http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (3): 397-402, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.397.402

Domain-based Software Concurrent Engineering

[1]Yuwan Gu, [1,2]Guodong Shi, [1]Dean Zhao and [2]Yuqiang Sun
[1]JiangSu University College of Electronic and Information Engineering, Jiangsu, Zhenjiang, 212013, China
[2]ChangZhou University International Institute of Ubiquitous Computing, Jiangsu, Changzhou, 213164, China

Abstract: The design idea of domain-based software concurrent engineering is proposed based on component-based software development. Its specific activities and the parallelism of events are dug and the parallel process model is established. The stress was reusable and parallel in the model. The parallel process model is described by Petri net, the reachable marking graph is established, the activity and safety of the process model is analyzed, it ensures reasonability of the model.

Key words: Domain, component, Petri net, reachable marking graph

INTRODUCTION

In the study, Domain-Based Software Concurrent Engineering (DBSCE) is proposed. Firstly, DBSCE requires direct putting core operation requirement of new user and reuse component development which satisfies to new system requirements on domain engineering. Secondly, DBSCE requires that domain engineering is also continuous evolution on the development process. Two aspects of that incarnate that domain engineering and software engineering iterative repeated nonlinear parallel development has no strict sequence relation character on the development process. The method requires that software and relative each sub-process (containing requirement analysis, architecture design, system design, system implementation, operation and maintenance etc sub-process) is analyzed, designed, developed integrately and parallel architecture-based based on domain engineering. The popular method of object-oriented (Zahran, 2002; Tong and Lixia, 2001a) is used for studying DBSCE in the paper.

Petri net is a mathematical object which can use graphical representation, by mathematics developed Petri net method and technique can be used for the static structure analysis and dynamic behavior analysis (Tong *et al*., 2003; Mu *et al*., 2008). Petri net is a popular modeling tool now widely used to describe the parallel, asynchronous, distributed and randomicity etc., using Petri net for complex parallel design system can accurately simulation and analysis and improve the description ability and the expression of model.

DBSCE PROCESS MODEL

Development of software produce is implemented by specific software development process, the quality of software produce and software productivity is determined by and large by the quality of software development process (Tong and Lixia, 2001b). Software process model (Daiping, 2008; Sheng *et al*., 2009; Horn, 2001) is abstract description for process, is key of all kinds of parallel components harmonize consistency¡¢parallel implement in the implementation software life cycle.

DBSCE process description: DBSCE emphasize on reusability and synchronization parallelism. Domain engineering forms problems of generality in system into special domain analysis model, special domain architecture, marking which component is reused in the development process of new application system and how to modify it for adapting to new system requirement, finally, reuse component is entered into reuse component base. At the same time, by software engineering, special case in specific application is solved, domain model, DSSA is disposed specifically and developing special component of the application, then special component and marking, tectonic component in domain engineering is assembled into a new system (Fig. 1).

Requirement analysis phase

Domain analysis: Carrying out analysis, digging, marking, protocol for a class of common character, knowledge, requirement in application domain. Mainly activities

Corresponding Author: Yuqiang Sun, ChangZhou University International Institute of Ubiquitous Computing, Jiangsu, Changzhou, 213164, China

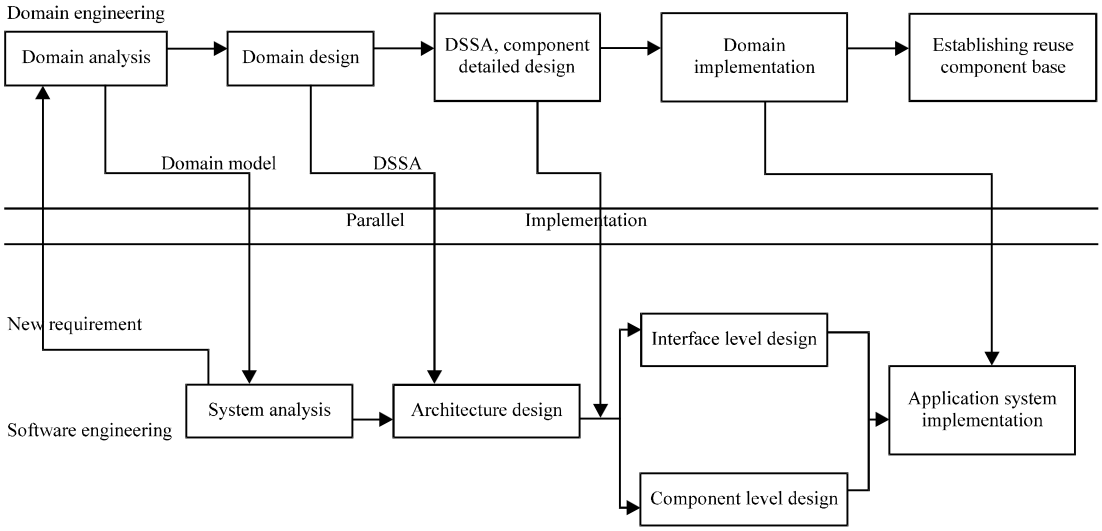

Fig. 1: The process model of software concurrent engineering

contain establishing domain requirement definition, establishing domain object-oriented analysis model, establishing domain term dictionary. The base of establishing domain requirement definition is requirement definition of each existing system in domain, the aim is that definiting which requirements are commom requirements and which requirements are variability requirments, forecasting future requirement, definiting binding time of variability requirment and expressing dependent and mutually exclusive relation during requirements, then introducing core operation requirement of new consumer, carrying out supplement and modulation for domain requirement definition. Considering to establish core model part from common part of requirement when establishing domain object-oriented analysis model, then considering changeable part. The establishing process is that wholly establishing a new object-oriented analysis model based on object-oriented analysis model of existing system. Element (class, attribute, method, relation etc.) of establishing model and requirement of requirement definition demand to establish traceability, namely fixed requirement traces to fixed element, requirement of many-to-one, optional requirement separately traces to element of many-to-one and optional element, The traceability is bidirectional. Participators of domain engineering have common language through establishing domain term dictionary, in order to implement. The process of unified terminology is that common part and changable part of requirement is separated based on standard specification. Finally, carrying out review for establishing domain model, among them, an important activity is that variability is fixed, in order to get a requirement model of existing system. The requirement model is high abstract level.

System analysis: Introducing consumer requirement, comparing requirement of new application system to domain analysis model of upper abstract level, ensuring requirement protocol of new application. It contains function requirement, capability requirement, data requirement, componet requirement etc. Taking each function of system into an example when analysis. An important activity is that carrying out modulation and modification for domain model from above in the stage, mianly manifesting on that fixed requirement, many-to-one requirement, optional requirement of consumer new requirement separately uses generalization, inclusion, extension mechanism of example to example model and detailed describing for example model, then examples are grouped and formed operation component.

Requirement analysis stage is a process of continuous repeaton, stepwise simplification.

Outline design stage

Domain design: There is large similarity or close on design, because system in the same domain is similarity or close in requirement. The foundation of domain design is mainly design of domain model and existing system in domain. Design stage contains three activities: Preliminary domain design, variability design, establishing traceability with domain analysis model. The tasks of preliminary domain design are that considering from whole, synthesizing all kinds of factors, solving conflict in design, establishing a basal object-oriented design model, providing preliminary solution for common requirement of domain analysis. The object-oriented design model is formed by supplement and modification based on object-oriented design model of existing system, also establishing a new object-oriented design model which modulates and supplements for object-oriented analysis

model, not simplification. Firstly, paying attention on common, steady part in model, secondly, turning to changeable part. Variability design is that separating changeable part from relatively fixed part in system combining with design pattern, assigning fixed part to DSSA, assigning changeable part to component, uniting the similar or the same component in the process of assignation. The interface between DSSA and component is syllabify and concision. DSSA which contains component protocol is formed by method and mechamism of abstract class, polymorphism, inheritance in design pattern. Establishing traceability of domain analysis model is that using CASE tool can find element successfully in object-oriented analysis model, when some class in object-oriented design model want to look for correlative element in object-oriented analysis mode, there are elements (class, relation) in object-oriented design model which correspond to changeable elements, fixed elements in object-oriented analysis model. DSSA system is ensured after above three activities but the DSSA is high abstract level, it satisfies to requirement of re-user further.

Architecture design: Ensuring architecture and specific implementation function of application system based on analysis model and DSSA, it can decompose to hiberarchy. At the beginning of decomposition, system can denote to set of architecture frame; each frame is formed by one or more design pattern. New system requirement component is assigned through design pattern, the similar or the same component is united together, then forming into large operation function component. At the same time, carrying out data architecture design based on ER graph for consumer new requirement but emphasizing on reuse existing data architecture in domain engineering. Data architecture design of software engineering firstly creates high abstract level data model based on data view of consumer, client. In the follow, data model is refined to specific representation of relation to implementation. Namely data architecture design contains all kinds of datebases and mappings from program to database. Carrying out modulation, supplement, simplification for DSSA based on two activities, in order to satisfy to new application system requirement.

Specific design stage

DSSA, component specific design: The activity implements for reuse component. Specific design in object-oriented method aims at single class, variability fixups on specfic component in domain engineering after domain design, the specific design of class in domain engineering is similar to in the application engineering. The only distinction is that it need to reference existing system in domain engineering when carrying out specific design. Design of graphical user interface is a activity which is neglected easilly in th stage. Graphical user interface has understanding at and large only for what about system and user interaction, is not concrete user interface.

Implementation phase

Domain implementation: The mostly tasks are reuse component development, testing work, choosing reuse component in new development domain to reuse component base.

Application system implementation: it contains special component development in new development system, adaptability midification of reuse component, joining of the whole software flow, component assembling and testing work.

DESCRIBING THE PARALLEL PROCESS MODEL BASED ON PETRI NET

Petri net has estate-based expression fashion, Graph exprsssion method of common software engeering domain is sometimes event-based expression, the latter is used in the paper. Humphrne and Kellner early in 1989 and points out that parallel exist in the any stage of software development (Qianmu *et al*., 2008). Existing large numbers of parallel activity in the process of software parallel development based on domain engineering. The parallel activity which is excavated is benefit to enhance the software development speed, shorten the development cycle and reduce cost. Parallel process is shown in Fig. 2.

Domain design and system analysis is parallel implemented after establishing domain model. It offers input for architecture design and DSSA, component detail design after establishing preliminary DSSA, two activities can be parallel implemented. DSSA, component detail design offers input for domain implementation, component level design, interface design, three activities can be parallel implemented. After domain implementation, on the one hand, component stores in reuse component base, on the other hand, component participates in application system implementation. Two activities can be still parallel implemented.

Parallelism which is high abstract level in Fig. 2 can be simplified further. Carrying out simplification uses layered fashion. Carrying out layered simplification takes example of domain analysis activity; domain analysis parallel process is shown in Fig. 3.

It can be found two important characters through observing analysis (Fig. 3); succession and parallelism. Establishing domain requirement definition contains three primary activities: Ensuring common requirement in domain, ensuring requirement variability in domain,

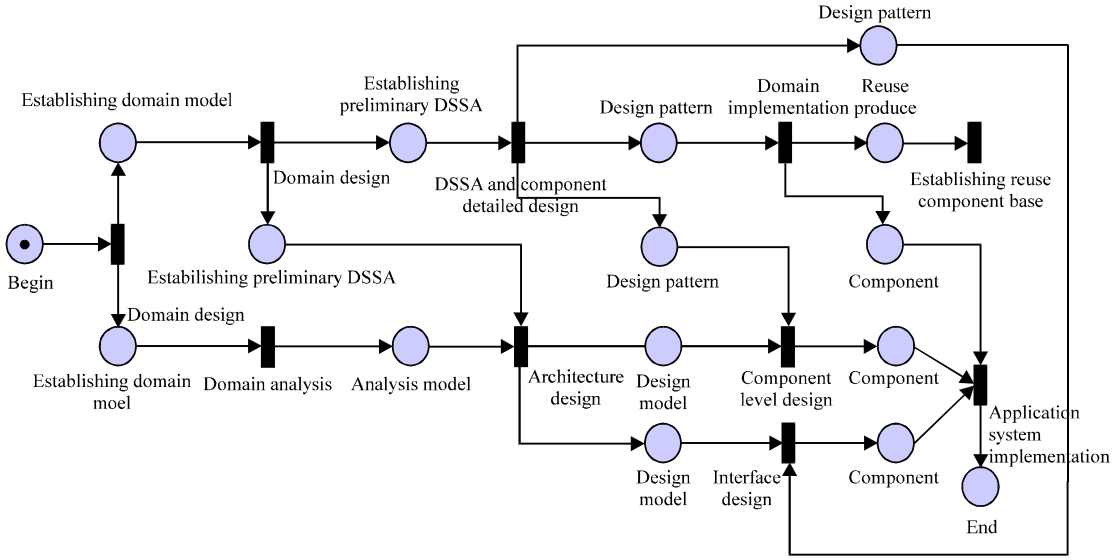

Fig. 2: Parallel process

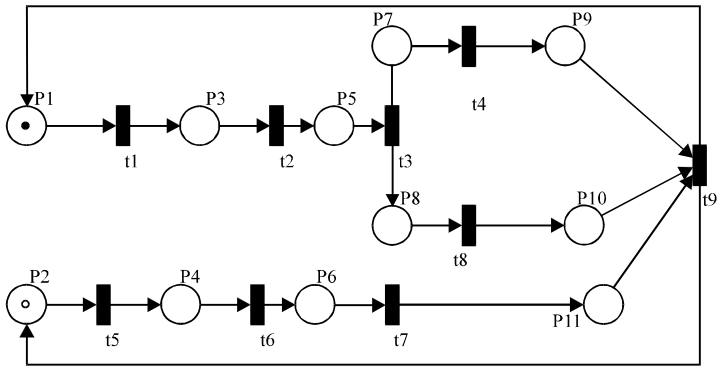

Fig. 3: Parallel process of domain analysis, t1: Ensuring common requirement in domain, t2: Ensuring requirement variability in domain, t3: Ensuring relation during variability requirement, t4: Establishing object-oriented analysis model, t5: Ensuring term, t6: Ensuring explanation, t7: Ensuring thesaurus, t8: Establishing traceability during domain requirement definition and t9: Review

ensuring relation during variability requirement. The three activities is sequence relation. Establishing domain object-oriented analysis model contains two parallel activities: Establishing object-oriented analysis model, establishing traceability during domain requirement definition. Establishing domain dictionary contains three sequence executive activities: Ensuring term, ensuring explanation, ensuring thesaurus. Establishing domain requirement definition and establishing domain object-oriented analysis model implements sequence but establishing domain dictionary interludes in the two activities and parallel implements with them. Domain analysis is repeating, gradually refined process, inclusive all activities are all repeating but not sequence relation on strict meaning, namely each activity can return to frontal activity for supplement and perfection.

Activity: It is easy to find that each change in net has possible to get occurrence right again namely in any reachable marking regardless what time is the system run, through analysis run of (Fig. 3). It indicates that each change is live in net, consequently, the whole system is live.

ANALYZING MODEL BASED ON PETRI NET

Getting two important characters from observing operation condition of Fig. 3 through the method of hand simulation: parallelism and activity. It can be observed

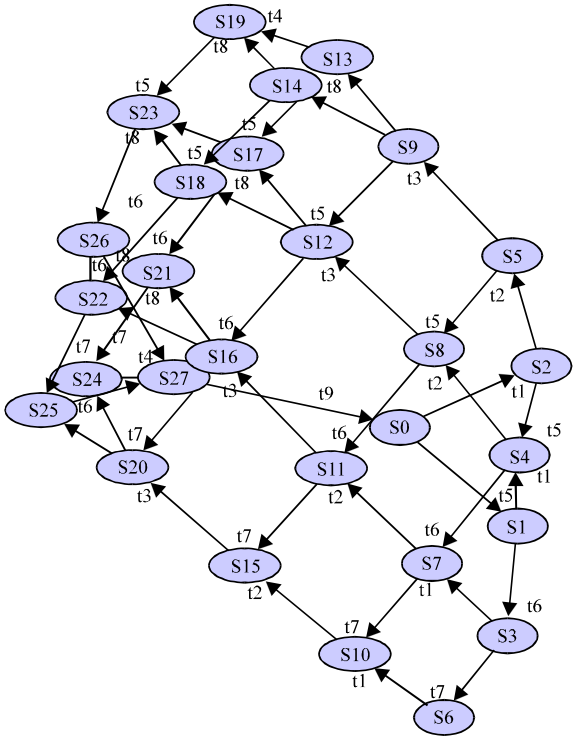

Fig. 4: Reachable marking graph, S0 = (0, 0, 0, 0, 0, 0, 0, 0, 0,1,1), S1 = (0, 0, 0, 0, 0, 1, 0, 0, 0,1,0), ..., S26 = (0, 1, 0, 1, 0, 0, 0, 1, 0,0,0), S27 = (0, 1, 1, 1, 0, 0, 0, 0, 0,0,0)

some characters for some simple net system. But it presents leak using observational method to analyze for complicated system. Figure 3 is analyzed in detail using reachable marking graph which is shown in Fig. 4.

Reachable marking graph takes reachable marking set R (M0) for vertex set, forming a directed graph takes direct reachable relation during markings for arc set. The condition of occuring sequence of estate change of net system and variance is analyzed by it, then getting net system relative character.

We can found that net system has following chacter according to reachable graph:

- No die marking in net. Because no outdegree of a node is zero in reachable marking graph
- Net system is a reversible net. Because reachable graph is a strongly connected digraph
- Setting up Sj∈S, the bound of Sj is1: B(Sj) = 1. Bound equals to the maximum of weight j of vector of each vertex in reachable marking graph in Petri net
- Net system is safety. Because the vector of each vertex is 0 or 1 vector in reachable marking graph
- Net system is activity. Because each directed path which starts from vertex S0 goes to a strongly connected sub-digraph finally, further more each t∈T is side labeled of a directed arc in each such strongly connected sub-digraph, in reachable marking graph

Figure 3 satisfies to security and activity through above analysis. The validation of Fig. 2 is also according to above process.

CONCLUSION

DBSCE emphasizes on mutual coordination, mutual learning between domain engineering and software engineering, domain engineering provides sustenance and choice for software engineering, software engineering provides modification and modulation for domain engineering, they are mutual promotion and development together. DBSCE is described by petri net, estate, phased achievements, activity etc informations of software process are abstracted and standardization and standardization of software process is simulated. However, with the improvement of system net complexity, the relation in reachable marking graph is more anfractuosity, how to reduce redundancy in net, so predigesting reachable marking graph is the next researchful content.

ACKNOWLEDGMENTS

Supported by The project of general office of Broadcasting and Television (GD10101) and Natural Science Fund in ZheJiang (Y1100314).

REFERENCES

Daiping, L., 2008. Software Architecture Tutorial. Tsinghua University Press, Beijing, China.

Horn, P., 2001. Autonomic computing: IBM's perspective on the state of information technology. IBM Research.

Mu, L.Q., X.M. Wu and Z. Hong, 2008. System Architeture Design (M). Qinghua University Press, Beijing.

Qianmu, L., X. Manwu and Z. Hong, 2008. Architecture Design. Tsinghua University Press, Beijing, China.

Sheng, L.D., C.D. Hua and L.A. Ping, 2009. Software Engineering Principles, Methods and Tools. Mechanical Industry Press, Beijing.

Tong, L. and W. Lixia, 2001a. Concurrency extending in software process activities. Comput. Appl. Software, 2001: 34-39.

Tong, L. and W. Lixia, 2001b. Parallelism of the software process mining. Comput. Appl. software, 18: 27-31.

Tong, L., K. Bing and J. Zhao, 2003. Concurrent Development Processes of Software (M). Science Press, Beijing.

Zahran, S., 2002. Software Process Reconstruction. Mechanical Industry Press, Beijing, China.