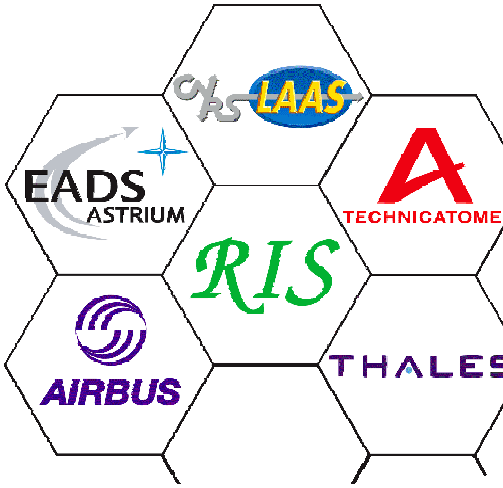
Open Source Software in Critical Systems: Motivations and Challenges

Philippe David, Hélène Waeselynck, Yves Crouzet
European Space Agency & LAAS-CNRS
WCC 2004-Top12

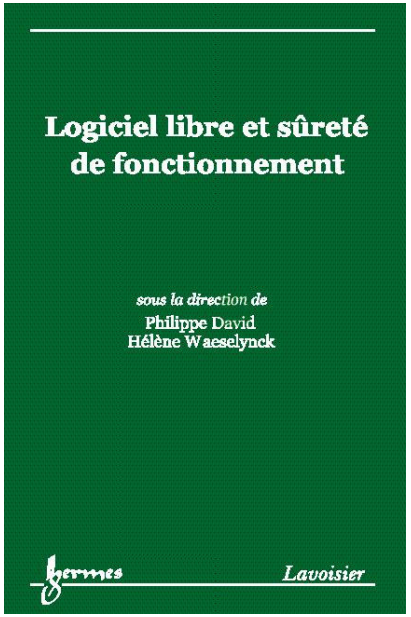
RIS
Network on Dependability Engineering



<http://www.ris.prd.fr>

- Working Group on OSS and Dependability
  (RIS members + participants from Industry ESA, SNCF, and Academia LSV, INRETS, IRISA)

- Publication of a book
  Hermes Science Publications
  (in French) -->



Some Facts

- **Some functions implemented by OSS are used in critical systems:**
  - Operating Systems
  - Communication protocols
  - language
- **OSS projects are more organised than one can usually think:**
  - Funded by associations or groups of industries that share a common interest.
  - OSS development is usually well organised.

➔ **Can we then expect benefits from using OSS in building critical systems?**

How we have conducted our analysis

- Analysing the economical parameters for the use of OSS, in a global manner at system level.
- Exchanging information with industries and laboratories.
- Using feed back information from using COTS.
- Taking into account critical system requirements:
  - Certification,
  - Detailed knowledge of the underlying technologies.
  - Maintenance aspects.

Evolution of critical systems

- **Critical applications are more and more widely used in our society.**
- **Production Cost is more and more constrained.**
  - ➔ Reuse is favoured instead of new software development: COTS.
- **Interoperability of systems is coming: systems of systems**
  - ➔ Use of interface standards is mandatory
  - ➔ Security must be taken into account.
- **Massive use of Software**
  - ➔ 48 kb onboard satellites in 1980 ➔ 1,2 Mb on Mars Express in 2003.
  - ➔ 25 Kb onboard A300B Airbus plane in 1974 ➔ 64 Mb on A380 in 2005.
- **Certification requirements extend to an increasingly larger set of industrial domains.**

Feed-back from using COTS in critical systems

- **System Integrator Needs**
  - Detailed knowledge of the COTS
  - COTS must adapt to the system
  - Certification file must exist
  - COTS must stay available during a 5-10 years period of time.
  - Long term Maintenance ensured for 10 to 20 years
  - Compliance with standards
  - Cost and details of the license must be negotiated

- **Encountered drawbacks**
  - Provider may not be interested in providing support
    ➔ cost of support
  - Integrator does not and cannot know the details of the COTS
    ➔ cost of the certification file
  - Diverging interests of user and provider due to the market evolution
    ➔ cost of freezing the version
  - Freezing a version for a long time
    ➔ maintenance cost
  - For small number of systems
    ➔ cost of licenses is significant
  - Proprietary clauses may constitute a blocking point for system
    ➔ negotiating the licenses is a key

Risk Mitigation

- **Risk due to COTS license**
  - Strategic problem.
  - Industry is used to manage it: license, property.
- **Risk due to COTS Failure**
  - Provider's liability is limited.
  - Failure propagation from COTS to the whole system is a real problem that must be managed by the integrator.
  - The system integrator has no/little knowledge about the COTS, support is necessary.
  - Confidence between provider and integrator is of prime importance. It is not sufficient when dealing with critical systems.
- **COTS provider disappearing**
  - Major industrial risk with no simple solution.
- **Risk due to OSS license**
  - Freedom of use.
  - GPL is contaminating other SW: Major point to be managed.
- **Risk due to OSS failure**
  - The system integrator is the only responsible.
  - Failure propagation must be managed by the integrator.
  - OSS source code is available, the integrator can acquire the technology. Support can be necessary.
  - The integrator must get confidence in the OSS. This is a major issue.
- **OSS evolution**
  - OSS can be maintained by the system manufacturer

Impact of the system maintenance

- **Life time of critical systems is quite long**
  - Satellites: 15 years
  - Command and control for nuclear propulsion in boats and submarines: 40 years
- **Maintenance issues are impacting the system design**
  - Architectural solutions must be used to minimize the impact of version updates.
  - ➔ The use of interface standards is favoured.
  - ➔ Wrapping mechanisms allow changing Software versions with minimum impact on the system.
- **Long Term Maintenance asks for risk mitigation actions to cope with the change of provider**
  - ➔ Availability of the source code is mandatory

Assets of using OSS in systems

COTS → group of users → standards for interface → OSS

Specific, Proprietary → Standard, Public → Easier Interoperability

- No restriction to access the source code
- Does this access to source code help in easing the design/development/maintenance of critical systems?
- Several scenarios are encountered:
  1. Acquisition phase of the OSS Technology.
  2. Adaptation phase of the OSS to the system.
  3. Building the certification file.
  4. Operational maintenance.
  5. Putting in place a long term maintenance team.
  6. Managing major system evolutions.

Scenarios of use of the OSS

	Technology Acquisition	Adaptation to system	Certification documents	Operational Maintenance	Long Term Maintenance	Major Evolutions	Synthesis
Scenario 1 ✓No certification ✓No maintenance	Not necessary	Done by OSS Provider	Not necessary	Done by OSS Provider	Use of Source Code	Done by OSS Provider	No investment. Risk is low and accepted. *Situation in Space today*
Scenario 2 ✓No certification ✓Maintenance	Done through OSS Provider	Done by OSS Provider	Not necessary	Done by Integrator	Done by Integrator	Done by Integrator	Technology Acquisition. Investment in an in-house OSS maintenance team.
Scenario 3 ✓Certification ✓No Maintenance	Done through OSS Provider	Done by Integrator	Done by Integrator	Done by OSS Provider	Done by OSS Provider	Done by OSS Provider	Technology Acquisition. Certification by the Integrator. No maintenance on OSS.
Scenario 4 ✓Certification ✓Maintenance	Done through OSS Provider	Done by Integrator	Done by Integrator	Done by Integrator	Done by Integrator	Done by Integrator	Technology Acquisition. Certification by the Integrator. Investment in the OSS maintenance team.

Assets and drawbacks

- **Access to source code**
  - Allows mastering the evolutions of the software
  - Independence from any provider
  - Major risk: in case of failure, got source but without getting corresponding knowledge. This is the same with COTS.
- **OSS Technology Providers**
  - Same process as for COTS, without licensing problems.
  - Provided support is often of better quality than for COTS as the provider core competence is the OSS technology and not selling license.
- **Technology Acquisition**
  - Detailed Technology Acquisition on the OSS may cost several person.years
  - Investment is heavy on short and long term in order to maintain the OSS team during the project life time.

Dependability of the OSS

- **Some design infrastructure must be used to host the OSS as:**
  - The OSS is potentially a point of failure whose modes are not known.
  - The OSS functionalities may be too abundant or not fully suitable.
- **Use of wrappers**
- **Partitioning the critical system into different criticality levels.**
  - Error confinement mechanisms allow critical systems to be open for interoperability with other systems.
- **Security**
  - Should be taken into account as the OSS has been developed by a third party, often not known.

Certification

- **Certification has a strong impact on the design of the system.**
- **Dependability and ability to be certified are not taken into account by OSS design.**
  - ➔Reluctance of industry to use the OSS.
    - Must be performed by the industrial user.
- **Our objective : to analyse the certification processes of the various industrial domains in order to**
  - identify methods and efforts for allowing system certification when using OSS
  - ➔ OSS must demonstrate a competitive advantage for the system
  - ➔ Without introducing new risks

Certification: overview on various industrial domains

- Levels of criticality are ordered in a similar way in all the industrial domains.
  - DAL (Development Assurance Level) in aeronautics
  - SIL (Safety Integrity Level) in railway

Category	Railway	Aeronautics	Space	Nuclear
No Impact	SIL 0	E	/	/
Impact on system	SIL 1-2	C-D	critical	B and C
Impact on human lives	SIL 3-4	A-B	catastrophic	A

Impact of the safety levels on the system architecture: aeronautics

- Safety analysis top down from system to all equipments that contribute to safety.
- Certification body has a dedicated referential for Software design and development, the DO-178B, who provides recommendations in the aim of guarantying the system safety:
  - Electrical command of the planes are software implemented.
- 5 software categories (A to E) are defined
  - Depending on the impact a software failure may have on the system.

System solutions to the use of software categories: aeronautics

Classification of failure conditions	Level of redundancy		
	0	1	2
Catastrophic	A	B	C
Dangerous	B	C	D
Major	C	D	D
Minor	D	D	D
No effect on safety	E	E	E
	DAL (Development Assurance Level)		

A critical software function able to lead to a catastrophic failure must be either:

- Not redounded. In this case, it is classified in software category A.
- Duplicated. Each of the two software versions is classified in category B.
- Triplicated. Each version is classified in category C.

Certification and dependability of OSS

- A critical system can be designed from less critical functions only if they are redounded and the redundancies are managed according the safety requirements of the system.
- Communication protocol or operating systems are potential candidates for use at level C or D.
  - ➔ Use of redundancies renders the certification feasible for use of OSS at level C or D.
- Use of OSS at levels A or B implies a dedicated development process where the Software is specifically developed and certified accordingly.
  - ➔ Building new OSS.

Development method

- **DO-178B defines objectives**
  - The certification case must contain proof elements that contribute to a negotiation between the industry and the certification body.
- **In nuclear, railway and space domains, a design and development method is imposed per category.**
- **Three major objectives:**
  1) Fault avoidance by applying rigorous development methods,
  2) Fault removal by using tests and integration tests,
  3) Protection from remaining faults through the use of dedicated functions for fault tolerance and robustness

➔ **It is possible to harmonise the certification processes among these domains: in terms of software life cycle and methods.**

Part of the certification effort can be shared by consortium of users

Bringing OSS to the level of use in a critical system requires two kinds of effort:

- Generic tasks: depend only the software and results may be used by all system willing to embed the OSS.
  - Documentation
  - Tests
- Specific tasks: depend on the system, mainly oriented towards safety and hardware interface.
  - Safety assurance plan. For SIL 1 et SIL 2 (C, D), safety requirements are quite limited.
  - Hardware integration.
    - Software test on hardware must be rerun for each project.

➔ Development and test efforts are mainly system independent or must be re run anyway.

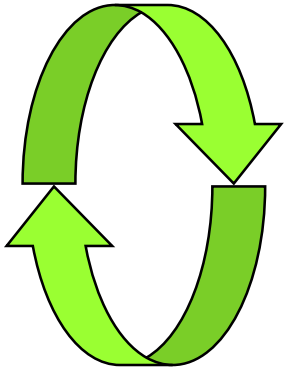
➔ Certification effort can be anticipated.

Usable methods for integrating an OSS in a critical system

- **Analysing the certification standards of the various industrial domains for critical systems allows us to conclude that:**
  - A list of common method can be used to adapt and integrate an OSS in a critical system.
  - These methods depend on the criticality of the function and not on the industrial sector in subject.
- **Dedicated solutions exist to embed OSS in critical systems**
  - wrappers
  - Partitioning
  - Protection mechanism: security?
- **System architecture must be based on interface standards**
  - Favour the use of OSS components
  - Enhance the system life time and ease the maintenance

Expected benefits of using OSS: virtuous circle

- Use of interface standards allows exchanging Software between projects and companies
- Building the OSS Certification file is a big effort, requires heavy investment but it can be shared.
- Reluctance of industry to use OSS comes from the perceived non compatibility of OSS with certification
  - We demonstrated that this is not true when using proper architectural solutions at system level.
  - Risks are better managed than with COTS
  - Building the OSS Certification file must be started by a group of user companies.
    - Sharing the effort
    - Common use of the file that will be enriched from various operational practices.
- Initiating the virtuous circle



Conclusion (1/3)

- **Use of COTS has demonstrated some limitations.**
- **Use of OSS brings real assets:**
  - Comply to standards
  - Lower risk
  - Source code use for adapting and maintenance
- **Perceived drawback**
  - Non compatibility with the certification process but we demonstrated that solutions exist:

➔ **Architectural solutions to host OSS components**

➔ **Starting the virtuous circle by an industrial initiative**

➔ **Industry can now be beneficial in contributing to the OSS community**

Conclusion (2/3): initiatives for promoting the use of OSS in critical domains

- **Setting-up a industrially shared set of methods and tools to use OSS in our systems: repository on internet.**
- **Upgrading OSS to fulfil industrial constraints.**
  - Common methods and tools
  - Sharing the OSS
  - Tools are put at the disposal of users.
  - development environments used to produce critical systems
- **Evaluating OSS, and capitalizing on their use**
  - Validation : characterisation of failure modes and performance
  - Wrappers can be made available
  - Configuration of OSS for dedicated use or hardware
  - Starting the common activity towards a certification file

Conclusion (3/3): Future must be prepared

- **OSS is not the only open source item:**
  - Open hardware (VHDL or C models)
- **System engineering is more and more using simulation models from which code is automatically generated.**
  - Models must be made freely available
- **A new license**
  - Industrial needs are not consistent with GPL conditions. A lot of new ad-hoc licenses are emerging.
  - A license for industrial use of the OSS must be established.
  - Can be an European initiative