ABSTRACT

This paper covers the development of multi-satellite image data receiving software, which is a part of a remote sensing ground station. We applied modern software engineering techniques, object-oriented methodology, to the development process in order to achieve high portability and reliability of the software.

1. SYSTEM CONTEXT AND CLASS DEFINITION

Major functions of the satellite image data receiving system are

- Control of image data receiving process schedules
- Real time data receiving and archiving
- Real time data processing and MWD (Moving Window Display)

We begin our analysis by considering the hardware which our software must execute. Figure 1 provides a process diagram which illustrates the hardware platform. Firstly, we defined hardware configuration clearly so that we could focus better on the system software. In the object-oriented system the analysis trends to focus on the vocabulary of its problem space.[1] Therefore, we selected key entities in the hardware configuration: DRC (Data Receiving Card), GPS time code receiving card and RAID as a storage device. These are the candidates of class to be defined.

In the software all data processings are synchronized with the system time of the host computer. We used a GPS time code receiving card for synchronizing the system time with the GPS time. We, therefore, defined the **TimeCode** class for handling this func-
tion. The responsibilities of this class are synchronizing the system time with the GPS
time and translating raw system time to some meaningful values.
For developing a highly portable system, the hardware architecture must be very simple.
We developed the DRC which supports the interrupt I/O and the DMA(Direct Memory
Access) data transfer. We, therefore, defined the DRC class of which the responsibility
is transferring incoming data to the main memory of the host computer.
We selected the RAID(Redundant Array of Independant Disks) system for archiving
fast and bulky incomming image data. We, therefore, defined the Storage class. It’s
responsibilities are archiving input data to the RAID system and securing the free space.
The SPOT-1,2 and KOMPSAT-1 were chosen for the target satellites of the software. In
order to handle the satellite-specific data processing parts, we defined the SPOTFor-
matter class and KOMPSAT1Formatter class. Figure 2 illustrates SPOT image data
formatting process. Whenever a target satellite is added we only have to define an addi-
tional satellite-specific formatter class. Finally, we defined the MWD class for
performing real time moving window display.

| Detect Frame Sync. | De-scrambling | Seperate Even Odd Bits | Image data extraction | Subsampling DPCM or Linear |

Figure 2 Formatting process of SPOT image data

2. SOFTWARE OPERATION SCENARIO AND USE CASES

The pass scheduling scenario is described below.
- A system operator inserts pass schedules containing information such as the
  satellite name, sensor name, data format, operator ID, the pass start time and the
  pass end time.
- At the pass start time, the software activates the receiving and MWD functions
  automatically. It stores the received image data to the RAID system and dis-
  plays the real-time scroll image.
- At the pass end time, it de-activates the functions automatically, and generates
  the pass schedule report and log files.

MWD playback is the other use case. The procedure of this use case is as follows.
- The system operator selects a specific pass data file and inserts some basic in-
  formation such as the satellite name, data format and input medium to the
  software for image formatting.
- Then, he/she presses the start playback button to activate the MWD playback.
- On the way the MWD playback is performed, the software supports the opera-
  tor-control functions such as pause and restart.
These scenarios illuminate the major use case of the software. We next consider the various other functions, which are related to the operators’ interaction with the software. Figure 3 shows a prototype GUI (Graphic User Interface) of the pass scheduling module of the software. It is very important to design and implement the prototype GUI in order to mitigate the risks involved in user-interface design.

Figure 3 Prototype GUI of Pass Scheduling Module

Inserting pass schedule is one of the major use cases of the software as described previously. When the pass data is processed and archived successfully the results of the pass schedule process are stored to the database and they are displayed at the pass schedule field of the GUI (Graphic User Interface). According to the major use cases of the software, we defined the PassScheduler and MWDPlayback classes. The responsibilities of the PassScheduler class are to insert, modify and delete pass schedules.

4. ARCHITECTURE FRAMEWORK

Since the image data downlink rate is stable the interrupt generated from a FIFO in DRC occurs at a regular interval. The receiving and MWD processing must be synchronized with the FIFO half full interrupt as illustrated in Figure 4.

Figure 4 Synchronization of FIFO half full interrupt

Figure 5 shows the states diagram of this software. Firstly, the I/O handler thread is initialized and it waits for a local interrupt from DRC. If a local interrupt occurs, it transfers the data in DRC FIFO to the main memory of the host computer using DMA. Then, it archives the input data to the RAID system. It checks the state of the MWDHandler thread. If the thread is not running, the I/O handler thread creates a new MWDHandler thread for real-time processing and MWD. And the I/O handler thread goes to the wait-state for the next FIFO half full interrupt.
Figure 6 shows a class diagram which expresses this architecture for the satellite image receiving system. Here, we can find most of the classes we defined earlier during the analysis. The main difference is, however, that we now show how all the key abstractions collaborate with one another. We defined a new class in this architecture, named as **Formatter**, whose responsibility is to serve as the collection of the each satellite’s formatter class.

5. CONCLUSIONS

The complexity of the satellite image receiving system requires a systematical development approach. We used a software-based and object-oriented method in order to obtain the portability and reliability of the system.
6. REFERENCES