Title

OXYDE
Executive Summary

	Name and Function	Date	Signature
Prepared by	*OXYDE Team*		
Verified by	*Jacques Séronie-Vivien* *OBSW Senior Expert*		Jacques SERONIE-VIVIEN 2023.12.15 09:46:28 +01'00'
Approved by	*Michaël Melchiore* *OXYDE Technical Officer*		Melchiore Michael 2023.12.15 09:42:06 +01'00'
Approved by	*Frédéric Maillard* *OXYDE Project Manager*		maillard Signature numérique de maillard Date : 2023.12.15 10:19:38 +01'00'

Document type	**Nb WBS**	**Keywords**
Technical Note (TN)		

See page 2 for Export Control Information concerning this document's content

Export Control Information concerning this document's content

The 2 sections below contain information related to dual-use regulations.

1/ European regulation controlled content

Country of origin for controlled data in this document:
☒ France☐ Germany☐ Spain☐United Kingdom

☒ Does not contain national dual-use controlled data.

☐ These commodities, technology or software are controlled by the European Union in accordance with dual-use regulation 428/2009 under classification.

2/ US Regulation controlled content

☒ Does not contain US origin EAR or ITAR controlled data.

☐ These commodities, technology or software are exported in accordance with the United States Export Administration Regulations. Diversion contrary to U.S. law is prohibited. These commodities, technology or software are controlled under ECCN.

☐ These commodities, technology or software are exported in accordance with the United States International Traffic in Arms Regulations. Diversion contrary to U.S. Law is prohibited. These commodities, technology or software are controlled under ECCN.

1 STUDY SUMMARY

OBSW development takes into account multiple design constraints linked with typical real-time SW development such as reliability, determinism and performance on constrained HW. Such constraints are often at odds with each other. For example, determinism encourages static memory and ressource allocation, but optimal performance requires dynamic exploitation of these same resources.

Current reference OBSW practices have managed to capture an acceptable tradeoff between those (often) conflicting constraints. But it comes at significant price for OBSW developers: numerous guidelines must be followed during development, and enforced during code reviews. This challenge has become quite apparent in recent years, as customers expect more complex SW features delivering optimized performance from multi-core HW targets.

In this context, the study development activities provide strong indications of the potential benefits of Rust and ECS-like design principles for OBSW development. Novel ECS-based scheduling techniques and Rust "fearless concurrency" properties combine to offer automatic and deterministic parallelization of tasks. Furthermore, a Rust-powered development environment should be significantly cheaper to maintain than its C equivalent due to both its configuration language consistency and its dramatically reduced instruction footprint.

The Rust ECS demonstrator and a traditional C OOP PLSW implementation were compared against benchmarking scenarios designed to generate significant CPU and network load respectively. Measurements show the ECS engine does not arbitrarily limit the performance of CPU or IO bound operations. Rust and C code can reach similar performances levels. Furthermore, one can wrap C/C++ code in Rust with no measurable runtime overhead when significant assets are available in those languages. Rust modern and growing ecosystem also allowed us to seamlessly integrate a HW accelerated data processing library to obtain significant performance gains (x4 throughput) at negligible costs.

Id	PLSW	CRC implementation	Mean computation time	Mean CPU load	Peak CPU load	Achieved Throughput
A	Reference	CRC32Fast (C++)	990 ms	25 %	40 %	404 MB/s
B	Demonstrator	CRC32Fast (C++ wrapped in Rust)	990 ms	25 %	40 %	404 MB/s
C	Demonstrator	crc32fast (Rust)	1 070 ms	25 %	50 %	374 MB/s
D	Demonstrator	crc32fast (HW accelerated Rust)	215 ms	25 %	50 %	1 860 MB/s

Table 1 CPU load benchmarks summary for a 400MB file

The ECS engine overhead remains to be measured more precisely, due our benchmarking environment limitations. Memory profiles showcase the significant investment our department has made into our current PUS core products over the last decades. By contrast, the demonstrator presents significant optimization opportunities, especially regarding runtime memory management. Still, the benchmarking results are really promising, given the study participants limited Rust expertise and allocated budgets. Most noticeable performance gaps originated from implementation differences which were, possible in both explored approaches.

Cost analyses from several experts in the department estimate a potential cost-saving opportunity of up to 15% on total SW development recurring costs. Two key optimisation factors emerged during estimates: Rust productivity and verification focused ecosystem and the ability of an ECS development framework to guide multi-team, concurrent SW developments.

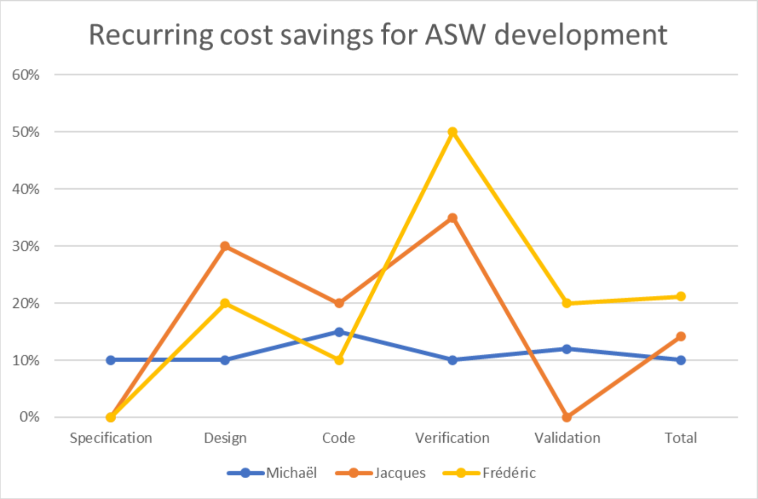

Figure 1 Recurring cost savings estimates for ASW development

Airbus Defense & Space intends on expanding the study benchmarking coverage with additional PUS features and more stress-oriented scenarios. In parallel, platform software development will require driving the Rust RTEMS port forward with the help of European partners and the Agency sponsorship. For payload projects, Rust and an ECS-like execution engine already seem like a compelling alternative to exploit new ARM-based multicore CPU while addressing the application state management issues coming from their non radiation hardened status.

DOCUMENT CHANGE LOG

Issue/ Revision	Date	Modified pages	Observations
1.0	07 Dec 2023	All	First official release
1.1	15 Dec. 2023	All	Remove copyright notice from document