J I S i m e i
(School of Electro-mechtronic Engineering (02),
P.O.Box 327, Beijing, 100081, P.R. China)

SOFTWARE ENGINEERING AND ITS APPLICATION IN AEROSPACE INDUSTRY[1]

Based on a brief review to the history of Software Engineering (SE), important conceptions and technical terms in this field are introduced, including Life Cycle of SE, Process Model, Software Architecture, and Design Method of different systems. Following a simple description of its standardization course and famous commercial development tools, the application status of SE in aerospace industry are analyzed, both domestically and oversea. Finally, the expectations are expressed.

Great achievements have been made in Science, Technology as well as in Sociology after the second world war, while the most significant progresses came forth in Aerospace Industry and Information Technology. As an essential requisite, computer has been widely used in our daily life and almost all kinds of workplaces. Aerospace industry was once a promotive factor to the development of computational technology, now it has been greatly benefited from it. Taking the software as an example, it has become a part of the aerospace systems, and actively involved in all stage of the system's life cycles, from conceptual planning, design, analysis, simulation, verification, manufacturing to the end deploying and maintenance. In most of the top-ranking companies in this field, seamlessly integrated platforms, which include various kinds of software packets, are highly employed to carry out a full digitalized modeling, design, performance analysis and verification of the aerospace products.

From the traditional aerospace engineering tasks, it is well known that the developing process of such a system often involves multi-disciplines and a complete team of engineers; when this process is implemented with the help of software packets, then with no doubt the software is often complicated and on large scale. Meanwhile, due to the uncertainty of investigated issue under some circumstances, software must be multi-functional and highly adaptive. All these made the efforts to develop appropriative software packets by the researchers themselves, not only time and money consuming, but also with low efficiency and reliability. Hence, people often purchase commercial software products for most of the current tasks.

On account of the specialized performances and complicated developing process, this kind of software packets is often costly, which could be

[1]Статья публикуется в авторской редакции.

affordable only by big companies or institutes. Besides, due to the all-purpose orientation of these software packets, some given problems are beyond their working scopes. Further more, considering on the world political, economical and security concerns, some hi-tech products are not obtainable world-widely. Hence it is very important for small group of researcher to use the software engineering theory and practical experiences as reference when developing their own aerospace software packets.

1. Software Engineering. Software design has experienced several developing stages in the last several decades, from program design in the earlier years, to the program system design, till the software engineering after 70^{th} last century. Adopting the well-developed engineering management, control and review methods, applying them to software analysis, design, coding, test and maintenance phases, Software Engineering employs generic engineering theories and methods to develop software products. It mainly focuses on the following aspects like development feasibility of software tasks, software architecture, software design methods and toolkits used for software design. It is software engineering, which makes software becoming a scientific engineering subject.

1.1. Software Life Cycle and Process Model. The whole time period of software from its kick-off to the end of its usage is called Life Cycle. To reduce the complexity of software development and maintenance, software life cycle is often divided into the following stages; each stage has their definite tasks.

Planning. To define the objectives of software project, and determine available resources and budget for its implementation. To study both technical and economical feasibility of the project, and estimate the project schedule.

Requirement Analysis. To discuss the customer's requirements. This stage is fulfilled through the cooperation of customers and software engineers, the final requirements should be explicit.

System Design. To convert all requirements into logic modules, each module answers certain requirements.

Program Coding. To code the program modules using a selected programming language.

Test. To verify the correctness and reliability of the software, usually startup with separate module tests, then assembled software test.

Operation and Maintenance. To correct bugs and faultiness in software, which are discovered through appliance of software product

In the life cycle of a software product, different projects usually adopt different Process Models depending on its scale, complexity and characteristics. Frequently used models including Waterfall Model, V model, Incremental Model, Evolutionary Model, Spiral Model, Fountain

Model and Intelligent Model, etc. Different models organize and implement the life cycle stages in different orders or sequences. A small scale software project with clear requirements will usually adopt the waterfall model for instance, the 6 stages of software life cycle will be implemented in sequential order, each stage generates its semi-finished products, and the target software product will be generated in the end. For projects which have only blurry system requirements, requirements can't be clearly defined or requirements are rapidly changing, Incremental Model or Evolutionary Model are often proposed, these two types of model permit to develop a software system based on limited requirements and when requirements being updated or changed, software can be quickly reinforced and amended from the beginning.

1.2. Software Architecture and Software Design Method. Software Architecture reflects the high level architecture of a system. There is yet a standard definition, but it often indicates the structure or structures with which the software system is constructed, they form all portions of the system, both the external characteristics and relationships between different portions.

Software Architecture is a system abstraction on a higher level, which reveals the earliest design of the system. It simplifies the system components and their inner relationships; however it can also distinctively draw the existing problems in the system. If each part of a software system can fluent communicates with each other in the selected architecture, then the usability and quality of the system can be more easily guaranteed. This also helps to improve the reusability of software components.

Several aspects should be mainly considered when the software architecture is built, i.e. how to organize all components and integrate them into a system; how to design the control structure and data flow in the system; how to solve the communication problems between data and control info; how to define communication protocols and interface control mode; how to mark off functions among components and modules; how to evaluate the performance and expandability of the system.

The representative architectures for software includes pipe-filtering system, hierarchy system, virtual pattern for simulating additional functions of hardware or software to keep the product transplantable, client / server system, data based system like warehouse system, hyper text system or blackboard system etc.

To different software architectures, different design methods can be implied. In principle, 3 types of methods for software development are often hired, (1) data-driven analysis and design method; (2) function-driven analysis and design method; (3) Object Oriented analysis and design method.

First type of method is mainly used for data-based systems, like warehouse system and hyper text system. Representative usage for function driven method is structural analysis and design, it takes a top-to-bottom approach and breaks down a complex system into a series of subsystems and subsystems into modules, each module accomplishes a simple function and keeps corresponding independency, thus makes the whole system easy to be implemented and maintained. Object oriented method combined the first 2 types of methods, is object-driven method, where object consists of both data and function. It starts up from analyzing the component objects in a system, taking the bottom-to-top approach to finish analysis and design. The main characteristics for object oriented method include: encapsulation of data and function makes objects easy to be expanded, inherited attributes of the son class from father class keeps software easy to be evolved, and messages link the objects dynamically.

Currently, Object Oriented method is a hotspot of investigation, but Structural Analysis and Design is still the well matured and widely used method in software engineering, especially in aerospace industry, its only shortage is the low reusability.

1.3. Standardization in Software Engineering. The concept of Software Engineering was firstly been brought forward in 1969, hence, the standardization efforts for it have never been ceased in international standardizing organizations. After 30 years' progress, the major standards for software process, technology, methods, tools and managements have been fully developed. The most remarkable one is the "Software Engineering Body of Knowledge (SWEBOK)", published in 2001 by Union of ISO and IEC technical committee. In this document, software engineering discipline was subdivided into ten Knowledge Areas, which included Software Requirements, Software Design, Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, Software Engineering Tools and Methods, Software Quality. For each knowledge area, detailed descriptions and specifications are given in this guide.

Taking the Software Requirements as an example, it is separated into 6 sub-areas:

Requirement Engineering Process. Objective of this sub-area is to decompose complex system into simple components.

Requirements Elicitation. Based on use case design and pattern design to recognize different software requirements, including customer's aim, application field of knowledge, operational and organizational background of the system etc.

Requirements Analysis. Using a series of analysis tools to detect and resolve conflicts between requirements, to discover the bounds of the

system and how it must interact with its environment, to elaborate system requirements and software requirements. For different design methods, analysis tools are different. Structural Analysis and Design uses data flow diagram, state transfer diagram, control flow diagram, data dictionary with process descriptions. For object oriented method, data flow diagram, state transfer diagram, entity-relationship diagram and use case analysis tools are employed.

Software Requirements Specification. In accordance with the requirement specification template, to create software requirement specification descriptions, numbering each items of requirement and giving clear indication of their origination. To create requirement tracing matrix, thus control project schedule.

Requirements Validation. To check the uniformity, integrity, validity and feasibility of requirements.

Requirements Management. To trace and control the update process of requirements. If an item is changed when the project is in progress, impact of the requirement change should be analyzed and recorded. All status of each item must be tracked.

Based on appropriate requirements analysis, the working group may choose proper process model and work out project developing plan; along with the project development progress, requirements concerned risks should be evaluated and mitigated.

Resembling Requirement Engineering, all the other 9 knowledge areas have been clearly defined and specified. Software Design contains both high level design and detailed design, which is separated into 6 sub-areas, including: basic concepts, key issues of software design, structure and architecture, software design quality analysis and evaluation, software design notations and software design strategies and methods.

1.4. Computer Aided Software Engineering (CASE).

Along with the engineering development and management of software, methods and tools used in software project are dramatically changing, more and more CASE tools are appearing.

CASE tools may consist of different components. Comparing with different stages of software life cycle, CASE tools can be divided into 2 types, high level tools and low level tools. The high level CASE tools may automatically form the project plan; help to generate requirements and requirement description documents; and lay the project course. Low level CASE tools help to implement software coding, testing and maintaining works automatically. At present, one can either find independent high level CASE tools, like Visio in Microsoft office packet, or integrated CASE tools, which combined both high and low level of design tools in a uniformed development environment, like STP from IDE company, Rose integrated

packet from Rational company. In China, the Blue Bird I System can support Waterfall process model and structural analysis and design method for software development. Its upgrade version Blue Bird II may support object oriented analysis and design method.

Using these CASE tools reasonably and skillfully can dramatically increase software productivity with high quality and low costs.

2. Application of Software Engineering in Aerospace Industry. Software Engineering has achieved great progress in the last half century; nevertheless, software crisis still exists. World-widely, 80∼90 % projects in information field can't reach their original target, 80 % exceeds their budgets, and around 40 % projects failed at the end. Figures and Situations in China are still worse[2].

2.1. Software Management and Acceptance at NASA. As the most famous aerospace organization in the world, NASA started up its software research works roughly 30 years ago. In 1976, Software Engineering Laboratory (SEL) came into existence in Goddard Space Flight Center (GSFC). Based on a series of research works, the laboratory has established corresponding standards for software process and acceptance.

Based on the experiences from many years' software projects, NASA has summarized a conclusion: there is no single solution for all problems; no any life cycle models, analysis and design methods, or test methods can fit all NASA software projects. To satisfy different requirements, there must be individual plan for each project. Therefore, NASA prompted its own document "NASA software management guide" to manage and regulate software development contract.

Generally speaking, aerospace software engineering has achieved significant progresses in software quality, reliability, reusability within these years. Many specialized CASE tools can provide integrated platform for digitalized flight vehicle design and simulation, among which, ModelCenter from Phoenix Integration company, iSight software packet by Engineous Software company and AML of Techosoft company are some examples of these CASE tools. Many leading aerospace companies are taking the advantage of these tools to win more and more contracts relying on effective and high quality designs created by them.

2.2. Domestic Applications of Software Engineering in Aerospace Field. The commencement of Software Engineering related works in China has experienced several decades' delay comparing to the world. In the past 10∼20 years, progresses have been made and a set of national standards have been established to standardize software engineering. But the software industry still needs more time and efforts to catch up the world's paces.

[2]http://media.ccidnet.com/media/ciw/1315/d0901.htm

In aerospace industry, software engineering concepts have been widely spreaded since 1996. On one of the aerospace industry's management meetings held in 1998, among the other work arrangements, several more items were listed, i.e. to improve aerospace software quality, to establish software certification criterions and acceptance standards in aerospace industry. Now though the situation is still far from satisfying, progresses are steadily achieved.

3. Conclusions and expectations. As a result of mixed reasons, software industry in China is still underdeveloped. Few work has been done concerning on software reusability, both theoretically and in practice. Wide gaps still exist comparing with many other countries. Hence it will be a primary task for our engineers in the near future, to realize the importance of software engineering and its impact in other fields, to master the basic theory, to use correct methods and tools, to learn from the experiences and lessons obtained by others, and to strive for more progresses in this field.

REFERENCES

1. Ian Sommerville. Software Engineering, Machinery Industry Publishing Company, Beijing, 2003. 1.
2. Shari Lawrence Pfleeger. Software engineering — theory and practice. Tsinghua University, Beijing, 2003. 8.
3. Len Bass, Paul Clements. Software architecture in practice. Tsinghua university, Beijing, 2002. 10.
4. Scott W. Ambler, Object Oriented Software Engineering. Machinery Industry Publishing Company, Beijing, 2003. 6.
5. Ian Sommerville, Pete Sawyer. Requirement Engineering, Machinery Industry Publishing Company, Beijing, 2003. 8.
6. Software Engineering Laboratory, SWEBOK, version 0.95, 2001. 5.
7. Zhou Su, Wang Wen. Software Engineering, Science publishing company, Beijing, 2002. 9.

Статья поступила в редакцию 8.07.2005

JI Simei (Цзи Симэй), acquired her PhD at Bauman Moscow State Technical University (МГТУ им. Н.Э. Баумана) in 2001. She works in Bijing Institute of Technology as an associate professor since 2002, and currently involves mainly in pedagogic as well as research works in fields of Flight Dynamics and Control, Flight Vehicle Design and Computer Aided Design.

Симэй Цзи защитила диссертацию на звание канд. техн. наук в 2001 г. в МГТУ им. Н.Э. Баумана. Доцент Пекинского технологического института. Специализируется в области динамики полета и управления полетом летательных аппаратов, проектирования летательных аппаратов, в том числе с использованием методов компьютерного моделирования.