Dependency on O.S. in long-term programs: Experience report in space programs

Patrick CORMERY
Tel: + 33 (0) 1.39.06.25.11
Fax: + 33 (0) 1.39.06.27.97
patrick.cormery@launchers.eads.net

Le Vinh Quy RIBAL
Tel: +33 (0) 1.39.06.29.07
Fax: +33 (0) 1.39.06.27.97
le-vinh-quy.ribal@ launchers.eads.net

Arnaud STRANSKY
Tel: + 33 (0) 1.39.06.32.18
Fax: + 33 (0) 1.39.06.27.97
arnaud.stransky@ launchers.eads.net

Abstract

Space programs like launchers are typical of long-term programs: the duration may exceed 15 years. On the other hand software components like Operating Systems evolve more quickly, about 4 to 5 years.

Illustrated in the particular case of EADS-LV space programs, with real time embedded software developed in Ada language, this paper gives an overview of:
- Dependency of an embedded software on Operating System
- The relatively fast evolution of technology, languages and standards
- Industrial solutions to keep the OS permanence during more than 10 years.

This document is the property of EADS LAUNCH VEHICLES and shall not be communicated to third parties and/or reproduced without prior written agreement.
Its contents shall not be disclosed. © - EADS LAUNCH VEHICLES - 2002

1. INTRODUCTION

This paper presents the main dependencies that are encountered in long-term programs, in the particular case of space launchers programs, with Flight Programs developed in Ada language.

Space programs are typical of long-term programs. The complexity of a space system, like a launcher, needs a deployment of competencies in many different fields, and some large industrial facilities. This kind of projects last generally more than 10 years, and sometimes beyond. For example:
- ATV (Aircraft Transport Vehicle): start of development in 1998, up to 2013: 15 years.
- Ariane 4: start of development in 1982, launches up to 2003: 21 years.
- Ariane 5: start of development in 1989, launches expected up to 2010: 21 years.

Therefore the critical components used, as the Operating System of the On-Board Flight Program, shall grant permanence for this duration (permanence of the product and support services such as maintenance, hot line…).

Moreover in launchers production multiple units of a launcher are produced during the project, and they evolve continuously. A target, like the On-Board Computer or the architecture of the Flight Program may keep the same but some features like algorithms or equipments may evolve. Therefore this requires maintaining knowledge and competency in the embedded software development, even for more than 15 years.

2. DEPENDENCY ON OPERATING SYSTEM

An Operating System provides services for real-time features: multi-tasking management, static and dynamic memory allocation, interrupt management…This is an essential part of a Launcher Flight Program, that shall manage cyclic and acyclic commands, parallel computing…

The main dependencies on an OS are:
- Ascendant compatibility between standards (Ada rendezvous behaviour, …),
- Specific real-time extension features (semaphores, …) which are included in the provided OS,
- Target that support the OS (microprocessor, family…) and host workstation for the Software Development Environment,
- Maintain of knowledge and competencies of the OS and user's development teams.

2.1 SEMANTIC FEATURES

In the case of an advanced language as Ada, specialised in multitasking and concurrent systems, the Operating System is nearly close to the language semantic. To keep the Ada standard semantic, the OS shall include an Ada runtime compliant to the standard.
Different ways may be chosen to support a multitasking Ada software:
- Ada software on an Ada OS: the most adequate method.
- Ada software on commercial OS, with inter-layer Ada runtime: this method allows using commercial software, but need the development of inter-layer software that may decrease the performances and the behaviour of the user software.
- Ada software on commercial OS, with procedural calls: in this case the runtime is not fully compliant to the Ada standards, and the software shall call OS-specific functions. This solution is the most dependent on a given OS

This document is the property of EADS LAUNCH VEHICLES and shall not be communicated to third parties and/or reproduced without prior written agreement.
Its contents shall not be disclosed. © - EADS LAUNCH VEHICLES - 2002

2.2 TARGET AND HOST WORKSTATION

Operating Systems are generally a part of a full package that includes OS, compiler, debugging tools, etc. This environment is dedicated to a set of microprocessor or microprocessor families. Experience shows that if software are in theory compliant for a given family (example: transputters, sparcs…), there are still incompatibilities in practice.

2.3 HUMAN RESOURCES AND KNOWLEDGE MAINTAIN

One last point to take into account is the knowledge of developer teams in some specific languages or Operating Systems. The staff training is a large part of a company life, and maintaining an old language or a product specific knowledge may affect a large part of the costs.

3. OPERATING SYSTEMS AND LANGUAGE EVOLUTION

In the case of Ada language, at least three different standards can be identified currently:

- Ada 83: the standard was established in 1983.
- Ada 95: an enhancement of the standard Ada 83 (12 years after Ada83). The Ada 95 is in theory fully compliant with the Ada 83 standard, with ascendant compatibility. The Ada 95 standard introduce among others the object-oriented programming (*"protected objects"*).
- Ravenscar profile: a subset of the Ada 95 standard, established in 1999 (4 years after Ada95). This profile is a set of restrictions of the Ada 95 standard, to be more determinist, and to facilitate the certification of a system based on this profile. This profile restricts in particular the Ada 83 rendezvous, concurrent with protected object behaviour.
- Ada 0Y: a Ada95 new amendment is currently (3 years after Ravenscar) studied by THE AFNOR (French standardisation organisation).

As Operating Systems comply with one or another standard, some semantics aspects may not be compliant between two different OS, despite these OS are all called "Ada compliant".

For example, the rendezvous semantic, a typical Ada features to synchronise tasks each together at precise points of execution is implemented by specific calls to task entries in Ada83. In Ada95 this mechanism is replaced by protected object, with suspending of a task on a protected flag include in the object. As in Ada95 the Ada83 rendezvous mechanism was kept, for ascendant compatibility, this was removed from the Ravenscar profile.
Therefore a Ravenscar-compliant OS is not fully compliant to Ada83 software, some changes have to be made to replace the rendezvous behaviour at least.

Moreover an OS may evolve on account of the OS provider. Reasons of change may be:

- bug correction,
- new requirements from other customers on a product,
- compiler change,
- provider commercial strategy: to change of technology, change of market…

So the dependency is a critical point to manage.

This document is the property of EADS LAUNCH VEHICLES and shall not be communicated to third parties and/or reproduced without prior written agreement.
Its contents shall not be disclosed. © - EADS LAUNCH VEHICLES - 2002

4. SOLUTION ON DEPENDENCY PROBLEM

The Operating Systems providers propose two main policies to preserve product knowledge:

- The freeze policy: the provider engages itself to keep the product at a given version, with documentation and development tools (software and hardware, like host workstations), during the freeze duration. In case of bug, the provider has the adequate equipment and shall be able to correct the bug.
- The knowledge repurchase: after a period of maintain by the provider, it can propose to the customer to purchase the rights on a product, like an Operating System, with source code, documentation, and rights of modification. The maintenance and bug correction became under the responsibility of the industrial customer. As this method grants that the product shall live as long as the project, this requires that the industrial customer develops and maintains a knowledge in some specific field like Operating System development.

If one of these mechanisms wasn't planned at the beginning of a program, and in case of change of a hardware part like the microprocessor, the industrial can ask a provider to redevelop a specific Operating System with the same interfaces than the previous one (in term of semantic and performances). Of course, this method has a cost impact higher than a commercial product.

5. CONCLUSION

This specific sort of application: space and long-term program, lead to a contradiction between the need to freeze the tools, and the evolution of technologies. Industrials have generally to choose some large and stable companies as OS providers to ensure the products permanence. This is to the detriment of the new technologies evolution.
At the beginning of such programs, industrials have to anticipate the future product and technology. Moreover companies may participate to standardisation working groups, to anticipate standards and specify some industrial requirements.

This document is the property of EADS LAUNCH VEHICLES and shall not be communicated to third parties and/or reproduced without prior written agreement.
Its contents shall not be disclosed. © - EADS LAUNCH VEHICLES - 2002