EVOLUTION-BASED SOFTWARE DEVELOPING ENVIRONMENT*

YING Jing(应 晶), HE Zhi-jun(何志均), WU Ming-hui(吴明晖)
(*Department of Computer Science, Zhejiang University, Hangzhou 310027*)

Received Mar.29, 1999; revision accepted Apr.16,2000

Abstract: This paper introduces the MHSC Methodology, and proposes an evolution-based integrated developing environment for MHSC; and addresses the components and their interrelations in the IDE (Integrated Developing Environment), based on which the evolutionary prototyping system development can be supported effectively.

Key words: IDE, evolution, MHSC
Document code: A **CLC number:** TP311.52

INTRODUCTION

Due to the software system's natural complexity, software development, especially large scale and complex system development, usually loses control of the plan. Current research indicates that there exists requirements bottleneck in the realm of software engineering (Hsia, 1993). The traditional waterfall development methodology cannot satisfy the customers' real needs because it assumes the users' requirements are steady and frozen before it is implemented. In fact, a software system can be validated only when it is executable. Because it is too expensive to fix errors, especially requirement errors, in later development stage, it is important to detect potential errors in the early stage of software requirements. The authors propose to use an executable specification language to construct the intermediate result of system requirement analysis, build an integrated development environment for it, and use transformations and refinements based domain knowledge to build an executable prototyping system. Customers can participate in the prototyping system demonstration, so the system requirements can be validated, and the definition and design can be kept consistent. Through the prototyping system evolution, it finally meets the customer's real needs.

Aiming at the software system requirements validated in early stage, we put forward MHSC, a methodology for high-level specification construction (Ying, 1995, 1997). MHSC supports a software constructive procedure with regeneration structure. The whole development procedure is shown in Fig. 1.

Based on the MHSC methodology, we address real-time domain system development and build an integrated developing environment MHSC/IDE.

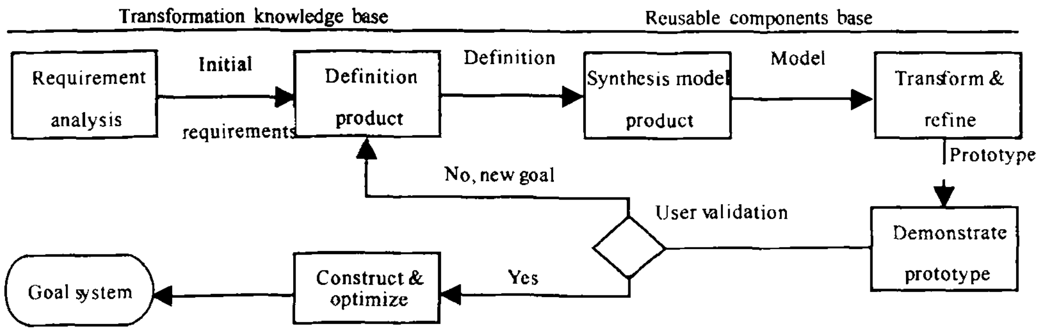

Fig.1 MHSC development procedure

* Project supported by NSFC(69703005) and Zhejiang Provincial Natural Science Foundation(697006).

MHSC/IDE

As an integrated developing environment, MHSC/IDE supports software system requirement analysis, design and configuration. The special characteristics of MHSC methodology in software evolution make requirements change and system maintenance become easier and less expensive, and decrease the probability of losing control in project development.

1. Coordination support environment (CSE)

The requirements analysis and design process for large scale and complex software systems is fundamentally a conversation among the participators (customers, designer, manager, and so on) to resolve the design issues. based on this principle, Rittel proposed the Issue-Based Information System (IBIS) model (Conklin, 1988), which has now been extended and put into applications as the gIBIS model, for instance. According to the MHSC methodology, we extend IBIS to: 1) Cooperative work support through distributed network; 2) Multi-dimensional representations (graphics, descriptive text, PSDL code, etc); 3) Import deductive mechanism, provide reference to detect conflict and merge override; 4) Permission and authority management; 5) Components version control

The MHSC/IDE system is composed of a series of interactive roles (components of two kinds: atomic and composed). We design the layer net architecture shown by Fig. 2. The roles of system are stored in a database through an interface that translates the component properties and methods to records. Client/Server is used to provide distributed computing. The data concurrency, consistency and security are controlled by DBMS.

2. Reusable software base system (RSBS)

Reuse is an important part of MHSC. If provided a relatively mature environment, software development can be automated in special domain. The reusable objects in MHSC include traditional functions, PSDL description modules, components following CORBA or COM standard and design patterns or architectures. Focusing on the PSDL description modules, we designed a reusable software base system which sup ports

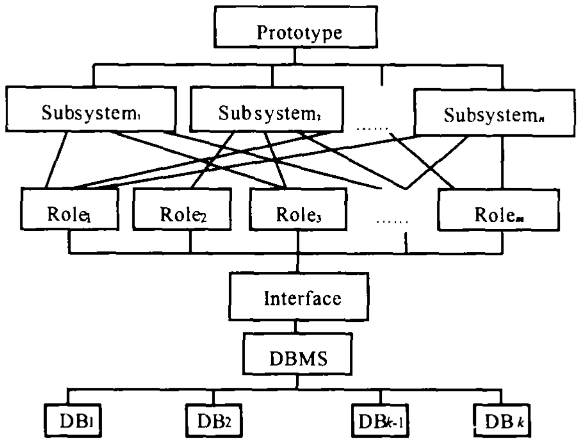

Fig.2 Layer net architecture

components acquisition, classification, storage, retrieve and integration. There are two classes of components base in MHSC: public and private. The components saved in public base are mature and credible, and have passed strict testing. They permit all members in a developing team to share information. They are read-only information, therefore, only the administrator has the privilege to modify them. The private bases are fully controlled by the owner independently. When a designer thinks a component has reuse value, he can put it into his private base and recommend it to the public base administrator. After the component passed test, it can be saved in public base. Components classified as primary facets, include using domain, application platform, produce language, function description and keywords. The components can be retrieved by filling fields with conditions and setting priority, selecting query type (match whole or not), then retrieval results will be listed by matching degree. The system also supports retrieving by relations between components through navigator.

3. Knowledge base management system (KBMS)

System evolution procedure is achieved by transformations and refinements based domain knowledge, which is possible only if the domain knowledge base is abundantly provided with suitable deductive mechanisms. In KBMS, domain knowledge and deductive formulas are stored in a DDB (Deductive Data Base). Domain knowledge is acquired by domain experts and designers with previous experiences. With domain knowledge accumulated, the decision support and automatic development will be better.

4. Procedure control system (PCS)

The PCS is composed of evolution control system and configuration system. Software system producing procedure is a component-evolution procedure through transformations. In this procedure, each designer is responsible for one or more components. For the relations between components, for example, UsedBy, Evolve, and ComprisedBy; the requirements change automatically to induce a chain of activities to propagate the changes down to the affected parts of the system design and implementation, which is called change propagation. Aiming at this problem, we propose an evolutionary development model MHSC/DM, and establish the rules for components transformation and version control. The model uses data graph to record components and their dependencies from system requirements analysis, through AffectedBy, Scope and Induce rules, and it can compute the set of components affected by the change to one component, and also provide decision support for alternatives selection, project scheduling and system configuration. On the basis of the model, prototype can be produced automatically or semi-automatically with the supports of domain knowledge.

5. Execution support system (ESS)

The execution support system of MHSC/IDE includes two classes of translators (between PSDL graph and PSDL code module, PSDL language to C++), C++ compiler and simulation scheduler system. The description of the system created by designers may be incomplete or even conflicting, so the domain knowledge is needed to help its translation. Through translation we can get executable code frameworks, for the system description is relatively mature and complete, these code frameworks can become executable system after being compiled.

The simulation scheduler system plays an important role in MHSC, especially in the real-time domain. Generally, the PSDL description of prototype is a high-level design, and the details of the process have not been filled in yet except for the time and resource constrains. But if these constrains cannot be satisfied, the detailed design is meaningless, so it is necessary to provide schedule diagnostic information which can assist design. We have implemented the simulation scheduler system based on extended Petri net, which is used to detect conflicts of components and submit corresponding reports for diagnosis and referenced resolve alternatives. After the high-level design passes the simulation scheduling, it can be refined in details until the prototype system is acceptable to the customers.

CONCLUSIONS

With regard to the evolutionary software system development to support executable specification, based on MHSC methodology, this paper proposes a corresponding integrated developing environment MHSC/IDE. A prototype environment has been built to put into practical application for software development. Through the IDE, large scale and complex software system development become well supported. It also lets high-level software construction and early validation become realizable, and facilitates transformation from requirements to implementation.

References

Conklin, J., Begeman, M., 1988, gIBIS: a hypertext tool for exploratory policy discussion, *ACM Trans. Office Inform. Syst.*, **6**(10):303 - 331.

Hsia, P., 1993, Status Report: Requirement Engineering, *IEEE Software*, **10**(6):75 - 79.

Ying, J., He, Z., Wu, Z., et al., 1995, A Methodology for High-level Software Specification Construction. *ACM SEN*, **19**(2):48 - 54.

Ying, J., He, Z., Wu, Z., et al., 1997. Building Executable Specification To Support Software Development. *Journal of Software*, **7**(5);350 - 359.