Software Development System for On-board Computers of Small Satellites

Tokyo University of Science
Sotaro KOBAYASHI and Shinich KIMURA
Objective and Background (1)

- The number of applications of small satellites has been increasing extensively
- Complexity of on-board software increases
- Meanwhile high reliability is required
- Cost and load should be kept in low

- High productivity of on-board software
Objective and background (2)

- Build standard software development environment
  - Make basic part and provide as “development environment”
  - Improve reliability when used in various situations
Objective and background (3)

- Utilize software skeleton
  - Wrap up the difference in hardware structures
  - Enhance recursive utilization of software

![Diagram of software levels with Hodoyoshi SDK highlighted]
Objective and background (4)

- Modularity and class structure of on-board software
- Advanced research with FIRST Program small satellite on-board software as model case
Division of functions

- Divide a completion of work into individual function

- **Equipment driver**
  - Interface initialize
  - Periodic TX (command)
  - Periodic RX (status)
  - Non-periodic TX RX

- **Initialization of the interface**
- **Periodic transmission (TX) of data requesting messages**
- **Periodic monitoring and receiving (RX) of status information**
- **Nonperiodic transmission of command messages and receiving of them**
Concept of modularity and class structure (2)
Concept of modularity and class structure (3)

- Create Super Class
  - Enable basic structure reusable

### DRIVER_SUPER

**Interface initialize**

- Com. output
- St. input
- St. checking
- St. output

**Internal modules**

- Periodic TX (command)
- Periodic RX (status)
- Non-periodic TX RX
Concept of modularity and class structure (A)

DRIVER_SUPER

Interface initialize
Periodic TX (command)
Periodic RX (status)
Non-periodic TX RX

Com. output
St. input
St. checking
St. output

Equipment driver A

DRIVER_SUPER

Interface initialize
Periodic TX (command)
Periodic RX (status)
Non-periodic TX RX

Com. processing for A
St. processing for A

Equipment driver B

DRIVER_SUPER

Interface initialize
Periodic TX (command)
Periodic RX (status)
Non-periodic TX RX

Com. processing for A
St. processing for A

Equipment driver C

DRIVER_SUPER

Interface initialize
Periodic TX (command)
Periodic RX (status)
Non-periodic TX RX

Com. processing for A
St. processing for A
Hardware Structure

- Computer is consist of CPU board and I/F board
- I/F board: SpaceWire, CMOS I/O, RS422, SPI, ActiveAnalog, PassiveAnalog • • •

GPS receiver
Sun sensor
Gyro sensor
Geomagnetic sensor
Star sensor
Reaction wheel
Magnetic torque
• • •
Implement (1)

- Make “[equipment name]_STRUCT” struct and manage
  - Prepare the below members and TX/RX data
Implement (2)

- User only need to describe small part for each sensor and add one line of the super class.
Verification with equipment

- The equipment interface is confirmed in table-satellites.
Summary

- We developed “Hodoyoshi SDK” that wraps up hardware and platform dependent interface into standard interface structure.
- The SDK is effective for recursive utilization of software, that provides us with increment of reliability and reduction of development load.
- Based on class structuration, program size is reduced as $\frac{1}{2} \sim \frac{3}{4}$ compared with non classified case.
Future Work

- Verification for launch and on-board demonstration of the system
- Expand the SDK to support various hardware platform and operating system
Thank you for your attention
Software Development System for On-board Computers of Small Satellites

Tokyo University of Science
Sotaro KOBAYASHI and Shinich KIMURA
Objective and Background (1)

- The number of applications of small satellites has been increasing extensively.
- Complexity of on-board software increases.
- Meanwhile high reliability is required.
- Cost and load should be kept in low.

- High productivity of on-board software.
Objective and background (2)

- Build standard software development environment
  - Make basic part and provide as “development environment”
  - Improve reliability when used in various situation
Objective and background (3)

- Utilize software skeleton
  - Wrap up the difference in hardware structures
  - Enhance recursive utilization of software

Hodoyoshi SDK
Objective and background (4)

- Modularity and class structure of on-board software
- Advanced research with FIRST Program small satellite on-board software as model case
Concept of modularity and class structure (1)

- Division of functions
  - Divide a completion of work into individual function

![Diagram showing equipment driver and interface functions]

- Interface initialize: Initialization of the interface
- Periodic TX (command): Periodic transmission (TX) of data requesting messages
- Periodic RX (status): Periodic monitoring and receiving (RX) of status information
- Non-periodic TX RX: Nonperiodic transmission of command messages and receiving of them
Concept of modularity and class structure (2)
Concept of modularity and class structure (3)

- Create Super Class
  - Enable basic structure reusable

**DRIVER_SUPER**

- Interface initialize
- Periodic TX (command)
- Periodic RX (status)
- Non-periodic TX RX

- Com. output
- St. input
- St. checking
- St. output
Concept of modularity and class structure (4)

DRIVER_SUPER

- Interface initialize
- Periodic TX (command)
- Periodic RX (status)
- Non-periodic TX RX

Internal modules

Inheritance

Equipment driver A

- DRIVER_SUPER
- Interface initialize
- Periodic TX (command)
  - Com. processing for A
- Periodic RX (status)
  - St. processing for A
- Non-periodic TX RX
  - Com. processing for A
  - St. processing for A

Inheritance

Equipment driver B

- DRIVER_SUPER
- Interface initialize
- Periodic TX (command)
  - Com. processing for B
- Periodic RX (status)
  - St. processing for B
- Non-periodic TX RX
  - Com. processing for B
  - St. processing for B

Inheritance

Equipment driver C

- DRIVER_SUPER
- Interface initialize
- Periodic TX (command)
  - Com. processing for C
- Periodic RX (status)
  - St. processing for C
- Non-periodic TX RX
  - Com. processing for C
  - St. processing for C
Hardware Structure

- Computer is consist of CPU board and I/F board
- I/F board: SpaceWire, CMOS I/O, RS422, SPI, ActiveAnalog, PassiveAnalog • • •

GPS receiver
Sun sensor
Gyro sensor
Geomagnetic sensor
Star sensor
Reaction wheel
Magnetic torque
• • •
Implement (1)

- Make “[equipment name]_STRUCT” struct and manage
  - Prepare the below members and TX/RX data
Implement (2)

- User only need to describe small part for each sensor and add one line of the super class.
Verification with equipment

- The equipment interface is confirmed in table-satellites.
Summary

- We developed “Hodoyoshi SDK” that wraps up hardware and platform dependent interface into standard interface structure.

- The SDK is effective for recursive utilization of software, that provides us with increment of reliability and reduction of development load.

- Based on class structuration, program size is reduced as $1/2 \sim 3/4$ compared with non classified case.
Future Work

- Verification for launch and on-board demonstration of the system
- Expand the SDK to support various hardware platform and operating system