Research on Domain Driven Design Based Domain Platform Architecture

Qinghu Wang
College of Computer Science and Technology, Institute of Computer Application Technology
Inner Mongolia University for the Nationalities
Tongliao, China
Qinghu_wang@126.com

Jin Chen
College of Computer Science and Technology
Inner Mongolia University for the Nationalities
Tongliao, China
jinchen@163.com

Hu Wen
Department of Information Center, The Affiliated Hospital
Inner Mongolia University for the Nationalities
Tongliao, China
wenhu@sina.com

Liang Zhang
Department of Information Center, The Affiliated Hospital
Inner Mongolia University for the Nationalities
Tongliao, China
zhangliang@sina.com

Jie Lian
College of Computer Science and Technology
Inner Mongolia University for the Nationalities
Tongliao, China
jielian@sina.com

Mingyu Bai
College of Computer Science and Technology
Inner Mongolia University for the Nationalities
Tongliao, China
mingyubai@163.com

Shicheng Qiao
College of Computer Science and Technology
Inner Mongolia University for the Nationalities
Tongliao, China
shichengqiao@hotmail.com

Mingyang Jiang
College of Computer Science and Technology
Inner Mongolia University for the Nationalities
Tongliao, China
jiangmingyang@163.com

Lisha Liu
Automation Workstation
People's Laborational Army 65367 Troops
Tonghua, China
lishaliu@163.com

*Zhili Pei
College of Computer Science and Technology, Institute of Computer Application Technology
Inner Mongolia University for the Nationalities
Tongliao, China
zhilipei@sina.com

Abstract—There are two major drawbacks in traditional software development process, to follow the database as the core of the software development pattern, it is difficult to achieve a high degree of flexibility of the system and meet the changing requirements of reconstruction. And domain software with complex business, there is no reasonable means to understand the needs of the system, and difficult to build a good business system to avoid the risk of change and upgrade reconstruction of the software, the system can not be fully respected business practice. In the process of the traditional software development, facing the complex business system requirements, due to the complexity of the business, it is repentant to make failure for perfect modeling mechanism of applications domain. As so, builded systems also lack the expansion needs for the business change, and in the development process often causing the disjoint in requirements, design and realization, confusing the whole system development process. This paper studies the strategy for building a platform based on Domain Driven Design, give the guidance scheme for realization of Domain analysis technology and the Domain Driven Design platform, and explores the roots of software process consistence on Domain Driven Design based platform.

Keywords-DDD; domain analysis; DSL; DD;architecture

I. INTRODUCTION

Computer software technology has flourished development, and in the quality of software products, scalability, maintainability, flexibility and support secondary development put forward higher requirements. In traditional software development pattern, following the

database as the core building the system, this development pattern is difficult to guarantee the requirements of system's flexibility and reconstruction change. Domain Driven Design(DDD) is a popular architecture model in present software development, DDD is based on Model-Driven Design(MDD), and the model metadata representation of design elements as the core of the system, ensuring a high degree of flexibility of the system. At the same time, it follows use quick development Extreme Programming (XP) in DDD system software development process as the major developed method, ensuring the high efficiency in development process, and reducing the risk of the software.

II. DOMAIN DRIVEN SPECIFICATION

The biggest advantage of the DDD lies in its respects for the objective domain actual, making the whole software process to focus on business practical more, to achieve the consistency of the software process.

A. Domain Model in DDD

DDD uses Domain Model as the key element of driven system running, according to different realization ways the Domain Model can be divided into four models.

- Blood loss model: Domain Object is pure data models which only has read and write attributes, the business does not include in Domain Object, business implement is placed in the service layer. It is the easiest model; it can't express the practical business usually.
- Anemia model: In addition to express domain entity's data attribute, it also includes some domain business logic which is not depended on Persistence operation, but these business logic are separated into service layer to deal.
- Congestion model: Not exit business object, persistent operating business is all realized in domain object. This model is more conform to principle and standard of Object-Oriented, service layer is a very thin layer, just only encapsulates some transaction control logic, may be seen as the appearance of the domain layer.
- Expanding blood model: All the data attribute and business logic, including the persistent logic, even the control logical are implemented in Expanding blood model. So it is not necessary for existing of the service layer in the DDD layered system which depends on this model, and layered system is also simplified to a certain extent. But responsible of this model are too scattered, which leading to instability and difficult maintenance.

B. The core elements of DDD

The key goal of DDD is using model-driven approach to guide the software model building and software development, DDD uses Domain Model as the key component to make the blueprint of system architecture. The model is very important in DDD, and Domain Model is the core of DDD. The model object in Domain Model is expressed by Entity, Value Object, Service, Aggregate, Repository, or Factory.

- Entity. An entity is a class of objects that have separate identifier, the identifier's responsibility is to distinguish the different entities, and the highlights of concerned on such objects are their properties. During the entire software life cycle, entity objects have continuity, the identifier to ensure that may maintain consistency for the state of entity.
- Value Object. The value object's key concerned is not to explain what the object is, we can only care what attributes it has, and it is not has life cycle concept.
- Service. In the business modeling process, there do not just modeling on entities and value objects, but also on behavioral modeling, modeling these behaviors in an independent component so-called service. The service is a generic concept usually be implemented in the service layer, may also be implemented in the model layer. .
- Aggregate. To reduce system coupling, we modeling a set of objects with associated relationships to a larger object container, which is aggregate. Aggregate make these relationship limit its inner, which has evidence boundary with the external entities, services, value objects, and other aggregates, and so may reduce the complexity of the system.
- Factory. Based on the object-oriented design principles that a single class with single responsibility, usually separate the building responsibility of the object, and the way is factory.
- Repository. Constructed logic database repository resource library in the domain model layer, it is repository. Repository charge for the actual interaction between the system and the database, and so the domain model layer gained its rightful clarity and emphasis.

III. DOMAIN ANALYSIS TECHNOLOGY

The domain analysis technology can comprehend and recognize the problem's essence, in the process of the domain analysis, we need to pay attention to the main affects of the domain experts, and the earlier stage of the requirement analysis through domain experts to lead the whole requirement process. After requirement becomes clearer, it needs in-depth requirement analysis which based on the initial obtained results of the requirement needs, it is possible to obtain the static structure expressed by business domain, and the interaction between business, as well as lifer-cycle model of business object.

A. Consistency of the language—DSL

Using the technology of domain analysis, in order to make the business experts, domain experts and software experts use the same method to understand and communicate with each other in the whole requirement process, it needs to use common language to communicate. In fact, the programmer usually has errors for understanding the business semantics in requirement period, as well as changing the structure specific property of building system and existing the business original semantics distorted. Business domain experts finish defining for business, the technicians are in change of understanding this business semantics, and then define the semantics of the software, and follow-up of design and

development work. Two types of domain experts' docking for work hard to avoid errors , if the process of defining the semantics of the software are also giving to business domain experts , the original business semantics will not be distorted.

United Model language (UML) is a consistent language for expressing the design building model, but for domain experts and business experts , UML is fairly strange . In order to avoiding the barrier on communication , we should design a requirement language system formed by nature language , which is in common use for business expects , domain experts and software experts , and it is the compromise for UML and nature language , this language called Domain-Specific Language (DSL) . DSL needs to build a complete set of language standard, and build the lexicon table for the language, reaching the intention of this domain language designer and user .

B. Consistency of mark—Three Meta-Color Model

To be able to offer domain system design phase model layer domain model design support as possible, we propose Three Meta-Color Demand Model of needs analysis that will analyze the business are actually abstract entities business software systems in the field, the value of the object and domain service , and cards with three colors of the three requirements described in the card object , articulate and consistent way mark of this domain object model called the Three Meta-Color Model .

IV. DDD ARCHITECTURE LAYERED GUIDANCE SYSTEM

Respecting domain objective business actually is following DDD architecture software system's biggest advantage, there is putting forward four layer guide program in DDD architecture standard, the domain model layer is foundation stone for ensuring domain's special. Any system and platform both have the design of the horizontal dimension and vertical dimension. Based on correspond between layer and view of responsible separation, domain model layer seems overstaffed, not well achieve seamless docking between layer , and there is not good to the layer development of the whole architecture system with a hierarchical architecture system standardization .

This article has a comprehensive know about the weakness of traditional four layer DDD architecture standard, domain model as the foundation stone to design, putting forward a new DDD architecture layer guide system , shown as Fig .1. The new architecture system includes Presentation Layer , Control Layer , Application Service Layer , Domain Service Layer, Domain Model Layer and Basis Implement Layer six layers, and following the design principles as follows. The designing respect business reality, in favor of business changes and expand of system;

System design for high-level abstraction, generalization is not chip face system design, design-oriented common, design reflects modular, inter-module is highly aggregate, and the module is a low coupling.

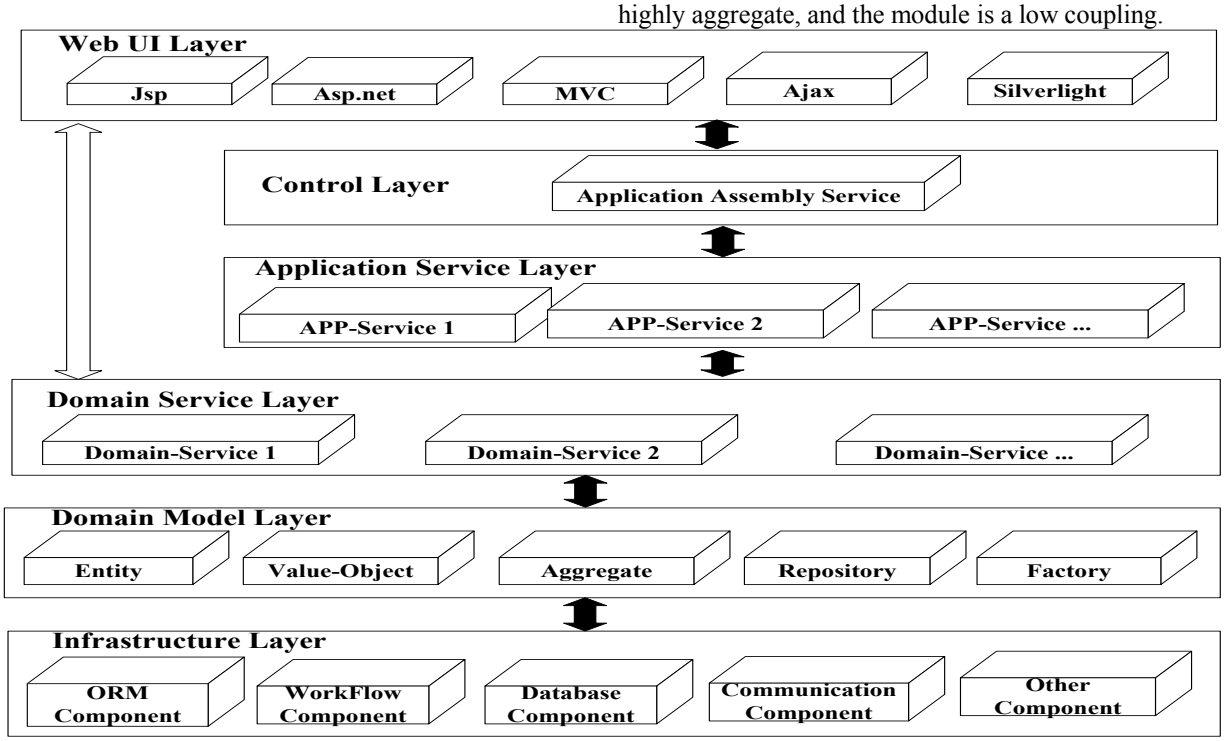

Figure 1. Layered DDD Architecture Guidance System

- Web IU Layer: Platform system and the window of user interaction, its design includes two aspects: Presentation Layer interface design and show logic design. Among them the show logic design in it need achieve separation between emerge and data, achieving emerge logic and business logic decouple absolutely.
- Control Layer: This layer is adapting to the system to design that will be expanded in the future, along with business complexity and calculate function requirement's improvement, the system need

server cluster configuration possibly one day. How to choose reasonable using server is a problem will be solved remain to platform, design the layer is main to go on using server's route.

- Application Service Layer: A very thin layer, to coordinate the activities if the application form relationships, is responsible for the whole system-level application services, including in the form of control, and the existence of things in terms of control of the system.
- Domain Service Layer: The main duty is controlling the use of domain model of domain model layer, we can understand it is domain model layer's appearance, making the design for it loose coupling possibly, and easy to build domain model.
- Domain Model Layer: The core layer of the architecture, the key is using iteration design thought to pick up and build domain model. According to the requirement to build original domain model, identify the obvious relationship between these domain models, and describe the domain conception of these domain models accurately. Analyzing the key business function of these domain models, separating which is model's responsible and Application Service Layer's responsible of domain conception expressed by domain model as far as possible. Drawing entities, the value of the object and domain service in original domain model, finding the border of the aggregate, and provide the repository for aggregate. Domain Model design need to follow facing abstract building model theory, adopting facing interface programming mode when achieve it, controlling the coupling between model layer's inside model. Recommend to use Anemia model in the new architecture system, the class diagram of the domain object in the Domain Model Layer as follows.
- Infrastructure Layer: Including support for the basic functions realization of the entire system architecture running, such as system control ORM framework persistence operation support system process-driven workflow and so on.

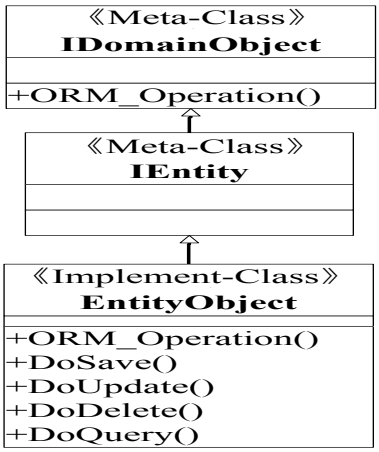

Figure 2. Class-Graphic Design of Domain Model(Entity e.g.)

V. CONCLUSIONS

DDD is perfect architecture of using achieves domain software, respect the business reality maximum. Traditional DDD standard exist some defects in defending the consistency of the software process and architecture maintainability aspects , the article analysis the essence of the domain software , pick up domain analysis theory of ensuring the consistency of the software process , reconstructed traditional DDD architecture mechanism , and gave the new DDD architecture layer guide system .How to build requirement , analysis , design , implement process automation of the domain software process will be the next emphasis of this article.

ACKNOWLEDGMENT

This work was financially supported by the National Natural Science Foundation of China (61163034, 61373067), the Inner Mongolia talent development fund(2011), the Grassland Excellent Talents Project of Inner Mongolia Autonomous Region(2013), Supported By Program for Young Talents of Science and Technology in Universities of Inner Mongolia Autonomous Region(NJYT-14-A09), the Inner Mongolia Natural Science Foundation(2013MS0911, 2013MS0910), the 321 Talents Project the two level of Inner Mongolia Autonomous Region(2010), and the science research project of Inner Mongolia University for the Nationalities(NMD1316, NMD1229). The Corresponding author is Zhili Pei, and the Corresponding email is zhilipei@sina.com.

REFERENCES

[1] Wesenberg H,Landre E,Ronneberg H.Using domain-driven design to evaluate commercial off-the-shelf software. The 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems,Languages,and Applications . 2006.

[2] Landre E,Wesenberg H,Olmheim J.Agile enterprise software development using domain-driven design and test first. The 22th ACM SIGPLAN conference on Object oriented programming systems and applications . 2007.

[3] Mat Wall,Nik Silver.Domain-Driven Design in an Evolving Architecture[EB/OL/].http://www.infoq.com/articles/ddd-evolving-archi tecture . 2008.

[4] Arie van Deursen,Paul Klint. Domain-Specific Language Design Requires Feature Descriptions[J]. Journal of Computing and Information Technology . 2002 (1).

[5] Quinn TP,Smith C,McCowan C N, et al.Arc sensing for defects in constant-voltage gas metal arc welding. Welding Research Supplement . 1999

[6] Liao H W,Niebur D.Load profile estimation in electric transmission networks using independent component analysis. IEEE Transactions on Power Systems . 2003

[7] Li D,Zeng A,Ye F, et al.Multivariate statistical process control for online monitoring of short-circuiting GMAW. Chi- nese Journal of Mechanical Engineering . 2003

[8] Hyvarinen A,Oja E.Independent component analysis: al- gorithms and applications. Neural Networks . 2000

[9] Kourti T.Process analysis and abnormal situation detection: From theory to practice. IEEE Control Systems Magazine . 2002

[10] James C,Lowe D.Using dynamical embedding to isolate seizure components in the ictal EEG. IEE Proceedings on Science,Measurement and Technology . 2000