will always replica manner which in turn affect the limitations. Questionable

efficiency: As the evaluation of each candidate solution, mainly due to the performance evaluation, takes several seconds, the overall approach is considerably time consuming. Here, software architects should run it in Parallel to other activities or overnight. A distribution of the analyses on a cluster of workstations could lead to significant improvements. It could also be possible to split the optimization problem into several independent parts that are solved separately and thus quicker.

Limited degrees of freedom: Currently, design options that over new degrees of freedom are not yet considered. For example, adding a new server results in further options to configure that server. Such de-sign options could be integrated by formulating the genotype as a tree structure rather than a vector. Simplistic cost model: The cost model used here is simplistic, as we only wanted to demonstrate the approach. We do not want to devise a new cost estimation technique. However, more sophisticated cost estimations techniques.

An architecture design method has been presented that explicitly addresses the non-functional requirements put on the architecture. The simulated output will always measure the metrics of the business application. The new model which is inserted in between the existing model will prove that the quality of the metrics is improved. It has been identified that the ability of a system to fulfill its non-functional requirements is, up to a considerable extent, restricted by its architecture. The proposed method starts with a functionality-based design phase in which a software architecture is designed purely based on the functional requirements. The architectural design method has been applied, in some form, in the design of systems, Experience shows that the method is provide appreciated support to the software engineers during architectural design.

6. CONCLUSION

An architecture design method has been presented that explicitly addresses the non-functional requirements put on the architecture. The simulated output will always measure the metrics of the business application. The new model which is inserted in between the existing model will prove that the quality of the metrics is improved. It has been identified that the ability of a system to fulfill its non-functional requirements is, up to a considerable extent, restricted by its architecture. The proposed method starts with a functionality-based design phase in which a software architecture is designed purely based on the functional requirements. The architectural design method has been applied, in some form, in the design of systems, Experience shows that the method is provide appreciated support to the software engineers during architectural design.

7. REFERENCES

[1]Eclipse.org. ATLAS Transformation Language (ATL). http://www.eclipse.org/m2m/atl/.

[2]A. Billig, S. Busse, A. Leicher, and J. G. Süss. Platform Independent Model Transformation Based on TRIPLE. In *Middleware'04: Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, pp. 493–511, 2004.

[2A] D. Ayed and Y. Berbers. UML Profile for the Design of Platform-Independent Context-Aware Applications. In MODDM'06: Proceedings of the 1st Workshop on Model Driven Development for Middleware, pp. 1–5, 2006.

[3] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.0. ttp://www.omg.org/docs/formal/08-04-03.pdf, 2008.

[4] S. Link, T. Schuster, P. Hoyer, and S. Abeck. Focusing Graphical User Interfaces in Model-Driven Software Development. In *ACHI'08: Proceedings of the 1st International Conference on Advances in Computer-Human Interaction*, pp. 3–8, 2008.

[5] D. Habich, S. Richly, and W. Lehner. ignoMDA: Exploiting Cross-layer Optimization for Complex Database Applications. In *VLDB'06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pp. 1251–1254, 2006.

[6] C. He, F. He, K. He and W. Tu. Constructing Platform Independent Models of Web Application. In *SOSE'05: Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering*, pp. 85–92, 2005

[6A] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, Inc., 1996.

[7] D. Ayed and Y. Berbers. UML Profile for the Design of Platform-Independent Context-Aware Applications. In *MODDM'06: Proceedings of the 1st Workshop on Model Driven Development for Middleware*, pp. 1–5, 2006.

[8] M. López-Sanz, C. Acuña, C. Cuesta, and E. Marcos. UML Profile for the Platform Independent Modeling of Service-Oriented Architectures. *Software Architecture*, pp. 304–307, 2007.

[8A] eclipse.org. Model to Model (M2M) Project.http://www.eclipse.org/m2m/.

[9] T. Fink, M. Koch, and K. Pauls. An MDA approach to Access Control Specifications Using MOF and UML Profiles. *Electronic Notes in Theoretical Computer Science*, 142:161–179, 2006.

[10] J. Bezivin, S. Hammoudi, D. Lopes, and F. Jouault. Applying MDA approach for Web service platform. In *EDOC'04: Proceedings of the 8th IEEE International Enterprise Distributed Object computing Conference*, pp. 58–70, 2004.

[11] M.Rahmouni and S. Mbarki. International Journal of Computer Science & information Technology. Vol 3, No 4, August 2011.

[12] L. DOBRICĂ, A. D. IONIŢĂ, R. PIETRARU, A. OLTEANU. U. P. B. Automatic Transformation of Software Architecture Models, Sci. Bull., Series C, Vol. 73, Iss. 3, 2011.

This paper may be cited as:
Manjula, G. and Mahadevan, G., 2014. Software Architectural Pattern to improve the Performance and Reliability of a Business Application using Model View Controller. *International Journal of Computer Science and Business Informatics, Vol. 10, No. 1, pp. 83-93*.