Table of Contents

1. Introduction

1.1 Purpose and Scope of the SAD (Software Architecture Document)

This document defines the high-level software architecture for the SmartDeviceLink (SDL) system. It describes the structure and the main components of the system, the project basis and dependencies. The goal of the document is to describe, in sufficient detail, the software components, their responsibilities, behavior, and interfaces. This document provides support for Luxoft, Ford and others to learn system design, limitations, stakeholders, and ways of extension and further development.

1.2 Definitions and Abbreviations

Abbreviations used in this document please find in the table below. For common abbreviations please refer to Standard Glossary [1].

Abbreviation	Expansion
BT	Bluetooth
GUI	Graphical User Interface
HMI	Human Machine Interface
IVI	In-Vehicle Infotainment
RPC	Remote Procedure Call
SDE	Software Development Environment
SDL	SmartDeviceLink
SEE	Software Engineering Environment
TTS	Text To Speech
VR	Voice Recognition

Definitions used in this document are in the table below. For common definitions please refer to Standard Glossary [1].

Definition	Description
Concern	A functional or non-functional requirement.
Model	A particular diagram or description constructed following the method defined in a viewpoint. These provide the specific description of the system, which can include identifiable subsystems and elements.

Stakeholder	An individual, group or organization that has at least one concern relating to the system.

1.3 Document Roadmap

The SW architecture of SDL Core is considered from different viewpoints. The short description of those covered in the current document please see in the table below.

Viewpoint	Viewpoint Description
Component	Functional type of view which describes the system's runtime functional elements and their responsibilities.
Component Interaction	Functional type of view which describes interactions of the system's functional elements. Component Interaction view uses component-level sequence or collaboration diagrams to show how specific components will interact. The purpose is to validate structural design via exploration of the software dynamics.
Use Case	Use Case View captures system functionality as it is seen by users. System behavior, that is what functionality it must provide, is documented in a use case model.
User Interface	Functional type of view which describes interfaces of the system's functional elements.
Data	Describes the way that the system stores, manipulates, manages, and distributes information. The ultimate purpose of virtually any computer system is to manipulate information in some form, and this viewpoint develops a complete but high-level view of static data structure and information flow. The objective of this analysis is to answer the questions around data content, structure, ownership, quality, consistency update latency, references, volumes, aging, retention, and migration.
Process State	Concurrency type of view. Process State View is used to model standard process dynamics that are independent of the loaded components. These dynamics may, for example, be part of a component management infrastructure that loads and controls components in the process. For process dynamics, it is often useful to think in terms of a standard set of states such as initializing, operating, and shutting down
Process	Concurrency type of view. Process View describes processes and process inter-communication mechanisms independent of physical hardware deployment

Development	Describes the architecture that supports the software development Process. This view addresses the specific concerns of the software developers and testers, namely code structure and dependencies, build and configuration management of deliverables, design constraints and patterns, and naming standards, etc. The importance of this view depends on the complexity of the system being built, whether it is configuring and scripting off-the-shelf software, writing a system from scratch, or something between these extremes.
Deployment	Describes the environment into which the system will be deployed and the dependencies that the system has on elements of it. This view captures the hardware environment that your system needs (primarily the processing nodes, network interconnections, and disk storage facilities required), the technical environment requirements for each element, and the mapping of the software elements to the runtime environment that will execute them.
Operational	Describes how the system will be operated, administered, and supported when it is running in its production environment. The aim is to identify system-wide strategies for addressing the operational concerns of the system's stakeholders and to identify solutions that address these
Logical	Logical view focuses on the functional needs of the system, emphasizing on the services that the system provides to the users. It is a set of conceptual models.

2 Case Background

2.1 System Context, Mission and Scope

SmartDeviceLink system is developed to serve as an intermediary between Ford`s vehicle Head Unit and an Application that runs at any of Mobile Devices:
- A Mobile Device might be connected via Bluetooth or Wi-Fi to the HU;
- The Application should be the SDL-enabled one.

The Mobile Device might be any of:
- Smartphone devices
- Tablet PCs

with operational system:
- iOs
- Android.

The SDL system allows Application to:
- Use vehicle HMI: VR, TTS, buttons (physical and touchscreen), vehicle display, audio system. etc.)
- Operate with Vehicle Data (seat belt position, transmission shift lever position, airbag status, etc.).

2.2 Product Stakeholders

Actors are stakeholders that interact with product directly.

Stakeholder Name	**Actor (Yes/No)**	**Concern**
Ford Company	No	Get the SDL system with enough quality and functionality that fulfill their goals
PM / Architect / Analyst	No	Use Customer Requirements Specification
Developers	Yes	Construct and deploy the system from specifications
Testers	No	Test the system to ensure that it is suitable for use

2.3 Business Goals

Luxoft delivered to Ford a prototype of POSIX compliant Applink Core in March, 2013. To support FORD goal of successful acceptance of Applink (new name is SmartDeviceLink) Core by open

source community of GENIVI consortium further enhancements will be required.

The purpose of the project is to develop component of SmartDeviceLink 2.0 Core with production quality according to automotive industry standard, adding new features required by Ford.

2.4 Significant Driving Requirements

The requirements are listed in the table below and ordered by descending of their significance from architectural solution point of view.

#	**Driving Requirement Description**
1.	System has to be POSIX-compliant to be easily ported on all POSIX standardized OSs.
2.	Transport for communication between Mobile Application and SDL system has to be added to system if necessary.
3.	APIs for communication between Mobile Application and SDL system described in appropriate documents have to be fully supported by the system.
4.	There has to be relatively easy way to port existing HMI Modules (such as UI, VR, TTS, etc.) to work with SDL system.
5.	APIS for communication between SDL system and HMI Modules have to be fully described in appropriate document and fully supported by SDL system.
6.	HMI Module in order to communicate properly with SDL system has to follow rules specified in appropriate documents (i.e. mentioned above API describing document).
7.	Mobile Application has to follow the rules specified in appropriate documents (i.e. mentioned above API describing document).in order to communicate with SDL system properly.
8.	Each request from Mobile Application, provided it is generated correctly according to appropriate protocol standard, has to be responded by SDL system

3 Solution Overview

The picture below shows the roadmap of the system development. For detailed description of workpackages WP1 - WP6 please refer to chapter "1.3.1. In scope" of PMP [2].

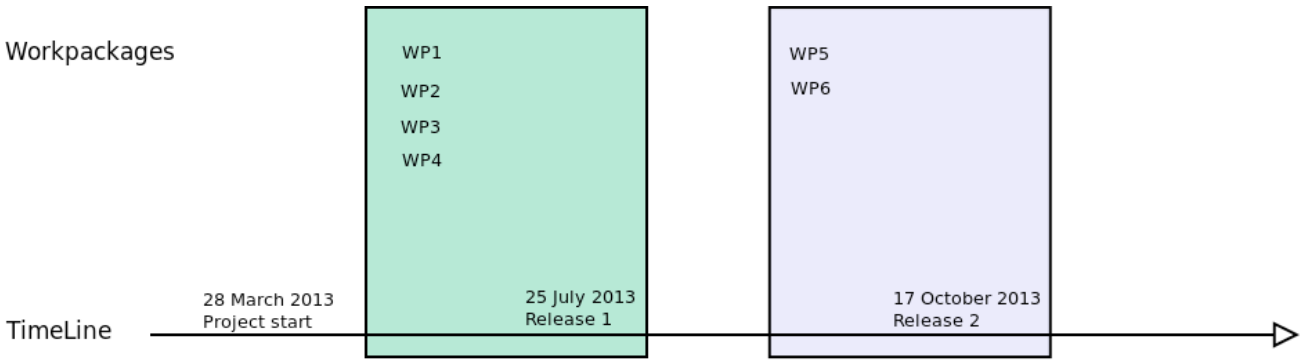

Figure 1 The Roadmap of System Development

4 Views

4.1 Components View

The view is represented by module and subsystem diagrams that show the system's export and import relationships.

The Components View diagram and its elements description please see below.

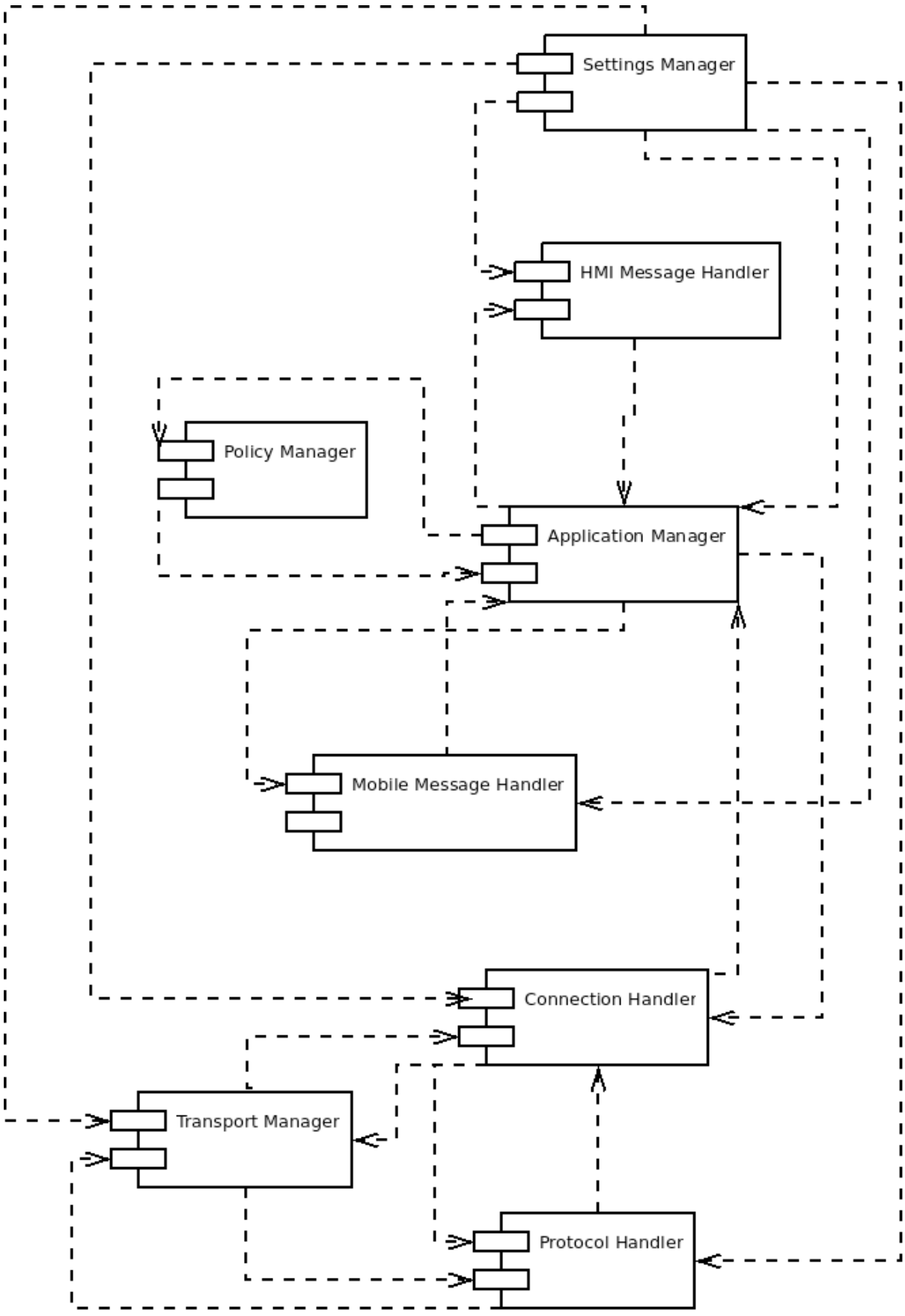

Figure 2: The Component View Diagram

Element Name	Element Description
Transport Manager	***Short Description:*** Component is used for handling low-level transport connections between SDL and Mobile Device. ***Responsibility is to:*** - Handle low-level message communication using different transport protocols. ***Relations:*** - ConnectionHandler - Protocol Handler - Settings Manager. ***Behavior:*** - Opens connection - Performs device discovery - Sends / receives messages. ***Constraints:*** For Bluetooth transport there are only 32 connections available
Protocol Handler	***Short Description:*** The component is used for: - Parsing low-level message structure - Notifying Connection Handler about starting/ending session - Setting protocol version of the received message. ***Responsibilities are to:*** - Translate messages from low-level format - Send formatted message to higher-level components - and vice-versa. ***Relations:*** - Transport Manager - Connection Handler - Mobile Message Handler

	- Settings Manager. ***Behavior:*** - Parses message header - Handles control messages - Handles single- and multi-frame messages. ***Constraints:*** Works with V.2 version of SDL protocol.
Connection Handler	***Short Description:*** The component is used for: - Handling one to one correspondence between: 1) Connections and sessions opened by Mobile Application and 2) Application representation in ApplicationManager. - Storing list of devices with device information (i.e. connections). ***Responsibilities are to:*** - Manage starting/ending of sessions; - Add/remove devices. ***Relations:*** - Transport Manager; - Protocol Handler; - Application Manager; - Settings Manager. ***Behavior:*** - Stores set of sessions for connection map; - Serves as a proxy for starting device discovery/updating device list. ***Constraints:*** Maximum number of sessions per one connection is 256.
Mobile Message Handler	***Short Description:*** The component is used for formatting messages to and from unified protocol-

	/API-independent format used by higher-level component. ***Responsibility is to:*** - Serve as proxy between two components that packs message to/from Protocol Handler format. ***Relations:*** - Protocol Handler - Application Manager - Settings Manager. ***Behavior:*** - Receives data from Protocol Handler - Parses and sends it to higher-level components - And vice-versa. ***Constraints:*** N/A
Application Manager	***Short Description:*** The component implements business logic of the SDL. ***Responsibility is to manage:*** - Applications - Relations between applications and messages - Permissions for messages. ***Relations:*** - Mobile Message Handler - Connection Handler - HMI Message Handler - Settings Manager. ***Behavior:*** - Creates internal representation of application. - Processes API calls both to/from Mobile side and HMI side.

	Constraints: 100Mb of ROM per Application.
PolicyManager	***Short Description:*** Component is used to: 1. enable advanced functionality and protect SDL APIs from unauthorized use 2. allow mobile applications to be granted elevated privileges, while providing usage data to a backend server 3. remotely manage SDL-enabled apps, including app-specific and device-specific access to system functionality. ***Responsibility is to:*** - Maintain applications permissions on the system ***Relations:*** - Application Manager ***Behavior:*** - Receives data from Application manager - Parses data - Stores in local storage - Provide data via Application Manager to mobile device and HMI - And vice-versa ***Constraints:*** N/A
HMI Message Handler	***Short Description:*** The component is used for: - Formatting message to and from unified protocol-/API-independent format used by higher-level component. - Providing adapters for different transport types between SDL and HMI. ***Responsibility is to:*** - Serve as proxy between two components that pack messages to/from HMI-specific format. ***Relations:***

	- Message Broker - Application Manager - Settings Manager. ***Behavior:*** - Receives data from HMI, - Parses data - Sends it to higher-level components - And vice-versa. ***Constraints:*** Processes messages received from only one HMI.
Settings Manager	***Short Description:*** The component is used for storing information about application configuration. ***Responsibility is to:*** - Store information about application configuration. ***Relations:*** - Transport Manager - Protocol Handler - Connection Handler - Mobile Message Handler - HMI Message Handler - Application Manager. ***Behavior:*** - Parses configuration file. ***Constraints:*** Only one configuration file is processed.

4.2 Component Interaction View

4.2.1 StartSession

The graphical diagram below describes the Starting Session Scenario.

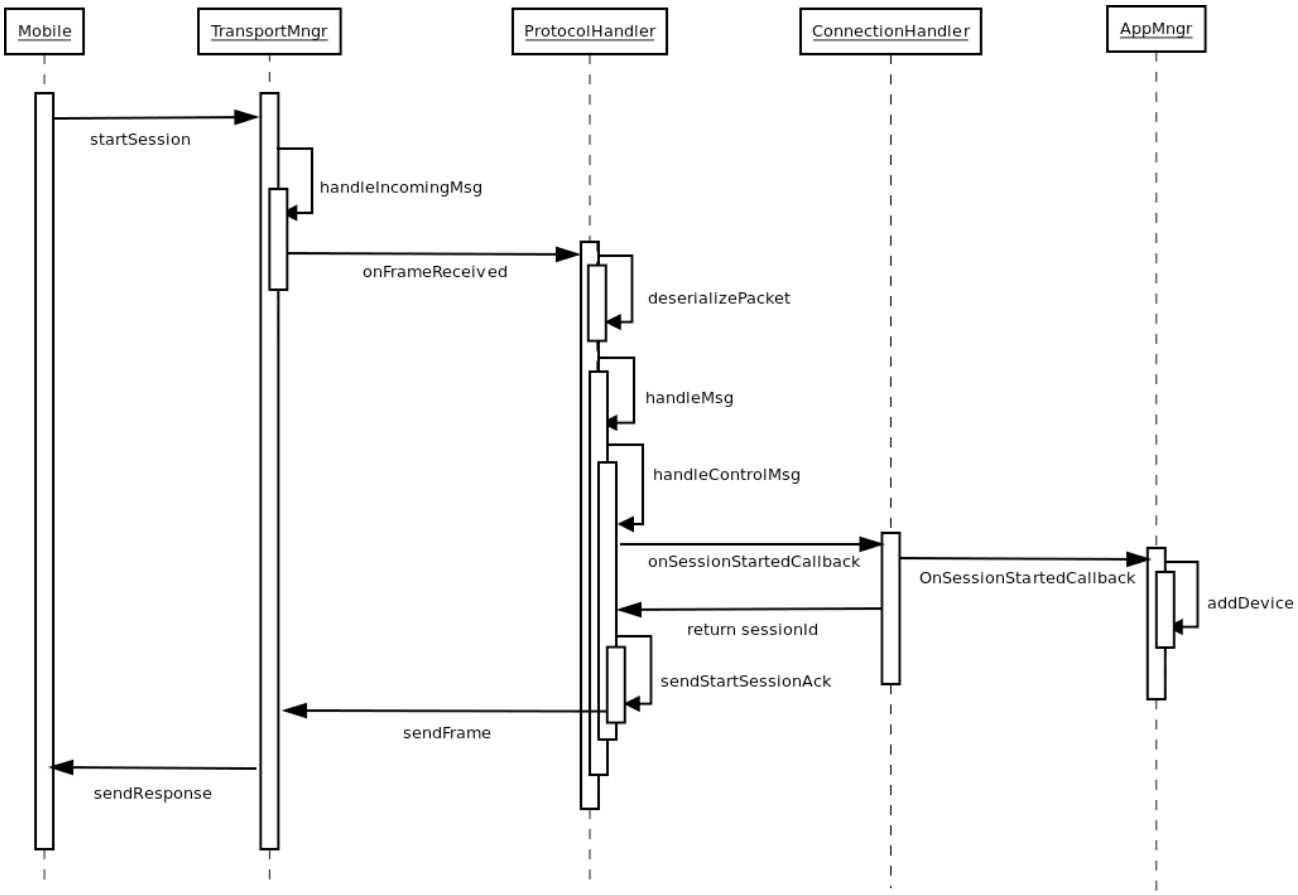

Figure 3: Graphical Diagram of Starting Session Scenario

Behavior:

A mobile application sends a request to establish a session.
Depending on resources, the system will or will not acknowledge the request to start session.

4.2.2 RegisterAppInterface

The graphical diagram below describes the Register Application Interface Scenario.

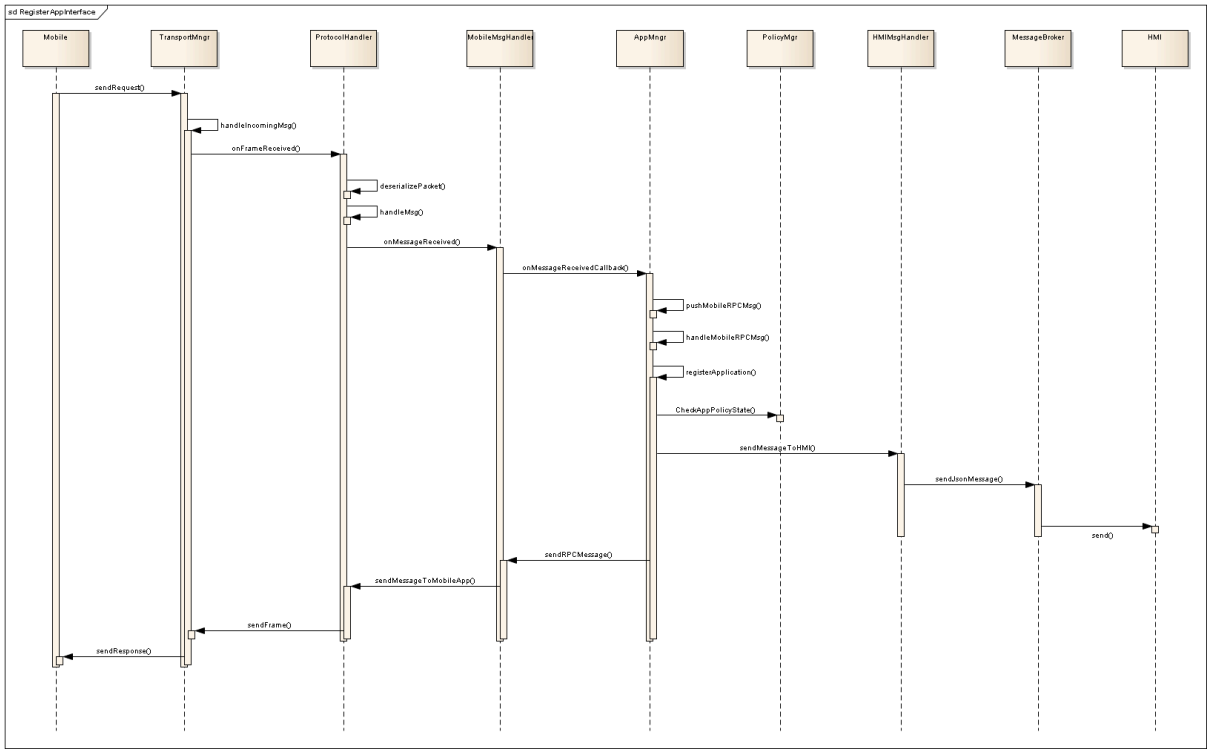

Figure 4: Graphical Diagram of Register Application Interface Scenario.

Behavior:

Establishes an interface with a mobile application.

The system will NOT accept/execute any other commands while processing the RegisterAppInterface request.

As a result:
- The mobile device will receive a response with corresponding return value.
- The HMI will receive a notification.
- Application will get initial permissions.

4.3 Use Case View

The following Use Case Diagram shows the actors, the processes and their interactions within SDL System.

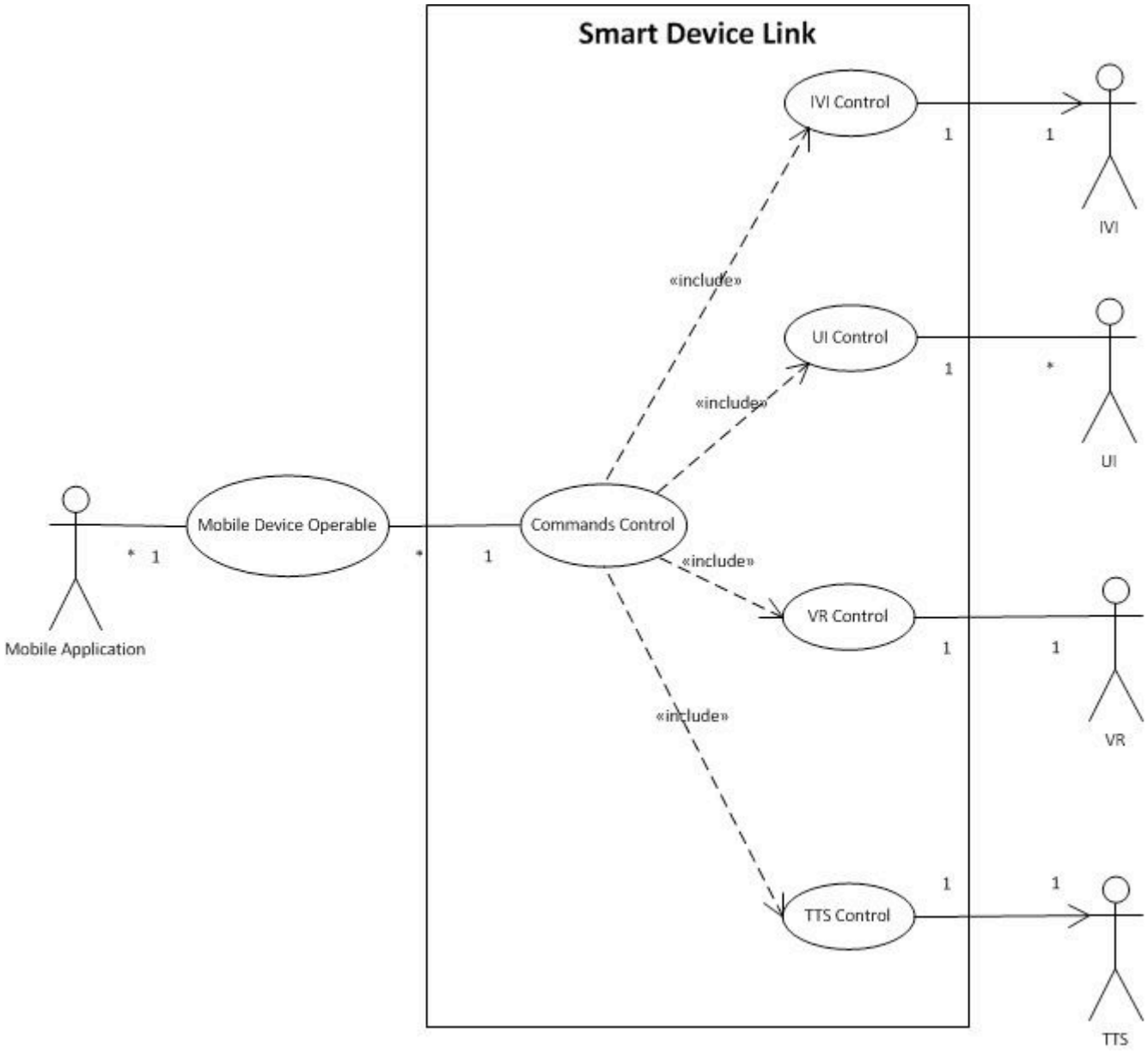

Figure 5: Use Case View Diagram

Element Name	Element Description
Voice Recognition Control	***Short Description:*** VR control is a part of SDL system. It allows SDL system to: - Manage Voice operating commands that arrive from mobile device to HMI - Process obtained results and notifications in reverse direction. ***Responsibilities are to:*** - Check the availability of VR module

	- Send VR commands to VR module if required - Process responses and notifications from VR module - Change/check the language of VR. ***Relations:*** - VR module - Commands control. ***Behavior:*** N/A ***Constraints:*** Is active only if VR module is available on HU.
UI Control	***Short Description:*** UI control operates with the commands which: - Come from the Mobile device and need to have the display in the vehicle UI - Come from HMI operating activities and need to be redirected to mobile device UI. Mobile application and HMI are the actors for this use-case. ***Responsibilities are to:*** - Check the UI availability - Process the requests from mobile application - Process the notifications and responses from HMI - Allow user to control the applications state - Allow choosing the current device. ***Relations:*** - HMI (physical/touchscreen buttons, display) - Commands Control. ***Behavior:*** N/A

	Constraints: N/A
TTS Control	***Short Description:*** TTS control allows managing text to speech operating commands from mobile device to HMI and vise-versa. HMI user and Mobile application user may operate with voice module on both devices via the application which implements the appropriate API-interface. ***Responsibilities are to:*** - Check TTS module availability - Send TTS commands to TTS module if required - Process responses and notifications from TTS module. ***Relations:*** - TTS module - Commands Control. ***Behavior:*** N/A ***Constraints:*** Is active only if TTS module is available on HU.
IVI Control	***Short Description:*** IVI Control - Describes how it's possible to operate with vehicle in-car information (temperature, seat position, etc.), which is changed under Mobile app user actions. - Allows mobile application to receive corresponding IVI data (temperature, seat position, etc.). ***Responsibilities are to:*** - Allow the mobile application subscribing for notifications from vehicle in-car information; - Process the requests from mobile application; - Process the responses and notifications from HMI.

	Relations: - Vehicle Data module - Command Control ***Behavior:*** N/A ***Constraints:*** N/A
Commands Control	***Short Description:*** Commands Control is a main SDL System part which manages communication between Mobile Application and HU Subsystems. ***Responsibility is to:*** - Allow the mobile application to communicate with above HU Subsystems. ***Relations:*** - Mobile Application - VR Control - TTS Control - IVI Control - UI Control ***Behavior:*** N/A ***Constraints:*** N/A

4.4 User Interface

Not applicable, since the User Interface is not the part of development.

4.5 Data View

The Data View shows relations between separated data types and actors that perform information processing in the system. It depicts contents of saved information and also visualizes information sources, processors and destination.

The following Diagram shows relations between separated data types and actors that perform information processing in the system.

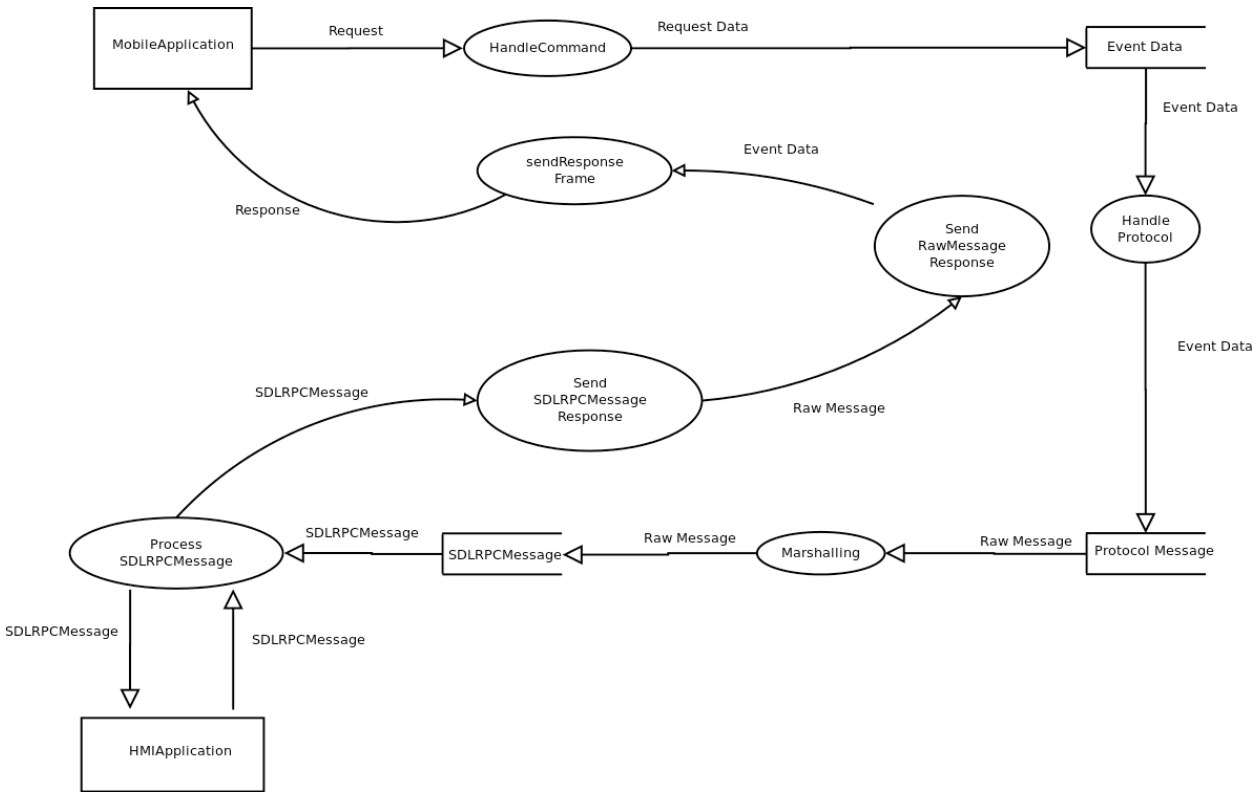

Figure 6: Data View Diagram

Element Name	Element Description
Event Data	***Short Description:*** Represents transport layer message. ***Responsibility:*** Contains binary data received from mobile device.

	Relations: - TransportManager ProtocolHandler ***Behavior:*** N/A ***Constraints:*** N/A
Protocol Message	***Short Description:*** Represents protocol layer message. ***Responsibility:*** Contains information about - Protocol version - Binary data - JSON string. ***Relations:*** - TransportManager - ProrocolHandler - MobileMessageHandler ***Behavior:*** N/A ***Constraints:*** Current implementation is based on V.2 protocol version.
SDLRPCMessage	***Short Description:*** Represents business layer message. ***Responsibility:*** Contains full information about:

	- RPC message - Protocol version - API version - binary data. ***Relations:*** N/A ***Behavior:*** N/A ***Constraints:*** Current implementation is based on V.2 API version.
HandleCommand	***Short Description:*** Callback for data received from mobile application. ***Responsibility:*** Receives binary frame from mobile application. ***Relations:*** - TransportManager ***Behavior:*** N/A ***Constraints:*** N/A
HandleProtocol	***Short Description:*** Callback for message received from mobile application.

	Responsibility: - Receives full message from mobile application - Parses binary headers. ***Relations*** - ProtocolHandler ***Behavior*** N/A ***Constraints*** Based on V.2 SDL protocol version
Marshalling	***Short Description:*** Parses JSON RPC message received from mobile application. ***Responsibility:*** RPC message received from mobile application. ***Relations:*** - ApplicatationManager ***Behavior:*** N/A ***Constraints:*** Based on V.2 JSON RPC API version
Process SDLRPCMessage	***Short Description:*** Business logic implementation. ***Responsibility:*** Main part of business logic implementation.

	Relations: - ApplicatationManager ***Behavior:*** Implementation depends on RPC command. ***Constraints:*** Based on V.2 JSON RPC API version.

4.6 Process State View

Not applicable, since the developed system works within one process.

4.7 Process View

The process view takes into account some requirements such as performance and system availability. It addresses concurrency and distribution, system integrity, and fault tolerance. The process view also specifies which thread of control executes each operation of each class identified in the logical view.

The figure below shows the diagram of the processes.

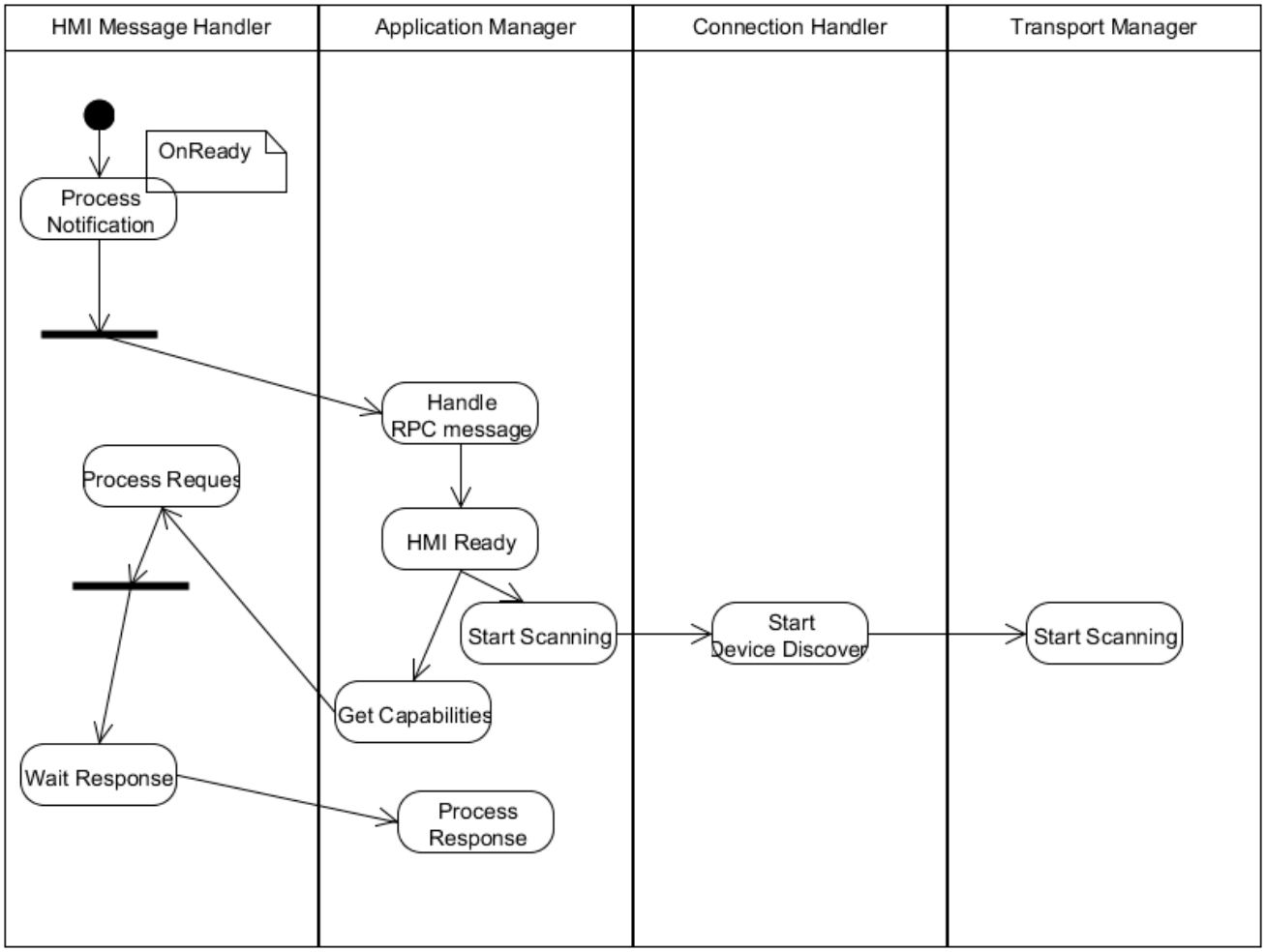

Figure 7: Process View Diagram

Element Name	Element Description
HMI Message Handler	***Short Description:*** Component is a proxy between transport layer for communication with HMI and Application Manager. ***Responsibility is to:*** - Convert transport-dependent messages to general format massages used by Application Manager. ***Relations*:** - MessageBroker

	- ApplicationManager ***Behavior:*** Processes message from HMI to application layer. ***Constraints:*** Process message from one adapter of HMI with corresponding transport
Application Manager	***Short Description:*** Component implements business logic of the SDL system. ***Responsibilities are to:*** - Manage behavior of Mobile Applications - Manage cooperation with HMI modules. ***Relations:*** - HMIMessageHandler - MobileMessageHandler - ConnectionHandler ***Behavior:*** Implements the main part of business logic. ***Constraints:*** Business logic constraints - please refer to Customer Requirements Specification [3].
Connection Handler	***Short Description:*** Component is a mid-layer between low-level connections and Application instances in Application Manager. ***Responsibilities are to:*** - Handle connections - manage sessions.

	Relations: - TransportManager - ProtocolHandler - ApplicationManager ***Behavior*** - Stores set of sessions for connection map; - Serves as a proxy for starting device discovery/updating device list. ***Constraints*** Maximum number of sessions per one connection is 256.
Transport Manager	***Short Description:*** Component is an abstraction over transport layer. ***Responsibility is to:*** - Manage low-level connections from Mobile Applications. ***Relations:*** - ConnectionHandler - ProtocolHandler ***Behavior:*** - Opens connection - Performs device discovery - Sends / receives messages. ***Constraints*** For Bluetooth transport there are only 32 connections available.

4.8 Development View

4.8.1 Implementation Technologies

The SmartDeviceLink is based on Linux as a freeware open-source OS. Using Linux OS provides a wide range of available free software libraries and SDK-s.

4.8.2 Modules and Code Base Organization

Development view organizes components in the system into logical, abstract groups called layers. The layers describe the major tasks that the components perform. The layers have different responsibilities and different providers

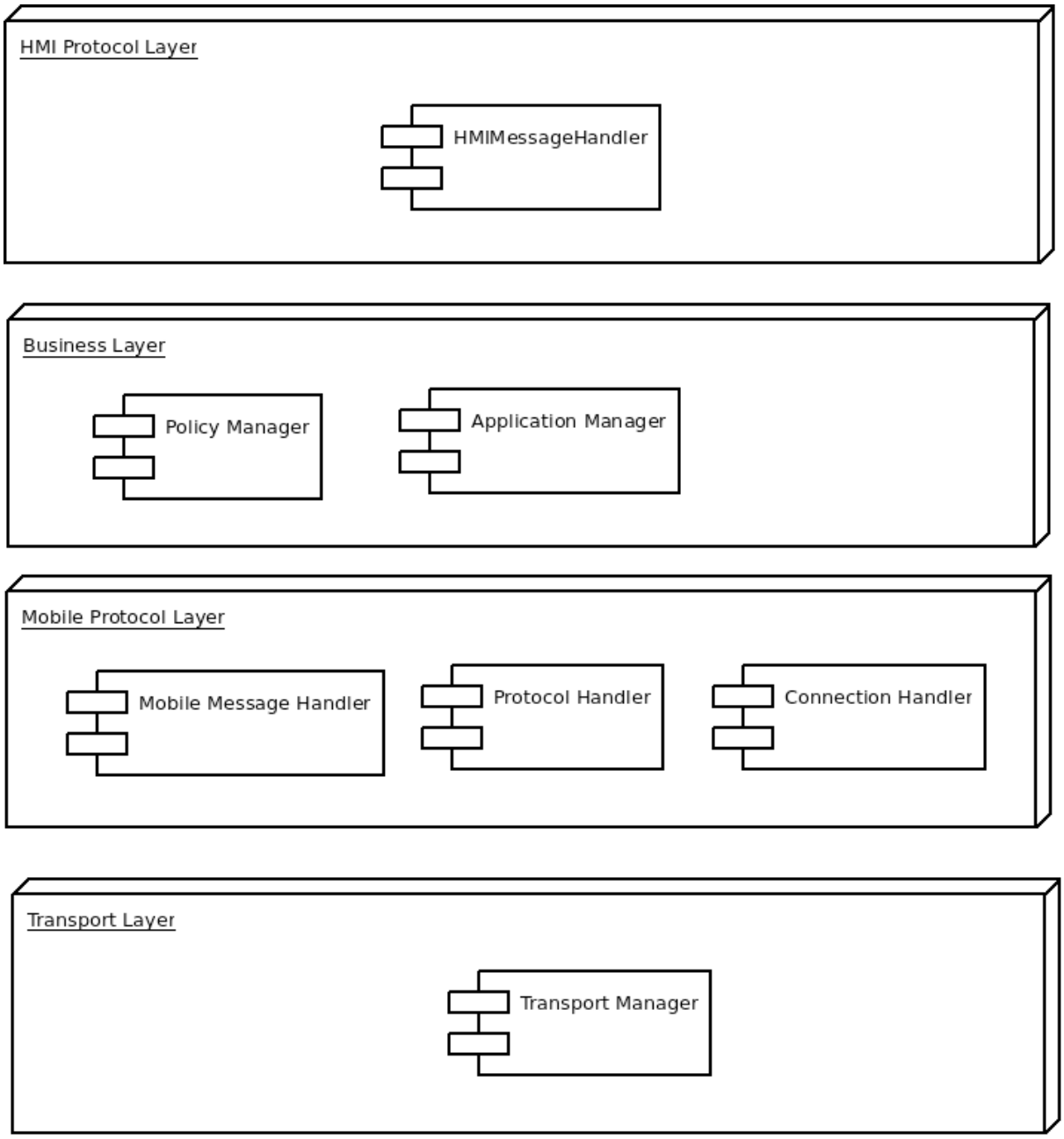

Figure 8: Development View Diagram

Element Name	Element Description
Transport Layer	***Responsibility*** Encapsulates different transport implementation ***Relations*** - Protocol layer ***Interfaces*** - ITransportManager - ITransportManagerDeviceListener - ITransportManagerDataListener ***Behavior*** - Opens connection - Performs device discovery - Sends / receives messages. ***Constraints:*** Current implementation works with Bluetooth and Wi-Fi only. Depends on type of transport.
Mobile Protocol Layer	***Responsibility:*** Encapsulates different protocol implementation. ***Relations:*** - Business layer - Transport layer ***Interfaces:*** - MobileMessageHandler - MobileMessageHandlerObserver - ConnectionHandler - ConnectionHandlerObserver

	Behavior: Parses headers of messages from transport layer and sends message body to MobileMessageHandler and vice versa ***Constraints:*** Depends on V.2 SDL protocol version.
Business Layer	***Responsibility:*** Represents main business logic implementation ***Relations:*** - HMI Protocol Layer - Mobile Protocol Layer ***Interfaces:*** - ApplicationManager - PolicyManager ***Behavior:*** Main business logic functionality. ***Constraints:*** Business logic constraints - please refer to CRS [3] for details
HMI Protocol Layer	***Responsibility:*** Represents HMI protocol and transport layers abstraction. ***Relations:*** - Business Layer ***Interfaces:*** - HMIMessageObserver - HMIMessageHandler ***Behavior:*** Processes messages from SDL to HMI module.

	Constraints: Depends on transport and protocol to HMI

4.8.3 Development Environment and Standards

Target OS: Ubuntu 12.04 and upper (x86 and x64)
Unit test framework: Google Test
Compiler and utilities: g++, make, cmake
Coding style: Google C++ Style Guide [5]

4.8.4 Testing Environment and Standards

For detailed information please refer to Unit Testing Plan [5], Test Plan [6] and PMP [2].

4.9 Deployment View

The deployment view takes into account the system's requirements such as system availability, reliability (fault tolerance), performance (throughput), and scalability. This view maps the various elements identified in the logical, process, and development views—networks, processes, tasks, and objects—onto the processing nodes.

The deployment diagram is used for modeling the static deployment view of a system.

The figure below depicts the deployment diagram for SDL system.

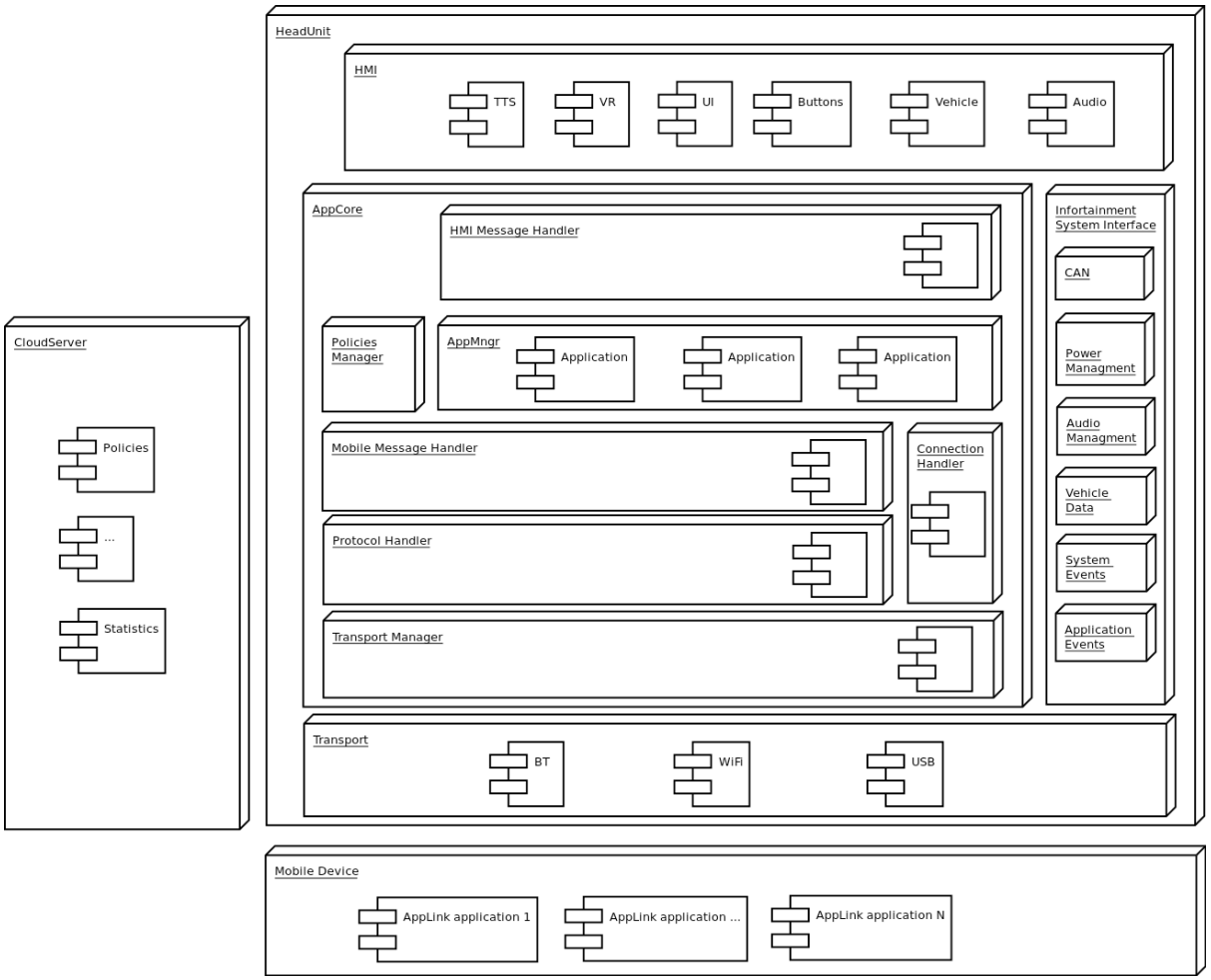

Figure 9: Deployment View Diagram

Element Name	Element Description
Mobile Device	***Short Description:*** The SDL application model permits multiple applications to be concurrently active and connected to the HU. A few of those applications may communicate with the user at a time using the HMI (depending on HMI). SDL uses the concept of HMI Levels to describe the current state of the application with regards to the level at which the head unit can communicate with it (and vice versa). ***Relations:*** - Runs **SYNK SDK**;

	- Receives policies updates from **Cloud Server**; - Sends statistics to **Cloud Server**. ***Requirements:*** Android OS / iOS.
Head Unit	***Short Description:*** HU HMI allows the user/driver to interact with the vehicle. This interface includes: - A set of presets - Media buttons (seek forward/backward, tune up/down, and play/pause) - Menu items - Graphic user interface - Voice commands, etc. The buttons, menus or other controls available may vary between vehicles and head units. The HU HMI Handler interfaces with SDL Core to support the API functionality. ***Relations:*** - Runs **HMI application** - Runs **SDL application** with ability to perform file operations inside directory specified in configuration file. ***Requirements:*** - POSIX-compliant OS - Availability of transport support: Bluetooth/Wi-Fi/USB.
CloudServer	***Short Description:*** A Server that provides information about: - Which applications are allowed to run in vehicle - What interfaces application is allowed to use. In addition, server provides: - System configuration, including the time of the next file update - Some important server information to the back end user

	Relations: - Sends policies updates to **Mobile Device**. - Receives statistics from **Mobile Device**. ***Requirements:*** N/A

4.10 Operational View

This view describes how the architecture provides the ability for operation/support teams to monitor and manage the system.

To make system more flexible and to support different platforms, SW provides a configuration component, which is able to change system behavior according to settings defined in `smartDeviceLink.ini` file.

Configuration Manager, Support Engineer or SW Engineer, time to time, need to change system behavior without recompiling the project. For this purpose the desirable settings could be changed in `smartDeviceLink.ini` file.

Element Name	Element Description
Configuration	Configuration component specifies the desirable system behavior on different platforms. Configuration component provides the following settings: – Desirable transport (WebSocket, DBus) – IP address and the port of the server – Specify global properties like `HelpPrompt`, `TimeoutPrompt`, `HelpTitle`, `HelpCommand` – `Default Timeout` for mobile application commands – Desirable location of the system data (log files, persistence data, temporary data)

4.11 Logical View

The Factory pattern is used for object creation without exposing the instantiation logic to the client. Client refers to the newly created object through a common (base) interface. The Singleton pattern is used to ensure that only one instance of the class is created.

The figure below depicts the logical view diagram for SDL system.

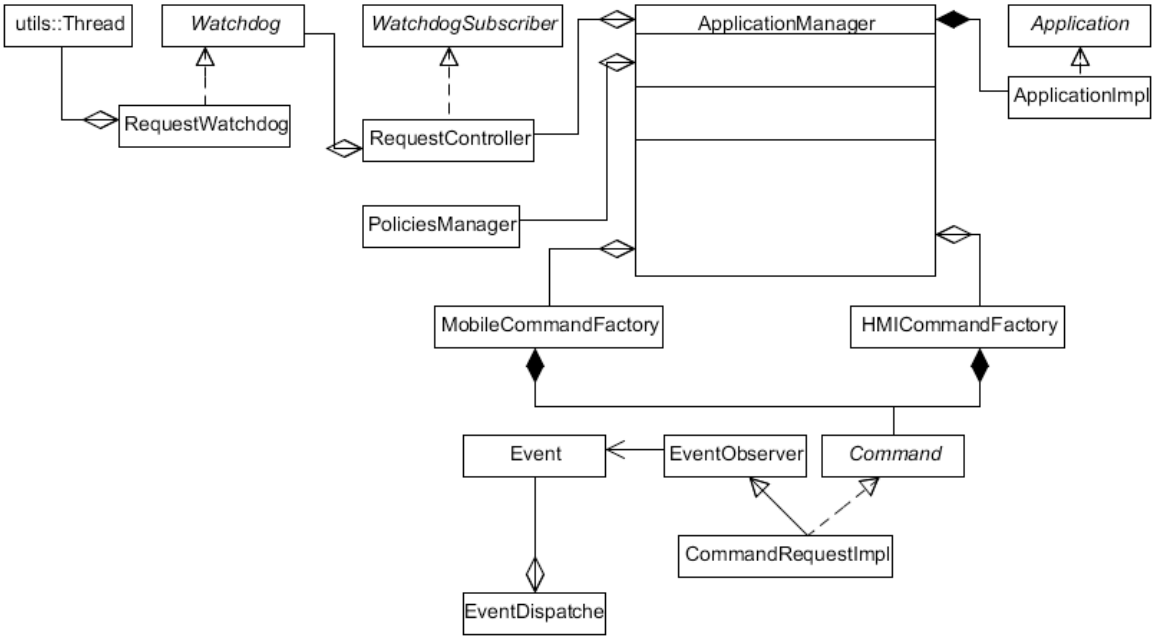

Figure 10: Logical View Diagram

Element Name	Element Description
Application Manager	***Short Description:*** - Main class of Application Manager component. - Manages all classes responsible for business logic in Application Manager component. - Contains list of applications and global properties for all applications.
Application	***Short Description:*** - Represents application in terms of SDL logic. - Contains application-specific information.
MobileCommandFactory	***Short Description:*** - Creates command objects form message recieved from Mobile side.
HMICommandFactory	***Short Description:*** - Creates command objects form message recieved from HMI side.
Command	***Short Description:*** - Encapsulates business logic of specific API call. - Performs actions necessary to execute API according to specification (i.e. stores received file, sends message to HMI/Mobile side).
CommandRequestImpl	***Short Description:*** - Encapsulates business logic of API request.
Event	***Short Description:***

	- Class for exchanging notifications.
EventDispatcher	***Short Description:*** - Responsible for managing sending events to subscribers
EventObserver	***Short Description:*** - Subscriber for specific event/events
Watchdog	***Short Description:*** - Checks timeouts of waiting for responses from HMI.
RequestController	***Short Description:*** - Responsible for handling timeouts of responses and notifications from HMI.
PoliciesManager	***Short Description:*** - Encapsulates policies-related activity: storing; managing; using.

5 View-to-View Relations

Each of the views specified in Section 3 provides a different perspective and design handle on a system, and each is valid and useful in its own right. Although the views give different system perspectives, they are not independent. Elements of one view will be related to elements of other views, and we need to reason about these relations.

Section 4 describes the relations that exist among the views given in Section 3. As required by ANSI/IEEE 1471-2000, it also describes any known inconsistencies among the views.

5.1 Logical-to-Component

The following table is a mapping between the elements in the logical view and the Components view. The relationship shown is *is-implemented-by*, i.e. the elements from the Components view shown at the top of the table are implemented by any selected elements from the Logical view, denoted by an "X" in the corresponding cell.

	Transport Manager	Connection Handler	Protocol Handler	Application Manager	HMIMessage Handler	Mobile Message Handler
ApplicationManager				x		
Application				x		
Version				x		
HMIMatrix				x		
HMIMatrixItem				x		
PoliciesManager				x		
PoliciesStorage				x		
PoliciesItem				x		
WatchDog				x		
ConnectionHandler		x				
ProtocolHandler			x			
TransportManager	x					
MobileMessageHandler						x
HMIMessageHandler					x	

5.2 Component-to-Layer

The following table is a mapping between the elements in the Component view and the Development view. The relationship shown is *is-implemented-by*, i.e. the layers from the Development view shown at the top of the table are implemented by any selected elements from the Component view, denoted

by an “X” in the corresponding cell.

	Transport Layer	Protocol Layer	Business Layer	HMI Protocol Layer
Transport Manager	x			
Connection Handler		x		
Protocol Handler		x		
Mobile Message Handler		x		
Application Manager			x	
Policy Manager			x	
HMI Message Handler				x
Settings Manager			x	

6 Solution Background

6.1 Architecture Design Approach

During the architecture designing the following aspects were primary considered:

1. **Weak components coupling.** The Observer pattern is used for implementing this requirement.

With the help of Observer it is possible to:

- Notify all observers about occurred changes via notifications
- Support each component separately
- Easily integrate new components into the system.

Publish-Subscribe design pattern was considered as an alternative to Observer design pattern. Still, it was postponed because of time limit and refactoring of existing code.

2. **Ensuring one instance of the class.** Singleton design pattern is used to ensure that only one instance of class is created.

3. **SW portability.** Facade design pattern is used for achieving this goal to wrap used system calls.

4. **System scalability.** Ensuring that transport extensions like adding D-Bus, CAN or other would not cause architecture refactoring was one of the main concerns.

For achieving this goal the following patterns are used:

- Adapter design pattern is used to provide permanent interface to transport layer.
- Abstract Factory design pattern is used to create the related objects without specifying their concrete classes directly.
- Command design pattern is used to treat requests as an object that provides possibility to add new request without existing code modification.

5. **Avoid memory leak.** SmartPointer design pattern is used for that point for proper memory deallocation.

6.2 Requirements Coverage

There are indirect requirements which may impact on Architectural decisions, such as limitation of usage of RAM, ROM, requirements to support specific SDL Core to HMI transport layers.
All the requirements of this kind were taken into account while creating Architecture Design. For more details related to the requirements please refer to the following sources:

- Requirements Analysis Table [7]
- Customer Requirements Specification [3]

6.3 Prototyping Results

Architecture prototyping was done to validate architecture on early stages. An evolutional prototyping technique was used. Thus all prototype components were used with non-significant changes and additional features for further development.

6.4 Open Questions and Known Issues

Please refer to Confluence Technical Documents [8].

6.5 Results Analysis

Not applicable, since no quantitative or qualitative analysis was performed.

8 List of Figures

9 Appendices

None

10 Change History

Version	Date	Status	Change description	Author/Editor
V.1.0	04/23/2013	Initial Version	Template creation; Chapter 4 'Views' creation	A.Kandul
V.1.1	04/30/2013	Draft	Chapter 5 creation	D.Klimenko
V.1.2	05/10/2013	Draft	Description for the Views creation	P.Vyshnevska
V.1.3	05/14/2013	Draft	Chapters 1,2,3 creation	A.Kandul
V.1.4	05/27/2013	Draft	Views diagrams and description update	A.Kandul
V.1.5	05/28/2013	Draft	Chapters 7,8 creation; formatting; grammar corrections	A.Britanova
V.2.0	05/30/2013	Draft	Ready for customer review	A.Britanova
V.2.1	09/30/2013	Draft	Changes in accordance to implementation	P.Vyshnevska
V.2.2	10/31/2013	Reviewed	Reviewed with customer	P. Vyshnevska P. Savyelyev
V.2.3	05/23/2014	Draft	Policy component updated	A.Oleynik

11 Approve History

Version	Approval Date	Issue ID	Approved By
V.2.2	11/01/2013		Pavel Savyelyev
V.2.2	11/06/2013	N/A	Julius Marchwicki