항공전자 시스템 소프트웨어의 개발 추세에 대한 연구

양성욱* · 이상철**

A Study on the Trend of an Avionics System Software Development

Sungwook Yang* · Sangchul Lee**

ABSTRACT

The importance of software development in the integration of an avionics system is continuously increasing. And we can expect enlarged software portion in the system integration for the more intelligent, reliable, and automated avionics system. For an avionics system software development the selection of a real-time operating system and internal avionics data bus protocol is very important from the viewpoint of the integration with the system hardware. In this paper, we present current trend of an avionics system software development including software development methodology, software development process, and international software process assessment improvement model.

Key Words: Avionics System(항공전자 시스템), Software Development(소프트웨어 개발), Real-Time Operating System(실시간 운영 체계), Software Development Process(소프트웨어 개발 절차)

1. 서 론

현대 항공기에 있어 항공전자 시스템의 중요도와 비용측면에 있어 그 비중이 계속 증가되고 있다. 이는 전기전자기술의 급속한 발달에 따른 항공전자기술의 빠른 발전과 나날이 증가되는 임무 요구사양에 그 원인이 있다 하겠다[1-3]. 항공전자 시스템의 요구사양은 하드웨어와 소프트웨어로 구현되는데, 연산속도(throughput)와 메모리 크기의 비약적인 발전 및 사용자의 복잡하고 다양한 요구사양으로 인하여 고 신뢰성, 자율성, 제어법칙 구현 등 현대의 다양하고 복잡한 기능 중 많은 부분이 소프트웨어적으로 구현되고 있다. 또한 현재의 개발 추세에 따라 상용화된 기성 개발품(COTS)을 활용하고 개방형 시스템 아키텍처(Open System Architecture)를 적용하게 되는 경우 항공전자 시스템 통합에서 소프트웨어의 역할은 과거와는 비교할 수 없을 정도로 중요하다. 더 지능적(Intelligent)이며 자동화된 항공전자 시스템을 개발하기 위해서 소프트웨어의 비중은 계속 늘어날 것이 예상된다. 국내의 많은 산업체들이 시스템과 소프트웨

†2007년 2월 20일 접수 ~ 2007년 3월20일 심사완료
* 한국항공대학교 항공우주 및 기계공학부 대학원
** 한국항공대학교 항공우주 및 기계공학부
연락저자, E-mail: slee@hau.ac.kr

어 영역을 통합시킨 산업체의 프로세스 개선 활동에 대한 평가 모형인 CMMI(Capability Maturity Model Integration) 인증을 획득하고 있는 것도 시스템 통합차원에서 소프트웨어의 중요성에 기인한다고 볼 수 있다. 본 논문에서는 항공전자 시스템 소프트웨어 개발 시 고려해야할 실시간 운영체계, 내부통신 프로토콜, 개발 방법론, 개발절차, 평가 모형 등 전반적인 사항과 그 동향에 대하여 살펴보기로 한다.

2. Software 개발

2.1 실시간 운영 체계

운영 체계는 interrupt 처리, 메모리 관리, 하드웨어의 I/O drivers와 error 관리 등을 제공한다. 특히 항공전자 시스템의 운영 체계는 신속한 처리를 위한 짧은 run-time kernel, 최소화된 latency 등을 가져야 하는 hard real-time 운영 체계를 요구한다[4]. 실시간 운영 체계는 지속적인 처리 속도 와 메모리 용량의 증가로 인해 Multi Thread Model에서 Multi Process Model로 변화하고 있다. Multi Thread Model은 운영 체계 kernel과 application이 합쳐져서 서로의 구분이 없는 하나의 큰 프로그램으로 작동하는 구조이며 공통의 작업영역(Memory)을 자유롭게 접근(access)하고, 운영 체계의 크기가 작고 비교적 작은 크기의 시스템에서 구현이 쉽고 빠르다. 하지만 kernel과 application이 하나의 프로그램으로 동작하기 때문에 사소한 버그(bug)가 시스템 전체를 파괴할 수 있다. 대표적인 운영 체계로 VxWorks, OSE, VRTX, pSOS, Nucleus Plus 등이 있다. Multi Process Model은 운영 체계 kernel

Table 1. Real-Time Operating Systems

RTOS	Vendor	Processor Supported	Min Size	Development Hosts	Network Support	Specialty Features
pSOS	Integrated Systems >WindRiver	68k	6KB	PC, Sun, HP, native	TCP/IP, Socket	Multiprocessor Support
pSOSelect	Integrated Systems	68k	2.5KB	PC, Sun		
pSOSystem	Integrated Systems	68k, x86, PowerPC,i960	15KB	PC, Sun, HP	TCP/IP, SLIP, FTP, PPP, OSI, Streams, NFS	Multiprocessor Support, SNMP
C Executive	JMI Software Systems	68k, x86, 29k, i960, MIPS, PA-RISC, SPRAC, PowerPC	6KB	PC, Sun	TCP/IP	Multiprocessor Support, SNMP
VRTX	Microtek Research > Mentor Graphic	68k, x86, SPARC	16KB	PC, Sun	TCP/IP, Streams, NFS, RPCs, SLIP	SNMP
OS-9	Microwave Systems	68k, x86, PowerPC	22KB	PC, Sun	TCP/IP, ISDN, SLIP	MS-DOS, Unix
VxWorks	WindRiver Systems	68k, x86, 29k, i960, MIPS, SPRAC, PowerPC	5KB	PC, Sun, HP, SGI, IBM	TCP/IP, Socket, NFS, FTP	Multiprocessor Support, VxMP, SNMP

이나 각 application들이 모두 독립적인 프로그램으로 동작하도록 설계되었고, 각 application은 서로의 메모리가 보호되어 있기 때문에 모듈 단위의 application 개발이나 모듈(기능)의 추가 와 변경이 쉽고 안정된 시스템 개발이 가능하다. 대규모의 시스템 개발이 용이한 장점이 있으나 실시간 운영 체계의 크기가 Multi Thread Model에 비해 크기 때문에 작은 시스템 개발에는 오히려 부담이 된다. 대표적인 운용체계로는 QNX, OS-9, LynxOS, RT-Linux, Windows CE 등이 있다. 실시간 운영 체계의 개략적 비교는 Table 1과 같다[4-6].

2.2 내부통신 프로토콜

항공전자 시스템의 데이터 버스로서 내부통신 프로토콜의 선정은 매우 중요하다. 신뢰도와 속도가 내부통신 프로토콜 선정의 중요한 요인이라 할 수 있는데 현재는 MIL-STD-1553B와 ARINC 429가 많이 사용되고 있다. 빠른 전송속도로 인한 Ethernet의 도입과 무장제어 분야에서 MIL-STD-1760의 적용이 주목된다. 최근 ARINC 429는 ARINC 629로 대체되고 있는 추세이나 ARINC 629의 coupling device는 Ethernet에 비해 크기가 크고 MIL-STD-1553B에 비해 무게가 무겁다[7]. Table 2에서 대표적인 내부통신 프로토콜을 비교하였다[7-10].

2.3 소프트웨어 개발 방법

소프트웨어 개발 방법은 구조적 방법론(Structured Methodology)에서 재사용성과 확장성이 뛰어난 객체지향 방법론(Object Oriented Methodology)을 사용하는 추세이다. 구조적 방법론은 1970~1980년대 각광을 받은 기법으로 계층적 분할, 논리적 시스템 모델 구축, 하향식(Top-Down) 접근방법이다. 이 방법은 데이터의 흐름과 구조를 중심으로 하며 사용하는 대표적

Table 2. Internal Avionics Data Bus Protocol

	MIL-STD-1553B	RS-232C	ARINC-429	ARINC-629	Ethernet
속 도	1 Mbps	최대 19.2 Kbps	12.5 Kbps 또는 100Kbps	2 Mbps	10 Mbps, 100 Mbps, 1 Gbps
적용 분야	항공, 우주, 군사용	일반용	항공, 우주, 군사용	항공용	인터넷
통신 방식	Half-Duplex (반-이중) (1 : n 통신)	Half-Duplex (Point-to-Point) (1:1 통신)	Simplex(단방향) with Multiple Receivers (1 : n 통신)	Full-Duplex (m : n 통신)	Half-Duplex (반-이중) (m : n 통신)
전송 매체	Coaxial Cable	Twisted Pair	Twisted Pair	Twisted Pair	Coaxial Cable 또는 Optical Fiber
유효전송거리	6 m	20 m	53 m (175ft)	100 m	200 ~ 500 m
특 징	Bus Controller가 Remote Terminal 들을 제어한다.	Transmit/Receive를 위한 동일한 Channel이 필요함	Transmit/Receive를 위한 독립된 Channel이 필요함	각각의 terminal이 data bus상의 다른 모든 terminal에 대해 Transmit/Receive이 가능함	Pear-To-Pear 통신을 수행함
항전 시스템 적용 분야	항공/우주분야의 Avionics System Bus에 적용됨	항공/우주 분야의 LRU와 LRU 사이의 저속 통신에 사용됨	항공/우주 분야의 LRU와 LRU 사이의 저속 통신에 사용됨	B777에 최초로 적용된 후 항공/우주 분야에 적용되는 단계임	인터넷 분야에 주로 응용되나, 빠른 통신속도로 인하여 항공/우주 분야에 적용되는 단계임

인 프로그램 언어로는 FORTRAN77, C, Ada83 등이 있다. 1990년대 이후, 객체지향 방법론이 각광을 받기 시작했는데 객체지향 개념을 적용한 소프트웨어 설계는 데이터와 데이터를 다루는 Function 및 Procedure를 하나로 묶어 객체(Object)라는 개념을 사용하여 실세계를 표현하는 방식이다. 객체지향 방법에서는 객체의 속성과 연산의 식별이 가능하고, 객체의 관계성을 정의할 수 있으며, 상향식(Bottom-Up) 개발이 주로 사용된다. 캡슐화(Encapsulation), 다형성(Polymorphism), 상속성(Inheritance) 등을 그 특징으로 한다[11-13]. 이 방법을 사용하는 대표적인 프로그램 언어는 C++, Ada95, JAVA 등이 있다.

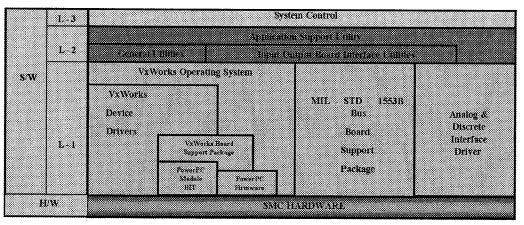

Fig. 1 Layered Software Structure

Fig. 1은 소프트웨어 모듈간의 독립성을 위해 사용되는 3-Layer 형태의 소프트웨어 개발 예를 보여준다. 이 경우 상위 계층의 Application 작성자는 하드웨어의 상세한 특성에 구애됨이 없이 Application Program 작성이 가능하다. 그리고 하드웨어의 변경이 발생할 경우에도 Layer-1의 해당 Driver 소프트웨어만 변경하면 되고 상위 Layer의 소프트웨어 변경을 최소화할 수 있는 장점이 있다[2].

2.4 소프트웨어 개발 절차

1995년 ISO, International Electrotechnical Commission에서 소프트웨어 개발과 관리를 위한 공통의 framework을 제공하는 Information Technology - Software Life Cycle Processes인 ISO/IEC 12207를 제정하였고 1998년에는 IEEE, Electronic Industries Association에서는 소프트웨어의 개발에 관한 이해와 적용을 위해 명확한 개념정의와 가이드 라인 제공하는 Software Life Cycle Processes인 IEEE/EIA 12207를 제정하였다. IEEE/EIA 12207는 ISO/IEC 12207의 내용을 전부 포함하고 있으며 같은 해 DoD(미국방성)에 의하여 채용되었다. Fig. 2는 IEEE/EIA의 발전과정이고, Fig. 3은 ISO/IEC 12207 프로세스 구성체계이다.

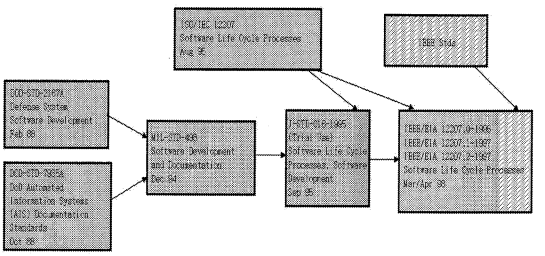

Fig. 2 IEEE/EIA 12207 History

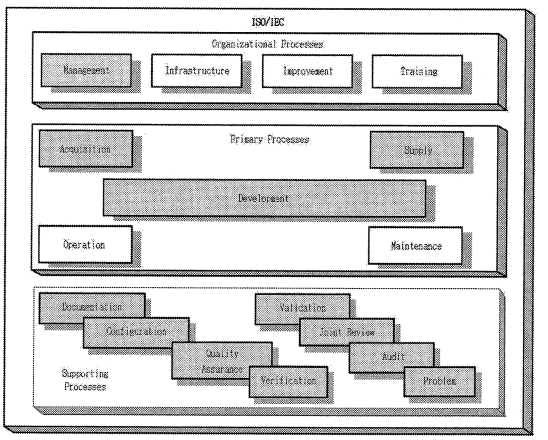

Fig. 3 ISO/IEC 12207 Process

소프트웨어를 개발함에 있어서 고려되어야 사항들은 요구사항 분석, 설계, 구현, 통합, 시험, 설치 및 인수 등 소프트웨어 제품 개발과 관련된 활동이 있으며, 이 사항들의 순차적인 일련의 과정이 소프트웨어 개발의 프로세스이다. Fig. 4는 소프트웨어 개발의 프로세스를 보여준다. 각각의 과정에서는 개발규모 및 성격에 따른 각 개발단계별 Documentation Tailoring

및 작성이 중요하다. Table 3은 MIL-STD-498을 적용한 소프트웨어 개발[14]에서 Documentation tailoring에 의해 산출되는 소프트웨어 문서의 예이다.

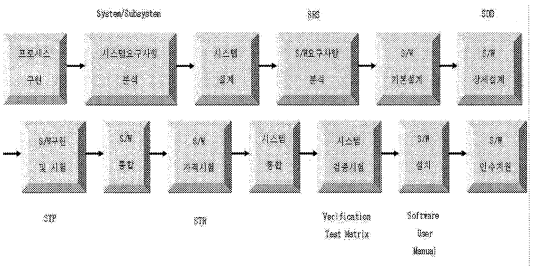

Fig. 4 Software Development Process

Table 3. Software Documentation

주요문서	기타문서
SDP(Software Development Plan)	
Operational Concept Description	Software Installation Plan
System/Subsystem Specification	Software Transition Plan
System/Subsystem Design Description	Software Product Specification
SRS (Software Requirements Specification)	Software Version Description
IRS (Interface Requirements Specification)	Software User Manual
SDD(Software Design Description)	Software Center Operator Manual
IDD(Interface Design Description)	Software Input/Output Manual
DDD(Database Design Description)	Computer Operation Manual
STP(Software Test Plan)	Computer Programming Manual
STD(Software Test Description)	Firmware Support Manual
STR(Software Test Report)	

2.5 평가 모형

평가 모형은 소프트웨어 능력 평가(Software Capability Evaluation)와 프로세스 개선을 목적으로 한다. 소프트웨어 프로세스의 진단(Software Process Assessments)을 통하여 조직의 현행 소프트웨어 프로세스의 상태를 진단하고 평가하여 소프트웨어 개발 중의 문제점을 개선할 수 있으며 결과적으로 품질 및 개발생산성 향상을 가져온다. 소프트웨어 능력 평가를 통해 발주기관은 소프트웨어 사업자 평가에 프로세스 평가 모형의 결과를 활용하고, 시스템 통합 사업자 평가에 대한 전문성 및 공정성을 확보할 수 있으며, 소프트웨어 사업자는 국제적으로 인정된 기준에 의하여 객관적 능력을 인정받아 대외 신인도를 제고할 수 있다. 국제적으로 인증된 평가 모형으로는 미국 CMU SEI의 CMMI, 유럽의 SPICE(ISO/IEC 15504)가 대표적이다. Fig. 5는 CMMI의 발전과정을 보여주고 있다[15].

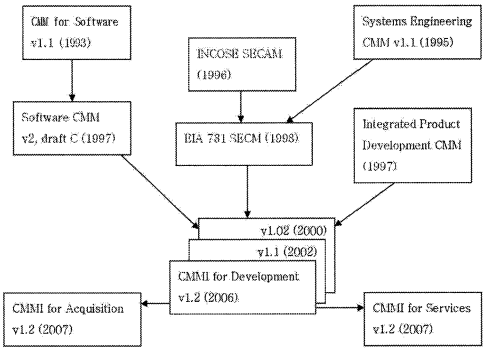

Fig. 5 History of CMMs

CMMI는 완성도(maturity levels)를 통해 분류된 22가지의 process area를 평가함으로써 각각의 process 완성도를 판단할 수 있으며, 또한 개선이 필요한 부분을 찾아낼 수 있다. CMMI에서 제시하는 완성도는 5단계(1~5 Levels)로 구성되어 있다. Table 4는 완성도를 구성하는 process area를 보여준다[15,16].

Table 4. Maturity Level and Process Area

Maturity Level	Characteristics	Process Area
5	continually improves its processes based on a quantitative understanding of the common causes of variation inherent in processes	Causal Analysis and Resolution Organizational Innovation and Deployment
4	the organization and projects establish quantitative objectives for quality and process performance and use them as criteria in managing processes	Organizational Process Performance Quantitative Project Management
3	processes are well characterized and understood, and are described in standards, procedures, tools, and methods.	Decision Analysis and Resolution Integrated Project Management +IPPD Organizational Process Definition +IPPD Organizational Process Focus Organizational Training Product Integration Requirements Development Risk Management Technical Solution Validation Verification
2	the projects of the organization have ensured that processes are planned and executed in accordance with policy	Configuration Management Measurement and Analysis Project Monitoring and Control Project Planning Process and Product Quality Assurance Requirements Management Supplier Agreement Management

SPICE (Software Process Improvement and

Capability dEtermination : ISO/IEC TR3 15504)는 S/W 프로세스에 대한 개선 및 능력 측정 기준으로 ISO/IEC 에 의하여 제정되었다. SPICE의 특징은 2가지 차원의 평가 모델을 기준으로 측정한다는 것이다. 2가지 차원은 프로세스 차원 (Process Dimension)과 프로세스 능력 차원 (Process Capability Dimension)으로 구분된다. 우선, 프로세스 차원은 5개의 프로세스 카테고리가 40개의 세부 프로세스로 구성되어 있고, 각 프로세스 별로 목적을 달성하기 위한 기준을 제시하고 있다. 다음으로 프로세스 능력 차원은 조직단위(Organization Unit)가 특정 프로세스를 달성하거나 혹은 달성 가능한 능력 수준을 판단하는 것으로 6개의 Capability Level(0~5 Levels)로 구성되어 있다. Fig. 6는 SPICE의 평가 체계를 보여준다.

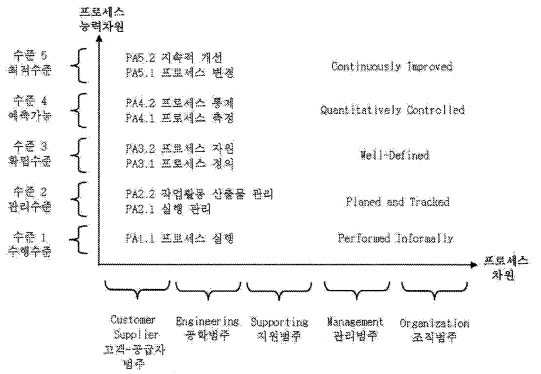

Fig. 6 Two Dimensional Model of SPICE

3. 결 론

항공전자 시스템 소프트웨어의 중요성은 계속 증가하고 있으며 개발에 사용되는 기법도 발전을 거듭하고 있다. 본 논문에서는 항공전자 시스템 소프트웨어 개발 시 필수적으로 고려해야 할 주요 사항들과 그 동향에 대하여 살펴보았다. 지속적인 프로세서 처리속도와 메모리 용량의 증가로 인해 여타 분야에서는 실시간 운영체계로 RT-Linux나 Windows CE 같은 Multi Process Model이 도입되고 있으나 현재 항공전자 시스템의 소프트웨어 개발 시 실시간 운영체계는 Multi Thread Model이 사용되고 있으며 이 중 VxWorks가 가장 많이 사용되고 있다[4]. 내부통신 프로토콜은 빠른 전송속도를 가진 Ethernet과 ARINC 629의 적용이 주목되며 소프트웨어 개발기법은 재사용성과 확장성이 뛰어난 객체지향 방법론이 늘어가는 추세이다. IEEE/EIA 12207과 같은 표준절차를 통한 개발에 있어 개발 규모 및 성격에 따른 각 단계별 Documentation Tailoring과 개발 문서의 작성이 긴요하며, 평가모형의 인증 획득을 통한 지속적인 개발생산성 및 소프트웨어 품질 향상이 필요하다 하겠다.

참 고 문 헌

[1] Newport, J. R., Avionic Systems Design, CRC Press, 1994

[2] 이상철, 김인규, 김영일, “무장관리컴퓨터 탑재소프트웨어 개발”, 한국항공우주학회지, 제 31권, 제 5호, 2003, pp. 124-133.

[3] 오우섭, 김성우, 최이주, 장진석, “한국형 헬기(KHP) 임무탑재장비 개발방안 연구”, 한국항공우주학회 추계학술발표회, 2005. 11.

[4] Cedeno, W., Laplante, P. A.,"An Overview of Real-Time Operating Systems", Journal of the Association for Laboratory Automation, Vol. 12, No. 1, 2007, pp. 40-45.

[5] Cooling, J.,"Real-Time Operating Systems for the Embedded World", IEE Open Control Systems - The Importance of Industrial Standards, 2004, pp. 4/0-4/10.

[6] Wind River Systems, Inc., VxWorks Programmer's Guide, March 1997

[7] Rieckmann,N., ARINC 629 data bus physical layer technology, Microprocessors and

Microsystems, Volume 21, Issue 1, 1997, pp. 13-20.

[8] Moir, I., Seabridges, A., AIAA Education Series Civil avionics systems, AIAA, 2003

[9] Moir, I., Seabridges, A., AIAA Education Series Aircraft systems, AIAA, 2003, pp. 315-337.

[10] Audsley, N. C., Grigg, A., "Timing analysis of the ARINC 629 databus for real-time application", Microprocessors and Microsystems, Vol. 21, Issue 1, 1997, pp. 55-61.

[11] Jacobson, I., Booch, G., Rumbaugh, J., The Unified Software Development Process, Addison-Wesley, 1999

[12] Douglass, B. P., Real-Time UML - Developing Efficient Objects for Embedded System, Addison-Wesley, 1998

[13] Smith, M. A., Object-Oriented Software in Ada95, International Thomson Computer Press, 1996

[14] MIL-STD-498 Software Development and Documentation, 1994

[15] Carnegie Mellon Software Engineering Institute, CMMI® for Development, Version 1.2, 2006

[16] Kossiakoff, A., Sweet, W. N., System Engineering Principles and Practice, John Wiley & Sons, Inc. 2003