Acta Astronautica 69 (2011) 583–594

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

An integrated development framework for rapid development of platform-independent and reusable satellite on-board software[☆]

Claas Ziemke[a,*], Toshinori Kuwahara[b,1], Ivan Kossev[c]

[a] *Institute of Space Systems, University of Stuttgart, Pfaffenwaldring 31, 70569 Stuttgart, Germany*
[b] *Graduate School of Engineering, Department of Aerospace Engineering, Tohoku University, 980-8579 Aoba, 6-6-01 Sendai, Japan*
[c] *Private, Stuttgart, Germany*

ARTICLE INFO

Article history:
Received 3 February 2011
Received in revised form
9 April 2011
Accepted 11 April 2011
Available online 20 May 2011

Keywords:
On-board software
Reusable
Platform-independent
System simulation
Open-source

ABSTRACT

Even in the field of small satellites, the on-board data handling subsystem has become complex and powerful. With the introduction of powerful CPUs and the availability of considerable amounts of memory on-board a small satellite it has become possible to utilize the flexibility and power of contemporary platform-independent real-time operating systems. Especially the non-commercial sector such like university institutes and community projects such as AMSAT or SSETI are characterized by the inherent lack of financial as well as manpower resources. The opportunity to utilize such real-time operating systems will contribute significantly to achieve a successful mission. Nevertheless the on-board software of a satellite is much more than just an operating system. It has to fulfill a multitude of functional requirements such as: Telecommand interpretation and execution, execution of control loops, generation of telemetry data and frames, failure detection isolation and recovery, the communication with peripherals and so on. Most of the aforementioned tasks are of generic nature and have to be conducted on any satellite with only minor modifications. A general set of functional requirements as well as a protocol for communication is defined in the SA ECSS-E-70-41A standard “Telemetry and telecommand packet utilization”. This standard not only defines the communication protocol of the satellite–ground link but also defines a set of so called services which have to be available on-board of every compliant satellite and which are of generic nature. In this paper, a platform-independent and reusable framework is described which is implementing not only the ECSS-E-70-41A standard but also functionalities for interprocess communication, scheduling and a multitude of tasks commonly performed on-board of a satellite. By making use of the capabilities of the high-level programming language C/C++, the powerful open source library BOOST, the real-time operating system RTEMS and finally by providing generic functionalities compliant to the ECSS-E-70-41A standard the proposed framework can provide a great boost in productivity. Together with open source tools such like the GNU tool-chain, Eclipse SDK, the simulation framework OpenSimKit, the emulator QEMU, the proposed on-board software framework forms an integrated development framework. It is possible to design, code and build the on-board software together with the operating system and then run it on a simulated satellite for performance analysis and debugging purposes. This makes it possible to rapidly develop and deploy a full-fledged satellite on-board software with minimal cost and in a limited time frame.

[☆] This paper was presented during the 61st IAC in Prague.
[*] Corresponding author. Tel.: +49 711 685 62326; fax: +49 711 685 63596.
E-mail addresses: ziemke@irs.uni-stuttgart.de (C. Ziemke), kuwahara@astro.mech.tohoku.ac.jp (T. Kuwahara), ikossev@gmail.com (I. Kossev).
[1] Tel.: +22 795 6990; fax: +22 795 6991.

0094-5765/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.actaastro.2011.04.011

1. Introduction

Small satellites can be categorized in respect to the satellite ordering customer into four different groups. The first group is constituted by the agency ordered satellites which are mainly scientific missions. The second group is constituted by the commercial and industry satellites. This group of satellites is characterized by the goal to create revenue through delivering a product to customers. The third group is constituted by the military satellites which are ordered and operated by military organizations. The last group of small satellites is formed by the university and amateur satellites [1]. These amateur and university small satellites are usually designed following different design principles compared with small satellites of the groups one to three. Those design principles are mostly "Keep it simple, Stupid!" (KISS) and mean that any unneeded complexity should be avoided in order to make the satellite more reliable.

In addition to small budgets, university and amateur satellite projects also are characterized by limited human resources. This lack of human resources, especially in the field of software engineering, the design principles and the usage of non-standard communication protocols often lead to non-optimal operation scenarios.

All of the aforementioned groups have in common that also small satellites follow Moore's law [2]. The computational power of the on-board data-handling subsystems as well as the data rates and storage capacity have increased significantly in the last years. This trend in data rates and amount is shown in Fig. 1: Moore's law for small satellites. Especially when using commercial of the shelf parts, the performance is significantly increased. The rising complexity of the hardware can only be dealt with a more complex software system [3].

In order to deal with rising complexity of hardware and software as well as with the limited human resources, an on-board software framework currently is being developed by the author in the course of a university small satellite program. The main design goals of this on-board software framework are re-usability, platform independence, interoperability and extendability. The usage of a framework for the development of a satellite on-board software allows the software engineers to focus on essential features such as control algorithms and mission specific functionalities. It also allows the reuse of generic parts of the on-board software. The usage of standardized communication protocols makes possible the usage of agency ground stations and fosters interoperability.

In addition to the on-board software framework itself, an integrated development flow is currently under development in order to test and verify the proposed on-board software framework. This development flow consists of the following steps: First the on-board software framework runs on Xenomai real-time Linux and can be tested extensively using standard software development practice such as unit testing, code coverage analysis, memory profiling and so on. [4]. After the basic functionality is tested, the on-board software runs inside a simulated on-board computer connected to a simulated satellite. The simulated parts of this environment are then gradually replaced with real hardware, starting with the on-board computer. The on-board software then runs on a development board or a breadboard-model of the on-board computer which again is connected to a simulated satellite. Finally, the on-board software runs on the on-board computer connected to the real satellite hardware. This development flow is called system simulation and is state-of-the-art in industrial satellite projects [5].

The goal of the integrated development framework described in this paper is to allow university and amateur satellite projects to use system simulation in the development process together with the proposed on-board software framework. This can lead to a great boost in productivity because the developers can concentrate on the architectural design of the on-board software and neither have to implement infrastructural features such as schedulers and interprocess communication methods nor testing infrastructures such as simulators. This again can, together with the usage of standardized communication protocols, lead to superior operational scenarios compared to conventional approaches.

2. Simulation framework

For embedded software and especially safety critical software profound testing and verification is essential. Traditionally the testing of the software is done by so called

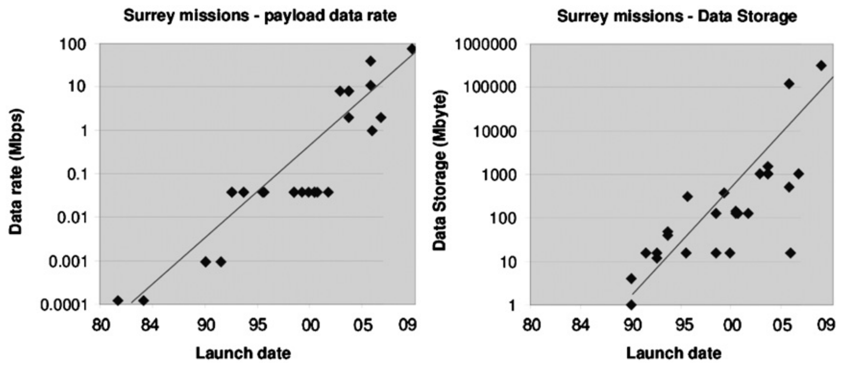

Fig. 1. Moore's law for small satellites [2].