인공위성 소프트웨어 개발동향

한국항공우주연구원 이종인
한국과학기술원 차성덕*

1. 서 론

인공위성 탑재 소프트웨어는 사람의 두뇌에 해당하는 부분으로 인공위성에 요구되는 모든 임무를 수행할 수 있도록 인공위성을 전체적으로 제어하는 기능을 수행한다. 인공위성에 요구되는 기능이 점차 증대됨에 따라 탑재소프트웨어의 기능도 점차 고도화되고 복잡한 임무를 수행할 수 있도록 발전해 가고 있다.

인공위성 탑재소프트웨어가 일반 소프트웨어와 다른 가장 큰 특징은 우주 환경에서 동작할 수 있도록 특수 제작된 탑재컴퓨터의 제한된 환경에서 정해진 작업이 일정한 시간 내에 수행되어야 하는 실시간 내장형 시스템이며 오류가 허용되지 않는 고도의 신뢰성과 지상의 제어 없이도 장기간 동작할 수 있는 자동화 기능이 요구된다는 점이다. 본 논문에서는 인공위성 탑재소프트웨어의 기능과 요소 기술에 대하여 소개하고 그 동안 국내에서 개발된 위성의 탑재소프트웨어에 대해서 기술하였다. 그리고 해외 위성 탑재소프트웨어의 전반적인 기술동향에 대해 분석하였다.

2. 탑재소프트웨어 개발기술

2.1 탑재소프트웨어 기능

탑재소프트웨어의 기능은 일반적으로 위성의 자세제어, 전력제어, 열 제어, 탑재체 제어 등을 수행하는 위성 제어 기능, 지상으로부터 명령을 수신하여 처리하고 위성의 상태 데이터와 임무 데이터를 저장하고 전송하는 위성 데이터 처리 기능, 임무에 따라 소프트웨어를 구동시키고, 상태를 모니터링하고, 이상 상태 발생 시 이를 감지하고 복구하는 시스템 관리 기능으로 구분할 수 있다.

2.2 탑재소프트웨어 요소기술

탑재소프트웨어를 개발하기 위해 필요한 기술은 위성 제어 소프트웨어 기술, 시스템 관리 소프트웨어 기술, 데이터 처리 소프트웨어 기술, 개발된 소프트웨어를 검증할 수 있는 소프트웨어 검증기술, 그리고 소프트웨어 재사용 기술로 구분할 수 있다.

위성 제어 소프트웨어는 위성의 자세를 제어하는 자세제어, 전력을 제어하는 전력제어, 우주환경에서의 열을 제어하는 열제어 그리고 탑재체 제어를 수행하는 것으로 각종 센서와 구동기를 이용하여 규정된 정확도를 유지하도록 제어하는 기술이 필요하다. 특히 자세제어 기술은 관성센서, 별 추적기 등의 센서 데이터로부터 자세를 결정하고, 구동기에 제어 명령을 내리는 절차가 정해진 시간 내에 반드시 처리되어야 하는 엄격한 실시간 처리가 요구된다.

시스템 관리 소프트웨어의 개발에는 소프트웨어가 탑재되는 탑재컴퓨터 환경에서 동작할 수 있도록 시스템을 초기화하고, 타이머, 인터럽트, 하드웨어 인터페이스를 위한 각종 드라이버를 제공하고 실시간 운영체제와 연동하여 실시간 스케줄링을 수행하는 시스템 소프트웨어와 위성의 각종 이상상태를 탐지하고 처리하는 장애관리(Fault Management) 및 자율성(on-board autonomy) 등에 대한 기술이 요구된다.

데이터 처리 소프트웨어는 지상으로부터의 명령을 처리하고, 위성에서 생성되는 각종 위성 상태 데이터와 임무 데이터를 실시간 전송 혹은 저장 후 전송하는 기능을 수행하는 것으로 대용량 메모리 관리기술, 대용량 데이터 압축기술, 실시간 데이터 처리 기술 등이 요구된다.

소프트웨어 검증기술은 소프트웨어가 요구되는 모든 기능을 수행하고, 오류 없이 동작함을 검증하는 것으로 개발 초기에는 주로 시스템 하드웨어가 없는 상태이므로 시뮬레이션 환경을 개발할 수 있는 기술이 요구되고, 그 후에는 대상 시스템 환경에서 시험을 수행할 수 있도록 소프트웨어 테스트 베드의 개발 및 이를 통한 소프트웨어 검증기술이 필요하다.

소프트웨어 재사용 기술은 일반적인 소프트웨어에도 해당되는 것으로 표준화된 인터페이스 기술, 소프트웨

* 종신회원

어 컴포넌트 기술, 객체지향 설계기술 등을 통하여 가능한 소프트웨어의 많은 부분을 재사용할 수 있도록 설계함으로써 전체적인 개발비용과 일정을 줄이기 위한 기술이다.

2.3 탑재소프트웨어 개발환경

탑재소프트웨어 개발을 위해서는 실시간 운영체제, 컴파일러, 디버거, Build 도구, 소프트웨어 버전관리 도구, 시험 및 검증도구 등이 필요하다. 실시간 운영체제는 자체 개발하여 사용하거나 VRTX, VxWorks 등의 상용 제품을 사용한다. 프로그래밍 언어로는 Ada, C, C++ 언어를 주로 사용하고, 일부 빠른 처리가 요구되거나 하드웨어 관련 부분에서는 어셈블리 언어를 사용한다. 시험과 검증을 위해 소프트웨어 테스트 베드나 시뮬레이터 등을 제작하여 사용하고, 시스템 통합시험을 위해서는 시험지원 장비 및 소프트웨어가 필요하다.

2.4 탑재소프트웨어 개발과정

위성탑재 소프트웨어의 개발은 DOD- STD-2167A/ IEEE12207과 같은 waterfall development model 이나 이를 수정한 개발 프로세스 표준을 주로 따르며 탑재소프트웨어 요구사항 분석 및 사양서 작성, 예비설계, 상세설계, 구현 및 유닛테스트, 통합시험, 검증시험, 운영 및 유지보수의 단계를 거쳐 개발된다. 근래에는 객체지향 소프트웨어 개발 방식이(Object-Oriented Software Development) 도입되고 있는데, HOOD (Hierarchical Object-Oriented Development) 방식은 유럽에서(ESA) 많이 사용되고 있다.

3. 국내 개발 동향

국내에서 최초로 발사된 과학실험 위성인 우리별 1호는 영국의 써리(Surrey) 대학에서 기술전수 및 공동개발 방식을 통하여 탑재소프트웨어 기술을 습득하였으며 이를 보완하여 우리별 2,3호의 소프트웨어 및 과학기술위성 1,2호의 소프트웨어도 자체 개발하였다. 다목적 실용위성인 아리랑 1호 위성의 경우도 미국의 TRW사와 공동개발을 통하여 관련 기술을 이전 받았으며 이를 토대로 아리랑 2호 위성의 탑재소프트웨어를 자체 개발하였다. 이러한 경험을 토대로 향후 우리나라의 저궤도 위성에 요구되는 탑재소프트웨어는 국내연구진의 자체 기술로 개발 가능하게 되었다.

3.1 우리별 위성 탑재소프트웨어

1992년, 1993년에 각각 발사된 우리별 1, 2호는 주 컴퓨터로 80C186을 사용하였고, 탑재소프트웨어는 위성체 데이터 시스템을 정기적으로 관리하는 시스템 운영 처리(Housekeeping), 위성체 각 서브시스템을 통제하는 위성명령 처리, 서브시스템 관리, 탑재체 관리 및 지상국과의 통신 등의 임무를 수행한다.

주 컴퓨터의 ROM에는 최소한의 부트스트랩 로더만이 구현되고, 다른 프로그램들은 위성이 발사된 후에 지상에서 위성으로 전송된다. 실시간 다중 태스크 관리 운영체제를 사용하여 다수의 독립적인 태스크들을 스케줄링하며, 스트림 방식으로 태스크 간 통신을 지원한다. OBC186 운영소프트웨어 구조는 그림 1과 같다. 부트스트랩 로더, 운영체제, 통신 프로그램, 자세제어 프로그램, 원격 검침 및 명령 프로그램, 탑재체 운영 프로그램들로 구성되어 있다. 파일서버, 원격검침 및 명령 서버, AX.25 서버, DASH 서버는 하드웨어 인터페이스를 담당하는 프로그램들이다[1].

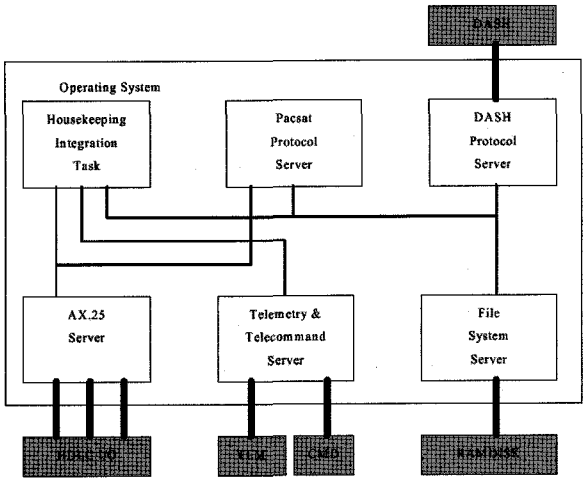

그림 1 OBC186 운영소프트웨어 구조

2호기에서는 부 컴퓨터로 80C186보다 성능이 우수한 인텔사의 80960MC를 사용한 KASCOM(KAIST Satellite Computer)을 개발하였고 운영체제 또한 Nucleus에 바탕을 둔 실시간 선점형 다중운영체제인 별지기(Star Keeper)를 개발하였다. 우리별 3호에서는 KASCOM을 주 컴퓨터로 사용하였고 자세제어로 반작용 휠을 사용한 3축 안정화 방식을 사용하였다. 과학기술위성 1호(우리별 4호)에서는 FPGA(Field Programmable Gate Array)를 이용하여 무게 및 크기의 소형화를 추구하였고, 네트워크 컨트롤러(Network Controller)를 내장함으로서 고속으로 위성 네트워크에 접속할 수 있도록 구현되었다. CPU와 운영체제는 우리별 3호와 동일하다. 과학기술위성 1호의 탑재소프트웨어 구성도는 그림 2와 같다[2].

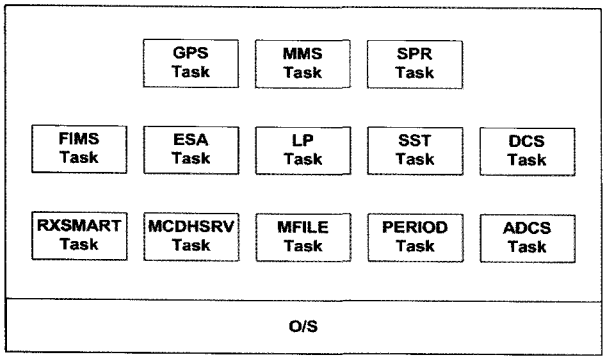

그림 2 과학기술위성 1호 탑재소프트웨어 구성도

GPS(Global Positioning System), MMS(Mass Memory System), SPR(Solar Power Regulator) PERIOD, ADCS(Attitude Determination and Control System), FIMS(Far-ultraviolet IMaging Spectrograph), ESA(Electro-Static Analyzer), LP (Langmuir Probe), SST(Solid State Telescope), DCS(Data Collection System) 등의 태스크들이 운영체제하에 실행된다.

3.2 아리랑 위성 탑재소프트웨어

아리랑 위성 1호기의 탑재컴퓨터는 지상과의 통신 및 탑재체 관리를 수행하는 OBC와 위성의 자세를 제어하는 RDU(Remote Drive Unit), 위성 전력/열 제어를 담당하는 ECU(EPS Control Unit)으로 구성된다. 각 컴퓨터는 프로세서로 80C186 CPU를 사용하고, MIL-STD-1553B 데이터 버스로 연결되어 있다. 그림 3은 위성과 소프트웨어와의 인터페이스를 나타낸 것이다[4].

세 개의 컴퓨터에 분산되어 탑재되는 탑재 소프트웨어는 세 개의 CSCI(Computer Software Configuration Item)로 구성되고, 각 CSCI에는 VRTX 운영체제가 설치되어 실시간 다중처리 환경을 제공한다. 그림 4는 탑재소프트웨어의 각 CSC(Computer Software Component)를 나타낸 것이다. CCI(Command & Communication I/F), SCP(Stored Command Processor), CDS(Command Dispatcher)는 명령처리 기능을 수행하고, DAQ(Data Acquisition), MMD(Mass Memory & Downlink Management)는 원격측정 데이터 처리 기능을 수행한다. EXE(Executives)는 프로세서 초기화 및 스케줄링, UTL(Utilities)은 각종 유틸리티 함수와 1553 통신 서비스 등을 수행한다. ACS (Attitude Control Subsystem)는 자세결정 및 제어 기능, EPS(Electrical Power Subsystem)는 전력 제어기능, TCS(Thermal Control Subsystem)는 열제어 기능을 수행한다.

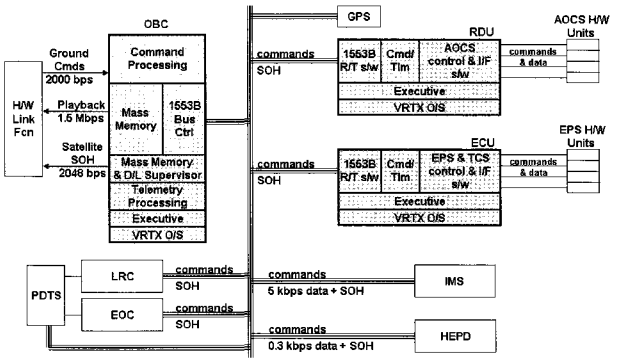

그림 3 아리랑 1호 위성과 소프트웨어 인터페이스

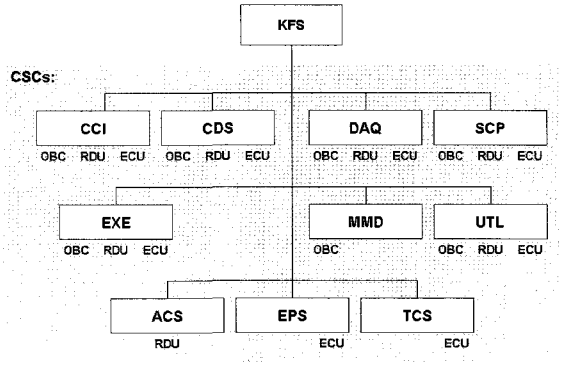

그림 4 아리랑 1호 탑재소프트웨어 CSCs

아리랑 위성 2호기의 경우에는 탑재소프트웨어 구조나 제어방식은 1호기와 유사하다. 단, 탑재컴퓨터가 80C386으로 변경되었고, 고 정밀 자세제어의 기능이 요구됨으로 RDU의 경우, 빠른 속도의 수치연산을 위하여 80387의 코-프로세서가 추가되었다. 그리고 별추적기, 자이로 등의 데이터 처리를 위하여 Local 1553B 버스가 추가되었다.

3.3 무궁화 위성 탑재소프트웨어

무궁화위성 1, 2호에는 탑재컴퓨터 없이 FPGA를 사용하여 하드 코딩되어 있다. 3호의 탑재컴퓨터에는 MIL-STD-1750A CPU와 MIL-STD- 1553B 버스를 사용하였다. 1553B 버스는 모든 위성 서브시스템 명령과 데이터의 송수신 경로가 되며 프로세서는 이를 감시하고 처리한다. 탑재컴퓨터는 2개의 프로세서 보드와 1개의 이중 전원공급기로 구성되어 있다. 그림 5는 탑재소프트웨어 주변 인터페이스를 나타낸 것이다. 1553B를 통한 RWA(Reaction Wheel Assembly), IMU(Inertial Measurement Unit), ESA/SSA (Earth Sensor/Sun Sensor Assembly), Payload RIU(Remote Interface Unit) 등의 인터페이스를 갖고 BCRTM(Bus Control Remote Terminal Monitor)으로부터 1553B 데이터 수신에 대한 인터럽트를 받고, CTU(Command Telemetry Unit)로

부터 Telemetry 전송 인터럽트를 받으며, WDT (Watch Dog Timer)로 MeOK 신호를 주기적으로 출력하도록 되어 있다[3].

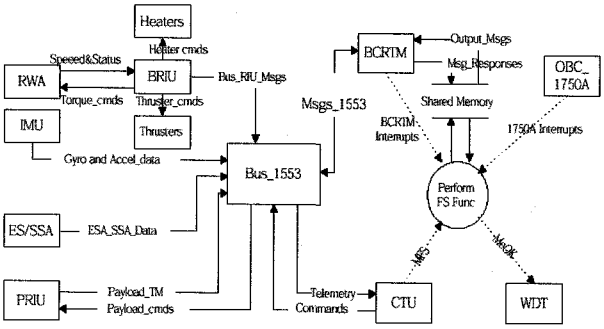

그림 5 무궁화 3호 탑재소프트웨어 인터페이스

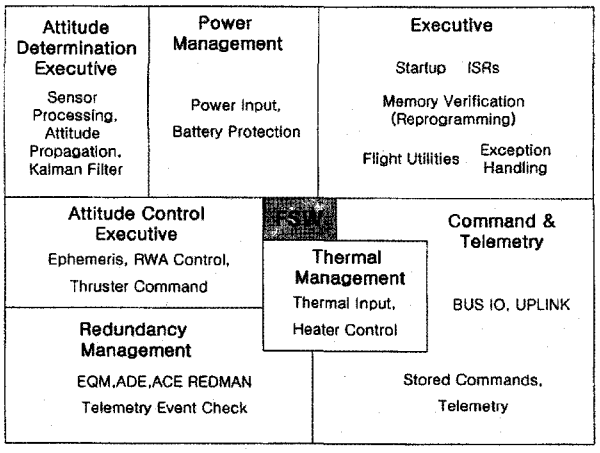

그림 6 무궁화 3호 탑재소프트웨어 구성도

탑재 소프트웨어는 원격명령 처리 및 원격측정 데이터를 전송하는 기능을 수행하는 원격측정 및 명령 소프트웨어, 센서 데이터 처리, 자세전파, 칼만 필터 등의 기능을 수행하는 자세 결정 소프트웨어, 반작용 휠, 추력기 등을 이용하여 자세를 제어하는 기능을 수행하는 자세제어 소프트웨어, 열 제어 소프트웨어, 전력제어 소프트웨어, 이상상태 발생 시 이를 해결하기 위한 이중화 관리 소프트웨어, 전체 소프트웨어를 스케줄링하기 위한 Executive 소프트웨어로 구성되어 있다. Executive는 16Hz로 수행되어 원경측정 2Hz, 원격명령 2Hz, 자세결정 8Hz, 자세제어 2Hz, 저장 명령처리 1Hz, 전력 및 열제어 0.125Hz로 활성화 시킨다[5]. 그림 6은 무궁화 3호의 탑재소프트웨어 구성을 나타낸다.

4. 국외 개발 동향

미국, 유럽 등 위성 개발경험이 많은 해외 위성 선진국의 경우 표준화를 통한 소프트웨어 재사용성을 높여 탑재소프트웨어 개발비용 절감 및 기간을 단축하고, 다양한 임무를 수용할 수 있도록 지능화 및 자동화에 대한 연구를 진행하고 있다. 또한 탑재컴퓨터의 성능향상에 따라 탑재소프트웨어에서 처리할 수 있는 능력이 증가되어 기존에는 지상에서 처리하던 기능들을 점차 위성에서 처리하도록 하는 방향으로 변해가고 있다. 해외의 위성 데이터 시스템 발전과정을 요약하면 그림 7과 같다.

구분	1990 (GRO)	1995 (XTE)	2000 (EO-1/MAP)	2011 (JWST)
프로세서	NSSC01 → 1MHz 64K 18-bit RAM	80386 → 16MHz 1MB RAM 512KB EEPROM	MIPS R3000 → 12MHz 32MB RAM 4MB EEPROM	Power PC 133MHz 50MB RAM 4MB EEPROM
외부 버스	MIL-STD-1553 → 1Mbps	SpaceWire → 100~400Mbps	SpaceLan 10M~1Gbps	
프로토콜	TDM → Fixed Major/ Minor Frame	CCSDS → Packet Telemetry Telecommand	IP/SCPS	
개발언어	Assembler → C → C++ → Automated Code Generation			
소프트웨어	Interrupt Driven → No OS	Interrupt Driven → Real Time OS	Interrupt Driven → RTOS, S/W Bus	Interrupt & Data Driven RTOS, Dynamic Objects

그림 7 위성 데이터 시스템의 발전단계

4.1 탑재소프트웨어 Cost-Risk 분석

위성 프로젝트에서 소프트웨어의 중요성이 점차 증가함에 따라 소프트웨어에 대한 정확한 비용과 위험도(Cost-Risk) 분석이 필요하다. 이에 대한 정확한 분석이 없다면 결국 비용증가, 일정지연, 프로젝트 실패확률 등이 증가하게 된다[7]. 그리고 개발방향도 하드웨어 개발 중심에서 하드웨어/소프트웨어가 통합된 시스템 개발 중심으로 변해가고 있다. 이러한 배경 하에 NASA JPL(Jet Propulsion Laboratory)에서 1995년~1999년 사이에 수행된 8개의 위성 프로젝트를 대상으로 Cost-Risk 증가 원인을 분석하고, 이를 감소시키기 위한 방안을 제시하였다[8]. 표 1은 Cost-Risk 증가 원인을 요약한 것이며 이에 대한 Cost-Risk 감소 방안은 표 2와 같다.

표 1 소프트웨어 Cost-Risk 증가 원인

구분	Cost-Risk 증가 원인
계획	- 초기 계획 단계에서 S/W를 고려하지 않고 S/W팀이 형성되지 않음 - S/W 규격 및 설계가 변경되기 쉬움
요구분석 및 설계	- S/W 설계가 단순히 서브시스템 레벨에서 수행 - S/W 영향에 대한 고려 없는 결정
개발경험 및 팀웍	- 관리자나 시스템 엔지니어가 S/W를 모름 - H/W-S/W간 시스템-S/W간 팀웍이 형성되지 않음
시험	- 불충분한 시험 - 초기에 시험 도구의 부재
소프트웨어 재사용	- 재사용 S/W에 대한 Review 부족 - 불필요한 코드가 많이 포함됨 - COTS를 비용 산정에서 누락

표 2 소프트웨어 Cost-Risk 감소 방안

구분	Cost-Risk 감소 방안
계획	- 계획단계부터 S/W에 대한 고려 - 비상계획을 포함한 위험요소 관리
요구분석 및 설계	- S/W를 시스템 통합 설계에 포함 - 표준화된 S/W Function을 구분
개발경험 및 팀웍	- 관리자나 시스템 엔지니어가 S/W를 잘 이해하고 시스템 레벨에서 관리 - S/W팀은 시스템을 이해해야 함 - H/W-S/W, 시스템-S/W간 팀을 형성
시험	- 개발초기 시험환경(시뮬레이터) 구축 - 다양한 S/W 시험환경 필요 - target과 개발시스템이 일치해야 함 - 사용사례가 있는 설계 도구 이용
소프트웨어 재사용	- 재사용 S/W에 대한 Review

4.2 탑재소프트웨어 기술 로드 맵

NASA GSFC(Goddard Space Flight Center)에서 2013년까지 계획되어 있는 위성 프로젝트를 지원하기 위하여 필요한 탑재소프트웨어 기술에 대한 기술 로드 맵을 작성하였다. 주요 목표는 필요한 기술 개발을 위한 전략과 계획을 세우고 요구 사항을 만족시키면서 고품질, 저비용의 소프트웨어를 개발하는 것이다. 세 가지의 기술 영역으로 구분하였는데 위성 응용기술(Spacecraft Applications Technology), On-Board 데이터 시스템 기술(On-Board Data System Technologies), 탑재소프트웨어 개발기술(Flight Software Development Technologies)로 구분되어 있다[9]. 그림 8은 탑재 소프트웨어 기술 로드 맵을 요약한 것이다.

위성 응용기술에서는 자율성을 향상시키기 위한 기술로 위성에서 이벤트에 따라 임무에 대한 스케줄링을 자동으로 수행하고, 각종 결함 발생 시 이를 자동으로 인식하고 복구할 수 있는 향상된 자율성이 요구된다. 임무 데이터의 처리도 방대한 양의 데이터를 무조건 저장해서 지상으로 전송하는 것이 아니라 위성에서 데이터를 추출하고 필터링을 수행하여 불필요한 데이터를 제거한다.

On-Board 데이터 시스템 기술은 표준화된 인터페이스, 통합된 외부 장치와의 인터페이스, 표준화된 정보교환 프로토콜 등을 통하여 소프트웨어 재사용성을 높이는 것이다. 그리고 계층적 구조(Layered Architecture)로 각각의 소프트웨어를 컴포넌트(Component)화하여 어느 특정 부분의 변경이 전체 시스템에 미치는 영향을 최소화 한다. 동적 프로그램 로딩 기술을 통하여 위성에 탑재되어 있는 소프트웨어 기능에 국한되지 않고, 새로운 기능을 추가하거나, 각종 이벤트에 따라 수행할 태스크를 교체하여 위성 운용 기능을 극대화 할 수 있다. 탑재소프트웨어 개발기술에서는 각 서브시스템 모델에 대하여 재사용 할 수 있는 라이브러리를 개발하

구분			2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015
위성 응용기술	향상된 자동화 기능		▶ Advanced Fault Detection, Determination, Isolation & Recovery ▶ Goal-Driven Operation, Autonomous Navigation ▶ Event-based Science Planning ▶ Adaptive Scheduler ▶ Trend Analysis ▶ Multi-S/C Collaboration
	임무데이터 처리		▶ Science Feature Extraction ▶ Discrimination & Selection of Science Data ▶ Science Data Fusion/Compression ▶ Constellation Data Sharing, Processing, Analysis
On-Board 데이터 시스템 기술	표준 인터페이스		▶ Middleware Trade Study ▶ Middleware Integration
	On-Board 네트웍		◀ 1553, Prototype Ethernet ▶ Spacewire ▶ Inter S/C Communication
	프로토콜		◀ UDP, TCP ◀ IP/EthernetIP/Spacewire-IP/1553 ▶NTP ▶ SNMP ▶ SOAP ▶ Proximity-1
	동적 로딩		◀ VxWorks, RTEMS
	운영체제		◀ VxWorks, RTEMS ◀ RTLinux Flt Qual
탑재소프트웨어 기술	재사용 라이브러리		▶ Generic Requirements, Repeatable Tests, Standards ◀ ACS Models, Informal re-use of C&DH Application ▶ Generic reusable C&DH Application
	개발도구	UML 모델링	▶ C&DH Models, Method & Tool, CM & Req. Subsystem Models ▶Test Script ▶Performance
		자동화 도구	▶ Autocode Gen., Common Req. Mgt, CM, Generic Performance Analysis ▶ FSW Test Tool ▶ Flight/Ground Database Tools
		통합시뮬레이션 환경	▶ Easily Configurable Simulation Systems(H/W I/Fs and Software) ▶ Resuable, Configurable FSW Test Systems Architecture

▶: 해당기술 개발 시작시점 ◀: 해당기술 기보유 또는 개발 종요시점

그림 8 탑재 소프트웨어 기술 로드 맵

고, 개발도구로 객체지향 개발도구인 UML(Unified Modeling Language) 모델링 기술을 활용하고, 코드와 테스트 스크립트(Test Scripts) 자동 생성기 등의 자동화 도구를 개발하는 것이다. 그리고 모든 탑재 소프트웨어 시험에 활용할 수 있도록 시뮬레이터와 탑재 소프트웨어 사이에 표준화된 인터페이스를 사용한 통합 시뮬레이션 환경 및 테스트 베드를 개발하여 활용하는 것이다.

5. 결 론

본 논문에서는 탑재소프트웨어의 기능과 개발기술, 국내 개발동향 및 국외 개발동향에 대해서 기술하였다. 국내의 경우는 초기단계에 해외 공동개발의 형태로 위성을 개발하면서 탑재소프트웨어 기술을 습득할 수 있었고, 이를 기반으로 현재는 위성의 임무와 요구사항에 맞도록 개발할 수 있는 능력을 어느 정도 갖추었다고 볼 수 있다. 해외 위성 선진국의 경우, 위성의 임무가 복잡 다양화 되고 탑재컴퓨터의 발전에 따라 종래에 지상에서 처리하던 것을 위성에서 처리하는 방향으로 변화되고 있으며, 위성의 자율성이 강화되어 가능한 한 지상의 제어 없이 자동적으로 처해진 상황에 대응하여 정해진 임무를 수행할 수 있도록 하는 추세이다. 또한 그 동안의 다양한 개발 경험을 바탕으로 개발일정과 비용을 절감할 수 있는 방안을 찾고, 각종 인터페이스 표준화를 통하여 재사용성을 높이는 연구가 진행되고 있다. 이제 국내에서도 과학위성, 아리랑 3호, 5호 및 통신방송해양기상 위성 등, 위성 프로그램이 많아지고 다양화됨에 따라 개별적인 위성의 임무와 요구사항을 만족하는데 그치지 않고 여러 위성에 공통적으로 적용할 수 있는 분야에 대한 표준화 방향을 모색하여 재사용을 통한 비용절감과 개발기간 단축을 도모하여야 할 것이다. 또한 정형기법 등을 적용한 신뢰성 향상, 지상의 제어 없이 장기간 자율적으로 임무를 수행할 수 있는 지능화, 다수의 위성들로 구성된 위성군(satellite cluster)의 운영 등에 대한 연구도 필요할 것으로 보인다.

참고문헌

[1] 김형신외, "우리별 1,2호 주 컴퓨터부," 한국우주과학회지 13권 2호, 1996.12.

[2] 연구보고서 "저궤도 과학 실험용 과학기술위성 1호 본체 및 지상국개발," 한국과학기술원, 2003.12.

[3] 연구보고서 "정지궤도 통신위성 핵심 서브시스템 및 운용시스템 개발," 한국항공우주연구원, 1998.

[4] 이종인외, "위성 탑재 소프트웨어 기술 동향," 제어자동화시스템공학회지, 제5권 제6호, 1999.

[5] 김주년외 "우주비행체의 실시간 시스템 및 소프트웨어 기술동향," 제어자동화시스템공학회지 제6권 제5 호, 2000.

[6] 채동석외, "위성탑재소프트웨어 개발동향," 항공우주기술산업동향 3권 1호, 2005.

[7] Nancy G. Leveson, "The Role of S/W in Spacecraft Accidents", AIAA Journal of Spacecraft and Rockets, 41, 4, 2004.

[8] Jairus M. Hihn, "Reducing Flight S/W Development Cost Risk: Analysis and Recommendations", JPL California Institute of Technology, 200.0

[9] Jain Marquart, "Flight Software Technology Roadmap", NASA GSFC, 2003.

이 종 인

1983. 2 서울대학교 계산통계학과(학사)
1985. 2 KAIST 전산학과(석사)
1985. 3~1990.10 삼성전자 정보통신 연구소
1990.11~1994.2 KAIST 인공위성연구센터
1994.3~현재 한국항공우주연구원 위성S/W 팀장
1999년~현재 KAIST 전산학과 박사과정
관심분야 : 실시간 내장형 시스템, 고신뢰 소프트웨어, 최장수행시간 분석
E-mail : jilee@kari.re.kr

차 성 덕

1983 B.S. University of California, Irvine
1986 M.S. University of California, Irvine
1991 UC Irvine Information and Computer Science(박사)
1990~1991 Hughes Aircraft Company, Ground Systems Group, Fullerton, CA
1991~1994 The Aerospace Corporation, El Segundo, CA
1994~현재 한국과학기술원 전자전산학과 교수
E-mail : cha@salmosa.kaist.ac.kr