Software Evolution from TET-1 to Eu:CROPIS

Olaf Maibaum (1), Ansgar Heidecker (2)

(1) German Aerospace Center (DLR), Simulation and Software Technology, Lilienthalplatz 7, 38108 Braunschweig, Germany

(2) German Aerospace Center (DLR), Institute of Space Systems, Robert Hooke Str. 7, 28359 Bremen, Germany

ABSTRACT

The base of the Eu:CROPIS (Euglena Combined Regenerative Organic food Production In Space) Attitude and Orbit Control System (AOCS) is the three layer AOCS software architecture of the TET-1 satellite (Technology demonstrator). Because of different AOCS requirements between TET-1 and Eu:CROPIS, a software reuse is only possible for software components in the interface layer. In the other two architecture layers, the software components have to be replaced by new implementations to fulfil the changed requirements of the Eu:CROPIS mission. In contrast to the former software evolution from BIRD (Bispectral Infra-Red Detection) to the TET-1 AOCS, the software evolution is forced in Eu:CROPIS by the reuse of software design principals applied in TET-1. Without software reuse we are able to change the underlying scheduling mechanisms from a fixed time approach to a more reactive software system presented in this paper.

1. INTRODUCTION

The reactive scheduling mechanism used in the Eu:CROPIS AOCS is a result from experiences from the BIRD mission [7]. This mechanism, named “tasking framework”, resolves one weakness in the BIRD and TET-1 AOCS software [8]: the scant timing for the control torque computation. The tasking framework is the core element in the operation system development of DLR’s OBC-NG (Onboard Computer – Next Generation) project, which will provide a distributed onboard computer platform for reconfigurable and high redundant systems. At the moment the tasking framework is implemented on top of Linux and is use in DLR’s ATON (Autonomous Terrain-based Optical Navigation) project and in the MAIUS (Atom-optical experiments on sounding rockets) mission. Eu:CROPIS uses a porting of the tasking framework from Linux to the ROBOSS operating system API on top of RTEMS.

The next chapter sketches the Eu:CROPIS mission, the used satellite bus, and the AOCS. Chapter 3 presents the tasking framework and implementation details for the AOCS. The focus is set on the usage of the reactive behavior for the implementation of the diagnostic report service defined by the packet utilization standard [6], which is one of the main benefits of the tasking framework. The paper closes with a conclusion.

2. EU:CROPIS

The mission Eu:CROPIS is the demonstration of the feasibility of restartable and sustainable life support systems. Such systems enable the production of food and atmosphere, and the utilization of waste like urine and phosphate [1]. Furthermore, the system should be reliable enough for long duration missions. The biological experiments require different levels of gravity. This is achieved by a spin stabilized satellite which can

change its rotation speed and thereby the gravity in the biological experiment compartments during mission time. The target gravities of Eu:CROPIS are 0.16 g (Moon) and 0.38 g (Mars) respectively in the biological experiment compartments. They are placed at a reference radius of 0.35 m measured from the designed spin axis. The launch is planned in 2017 with a Falcon 9 as a piggy back start.

The used satellite bus is based on the DLR compact satellite bus program, which is a research and development platform in a component-oriented design. The bus for the Eu:CROPIS mission is a spin stabilized platform with a cylindrical body with a diameter of 1000 mm and a height of 1100 mm [3]. The bus has two sections: one with the separated payload compartments at the upper deck, and one with the bus section. The mass of the satellite is around 230 kg. The orbit is sun synchronous with at least 600 km altitude.

The main requirement to be fulfilled by the AOCS is the generation of gravity in the biological compartments. Beside this, the AOCS is responsible to orient the z-axis into sun direction to ensure power generation by the solar panels. Thereby, the satellite bus is spinning around the z-axis between 5 to 31 rpm. To satisfy the power generation with the solar panels the spin axis has to be reoriented by ~1 deg/day to keep a sun pointing attitude

As actuators, the AOCS uses three magnetic torquers to control the rotation and spin axis. It uses two magnetometers, 10 sun sensors, and four angular rate sensors with a tetrahedral mounting. In addition, two GPS receivers provide navigation information.

The attitude controller uses five control state modes. The detumbling mode damps the rates of the satellite body sufficiently. It is the first mode when the AOCS boots up. The second mode is the spin up/down mode to change the spin rate around the z-axis. The spin mode holds a spin rate and orients the solar panels into the sun. For the solar panel deployment, the AOCS provide a deployment mode to handle the moments of inertia change. To indicate a problem in the AOCS, the fifth mode is the AOCS safe mode keeping the solar panels into sun direction.

The attitude controller uses an Unscented Kalman Filter (UKF) approach as core of the attitude determination which is designed for spin stabilized satellite. For a detailed description of the used UKF and the AOCS see [4].

3. TASKING FRAMEWORK

The Eu:CROPIS AOCS uses a reactive scheduling mechanism to control the order and timing of computation tasks, named as tasking framework. Starting point for the implementation is the way how estimator and predictor modules were organized in the BIRD AOCS. These modules are executed in a fixed order at a fixed time in the control cycle to combine all sensor inputs to an accurate attitude state vector. For TET-1, this static approach has been led to a scant timing between controller computation and the commanding of the control torque actuators. During the launch and early orbit phase (LEOP), a further timing violation of another bus application occurred which led to an unexpected AOCS state. Figure 1 A) depict an example of such a timing.

In the new tasking framework, the timing behavior has been changed. Instead starting the computation at a predefined time in the computation cycle, it starts now whenever the information is available. All information values are stored in messages distributed by channels. The channels initiate the computation when all defined conditions are met. The timing with the tasking framework is depicted in Figure 1 B). In addition, the computed values of the estimator and predictor modules are not stored in one AOCS state vector anymore but handled as messages on channels inside the AOCS software. These channels provide the synchronization mechanism for the data. This can be implemented as single or double buffers, or as more complex data structures like FIFO queues. The synchronization is triggered by the scheduler and a set of three methods which should be overloaded by the synchronization mechanism.

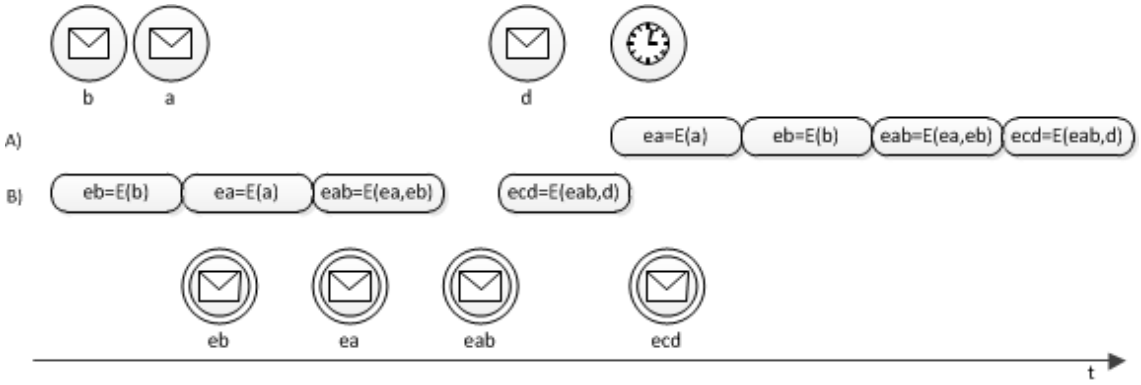

Figure 1 Timing: A) Procedural B) Tasking

In the tasking framework, all computations are performed by tasks instead of threads. A task can subscribe to suitable channels needed to provide the information for the computation. Each task is started when a specified amount of information is available on all subscribed channels. It can also start immediately when a corresponding channel is subscribed as final input. These specifications are configured by a task input, which establishes the subscription between channel and task. In contrast, a thread is only started once during the run time of the onboard software and should be implemented as an endless computation loop for the specified time frame.

For computations requiring predefined timings in the computation cycle, the tasking framework provides a task event, which is triggered relatively to the reset operation or periodically by the onboard clock. This task event can be subscribed by a task instead of a channel in order to operate with a defined timing. A time-out is reached when a task event subscribed as final input.

One computation can split into several tasks but managed as one group. This allows the parallelization of a computation on multi-core processor platforms. The behavior of tasks inside such a task group is slightly different to the behavior of an isolated task. By default, a task and all inputs of a task are reset by the scheduler when a task finishes the computation. The task can then be activated again as soon as the specified amount of information is available on the subscribed channel. For a task group, this reset is only executed when all tasks in the task group have been executed. Thus, a task inside a task group can only start again when all other tasks inside the group have been executed at least once in the previous loop. Such groups are used for example in the Eu:CROPIS

AOCS for all tasks of the estimator, predictor and controller block or for the hand shaking communication protocol with actuators.

The reactive organization of the AOCS allows an immediate filtering for the diagnostic reporting service of the packet utilization standard (PUS). For this purpose, a filter task exists in the software which can dynamical be associated to a subset of channels in the AOCS. A new data item is filtered immediately when it is pushed to the channel. The subset of channels provides for the message data scalar values which can easily be filtered. These values could be e.g. a rotation speed, a binary value, or the angle between rotation axis and the sun vector. Beside the relative filter defined by PUS, the Eu:CROPIS AOCS provides also filter for maximum, minimum, and epsilon values.

4. CONCLUSIONS AND OUTLOOK

This paper present the reactive scheduling mechanism used in the Eu:CROPIS AOCS. This kind of scheduling removes the overestimated time gaps between processing steps in the BIRD and TET-1 software. For TET-1, the scant timing provoked a malfunction of the AOCS during the LEOP, caused by a timing violation in another bus application. For Eu:CROPIS, such kind of malfunctions with respect to scheduling and the used synchronization between computation steps in the controller are no longer possible.

The described filter concept used by the implementation of the diagnostic reporting service shows how observations can be integrated into a computation process. Besides for the normal operation of the AOCS, such observations can also be used for FDIR (Failure detection and recovery) and surveillance checks. Furthermore, the split of processing chains in several computation tasks allows the parallelization of onboard computing which yield more computation power. This is addressed in DLR's OBC-NG project. The Eu:CROPIS AOCS demonstrates already today the potential power of distributed and reconfigurable onboard systems expected to become available in the next years also for the space domain.

5. REFERENCES

[1] G. Bornemann, K. Waßer, R. Hemmersbach, R. Gerzer, R. Anken, J. Hauslage. C.R.O.P.-Combinded Regenerative Organic Food Production: waste and wastewater processing by trickling filters. In: EHBLSS 2013 Proceeding & Abstracts Book. Beijing, China (2013).

[2] F. Dannemann, F. Greif, Software Platform of the DLR Compact Satellite Series. In: Proceedings of the 4S Symposium, Mallorca, Spain (2014).

[3] O. Mierheim, T. Glaser, C. Hühne, H. Müller, E. Kheiri, Modal Frequency Adjustment of the Eu:CROPIS Satellite Structure. In: Proceedings of the 13th European Conference on Space Structure, Material & Environmental Testing. Braunschweig, Germany (2014).

[4] A. Heidecker, T. Kato, O. Maibaum, M. Hölzel, Attitude Control System of the Eu:CROPIS Mission. IAC 2014, Toronto, Canada (2014).

[5] D. Lüdtke, K. Westerdorff, et. al. In: Proceedings IEEE Aerospace Conference 2014: OBC-NG: Towards a Reconfigurable On-board Computing Architecture for Spacecraft. Big Sky, USA (2014).

[6] ECSS-E-70-41A: Packet Utilization Standard. (2003)

[7] K. Brieß, W. Bärwald, T. Gerlich, H. Jahn,F. Lura, H. Studemund. The DLR Small Satellite Mission BIRD. In: Digest of the 2nd International Symposium of the International Academy of Astronautics. pp 45-48. Berlin, Germany (1999)

[8] O. Maibaum, T. Terzibaschian, C. Raschke, and A. Gerndt. Software Reuse of the BIRD ACS for the TET Satellite Bus. In: Digest of the 8th International Symposium of the International Academy of Astronautics. pp. 409-412. Wissenschaft und Technik Verlag, Berlin (2011)