D3.10 Lessons learned from industrial case studies Version 1.0

Document Information

Contract Number	611085
Project Website	www.proxima-project.eu
Contractual Deadline	m36, 31-2016-September
Dissemination Level	RE
Nature	O
Authors	
Contributors	Mikel Azkarate-askasua (IKR), Fabrice Cross (AST), Franck Wartel (AIF)
Reviewer	Mikel Azkarate-askasua (IKR)
Keywords	Industrial case studies, experimental evaluation, analysis process

Notices:
The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n^o 611085.

Change Log

Version	Description of change
v0.1	First Draft of the Document
v0.2	Include feedback from internal review
v0.3	Include feedback from GA review
v1.0	Initial Draft released to the European Commission

Contents

Executive Summary

Case studies have been conducted by industrial partners within the consortium to evaluate the PROXIMA approach and resulting tools, in particular against current practice in the industry in terms of requirements, certification, integration and verification strategies. These studies cover various domains where real-time systems are relevant: avionics, space, railway and automotive. This deliverable captures an overview at MS3 of the status of the infrastructure defined within PROXIMA to support the case studies. Focus is given to feedback from industrial partners regarding the impact and evolution of the evaluated approaches.
The document is organized as follows: in Section 1 we briefly recall the industrial case studies in PROXIMA and discuss on the current practice of software/system integration and verification, elaborating on possible discrepancies between consolidated practice and the PROXIMA approach. On that light, in Section 2, we collect industrial considerations and feedback on the PROXIMA process, especially focusing on its procedural steps and its requirements on the end-user.

1 Case studies and current industrial practice

In the implementation of each case study the industrial user is likely to diverge from the standard approach and consolidated practices to the verification and qualification of software systems in use within his organization. In the following we recall the essential details of each case study and briefly present the approach currently adopted within the specific organization, which is considered representative of the standard practice in the respective domain.

1.1 Avionics case study

The avionics case study focuses on extracts from the flight control system IMA application, namely the Flight Control Data Concentrator (FCDC) and the Weight and Balance Back-up Computation (WBBC). These functions are responsible for in-flight status and failure data, and of the computation of the centre of gravity and weight of the aircraft. Both the FCDC and WBBC are DAL-B A653 applications. The FCDC has been selected for its representative characteristics; lessons learned from the FCDC and WBBC apply to comparable pieces of software such as the Primary Flight Control Software.

Current approach to timing analysis

For IMA applications, running on top of an ARINC-653 operating system, WCET figures are derived by applying standard measurement-based techniques; conversely, applications that are still part of the federated approach paradigm are analysed with static analysis methods, which can be successfully applied thanks to favourable conditions, such as the absence of an operating system layer, cache disabled, lockable or write through, removal of asynchronous source of interference (e.g., preemption, multiple timers) and the existence of an accurate model of the hardware. The selected FCDC and WBBC (DAL-B), and critical DAL-A statically analysed software all comes with unitary tests, integration tests and high level verification tests. However, unitary tests might not be generally available for DAL-B or lower assurance levels where alternate methods might be used to assert DO-178 compliance. Each application selected for the case study is developed and verified in isolation, following the well-known V development cycle reported in Figure 1. For example, the verification of the FCDC IMA application from the case study will follow exactly the above model.
The objective of the *High-level Specification/High-level Verification Tests on Integration Bench* is to verify the whole software functionalities against high-level requirements. This is mainly achieved by performing test campaigns in the presence of the real I/O of the real devices. Tests at this stage also include the real OS in its representative configuration with the complete API (OS and I/O).
High-level Software Components Requirements/Integration Tests, instead, aims at verifying that each subset realizes all its high-level requirements on interfacing with other subsets (which are contextually simulated) of the FCDC and with the actual basic software via the Operating System API. Tests are partially performed on a COTS-based representative environment or on a functionally representative simulator. At this stage, the control flow and the data flow inside the target subset are analysed and the structural coverage analysis of the code is realized. For the

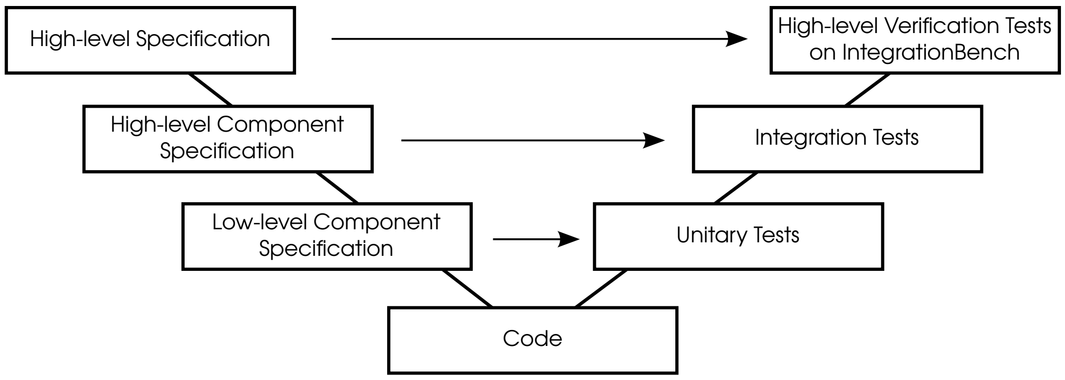
High-level Specification
High-level Verification Tests on IntegrationBench
High-level Component Specification
Integration Tests
Low-level Component Specification
Unitary Tests
Code

Figure 1: Standard V-model.

FCDC application, the aim is to test 100% of the instructions and decisions. In case this is not achieved by the functional integration tests, the structural coverage requirements are met with tests from the unitary tests activity. Finally, at the Low-level Design/Unitary Tests phase, the objective is to verify that each component of the subset fulfils all its low-level requirements. This is achieved by executing UTPs (Unitary Test Procedures) in a representative not-yet-final real target environment. Based on the representativeness of the test environment complementary tests might be performed on the Software Integration Bench (SIB) using the real target. For the execution of these component tests, the called services are in place or simulated. As already mentioned, when required, these tests concur to the completion of the structural coverage.

Approach followed in the case study

Currently WCET analysis and verification of the selected IMA application (ported to PROXIMA target in the scope of the case study) is performed on bench, thanks to a wise (manual) selection of a limited number of input test vectors which reflect what is considered to be the worst operational case (number of conditional nodes activated, number of errors to manage, etc.) by engineering judgement of the system designer. The only relevant difference identified between IMA application development and the non-IMA ones selected for PROXIMA case studies (such as primary flight control or I/O gateway module) is that in the latter case the real CPU and I/O boards are used for all tests. For industrial reasons during IMA application development, a representative simulator or abstract verification techniques may be substituted for the concrete platform.

The approach followed in the basic case study experiments will be similar to measurement-based WCET analysis currently performed to cover the high-level verification tests on bench for IMA software. In that setting the system under tests is considered as a *black box*, with just the minimum instrumentation support required to collect execution time at the granularity level of functions (C language function level). It is important to highlight that the required instrumentation is going to be embedded in the final binary and that certification authorities are typically reluctant to keep instrumentation code which is not direct emanation of a functional requirement. *At this stage AIF identifies no critical discrepancies between the case study and the current verification strategy, which typically consists in high-level measurement-based timing analysis. For the IMA application selected* *for the case study measurement-based is already the current standard; for non-IMA critical software, no static analysis method has been considered to be sustainable when scaling to multicore.*

1.2 Space case study

The space case study selected for PROXIMA consists in a satellite payload implementing the functions of an integrated active optics controller for space telescopes. It includes a low-criticality data processing application (computing the wave front error using data from sensors) and a higher criticality control application, which operates on the actuators controlling mirror displacements. The propagation of timing errors from low criticality to high criticality application must be avoided.

Current approach to timing analysis

The verification and validation of the timing behaviour currently relies on engineering judgment and some automated tools developed internally. During the design phase, in case of software reuse, assumptions are made based on the past experience. Early figures are then fed to bespoke analysis tools for the computation of a static scheduling plan. On a singlecore setting, the fact that the software is mostly periodic allows to start the development phase with enough confidence that the software will be schedulable. Subsequently, during a unitary tests phase timing measurements are collected to verify that the assumptions made at design time are still valid and that the scheduling plan is correct. At this point corrective action can be taken whenever relevant variations are detected. Finally, during the validation phase new measurements are collected to validate the high-level functional behaviour and verify that all deadlines are met. At this point changes, even in scheduling plan, are extremely onerous and thus need to be avoided.

Approach followed in the case study

Also for the space case study, the basic approach for the experiments will be extremely similar to the measurement-based WCET analysis approach already followed. In the case study, high-level input vectors will already be available for the whole software. However, there is no guarantee on the degree of coverage: additional input vectors could be defined, but tool support for this activity would be extremely useful.

1.3 Railway case study

The railway case study, is a mixed-criticality application comprised of two subsystems: European Train Control System (ETCS) signalling and Traction control (control loop). Both subsystems act as closed-loop control algorithm and as a consequence, they continuously require of feedback information from the train environment (e.g., traction encoder, accelerometer, etc.). Additionally, the ETCS subsystem is triple-redundant to meet SIL-4 safety requirements: the PROXIMA node also receives information from two additional nodes that are performing the same functionality on separate platforms, to compare the outputs and check that there are no discrepancies (i.e., voting).
The ETCS signalling subsystem includes three main tasks executed sequentially

as shown in Figure 2, namely the Odometry (OMS), Emergency (ES) and Service (SS) functions. The Odometry module is the responsible of estimating a set of parameters based on the information received from the train environment (e.g., estimated trains position) through a communication channel. The outputs computed by the odometry module are required by the Emergency and Service functionalities to control the braking system.

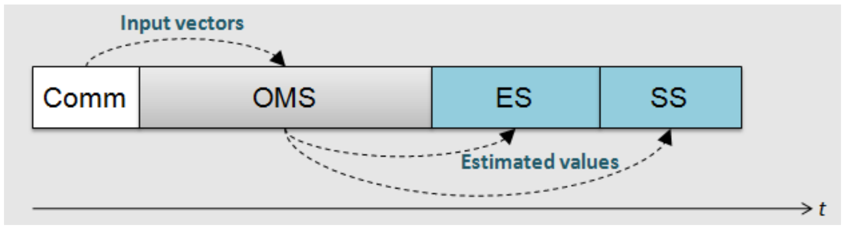

Figure 2: High-level view of the railway case study.

Current approach to timing analysis

Currently IK4-IKERLAN does not use any timing analysis formalism within the projects with its clients, mainly, railway, control systems and industrial machinery, which do not require explicit WCET analysis.

Approach followed in the case study

The Emergency (ES) module of the ETCS subsystem is defined as the Unit of Analysis (UoA) to conduct the probabilistic Worst-case Execution Time (pWCET) analysis of PROXIMA. This ES function has a set of data dependent paths. In particular, they depend on the data computed by the odometry subsystem which in turn requires information coming from several sensors on the train environment. In order to have a controlled scenario, a set of input vectors have been defined. These input vectors parameterize the train environment and provide a basic block level coverage of the UoA.
For the collection of measurements, an Ethernet communication channel is used to send the input vectors to the PROXIMA platform. A minimum of 1000 runs are executed for each of the input vectors and the execution times of each run are collected. Each input vector corresponds to a path of the ES function, with a total of 10 paths to achieve basic block level coverage of the UoA. With these measurements a pWCET curve can be constructed for each of the traversed paths or the Extended Path Coverage (EPC) technology can be applied to have an overall pWCET curve that serves for any path of the UoA.
A set of input vectors has been created to fit the procedure that is normally carried out at unitary testing to measure the timing of relevant paths.

1.4 Automotive lightweight case study

The application originally intended for the case study originally involved both hardware and software from a HybridPack2 (packaged IGBTs for electric motor of hybrid cars) control module. Unfortunately, we will not be able to access all the required HW and SW modules and a substitute case study has been identified.

The new case study will reuse an automotive application from the CONCERTO Project[1], partially funded by the ARTEMIS Joint Undertaking, on which University of Padua is involved. The case study application will consist in an implementation of an automatically generated AUTOSAR-compliant software architecture. Basic element of the architecture are software containers, whose functional behaviour will be provided by functional code generated from Simulink models. The automotive RTOS selected for the case study is `Erika Enterprise RTOS`[2], a certified OSEK/VDX-compliant open-source RTOS, targeting various singlecore and multi-core micro-controllers. Erika is specifically designed to meet application and system level automotive domain requirements as a real-time support with several advanced features keeping however a small memory footprint. It also meets most of the AUTOSAR 4 OS requirements, particularly concerning the multi-core support. A specific system configuration can be defined via an *OSEK Implementation Language* (OIL) specification file, which consists in a set of objects and attributes as parameters at the OS level and application level. The OIL configuration directly affects the system deployment phase and the run-time behaviour as it specifies a set of parameters for the target platform (i.e. CPU and MCU data), the RTOS deployment directive (i.e. into RAM or FLASH memory) and the activation of specific OS features, such as (*a*) task model and attributes – preemptive and non-preemptive multitasking, periodic and non-periodic task activation – (*b*) scheduling attributes – fixed-priority with immediate priority ceiling protocol, earliest-deadline-first scheduling, server-based scheduling – (*c*) shared resources, events, counters and alarms (*d*) interrupts and errors handling, and (*e*) tasks and OS hooks functions Erika is a fully partitioned RTOS: tasks are mapped to specific objects (i.e., cores) in the OIL and must be statically defined at compile time.

The automotive is a lightweight case study and cannot guarantee the same level of involvement by the respective industrial partner (IFX). The current industrial practice in the reference domain has not been considered in this report.

[1]The CONCERTO Project, `http://www.concerto-project.org/`
[2]Erika Enterprise RTOS, `http://erika.tuxfamily.org/drupal/`

2 Considerations on the PROXIMA approach

The PROXIMA approach, in line with current industrial practice, focuses on measurement-based techniques to evaluate the behaviour of a UoA. Another aspect of the PROXIMA philosophy is the focus on techniques mostly transparent to the end users, through automated tools and platform-level methods, to ease the integration process. The successful integration of the case studies on a variety of software and hardware stacks is a testimony to that effort.

The analyses needs not to be provided with detailed information regarding the analysed application, and only limited knowledge of the underlying platform is required. The additional requirements on the end-user are minimal compared to current industrial practice; where current practice may rely on engineered worst-case scenarios, PROXIMA relies instead on automatic and proven methods. The proposed approach still allows for the analysis of complex multicore architectures where deterministic techniques or HWM estimates are either absent or untractable. They have been confirmed to be viable during the course of the case studies.

The PROXIMA tools and methods are further supported by strong scientific arguments. Focus has been given on ensuring the collected and processed observations lead by construction to sound timing estimates. This is supported by positive feedback from the certification authorities.

2.1 Instrumentation and coverage

The insertion of code-level instrumentation, as required by the analysis framework, is not always part of the industrial practice where the embedding of instrumentation code is typically avoided to prevent changes between analysed and deployed systems. Instrumentation code may require additional proofs or tests (e.g. to prove functional equivalence between instrumented and non-instrumented code).

Moreover it is worth noting that, whereas the instrumentation policy and tracing are platform-agnostic, instrumentation infrastructures are not: the sustainable cost and level of instrumentation depends on the tool support. As an example, the inclusion of contextual instrumentation, capturing information beyond timestamps, may require minimal changes in the application to include the Performance Counter monitoring primitives.

With respect to test coverage and associated metrics, some PROXIMA methods involve greater coverage requirements. Test cases are in practice often defined through engineering judgement to exercise the worst-case scenarios, but those may not satisfy the requirements of the PROXIMA approach. The coverage metrics in the approach may also not match the standard high-level verification strategy, resulting in additional costs and delays to collect the required traces. The application of the Extended Path Coverage (EPC) technique[3] , as an example, relies on the collection of sufficient data for each basic block, units of sequential code with no interleaved branch, in the application.

Any requirements on code coverage in a new method should be assisted with appropriate tools taking into consideration industrial constraints on costs and delay, and

[3]EPC is an extension to MBPTA to improve on the representativeness of input data and is first introduced in deliverable D3.4.

software constraints. An application may indeed re-use unknown software, such as IMA or COTS modules, or feature unreachable branches in a fully configured and integrated platform.

2.2 Collection of measurements

Measurement-based WCET analysis is typically performed on data collected from high-level tests. Additional tests may be performed using low-level platforms, such as a functional simulator, to evaluate non-temporal properties. Discrepancies between the low-level and high-level test platforms may prevent the use of low-level observations. This is notably the case for the PROXIMA approach to COTS multicore platforms as they rely on target-dependent performance counters and internal tracing. PROXIMA can still benefit from the collection of data from lower-level tests, to collect information other than timing. Structural properties of the code and other information prove useful, for example, in the application of the EPC approach.
Incremental verification is a main concern to the industrial users. The PROXIMA approaches to multicore interferences, detailed in D3.8, offer the required support for incremental verification. The impact of interferences is estimated and can be maximised as an external metric factored in the observations fed to MBPTA. The resulting estimates for different UoA can therefore be combined without additional analysis. The PROXIMA approaches also support a parametric evaluation of interferences based on properties of the analysed task's co-runners. This allows the computation of tighter pWCET estimates during integration by reducing the pessimism inherent to a maximised, fully-composable interference profile.
The PROXIMA approach requires the collection of larger numbers of observations than traditional High WaterMark (HWM) analysis. This stems from the coverage requirements of the different analyses (§ 2.1) to provide sufficient confidence in the observed interferences scenarios or structural coverage. The process relies on automatic methods replacing test-cases defined through engineering judgement. Although more time-consuming, the process of collecting the observations fed to the timing analyses is simplified and strengthened.

2.3 Tool support and scalability

Preliminary analysis steps such as instrumentation and data collection have been automated so that later procedures can be smoothly and transparently integrated in the current development process, at any test level. This also includes automated support to detect the actual degree of coverage and any other relevant metric in PROXIMA. The guidance provided by the tools to identify insufficient coverage and guide the definition of additional test vectors or requirements is an important factor. Those steps have been taken by the EPC and VICI approaches, and to a lesser extent the EVT-based analysis (all detailed in D3.8). Those can be further refined, as the end-user may still be prompted to provide assumptions regarding contenders for the inter-core shared resources.
The core analysis steps, such as trace processing, i.i.d. tests, definition of minimum number of runs and final pWCET estimate have been integrated in a set of tools and analysis scripts. The interpretation of results may still require technical

support, but those cases are limited to occurrences properly flagged by the analysis tools. Integration within the Rapita Validation Suite (RVS) focuses on the instrumentation and MBPTA analysis, with intermediate steps provided through documented and exemplified tools operating on the intermediate traces. All results are accessible from the RVS interface but can be only partially exported.

Scalability of any approach is a critical concern when it comes to industrial application. In this view, it is of utmost importance to carefully evaluate the instrumentation level required by the analysis as it will directly impact the scalability of the tool chain to complex industrial-scale programs. As already noted in Section 2.1, the sustainable level of instrumentation also depends on the instrumentation infrastructure. The same reasoning also applies to the algorithmic part of the analysis: the core timing analysis algorithm is currently defined in the R language, which despite being an interpreted language managed to scale to purpose during the case study. Other algorithms have been implemented in different interpreted or compiled languages without any negative feedback regarding their scalability.

Acronyms and Abbreviations

- API: Application Programming Interface
- COTS: Commercial Off-The-Shelf
- DAL: Design Assurance Level
- EPC: Extended Path Coverage
- ETCS: European Train Control System
- FCDC: Flight Control Data Concentrator
- GUI: Graphical User Interface
- IMA: Integrated Modular Avionics
- IOM: I/O Module
- MBPTA: Measurement-Based Probabilistic Timing Analysis
- MCU: Micro-Controller Unit
- OIL: OSEK Implementation Language
- pWCET: Probabilistic Worst-Case Execution Time
- RTOS: Real-time Operating System
- RVS: Rapita Validation Suite
- SIB: Software Integration Bench
- UTP: Unitary Test Procedure
- WBBC: Weight and Balance Back-up Computation
- WCET: Worst-Case Execution Time