Viewpoint Based Problem Modeling Technique and Its Application

Zhishang Yang
Department of Computer and Information Engineering
Guangxi Normal University, Guilin, China
Email: zhishang_yang@163.com

Zerui Sun and Shenluo Li
Department of Computer and Information Engineering
Guangxi Normal University, Guilin, China
Email: {gxnuszr, lishenluo}@163.com

***Abstract*—Jackson's Problem Frame (PF) approach emphasizes the importance of modeling and analyzing the world outside of computer before drifting into designing solutions. Furthermore, it uses problem diagram to vividly capture requirements, problem domains, and their relation in an intuitive and structural way. After visualizing the problem, it provides a solid foundation for communication between software stakeholders, decreasing the possibility of mismatch between customer's real needs and software system developers' understanding of customer needs. Thus PF approach is a useful tool in Requirement Engineering (RE).**

While today's development activities of complex system invariably deeply roots in large scare problem context and involves various kinds of stakeholder who have different perspectives or viewpoints on the problem they are addressing. It is vital to explicitly represent and analyze these viewpoints, their relationships, then integrate them into a complete and consistent form to ease further overall architecture design and development.

This paper explores the integration of viewpoint concepts with problem frames methodology which aim at providing a more flexible and practical way to express requirements from different classes of stakeholders and the problem context they concerned, then, we integrate these viewpoint based sub-problem together to provide basis for subsequent phrases in software life cycle.

***Index Terms*—problem frame, viewpoint, architecture design, requirement engineering**

I. Introduction

Since the rumored "requirements problems" has been confirmed [1] and more and more people recognize the critical nature of requirements in Software Engineering (SE), RE has gradually been established as an important sub-field of SE [2]. However, although almost 20 years have been past, the problem of mismatch between what the customer's needs and what the software developer understands the customer's needs [3] still exist, this problem often leads to project failures, cost overruns and late deliveries. As Bashar et al [4] pointed out, better modeling and analysis of problem domains, as opposed to the behavior of software, are still the key research areas.

PF approach [5, 6] captured the model of problem that the software stakeholders confronted by providing a framework for representing domains, requirements relevant to the problem, their relationship, thus helps a lot in customer requirement elicitation and the elimination of misunderstanding between developers and customers about the problem at hand. However, at present, how to locate and bound the complex real world problem still heavily depends on the experience and technique of requirement engineers, and systematical guide in a natural and understandable way of the problem discovering process is still absent in the literature, these seriously hinder the application of PF approach in industry.

In this paper, we propose to combine the concepts of viewpoint [16] with PF approach to improve the requirement elicitation and problem locating and bounding process, and at the same time, provide solid foundation for further architecture design. The rest of the paper is organized as follows: Section II introduces the conceptual basis of our technique. Section III proposes a process model that guild the practitioner to use our technique. Section IV validates the usability of our technique by a real problem. Section V presents related works and makes some discussion. Section VI proposes a meta-model for future work.

II. Conceptual Basis

In this section, we will describe the concepts that are relevant and illustrate the reasons for integrating them in our technique. Some discussions are also provided.

A. Problem Frame Approach

In PF approach, software problem is defined as a task to be accomplished by software development and modeled by problem diagrams. Problem diagrams describe a particular problem and show the problem parts: the requirement, the domains, and the interfaces and references among them. The domains are the parts of the world that are relevant. The parts of the world in which the problem locate is called problem context and are shown in a context diagram. Context diagram structures the world into machine domain and problem domains, and shows how they are connected. The machine domain is domain about which machine to be built in a software development problem, how it is built in the form of software and deployed by running the software on a general-purpose computer. Interface is the connection among two or more domains consisting of phenomena that they share. Phenomenon is the element of what we can observe in the world such as event, state. Requirement phenomenon is the phenomenon of a problem or a domain that are the subject of requirement references. Specification phenomenon is the phenomenon of a problem or a domain that are shared with the problem machine. These concepts and their relation are briefly scratch in figure1.The rectangles with a double vertical stripe, a vertical stripe and without vertical stripe is the machine, the design domain and physical domain respectively, the solid line and dotted line represent the interface and requirement reference or constrain respectively, and the dashed oval represents the requirement. A systematic account of PF approach is beyond the scope of this paper and can be found in [6].

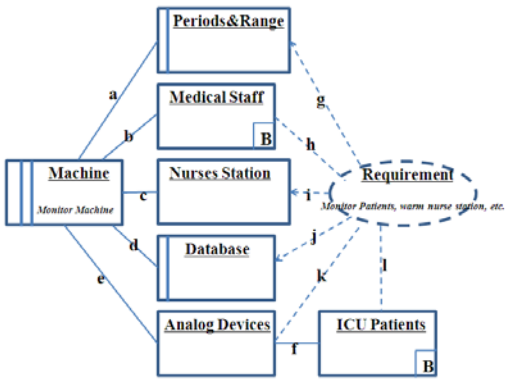

Figure1. Patient monitoring problem diagram form [6], modified.

B. Problem Decomposition and Composition

In RE, the sub-field of SE, which aims to cope with the problem in the early phase of software life-cycle, the application of the ‘divide and conquer’ principle can be found in various methodology, such as use case[7], scenario[8], goal[9], PF approach[5,6].

In [6], Jackson point out that the key to mastering problem size and complexity is decomposition, which means to break down a given large and complex problem into a number of smaller and simpler sub-problem. There are plenty work about problem projection or decomposition in the literature, such as [10,11,12,13,14,15], but not many to deal with the problem of how to completely capture and model the complex reality problem, which is the input of the projection and decomposition process and also the basic for architecture and solution design.

C. Extending PF Approach with Viewpoint

As you see in Figure 1, the dashed oval named Requirement which is a kind of composite requirement represents the requirement in traditional problem diagram. The requirement description of this node is some kind of integrate description that generated through synthesizing various classes of stakeholder requirements by the requirement engineer. In another word, it represents a synthesized view of all kind of stakeholders. We argue that the more natural or close to reality of the method we use in requirement elicitation and modeling process, the more complete and comprehensive information we will get, and that information is good for further analysis and design activities. So in our points of view, primarily, there are three kinds of drawback while using a composite requirement node or description in a problem diagram to represent all kind of stakeholder’s requirements:

1) It violate the basic nature of software problem that multiple classes stakeholders involve in large-scale software development often hold multiple view on the system being developed, and each single class of view concerns about different subset of domains and their relations in the problem context.

2) It is hard for requirement engineers to composite various kinds of requirement description from different classes of stakeholder into a single, complete, consistent requirement.

3) In complex realistic problem, this kind of representation will become the obstacle of further analysis and design such as architecture design, problem decomposition, etc.

Due to the problem discuss above, we propose to extend PF approach with the idea of viewpoint in RE. In history, there is plenty of work about viewpoint in SE or more specifically in RE, while normally different methods have different purposes, the definition of viewpoint also vary largely. For example, in CORE [16], viewpoint is defined in two levels. The first level consists of all entities that interacting or affecting the system in some way. The second level concerned with defining viewpoints that are sub-processes of the system and bounding viewpoints of entities that interacted indirectly with the system; in [17], viewpoint is seen as an external entity interacting with the system which being analyzed, but one can exist without the presence of the system; in [18], while being treated as vehicle for separation of concerns, viewpoints was define as loosely coupled, locally managed, distributable objects encapsulating partial representation knowledge, development process knowledge, and specification knowledge, about a system and its domain; in[19], viewpoint represents a particular perspective or a set of perceptions of the problem domain, often associated with a 'viewer' or agent that maintains and accepts responsibility for that viewpoint, therefore captures the domain knowledge and understanding related to a particular role or view of the problem domain adopted by the viewpoint agent. To ensure as far as possible that the system can meets the needs and expectations of different classes of stakeholders, it is necessary to capture, model, analysis, and understand their various viewpoints and to detect and eliminate any inconsistencies and conflicts between these viewpoints.

We inherit the work in [20, 21, 22, 23] to give the formal definition of concepts that related to our work. We define viewpoint as:

$$V= \{M, D, L, VOR\} \quad (1)$$

V is short for viewpoint, which encapsulates elements that concerned or catches interest of stakeholders from specific classes, ignoring those that are unrelated. Formula (1) demonstrates that a viewpoint contains four aspects:

a) VOR, short for viewpoint oriented requirement, which represent requirement of stakeholders from specific classes, of whom have the same or similar perspective or desire to develop the system.

b) D denotes domains in the problem context which related to VOR. Also, D is defined as:

$$D=\{x|x\in B\vee x\in P\vee x\in X\vee x\in C\vee x\in D\} \quad (2)$$

Formula (2) means that the type of D is constrained by the following elements: B(biddable domain), P(physical domain), X(lexical domain), C(causal domain), D(design Domain), for more details about domain type, please refer to [6].

c) M denotes sub-machine (software component) which interact with related domains to satisfy corresponding VOR.

d) L denotes interface between different domains, domain and VOR or domain and machine, for which information is transmitted and internal viewpoint structure is constructed. Also, L is defined as:

$$L=\{(x,y)|x,y\in(M\vee D\vee VOR)\wedge x\neq y\} \quad (3)$$

Formula (3) means that L is a set of which each element is a tuple, the elements in each tuple belong to set (M∨D∨R) and they can’t equal to each other, this means that we view every domains in a problem diagram as ‘meta-domain’ and thus they can’t be disassembled. Actually, set L defines the syntagmatic relation inside each viewpoint.

Zave et al [20] and Hall et al [21] view a software problem as:

$$K, S|\text{-}R \quad (4)$$

And

$$W, S|\text{-}R \quad (5)$$

Respectively, formula (4) and (5) mean that the way to solve the problem is to find S which combines with K or W can satisfy R, where K is a description of the problem domain and W is the real-world context, S is the specification of the solution, and R is the problem requirement. Obviously, M and VOR in (1) fall into the concept of S and R in (4) and (5) respectively, D and L in (1) fall into the concept of K in (5) or W in (6).

Correspondingly, the diagrammatic representation of viewpoint is a viewpoint diagram (VD), we formally define VD as follow:

$$VD= \{M, D, L, VOR\} \quad (6)$$

M,D,L,VOR represent the graphical notation of M,D,L,VOR in formula (1) respectively and their diagrammatic representation is the same as in [6].

Confront to the fact that the development of complex system unavoidably involves various classes of stakeholders who have different perspectives (viewpoints) on the problem they are addressing, the system developing and they concern about only part of the whole problem, then we define the software problem as:

$$P=\bigvee_{i=1,\ldots,n} V_i \quad (7)$$

P is short for problem. Formula (7) means that a problem is the union of a series of viewpoints. We use VOPD (Viewpoint-oriented problem diagram) to graphically represent P, so we have:

$$VOPD=\bigvee_{i=1,\ldots,n} VD_i \quad (8)$$

Formula (8) means that a VOPD is the union of VDs.

III. Process Model for Viewpoint Based Problem Modeling Technique

We propose a process model, as Figure 2 depicts, to guide the practitioner to use our technique in reality software development activity. The process model only concern about the activities of problem bounding and locating and system architecture design for the purpose of emphasizing the advantage of our technique for further system architecture design. Other activities behind this model will be the same as activities in traditional software lifecycle.

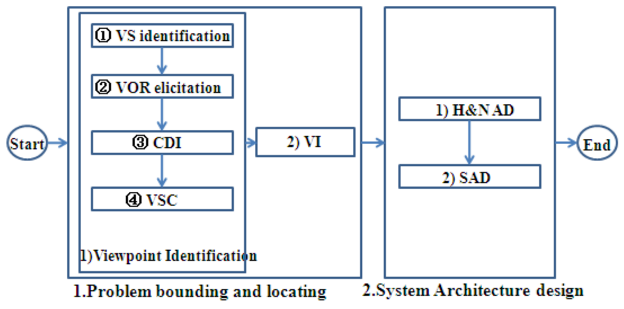
Start
① VS identification
② VOR elicitation
③ CDI
④ VSC
1)Viewpoint Identification
2) VI
1.Problem bounding and locating
1) H&N AD
2) SAD
2.System Architecture design
End

Figure2. A process model for viewpoint based problem modeling technique.

Detail explorations about the process model can be found in table I.

TABLE I.

Abbreviation Explanation of Process Model.

Abbreviation	Original Spelling	Description
VS identification	Viewpoint Source Identification	Represent specific classes of stakeholder which contain the similar or same view/requirement about the system to be build.
VOR elicitation	Viewpoint Oriented Requirement elicitation	Represent requirements from specific VS.
CDI	Concerned Domains Identification	Identify domains concern by specific VOR.
VSC	Viewpoint Structure Confirmation	Confirms the structure intern to each viewpoint.
VI	Viewpoint Integration	None.
H&N AD	Hardware and Network Architecture Design	None.
SAD	Software Architecture Design	None.

As you can, one of the core activities in the 'problem bounding and locating' phase is 'viewpoint identification'. Given the fact that different works about viewpoint existing in the literature have different purpose, thus the guidance for viewpoint identification varies from method to method. For example, CORE [16] suggests a session of 'brainstorming', to identify possible sets of viewpoint, such as users, buyers and specific of the system, then distilled them into a set of functional and non-functional viewpoints, furthermore, functional viewpoints is divided into a set of bounding and defining viewpoints; Kotonya and Sommerville [17] consider viewpoint identification as the identification of 'system authorities', people or documents that have an interest in or specialist knowledge of the application domain, including system end-users, system procurers, system specialists, documentation on existing systems, etc. Nuseibeh and Kramer[18] propose to identify viewpoint by instantiating the so call viewpoint template which contains five viewpoint slots: the style slot, the work plan slot, the domain slot, the specification slot, the work record slot; Darke and Graeme[19] identify viewpoint primarily through recognizing different 'viewers' or agents, and the domain knowledge or understanding they capture.

For the purpose of 'sensitive to how people perceive and understand the world around them, how they interact…', as Bashar et al point out in [4], we first identify the system stakeholders and their requirements. According to the similarity of the requirements (the domains that concerned, the desire effects of the domains upon the system being developed), we then classify the stakeholders into different classes and regard them as different VS. After that, we explicitly record these VOR and CDI, explore the structure internal to each viewpoint. Finally, we integrate these viewpoints into a complete and consistency problem diagram. In the system architecture design phase, we carry out the activity of 'Hardware and Network Architecture Design', 'Software Architecture Design' based on the output of the early phase.

In the next section, we will demonstrate the availability of our technique by modeling and designing the architecture of a real problem.

IV EXEMPLE

We follow the activities introduced in the process model which proposed in Figure 3 to model and solve a hypermarket management problem, to demonstrate the usability of our technique.

A. Problem Bounding and Locating

While subsequently carry out the activities introduce in our process model and with the help of various kind of traditional elicitation techniques such as questionnaires, surveys, interviews, and analysis of existing documentation (organizational charts, process models or standards, and user or other manuals of existing systems), we get the following result.

Activity 1, VS identification. We identify the following VS as table II depicts.

TABLE II

SYSTEM VS TABLE.

Domain/ VS Id	Domain/ VS Name	Domain Type	Description
1	Customer	B	Those who buy goods and pay by cash or credit card.
2	Cashier	B	Those who receive good, cash or credit card from customer and operate the POS system and finish the due properly.
3	Weigh Good Seller	B	Those who sell special goods that should be paid according to its weigh.
4	Manager	B	Those who are responsible for system database and stock maintenance, promote sale, purchase of merchandise, financial settlement, etc.
5	General Manager	B	Has the highest priority and can access all information of the system and provide management decision.

Activity 2, VOR elicitation. The relation between requirements and corresponding VS are details in table III.

TABLE III

VS AND REQUIREMENTS RELATION TABLE.

Domain/ VS Id	Requirement
1	1) Present goods/ sticky notes, cash and receive bill and payment; 2) Present goods/ sticky notes, credit card , input the password and receive bill, credit card with proper amount of money deducted; 3) Present goods and receive the partial bill.
2	4) Receive goods/ sticky notes, cash, use the barcode reader to gather goods information and input the amount of money received and the amount of money should return will be display, the bill will be print automatically. 5) Receive goods/ sticky notes, credit card, use the barcode reader to gather goods information and proper amount of money was deducted from the credit card, the bill will be print automatically.
3	6) Receive goods, use the barcode reader to gather goods information and use the Electronic Scale to weigh the goods received then the sticky notes print automatically.
4	7) Check and maintain system database and stock, promote sale plan generation, purchase order generation, financial statistics generation.
5	8) Access all kind of information of the system, management suggestion generation.

Considering the scenario interaction between different VSs, we then define the VORs, its VS and requirement as table IV demonstrate.

For the convenience of description, we list domains and their ID other than those details in above table in table V.

TABLE IV
VOR TABLE.

VOR Id	Related VSs	VOR Description
1	1,3	Customers present goods and sellers present sticky notes with information about this due.
2	1,2	Customers present goods/ sticky notes and payment, cashiers present balance and bill.
3	1,2	Customers present goods/ sticky notes, credit card then input password; cashiers present bill.
4	4	Check and maintain system database and stock, promote sale plan generation, purchase order generation, financial statistics generation.
5	5	Access all kind of information of the system, management suggestion generation.

TABLE V
SYSTEM DOMAIN TABLE.

Domain Id	Domain Name	Domain Type	Description
6	Barcode Reader	C	Identify barcodes that pasted on all kinds of goods, and then transmit the information to the POS Host.
7	Bill Printer	C	Receive bill information from the POS Host and print it accordingly.
8	Customer Display	C	Display due information for the customers.
9	Smartcard Reader	C	Identify Id of credit card and transmit it to POS Host.
10	CCPK	C	Short for credit card password keyboard.
11	Cash Drawer	C	Pushed open according to the command of POS Host, and then closed by the cashier.
12	Electronic Scale	C	Display the catalogue and unit price of the goods, calculate the total price according to goods' weight, and then print the sticky notes which contain due information, such as barcode, unit price and weight of the goods, total price, etc..
13	POS Server	M	Machine composited of special-purpose hardware and software. It runs POS server application to manage large amount of POS Hosts and communicate with E-Bank Server and Database Server.
14	POS Host	M	Machine composited of special-purpose hardware and software. It runs POS client application to manage other special domains from 6-11 to finish the due.
15	E-Bank Server	C	Machine composited of special-purpose hardware and software. It is well designed to provide services for electronic transaction (ET).
16	Central Management Server	M	Machine composited of special-purpose hardware and software. It monitors the operation of Database Server, Management Workstation and POS Server and so the whole system. Accessed by general manager only.
17	Database Server	M	Machine composited of special-purpose hardware and software. It provides access to Optical Storage Device.
18	Optical Storage Device	P	Machine composited of special-purpose hardware. It stores all information of the whole system and provides fast access for later usage.
19	Management Workstation	M	Machine composited of special-purpose hardware and software. It is used by manager to maintain basic information, promote sales, produce order form, and all kinds of final statement.

Activity 3, CDI. Domains related to each VOR are details in table VI.

TABLE VI
VOR AND DOMAINS RELATION TABLE.

VOR Id	VS Id	Concerned Domain id
1	1,3	12,14
2	1,2	6,7,8,11,14
3	1,2	6,7,8,9,10,14
4	4	13,14,17,18,19
5	5	13,15,16,17,18,19

Activity 4, VSC. While using requirement node and biddable domain node to denote for VOR node and VS node respectively, we can use the graphical notation of PF methodology to model the identified viewpoints as bellows.

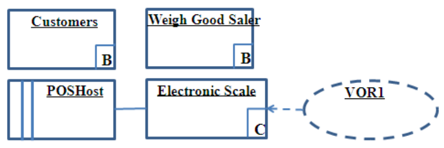

Figure 3a. Viewpoint1

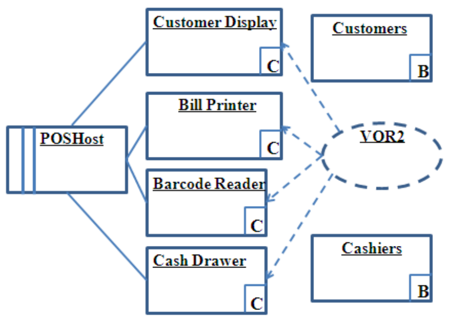

Figure 3b. Viewpoint 2.

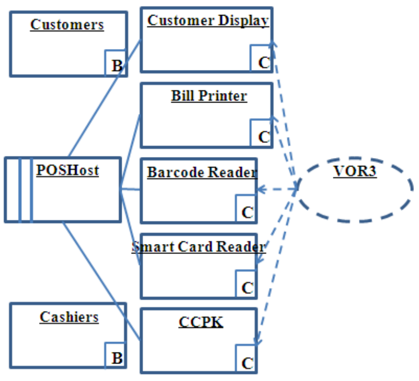

Figure 3c. Viewpoint 3.

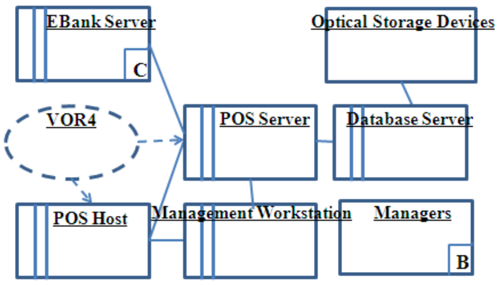

Figure 3d. Viewpoint 4.

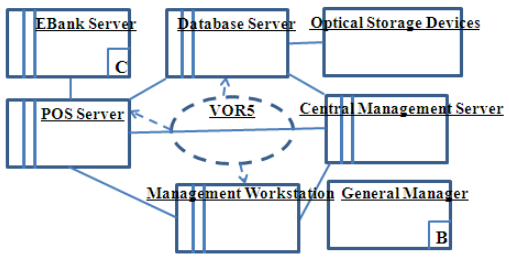

Figure 3e. Viewpoint 5.

Activity 5, VI. After the integration of above viewpoints, we then get the VOPD as figure 3f in page 7 depicts.

TABLE VII

SYSTEM HARDWARE LIST.

Device	Number
POS terminal	N
Electronic scale	N
Management workstation	N
Management server	N
DB server	N
POS server	N
Optical switch	N
Optical storage device	N
Referral server	N
Proxy server	N
Firewall	N

Note that we haven't directly connect biddable domains with the machines, because actually they interact with the system through human-machine interfaces such as keyboard, mouse, display, and this kind of component is implicitly embedded into domain of type 'M'. All these graphical notations are the same as [6].

B. System Architecture Design

Given above VOPD, we then extract the problem context as figure 4 in page 7 depicts.

Note that string '1-*' and '*-1' mean that one POS Server and one Management Workstation manage a large amount of POS Hosts.

Activity 1, Hardware and Network Architecture Design. According to above information gather, we then list the system hardware in table VII.

Note that what we design is a common solution, the exact number of device will be depending on the scale of application. So 'N' in the 'Number' column is denoting for any positive integer. We then design the system hardware and network architecture as figure 5 in page 7 depicts.

Our design has the following advantages:

High efficiency. While implementing the strategy of separate between data and applications, different applications, we can guarantee that every module of the system can run independently and efficiently.

Security. By adopting network security techniques such as isolate between intranet and extranet, intrusion detection, VPN, access control, we can guarantee the system run safety.

Activity 2, Software Architecture Design. The technologies we use are described in table VIII.

TABLE VIII.
TECHNOLOGIES LIST OF SOFTWARE ARCHITECTURE.

Item	Description
Software system architecture	MVC
Programming language	J2EE
OS	Windows/Linux/Unix
DB	MySql/SqlServer/Oracle
Web Server	Tomcat/Websphere
Secondary developing method	Modularization-based approach (Inheritance, override), Interface-based approach
Development tool	myEclipse
Front-end technology	Spring+Extjs
Back-end technology	Mybatis/Hibernate
ReportViewer	jfreeChar

The hierarchical relation about these techniques are vividly depict in figure 6.

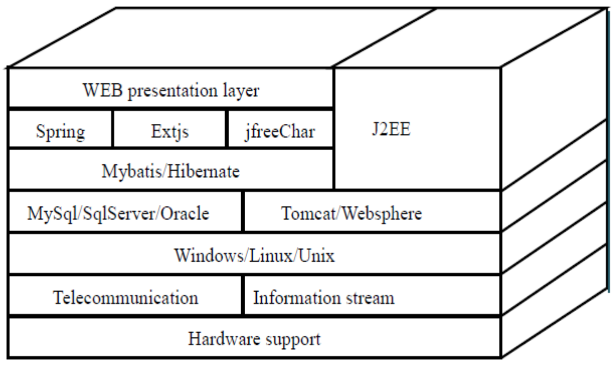

Figure 6. Software Architecture.

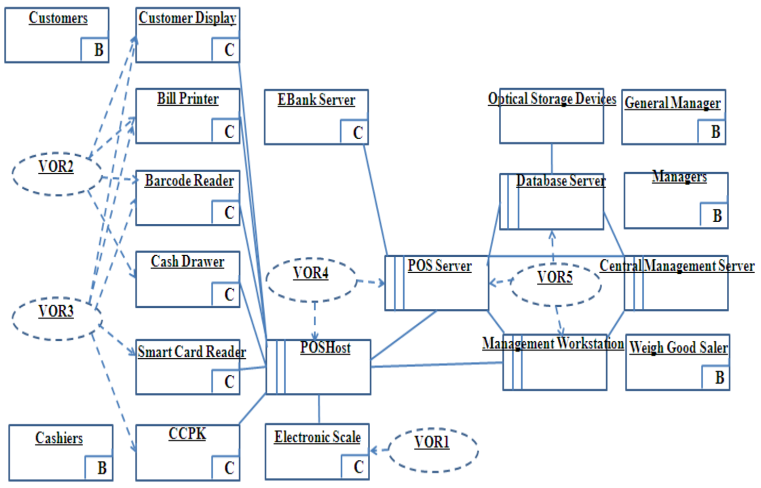

Figure 3f. Hypermarket management VOPD.

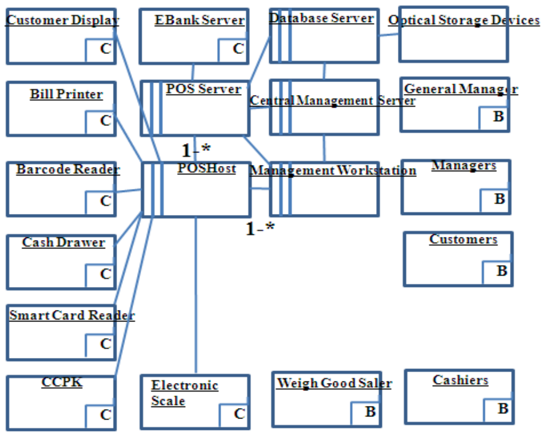

Figure 4. Hypermarket management problem context diagram.

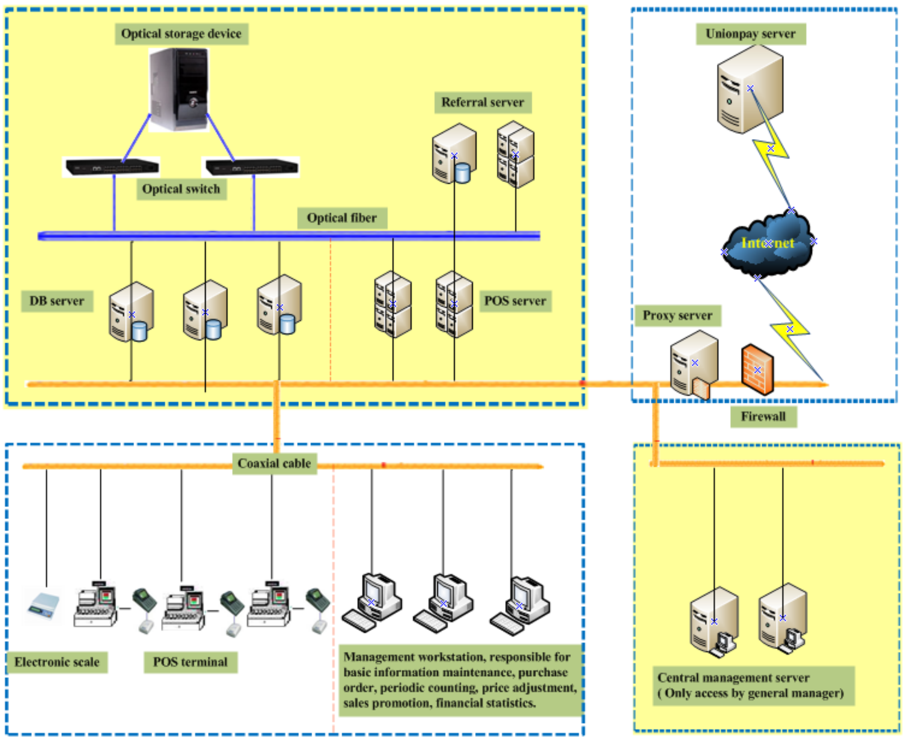
Optical storage device
Referral server
Unionpay server
Optical switch
Optical fiber
Internet
DB server
POS server
Proxy server
Firewall
Coaxial cable
Electronic scale
POS terminal
Management workstation, responsible for basic information maintenance, purchase order, periodic counting, price adjustment, sales promotion, financial statistics.
Central management server
(Only access by general manager)

Figure 5. Hardware and Network Architecture.

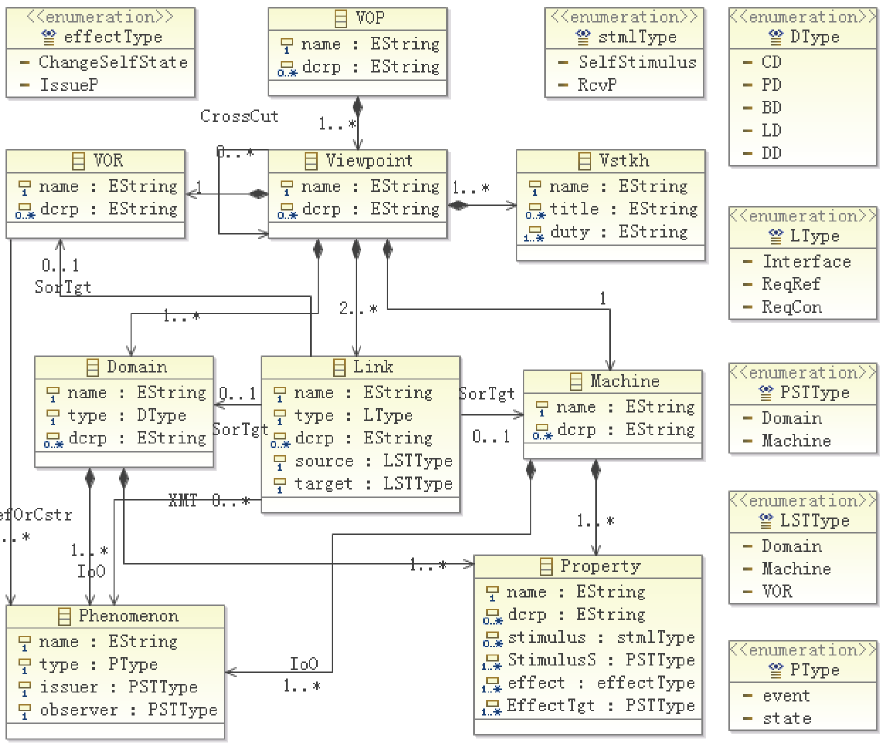

Figure 7. A viewpoint based on problem meta-model.

In figure 6, system hardware is demonstrated in figure 5. While using the standard communication protocol such as TCP/IP or R485, our software system can run on multi-platform, include Windows, Linux and UNIX. According to the scale of the application, we can use mysql, sqlserver or oracle as database, and choose tomcat or websphere as web server according to the OS we use. Within J2EE framework, we use 'Spring+Extjs' and 'mybatis/hibernate' as front-end demonstration technique and back-end manipulation technique respectively. In the end, we use jfreeChar middleware to generate statistical analysis report.

Our design has the following advantages:

High efficiency. While using the hierarchical structure to separate the application from the server, the overall performance of system was increase greatly.

Strong extension capabilities. Hierarchical structure makes it very easy to extend presentation layer such as add IPad client or migrate databases such as change from SQL Server to Oracle.

Low cost. Use hierarchical structure can lead the structure of problem more easily to read and understand, thus greatly improve the development productivity and lower the development and maintenance cost.

V Related Works and Conclusion

In spite of several searches, there are not many works explicitly handling problems frames and viewpoint. Laney, R., L. Barroca, et al [24] introduce Composition Frames to handle inconsistent requirements problem. Lencastre et al [25, 26] integrates aspect concepts in Aspect-Oriented Software Development (AOSD)[27] into Problem Frames to handling crosscutting concerns, in [28], they go further to composite problem by aspect. Rapanotti, L., J. G. Hall, et al [29] introduce Architectural Frames – combinations of architectural styles and Problem Frames to facilitate problem analysis, decomposition and subsequent recomposition. Similarly, Choppy, C. H., et al [30] define architectural pattern for different problem frames, while a complex problem can decompose into subproblems that match the basic frame, the software architecture for the overall problem can be construct by compositing corresponding architectural pattern.

Our experience in applying this approach in real case is meaningful but not very satisfactory, the initial feedback we received from this empirical study confirms that:

1) This approach respects the fact of practical software development activity, providing a natural and useful way to elicit and model different classes of stakeholder's requirement;

2) The captured subproblems and the integrated huge problem provide a basic infrastructure for internal and external viewpoint analysis;

3) This method provide hint to perform problem projection thus ease the way of problem decomposition;

4) the integration of these viewpoint based problem model provide solid foundation for overall system architecture design, it still useful in the situation that when the problem at hand can't fix into any basic frame or it can't be decomposes into subproblems that match basic frame.

However, although with the help of Openpf [31], perform viewpoint integration manually proved to be a tedious and error-prone task, the success of system architecture design also heavily depends on the complexity of the problem and the experience and professional capability of the analyst or designer. This greatly restricts the method's application in real context. Thus we go further to propose a meta-model, as figure 7 in page 8 depicts, for tool building to ease the application of our approach.

VI A META-MODEL FOR FUTURE WORK

There are plenty of works in the literature containing introductions of meta-model, which support PF approach[25,26,32] or Problem decomposition[33,34], but none of them explicitly model domain properties and support viewpoint base problem modeling technique. Eclass 'Vstkh' is short for viewpoint stakeholder; eclass Property' uses the relationship between domain phenomena to explicitly model properties of domains and machine; enumeration 'stmlType' is short for 'stimulus type', 'effectType' enumerates possible effects upon different kinds of stimulus, 'PSType' enumerates the source or target type of a phenomenon, 'LSTType' enumerates the source or target type of a link. Here, we do not provide detailed discussion about the meta-model, as its design is strictly conform to the definition in section II.

Summarize experience in building tool which support transforming requirements into specifications [35], We plan to exploits the Eclipse Graphical Modeling Framework (GMF) [36] technology for the definition of viewpoint based problem model editors based on our meta-model, furthermore, make full use of model to model(M2M) transformation technique such as ATL [37] and model to text(M2T) techniques such as Acceleo [38] to develop plug-ins on our model editor to make it support automatic viewpoint integration, viewpoint based problem projection and system architect generation. Future work also includes applying our approach to more empirical studies.

ACKNOWLEDGMENT

We would like to thank Dr.Li, Dr.Yu and TongYi for the kindly instruction and useful comments on this paper. This work was supported by Innovation Project of Postgraduates in Guangxi, China under Grant No. YCSZ2012059.

REFERENCES

[1] T. E. Bell and T. A. Thayer. "Software requirements: Are they really a problem?" *In Proceedings of the 2nd international conference on Software engineering, IEEE Computer Society Press*, Los Alamitos, CA, USA, 1976, pp. 61–68.

[2] S. Greenspan, J. Mylopoulos and A. Borgida, "On formal requirements modeling languages: RML revisited," *In ICSE '94: Proceedings of the 16th international conference on Software engineering*, pp.135–147, May 1994

[3] J. Castro, M. Kolp and J. Mylopoulos, "A Requirements-Driven Development Methodology." *Advanced Information Systems Engineering*. 2001, pp.108–123.

[4] Bashar Nuseibeh and Steve Easterbrook, "Requirements engineering: a roadmap," *In Proceedings of the Conference on The Future of Software Engineering (ICSE '00), ACM*, New York, NY, USA, 2000, pp.35–46.

[5] Michael Jackson. Software Requirements and Specifications: a lexicon of practice, principles and prejudice. *Addison-Wesley*, 1995.

[6] Michael Jackson. Problem Frames: analyzing and structuring software development problems. *Addison-Wesley* Longman, Co., Inc., Boston, MA, USA, 2001.

[7] Ivar Jacobson and Pan-Wei Ng, "Aspect-Oriented Software Development with Use Cases," *Addison-Wesley Object Technology Series*, 2004.

[8] S. Robertson and J. Robertson, Mastering the Requirements Process, *Addison-Wesley*, 1999.

[9] E. Yu. "Towards modeling and reasoning support for early-phase requirements engineering." *In Proceedings of the 3rd IEEE Int. Symp. On Requirements Engineering*, Washington D.C, USA, pp. 226–235, Jan 6-8, 1997.

[10] D. Jackson, M. Jason, "Problem decomposition for reuse," *Software Engineering Journal*, vol.11, pp.19-30, Jan 1996.

[11] L. Rapanotti, J. G. Hall, et al. "Architecture-driven problem decomposition." *In Proceedings of the 12th IEEE International Requirements Engineering Conference*, pp. 80-89. 2004.

[12] J. Zhi and L. Lin. "Towards automatic problem decomposition: an ontology-based approach." *In Proceedings of the 2006 international workshop on Advances and applications of problem frame*. *ACM*, New York, NY, USA, pp. 41-48.

[13] C. Xiaohong, J. Zhi. "A Scenario-Based Problem Decomposition." *The 9th International Conference for Young Computer Scientists, ICYCS*, pp.1150-1155. 2008.

[14] J. Zhi, C. Xiaohong, et al. "Performing Projection in Problem Frames Using Scenarios." *Software Engineering Conference. APSEC '09. Asia-Pacific*, pp. 249-256. Dec 2009.

[15] C. B. Haley, R. C. Laney, B. Nuseibeh. "Using Problem Frames and projections to analyze requirements for distributed systems." *In: Proceedings of the Tenth International Workshop on Requirements Engineering: Foundation for Software Quality, co-located with the 16th International Conference on Advanced Information Systems Engineering*, Riga, Latvia. pp. 7-8. Jun 2004.

[16] A. Dardenne, S. Fickas, et al. "Goal-directed concept acquisition in requirements elicitation." *Proceedings of the Sixth International Workshop on Software Specification and Design*, pp. 14-21. 1991.

[17] G. Kotonya, I. Sommerville, "Viewpoints for requirements definition." *Software Engineering Journal*, vol.7, pp.375-387, Nov 1992.

[18] B. Nuseibeh, J. Kramer, et al. "A framework for expressing the relationships between multiple views in requirements specification." *IEEE Transactions on Software Engineering*, vol. 20, pp. 760-773.1993.

[19] P. Darke, G. Shanks, "Stakeholder viewpoints in requirements definition: A framework for understanding viewpoint development approaches." *Requirements Engineering*, vol.1. pp. 88-105.1996.

[20] P. Zave and M. Jackson, "Four dark corners of requirements engineering." *ACM Transactions on Software Engineering and Methodology*. Vol. 6. pp. 1–30, January, 1997.

[21] J. G. Hall, L. Rapanotti, et al. "Problem Oriented Software Engineering: Solving the Package Router Control Problem." *IEEE Transactions on Software Engineering*, Vol. 34, pp. 226-241, March-April, 2008.

[22] J. G. Hall, L. Rapanotti, and Michael Jackson, “Problem frame semantics for software development.” *Software & Systems Modeling*, vol. 4, pp. 189-198. 2005.
[23] M. Jackson and P. Zave. “Deriving specifications from requirements: an example.” *Proceedings of the 17th international conference on Software engineering. Seattle, Washington, United States, ACM*, pp. 15-24. 1995.
[24] R. Laney, L. Barroca, et al. “Composing requirements using problem frames.” *RE '04 Proceedings of the Requirements Engineering Conference, 12th IEEE International*. pp. 122-131, 2004.
[25] M. Lencastre, J. Araujo, A. Moreira, J. Castro, “Towards aspectual problem frames: an example” *Expert Systems*, vol. 25, pp.74-86. 2008.
[26] M. Lencastre, J. Araujo, A. Moreira, and J. Castro. “Analyzing crosscutting in the problem frames approach.” *In Proceedings of the 2006 international workshop on Advances and applications of problem frames. ACM*, New York, NY, USA, pp. 59-64. 2006.
[27] A. Rashid, A. Moreira, J. Araújo, “Modularisation and Composition of Aspectual Requirements”, *AOSD’03*, Boston, USA. pp. 11-20. 2003.
[28] M. Lencastre, A. Moreira, et al. “Aspects Composition in Problem Frames.” *International Requirements Engineering, 2008. RE '08. 16th IEEE*, pp. 343-344. 2008.
[29] L. Rapanotti, J. G. Hall, et al. “Architecture-driven problem decomposition.” *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. pp. 80-89. 2004.
[30] C. H. Choppy, D.; Heisel, M. “Component composition through architectural patterns for problem frames.” *Software Engineering Conference. APSEC 2006. 13th Asia Pacific*, pp. 27 – 36. 2006.
[31] OpenPf, http://mcs.open.ac.uk/yy66/.
[32] P. Colombo, L. Lavazza, et al. “Towards a Meta-model for Problem Frames: Conceptual Issues and Tool Building Support.” *ICSEA '09. Fourth International Conference on Software Engineering Advances*, pp. 339-345. 2009.
[33] D. Hatebur, M. Heisel, and H. Schmidt. “A Formal Metamodel for Problem Frames.” *Proceedings of the 11th international conference on Model Driven Engineering Languages and Systems*, Toulouse, France: Springer-Verlag, pp. 68-82, 2008.
[34] L. Lavazza, A. Coen-Porisini, et al. “A Meta-model Supporting the Decomposition of Problem Descriptions.” *Fifth International Conference on Software Engineering Advances*, pp.50-57. 2010.
[35] Zhi Li, Jon G. Hall, Lucia Rapanotti. “On the systematic transformation of requirements to specifications”, *Requirements Engineering*, Springer-Verlag, May, 2013.
[36] Graphical Modeling Framework (GMF), http://wiki.eclipse.org/GMF.
[37] ATL, http://www.eclipse.org/atl/.
[38] Acceleo, http://www.eclipse.org/acceleo/.

Zhishang Yang, born in 1988, M. S. His research interests include requirement engineering and software quality management.

Zerui Sun, born in 1988, M. S. His research interests include requirement engineering , image processing and data hiding.

Shenluo Li, born in 1987, M. S. His research interests include requirement engineering, machine learning and data mining.