COTS Real-Time Operating Systems in Space

Ljerka Beus-Dukic

School of Computing and Mathematics, University of Northumbria at Newcastle, Ellison Building, Newcastle upon Tyne, NE1 8ST; Tel: 0191 2273974, ljerka.beus@unn.ac.uk

1. Introduction

Space domain applications no longer depend solely on specially developed hardware and software. The specific requirements of this highly demanding application domain are being increasingly met by commercially available components. One software component which is definitely making its way into space projects is the real-time operating system (RTOS). Apart from reducing the cost of development and validation, commercial off-the-shelf (COTS) operating systems offer the flexibility that is necessary to succeed in the battle of rapid technological changes and perennial space applications.

Space Applications

Space applications can be broadly divided into ground and on-board applications. Ground systems provide ground and operational support for space missions (e.g. International Space Station). Ground support may include a main application configuration database (e.g., the International Space Station has 2 million components under configuration control), software development environments, simulation and test support. Depending on the type of spacecraft and mission requirements, ground applications can provide different kind of operational services. For example, satellite control centers include functions like satellite surveillance and technical control, orbit and attitude control, payload service and satellite expert appraisal.

On-board systems are responsible for monitoring, controlling and collecting data from spacecraft bus systems and payloads (e.g., imaging system). One example of an on-board system is embedded software application for the satellite. The on-board software supervises the satellite or the payload instrumentation, enables satellite to be observed and controlled, and protects it in case of failure.

The main functions of the satellite on-board computers are to:

- Monitor and control spacecraft functions and health
- Scheduling and payload support
- Attitude and orbit control
- Data handling
- Processing system backup
- Communication support

In the space domain, there are many critical real-time components embedded in systems that are part of long lifetime space missions or manned transportation (Powell01). Examples of such applications are:

- Autonomous spacecraft (e.g., the Rosetta spacecraft will fly 10 years to its target, orbit of the comet Wirtanen, with many periods of autonomous operation) and long lifetime scientific probes (e.g., Huygens probe, launched in 1997 should descend to the surface of Titan in 2004)
- Space systems which are tolerant of very harsh environments (e.g., European Robotic Arm)
- Automatic systems performing rendezvous with manned systems (e.g., Automated Transfer Vehicle)
- Manned space vehicles and platforms (e.g., a Crew Transport Vehicle)
- Launch vehicles (e.g., Earth-to-orbit launchers for manned spacecrafts)

Additionally, applications within airborne systems (used to monitor and control an aircraft and to display flight control and navigation data to pilots) are safety critical if the impact of a failure can prevent an aircraft from a continued safe flight and landing.

2. Criteria for selection of COTS RTOS

Compared to requirements for a custom made software component, criteria for the selection of a COTS software component need to be much more flexible and much less specific. COTS components are usually designed with the software marketplace in mind, rather than a specific end-user's need. RTOSs are no exception and there are about 100 RTOSs available on the market today. However, tailoring the criteria for choosing a COTS RTOS has to take into account specific needs of the application domain. For example, one particular requirement specific to the space domain is perenniality, which is the ability of the product to last the lifetime of a space project (e.g., 15 years). This is a complex requirement and includes variety of criteria such as compliance to standards, configurability, development environment, and vendor's business stability.

The following sections will describe the criteria chosen for the selection of COTS RTOS in the space domain.

Hardware

A potential customer for an RTOS is rarely in the position to choose the hardware (i.e. target processor) on which this RTOS will run. Therefore, RTOS availability on a particular type of hardware (e.g., current standard on-board computer is ERC-32, radiation-hardened implementation of the SPARC architecture) is usually the most important criterion for the selection of an initial subset of operating systems.

Device drivers provide interface to hardware that is not supported by the operating system. However, the RTOS kernel has to provide some support for hardware devices to enable the device drivers to perform their function. Device drivers for standard hardware (e.g. Ethernet) can be bought from various suppliers but they need to be recognisable by the RTOS. Device drivers for non-standard hardware very often need to be written by the developers, hence the importance of having facilities to add them seamlessly to the RTOS. Some RTOS vendors offer board support packages in the form of libraries, providing a software interface to the hardware functions of the

board. These libraries may include facilities for hardware initialisation, interrupt handling, hardware clock and timer management.

Design

Operating system can be designed as a monolithic (one piece of software with many interacting modules), layered or client-server structure. A minimal kernel and a set of server and client tasks, makes it a lot easier for the vendor to offer a scalable operating system with more or less services. Scalability means that a set of services used by an application can be optimised, and is even more useful if there is a possibility of dynamic downloading and uploading of services. On the other hand, configurability is a capacity of a RTOS to accommodate changes in underlying hardware (e.g. location and migration transparency).

The space application is a system for guidance, navigation and control of autonomous spacecraft with critical phases. For example, the Rosetta spacecraft will have a number of phases when real-time control from ground will not be feasible. Additionally, the software must function in an environment which is highly constrained in both the memory space and processor power (e.g., the distance from the Sun limits the amount of power available from the solar panels and equipment needs to be powered down). Hence the need for configurability and component optimisation in critical phases.

Space missions require long-term maintenance and support. For some missions, it is very important to have the ability to upgrade code after spacecraft launch in order to correct bugs found during the operation phase, to make modifications due to changes in the environment requirements, and to improve certain functions.

Standards

The space domain covers a lot of very different applications from ground segments, launchers and satellites to probes. However, majority of applications is built according to two main standards: ECSS and DO-178B. The development of all software under the umbrella of European Space Agency (ESA) has to be done in accordance with the European Cooperation for Space Standardisation (ECSS) space product assurance standards (ECSS96) which have recently replaced the quality standard ESA PSS-05. These standards are mainly focused on the software process rather than on the product and, not surprisingly, COTS RTOS vendors have largely ignored them. This is not the case with DO-178B (RTCA92), internationally recognised assurance standard for software development within the aerospace industry. As all new space projects will be facing the DO-178B certification process, some RTOS vendors have already made products addressing the standard's requirements.

An RTOS needs to interact with application components and its environment through well-defined interfaces. With the technology being in a constant state of flux, it seems reasonable to choose RTOS conforming to a widely accepted interface standard. RTOS conformance to the POSIX 1003.1 standard was chosen as a major criterion for the selection of COTS RTOS in the GUARDS project implemented in a space domain (Powell01).

The certification issue has a significant impact on the use of COTS RTOS in safety and mission-critical space applications. Commercial vendors are generally not interested in making their products certifiable to a specific industry standard. However, there are notable exceptions. The certification standard DO-178B defines software criticality levels based upon the contribution of software to potential failure conditions. The most critical level, level A, includes software with anomalous behaviour which can cause or contribute to a failure of a system function resulting in a catastrophic failure condition for the aircraft. Wind River claims that their product "Tornado for DO-178B" (Tornado is Wind River's development platform), not only addresses level A certification requirements, but also includes a binary and source code for the certifiable version of VxWorks as well as documents needed to support the certification process. Aonix, known for its Ada solutions and software development tools, provides products with compilation systems that protect against the use of unverifiable constructs and implement Ada runtime environment that is certifiable to DO-178B level A (C-Smart, Raven).

Development environment

The Ada programming language has been traditionally language of choice in space applications. As this practice is continuing, provision of Ada support remains an important criterion for the selection of COTS RTOS in the space domain.

Apart from compilers and debuggers, other application development tools (e.g., tools for run-time software analysis) are increasingly becoming available as RTOS vendors are competing for the market share.

Facilities

The functionality of any COTS software component is specified with the view of a generic customer, and it is likely that it will both exceed and fail to meet the functionality needed by a specific end-user. COTS RTOS facilities which are of special interest are:

- tasking model (e.g., support for processes and threads),
- inter-process communication mechanism (e.g. , message queue, mailbox),
- memory protection (i.e. memory space with exclusive rights of access can be allocated); also a possibility of software segregation to ensure that more critical software is safely divided from less critical software (note that there is no commercially available operating system providing such support),
- scheduling policy (e.g., fixed priority) and priority inversion method (e.g., priority inheritance)

Vendor

When a commercial software component is going to be used in a mission-critical application lasting 10-15 years, vendor's credentials and stability become very important. The same applies to a component upgrade policy, and a long-term component support strategy. Even when users trust the vendor, they prefer not to be the only users of the product. In a tightly knit space community, the vendor's record is closely monitored and reported.

Due to the small number of players in the space market, the needs of space applications will never drive the world market. However, some RTOS vendors are keen to keep their space customers happy; good examples are Aonix and Wind River.

3. Conclusions

The steady evolution of space application requirements and technologies is pushing the space community towards using commercially available hardware and software components. Many European commercial space projects are already using or considering the use of COTS RTOSs. Some RTOS characteristics are of particular importance to space application developers (more details given in (Beus-Dukic00)). Apart from the obvious ones like availability on a particular type of target processor and type of tasking model, the RTOS chosen for a space project must have support for specific development languages and a collection of application development tools. Furthermore, in applications with safety-critical software components, COTS RTOS needs to be certifiable, the challenge only a few vendors can currently meet.

4. References

Beus-Dukic L. (2000), Criteria for Selection of COTS Real-Time Operating System: a Survey, Proc. of the Data Systems In Aerospace Conference (DASIA 2000), Montreal, Canada, 22-26 May 2000 (ESA SP-457), ESA Publications Division, pp. 387-92.

ECSS (1996), http://www.estec.esa.nl/ecss.

Powell D. (Ed.) (2001), A Generic Fault-Tolerant Architecture for Real-Time Dependable Systems, Kluwer Academic Publishers, 260pp., ISBN 0-7923-7295-6.

RTCA/DO-178B (1992), Software Considerations in Airborne Systems and Equipment Certification, http//www.rtca.org.