

**<Project Name>**

SW Architecture Design

<Feature name>

|  |  |
| --- | --- |
| **Document Code** | **172e-BM/DE/HDCV/FSOFT** |
| **Version** | **1.0** |
| **Effective Date** | **22-May-2018** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **About This Template**   * This template is for writing the Software Architecture Design (SAD) used in FPT Software  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Version | Date | Comment | Author | | | 1/0 | 22/05/2018 | Initial Create | FPT Software |  | |

|  |
| --- |
| **Read me first**   * This template is a basic format for a software architecture design, and it consists of guidelines (green box) and examples. * You should read this general guideline before writing this document. * After completion of this document, this guideline (green box) should be deleted. * The content of this template might not be 100% correct for all projects, so depending on project size and characteristic, it might be tailoring for some content. * The definitions of SW Element (or SW Module), SW Component, and SW Unit used in this document are as follows.   + SW Element: SW can be divided into several features, and the feature unit is defined as element.   + SW Component: It is the lowest level element and this is the unit of SDD (SW Detailed Design).   + SW Unit: It is the smallest unit of SW component that is no longer divided. |

|  |
| --- |
| **General Guidelines!! This guideline should be read before writing this doc.**   1. **Basic Rule on SAD Writing**  * This SAD should identify the one or more SW components that should be specified detailed design in the SDD * The scenarios of SW non-functional requirements should be specified to be able to measure quantitatively and it should be reflected in the design document * There is no restriction on the Design Tool (Microsoft Vision, Astah, EA)  1. **Rules for writing SAD Documentation (source: Documenting Software Architectures : Views and Beyond, Second Edition**) 2. Write documentation from the point of view of the reader, not the writer. 3. Avoid unnecessary repetition of information in the document. 4. Avoid ambiguity. Always explain your notations in the diagram. 5. Fill in the part of not yet decided as “to be determined” instead of blank. 6. Record the evidences of adopting this design. It is useful to review or change the design in the future. 7. Do not reflect the design changes immediately in the document during development. Instead reflect them according to the version control and release plan. 8. Request review to the target audience. 9. **References**   This template has been revised with reference of below standard documents and professional materials related to the software architecture and existing SAD template and guideline documents of FPT.   * Automotive\_SPICE\_PAM\_v3.0 (SWE.2 Software Architecture Design) * Software Architecture in Practice, Third Edition * Documenting Software Architectures: Views and Beyond, Second Edition * SAD templates and guidelines already depveloped by FPT Software  1. **Document Format**   The following basic rules should be observed.   * Text font: Arial * Attach captions and titles for all figures and tables. * Insert caption automatically: In the Microsoft Word Program, [Reference] > [Caption Insertion] > [Level] > Select Figure or Table > Select OK * Caption position: Below the picture and above the table   e.g.) Figure 1. Picture title, Table 1. Table Title   1. **Document Versioning, File Naming Rule**   Refer to the Configuration Management Guidelines (CM Plan) maintained within the project. |

About This Document

Document Information

|  |  |
| --- | --- |
| Issuing authority | Enter Document Owner |
| Configuration ID | Enter CI ID in CMBook |
| **Status of document** | In Progress / Approved / Released |

Revision History

|  |
| --- |
| The document history is organized in the timing order that the first history is at the top and the most recent history is at the bottom |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Verion | Date | Comment | Author | Approver |
| 0.1 | 2017-08-18 | Initial draft |  |  |
| 0.2 | 2017-08-30 | Design changed based on review with. |  |  |
| 1.0 | 2017-09-20 | Initial Release version 1.0 |  |  |

Purpose

|  |
| --- |
| Describe the purpose of this document. |

This document specifies the software architecture design for XXX to support Software Requirement. This design document also serves as a guideline on how each software components in the system should be implemented and how the components should interact with each other.

Scope

|  |
| --- |
| Describe the scope of this document. |

* Architectural Drivers
* SW Architectural Representations
* Resource Consumption Objectives
* Interface Design
* Interation Design
* Architecture Alternatives

Audience

|  |
| --- |
| Describe the readers of this document. |

This document is targeted to the following audiences:

* Vehicle OEM
* Hardware Architect
* Hardware Engineer
* Hardware Tester
* Software Architect
* Software Developer
* Software Tester

Related Documents

|  |
| --- |
| Describe the relevant documents with this document.  Write the document name. If there’re multiple versions, write the version numbers as well. |

Documents related to this document include:

* SRS (Software Requirement Specification)
* GIS-724 Call Request v1.0
* GIS-723 Feature Precedence v1.0

Conventions

Indicate what you want to emphasize or what you can help.

NOTE

A note will be used to emphasize important information or to provide a reference to the related information

Caution

A caution will be used to alert the reader to pay careful attention to certain information provided in this document, which if neglected, can potentially causes issues

Acronyms / Glossary

|  |
| --- |
| Describe the explanation of acronyms used in this document. Write the acronyms in alphabetical order. In addition to the abbreviation explanation, if there’re terms need to be explained, it could be added. |

|  |  |
| --- | --- |
| Acronym | Description |
| C&C View | Component-and-Connector View |
| IHU | IVI Head Unit |
| SAD | Software Architecture Design |
| SDD | Software Detailed Design |

Table of Contents

|  |
| --- |
| The table of contents should be generated automatically. Click the right mouse button on the table below and select [Update Field] menu. This will automatically reflect the changed table of contents. |

[1 Architectural Drivers 9](#_Toc144110008)

[1.1 SW Main Features 9](#_Toc144110009)

[1.2 Quality Attributes 10](#_Toc144110010)

[1.3 Constraints 11](#_Toc144110011)

[1.3.1 Business Constraints 11](#_Toc144110012)

[1.3.2 Technical Constraints 11](#_Toc144110013)

[1.4 Change compare between different of model series 11](#_Toc144110014)

[2 SW Architectural Representations 12](#_Toc144110015)

[2.1 Static View 12](#_Toc144110016)

[2.1.1 SW Component Descriptions 12](#_Toc144110017)

[2.2 Dynamic View 15](#_Toc144110018)

[2.2.1 Task Design 16](#_Toc144110019)

[2.2.1.1 Task Structure 16](#_Toc144110020)

[2.2.1.2 Task Scheduling 17](#_Toc144110021)

[2.3 Deployment View 17](#_Toc144110022)

[3 Resource Consumption Objectives 19](#_Toc144110023)

[4 Interface Design 20](#_Toc144110024)

[4.1 SW Component Interface Table 20](#_Toc144110025)

[5 Interaction Design 21](#_Toc144110026)

[5.1 Boot Startup Management 21](#_Toc144110027)

[5.2 Application Launch 22](#_Toc144110028)

[5.3 Failure Handling 22](#_Toc144110029)

[5.3.1 Factory reset 22](#_Toc144110030)

[5.4 Sequence XXX 23](#_Toc144110031)

[6 Architectural Alternatives 24](#_Toc144110032)

[6.1 Architectural Design 1 [e.g. Native implementation of XXX core function] 24](#_Toc144110033)

[6.2 Architectural Design 2 24](#_Toc144110034)

Figures

|  |
| --- |
| The table of figures should be generated automatically. Click on the table below, right-click and select [Update Field] menu, the changed table of contents is automatically reflected. |

[Figure 1 Module View 12](#_Toc490825393)

[Figure 2 C&C View 15](#_Toc490825394)

[Figure 3 C&C View for Task Scheduling 17](#_Toc490825395)

[Figure 4 Deployment View 18](#_Toc490825396)

Tables

|  |
| --- |
| The table of tables should be generated automatically. Click the right mouse button on the table below and select [Update Field] menu. This will automatically reflect the changed table of contents. |

[Table 1 Software Main Features 1 9](#_Toc491004387)

[Table 2 Software Main Features 2 10](#_Toc491004388)

[Table 3 Quality Attribute Requirements 10](#_Toc491004389)

[Table 4 Business Constraints 11](#_Toc491004390)

[Table 5 Technical Constraints 11](#_Toc491004391)

[Table 6 Difference of model series 11](#_Toc491004392)

[Table 7 SW Component Descriptions 13](#_Toc491004393)

[Table 8 Task Structure 16](#_Toc491004394)

[Table 9 Resource Comsumption Objectives 19](#_Toc491004395)

[Table 10 SW Component Interface Table 20](#_Toc491004396)

[Table 11 Concerns for Design Alternatives 24](#_Toc491004397)

# Architectural Drivers

|  |
| --- |
| This chapter consists of the requirements which influence in the overall SW configuration or performance and quality. In this chapter, describe the SW main features, quality attributes and constraints based on the SW requirements. |

## SW Main Features

|  |
| --- |
| * (Mandatory) Describe the major functions as text or table format. * (Optional) You can copy from SRS because it is important to have consistent with SRS. |

The IHU\_MAIN module is decomposed into three major components; AppManager and AppLauncher, and each of them is decomposed into sub components as shown in below figure

Figure : Component Diagram for IHU\_MAIN feture



Table 1 Software Main Features 1

|  |  |  |  |
| --- | --- | --- | --- |
| Level 1 | Level 2 | Level 3 | Descriptions |
| IHU\_MAIN | Application Manager | Application Lifecycle Management | - Operates application such as resuming app, pausing app and launching app.  - Manages application state: applications can have states such as launched, paused, etc. |
| Application Information | … |
| … | … |
| Application Launcher | Boot Startup Management | - Decides the boot startup sequence based on various factors including the last used application.  - Delivers the LUC information to Applicatio when starting the Application |
| Application Monitor | … |
| … | … |
| AppCommon API | Client API | - Registers application for communication  - Event Notification to and from applications |

Table 2 Software Main Features 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Level 1 | Level 2 | Level 3 | Level 4 | Descriptions |
| Multimedia | Audio Player | Disc Audio Player |  | Plays music that is stored in disc (CD, MP3, DVD). |
| USB Audio Player |  | Plays music that is stored in USB. |
| Telematics | Telephony | Call | Voice Call Request | Requests voice call connection to the given number. |
| Voice Call Answer | Answers the incoming voice call. |
| … | … |
| Remote Vehicle Control | Remote Door Lock | Remote door lock. |
| … | … |

## Quality Attributes

|  |
| --- |
| * (Mandatory) Create scenarios about the quality requirements specified in the SRS. * (Mandatory) Scenarios should be drawn up to be measured quantitatively. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario #** | Scenario | Quality Attribute | SRS ID | Priority |
| Scenario number: | Write this to measure the scenario for quality attributes quantitatively. If the requirements described in the SRS are written quantitatively, it could be copied | Fill in the relevant quality attributes. | SRS ID | Record the priority written in the SRS. |

Table 3 Quality Attribute Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scenario # | Scenario | Quality Attribute | SRS ID | Priority |
| 1 | The IVS shall generate an eCall within 1 second after the confirmed triggering signal is received. | Performance | QA1\_1 | High |
| 2 | … | Security |  | … |
| 3 | … | Avaibility |  | … |
| 4 | … | … |  | … |

## Constraints

|  |
| --- |
| (Mandatory) Describe constraints in terms of business and technique. |

### Business Constraints

|  |  |  |
| --- | --- | --- |
| **ID** | Business Constraint | SRS ID |
| Business constraint ID | Describe constraints in terms of business. It can be copied from constraints of SRS. | SRS ID |

Table 4 Business Constraints

|  |  |  |
| --- | --- | --- |
| ID | Business Constraint | SRS ID |
| BC1 | Deploy the architecture system in 10 weeks. |  |
| BC2 | ... |  |

### Technical Constraints

|  |  |  |
| --- | --- | --- |
| **ID** | Technical Constraint | SRS ID |
| Technical constraint ID | Describe constraints in terms of technique. It can be copied from constraints of SRS. | SRS ID |

Table 5 Technical Constraints

|  |  |  |
| --- | --- | --- |
| ID | Scenario | SRS ID |
| TC1 | The system consists of a head office server located at the head office, and the POS terminals placed at store cashiers |  |
| TC2 | ... |  |

## Change compare between different of model series

|  |
| --- |
| Describes how variants for different model series or configurations are derived. |

Table 6 Difference of model series

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Features** | **MY18** | **MY19** | **MY20** | **SRS ID** | **Description** |
| Audio Player. Audio Volume Level for audio modes | O | X | O | UIS10456  UIS10457 | Set audio volume parameter is disable in case MY19 model |
| Language | English, French, Spanish | English, Mandarin | English, Arabic, German, French, Italian, Danish, Portuguese, Dutch, Polish | UIS4598 | Change languare setting in configuration file |
|  |  |  |  |  |  |

# SW Architectural Representations

|  |
| --- |
| * Define the SW architecture as types of Module view, C&C view and Deployment view in the various stakeholder perspectives. * The main purpose of the SW architectural representation is to identify the SW components. |

## Static View

|  |
| --- |
| * (Mandatory) Define the SW architecture as a module view type. * (Mandatory) In the [2.1.1 SW Component Descriptions](#_SW_Component_Descriptions) chapter, the components identified in the below are explained. |

This is the software architectural design of OOO. The IHU\_MAIN module is decomposed into three major components; AppManager, AppLauncher and AppCommon, and each of them is decomposed into sub components as shown in below figure,



Figure 1 Module View

### SW Component Descriptions

|  |
| --- |
| * Describe the components identified in the [2.1 Static View](#_Module_View) chapter. The SRS ID for each component must be filled in. * According to the SW feature and size, the features might not be classified. * When there’re thousands of requirements, the traceability with requirements might be managed easily by separate excel file. * (Optional) If there’re reusable or purchasable components, mark them by adding separate rows. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Feature** | SW Component ID | **SW Component Name** | **Category** | **Description** | **SRS ID** |
| Feature name | Component ID | Component name | Category | Component description | SRS ID |

Below table describes the software components which are speicified in the chapter 2.1.

Table 7 SW Component Descriptions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature | SW Component ID | SW Component Name | Category | Description | SRS ID |
| *AppManager* | *IHU\_MAIN-SWC-001* | *AppManager Controller* | *D(L)* | *Provides the lifecycle management such as resume, pausing Application.*  *The main responsibilities include:*   * *Acts as façade controller for the AppManager component* * *Register Application to AppManager* * *Retrieve Application Information* * *GoBack, GoHome* * *Set Application Foreground or Background* | *SWRS-IHU\_MAIN-128~133* |
| *IHU\_MAIN-SWC-002* | *Window Manager* | *D(L)* | *Interacts with Wayland server to show/hide the surface linked with a specific Application.*  *The main responsibilities include:*   * *Surface Registration & Deregistration* * *Reguest Foreground* | *SWRS-IHU\_MAIN-131, 132* |
| *IHU\_MAIN-SWC-003* | *Application* | *D(L)* | *Stores Application related information such as various configuration, application state.*  *The main responsibilities include:*   * *Activate, Deactivate, Launch, Pause, Resume, Stop Application* * *Store Application Configuration* * *Store Reason Event* | *SWRS-IHU\_MAIN-644, 645* |
| *IHU\_MAIN-SWC-004* | *History Management* | *D(L)* | *Manages the history related for application state, audio-video state. The previous state can be restored on a specific event.*  *The main responsibilities include:*   * *Store the reference for the current foreground Application* * *Retrieve Application that was previously shown as foregrounds* | *SWRS-IHU\_MAIN-647, 648* |
| *IHU\_MAIN-SWC-005* | *Command Scheduler* | *D(L)* | *Schedules when to execute commands and what to skip commands for better performance.*  *The main responsibilities include:*   * *Store commands for later execution* * *Scheduling commands whether to execute commands* * *Execute & skip Command* | *SWRS-IHU\_MAIN-652, 653* |
|  | *IHU\_MAIN-SWC-006* | *Event Handling* | *D(L)* | *Handles key event, power event, etc.*  *The main responsibilities include:*   * *Handle Key Event* * *Handle Power Event* * *Handle Response Event* | *SWRS-IHU\_MAIN-142, 143* |
| *IHU\_MAIN-SWC-007* | *Runtime Env. Support* | *D(L)* | *Provides functions to interact with another application platform such as Web Application Framework.*  *The main responsibilities include:*   * *Manages Web Applications* | *SWRS-IHU\_MAIN-145* |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Legend | | | |
| Category | Description | Category | Description |
| *D* | Newly Developed | (L) | The main developer is FPT |
| *R* | Reused without modifications  - Modification Rate on Code under 10% | (Open) | This component is based on open source. |
| *M* | Reused with modifications - Modification Rate on Code over 10% |  |  |

## Dynamic View

|  |
| --- |
| * (Mandatory) Define the SW architecture as a component & connector type and describe the connection relation and communication way. The component of C&C View means the basic operation elements or data storages performed in the system. The component types consists of client, server, filter, object, DB, etc. * (Mandatory) The interface identified in this chapter should be specified in the [4. Interface Design](#_Interface_Design) chapter. |

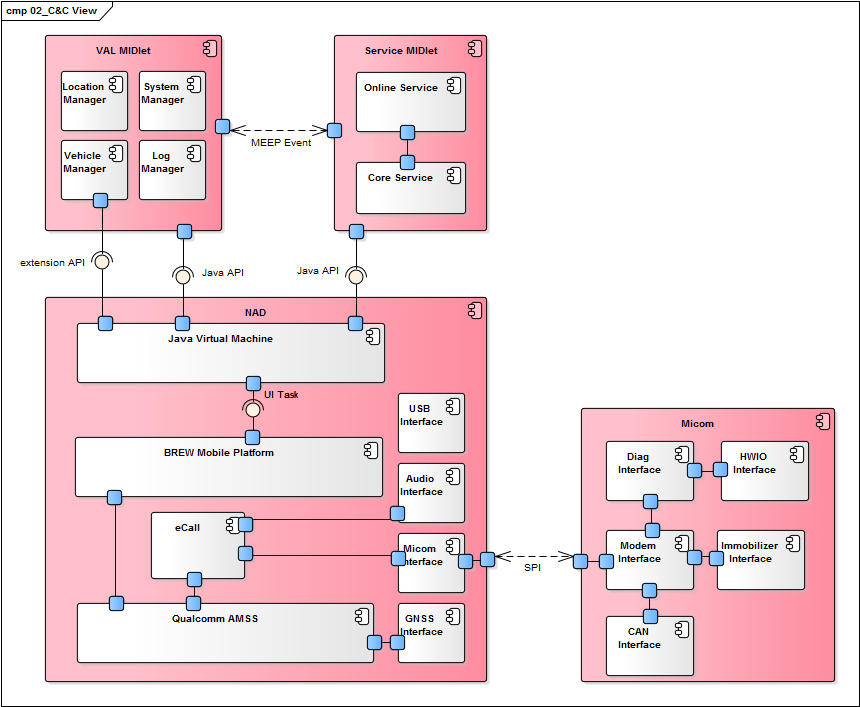


Figure 2 C&C View

The above diagram expresses the components that compose the OCU SW architecture and the connectors connect between each component...

1. Service MIDlet consists of the Java Online Service and the Java Core Service and use the standard Java API.
2. …
3. …
4. …

### Task Design

|  |
| --- |
| * (Mandatory) Define the task structure by identifying the tasks to be operated during run time, and describe the components and functions corresponding to each task. The task means run unit such as process or thread, and so on. * The task is designed to fit each environment because the run unit varies depending on each environment such as OS, programing language, etc. That is, the task type and the detail content described in constraints according to the platforms such as BSP base, OS base, AUTOSAR base, etc. |

This chapter describes the tasks that the IHU\_MAIN module requires. The task is the runtime unit that executes a job periodically or event-basis. The task is represented as process, thread, etc.

#### Task Structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task ID** | **Task Name** | **Description** | **SW Component Name** | **Constraints** |
| Feature name \_SWT\_XXX | Task name | Describe the task roles. | Component name | In general, the contents described in the constraints are as follows.  - Cycle Time  - Processing Time  - Memory Usage (Heap, Stack size)  Mostly the attribute items (e.g. priority, cycle time, memory size) when creating processes, threads, etc. in code or configuration files can be described in here. |

The task may have constrains that the maximum cycle time, etc. that needs to be described in the below table.

Table 8 Task Structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task ID | Task Name | Description | SW Component Name | Constraints |
| IHU\_MAIN\_SWT\_001 | AppManager Process | This task runs as a process, and receives requests for resuming and pausing Applications. | AppManager | Cycle time: n/a (request basis processing)  Processing time: Maximum startup time to start last audio/video source is 10 seconds. |
| IHU\_MAIN\_SWT\_001 | AppLauncher Process | … | … | ... |

#### Task Scheduling

|  |
| --- |
| Describe the result of task scheduling considering the execution time of identified tasks, order, period, etc. At that time, whether there’re no problems between tasks and it’s operated within the max processing time, etc. should be reviewed. |

When the IHU\_MAIN module is executed at the startup of IHU System, AppManager and AppLauncher are executed as separate processes. They load AppCommon library to communicate with other entities (e.g., AppManager and AppLauncher use AppCommon library to communicate).

AppManager manages the lifecycle of Applications such as resuming and pausing them. Application may be the native Application written with QT, or the Web Application. The runtime view for AppManageris shown in the following figure:



Figure 3 C&C View for Task Scheduling

## Deployment View

|  |
| --- |
| (Mandatory) Define the SW architecture as a deployment view format expressing the relations between SW and HW composing the system based on the C&C view defined in the [2.2 Dynamic View](#_C&C_View) chapter and describe the major considerations.. |

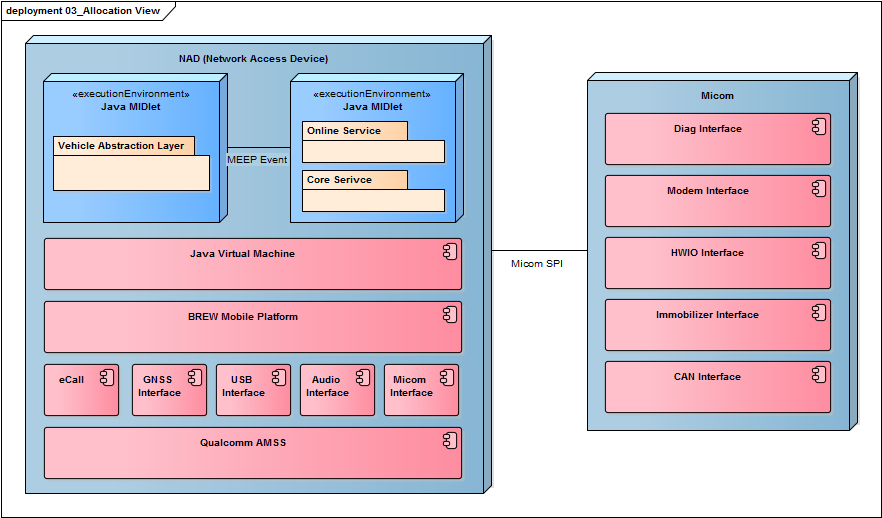


Figure 4 Deployment View

# Resource Consumption Objectives

|  |
| --- |
| (Mandatory) Estimate the necessary resources for CPU, runtime memory, ROM memory identified 2.1.1 chapter. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | SW Component Name | **CPU Load** | **CPU Load** | **Runtime Memory** | ROM Size | Remarks |
| Feature name | Component name | CPU load ratio for component (%) | CPU load ratio for element (%) | Runtime memory size for element | ROM size for element | Remarks |

Below table shows the estimated CPU load, runtime memory and rom memory for each software components.

Table 9 Resource Comsumption Objectives

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature | SW Component Name | CPU Load | CPU Load | Runtime Memory | ROM Size | Remarks |
| AppManager | AppManager Controller | 15% | 30% | OO MB | OO MB |  |
| … | … |  |
| … | … |  |
| AppLauncher | … | 3% | 40% | OO MB | OO MB |  |
| … | … |  |
| … | … |  |
| … | … |  |
| … | … |  |
| … | … |  |
| … | … |  |
| … | … |  |
| AppCommon | … | 10% | 70% | OO KB | OO KB |  |
| … | … |  |
| … | … |  |
| … | … |  |
| … | … |  |

# Interface Design

|  |
| --- |
| Describe the all interfaces identified [2.2 Dynamic View](#_C&C_View) chapter for each component in the below table. |

## SW Component Interface Table

|  |
| --- |
| (Mandatory) Define interfaces for each SW component. |

|  |  |  |  |
| --- | --- | --- | --- |
| **SW Component Name** | Interface Name | **Type** | **Parameters** |
| Component name | Interface name | Use one of the values below (can be added) - call: Local function call - dbus: Communication using dbus - LIN: | In / Out parameter description |

This section specifies the interface related to Application Launcher.

Table 10 SW Component Interface Table

|  |  |  |  |
| --- | --- | --- | --- |
| SW Component Name | Interface Name | Type | Parameters |
| Application Loader | MakeLaunchAppList | call | In:   * context data: LUC, country information, car variant   Out:   * Boot sequence list: list in which order the Application shall be started |
| … | … | … |
| … | … | … |
| AppProcess | launch | call | N/A |
| checkRestart | call | In:   * Restart count: the restart number until now |
| … | … | … |
| Logger | log | call | In:   * Data: data to be logged |

# Interaction Design

|  |
| --- |
| This chapter specifies the interactions between software components to fulfill the functions assigned to the OOO module.  Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.). |

## Boot Startup Management

When the IHU starts up, the Application Launcher decides the boot sequence of Applications. The sequence may depend on LUC, country variant, car variant, etc. After deciding the boot sequence, the App Launcher starts each Application accordingly.



* Step 1.0: MICOM sends the boot context data during the start up. It may include the LUC, country info, car variants, etc. Depending those information, the boot sequence may vary.
* Step 1.1: DBus Manager notifies the Application Loader for the MMUS event.
* Step 1.2: Application Loader builds the boot sequence by calling the ‘MakeLaunchAppList()’ API.
* Step 1.3: After building the boot sequence, the Application Loader starts the Application Manager at first. This is because that the Application Manager needs to run before any other Applications are running so as that the App Manager can communicate with the Applications.
* Step 1.4: Application Loader calls the launch() API of the AppProcess component. In this step, the AppProcess component represents the Application Manager.
* Step 1.5: AppProcess component uses the QProcess in the QT library to starts the Application Manager. For this, the location information is delivered to the QProcess.
* Step 1.6: The QT library QProcess starts the Application Manager.
* Step 1.7: After the successful starting of the Application Manager, the QProcess signals back the Application Loader to notify the event (the Application Manager started successfully).
* Step 1.8: The Application Loader starts the Applications by calling the LaunchApp() API.
* Step 1.9: The Application Loader calls the launch() API of the AppProcess for starting the Native Application. The AppProcess first placed in the list shall be started first.

## Application Launch

This procedure will be invoked when other Applicatoin requests launching Application via the App Manager.



* Step 1.0: App Manager requests launching a Native Application that is identified by the appId. AppManager makes that request by using the AppCommon library.
* Step 1.1: The Application Loader finds the AppProcess component that is requested by the App Manager.
* Step 1.2: Application Loader calls the launch() API of the AppProcess found in the step 1.1.
* Step 1.3: AppProcess requests to launch the Native Application by delegate it to the QProcess (QT library).
* Step 1.4: QProcess launches the Native Application requested at the Step 1.3.

## Failure Handling

### Factory reset

When the Application Manager is signaled for the factory reset, the Application Manager can request each Application to perform the factory reset. Once each Application finishes the factory reset, it signals back so as for the Application Manager to know whether the factory reset is complete.



* Step 1.0: The Setting Application or other entity requests the factory reset by calling the factoryReset() API of the App Manager Controller.
* Step 1.1: App Manager Controller creates a new command that represents the factory reset.
* Step 1.2: The App Manager Controller adds the newly created command to the Command Scheduler by calling the ‘addCommandAndExecute()’ API. This API immediately invokes the command and stores the executed commands.
* Step 1.3, 1.5: To execute the factory reset commands, the Command Scheduler signals each Native Application; hence each Application performs the factory reset. How to perform the factory reset for each Native Application is out-of-scope of this module.
* Step 1.4, 1.6: After the factory reset, each Native Application signals back the Command Scheduler to notify whether the factory reset was successful or not.

## Sequence XXX

# Architectural Alternatives

|  |
| --- |
| (Mandatory) Suggest the alternatives per the design decision, and record the evidence for selected designs. |

## Architectural Design 1 [e.g. Native implementation of XXX core function]

The following core functions of XXX need to be implemented using native because starting XXX should not be effected by any architecture related to JavaVM or abstraction. (From customer requirement ID XX, XX)

- …

- …

- ….

For this, design alternatives are considered with the following concerns:

Table 11 Concerns for Design Alternatives

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Concerns | Alternative #1:  using Java XXX | Alternative #2:  using native XXX |
| AD\_1 | Reduce time to market for future enhancements | **Yes** | No |
| AD\_2 | Reduce costs | **Yes** | No |
| AD\_3 | Reduce risks | No | **Yes** |
| AD\_4 | System qualities | No | **Yes** |
| AD\_5 | Reuse existing infrastructure | **Yes** | Unknown |
| AD\_6 | Use proven technologies | Unknown | **Yes** |
| AD\_7 | Performance | No | **Yes** |

In terms of reuse or maintenance, it is advantageous to use Java, however, the core function of XXX is XXXX. Therefore, we decided to implement XXX in native application.

## Architectural Design 2

-The End-