D6.4 Evaluation of Telecommunications demonstrator

Project number:	644080
Project acronym:	SAFURE
Project title:	SAFURE: SAFety and secURity by dEsign for interconnected mixed-critical cyber-physical systems
Start date of the project:	1st February, 2015
Duration:	40 months
Programme:	H2020-ICT-2014-1

Deliverable type:	Report
Deliverable reference number:	ICT-644080 / D6.4/ 2.0
Work package	WP 6
Due date:	May 2018 – M40
Actual submission date:	07th August 2018

Responsible organisation:	TCS
Editor:	DR
Dissemination level:	PU
Revision:	2.0

Abstract:	This document presents the evaluation covers the modelling of tasks and resources using the SymTA/S tool, the description of the test methodology, tests of the elements to be evaluated in the systems, and a synthesis of the requirements compliance. It provides a conclusion on the adequation of Android-based terminals and connected systems to provide the safety and security properties for the telecom use-case.
Keywords:	Security, safety, Android, modelling, validation, testing, telecommunications

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644080.
This work was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0025. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

Editor

Dominique Ragot (TCS)

Contributors (ordered according to beneficiary numbers)

André Osterhues (ESCR)

Don Kuzhiyelil (SYSG)

Björn Gebhardt (SYM)

Elodie Leveugle (TCS)

Executive Summary

This document is and update on an initial version of D6.4, including an update on the extend of satisfaction of functional requirements. It presents the results of the evaluation of the Telecom use case demonstrator, as described in D6.3. This demonstrator is based on an Android smartphone connected to a smartband and providing safety and security capabilities. The evaluation covers the modelling of tasks and resources using the SymTA/S tool, the description of the test methodology, tests of the elements to be evaluated in the systems, and a synthesis of the requirements compliance according to the structure defined in D1.3 “SAFURE Framework Specifications”.

It is possible to provide additional security in Android by using additional security components such as Cycurlib, in order to ensure better control of health-related data.

The addition of safety capabilities is presently quite difficult considering the lack of control of the Android platform by applications. Hence safety is limited to application monitoring and alert propagation whenever degraded conditions can be detected.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The aim of the telecom use case is to provide a test platform integrating SAFURE components and able to be evaluated for safety and security aspects. This document presents the test methodology and the associated results.

The telecom use case and the elements composing it are described in the following chapter. The architecture of the demonstrator is also presented with a focus on the safety and security components.

Modeling work and results with respect to task description and timing analysis are presented in Chapter 3.

Chapter 4 presents the test plan methodology, the elements that are evaluated and the evaluation results based on the requirements structure provided in D1.3" SAFURE Framework Specifications". The latter results include the two variants of the architecture for the telecom use case, namely:

- The architecture where separation is provided by a hypervisor on the smartphone.
- The architecture without a hypervisor on the smartphone where separation is partly achieved by the infrastructure along with the smartphone.

Since only the architecture without a hypervisor has been implemented (cf. D6.3), the results for the hypervisor-based architecture have been inferred from similar architectures implemented on other platforms.

Chapter 2 Demonstrator Description

The demonstrator is shortly described hereafter. The use case and interacting devices are presented firstly. Then the architecture of the demonstrator and the components included are described. Further details about the demonstrator are provided in D6.3 “Telecommunications prototype”.

2.1 Use case

The use case is a body area network in where there is a mix of critical and non critical devices as well as secure and non-secure functions.

The goal of the use case is to provide additional capabilities to support the use of critical devices in a secure environment, and the impact of these capabilities on the overall system.

2.1.1 Medical devices

For reasons mostly related to availability, the medical device has been substituted by a smartband. It can illustrate similar concerns and provides the same interfaces, with the monitoring capability of body constants such as heartbeat rate. However it is not able to act on body. This is in line with the project general rules with respect to medical devices.

2.1.2 Hardware platform

The hardware platform consists of a smartphone and an infrastructure which jointly should implement the whole set of capabilities for safety and security as described in D1.2 and according to methodology described in D1.3.

2.2 Demonstrator architecture

The demonstrator architecture is depicted as follows:

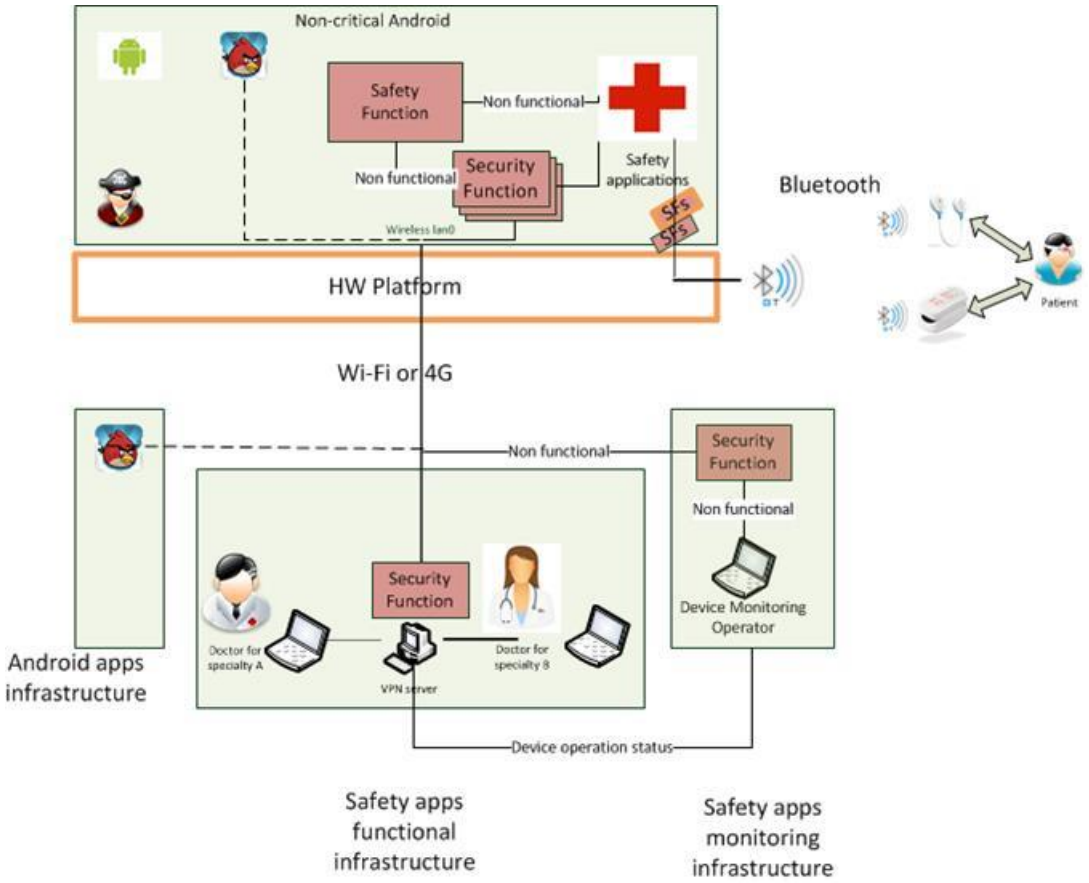

Figure 1: Architecture of the demonstrator

The smartphone (upper part of the schema) is based on Android and provides the additional security and safety functions which are detailed hereafter.

The infrastructure (lower part of the schema) is based on a PC-based system running Linux, and provides the counterparts of the security and safety functions provided by the smartphone.

2.2.1 Security components

The security components are of several kinds:

On the smartphone:

- The native Android security components
- The SAFURE Cycurlib component
- The Matrix component

On the infrastructure:

- The native Linux security components
- The SAFURE Cycurlib component
- The Matrix/Riot component

These components can be used in conjunction or in isolation to provide the necessary security capabilities

2.2.2 Safety components

The safety components are

On the smartphone:

- The monitoring component
- The supervision component
- The logging component

On the infrastructure:

- The log manager component
- The alert manager component
- The reporting component

Chapter 3 Timing Analysis

In the SAFURE telecommunication use case, safety-critical medical applications associated with security-critical applications need to be running alongside non-critical applications on a common platform (smartphone or tablet). The critical applications can run either in a periodic or asynchronous manner, need to be preemptive on any other non-critical application to ensure potential safety requirements while avoiding overload of the running hardware and seamless experience for the end-user of the platform. This functioning scheme requires in-depth analysis of potential timing issues at core level.

For these analyses Symtavision's timing analysis tool "SymTA/S" was used to investigate different scenarios for concurrency of both medical (MedicalApp) and critical (CApps) and non-critical (NCApps) applications according to the simplified architecture, previously captured in the Capella modelling tool in Figure 2.

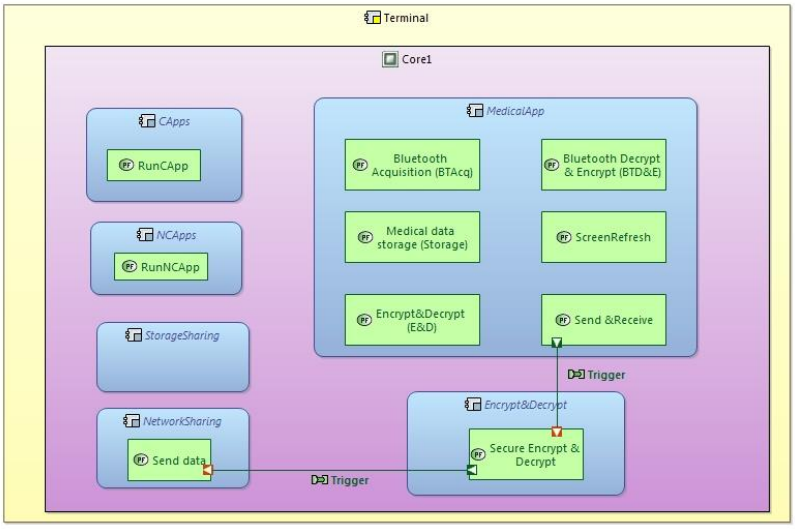

Figure 2: Architecture for SymTA/S timing analysis

This architecture was used as a basis to create the Runnables and Tasks in SymTA/S as shown in Figure 3.

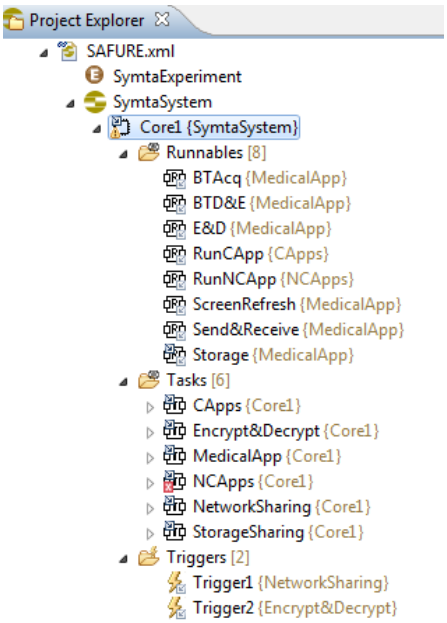

Figure 3: SymTA/S project explorer tree of Task, Runnables and Triggers

For each element, several parameters need to be fulfilled such as the priority, the execution time range and period (when applicable). Triggers can also be used to account for event based activation of tasks. An example is shown in Figure 4. These parameters are key to investigating the impact of processes number, length and interdependence on the viability of architecture, thus allowing design-space exploration.

Name	Value
ResponseTime: Value	[100 ms;250 ms]
PropagatedModel: Activation	P(300 ms)+J(150 ms)
PropagatedModel: Synchronization	Clock1
PropagatedModel: Offset	100 ms
Question: Analyse	true
Question: Extended Results	true
ExecutionBacklockBuffer: Max	1
ResourceTimingConstraints: min. Core Execution Time	0 ms
ResourceTimingConstraints: max. Core Execution Time	
ResourceTimingConstraints: min. Effective Execution Time	0 ms
ResourceTimingConstraints: max. Effective Execution Time	
BufferAccessOverheadLoad: Buffer Access Overhead	0
Load: Total	40%
Load: Execution	0.4
Load: Scheduling Overhead	0%
LoadConstraint: min. Total Load	0
LoadConstraint: max. Total Load	
ResourceConsumptionTime: Virtual TCore	[100 ms;120 ms]
Status: Status	Success

Name	Value
ResponseTimeJitter: Jitter	150 ms
Criticality: Criticality level	
OsekTaskParameter: Task Type	Preemptive
OsekTaskParameter: Activation Overhead	
OsekTaskParameter: Termination Overhead	
OsekTaskParameter: Priority	4
OsekTaskParameter: Non Preemption Group	
OsekTaskParameter: Blocking Time	
Internal: Activation	P(300 ms)
Synchronization: Synchronization	Clock1
Synchronization: Offset	0 ms
Synchronization: Enabled	true

Figure 4: Example of properties for CApps Task

The software then runs simulations of these multi-processes on a single core and provides a series of interpretable data on the various tasks, runnables, core load, etc.

The feature that was the most used in the telecommunication case was the Gantt Diagram, which provides a visual overview of the concurrent running tasks. Used to identify timing bottlenecks or delays, it quickly gives a sense of acceptable configurations and overall impact on safety-security and user experience.

Figure 5 and Figure 6 below are examples of the results obtained with the tool and how it was used to investigate the optimization of concurrent critical and non-critical tasks running on a single core.

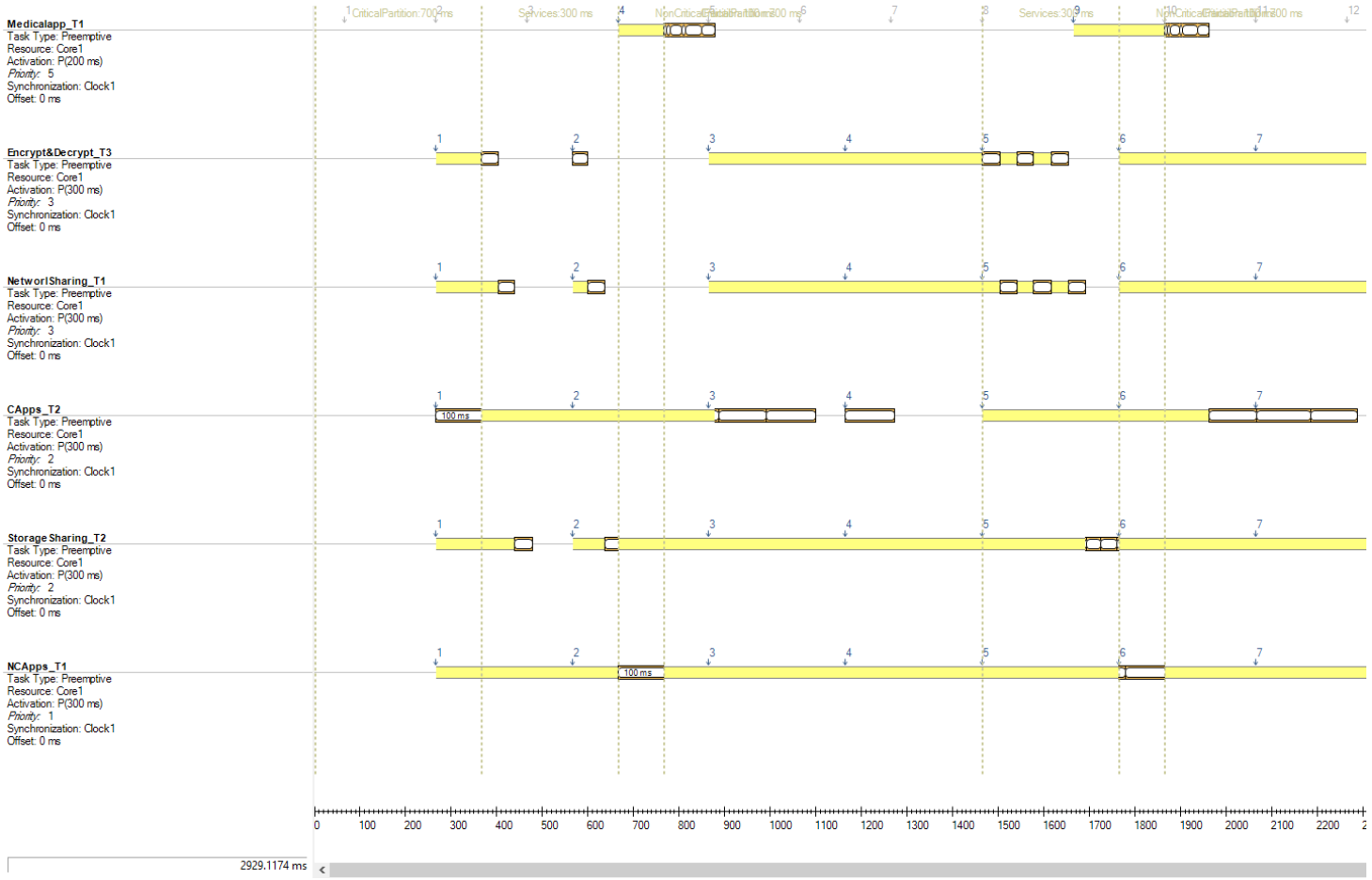

Figure 5: SymTA/S Gantt Diagram - Run with no optimization

Figure 6: SymTA/S Gantt Diagram – run with first sequencing optimization

In the examples of Gantt diagrams above (Figure 5 and Figure 6), each task has been given a priority (from 0 to 5, the highest here) and a repetition period (in ms), and a duration which is the sum of the duration of its runnables. The SymTA/S software then runs the simulation taking into account those parameters and provides the diagrams. All tasks are running on the same core, therefore in a sequential manner. At all time the running task is represented by a white box on its line and the priority setting will decide which task is run from start to end. Tasks with lower priority can be delayed or even pre-empted, as represented by the yellow lines and interrupted white boxes. In the first diagram, one can see that most tasks are delayed, even the task with highest priority (first line).

We then performed some optimization that account for the specificities of some of the tasks: reassigning priorities, introducing trigger-inducing tasks instead of periodic repetition and adding offsets (in ms). Indeed, as in the telecommunication case some event can occur on a non-regular basis, it is important to account for specific sequencing of tasks.

In the last example, we insure that the objectives are met for most of the time. However, the periodicity and priority of the MedicalApp task is such that the NCApps are almost constantly delayed. This could translate into poor user experience. Also some instances of the critical task turn out to be pre-empted by the MedicalApp task, which raises the question of security versus safety in this mixed context.

These examples show how such timing analysis tool was used to try and solve some of the issues raised and addressed in the SAFURE project. However, fully conclusive usage of this tool would require a more specific solver which could better apprehend event based scheduling on and multi-core mapping.

Chapter 4 Test plan

The test plan is focused at testing non-functional aspects of the system related to SAFURE capabilities and in the meantime to be able to evaluate qualitatively at least the impact of the added capabilities on the behaviour of the system. For instance it can be useful to evaluate the perturbations introduced by a safety feature on a non-safety one, and of greater importance to evaluate the opposite.

4.1 Methodology

In order to evaluate the requirements the following methodology has been chosen.

1. We have identified the elements that can be evaluated according to the SAFURE framework. These elements have to be measureable and their measurement in nominal mode shall be known.
2. Then we have identified the tests that can be done when these elements are not working in nominal mode and how the degraded mode can be detected and, if applicable, quantified

4.2 Elements to be evaluated

The elements to be evaluated are :

- The smartwatch
- The Bluetooth link
- The smartphone
- The WiFi link
- The 2G/3G/4G link
- The infrastructure

For each of these we provide tables indicating

- What items can be evaluated
- The tests can evaluate each or several of the items

4.2.1 Smartband

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
W1	Battery level	> 10%	< 10% and > 2%	< 2%
W2	Heartbeat sensor	OK/on		KO/off
W3	Heartbeat sensor calibration	Done in last 24h	Done in last 72h	Not done

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
W1	Battery drain/charge close to thresholds.	N,D,A	> 1h	No
W2	Isolate sensor from wrist	N,A	< 1min	No
W3	Activate calibration	None	Unknown	Yes

4.2.2 Bluetooth link

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
B1	Connected devices	< 3 including 1 critical	> 3 including 1 critical	No critical device connected
B2	Link signal level for each critical device	> 50 %	> 10 % and < 50 %	< 10 %
B3	Link interference for each critical device	Low	Moderate	High
B4	Link jitter level for each critical device	Low	Moderate	High

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
B1	Fetch information from settings.	N,D,A	< 1min	Yes
B2	Move smartphone away from smartwatch	N,D,A	< 1min	No
B3	Use several smartwatches	N,D,A	< 1min	No
B4	Use other active Bluetooth devices	N,D,A	< 1min	No

4.2.3 Smartphone

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
S1	Critical apps running	< 3 and > 0	> 3	0
S2	Monitoring of each critical app	App-dependent	App-dependent	App-dependent
S3	Monitoring frequency reported for each critical app	Yes	No, guessed by monitor	No, not guessed by monitor
S4	Uptime of each critical app			
S5	Number of failures/ restarts of each critical app			
S6	Time of last failure/restart			
S7	Cause of last restart	By planned action or user action	After Failure	Unknown
S8	Battery level	> 10%	< 10% and > 2%	< 2%
S9	Supervision status	OK	degraded	KO or not reported
S10	Safety subsystem autotests	Done in last 1h	Done in last 24h	Not done in last 24h
S11	Last acknowledgement received from infrastructure			

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
S1	Identify critical apps by name	N,D,A	< 1min	Yes
S2	Critical app-dependent test	N,D,A	< 10min	Yes
S3	Periodic cooperative monitoring of critical app	N,D,A	< 1min	No
S4	Event-driven reliability criteria based on S4-S7 Trigger applications restart	Not applicable	App-dependent	No
S5	Cf. S4			
S6	Cf. S4			
S7	Cf. S4			
S8	Battery drain/charge close to thresholds.	N,D,A	> 1h	No
S9	Stop/start supervision application	N,D,A	< 1min	No
S10	Start/stop safety functions test	N,D,A	< 1min	Yes
S11	Same test as I2			

4.2.4 WiFi link

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
F1	Link security	WPA2	WPA	none
F2	Link signal level	> 50 %	> 10 % and < 50 %	< 10 %
F3	Link interferencewith respect to other beacons	Low	Moderate	High
F4	Link jitter level	Low	Moderate	High

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
F1	Change WiFi beacon	N,D,A	< 5min	No
F2	Move smartphone away from WiFi beacon	N,D,A	< 1min	No
F3	Use several colocated WiFi beacons	N,D,A	< 1min	No
F4	Use other active WiFi devices	N,D,A	< 1min	No

4.2.5 2G/3G/4G link

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
G1	Link type : 2G/3G/4G	4G	3G/2G	None
G2	Link signal level	> 50 %	> 10 % and < 50 %	< 10 %

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
G1	Change network settings for SIM	N,D,A	< 1min	No
G2	Move smartphone in area with low connectivity (indoor)	N,D,A	< 1min	No

4.2.6 Infrastructure

Items that can be evaluated

No	Item	Nominal range	Degraded range	Alarm range
I1	Last status info received from each smartphone	< 3 h	> 3 h	> 24 h
I2	Report of acknowledgement	< 3 h	> 3 h	> 24 h

	received by infrastructure			

Tests that can be used

No	Test	Monitoring ranges	Test typical duration	Remotely triggable
I1	Switch smartphone to airplane mode	N,D,A	> 3h	No
I2	Visual display at monitoring console	N,D,A	> 3h	No

4.3 Requirements compliance

The requirements from D1.2 have been evaluated in two cases:

1. The initial architecture with PikeOS. Information is based on knowledge on similar systems and extrapolated to the specific test case.
2. The revised architecture with Android. Information is based on implementation results for the Telecom prototype

4.3.1 Common Integrated Requirements

Use Case	ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android	TableID
INTEG	CR-NF-001	Hypervisor shall provide real-time guarantees when scheduling virtual machines/partitions	Covered by State of the art and SAFURE	Yes in PikeOS	No	CIR01
INTEG	CR-NF-003	The real-time OS should provide ways to access the hardware monitoring features of the hardware platform. Virtualization needs to have a minimal impact on the availability and accuracy of the monitoring features		Unknown	No access with Android. No real-time platform	CIR02
INTEG	CR-NF-032	An upper bound must be computed on the delay of communications over Ethernet for safety-critical traffic	Applies to all UCs for which timing analysis shall be performed	Unknown	No access with Android. No real-time platform	CIR03
INTEG	CR-NF-033	An upper bound must be computed on the delay of communications over Ethernet for safety-critical traffic also in the presence of unknown/expected traffic	Applies to all UCs for which timing analysis shall be performed	Unknown	No access with Android. No real-time platform	CIR04
INTEG	CR-NF-034	An upper bound must be computed on the hardware utilization of communications over Ethernet (bandwidth, buffer) for safety-critical traffic	Applies to all UCs for which timing analysis shall be performed	Unknown	No access with Android. No real-time platform	CIR05
INTEG	CR-NF-013	Hypervisor should provide support to treat energy/temperature information on scheduling level or propagate it to the dedicated user applications	in hw-virtualization mode, PMAU and scheduling/synchronization API can be used by application	Unknown	Not applicable	CIR06
INTEG	CR-NF-014	Hypervisor shall provide means to confine HW-based covert/side channels	in hw-virtualization mode, PAMU and scheduling/synchronization API can be used by application as appropriate setup of time and partition isolation	Unknown	Not applicable	CIR07
INTEG	CR-NF-017	Hypervisor should provide support for PKI	e.g. file provider API	Unknown	No Hypervisor but PKI through Android and Citadel	CIR08
INTEG	CR-NF-020	The cryptographic services shall provide a common interface to Hardware Security Models and Software libraries.	Covered by Safure. Interfaces are provided so that other software applications do not need to know the implementation of all cryptographic services	Unknown	Yes with some Android versions	CIR09
INTEG	CR-NF-023	Hypervisor shall provide temporal and spacial separation of applications	Covered by State of the art and SAFURE	Yes	No applicable	CIR10
INTEG	CR-NF-025	Multiple safety/security criticality levels have to be considered for software/hardware components, not only a 'naive' separation between critical and non-critical (best-effort). These different level of criticality have to be taken into account at tool, especially at the analysis level of the tool composing the toolflow.	Generic from the DoA. Integrated for Safety.	Yes through PikeOS	No	CIR11
INTEG	CR-NF-029	The proposed HW platform to be evaluated in WP4 for final selection should encompass some shared HW resources shared by several cores (>4) such as shared memory (such as distributed memories or caches, preferably distributed SRAM memories) but also the SoC interconnect and I/O devices. The real-time analysis should not only take the shared memory into account but also other resources	WP4 requirements. Covered by the chosen HW platform	Yes for HW platform	Yes for HW platform	CIR12
INTEG	CR-NF-030	According to the system predictability criteria defined by the PREDATOR project, there is a strong need for large local memories on the multicore platform. The size of the local memories should be enough for the storage (instructions & data) of any single application task	To control interferences. Covered by the chosen HW platform	Yes	Yes	CIR13
INTEG	CR-NF-031	The selected hardware platform should encompass multi-core technology with at least 4/8 cores such as the 4-core i.MX6Q, the 8-core P4080 or the 12-core T4240. To make sure that all techniques proposed in the SAFURE project are scalable, dual-core architectures should be avoided as they usually encompass specific non-scalable features	From the DoA, targeting multi-cores. Covered by the chosen HW platform	Yes. Quad-core	Yes. Quad-core	CIR14

Table 1: Common integrated requirements

Although the hardware platform complies to these requirements the capabilities usable by PikeOS and Android, despite being complementary, are not mixable on this hardware platform. In order to have a better coverage it is needed:

- To extend Android in order to support some more real-time capabilities, for instance by enabling access to other scheduling policies already existing in the Linux kernel.
- To enable virtualization at the HW platform level or at the Android level in order to be able to integrate a hypervisor such as PikeOS to manage real-time aspects and HW monitoring.

4.3.2 Common Functional Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
CR-F-001	Mixed-critical safety requirements and time-critical requirements need to be coupled in at least one of the use-case supporting PikeOS, including the possibility to run concurrently different tasks with different safety levels, or the ability to support a degraded mode for lowest critical tasks	Requirements for the research performed in WP4. Else WP4 will use a dedicated prototype. Integrated in the WP4 prototype	Yes	No
CR-F-002	The use-cases should quantify their usage and requirements in term of accesses to the different shared hardware resources of the target platforms for the adaptive solution to guarantee the associated requirements based on observed behavior.	Requirements for QoS algorithm developed in WP3.	not addressed unless portable onto PikeOS	not addressed unless portable onto Android, partially addressed by modelling

Table 2: Common functional requirements

None of these requirements could be satisfied in the architecture with Android.

For CR-F-001 it has been alleviated by using a dedicated architecture for the WP4 prototype. Support for mixed-criticality is not existing in Android and it could be implemented using an hypervisor should the HW architecture support it. This is the case for the ARMv8 architecture but among the smartphones tested built with ARMv8-based cores or similar cores, none was supporting this feature.

For CR-F-002 the usage requirement has been modelled but the tools available to monitor it on PikeOS and Android are providing a coarse-grain application view and not the more fine-grain resource view needed. Furthermore, without direct access to PMIC or PMU it is not easy to benchmark these tools

4.3.3 Common Non-Functional Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
CR-NF-002	All the use cases should use tools and SW that is an expression of an ackowledged standard or has a reliable open source implementation		Yes except SymtaVision, Android Studio, CycurLib, PikeOS not open source	Yes except SymtaVision, Android Studio, CycurLib: not open source
CR-NF-005	System description (topology, etc) must be available in an accessible format	Applies to all UCs for which timing analysis shall be performed	Yes	Yes
CR-NF-006	System configuration (communication, tasks, etc) and timing properties (execution times, frame sizes, etc) must be available in an accessible format	Applies to all UCs for which timing analysis shall be performed	Yes for PikeOS partitions	Partially
CR-NF-007	System constraints (deadlines, max load, etc) should be available in an accessible format	Applies to all UCs for which timing analysis shall be performed	Yes for native partitions	Partially
CR-NF-008	Timing behavior must be known/specified for all arbitration points (CPU scheduler, network arbitration, shared resource access, ...)	Applies to all UCs for which timing analysis shall be performed	feasible	Observation only
CR-NF-009	For unknown time consumers (attackers), constraints should be specified (e.g. what resources are affected)	Applies to all UCs for which timing analysis shall be performed	Mapping of SW on HW - Arbiter PikeOS	Mapping of SW on HW - Arbiter Android (no control)
CR-NF-010	Standard arbitration protocols should be used for OS and networks (e.g. AUTOSAR, OSEK, Ethernet)	There will likely be be no support from SYM for non-standard / custom protocols for timing analysis	OK	OK
CR-NF-011	Timing properties (task/runnable execution times, interrupt activation models, network traffic properties) should be derived via tracing, static analysis or budgeting	Applies to all UCs for which timing analysis shall be performed	Possible with PikeOS but not on smartphone	Partially possible with Android Studio
CR-NF-012	WCET analysis techniques and dedicated isolation techniques should provide Time Composability in target multi-core systems by providing features allowing us to compute or bound the co-running interference overhead.		Not likely. Difficult to apply	Not likely. Difficult to apply
CR-NF-015	Hypervisor shall support secure boot of the whole system and each partition separately		Secure boot in Hypervisor	No Secure boot. Platform dependent
CR-NF-016	Hypervisor shall provide secure update of a partition		Yes	No
CR-NF-018	The SAFURE platform must provide services for cryptographic mechanisms and handling cryptographic objects (i.e. keys, certificates). The services must include the following features: a) Managing cryptographic keys. (Generating, deleting and storing keys) b) Calculation of cryptographic functions: - Signature generation and verification - MACs (message authentication codes) - Encryption and decryption c) Managing cryptographic certificates. (Storing and updating certificates)	Covered by State of Art and Safure. This requirment should be fullfilled if a system wants to provide security like confidentiality, integrity, and authenticity.	CycurLIB on PikeOS	Android Security and/or CycurLIB on Android

Table 3: Common non-functional requirements, part 1

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
CR-NF-019	The cryptographic services must provide a configuration mechanism to define the access methods and rights to the cryptographic objects. a) The configuration shall only be done by authorized entities. b) The access rights shall be enforced by the security architecture. c) Access rights must be definable for - Roles and Users - Services - Domains d) Access rights shall define: - Overall access - Access to individual functions using the cryptographic objects. (i.e. generating or deleting keys) e) Usage rights of cryptographic objects should be defined: - Keys for encrypting, decrypting, signing, verifying. - If keys can be deleted, exported, derived or not.	Covered by State of Art and Safure. This requirment should be fullfilled if a system wants to provide access control.	CycurLIB on PikeOS	Android Security and/or CycurLIB on Android
CR-NF-021	A software component should not be allowed to alter, contaminate or delay another software component's code, I/O, scheduling, or data storage areas in uncontrollable ways, especially from the less critical components to the most critical ones. Time isolation and Spatial isolation have to be ensured. New isolation mechanisms can be introduced to ensure software independence in multi-core systems, enabling the safe execution of software components with different criticality levels.	Generic from safety definition	Ensured by partitions	Isolation provided by Android at application execution, not at application installation
CR-NF-022	Failure on hardware unique to a software component should not cause adverse effects on any other software component.	Generic from safety definition	Not likely. Difficult to test	Not in the general case. Maybe for some specific components such as accelerometer and GPS. Difficult to test.
CR-NF-024	Mixed-criticalities must be supported in hardware.	sufficient isolation	No. Isolation provided in SW by PikeOS partitions	No
CR-NF-026	Incremental changes should be supported in the design and verification. The tools should exploit the isolation to keep the effects of incremental changes as small as possible for the higher levels of criticality. This feature is required for incremental certification.	Generic from mixed-critical definition	Incremental/ iterative approach supported by design and modeling	Incremental/ iterative approach supported by design and modeling
CR-NF-027	Hypervisor shall support the platform selected in the telecom use-case		No	No Hypervisor
CR-NF-028	The selected hardware platform has to provide monitoring features such as Performance Monitoring Counter (PMC) or hardware counters, allowing to monitor the timing behavior, the runtime workload on the different hardware resources, and power consumption or energy related features.	For monitoring features required by WP3 and WP4	No. Partial documentation of PMC	No. Partial documentation of PMC

Table 4: Common non-functional requirements, part 2

The requirements that are related to time analysis (NF5 to NF12) are very partially covered by Android and there is no possibility to handle them more completely within Android.

Security requirements (NF15 to NF19) are covered for the cryptography but not for the secure boot. As these security requirements are of major importance for SAFURE, more information of CR-NF-018 and CR-NF-019 is provided hereafter.

CR-NF-018 coverage on PikeOS

CycurLIB has been integrated into PikeOS as a File Provider that runs in a separate partition.

In this partition, cryptographic keys are managed, this covers part a) of CR-NF-018.

Also, the calculation of cryptographic functions is performed in this partition, this covers part b) of CR-NF-018.

Cryptographic certificates are stored in this partition and are not accessible to other user partitions. The certificate can be updated using the Secure Update process with the update packet containing the new certificate. This covers part c) of CR-NF-018.

CR-NF-018 coverage on Android

CycurLIB is integrated into the app using JNI.

Cryptographic keys are managed in the Android app, which covers part a) of CR-NF-018.

The cryptographic calculations take place in CycurLIB (as part of the Android app), which covers part b) of CR-NF-018.

The management of certificates is handled in the Android app, externally from Cycurlib and using the infrastructure provided by Android, compliant to Java Cryptography Architecture (JCA). Cryptographic certificates are stored within Android.

CR-NF-019 coverage on PikeOS

Access rights/methods to the partition hosting the CycurLIB is configured statically by the system integrator who is the trusted entity. And this configuration cannot be changed during runtime. This covers part a) of CR-NF-019.

During runtime, PikeOS ensures that the access to File Providers in the CycurLIB partition is according to the security architecture configured statically by the system integrator. This covers part b) of CR-NF-019.

By allocating different partitions to different roles, users, services, and domains and using PikeOS separation mechanism to provide different access rights/methods to the CycurLIB partition's File Providers, part c) of CR-NF-019 can be fulfilled.

Overall access is handled using File Provider access rights. To have individual cryptographic functions with separate access controls, the functions can be realized as different virtual File Providers within the CycurLIB partition. This covers part d) of CR-NF-019.

Similarly, the usage rights of cryptographic objects/services can be mapped to access rights on the virtual File Providers, this covers part e) of CR-NF019.

CR-NF-019 coverage on Android

As a library, Cycurlib does not provide support for these capabilities.

Configuration is done in the development IDE (e.g. Android Studio).

Access and usage rights to cryptographic objects are handled by the Android app, which is entirely dependent of Android permissions infrastructure. If the security infrastructure is aligned to Android permissions e.g authorized entities are applications, then a) b) c) and d) are provided. If the security infrastructure is not aligned to Android permissions, then none is provided.

As there is limited settable usage rights to cryptographic objects in Android from `android.security.keystore` introduced in Android 6.0 and applying only to private and secret keys, support for e) is only partial at this time.

When the Android device is rooted, the user might be able to access cryptographic objects.

Safety and mixed-criticality requirements

Safety requirements are very partially covered by Android. However with Google Project Treble integration in Android Oreo (although in the security scope) the updates may be more easily feasible which could provide more compliance for CR-NF-026.

Almost no support for mixed-criticality is available. Proprietary hardware modifications made by some vendors inhibit almost completely the use of PMC in some chips.

4.3.4 Telecom Integrated Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android	TableID
S1-F-001	Linux/GNU based OS for the COTS	Needed for integration of thermal protection mechanisms	Yes. Linux partitions	Yes. Android is Linux-based	TIR01
S1-F-010	Hypervisor shall be able to execute Linux and other runtime environements	Covered by State of the art and SAFURE	PikeOS on smartphone KO	Not applicable. No hypervisor	TIR02

Table 5: Telecom integrated requirements

Although Android is based on Linux, integration of thermal protection mechanisms could not be done due to the absence of an API to support it in Android similarly to the API that exists in Linux.

4.3.5 Telecom Integrated Non-Functional Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
S1-NF-003	One of the HW platforms must include a COTS multi-core with at least 4 cores (e.g. Freescale i.MX6Q, Freescale P4080)	This requirement should be compatible with TRT ones on this case study. The HW platform chosen provides this feature, so this requirement is covered	Yes	Yes
S1-NF-004	The COTS multi-core in the previous requirement must include some on-chip shared resources across cores: at least (1) a shared interconnection network between the cores and a shared cache or shared memory, and (2) a shared memory controller. It is also valuable if such multi-core includes a cache memory shared across cores	This requirement should be compatible with TRT ones on this case study. The HW platform chosen provides this feature, so this requirement is covered	Yes	Yes
S1-NF-010	The device shall protect communications with the IMDs (Implantable Medical Devices) and with the medical cloud server in accordance with the SFPP security requirements	Communication with IMD devices : to ensure a compatibility with existing devices, security mechanism implemented in the Bluetooth protocol are used.	Partially OK through Hypervisor Network	Partially OK through Matrix rooms for communication with Cloud. Partially OK for Bluetooth LE.
S1-NF-019	The hardware platform shall offer multiple cores	All platforms selected by SAFURE are multicore.	OK	OK
S1-NF-021	The hardware platform shall offer an USB interface	All platforms selected by SAFURE have an USB interface	OK	OK
S1-NF-030	MPSoC (Mulit Core System on Chip)	Fundamental use-case requirement. Covered by the chosen hardware platform	Yes	Yes
S1-NF-031	One Temperature Sensor per Core	Required for integrating thermal protection mechanisms. Covered by the chosen hardware platform	No. One temperature sensor per SoC	No. One temperature sensor per SoC
S1-NF-032	The resolution of the Temperature Sensors needs to be equal/smaller than 1 K	Required for integrating thermal protection mechanisms. Covered by the chosen hardware platform	No. Unknown	No. Unknown
S1-NF-033	The System has to have power or thermal management build in.	Required for providing thermal protection Covered by the chosen hardware platform	Native on Xperia	Native on Xperia

Table 6: Telecom integrated non-functional requirements

These requirements are almost all HW-related and are covered in the same way for any software architecture. The requirement S1-NF-010 deals with security and protection profile. No such compliance exists for Android so it has to be covered at least partially by the middleware related to the IMD device.

4.3.6 Telecom Functional Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
S1-F-002	The functional architecture of the telecommunications use case(s) should be defined (at least in part) by means of a formal (possibly standard and commercial) modeling language		XML/MARTE	XML/MARTE
S1-F-003	The device shall provide applications to control and monitor the IMDs. This application shall be configurable by authenticated user only	An application will be developped to monitor/control a medical device or a simulated device. It will depend on the availability of a device using open communication protocols and providing an API/SDK to access the sensor streams. -> To be developped	OK	OK
S1-F-004	The device shall be able to forward data recorded or process in the critical environment to a cloud server. This requirement implies the existence of inter-partition communication means.	An application will be developped to transmit the data from the critical partition to a cloud server. -> to be developed	OK	Communication to Rooms
S1-F-005	The device shall allow the update of medical applications over the air. For example the update could be stored on a cloud server.	An android market(not the Google Play market) will be used to store the application. An OSS such as Fdroid could be used to create our own repository containing the application. -> to be developed	OK	OK
S1-F-006	The device shall provide the Android operating system with all basic applications (browser, mail client, multimedia player, phone client etc).	It will depend on the features offered by the hypervisor and specifically the screen sharing between two Android partitions. In this case, the non-critical partition will contain basic applications. -> Feature provided by a partner and configuration to be made	OK for the device. KO for PikeOS	OK
S1-F-007	The device shall provide a mechanism to separate the domain specific applications (e.g. IMD applications) from the general purpose application or prohibit installation of those general purpose applications by users.	The separation between the IMD applications is made by design. In fact, PikeOS used to separate between critical applications (IMD apps) and general purpose applications. -> By design	Yes	No
S1-F-008	A mechanism shall enforce authenticity and integrity of the software stack in accordance with the SFPP security requirements.		No	Android apps signing and/or CycurLib for integrity
S1-F-009	Remote control of the platform shall be available to legitimate users in accordance with the SFPP security requirements.	Control order of IMD devices are transmitted from the medical server over the specific VPN used to transmit medical data. After that, these data are sent to the IMD having actuators. -> -> To be developed	No	Matrix framework

Table 7: Telecom functional requirements

Requirements such as S1-F-003 need to be verified by inspection of application code and this requires access to source. However for vendor-applications controlling connected objects, the source code is not available and there is no way to ensure that no other party may have access to the data. Moreover the use of these applications is very often submitted to acceptance of terms and conditions that are quite long and difficult to understand and as a

consequence not read but accepted by the average user, who wants to use the device altogether.

Since these devices collect data that can be used for medical usage it is very important that there is a compliance of these licences which can be given by an independent entity whenever health related data can potentially be collected by the device or application provider. To mitigate this problem based on technical elements the use of a framework such as Matrix, with open protocols and open-source codebase, and independent from any device vendor or platform (Android or iOS) maker shall be mandatory for all data having potential medical or health purpose. This enables the user to have control and confidence over its data by choosing devices and applications that are approved by an independent entity.

Furthermore the use of this framework enables to have interoperating capabilities for applications which allows to break the vertical device vendor scheme where data is firstly controlled by the device vendor and then by the user. Interoperating capabilities allow third parties to have access to the data provided, that they comply to the access requirements enforced by the framework, independently of the device vendor. For the data, which indeed belongs to the user, this proposed scheme is user-centric and service-oriented whereas the actual scheme is vendor-centric and business-oriented, leaving the user with a view of its own data under the sole control of the device vendor.

4.3.7 Telecom Non-Functional Requirements

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
S1-NF-001	The critical environment containing medical application shall implement a RTOS enforcing the security policy regarding real-time communication needs.	An Android partition is used as a critical environment. Security policy is ensured by design by using an hypervisor(separation kernel) and by using Android permissions.	Verified with PikeOS	no critical environment for medical application. Verification through third party app to monitor qualitatively RT of medical apps. Loss of predictability. Observation only. WebRTC (seamless video flux property)
S1-NF-002	The operating system running on the PikeOS hypervisor should be kept as minimalistic as possible, allowing direct access on the hardware close to the bare bone style. Complex unpredictable scheduler policies such as the ones included in Linux systems should be avoided for safety critical systems, especially those with time-critical requirements	Requirements for controlling interferences on time critical systems	Yes for native partitions	No
S1-NF-005	Performance monitoring counters (PMCs) must be abundant and allow tracking activities occurring in the on-chip shared resources such as the number (and preferably also the type) of access to the on-chip interconnection network and the memory controller indicated in S1-NF-004	This requirement should be compatible with TRT ones on this case study.	No for smartphone	No for smartphone
S1-NF-006	The device temperature shall remain under 45°. In particular, this shall be the case when the Android environment is being intensively used.	For now, no partner from the WP4 committed to implement this functionality in PikeOS scheduler.	Native T sensor	Native T sensor
S1-NF-007	Different application modes of the devices for low, medium and high computational effort	Enables sophisticated thermal protection mechanisms	Not addressed	Not addressed
S1-NF-008	Different applications with different thermal characteristics for each core	Enables sophisticated thermal protection mechanisms	Not addressed	Not addressed
S1-NF-009	The applications have to be periodic	Required for providing thermal protection	Not applicable	Not applicable
S1-NF-011	The device shall protect in confidentiality and authenticity critical data in accordance with the SFPP security requirements. In particular application data shall be protected in confidentiality, integrity, authenticity and availability.	These properties are ensured by using security mechanisms provided by Android(Cipher class) or CycurLIB with PikeOS.	CycurLIB on PikeOS	CycurLIB / Matrix on Android
S1-NF-012	Access to the device, and especially access to the critical environment shall be granted only after a correct authentication of the user in accordance with the SFPP security requirements.	Android authentification mechanism(local or authentificating server) will be used	Verified	Not verifiable, no explicit critical environment
S1-NF-013	The device shall implement a separation kernel with at least one partition for non-critical applications and one partition for critical applications in accordance with the SFPP security requirements.	An hypervisor compatible with the hardware platform is to separate the 2 envrionments -> Ensured by Design(Hypervisor and architecture supporting the device)	Verified	No partition, no explicit critical environment

Table 8: Telecom non-functional requirements, part 1

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
S1-NF-014	The telecommunications use case should provide one example of communication or interaction with security concerns/issues that can be expressed in a quantitative and formal way.		Example to be defined	Example to be defined
S1-NF-036	The device shall protect the anonymity and the confidentiality of the medical data transmitted to the medical staff	A specific VPN will be used to transmit only the medical data between the terminal device to a medical server.-> To be developped	?	Matrix framework
S1-NF-037	The device shall protect the privacy, the anonymity and the confidentiality of the data transmitted to the support product staff(manufacturer, seller of the product ...)	A specific VPN will be used to transmit only the data concering IMD devices. The VPN will be used between the terminal device and a server used by the support product staff. These data will be used by the support team to ensure the correct functionning of the IMD devices. -> To be developped	?	Matrix framework
S1-NF-038	Anonymity: A subset of the medical data shall be provided to authorized users, without any information that may reveal the identity of the IMD holder		?	Matrix framework
S1-NF-039	Privacy: The device shall be able to ensure that a subset of the data is accessible only to the terminal holder and to other users to whom the terminal holder has granted access		?	Matrix framework
S1-NF-015	Application Should run Critical (medical application) and non-Critical application (mail/social network/game/...) at the same time on the same system	Fundamental usecase requirement	Demonstration with SYMTA/S	Demonstration with SYMTA/S
S1-NF-016	The hardware platform shall be able to run Android above pikeOS. Preferably the latest version of Android : Android 5.0 a.k.a Lollipop	This will be ensured by using the work made by SYSGO. The Android OS will be used as a partition in PikeOS -> The Android personnality will be be provided by a partner	Yes	Yes but No PikeOS
S1-NF-017	The hardware platform shall be able to run the separation kernel PikeOS	The platform selected by TCS for the telecommunication use case will be supported by PikeOS. -> SYSGO will provide an installation with a PS Provided by a partner	Yes	No
S1-NF-018	The hardware platform shall be able to run Linux OS above pikeOS	SYSGO will provide a Linux running PikeOS for the telecommunication platform-> Provided by a partner	Yes	No
S1-NF-020	The hardware platform shall offer a GPU addressed by at least one partition	Either PikeOS provide a direct access to the GPU of the platform or provides a specific driver to have an access from multiple partitions to the GPU(indirectly).	OK	OK

Table 9: Telecom non-functional requirements, part 2

ID	Requirement Description	Notes	Architecture with PikeOS	Architecture with Android
S1-NF-022	The hardware platform may offer an SDHC interface		OK	OK
S1-NF-023	The hardware shall offer a 3G/4G interface	All smartphones and some tablets have a 3G/4G interface	OK	OK
S1-NF-024	The hardware shall offer a Wi-Fi interface in order to communicate with the cloud server.	The chosen platform provides a WIFI interface	OK	OK
S1-NF-025	Documentation about the hardware platform shall be available and detailed enough to design a BSP.	For now, we don't have enough information about the SAFURE platforms and SoC.	No	No
S1-NF-026	The hardware platform shall allow to configure the boot loader.	Some manufacturers such as SONY allow us to configure the bootloader.	No	No
S1-NF-027	The hardware platform shall be preferably a smartphone, a tablet, or a development tablet (in this order)	For now, the choice of a smartphone is still relevant	OK	OK
S1-NF-028	The underlying hardware shall provide an hardware virtualization mechanisms set	For now, the choice of a smartphone is compliant with hardware virtualization mechanisms.	No	No
S1-NF-029	The underlying hardware may provide an NFC interface	For now, the choice of a smartphone provides an NFC interface -> Choice of the platform	No	No
S1-NF-034	Minimum one power sensor for the MPSoC	Low priority. This requirement would enable the study of power covert channels	No power sensor. Battery level sensor	No power sensor. Battery level sensor
S1-NF-035	The number of the applications has to be limited	Required for providing thermal protection	Yes for native partitions	No

Table 10: Telecom non-functional requirements, part 3

In general the telecom non-functional requirements are covered very partially, and mostly not by Android. Partial coverage is brought by the middleware such as Matrix. As stated before the combination of PikeOS and middleware features would bring a much better coverage of these requirements.

4.4 Applications integration process verification

Integrating applications under Android has been done using the Android Studio tool. Other tools, with a strong technical interest, such as cross-platform development tools (for Android and iOS), have been considered but not used due to their proprietary character and license cost.

The Android Studio tool is very easy to use and well suited for Android development. However it requires to have a connection to internet due to dependencies resolution that can be made at almost any part of the production and execution process

From an industrial perspective it is hence very difficult to ensure that

- all components needed for production are available locally
- dependency check will not require to fetch components from Internet should a single component be marked as potentially obsolete by the build or execution system

As a matter of facts we have first tried to maintain Android studio disconnected from Internet and we have incorporated elements required on a manual basis by duplicating them from a shadow Android Studio connected to internet. The list of elements needed is difficult to

establish and regularly the unconnected build system would block for dependency miss. Maybe this is related to our relative inexperience with the Android Studio tool, however it is not the default use of this tool and we found only little documentation for our unconnected use.

In a second step, and to ease development we switched to connected use and the previous problems disappeared.

Android Studio releases are quite numerous over time. The integration started with Android Studio 2.1.1 and is now using Android Studio 3.1. Not all intermediate releases have been used. Release update was no problem provided the permanent connection to internet.

For long-term maintenance of applications and middleware, this raises three main problems:

1. the prefetching and build process of applications, with components whose availability over time is not warranted, shall be evaluated
2. the update cycle of Android itself adds some obsolescence to these applications that have to follow since most recent smartphones only support the most recent Android versions
3. the Android update process which allows to update an application but does not allow to revert to the previous version, should the update bring unexpected problems on a specific smartphone

4.5 Integration of application-independent components to Android

As the initial plan for the demonstrator was to integrate Cycurlib, it has been completed by another component for low-throughput video streaming in order to validate the overall process over a wider functional range covering limited bandwidth availability.

4.5.1 CycurLIB port to Android

The Cycurlib port to Android has provided the following results:

- Performance check for cryptographic algorithms (e.g. AES, SHA-2, ECDSA, EdDSA)
- Integration into Android apps via JNI
- Secure communication demo
- Encrypted with AES
- Integrity-protected with MACs
- Secure update demo
- Encrypted with AES

Figure 7: Cycurlib port to Android

Figure 8: Cycurlib secure communications demo

Since it had been successfully demonstrated on a DragonBoard 810, there was no major interest except for performance measurements to demonstrate it again on a commercial smartphone.

4.5.2 Gstreamer integration to Android

Following the scheme of integration of Cycurlib, the Gstreamer software has been integrated to Android along with a very low-rate video encoder based on H264.

A sample application has been designed around the Gstreamer software using the same JNI technology that was used for Cycurlib.

As a result this application is able to provide a very low video rate under a constrained bandwidth by using only software resources on the smartphone.

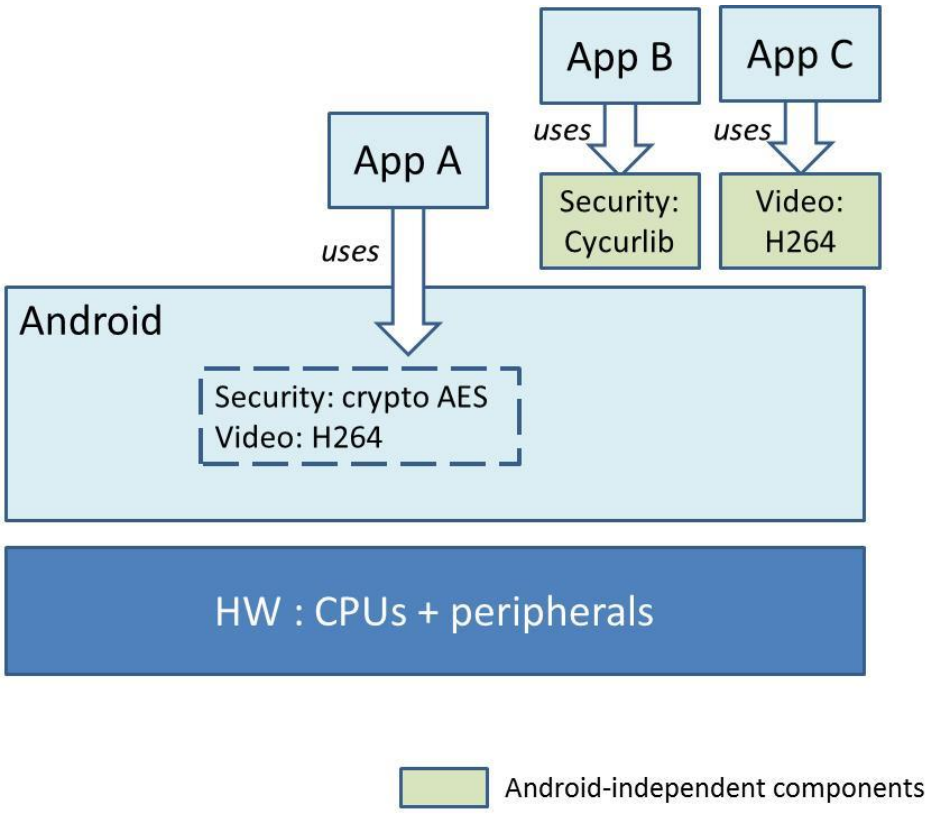

Figure 9: Integration of Android-independent components

The interest of Android-independent components is

- They can be updated independently from Android, for safety or security reasons
- They can be audited separately or within Android, since they have well-defined interfaces and contents

The problems raised by such components are*

- They must be adapted to be integrated to several Android versions
- They cannot be shared simply by several applications, especially for updates
- They rely on access rights provided by Android, which are application-level centric

Chapter 5 Summary and conclusion

This document has reviewed the evaluation of the telecommunications prototype. It has provided the test cases performed and the traceability of the SAFURE requirements as described in D1.2. This document has presented the aspects related to integrating the telecom prototype.

In conclusion, it has been shown that the integration using an evaluation board, as done in SAFURE WP4, and using a real product such as a smartphone, both assembled using individual components that are indeed very similar, may exhibit very different outcomes in the feasibility and in the SAFURE support of these platforms.

Despite the SoC capability to support hypervisor mode, no hypervisor support for current smartphones really exists presently. The initial approach in SAFURE which is to enable security and safety by design has been adapted by necessity to bringing safety and security by architecture.

The safety features can be introduced by adapting applications to a distributed safety infrastructure which on the smartphone does the monitoring of the corresponding applications. However there is presently no way to make sure these safety applications cannot be starved from resources since Android does not support a-priori resource allocation and arbitration for applications.

The security features can be introduced by connecting the corresponding SAFURE applications through an open middleware such as Matrix which provides only open protocols. This allows certification by independent entities that the user remains in control over its own data.

Chapter 6 List of Abbreviations

Abbreviation	Explanation
API	Application Programming Interface
AUTOSAR	AUTomotive Open System ARchitecture
BSP	Board Support Package
COTS	Component Off The Shelf
GNU	Gnu is Not Unix
GPU	Graphics Processing Unit
HW	Hardware
IMD	Implantable Medical Device
I/O	Input/Output
LE	Low Energy
MAC	Medium Access Control
MARTE	Modeling and Analysis of Real-Time Embedded Systems
NFC	Near Field Communications
OS	Operating System
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
OSS	Open Source Software
PAMU	Peripheral Access Management Unit
PKI	Public Key Infrastructure
PMC	Performance Monitoring Counter
PMIC	Power Management Integrated Circuit
PMU	Power Management Unit
QoS	Quality of Service
RT	Real Time
RTOS	Real Time Operating System
SDHC	Secure Digital High Capacity
SDK	Software Development Kit
SFPP	Security Framework Protection Profile
SoC	System on Chip
SRAM	Static Random Access Memory
SW	Software
TCS	Thales Communications and Security
TRT	Thales Research and Technology
UC	Unit of Computing

Abbreviation	Explanation
USB	Universal Serial Bus
VPN	Virtual Private Network
WCET	Worst-Case Execution Time
WiFI	Wireless Fidelity
WPA	WiFi Protected Access
XML	eXtended Markup Language

Table 11: List of Abbreviations