Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 13 (8): 1227-1232, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.1227.1232

Application Research on Legacy Software Reengineering in Automated Test System

[1]Ai-Ping Li, [2]Xue-Ping Li, [1]Li-Guo Duan and [1]Zheng-Hua Wang
[1]College of Computer Science and Technology, Taiyuan University, Taiyan, China
[2]Polytechnic Institute, Taiyuan University of Technology, Taiyuan, China

Abstract: The legacy software of Automatic Test System (ATS) type has a broad market prospect. A method of hierarchical software architecture based on Component-Based Software Development is provided in this paper to solve the reengineering problem of signal-based test program sets in ATS and the realization of ATLAS translator is taken as the example. This paper discussed the diversified and complicated problem of "the selection and distribution for instruments" during the development of ATLAS Compiler, which is widely used in aviation test field and gave the detailed description for realizing the "the selection and distribution for instruments" model and described the interface state for resource allocation component by Microsoft COM model. The COM model resolved the resource invoking and semantic analysis for signal related statement during the accomplishment of ATLAS compiler project. At the same time, the above method provided a reference to domain-component reengineering problem in ATS.

Key words: Legacy software reengineering, automatic test system, component-based software development, semantic analysis of ATLAS compiler

INTRODUCTION

With the rapid development of software, hardware, network technology, measuring technology and instruments, Automatic Test System (ATS) is becoming more and more important in the design, manufacturing and verification of various kinds of modern equipment systems. Particularly, ATS with proprietary intellectual property rights is playing an increasingly significant role in areas both for military use and for critical civil use. There are vast market needs for reengineering which extracts key businesses and reusable components, especially those related to the business, from the legacy system of automatic test type.

Software reuse is the process of reusing "the software designed for reuse purpose" (Zhang *et al*., 2001; Tracz, 1995). ATS legacy system reengineering based on software reuse technology can provide valuable generic. This paper proposes that, in GTEST (Li, 2006) of proprietary intellectual property, component-based signal-oriented typical software framework should be adopted to solve reengineering issues of signal-based test program sets in ATS. It bases the argument on the characteristics of component-based software development and the research and realization of ATLAS (Abbreviated Test Language for Avionics Systems) translator design that is based on GTEST Software Platform, as well as ARINC's (Aeronautical Radio Incorporation) SMART (Standard Modular Avionics Repair and Test) system and the ATLAS compiling solutions of TYX's PAWS (Professional ATLAS Work Station). In addition, this paper solves the semantic realization of ATLAS translator design using component technology and offers utilizable solutions to various problems in ATS, such as the reengineering of domain-specific knowledge component.

CURRENT RESEARCH

Software reuse is not only an important research domain in software engineering, but also a major subject in the resolution of software crisis and development of software industry (Frakes and Kang, 2005). With its application gradually becoming the mainstream study of software engineering, a lot of large-scale and systematic software reuse, under the influence of mechanisms of assembling and abstraction, have generated systems that are hierarchical and in accordance with the knowledge level of applied domain business.

As an abstract description of the intrinsic feature of component, component model is regarded to be one of the key elements of successful reuse. Software reengineering is the process of transforming the original legacy system into new system which better satisfies customers' demands, performs better and enjoys easier management and maintaining (Yang, 2011; Liu *et al*., 2012).

Automatic test technology has undergone several developments: from special purpose to general purpose and from the early emphasis on the study of ATE

Corresponding Author: Li-Guo Duan, College of Computer Science and Technology, Taiyan, China

(Automatic Test Equipment) technology to the establishment of ATS architecture and equal importance attached to the studies of ATE technology and the development and transplantation of TPS (Test Program Set). There are two main methods of developing TPS: one is equipment-oriented and the other signal-oriented. The former has already had directly usable graphic programming tool; while the latter mainly uses ATLAS (Abbreviated Test Language for All Systems). Signal-oriented ATLAS test program can well solve transplantation issues of TPS and realize independence between test program and test equipment. There are two major ATLAS development environments for test program abroad: TYX's PAWS and ARINC's SMART; while at home, ATLAS compiling environment with proprietary intellectual right is still undergoing development and trial within a limited range and it is urgent to develop a TPS development and operation environment which has comprehensive functions and is advanced and practical.

RELATED KNOWLEDGE

A brief introduction to ATS: ATS generally refers to an equipment system that makes computer as its core and can automatically complete particular test tasks under the control of programs (Zhao, 2003). It is usually built on the standard measurement and control bus or equipment bus. A typical ATS is made up of three parts: ATE, TPS and TPS development and operation environment (Li *et al*., 2004), as shown in Fig. 1.

The architecture of ATS: The architecture of ATS features in distinct layers, as shown in Fig. 2. An integral test system basically provides a complete solution through several bottom-up interrelated layers (Duan and Li, 2005). The highest layer is the executable applicable layer, which can finish all kinds of automatic test tasks and provide various test management service through visual friendly test operation and display interface as well as comprehensive data analysis and treatment functions. The second layer is equipment driving layer, which serves as the intermediate link to communicate with application programs above I/O interface software, providing software program set required to finish a particular equipment control and communication. It cannot only make good use of IVI, but also follow traditional equipment driving realization. The next layer is I/O base layer, which provides interface to bottom hardware driving and serves as software connector between computer and equipment. Generally, its communication with hardware is in accordance with VISA standard. The lowest hardware layer is the base of whole test system, containing equipment which communicates with computer by different I/O buses and responsible for data collection. The equipment module includes VXI, GPIB and serial interface etc. Therefore, hardware, I/O base, equipment driving and executable application program make up a bottom-up ATS architecture.

Current requirements of ATS software architecture mainly include:

- ATS software architecture should be open by nature and software module should be reusable. In the process of developing new ATS software, making full use of and upgrading existing module can reduce time and cost
- Instrument should be interchangeable, which means that when changing equipment, the original ATS software and TPS operation should not be affected
- TPS is portable, which means that TPS can be transplanted to new ATS without revision

To sum up, this paper proposes to use component-based ATS software architecture to solve reengineering issues of signal-based test program set in ATS, so as to provide architecture-related suggestions to the reuse of test-specific knowledge of ATS legacy system.

Component-based ATS software architecture: As a further development of object-oriented technology,

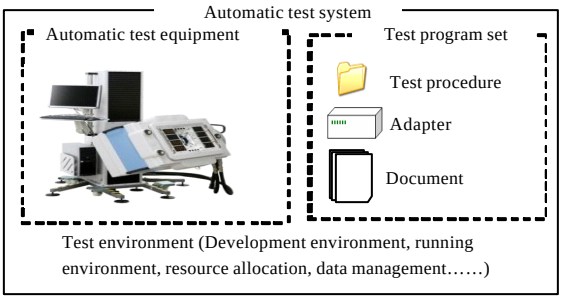

Fig. 1: Component of ATS

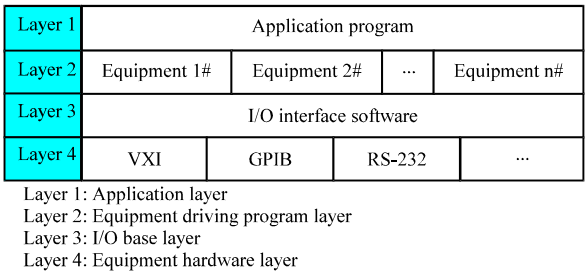

Fig. 2: Architecture of ATS

Signal-oriented test application program
Virtual resource extraction
Match process of resource
Component wrapper of actual resource
Instrument driver
Instrument hardware

Fig. 3: A typical signal-oriented software framework based on COM component technology

component technology packages modules for a certain purpose into a component so that customers can only call function module through component interface, whose inter-communication follows unified protocol. Different instrument hardwares share the same calling interface, as long as they are packaged in the same component interface. So changing instrument only needs to load the driving program component, without influencing the execution of TPS program. Component technology lays emphasis on the independence, interchangeablility and functionality of component. Practical and functional software system contains many components so as to make the reusability, openness and moduality of the system possible.

Component Object Module (COM), a component standard proposed by Microsoft, is widely used in Windows, offering interactive standard between components and their operation environment. A typical signal-oriented software framework based on COM component technology is shown in Fig. 3.

Since signal-oriented test language programming can realize TPS' portability and COM component technology makes instruments interchangeable by neglecting the difference between drivers, this architecture provides a convenient framework for the execution of signal-oriented test program. However, it has some drawbacks for instrument-oriented test, because the package of component is based on resource classification.

COMPONENT TECHNOLOGY AND ARCHITECTURE MODEL OF THE ATLAS COMPILER SYSTEM

Component and component-based software engineering: Generally speaking, component refers to a unit software with complete meaning, accurate grammar and reusable value. It is a recognizable unit in the process of software reuse (Zhang, 2006). Component-based software engineering emphasizes the use of reusable software "component" in the design and construction of computer-based system. In this way, the realization of many parts of the application solution is outsourced through modularization, which makes the selection, evaluation and assembling of component a principle way of software establishment.

For an actual system, system component is integrated by simpler low-level components, usually regarded as subsystem. In order to support the hierarchical decomposition of the system, we bring in the concept of composite component, i.e., system can be seen as a special composite component, which exists only as a common composite component when the system is integrated into a larger one. Correspondingly, atom component is the smallest unit that cannot be divided in software development. It has its own realized aspect; while composite component expresses the composition between its member components and do not have its own realized aspect (Zhang *et al*., 2001).

Zhang *et al*. (2001) have proposed a formal definition of component model:

- Component::= <component specifications, component realization>
- Component specifications ::= <interface part, structural part>
- Interface part ::= <function set of external provide, function set of external request, service set>
- Service ::= <atom component structure>| <composite component structure>
- Atom component structure ::= <quotation of component realization>
- Composite component structure ::= <quoted component type, example declaration, example connection, projection>

The architecture model of ATLAS compiler system: Compiler is a kind of computer software that can translate a high-level language into another language (target code). General compilers are composed by several successive parts, including lexical analysis, grammatical analysis and semantic analysis, generation of intermediate code, code optimization and generation of target code.

Generally, in accordance to different phases of the compiler realization and the relationship among them, the construction of regular high-level language compiler adopts the style of "pipe -- filter", as shown in Fig. 4. However, as a universal testing language, the "equipment selection and distribution" module in the construction of ATLAS compiler can adopt hierarchical software architecture, since it involves many semantic interactions with testing equipments.

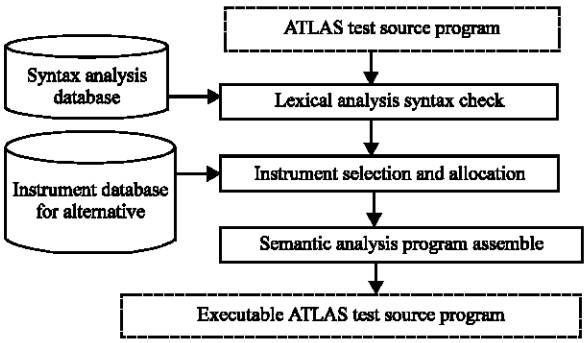

Fig. 4: ATLAS language compiler model

In Fig. 4, the lexical analysis and grammatical analysis can be completed with the help of relatively mature tools like LEX/YACC, so as to make developers concentrated on the feature analysis of source code language itself. But the resting parts have to be finished manually. Since the tested equipments covered in the testing objects of common ATS vary a lot in types and models and each has many testing items, the scale and items that need to be tested is very large. As a signal-oriented common testing language, ATLAS contains many sentences that require coordination of testing equipments (which might need frequent change). To translate these sentences involves many problems of semantic analysis which do not exist in the construction of common language compiler, e.g. how to select suitable equipment when different signals are tested and how to control these equipments so as to complete the required test of a certain or some signals. With regard to the same type of testing equipments produced by different factories, problems like how to differentiate and select can be solved in the "equipment selection and distribution" of ATLAS language compiler in Fig. 4.

ATLAS LANGUAGE COMPILER BASED ON COMPONENT

Typical ATLAS signal statement:

```
……
REQUIRE,'DMM3', SENSOR(RES), IMPEDANCE,
CONTROL, RES RANGE 0 OHM to 10000000 OHM continuous,
CNX HI LO $
……
VERIFY, (RES INTO 'IMPEDANCE'), IMPEDANCE,
NOM 1 KOHM UL 1.1 KOHM LL 0.9 KOHM,
RES MAX 50 KOHM,
CNX HI X2-A5 LO X2-C5 $
……
```

This is a use case of typical signal statement in ATLAS program. The test resource requirements that may be used in test program are given through "REQUIRE" statement in the beginning of test program "preamble segment" by tester. Each signal statement in test program "executive segment" needs to look for the matched virtual resource from all the virtual resources listed in "REQUIRE" statements. Then some (set of) appropriate actual resource(s) is selected from "instrument database for alternative" according to matched virtual resource to provide usage for test program.

For some virtual resource, there may be many instruments produced by different factory fitting the test requirement in "instrument database for alternative". Which actual instrument is chosen for the test procedure is decided by the bottom development platform to realize actual instrument test or by artificial selection. What need ATLAS compiler to do includes a series of operations, such as, to upload the driven program matched to the selected actual instrument, to detect the executive process of instrument and to complete the interaction with instrument etc. Once some actual instrument is selected, ATLAS compiler needs to upload the driven program of this selected actual instrument. Every operation designated by signal statement is executed by driven program corresponding to the selected instrument. Just previously mentioned, "instrument database" can deploy instrument made by different manufacturer. Test program can also select instrument made by different manufacturer to complete some test program. In conclusion, how to process the upload problem of instrument driven program made by different manufacturer flexibly by ATLAS compiler is a difficult problem.

The development environment selected of the ATLAS compiler construction (Li, 2006) participated by authors is Microsoft Visual Studio and the realization of component is Microsoft COM realization model. The communication and interaction among component objects and application programs is realized through COM specification and COM library provided by system. The COM library is realized by operating system and developing programmer needn't care for the realized details. This "flexibility" based on COM component technology provides us a good solution for our solving resource matching problem.

The typical software architecture diagram for semantic analysis realization of signal corresponding statement based on COM component technology is showed as Fig. 3. The component realization schematic diagram of the signal corresponding statement in ATLAS compiler is showed as Fig. 5.

According to the definition given by ZHANG ShiKun in reference 1, the formal definition of ATLAS test module semantic transformation is as follows:

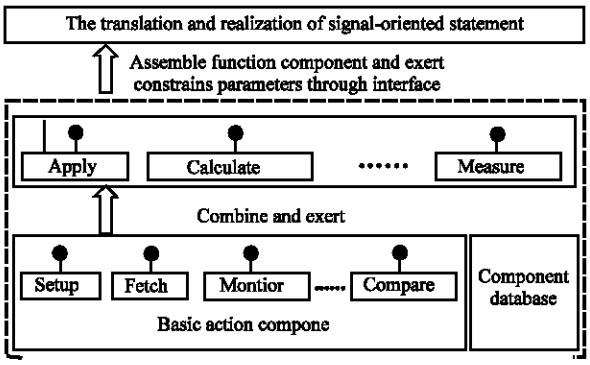

Fig. 5: Components realization schematic diagram of semantic translation to test module

```
Atom components include:
SETUP, FETCH, MONITOR, ……, COMPARE;
Composite components include:
APPLY, CALCULATE, ……, MEASURE;
The example for component description is as follows:
Component SETUP?=< IVariableCollectPtr pVariable, the detailed realization of atom component>
……
Component VERIFY?=<FETCH component, COMPARE component, …, example declaration, example connection, mapping>
……
```

Application and realization of component model: According to the frequently-used instrument operations during test process, the C++ interface illustration for resource deployment component during realization is as follows:

```
class __declspec(uuid("92527F51-331A-4F58-B38E-2C407277F00E"))
IDriver : public IUnknown
{ public:
    virtual HRESULT _stdcall setup(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall fetch(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall change(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall enable_event(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall disable_event(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall monitor(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall reset(IVariableCollectPtr pVariable) = 0;};
class __declspec(uuid("ED071509-E5AC-4057-ADA4-9F4185AC0D32"))
IRelay : public IUnknown
    { public:
    virtual HRESULT _stdcall connect(IVariableCollectPtr pVariable) = 0;
    virtual HRESULT _stdcall disconnect(IVariableCollectPtr pVariable) = 0;};
```

The above interface shows that every actual resource can provide methods, such as setup(IVariableCollectPtr pVariable), fetch (IVariableCollectPtr pVariable), change(IVariableCollectPtr pVariable), enable_event(IVariableCollectPtr pVariable), disable_event(IVariableCollectPtr pVariable), arm(IVariableCollectPtr pVariable), monitor(IVariableCollectPtr pVariable), reset(IVariableCollectPtr pVariable), compare(IVariableCollectPtr pVariable) etc., to realize the operation to actual instrument by COM component. The parameters transmission between test program and actual program and the pin connections to each actual resource are all transmitted through component interfaces (such as the formal parameter explanation at parameter location to each method in above code, the parameter requirements in method connect and disconnect etc.).

The driven programs corresponding to different instrument are all hided in the realization or COM component through above Communal COM component. COM component method is a good solution for realization of bottom test platform because it solves the unity call problem among various test instruments and provides concise solution for signal statement semantic transformation during realization of ATLAS compiler.

In addition, test of TPS containing domain-specific knowledge can also define relevant interface standard according to the realization method of semantic translation in above-mentioned translator system, so as to realize the reuse of legacy system through unified package and call.

CONCLUSION

The reuse of domain-specific knowledge in automatic test system has broad market prospect. Component-based software reuse can provide an important realization method for reusing legacy software. Aimed at the intrinsic features of ATLAS, this paper proposes to adopt COM component-based signal-oriented typical software framework in GTEST of proprietary intellectual right, so as to solve compiling issues of ATLAS test program. It makes use of the component technology of software reuse to resolve translating problems of signal-involved sentences in ATLAS compiler design. It also utilizes COM component technology to package interfaces of test instrument, well settles problems like correspondence between virtual resource and actual equipment and semantic translation of signal sentence and puts forward the issue of tackling reengineering of domain-specific knowledge component in ATS by component technology.

ACKNOWLEDGMENTS

This work is financially supported by the Science Research Project in Shanxi Province (20120321030) and Youth Foundation of Taiyuan University of Technology (No. 2012L067).

REFERENCES

Duan, L.G. and A.P. Li, 2005. Automatic test system modeling based on unified modeling language (UML). Proceedings of 6th International Symposium on Test and Measurement, June 1-4, 2005, Dalian, China, pp: 280-283.

Frakes, W.B. and K. Kang, 2005. Software Reuse research: Status and future. IEEE Trans. Software Eng., 31: 529-536.

Li, A.P., 2006. Research and realization of ATLAS translator based on visual test software platform GTEST. Ph.D. Thesis, XiDian University, Xi'an, China.

Li, X.S., Y. Zuo and J. Sun, 2004. Integration Technology of Automatic Test System. Electronic Industry Press, Beijing, China.

Liu, H., C.B. He, Y. Tang and S.P. Huang, 2012. Research on service-oriented scholar information service clouds platform. J. Convergence Inform. Technol., 7: 333-339.

Tracz, W., 1995. Confessions of a Used Program Salesman: Institutionalizing Software Reuse. Addison Wesley Longman. London, UK.

Yang, J., 2011. Research and application on software re-engineering theory. Master's Thesis, Anhui University, Anhui, China.

Zhang, S.K., W.J. Zhang, X. Chang, L.F. Wang and F.Q. Yang, 2001. Building and assembling reusable components based on software architecture. J. Software, 12: 1351-1359.

Zhang, Y.S., 2006. Software Architecture. TsingHua University Press, Beijing, China.

Zhao, H.B., 2003. Technical Specifications and System Integration of Virtual Instrument. TsingHua University Press, Beijing, China.