LGE Internal Use Only

SDCM  
(Smart Device Connection Manager)

Wireless Phone Projection 대응 설계

VS 연구소 Phone Projection Unit  
Bonhun Koo

About This Document

Document Information

|  |  |
| --- | --- |
| Issuing authority | LGE |
| **Status of document** | In Progress / Approved / Released |

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Comment | Author | Approver |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 1.2 | 2023-04-9 | 2차 리뷰 사항 업데이트 | 구본훈 책임 | 김태호 이사 유휘상 책임  정영태 책임 |
| 1.1 | 2023-03-07 | 1차 리뷰 사항 업데이트 | 구본훈 책임 | 김태호 이사  유휘상 책임  정영태 책임 |
| 1.0 | 2023-02-24 | Initial document | 구본훈 책임 |  |

**About This Document**

Purpose and Scope

본 문서는 SDCM 의 Architecture, 기능 그리고 Interface에 대해 기술한다.

Audience

본 문서는 다음의 관계자에게 open 된다.

* Project Manager and Leader
* SDCM 개발자
* SDCM 과 연동하는 Client 개발자

Related Documents

[1] AndroidAuto – Google spec

[2] CarPlay - Apple spec

[3] CarLife - Baidu spec

[4] Hyundai – Wireless Phone Projection scenario

[5] Software Architecture in Practice (4th)

[6] Documenting Software Architectures (2nd)

## Acronyms / Glossary

|  |  |
| --- | --- |
| Acronym | Description |
| SyAD | System Architectural Design |
| SyRS | System Requirements Specifications |
| SRS | Software Requirements Specifications |
| SAD | Software Architectural Design |
| SDD | Software Detailed Design |
| HLD | High Level Design |
| LLD | Low Level Design |
| BT | Bluetooth |
| SDCM | Smart Device Connection Manager |
| DCM | Device Connection Manager |
| HU | Head Unit |
| iAP2 | iPod Accessory Protocol |
| MTP | Media Transfer Protocol |
| MSC | Mass Storage Class |
| USB | Universal Serial Bus |
| SPP | Serial Port Profile |
| MARS | Module Archive and Reuse System  http://mars.lge.com/ |

|  |  |
| --- | --- |
| Glossary | Description |
| Technology | Phone projection 의 종류로는 다양한 것이 있습니다. iPhone 에서 사용되는 CarPlay, Android 단말에서 사용되는 AndroidAuto 그리고, Android 단말, iPhone 에서 사용할 수 있는 CarLife (중국향만 지원) 등이 있으며, 이러한 Phone projection 을 내부적으로 Technology 라고 명시하고 있습니다. |
| Android Auto | Android Auto는 구글이 개발한 자동차용 연동 기술로, 안드로이드 스마트폰을 자동차에 연결하여 안전하고 편리하게 운전 중에도 안드로이드 스마트폰을 사용할 수 있게 해줍니다. Android Auto를 사용하면 자동차의 H/U (Head Unit) 화면에 안드로이드 스마트폰의 내용을 표시할 수 있습니다. 이를 통해 음악, 전화, 메시지, 지도 등의 구글의 기본 앱을 자동차의 H/U (Head Unit) 에서 직접 조작할 수 있으며, 음성 명령을 통해 제어할 수도 있습니다. |
| CarPlay | CarPlay는 애플사에서 개발한 자동차용 연동 기술로, iPhone을 자동차에 연결하여 안전하고 편리하게 운전 중에도 iPhone을 사용할 수 있게 해줍니다. CarPlay를 사용하면 자동차의 H/U (Head Unit) 화면에 iPhone의 내용을 표시할 수 있습니다. 이를 통해 음악, 전화, 메시지, 지도 등의 애플의 기본 앱을 자동차의 H/U (Head Unit) 화면에서 직접 조작할 수 있으며, Siri를 통해 음성으로도 제어할 수 있습니다. |
| CarLife | CarLife는 자동차용 스마트폰 연동 기술로, 자동차 내에서 모바일 기기를 사용할 수 있게 해주는 솔루션입니다. CarLife를 사용하면 자동차 내부의 H/U (Head Unit) 화면에 모바일 기기의 내용을 표시할 수 있으며, 모바일 기기를 자동차 내부에서 쉽게 조작할 수 있습니다. 음악, 전화, 메시지, 지도 등의 기본 앱을 자동차의 H/U (Head Unit) 에서 직접 조작할 수 있습니다. 중국향 차량에서 CarLife 가 주로 사용되어지고 있습니다. |
| MirrorLink | MirrorLink는 자동차용 스마트폰 연동 기술로, 안드로이드와 iOS 기반 모바일 기기를 자동차의 H/U (Head Unit) 에 연결하여 운전 중에도 안전하게 모바일 기기를 사용할 수 있게 해주는 기술입니다.  MirrorLink를 사용하면 자동차 내부의 H/U (Head Unit) 화면에 모바일 기기의 내용을 표시할 수 있으며, 모바일 기기를 자동차 내부에서 쉽게 조작할 수 있습니다. 음악, 전화, 메시지, 지도 등의 기본 앱을 자동차의 H/U (Head Unit) 에서 직접 조작할 수 있습니다. |
| iPod | iPod는 애플이 개발한 포터블 미디어 플레이어로, 음악, 비디오, 사진 등 다양한 멀티미디어 파일을 재생할 수 있는 디바이스입니다. iPhone 도 iPod 기능을 포함하고 있는 데, iPhone 을 자동차에 연결하면, iPod 으로 동작할 수 있으며, 음악을 재생하면서 Cover Art 를 표시하는 역할을 해 줍니다. |

[About This Document 3](#_Toc131950798)

[Acronyms / Glossary 4](#_Toc131950799)

[1 Overview 10](#_Toc131950800)

[1.1 프로젝트 소개 10](#_Toc131950801)

[1.2 Phone Projection 기능 소개 10](#_Toc131950802)

[1.3 SDCM 서비스 시스템 소개 11](#_Toc131950803)

[1.3.1 Context Diagram 12](#_Toc131950804)

[1.3.2 Stakeholders 13](#_Toc131950805)

[1.3.3 Task Members 13](#_Toc131950806)

[1.3.4 Scrum 활동 내역 13](#_Toc131950807)

[1.4 설계 범위 14](#_Toc131950808)

[1.5 과제의 핵심 목표 14](#_Toc131950809)

[2 Architect Driver 15](#_Toc131950810)

[2.1 Functional Requirements 15](#_Toc131950811)

[2.2 Use Case List 18](#_Toc131950812)

[2.2.1 Use Case Diagram 18](#_Toc131950813)

[2.2.2 Use Case Scenarios 18](#_Toc131950814)

[2.3 Quality Attribute 26](#_Toc131950815)

[2.3.1 Architectural Approach/Analysis 26](#_Toc131950816)

[2.3.2 Prioritization of Quality Attributes 30](#_Toc131950817)

[2.4 Constraints 30](#_Toc131950818)

[2.4.1 Business constraints 30](#_Toc131950819)

[2.4.2 Technical constraints 30](#_Toc131950820)

[3 Architectural Design 31](#_Toc131950821)

[3.1 Layered Architecture 31](#_Toc131950822)

[3.2 Layer 구성 31](#_Toc131950823)

[3.3 Static Perspective 32](#_Toc131950824)

[3.3.1 1st Decomposition 32](#_Toc131950825)

[3.3.2 2nd Decomposition 33](#_Toc131950826)

[3.3.1 Module View 35](#_Toc131950827)

[3.3.2 설계 Module Role 및 FR 담당 부분 정리 36](#_Toc131950828)

[3.4 Dynamic Perspective 38](#_Toc131950829)

[3.4.1 1st Level Decomposition – C&C View 38](#_Toc131950830)

[3.4.2 품질 속성 : QA.09 Project 별 기능 추가 시 공용 부의 독립성 보장 39](#_Toc131950831)

[3.4.3 설계 대안 : Publish-Subscribe vs Peer to Peer 39](#_Toc131950832)

[3.4.4 설계 결정 및 근거 : Publish-Subscribe 결정 42](#_Toc131950833)

[3.4.5 Risk 식별 및 개선 전략 : Publish-Subscribe 43](#_Toc131950834)

[3.4.6 세부 설계 QA.09 Design 1. Message Broker design 44](#_Toc131950835)

[3.4.7 품질 속성 : QA.07 Usability 공용 부와 변동 부의 분리 45](#_Toc131950836)

[3.4.8 설계 대안 : Maintain Task Model (Tactic) vs Toggle Features (Tactic) 45](#_Toc131950837)

[3.4.9 설계 결정 및 근거 : Maintain Task Model Tactics 48](#_Toc131950838)

[3.4.10 Risk 식별 및 개선 전략 : Maintain Task Model Tactics 49](#_Toc131950839)

[3.4.11 2nd Level Decomposition – C&C View 50](#_Toc131950840)

[3.4.12 품질 속성 : QA.07 Usability 공용 부와 변동 부의 분리 (2nd Level) 51](#_Toc131950841)

[3.4.13 설계 대안 : Task + Event driven vs Task + Thread wait 51](#_Toc131950842)

[3.4.14 설계 결정 및 근거 : Task + Thread wait 53](#_Toc131950843)

[3.4.15 Risk 식별 및 개선 전략 : Task + Thread wait 54](#_Toc131950844)

[3.4.16 세부 설계 QA.07 Design 3. Task Manager - Task Operator 55](#_Toc131950845)

[3.4.17 품질 속성 : QA.06 Maintainability 연결 관리의 독립성 보장 56](#_Toc131950846)

[3.4.18 설계 대안 : Client type 별로 관리 vs Technology 진행 상황 별로 관리 56](#_Toc131950847)

[3.4.19 설계 결정 및 근거 : Technology 진행 상황 별로 관리 57](#_Toc131950848)

[3.4.20 Risk 식별 및 개선 전략 : Connection Handler 59](#_Toc131950849)

[3.4.21 세부 설계 QA.06 Design 4. Connection Handler 59](#_Toc131950850)

[3.5 Allocation View 60](#_Toc131950851)

[3.5.1 Allocation Style 60](#_Toc131950852)

[3.5.2 Implementation Style 60](#_Toc131950853)

[3.6 Traceability summary 62](#_Toc131950854)

[4 Architectural Validation 63](#_Toc131950855)

[4.1 Maintainability QA06 63](#_Toc131950856)

[4.2 Usability QA07 64](#_Toc131950857)

[4.3 Modifiability QA09 65](#_Toc131950858)

[5 Wrap up 66](#_Toc131950859)

[5.1 사업적 성과와 기여 66](#_Toc131950860)

[5.2 Lessons Learned 67](#_Toc131950861)

[5.3 Question and Answer 68](#_Toc131950862)

**Figures**

[Figure 1‑1 SDCM 서비스 시스템 소개 11](#_Toc131950863)

[Figure 1‑2 Context diagram 12](#_Toc131950864)

[Figure 2‑1 Use case diagram 18](#_Toc131950865)

[Figure 2‑2 Use case 1 diagram 19](#_Toc131950866)

[Figure 2‑3 Use case 2 diagram 20](#_Toc131950867)

[Figure 2‑4 Use case 3 diagram 21](#_Toc131950868)

[Figure 2‑5 Use case 4 diagram 22](#_Toc131950869)

[Figure 2‑6 Use case 5 diagram 23](#_Toc131950870)

[Figure 2‑7 Use case 6 diagram 24](#_Toc131950871)

[Figure 2‑8 Use case 7 diagram 25](#_Toc131950872)

[Figure 3‑1 Static view - 1st decomposition diagram 32](#_Toc131950873)

[Figure 3‑2 Static view - External layer diagram 33](#_Toc131950874)

[Figure 3‑3 Static view - Service layer description 34](#_Toc131950875)

[Figure 3‑4 Module View 35](#_Toc131950876)

[Figure 3‑5 Dynamic view - 1st decomposition 38](#_Toc131950877)

[Figure 3‑6 Dynamic view - Design 1 39](#_Toc131950878)

[Figure 3‑7 Dynamic view - Message Broker 44](#_Toc131950879)

[Figure 3‑8 Dynamic view - Design 2 45](#_Toc131950880)

[Figure 3‑9 Dynamic view - 2nd decomposition 50](#_Toc131950881)

[Figure 3‑10 Dynamic view - Task Operator 51](#_Toc131950882)

[Figure 3‑11 Dynamic view - Task Operator (Tasks) 53](#_Toc131950883)

[Figure 3‑12 Dynamic view - 2nd level (Task Manager) 55](#_Toc131950884)

[Figure 3‑13 Dynamic view - Connection Manager 56](#_Toc131950885)

[Figure 3‑14 AVN 구성요소 60](#_Toc131950886)

[Figure 3‑15 Implementation Style 61](#_Toc131950887)

**Tables**

[Table 1‑1 Element Responsibility in System Context 12](#_Toc131950888)

[Table 1‑2 Stakeholders 13](#_Toc131950889)

[Table 1‑3 Task Members 13](#_Toc131950890)

[Table 2‑1 Functional Requirement 15](#_Toc131950891)

[Table 2‑2 Use case 1 scene 19](#_Toc131950892)

[Table 2‑3 Use case 2 scene 20](#_Toc131950893)

[Table 2‑4 Use case 3 scene 21](#_Toc131950894)

[Table 2‑5 Use case 4 scene 22](#_Toc131950895)

[Table 2‑6 Use case 5 scene 23](#_Toc131950896)

[Table 2‑7 Use case 6 scene 24](#_Toc131950897)

[Table 2‑8 Use case 7 scene 25](#_Toc131950898)

[Table 2‑9 QA-01 Performance scenario 26](#_Toc131950899)

[Table 2‑10 QA-02 Performance scenario 26](#_Toc131950900)

[Table 2‑11 QA-03 Performance scenario 27](#_Toc131950901)

[Table 2‑12 QA-04 Performance scenario 27](#_Toc131950902)

[Table 2‑13 QA-05 Interoperability scenario 27](#_Toc131950903)

[Table 2‑14 QA-06 Maintainability scenario 28](#_Toc131950904)

[Table 2‑15 QA-07 Usability scenario 28](#_Toc131950905)

[Table 2‑16 QA-08 Reusability scenario 29](#_Toc131950906)

[Table 2‑17 QA-09 Modifiability scenario 29](#_Toc131950907)

[Table 2‑18 Prioritization of quality attributes 30](#_Toc131950908)

[Table 3‑1 Static View - Layer description 32](#_Toc131950909)

[Table 3‑2 Static View - External layer description 33](#_Toc131950910)

[Table 3‑3 Static View - Service layer description 34](#_Toc131950911)

[Table 3‑4 Static View - 설계 Module role 36](#_Toc131950912)

[Table 3‑5 Static View - Module / FR 연관 36](#_Toc131950913)

[Table 3‑6 Dynamic view - 1st decomposition / responsibility 38](#_Toc131950914)

[Table 3‑7 Dynamic View – pub-sub, peer to peer 39](#_Toc131950915)

[Table 3‑8 Dynamic View - Message Broker, Communicator 42](#_Toc131950916)

[Table 3‑9 Dynamic View – Topic / Subject 44](#_Toc131950917)

[Table 3‑10 Dynamic view - Maintain Task Model / Toggle feature 46](#_Toc131950918)

[Table 3‑11 Dynamic view - Design 2 (Task Manager) 48](#_Toc131950919)

[Table 3‑12 Dynamic View - Design 3 (Task Manager) 51](#_Toc131950920)

[Table 3‑13 Dynamic view - Connection manager 57](#_Toc131950921)

[Table 3‑14 Traceability summary 62](#_Toc131950922)

# Overview

## 프로젝트 소개

SDCM(Smart Device Connection Manager)의 기능은 H/U (HeadUnit / 차량용 Navigation) 과 Phone, USB stick 등의 연결 과정을 주관해 주는 역할을 하고 있습니다.

기기(Phone) 연결에는 AndroidAuto(Google), CarPlay(Apple), MirrorLink 및 CarLife(Baidu)가 있습니다. H/U 에 Phone 이 인식되는 지 Udev monitoring (Linux platform 에서는 USB 장치를 인식하는 순간 Udev event 를 발생시킴) 을 하다가 Phone 이 인식되는 순간 어떤 종료의 Phone 인지 판별하게 되며, 해당 정보를 사용하여 Phone Projection (AndroidAuto/ CarPlay 등) 을 사용할 수 있도록 연결 요청을 하게 됩니다.

H/U 에 Phone 이 인식되는 방법이 Wired (USB cable 연결) 방식과 Wireless (BT Spp / WIFI) 방식이 있습니다. 양산 Project 의 경우 Wired 만 인식하고 있었으며, 이번 프로젝트 설계는 Wireless 를 인식, 연결 진행하기 위한 설계입니다.

Phone projection 의 연결 관리 외에 SDCM은 USB 스틱이 삽입되었는지 여부를 인식합니다. SDCM은 USB 스틱을 인식한 후 USB 스틱 장치를 mount 처리 (Linux platform 에서는 USB 스틱을 인식하여 mount 처리를 해야 App 들이 사용할 수 있음) 한 다음 USB 스틱 정보를 미리 등록된 클라이언트 (미디어 앱 등) 에 정보를 보냅니다. 해당 정보를 수신한 Client (미디어 앱 등) 은 USB 장치에 접근하여 음악 재생, 영상 재생 등을 진행할 수 있습니다.

## Phone Projection 기능 소개

Phone projection 기능은 Android 단말, iPhone 단말 등에서 H/U (HeadUnit) 과 연동하여 미러링을 하는 기능을 지칭합니다. Android 단말에서는 AndroidAuto, MirrorLink, CarLife 등의 서비스가 있으며, iPhone 단말에서는 CarPlay, CarLife 등의 서비스가 Phone projection 기능으로 제공되고 있습니다.

현재 양산되는 현대 차종에서는 Phone projection wired 기능이 지원되고 있으며, 지원되는 방식은 USB 케이블에 Phone 을 연결하면 해당 장치를 인식하여 Phone projection 이 진행되는 방식입니다.

위와 같은 방식을 wired phone projection 이라고 하며, 무선 기능을 이용하여 Phone projection 을 진행하는 것은 wireless phone projection 이라고 합니다.

Wired phone projection 이 범용적으로 사용하는 방법이며, Wireless phone projection 은 일부 차종에서 지원하고 있습니다.

Wired phone projection 의 경우 연결된 USB 장치를 인식하여 Phone projection 을 진행하면 되지만, Wireless phone projection 은 이보다 조금 더 복잡한 과정을 거쳐야 합니다.

Wireless phone projection 연결 시작으로 BT Spp (Serial port profile) 을 이용하여 시작이 되며, BT 의 데이터 전송 폭의 한계로 이내 WIFI 로 전환하여 데이터 통신을 하게 됩니다.

Wireless phone projection 을 지원하기 위해서는 BT/ WIFI 기능이 지원되어야 합니다.

BT/ WIFI 기능이 지원되면 Wireless phone projection 을 구현할 수 있으며, 현재 양산된 차량에서는 BT 는 기본으로 지원하고 있으며, WIFI 기능도 함께 탑재된 차량이 많습니다.

이번 과제에서는 USB 연결 기능 외에 BT/ WIFI 를 사용하는 Wireless phone projection 을 용이하게 기능 적용하고자 설계가 진행되었습니다.

## SDCM 서비스 시스템 소개

SDCM은 Client (App/ Core/ Service) 등과 통신하여 Phone Projection 연결 등을 관할합니다.  
연결되는 장치들 (Phone/ USB media) 등을 인지하여 연결/해제 동작이 필요한 경우 자체적 판단 또는 List app 의 요청에 의해 Core (AndroidAuto/ CarPlay/ CarLife 등) 에 연결 / 해제 요청하여 Phone Projection 진행을 하도록 합니다.

USB 대용량 메모리 장치들이 인지되는 경우 Mount/ Un-mount 등의 동작을 수행 하며, 해당 정보들을 Client (Media app 등) 에 broadcast 하는 역할을 합니다.

Media app 은 해당 정보를 받아 미디어 (음원/ 비디오) 등을 재생하게 됩니다.

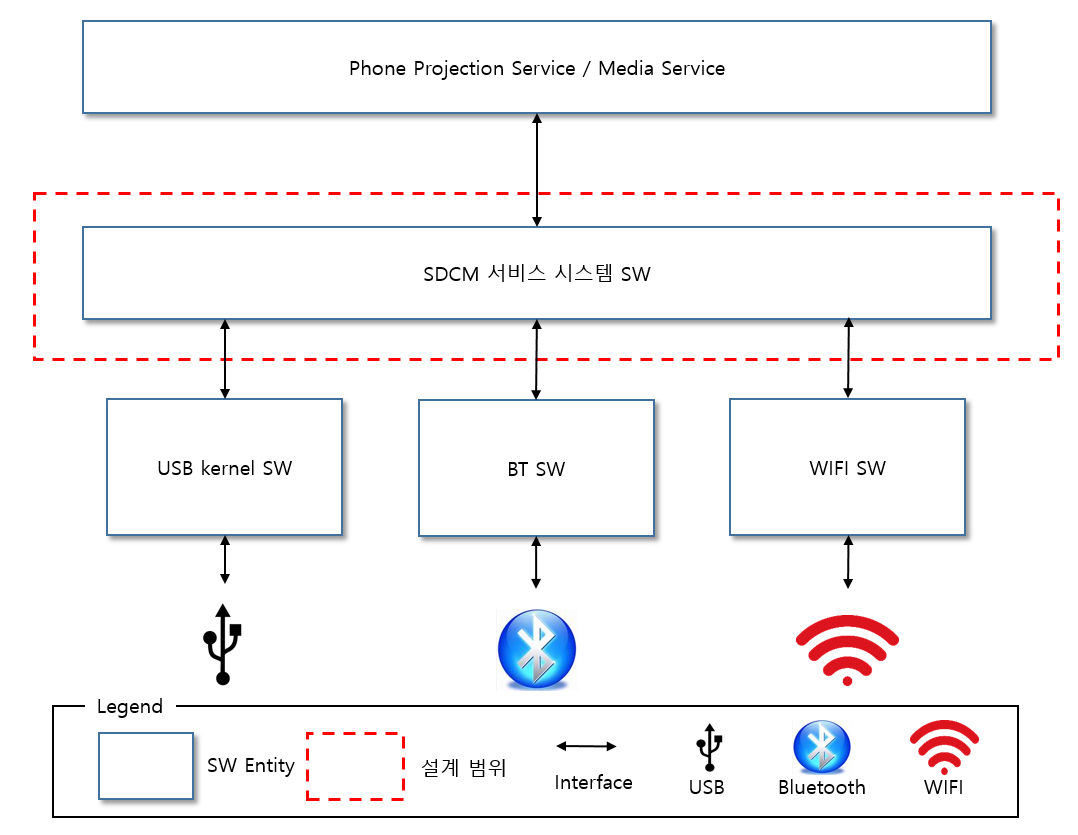


Figure 1‑1 SDCM 서비스 시스템 소개

### Context Diagram

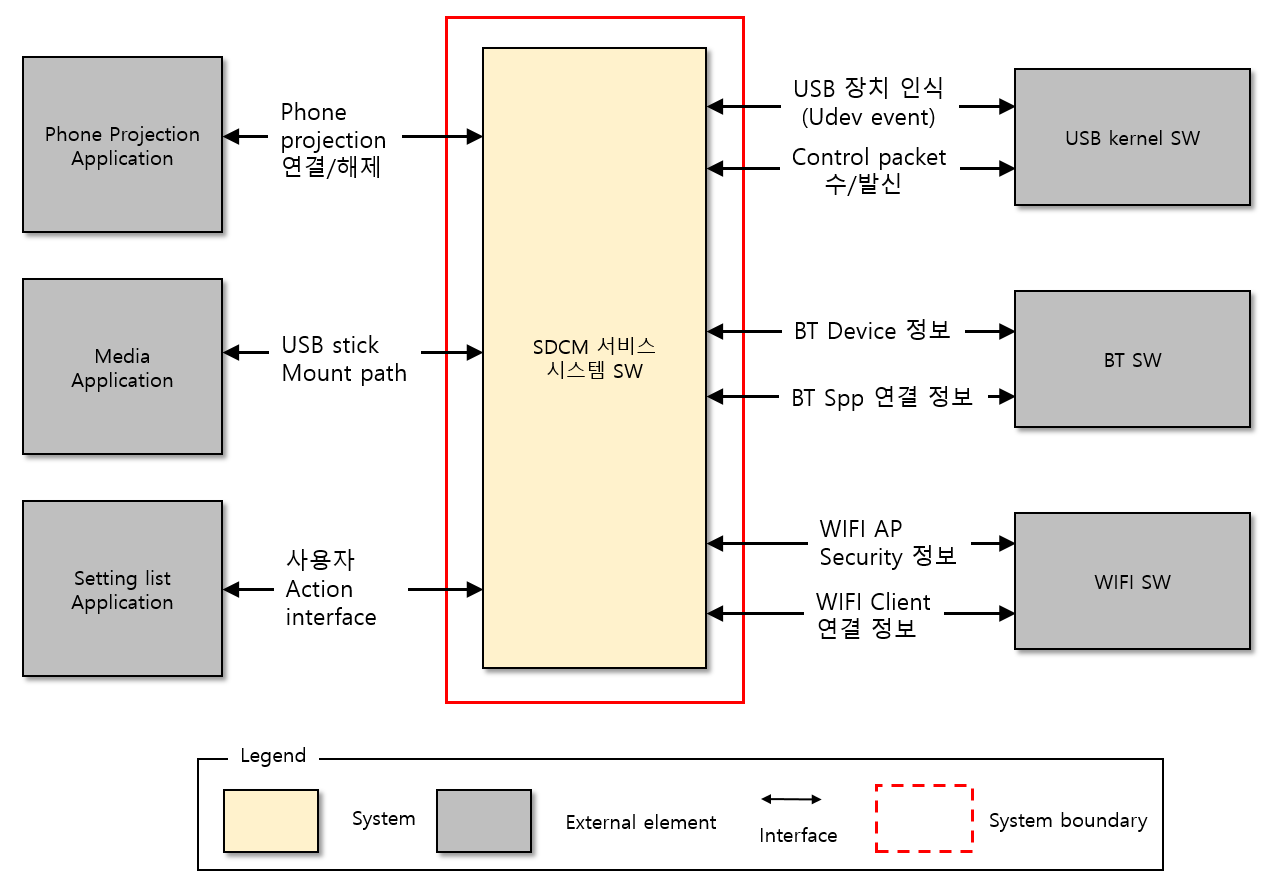


Figure 1‑2 Context diagram

Table 1‑1 Element Responsibility in System Context

|  |  |
| --- | --- |
| Element | Responsibility |
| Phone Projection  Application | - Phone projection 화면 표시를 주관하는 application  - CarPlay / AndroidAuto / CarLife 등의 HMI App  - SDCM 으로부터 연결/ 해제 등의 요청을 받아서 진행함 |
| Media Application | - SDCM 으로부터 USB stick 의 mount path 정보를 획득함  - 획득한 USB mount path 를 이용하여 음악/ 동영상 등을  재생함 |
| Setting list Application | - Phone projection device 장치 목록을 표시하는 기능  - BT 장치 목록을 표시하는 기능  - 사용자의 연결/ 해제/ 삭제 요청 등의 입력을 받아 SDCM  에게 전달하며, SDCM 은 해당 요청을 수행 후 결과 반환 |
| SDCM 서비스 시스템 SW | - USB 연결 장치/ BT 연결 장치 등을 인식  - 해당 장치의 종류를 판별하며, Phone projection 연결 과정을  주관 함  - iPhone 연결시 CarPlay / CarLife 등의 연결을 주관함  - Android phone 연결시 AndroidAuto / CarLife 등의 연결을 주관함  - USB stick 연결시 USB mount 작업 및 정보를 broadcast 함 |
| USB kernel SW | - Linux 에서 제공하는 기능으로 USB port 에 연결되는 장치  정보를 인식하여 Udev event 를 발생시킴 |
| BT SW | - Bluetooth 기능으로 A2DP (음악 재생) / HFP (전화 통화) 등의  profile 외에 BT SPP (Serial Port Profile) 을 연결하여  Wireless Phone Projection 기능에 활용함 |
| WIFI SW | - BT 의 데이터 전송 폭이 낮기에 보다 많은 Data 를  전송하기 위해 WIFI 를 활용함  - Wireless phone projection 을 사용하기 위해 H/U (Head Unit) 은 WIFI AP 모드 상태로 대기해야 함 |

### Stakeholders

Table 1‑2 Stakeholders

|  |  |  |
| --- | --- | --- |
|  | Role | Responsibility |
| Phone Projection Unit | Architect &  Developer | - SDCM 개발  - SDCM Wireless phone projection 설계 |
| AndroidAuto part  CarPlay part | Developer | - SDCM 의 연결 / 해제 요청 대응  - Phone Projection 기능 개발  > AndroidAuto : Google 요구 사항 대응  > CarPlay : Apple 요구 사항 대응 |
| VS / 품질 보증팀 | 검증 | - SDCM 연결 기능 검증 |
| 현대 요구사항 부서 | Spec Requestor | - Spec 배포 및 검토 |
| LG UX 시나리오 팀 | Scenario scribe | - Spec 에 따른 시나리오 정의 및 검토 |

### Task Members

Table 1‑3 Task Members

|  |  |  |
| --- | --- | --- |
| Name | Role | Responsibility |
| 구본훈 책임 | Task Leader  / Developer  / Scrum Master | - SDCM wireless phone projection 기능 설계  - SDCM Task Manager / Task Operator 기능 구현  - Scrum Master |
| 윤영준 책임  김영훈 책임  방기훈 선임 | Task Members  / Developers | - SDCM wireless phone projection 기능 설계 리뷰  - Sprint 진행 및 Scrum review 참여  - SDCM Connection Manager / Message Broker 구현 |

### Scrum 활동 내역

* SDCM wireless phone projection SCRUM 활동 기간 : 2022/12/19 ~ 2023/04/07 (Task 종료)
* Scrum 기간 2주 단위로 총 8개의 Sprint (16주) 진행 완료
* Scrum 기간 내 발행한 commit 수 : 149개 구현 완료
* Scrum 기간 내 수행 ticket 수 : 107개 과제 수행 완료

## 설계 범위

SDCM 의 Wired 기능은 현대 양산 Project 에서 기 개발 및 적용되어 탑재되어 있습니다.

이 문서에서 기술하는 설계는 양산 Project 의 SDCM 대비 Wireless Phone Projection 기능 추가로 인해 내부 설계를 개선한 부분을 기술하였습니다.

현대에서는 아래와 같은 요구를 하고 있습니다.

- 현대 양산된 Project 포함하여 Wireless Phone Projection 적용 필요

> 23년도 적용 목표로 현대 차량에서 Wireless Phone Projection 적용 요청이 인입 된   
상황입니다.

- 양산된 차종에 대해서는 SW update 를 통해 기능 적용

> 기 출하된 H/U (Head Unit) 에 대해서 SW Update 기능을 통해서 기능 업데이트가 필요한 상황이며, 기존 기능 (Wired Phone Projection) 에 영향을 주어서는 안 되는 상황입니다.

Wireless Phone Projection 은 Phone 과 BT / WIFI 를 통해 data 를 주고 받게 되며, 기 구현된 기능 (Wired 처리 기능) 대비 BT, WIFI 등의 정보도 수신 및 활용해야 합니다.

위와 같은 현대 Project 의 Wireless Phone Projection 기능 적용을 위해 SDCM 개선이 필요한 상황이었으며, 보다 용이한 기능 추가를 위해 개선 설계를 진행하였습니다.

## 과제의 핵심 목표

- 양산된 기능 (Wired Phone Projection) 에 영향을 주지 않아야 함  
- 다양한 Project 에 동시 적용에 따른 공용화 필요 (각 Project 별 차이점 지원)  
- 향후 Wireless Phone Projection 과 같이 신규 기능 적용 시 유연한 적용을 위한 설계 필요  
- 양산된 Project 에서의 SDCM 은 위와 같이 외부 Client 와 서로 통신하고 있으며, D-Bus (IPC) 를   
 wrapping 하고 있는 libDCM shared library 를 제공하고 있습니다. 기존 Client 에는 영향을 주지   
 않도록 설계 필요 (개선 설계를 적용하더라도 외부 interface 변경은 없음)  
- Wireless phone projection 기능 대응 및 향후 신규 기능 추가를 위해 SDCM 구조 개선 진행

# Architect Driver

## Functional Requirements

Table 2‑1 Functional Requirement

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **FR ID** | **SW Requirements** | **Priority\*** |
| USB Device | FR-001 | SDCM은 USB 장치 연결 여부를 확인할 수 있어야 한다. | Middle |
| FR-002 | SDCM은 USB 장치 해제 여부를 확인할 수 있어야 한다. | Middle |
| FR-003 | SDCM은 USB 에 연결된 장치 종류(Android/Apple/etc.)를 파악할 수 있어야 한다. | Middle |
| FR-004 | SDCM은 한 개 이상의USB 연결 장치를 지원해야 한다. | Low |
| iPod | FR-005 | USB에 연결된 장치가 Apple인 경우, SDCM은 iPod 이 연결될 수 있도록 iAP2 service 에 연결 요청 할 수 있어야 한다. | Middle |
| FR-006 | USB에 해제된 장치가 Apple인 경우, SDCM은 iPod이 해제될 수 있도록 iAP2 service 에 해제 요청 할 수 있어야 한다. | Middle |
| CarPlay | FR-007 | USB에 연결된 장치가 Apple인 경우, SDCM은 CarPlay이 연결될 수 있도록 iAP2 service 에 연결 요청 할 수 있어야 한다. | Middle |
| FR-008 | USB에 해제된 장치가 Apple인 경우, SDCM은 CarPlay 이 해제될 수 있도록 iAP2 service 에 해제 요청 할 수 있어야 한다. | Middle |
| AndroidAuto | FR-009 | USB에 연결된 장치가 Android인 경우, SDCM은 MTP 로 mount 처리 후 해당 정보를 App 쪽으로 전달할 수 있어야 한다. | Low |
| FR-010 | USB에 해제된 장치가 Android인 경우, SDCM은 MTP 로 un-mount 처리 후 해당 정보를 App 쪽으로 전달할 수 있어야 한다. | Low |
| FR-011 | USB에 연결된 장치가 Android인 경우, SDCM은 Android Auto를 연결할 수 있도록 AndroidAuto core 에 연결 요청 할 수 있어야 한다. | Middle |
| FR-012 | USB에 해제된 장치가 Android인 경우, SDCM은 Android Auto을 해제 할 수 있도록 AndroidAuto core 에 해제 요청 할 수 있어야 한다. | Middle |
| CarLife | FR-013 | USB에 연결된 장치가 Apple또는 Android인 경우, SDCM은 CarLife을 연결 할 수 있도록 CarLife core 에 연결 요청 할 수 있어야 한다. | Low |
| FR-014 | USB에 해제된 장치가 Apple또는 Android인 경우, SDCM은 CarLife을 해제 할 수 있도록 CarLife core 에 해제 요청 할 수 있어야 한다. | Low |
| BT | FR-015 | SDCM은 BT On/Off 상태를 확인하여 off 인 경우 BT 연결 요청에 대해서 reject 할 수 있어야 한다. | Middle |
| FR-016 | SDCM은 BT 장치 연결 여부를 확인하여 Device 정보를 업데이트 할 수 있어야 한다. | Middle |
| FR-017 | SDCM은 BT 장치 해제 여부를 확인하여 Device 정보를 업데이트 할 수 있어야 한다. | Middle |
| FR-018 | SDCM은 BT 장치의 종류 (순수 BT장치/ iPhone/ AndroidAuto phone) 를 판단하여 Device 정보를 업데이트 할 수 있어야 한다. | High |
| FR-019 | SDCM은 한 개 이상의BT 연결 장치를 지원해야 한다. | Middle |
| FR-020 | BT에 장치가 연결된 경우, SDCM은 Wireless Phone Projection 사용을 위해 SPP을 연결 할 수 있어야 한다. | High |
| FR-021 | BT에 장치가 연결된 경우, SDCM은 Wireless Phone Projection 연결을 위해 사용한SPP을 해제 할 수 있어야 한다. | High |
| WIFI | FR-022 | SDCM은 WIFI On/Off 상태를 확인하여 Wireless Projection 가능한 상황인지 판단할 수 있어야 한다. | High |
| FR-023 | SDCM은 WIFI 대역폭(2.4G/5G)상태를 확인하여 특정 지역 진입 시 Wireless Phone Projection 사용 가능 여부를 결정할 수 있어야 한다. (특정 지역에서 5GHz WIFI band-width 사용 불가 등의 제약 조건 있음) | High |
| FR-024 | SDCM은 WIFI 연결된 장치의 상태를 확인하여 Wireless Phone Projection 연결 관리할 수 있어야 한다. (Wireless Phone Projection 의 device 가 WIFI 가 연결 해제 되는 경우 비정상 종료로 판단) | High |
| iPod | FR-025 | BT로 SPP 연결된 장치가 Apple인 경우, SDCM은 iPod Wireless사용을 위해 iAP2 core 에 연결 요청 할 수 있어야 한다. | Middle |
| CarPlay | FR-026 | BT로 SPP 연결된 장치가 Apple인 경우, SDCM은 CarPlay 사용을 위해 iAP2 core 에 연결 요청 할 수 있어야 한다. | Middle |
| AndroidAuto | FR-027 | BT로 SPP 연결된 장치가 Android인 경우, SDCM은 Android Auto 연결을 위해 AndroidAuto core 에 연결 요청 할 수 있어야 한다. | High |
| Device List App | FR-028 | SDCM은 Application (Setting list app 등) 이 요청하는 Technology (AA/CP/ML 등)을 연결 요청 할 수 있어야 한다.  - Technology 는 상위 Glossary 에 명시함 | Middle |
| FR-029 | SDCM은 Application (Setting list app 등) 이 요청하는 Technology (AA/CP/ML 등)을 해제 요청 할 수 있어야 한다.  - Technology 는 상위 Glossary 에 명시함 | Middle |
| Switching | FR-030 | 한 개 이상의 장치가 BT 또는 USB에 연결된 경우, SDCM은 연결된 장치의 Technology를 전환 요청 할 수 있어야 한다. (Switching 기능 : A device 의 Android Auto 해제 후 B device 의 CarPlay 연결 요청 등) | High |
| Device list | FR-031 | BT또는 USB로 장치가 연결된 경우, SDCM은 장치목록에 장치를 추가할 수 있어야 한다. | Middle |
| FR-032 | BT또는 USB로 장치가 해제된 경우, SDCM은 장치목록에 장치를 변경할 수 있어야 한다. | Middle |
| FR-033 | 장치의 상태가 변경된 경우, SDCM은 장치목록의 장치를 변경할 수 있어야 한다. | Middle |
| FR-034 | SDCM은 추가/삭제/변경 정보를 Application(Setting list app 등) 에게 전달 할 수 있어야 한다. | Middle |
| FR-035 | SDCM은 장치 목록을 OEM 요청에 맞게 정렬할 수 있어야 한다. - 연결된 마지막 Device 가 최 상단에 보여야 한다. - 사용자가 정의한 Priority 에 따라 목록을  표시하여야 한다.  - BT Pairing 이 되지 않은 Device 라도 Wired 에 연결 시 최 상단 목록에 해당 장치 정보가 표시되어야 한다. | High |
| Reconnect | FR-036 | 시스템이 부팅되었을 때, SDCM은 이전 연결 장치를 재 연결 할 수 있어야 한다. | Middle |
| Recovery | FR-037 | SDCM은 비정상적으로 연결 해제 된 장치를 재 연결할 수 있어야 한다. | Middle |
| FR-038 | 장치 연결 관리를 위해서 여러 고객사의 요구사항과 예외상황을 효과적으로 처리할 수 있어야 한다. | Middle |

\* : High – Wireless Phone Projection 적용 요구사항의 신규  
 FR 항목으로 중요한 신규 구현 기능

Middle – 양산된 SDCM 기능에 기 구현 완료된 기능으로  
 중요도 높은 기능

Low – 양산된 SDCM 기능에 기 구현 완료된 기능으로  
중요도 낮은 기능

## Use Case List

### Use Case Diagram

SDCM 의 Functional requirement를 바탕으로 Use Case를 아래와 같이 Use Case Diagram으로 도출 하였다.

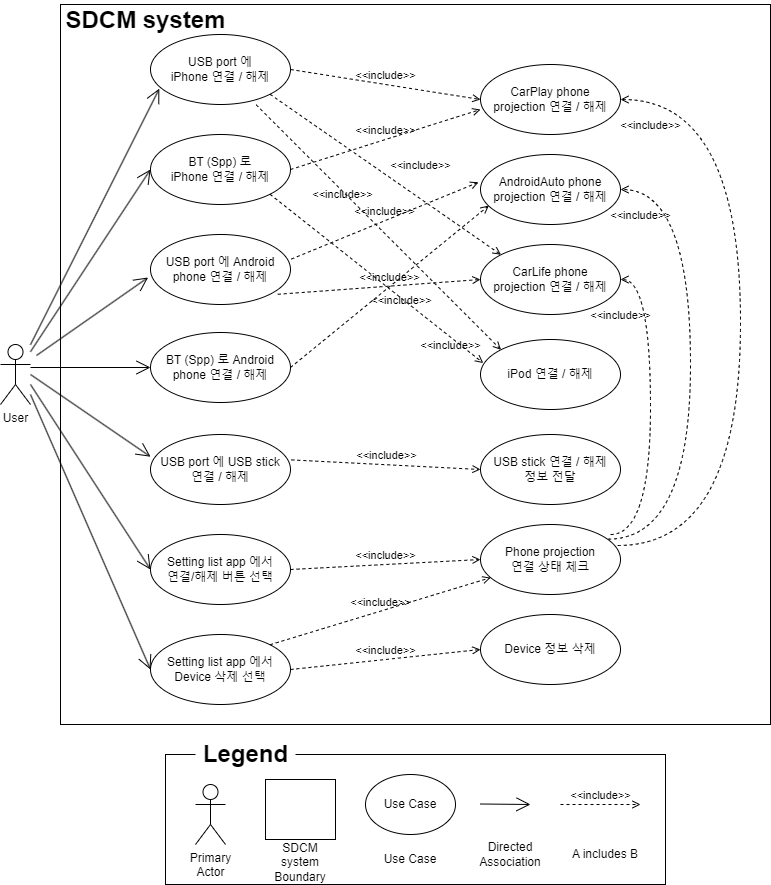


Figure 2‑1 Use case diagram

### Use Case Scenarios

본 절에서는 SDCM 의 각 Use case 에 대한 description 과 Use case diagram 을 기술한다.

#### USB port 에 iPhone 연결/ 해제

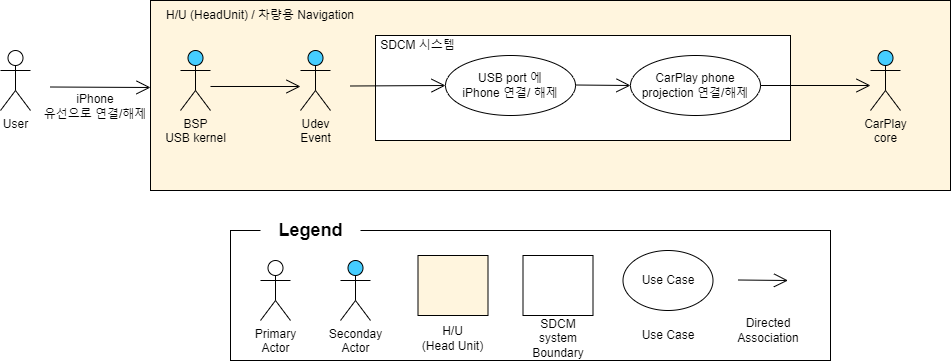


Figure 2‑2 Use case 1 diagram

Table 2‑2 Use case 1 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : USB port 에 iPhone 연결/ 해제 | | Use case ID : UC-001 |
| Primary Actor | User | |
| Secondary Actor | - BSP USB kernel  - Udev event  - CarPlay core | |
| Description | 사용자가 iPhone 을 H/U 의 USB port 에 연결하여 CarPlay phone projection 을 사용할 수 있게 되는 과정입니다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - Phone projection 을 지원하는 USB port 가 있어야 한다. | |
| Main Path | 1. BSP kernel 에서 장치를 인식한다.  2. 인식 / 해제된 장치는 Udev event 가 발생한다.  3. Udev event 를 monitoring 하고 있던 USB 는 해당 장치에 대해서 정보를 받는다.  4. 해당 장치가 어떤 기기인지 식별한다.  5. iPhone 으로 식별된 경우 CarPlay phone projection 이 가능한 상황인지 또는 기 연결되어 사용중인 상태인지 체크한다.  6. CarPlay phone projection 이 가능한 경우 CarPlay core 에게 연결 / 해제 요청을 진행한다.  7. CarPlay core 의 연결 완료 응답을 기다린다. | |
| Post conditions | 연결 완료된 경우 마지막 연결 정보 상황을 업데이트 한다.  연결 실패된 경우 Port reset (Phone 의 vBUS (전원 control) 5V 를 일시적으로 0V 로 낮춤) 하여 재 연결 가능 상황으로 복구함 | |
| Alternative Path | 4a. iPhone 으로 식별되었지만, 사용자가 CarPlay 사용 OFF 등으로 CarPlay 사용 가능한 상황이지 않은 경우  4a.1. SDCM 은 더 이상 연결 진행하지 않고 Idle 상태로 대기함 | |

#### BT(Spp) 로 iPhone 연결 / 해제

Figure 2‑3 Use case 2 diagram

Table 2‑3 Use case 2 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : BT (Spp) 로 iPhone 연결 / 해제 | | Use case ID : UC-002 |
| Primary Actor | User | |
| Secondary Actor | - BT Module  - BT Manager (현대 H/U : vBT)  - CarPlay core | |
| Description | BT (Spp) 가 연결되어 CarPlay Wireless phone projection 이 연결되는 과정이 진행된다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - BT (Spp) 가 사전에 BT pairing 완료 되어야 한다. (Phone <-> H/U 간)  - WIFI 가 AP mode 상태여야 한다. | |
| Main Path | 1. BT Module 에서 BT (Spp) 장치를 인식한다.  2. BT Manager 에서 연결 / 해제 정보를 발생시킨다.  3. BT Manager 정보를 monitoring 하고 있는 SDCM 은 BT (Spp) 의 연결 정보를 받는다.  4. 해당 BT 장치가 어떤 기기인지 식별한다.  5. iPhone 으로 식별된 경우 CarPlay phone projection (Wireless) 이 가능한 상황인지 또는 기 연결되어 사용중인 상태인지 체크한다.  6. CarPlay phone projection (Wireless) 이 가능한 경우 CarPlay core 에게 연결 / 해제 요청을 진행한다.  7. CarPlay core 의 연결 완료 응답을 기다린다. | |
| Post conditions | 연결 완료된 경우 마지막 연결 정보 상황을 업데이트 한다.  연결 실패된 경우 BT (Spp) 연결을 해제한 후 재 연결을 진행하여 연결 가능 상황으로 복구함 | |
| Alternative Path | 4a. iPhone 으로 식별되었지만, 사용자가 CarPlay 사용 OFF 등으로 CarPlay 사용 가능한 상황이지 않은 경우  4a.1. SDCM 은 더 이상 연결 진행하지 않고 Idle 상태로 대기함 | |

#### USB port 에 Android phone 연결 / 해제

Figure 2‑4 Use case 3 diagram

Table 2‑4 Use case 3 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : USB port 에 Android phone 연결/ 해제 | | Use case ID : UC-003 |
| Primary Actor | User | |
| Secondary Actor | - BSP USB kernel  - Udev event  - AndroidAuto core | |
| Description | 사용자가 Android Phone 을 H/U 의 USB port 에 연결하여 AndroidAuto phone projection 을 사용할 수 있게 되는 과정입니다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - Phone projection 을 지원하는 USB port 가 있어야 한다. | |
| Main Path | 1. BSP kernel 에서 장치를 인식한다.  2. 인식 / 해제된 장치는 Udev event 가 발생한다.  3. Udev event 를 monitoring 하고 있던 USB 는 해당 장치에 대해서 정보를 받는다.  4. 해당 장치가 어떤 기기인지 식별한다.  5. Android phone 으로 식별된 경우 AndroidAuto phone projection 이 가능한 상황인지 또는 기 연결되어 사용중인 상태인지 체크한다.  6. AndroidAuto phone projection 이 가능한 경우 AndroidAuto core 에게 연결 / 해제 요청을 진행한다.  7. AndroidAuto core 의 연결 완료 응답을 기다린다. | |
| Post conditions | 연결 완료된 경우 마지막 연결 정보 상황을 업데이트 한다.  연결 실패된 경우 Port reset (Phone 의 vBUS (전원 control) 5V 를 일시적으로 0V 로 낮춤) 하여 재 연결 가능 상황으로 복구함 | |
| Alternative Path | 4a. Android phone 으로 식별되었지만, 사용자가 AndroidAuto 사용 OFF 등으로 AndroidAuto 사용 가능한 상황이지 않은 경우  4a.1. SDCM 은 더 이상 연결 진행하지 않고 Idle 상태로 대기함 | |

#### BT(Spp) 로 Android phone 연결 / 해제

Figure 2‑5 Use case 4 diagram

Table 2‑5 Use case 4 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : BT (Spp) 로 Android Phone 연결 / 해제 | | Use case ID : UC-004 |
| Primary Actor | User | |
| Secondary Actor | - BT Module  - BT Manager (현대 H/U : vBT)  - AndroidAuto core | |
| Description | BT (Spp) 가 연결되어 AndroidAuto Wireless phone projection 이 연결되는 과정이 진행된다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - BT (Spp) 가 사전에 BT pairing 완료 되어야 한다. (Phone <-> H/U 간)  - WIFI 가 AP mode 상태여야 한다. | |
| Main Path | 1. BT Module 에서 BT (Spp) 장치를 인식한다.  2. BT Manager 에서 연결 / 해제 정보를 발생시킨다.  3. BT Manager 정보를 monitoring 하고 있는 SDCM 은 BT (Spp) 의 연결 정보를 받는다.  4. 해당 BT 장치가 어떤 기기인지 식별한다.  5. Android phone 으로 식별된 경우 AndroidAuto phone projection (Wireless) 이 가능한 상황인지 또는 기 연결되어 사용중인 상태인지 체크한다.  6. AndroidAuto phone projection (Wireless) 이 가능한 경우 AndroidAuto core 에게 연결 / 해제 요청을 진행한다.  7. AndroidAuto core 의 연결 완료 응답을 기다린다. | |
| Post conditions | 연결 완료된 경우 마지막 연결 정보 상황을 업데이트 한다.  연결 실패된 경우 BT (Spp) 연결을 해제한 후 재 연결을 진행하여 연결 가능 상황으로 복구함 | |
| Alternative Path | 4a. Android phone 으로 식별되었지만, 사용자가 AndroidAuto 사용 OFF 등으로 AndroidAuto 사용 가능한 상황이지 않은 경우  4a.1. SDCM 은 더 이상 연결 진행하지 않고 Idle 상태로 대기함 | |

#### USB port 에 USB stick 연결 / 해제

Figure 2‑6 Use case 5 diagram

Table 2‑6 Use case 5 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : USB port 에 USB stick 연결 / 해제 | | Use case ID : UC-005 |
| Primary Actor | User | |
| Secondary Actor | - BSP USB kernel  - Udev event  - Media app client | |
| Description | USB stick 을 인식하여 USB 장치를 mount 처리 후 Media app 에 해당 정보를 전달하는 기능 | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - USB stick 지원하는 USB port 가 있어야 한다. | |
| Main Path | 1. BSP kernel 에서 장치를 인식한다.  2. 인식 / 해제된 장치는 Udev event 가 발생한다.  3. Udev event 를 monitoring 하고 있던 USB 는 해당 장치에 대해서 정보를 받는다.  4. 해당 장치가 어떤 기기인지 식별한다.  5. USB stick 장치로 식별된 경우 해당 장치를 mount / un-mount 처리한다.  6. mount / un-mount 후에는 해당 정보를 media app client 에게 mount path 와 함께 전달한다. | |
| Post conditions | 없음 | |
| Alternative Path | 5a. 식별된 USB stick 장치의 partition file system type (exFat/ NTFS/ FAT16/ FAT32 등) 을 식별한 결과 미지원하는 file system type 인 경우  5a.1. INVALID Type error 를 media client app 에게 전달한다.  5a.1.1 media app 에서는 적절한 에러 문구를 사용자에게 표시한다. | |

#### Setting list app 에서 연결 / 해제 버튼 선택

Figure 2‑7 Use case 6 diagram

Table 2‑7 Use case 6 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : Setting list app 에서 연결 / 해제 버튼 선택 | | Use case ID : UC-006 |
| Primary Actor | User | |
| Secondary Actor | - Setting Device list app  - libDCM shared library (D-Bus wrapper)  - Phone projection core | |
| Description | H/U 내의 Setting list app 에서 사용자가 특정 device 를 연결/ 해제 버튼을 선택하면, 해당 Phone projection 의 연결/ 해제를 진행한다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - Phone projection 이 연결되어 있거나, 가능한 상황이어야 한다. | |
| Main Path | 1. User 는 setting list app 에 진입한다.  2. 연결/해제를 진행할 Device 의 연결 / 해제 버튼을 선택한다.  3. 연결/해제를 선택한 Device 정보가 SDCM 으로 인입 된다.  4. SDCM 에서는 해당 장치가 사용 가능하거나 해제 가능한 상황인지 판단한다.  5. 해당 장치가 어떤 Phone projection 을 지원하는지 판단한다.  6. 해당 Phone projection core 에 연결 / 해제 정보를 요청한다.  이후 과정은 UC-001/UC-002/UC-003/UC-004 과정과 동일 | |
| Post conditions | -연결/해제가 완료되면, 해당 State 정보를 setting list app 에 전달한다.  - Setting list app 에서는 해당 정보를 받아서 상태를 업데이트 한다. | |
| Alternative Path | 4a. 사용자가 선택한 장치가 사용 불가능한 경우  4a.1. 사용 불가능한 error code 를 setting list app 에 보낸다.  4a.2. setting list app 에서는 적절한 에러 팝업을 띄워 사용자에게 알려준다. | |

#### Setting list app 에서 Device 삭제 선택

Figure 2‑8 Use case 7 diagram

Table 2‑8 Use case 7 scene

|  |  |  |
| --- | --- | --- |
| Use Case Title : Setting list app 에서 삭제 버튼 선택 | | Use case ID : UC-007 |
| Primary Actor | User | |
| Secondary Actor | - Setting Device list app  - libDCM shared library (D-Bus wrapper)  - Phone projection core | |
| Description | H/U 내의 Setting list app 에서 사용자가 특정 device 를 삭제 버튼을 선택하면, 해당 Phone projection 의 해제 후 삭제를 진행한다. | |
| Preconditions | - H/U 의 전원이 들어와 있어야 한다.  - Phone projection 이 연결되어 있거나, 가능한 상황이어야 한다. | |
| Main Path | 1. User 는 setting list app 에 진입한다.  2. 삭제를 진행할 Device 의 삭제 버튼을 선택한다.  3. 삭제를 선택한 Device 정보가 SDCM 으로 인입 된다.  4. SDCM 에서는 해당 장치가 사용중인 상황인지 판단한다.  5. 해당 장치가 사용 중이라면 해당 Phone projection core 에 연결 / 해제 정보를 요청한다.  이후 과정은 UC-001/UC-002/UC-003/UC-004 과정과 동일  6. Phone projection 이 해제가 완료되면, 해당 Device 정보를 삭제한다. | |
| Post conditions | -삭제가 완료되면, Update 된 device list 를 setting list app 에 전달한다.  - Setting list app 에서는 해당 정보를 받아서 Device 목록을 갱신한다. | |
| Alternative Path | 5a. 사용자가 선택한 장치가 해제 불가능한 경우 (일시적 에러 상황 등)  5a.1. 삭제 요청 수행 불가능한 error code 를 setting list app 에 보낸다.  5a.2. setting list app 에서는 적절한 에러 팝업을 띄워 사용자에게 알려준다. | |

## Quality Attribute

SDCM 서비스 시스템이 만족해야 하는 Quality Attribute Scenario 등과 각 항목별 우선 순위 등을 기술한다.

### Architectural Approach/Analysis

#### [QA-01 Performance] 연결 성능

Table 2‑9 QA-01 Performance scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Performance** | SDCM은 4초 안에 연결 시도 후 장치에서 세션 (신규 연결)을 만들 수 있어야 합니다. |
| Stimulus | 단말 연결 |
| Source(s) of the stimulus | Phone |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 해당 단말에 맞는 Phone Projection 연결 진행 |
| Response Measure | 연결 요청 후 완료 응답을 받기까지 4초 안에 완료되어야 한다. |

#### [QA-02 Performance] 해제 성능

Table 2‑10 QA-02 Performance scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Performance** | SDCM은 세션 연결 해제 후 1초 내에 장치를 사용 가능하게 만들 수 있어야 합니다. |
| Stimulus | 단말 해제 |
| Source(s) of the stimulus | Phone |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 해당 단말에 맞는 Phone Projection 해제 진행 |
| Response Measure | 해제 요청 후 해제 완료 응답을 받기까지 1초 안에 완료되어야 한다. |

#### [QA-03 Performance] 대기 상태 진입

Table 2‑11 QA-03 Performance scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| Performance | SDCM은 누락된 이벤트 없이 모든 연결 이벤트를 처리해야 하고, 1초 이내로 Idle 상태로 진입해야 합니다. |
| Stimulus | 수행 요청 이벤트 발생 |
| Source(s) of the stimulus | SDCM 을 사용하는 외부 Client 요청 |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 해당 요청을 수행한다. |
| Response Measure | 1초 안에 요청 수행이 완료되고 대기 상태로 진입한다. |

#### [QA-04 Performance] 요청 결과 업데이트

Table 2‑12 QA-04 Performance scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| Performance | SDCM은 외부 Client 요청에 대한 결과(FAIL/SUCCESS/FAIL이유)를 1초 이내에 결과 업데이트해야 합니다. |
| Stimulus | 수행 요청 이벤트 발생 |
| Source(s) of the stimulus | SDCM 을 사용하는 외부 Client 요청 |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 외부 요청에 대해서 결과를 업데이트 해야 한다. |
| Response Measure | 1초 이내에 요청 결과를 업데이트 한다. |

#### [QA-05 Interoperability] 외부 Client 에 동일 Interface 제공

Table 2‑13 QA-05 Interoperability scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Interoperability** | SDCM은 외부 환경에 관계없이 애플리케이션에 동일한 인터페이스를 제공해야 합니다. 전 Project 의 interface 는 100% 동일해야 합니다. (다양한 Project 가 진행되더라도 Client 에 동일 interface 제공) |
| Stimulus | Interface 추가 |
| Source(s) of the stimulus | 각 Project 의 요구 사항 |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 전 project 의 interface 는 동일해야 한다. |
| Response Measure | Project 가 다양하더라도 Client 가 사용하는 interface 는 모두 100% 동일해야 한다. |

#### [QA-06 Maintainability] 연결 관리의 독립성 보장

Table 2‑14 QA-06 Maintainability scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Maintainability** | 연결 관리의 유연성과 독립성을 위해 각 장치의 상태 정보를 효과적으로 캡슐화하고 관리해야 하며, 각각의 연결 과정이 100% 독립적으로 운용되어야 합니다. |
| Stimulus | 연결 / 해제 진행 |
| Source(s) of the stimulus | Phone |
| Artifact | SDCM 서비스 시스템 |
| Environment/Context | H/U (Head Unit) |
| Response | 연결 / 해제 정보 관리가 독립적으로 운용이 되어야 한다. |
| Response Measure | Device 별/ Technology 별로 연결 관리에 연결 정보 영향을 받지 않고, 각각 100% 독립적으로 운영이 되어야 한다. |

#### [QA-07 Usability] 공용 부와 변동 부의 분리

Table 2‑15 QA-07 Usability scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Usability** | SDCM은 공용 부 소스를 수정하지 않고, 새로운 Project 가 추가되더라도 변경 사항은 100% 변동 부에서의 변경만 발생해야 합니다. |
| Stimulus | Project 별 신규 기능 추가 |
| Source(s) of the stimulus | Project 별 신규 기능 추가 요구사항 |
| Artifact | SDCM 서비스 시스템 소스 |
| Environment/Context | Project Model 소스 |
| Response | 변동 부에서의 코드 수정으로 신규 기능 추가한다. |
| Response Measure | 공용 부의 코드를 수정하지 않고, Project 별 수정 사항을 변동 부에서만 100% 발생하여야 한다. |

#### [QA-08 Reusability] 공용 부의 library 빌드

Table 2‑16 QA-08 Reusability scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Reusability** | 공용 부는 library 로 독립적인 빌드가 되어야 하며, 변동 부의 코드는 공용 부 library 를 참조하여 구현할 수 있어야 합니다. |
| Stimulus | 공용 부 빌드 |
| Source(s) of the stimulus | 공용 부 소스 |
| Artifact | SDCM 서비스 시스템 소스 |
| Environment/Context | Project 모델 소스 |
| Response | 공용 부만 별개로 빌드가 되어야 한다. |
| Response Measure | 공용 부만 별도 library 형태로 빌드가 되면, 변동 부에서는 Library 사용 형태로 공용 부를 사용하여야 한다. |

#### [QA-09 Modifiability] Project 별 기능 추가 시 공용 부의 독립성 보장

Table 2‑17 QA-09 Modifiability scenario

|  |  |
| --- | --- |
| **Quality Attribute** | **Scenario** |
| **Modifiability** | 신규 Phone Projection 기능을 추가 적용 할 때 공용부의 기존 구조 변경 없이 신규 기능 추가가 가능해야 합니다. |
| Stimulus | Project 별 신규 기능 추가 |
| Source(s) of the stimulus | Project 별 신규 기능 추가 요구사항 |
| Artifact | SDCM 서비스 시스템 소스 |
| Environment/Context | Project Model 소스 |
| Response | 공용부의 구조 변경이 없어야 한다. |
| Response Measure | 공용부의 구조 변경 없이 신규 기능 추가가 변동 부에서만 발생하여야 한다. |

### Prioritization of Quality Attributes

Table 2‑18 Prioritization of quality attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Category** | **QA Description** | **Priority \*1** | **Difficulty \*2** |
| QA.01 | Performance | 연결 성능 | Middle | Middle |
| QA.02 | Performance | 해제 성능 | Middle | Low |
| QA.03 | Performance | 대기 상태 진입 | Middle | Middle |
| QA.04 | Performance | 요청 결과 업데이트 | High | Middle |
| QA.05 | Interoperability | 외부 Client 에 동일 interface 제공 | Middle | Middle |
| **QA.06** | **Maintainability** | 연결 관리의 독립성 보장 | **High** | **High** |
| **QA.07** | **Usability** | 공용 부와 변동 부의 분리 | **High** | **High** |
| QA.08 | Reusability | 공용 부의 library 빌드 | High | Middle |
| **QA.09** | **Modifiability** | Project 별 기능 추가 시 공용 부의 독립성 보장 | **High** | **High** |

\*1 : High – 일정 지연 가능성이 높은 사항  
 Middle – 일정 지연은 없지만, 개선이 필요한 사항  
\*2 : High – SDCM 전체 구조를 개선해야 하는 사항  
 Middle – refactoring 수준에서 대응이 가능한 사항

Low – 간단한 수정으로 진행할 수 있는 사항

우선순위)

* QA-01/ QA-02 의 경우 Performance QA 로 중요한 항목이지만, legacy SDCM 에서 만족하는  
  항목이라서 중요 우선 순위를 낮추었습니다. 이번 설계에서는 신규 기능 (Wireless Phone Projection) 탑재 용이하게 하기 위한 설계에 우선 순위를 두었습니다.

## Constraints

### Business constraints

* 23년 내에 Wireless Phone Projection 기능이 구현되고, AndroidAuto / CarPlay 인증이 완료되어야 한다.
* 양산된 Project 뿐 아니라 현재 개발중인 Project까지 일정상 동시 인증 진행이   
  되어야 한다.

### Technical constraints

* 양산된 Project 에서 SW Upgrade 를 통해 Wireless Phone Projection 기능  
  탑재해야 한다.
* Wireless Phone Projection 기능 추가되지만, 기존 Wired Phone Projection 기능에   
  영향이 없어야 한다.

# Architectural Design

## Layered Architecture

**Decision**

SDCM 시스템은 주요 품질 속성인QA07/ QA09 를 반영하여 Layered Architecture 설계를 진행합니다.

**Reason**

SDCM 에서는 아래와 같은 특성이 있습니다.  
 - 한 Project 에 탑재되어야 하는 특성상 해당 Project 의 특성에 따라 SDCM 내부 코드가  
 변경이 되어야 합니다.

예로, A Project 에는 A 사 BT solution을, B project 에서는 B 사 BT solution 을 탑재함으로  
 인해 서로 수신하는 코드부가 달라지고, 이로 인한 project 별 diff 가 존재하고 있습니다.

- interface 변경으로 인한 SDCM 내부의 코드가 변경이 되어야 하고, 이로 인해 코드  
 전체가 재 검증을 해야 하는 상황들이 발생합니다.

이러한 Project 상황들을 고려할 때 외부와의 interface 들 (변동부) 과 내부 연결 처리 관리 (고정부) 를 분리하는 게 필요한 상황입니다.

Layered Architecture 설계는 상위 layer 에서 하위 layer 만 사용하는 것으로 하위 외부 interface 로 인한 변경 의존도를 상위 layer 에서는 받지 않도록 함입니다.

외부 interface 들을 분리해 냄으로 인해 Project 별 특이 사항들은 변동 부분에서 발생하고,

내부 연결 처리하는 공용 부에서는 변동을 최소화 할 수 있습니다.

## Layer 구성

**Decision**

SDCM 시스템은 외부 interface 를 수신하고 처리하는 External Layer 와 내부 연결  
로직을 담당하는 Service Layer 로 분리하였습니다.  
 1) External layer 에서는 외부의 수신 요소들 (BT / WIFI / Application 요청 / ValetMode 인지 상황   
 / 도난 모드 인지 상황 / SOS Call 상황 등) 다양한 외부 환경의 변화 등을 인지하는   
 layer 입니다.  
 2) Service layer 는 External Layer 상황들을 interface 로 해당 변화 상황에 따라 연결 동작을  
 처리하는 역할을 하게 됩니다.

**Reason**

Project 별 다양한 인지 상황들, 다양한 인지 방법들 (D-Bus / Shared Memory / K-IPC / PCL 등) 에  
 대한 대응은 External Layer 에서 담당하기에 해당 부분을 확장/ 변경/ 추가하면 되며, Service   
 Layer 에서는 정제된 해당 정보들을 처리하면 됩니다.

신규 상황들이 발생하더라도 Service Layer 의 변동은 줄일 수 있습니다.

## Static Perspective

### 1st Decomposition

SDCM 서비스 system 은 다음과 같이 외부와 통신하는 외부 event 들을 수신하는  
External layer 와 내부 SDCM 의 관리를 주관하는 Service layer 로 구성하였습니다.

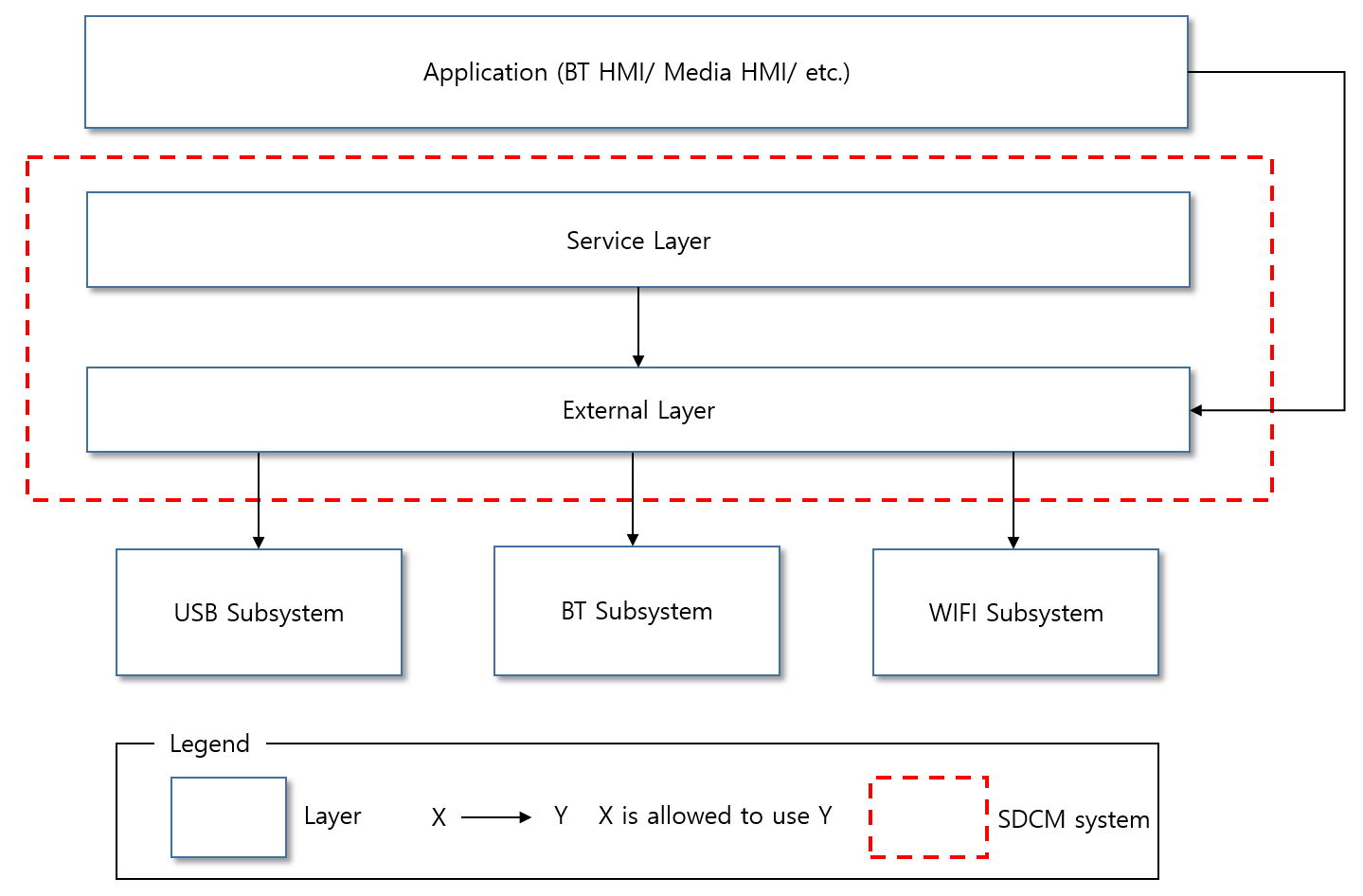


Figure 3‑1 Static view - 1st decomposition diagram

Table 3‑1 Static View - Layer description

|  |  |
| --- | --- |
| **Layer 명칭** | **Layer 역할 정의** |
| **Service Layer** | - 기능 요구 사항 대응  - 연결 진행 상황 관리  - 예외 상황 처리 |
| External Layer | - Project 별 다양한 listening 수단을 처리  - USB kernel 정보 처리  - BT 정보 처리  - WIFI 정보 처리  - Project 별로 같은 BT 지만 BT module 이 달라서 listening 방법이  달라질 수 있으며, 이러한 수신 부를 담당함 |

### 2nd Decomposition

* 모듈 분리 및 기능별 응집도 (Cohesion) 에 따른 구성 – 5개 모듈 도출  
    
  1) External Layer
  + 외부 Client 와 통신 (D-Bus IPC) 하기 위한 Client stub 모듈
  + 내부 발생 이벤트 전달을 위해 적용된 Event Publisher 모듈 **(QA 09 Modifiability 대응)**

2) Service Layer

* + AndroidAuto/ CarPlay 등의 연결 정보 주관하기 위한 Tech Handler 모듈
  + 공용화 적용을 위해 Project 차이점을 관할하기 위한 Deploy Configure 모듈 **(QA 07 Usability 대응)**
  + Functional requirement 에서 도출한 feature 들을 대응하기 위한 Tech Procedure 모듈 **(QA 06 Maintainability 대응)**

#### External Layer

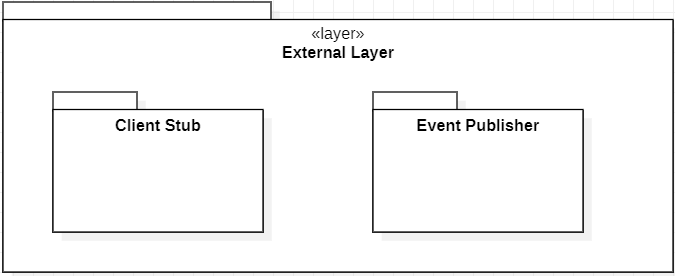


Figure 3‑2 Static view - External layer diagram

Table 3‑2 Static View - External layer description

|  |  |
| --- | --- |
| **Module** | **Description** |
| **Client stub** | - SDCM 기능은 Kernel / BT / WIFI / User action 등에 의해 발생하는 event  들에 대해서 내부에서 활용 후 Client 에 전달해 줘야 합니다.  - 다양한 Client 들이 SDCM 에게 등록하여 상태 정보를 획득하게 됩니다.  이를 관리할 Client stub 이 필요하며, 해당 Client stub 에서는 SDCM 을 사용하는 Client 들에게 정보를 전달할 수도, 요청을 받을 수도 있습니다.  - 현대 Linux platform 에서는 D-Bus 로 Process 간 통신을 하고 있기에 D-Bus 관련한 기능도 함께 구현이 필요합니다. |
| **Event Publisher** | - 다양한 내부 상황과 외부의 요청들을 처리하기 위해서 Event Publisher  (Event 발생기) 가 필요합니다.  - SDCM 내부적으로 event driven 방식에 의해 처리하고 있으며,  이러한 이벤트들을 순차 처리할 수도 있고, 우선 순위에 의해 event 인입을  조절할 수도 있습니다.  - 내부 이벤트 생성 및 전달을 담당할 모듈로 Event Publisher 가 필요합니다. |

#### Service Layer

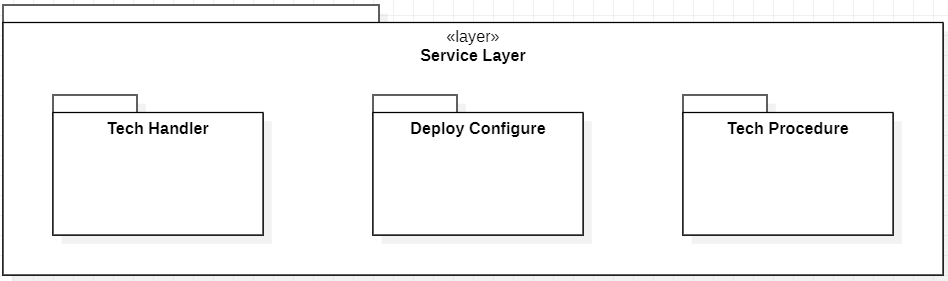
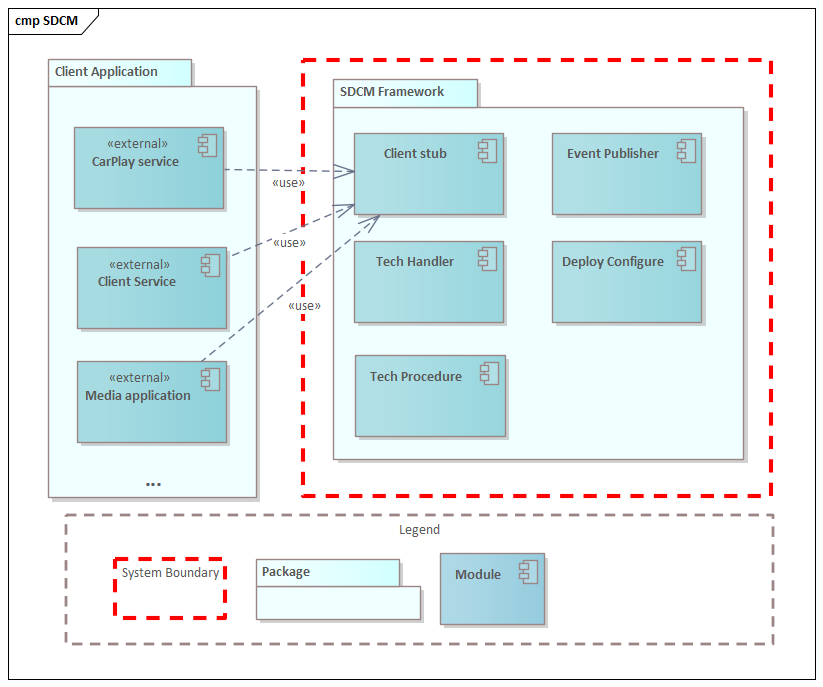


Figure 3‑3 Static view - Service layer description

Table 3‑3 Static View - Service layer description

|  |  |
| --- | --- |
| **Module** | **Description** |
| Tech Handler | - SDCM 은 AndroidAuto, CarPlay, CarLife 등 다양한 Technology 와 Legacy BT  Profile (A2DP : 음악 재생/ HFP : 통화 / PBAP : 주소록 관리 등) 의 연결 상태  체크와 연결 / 해제 요청을 진행할 수 있어야 합니다.  - 동시에 다양한 Technology 가 동시에 연결/ 해제 요청 등을 수행해야 하기에  각각의 상태들을 관리하고, 저장하는 Technology Handler 가 필요합니다. |
| **Deploy Configure** | - SDCM 은 현대, 르노, 폭스바겐, JLR 등 다양한 사업자에 탑재되고 있는 기능입니다. - 사업자마다 요구사항이 다르며, 이러한 사업자 별 차이 나는 부분도  지원해야 합니다. 예로 AndroidAuto 연결의 경우 현대 project 는 Android 단말을 USB (wired) 로 연결하면, AndroidAuto phone projection 연결을 진행하며, Phone projection 을 사용할 수 있습니다. 이때 AndroidAuto 연결 요청 외에 SDCM 에서 진행하는 부분은 없습니다.  반면, 폭스바겐 project 의 경우 Android 단말을 USB (wired) 로 연결하면, AndroidAuto phone projection 연결을 요청한 후 10초 이내에 BT HFP (통화를 위한 BT legacy profile) 를 연결되지 않으면, SDCM 에서 BT HFP 를 연결하러 가야 하는 요구사항이 있습니다.  - 이렇듯 외부의 자극은 Android phone 연결이지만, 내부 동작처리는 사업자마다  상이하며, 이에 대한 지원을 해야 합니다.  그 처리를 위해 Deploy configure (사업자별 차이 지원) 기능이 필요합니다. |
| **Tech Procedure** | - Technology Handler 가 CarPlay, AndroidAuto, CarLife 등 기능의 상태 정보를 저장하였다면, Technology Procedure 는 Technology Handler 를 사용하여 연결/ 해제 등의 판단을 하게 됩니다. 기능 요구사항 중 연결 가능한 상황이 되었을 때 (자동차 부팅 완료 등) Technology 를 연결해야 한다는 요구 사항이 있습니다.  - 자동 연결에 대한 요구사항이며, 이러한 요구 사항들을 처리하기 위해서는 Technology Handler 를 사용하여 이전 연결 이력을 판단하여, 연결 진행 여부를 결정해 줘야 하는 module 이 필요합니다. 이에 Technology Procedure 라는  기능이 필요합니다. |

### Module View

Figure 3‑4 Module View

use

이전 과정에서 식별된 Client Sub / Tech Handler / Tech Procedure / Event Publisher  
Deploy Configure 로 작성하였습니다.

### 설계 Module Role 및 FR 담당 부분 정리

Table 3‑4 Static View - 설계 Module role

|  |  |
| --- | --- |
| **Name** | **Responsibility** |
| Client stub | 외부 모듈과의 통신을 담당  현대 Project 의 경우 IPC (D-Bus) 를 사용하여 통신함 |
| Event Publisher | 외부 모듈의 정보, 내부 모듈의 정보 등은 event 생성하여 정보를broadcast 하는 역할 |
| Tech Handler | 각각의 Technology (CarPlay/ AndroidAuto/ BT) 등의 정보를 관할하기 위해 각각 Service 로 분리되어 있음 |
| Tech Procedure | 연결의 진행 상태를 관리 및 업데이트 하고 있으며, 외부 연결 요청 등에 대응하는 기능 (Feature requirement 와 연관됨) |
| Deploy Configure | Project 별 차이점 나는 부분들은 Policy 로 관리하고 있으며, 각각 Project 별로 차이가 발생하는 부분들을 관리 및 수행 |

3.3.2 절에서 식별된 Module 기준으로 작성하였습니다.

식별된 Module 로 아래와 같이 FR 대응 가능합니다.

Table 3‑5 Static View - Module / FR 연관

|  |  |
| --- | --- |
| **Name** | **FR Responsibility** |
| Client stub | FR-001 : 외부의 장치 연결 Kernel 이벤트 수신  FR-002 : 외부의 장치 해제 Kernel 이벤트 수신  FR-003 : 연결된 장치의 종류 파악  FR-004 : 한 개 이상의 장치 지원  FR-023 : WIFI 정보 인입 및 처리  FR-024 : WIFI 연결된 상태 정보 관리  FR-034 : 장치목록 전달 (Setting HMI 등) |
| Event Publisher | FR-005 / FR-028 : iPod 연결 요청  FR-006 / FR-029 : iPod 해제 요청  FR-007 / FR-028 : CarPlay 연결 요청  FR-008 / FR-029: CarPlay 해제 요청  FR-009 : MTP mount 처리 후 정보 전달  FR-010 : MTP unmounts 처리 후 정보 전달  FR-011 / FR-027 / FR-028 : AndroidAuto 연결 요청  FR-012 / FR-029 : AndroidAuto 해제 요청  FR-013 / FR-028 : CarLife 연결 요청  FR-014 / FR-029 : CarLife 해제 요청 |
| Tech Handler | FR-005 / FR-006 : iPod 정보 저장  FR-007 / FR-008 : CarPlay 정보 저장  FR-009 / FR-010 : MTP 정보 저장  FR-011 / FR-012 : AndroidAuto 정보 저장  FR-013 / FR-014 : CarLife 정보 저장  FR-016 / FR-017 : BT 연결 정보 관리  FR-018 : legacy BT / Wireless PP 관련 연결 정보 관리  FR-025 / FR-026 : iPod wireless 연결 정보 관리  FR-031 / FR-032 / FR-033 : USB 장치 연결/ 해제시 정보 관리 |
| Tech Procedure | FR-005 / FR-028 : iPod 연결 요청  FR-006 / FR-029 : iPod 해제 요청  FR-007 / FR-028 : CarPlay 연결 요청  FR-008 / FR-029: CarPlay 해제 요청  FR-009 : MTP mount 처리 후 정보 전달  FR-010 : MTP unmounts 처리 후 정보 전달  FR-011 / FR-027 / FR-028 : AndroidAuto 연결 요청  FR-012 / FR-029 : AndroidAuto 해제 요청  FR-013 / FR-028 : CarLife 연결 요청  FR-014 / FR-029 : CarLife 해제 요청  FR-015 / FR-019 : BT 연결 관리  FR-020 : BT Spp profile 연결 요청  FR-021 : Wireless phone projection 을 위한 BT Spp 연결 진행  FR-022 : WIFI ON/ OFF 상태 확인  FR-033 : 장치목록 변경 처리 |
| Deploy Configure | FR-030 : Technology Switching 관련 처리  FR-007 / FR-028 : CarPlay 연결 요청  FR-008 / FR-029: CarPlay 해제 요청  FR-011 / FR-027 / FR-028 : AndroidAuto 연결 요청  FR-012 / FR-029 : AndroidAuto 해제 요청  FR-013 / FR-028 : CarLife 연결 요청  FR-014 / FR-029 : CarLife 해제 요청  FR-035 : OEM 요구사항 적용  FR-036 : 이전 연결 장치를 재 연결  FR-037 : 비정상 종료 시 재 연결  FR-038 : 여러 고객사의 요구 사항 적용 |

식별된 모듈들로 FR 요구사항 구현 role 부여 가능함을 확인 완료

## Dynamic Perspective

### 1st Level Decomposition – C&C View

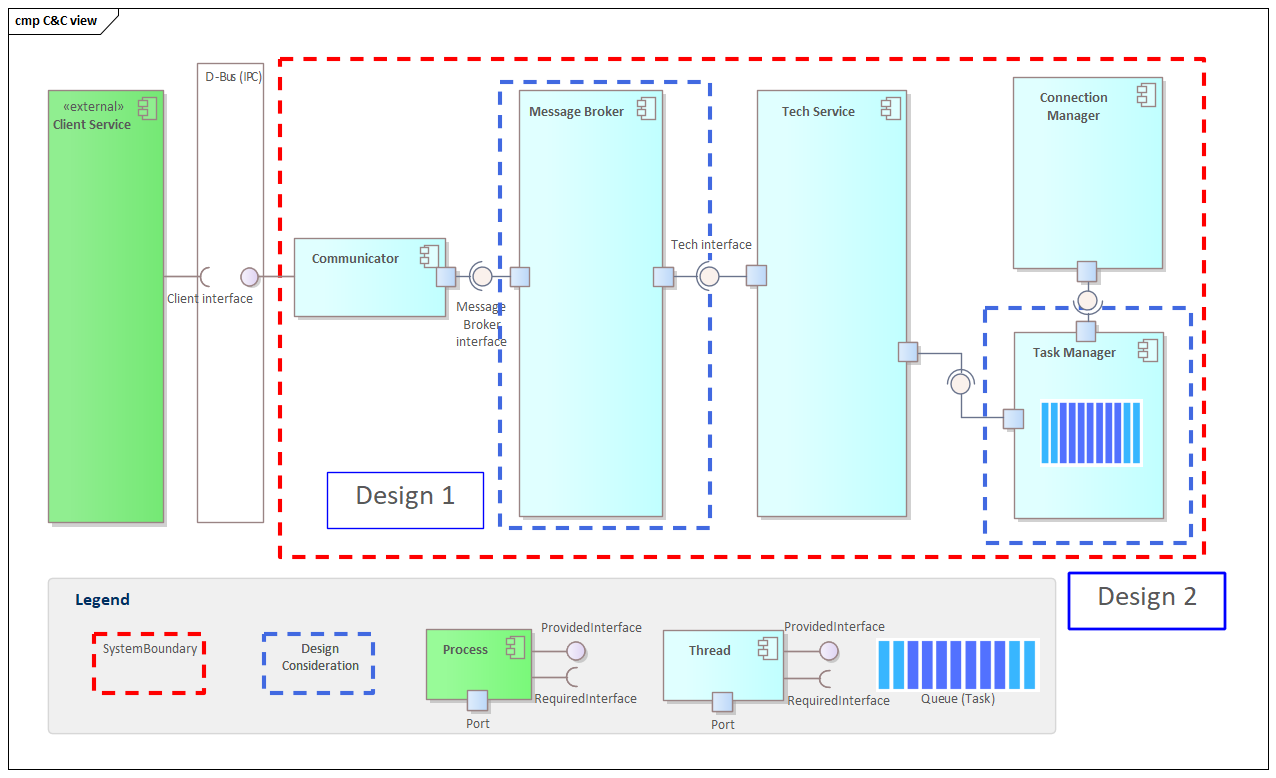


Figure 3‑5 Dynamic view - 1st decomposition

Table 3‑6 Dynamic view - 1st decomposition / responsibility

|  |  |
| --- | --- |
| **Name** | **Responsibility** |
| Communicator  **(**Static view 에서  식별된 **Client stub module)** | 외부에서 오는 D-Bus (IPC) 를 통해 전달되는 정보를 내부에 전달하는 역할 |
| Message Broker  **(**Static view 에서  식별된 **Event Publisher module)** | 내/외부에서 발생하는 event 들을 사전에 등록된 Subscriber 에게 전달해 주는 역할 |
| Tech Service  **(**Static view 에서  식별된 **Tech Handler module)** | AndroidAuto, CarPlay, BT 등 각각의 정보들을 담당하는 service 들 |
| Connection Manager  **(**Static view 에서  식별된 **Tech Procedure module)** | 각 Device 별 연결 상태, 연결 과정 주관 및 상태 업데이트 등을 담당 |
| Task Manager  **(**Static view 에서  식별된 **Deploy Configure module)** | Project 별로 차이가 나는 부분들을 Task 라는 작은 덩어리로 나열하여 Project 별 diff 를 처리하는 역할 |

### 품질 속성 : QA.09 Project 별 기능 추가 시 공용 부의 독립성 보장

* Project 에서 신규 기능 추가 시마다 공용 부가 변동되고 있고, 이로 인해 안정화된  
  타 기능까지도 재 검증을 해야 하는 상황이기에 QA-09 를 개선 과제로 선택하였습니다.

### 설계 대안 : Publish-Subscribe vs Peer to Peer

* 외부/ 내부에서 발생하는 Event 들에 대한 전달 설계사항
  + **Publish-Subscribe** 패턴  
    - 1:N 방식으로 사전에 등록된 subscriber 에게 event 들 전달함
  + **Peer to Peer** 패턴  
    - Target 을 지정하여 해당 module 에게만 event 를 전달함

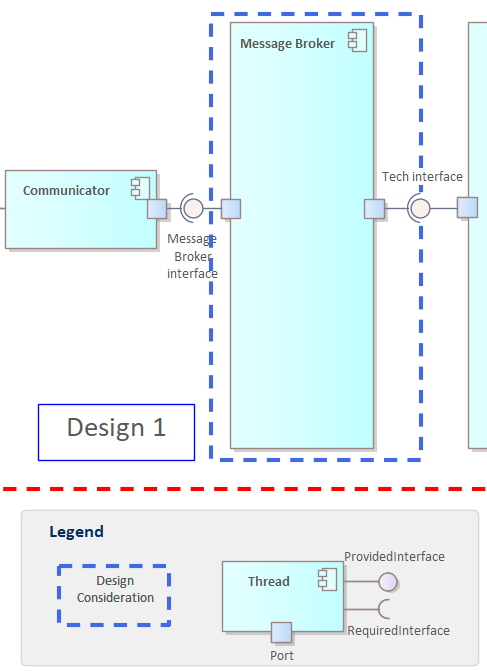


Figure 3‑6 Dynamic view - Design 1

Table 3‑7 Dynamic View – pub-sub, peer to peer

|  |  |  |
| --- | --- | --- |
|  | **Publish-Subscribe** | **Peer to Peer** |
| Pros | 1. 기능 확장에 용이함  2. 신규 기능 적용 시 사전에 필요한 정보들을 subscribe 해 놓으면 다른 기능 영향 없이 정보 수신 가능함 | 1. 사전에 명확하게 전달해야 할 target 모듈에 정보를 전달하므로 오 전달 가능성 없음  2. 중간 전달자가 없으므로 Performance 유리함 |
| Cons | 1. 각 Message 별로 사전에 구분 필요함 (이 설계에서는 Topic/ Subject 개념 도입 필요)  2. Peer-to-Peer 대비 구현 난이도 높음 | 1. 신규 기능 추가 시 발신자 측에 신규 기능 코드 추가 필요 (기존 코드 영향 받음)  2. Layer 분리가 되지 않아 설계 측면에서 Target 대상에 대한 dependency 가 발생함 |

* QA-09 : 신규 Phone Projection 기능을 추가 적용 할 때 공용부의 기존 구조 변경 없이 신규 기능 추가가 가능해야 한다. 를 QA 로 선정하였었습니다.  
    
  해당 기준에 부합한 방식이 Message Broker 방식이라고 생각합니다.  
  기존 SDCM 에서는 신규 기능 추가 시 기존 코드에 영향을 지속적으로 주었습니다.  
  한 예로 과거 경험상 CarLife 라는 신규 기능을 추가할 때 6개월이라는 시간이 소요되었으며,  
  CarLife 신규 기능을 개발하면서 기존 기능 (CarPlay, AndroidAuto) 등의 코드에 영향을 주어  
  의도치 않은 Side effect 등이 발생하였습니다.  
  이러한 Side effect 를 발생한 원인 중 하나는 아래와 같은 상황입니다.  
    
  예로 USB Service 에서 iPhone 을 인식하고, CarPlay 를 연결을 진행하는 코드가 있었습니다.  
  신규 기능을 넣으면서 아래와 같은 if 조건들이 추가되었으며, 이러한 코드들이 지속 늘어나고 있었습니다.

|  |
| --- |
| Code sample |
| if (Detected Technology == AndroidAuto) {  …  AndroidAuto.process()  } else if (Detected Technology == CarPlay) {  …  CarPlay.process()  } else if (Detected Technology == 신규 기능) {  …  신규 기능.process()  } |

이런 조건 문들로 인해 기존 영향을 받으며, 신규 기능 구현 진행 시마다 기존 코드

영향 여부를 지속 체크해야 하였습니다.

Message Broker 처리시 아래와 같이 동작하게 됩니다.

|  |
| --- |
| Code sample |
| Pre-condition  AndroidAuto : MessageBroker.subscribe(TOPIC::CORE, SUBJECT::ANDROID\_AUTO)  CarPlay : MessageBroker.subscribe(TOPIC::CORE, SUBJECT::CARPLAY)  신규 기능 : MessageBroker.subscribe(TOPIC::CORE, SUBJECT::신규 기능)  AndroidAuto 장치 인식 시점  MessageBroker.publish(TOPIC::CORE, SUBJECT::ANDROID\_AUTO) |

위와 같이 기존에는 Target 을 지정해 주어야 하기 때문에 (1:1 지정 방식)

Sample code 와 같이 if, else if, else 등의 코드로 인식되는 Technology 를

판단하였다면, 현재는 인식하는 단말에 대해서 TOPIC 과 Subject 로 publish 만 하면 되고,

Subscriber 도 사전에 내가 관심 있는 Topic 과 subject 로 등록을 해 놓았기에

그 정보가 도착하면 그대로 쓰면 되는 상황입니다.

Subscriber 입장에서는 도착하는 정보는 내가 사전에 관심 있다고 등록해 놓은

Event 만 받을 수 있기에 신규 기능 처리에 있어서 타 기능이 영향을 받지 않게 됩니다.

QA09 만족을 위해 Message Broker 처리가 1차적으로 기능 구현 시 기존 기능 영향 받는

부분을 제거하였습니다.

단점으로는 Message Broker 라는 모듈을 구현해야 하고, TOPIC/ Subject 등을 사전에

용도에 맞게 잘 나눠놔야 합니다.

하지만, 한번 Message Broker 를 설계하고 나면, 기존 기능들은 TOPIC/ Subject 로

메시지 처리를 할 수 있게 되며, 신규 기능이 생긴다면 아래와 같이 처리하면 됩니다.

- TOPIC::신규 기능 / SUBJECT::신규 기능

위와 같이 신규 기능에 대해 TOPIC, SUBJECT 를 정의하여 구현하면, 기존 기능들은

신규 기능의 메시지 (Topic/ Subject) 를 subscribe 를 한 적이 없기에 영향을 받지 않고,

신규 기능만 해당 메시지들을 받게 되어 기존 기능 영향력을 줄일 수 있는 효과가 있습니다.

이에 Message Broker 처리안으로 진행하였습니다.

### 설계 결정 및 근거 : Publish-Subscribe 결정

* 설계 결정 안 : **Publish-Subscribe** 패턴
  + 신규 기능을 추가하는 Wireless Phone Projection 에 대응해야 하는 상황에 적합함
  + 추후 기능 추가 시 기존 기능에 영향 없이 신규 기능 추가 가능함  
    (**QA 09 Modifiability** 요구사항 만족)
  + Message Broker 적용으로 인한 성능 저하는 미미한 수준 (측정 결과 1 ms 이내)

Table 3‑8 Dynamic View - Message Broker, Communicator

|  |  |  |
| --- | --- | --- |
| **Tactics** | **중개 역할 모듈 (Message Broker)** | **외부 상황 전달 모듈 (Communicator)** |
| **Reduce Coupling** | 1. 사전에 등록된 Subscriber 에게 publish 된 event 전달하는 역할 수행  2. Topic / Subject 로 사전 subscribe 하기에 구체화된 대상에 대해서 dependency 관계를 갖지 않음 | 1. 외부에서 전달된 event 들을 Topic / Subject 로 구분하여 Message broker 로 publish 함  2. 1) 번의 event 들이 어떤 모듈이 받아갈지 Communicator 에서는 구체화되지 않음 (Coupling 저하 효과) |
| **Increase**  **Cohesion** | 전달된 Topic / Subject 별로 Subscriber 에게 전달하며, Subscriber 는 Topic/ Subject 처리별로 응집하여 구현 (Tech Service) | Topic / Subject 별로 Event 를 분류하며, 응집하여 전달함 (Publisher 역할) |

* Message Broker 로 publish/ subscriber 를 두어 event 처리의 주체를 분리하였습니다.  
  신규 기능 추가 시 기존 기능에 영향을 받지 않도록 분리한 사항입니다.  
  해당 설계 안에 따라 TOPIC/ SUBJECT 등을 구분하고, 기 구현된 기능에 대해서는  
  TOPIC/ SUBJECT 로 사전 등록이 필요하며, 신규 기능 추가 시에는 TOPIC/ SUBJECT 들을

신규 정의하여 기 구현된 코드에 영향을 주지 않을 수 있습니다.

### Risk 식별 및 개선 전략 : Publish-Subscribe

* **Risk**  
   다양한 Technology 적용시 관리해야 하는 subscriber 들이 많아짐
  + Publish-Subscribe 안으로 적용 시 일반적으로 관심 항목으로 등록을 하는 게  
    범용적으로 사용하는 방식이었습니다.
  + Message Broker 등의 방식이 범용적으로 사용하는 publish-subscribe pattern 의  
    한 종류로 판단이 되었고, 해당 안으로 진행하고자 하였습니다.

그러나, 일반적으로 1차원적인 등록 방식이기에 여러 개의 Technology 를  
적용 시에는 관심 항목들의 enum 을 확장해야 하였고,  
이에 대한 subscriber pool 관리도 1차원적으로 설계가 되었어야 했습니다.

추후 확장성 및 사용성에 문제가 있다고 판단이 되었습니다.

* **개선 전략**
  + 범용적으로 사용하는 1차원 등록방식이 아닌 2차원 방식으로 등록하도록 개선 설계 진행하였습니다.
  + 1차원 방식 : Technology\_AndroidAuto  
     Technology\_CarPlay  
     Technology\_BT\_HFP  
     Technology\_BT\_A2DP
  + 2차원 방식 :

|  |  |
| --- | --- |
| **Topic** | **Subject** |
| TOPIC\_CORE | ANDROIDAUTO  CARPLAY  MIRRORLINK  IPOD |
| TOPIC\_BT | HFP  A2DP  PBAP  SPP |

### 세부 설계 QA.09 Design 1. Message Broker design

* 적용 패턴 : **Publish-Subscribe** 패턴
  + Topic/ Subject 로 Event 를 Category 함

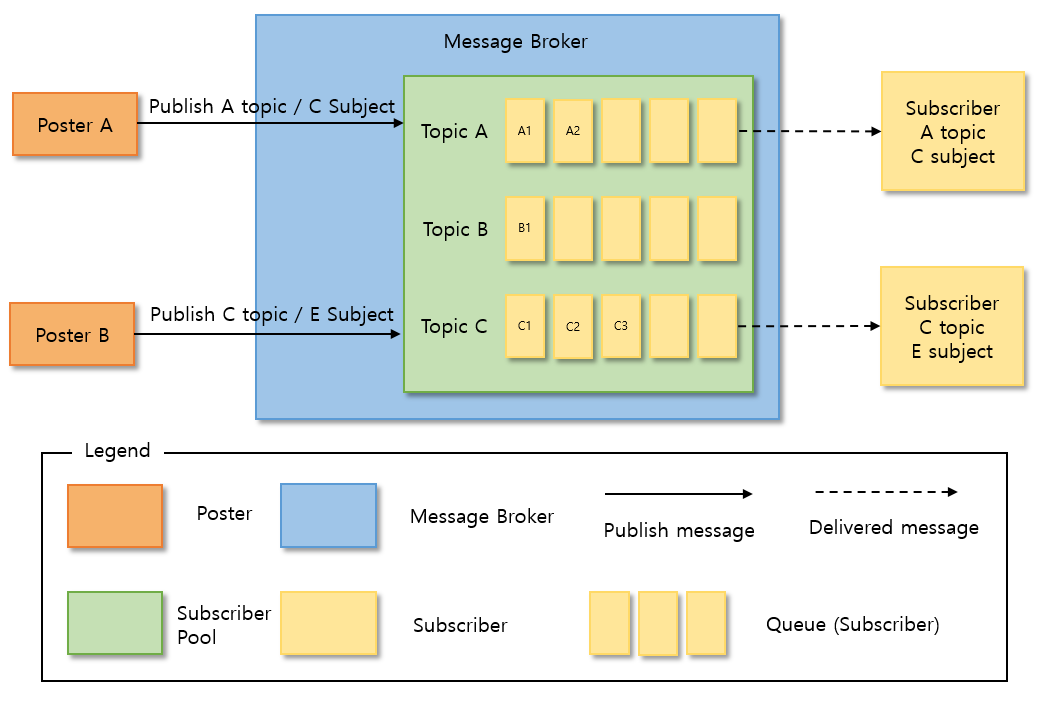


Figure 3‑7 Dynamic view - Message Broker

Table 3‑9 Dynamic View – Topic / Subject

|  |  |
| --- | --- |
| **Topic** | **Subject** |
| TOPIC\_CORE | ANDROIDAUTO  CARPLAY  MIRRORLINK  IPOD |
| TOPIC\_BT | HFP  A2DP  PBAP  SPP |
| TOPIC\_WIFI | AP\_STATUS  LINKLOSS |
| TOPIC\_USB | MTP  STORAGE |

> Message Broker 는 사전에 관심 있는 주제 (Topic/ Subject) 를 사전에 등록된 subscriber 에게 메시지 발생 시 전달하게

됩니다.

예로 Android Auto service 의 경우

Topic : TOPIC\_CORE / Subject : ANDROIDAUTO 로

Message Broker 에 사전에 등록을 해 놓으면,

Android 단말 연결 시 Android Auto service 에 메시지가

전달되게 됩니다. 반면, CarPlay service 에서는 해당 정보를

수신 받지 않게 됩니다.

### 품질 속성 : QA.07 Usability 공용 부와 변동 부의 분리

* Project 마다 동작의 차이가 다양하게 발생하고 있습니다.  
  예로 AndroidAuto 연결을 할 때, A project 는 AndroidAuto 연결만 하면 되지만,  
  B project 는 AndroidAuto 연결 후 10초 대기 후 HFP 연결 하는 동작을 해야 합니다.  
  이에 대한 유연한 대응을 할 수 있도록 QA 로 선정하였습니다.

### 설계 대안 : Maintain Task Model (Tactic) vs Toggle Features (Tactic)

* Project 별 차이 나는 사항들에 대해서 적용 가능한 Tactics
  + **Maintain Task Model**  
    - Task 안에 사용자 의도의 context 들을 저장한 후 수행하는 방식
  + **Toggle Feature**- 기능을 ON/ OFF 하여 의도적으로 필요한 기능을 사전에 활성화 / 비활성화함

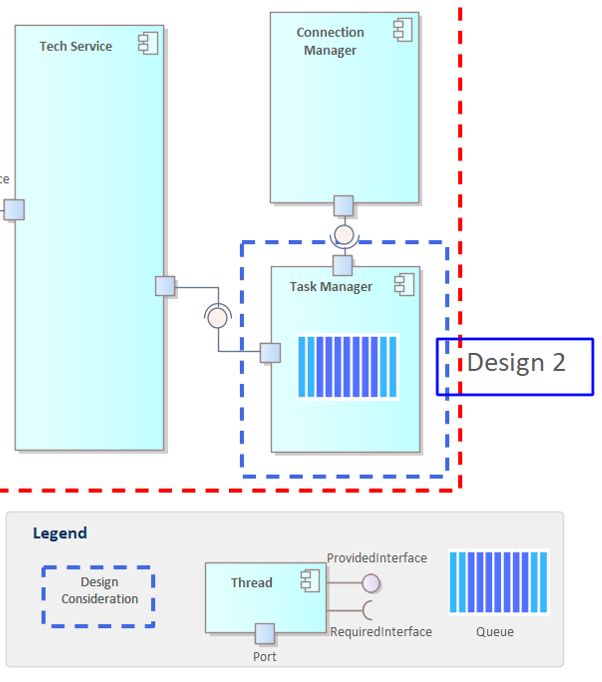


Figure 3‑8 Dynamic view - Design 2

Table 3‑10 Dynamic view - Maintain Task Model / Toggle feature

|  |  |  |
| --- | --- | --- |
|  | **Maintain Task Model** | **Toggle Feature** |
| Pros | 1. 공용 부에서는 변동 부에서 생성한 Task 를 수행하면 되어, 공용화 가능함  2. 각각의 Project 간의 차이점 나는 부분들에 대응 가능함 | 1. 구현이 쉬움 (간단하게 if define (#ifdef) 또는 if - else 구문 등으로 기능 feature ON/OFF 가능함)  2. 일반적으로 사용하는 방식이라서 대부분의 개발자가 코드 이해도가 높음 |
| Cons | 1. 작은 Task 형태로 일거리를 사전에 정의하여야 함  2. 해당 Task 안에 context 를 포함한 정보가 모두 포함되어야 함 | 1. Runtime 중에 원하는 대로 feature toggle 구현이 어려움  2. 공용화 코드에도 featuring 으로 인한 코드 가독성 저하 |

- QA07 : SDCM은 공용 부 소스를 수정하지 않고, 새로운 Project 가 추가되더라도 변경

사항은 100% 변동 부에서의 변경만 발생해야 합니다.

위의 사항을 대응하기 위해 설계한 부분입니다.

공용 부에서는 변동 부에서 만들어지는 Task 만 수행하면 됩니다.

변동 부에서는 수행해야 하는 일거리들을 Task 를 구성하여 담아주면,

공용 부에서는 해당 Task 를 수행하면 되게 됩니다.

즉, 아래와 같이 recipe 개념이 도입 되었습니다.

목표는 AndroidAuto phone projection 연결입니다.

그러나, 각 사업자/ 향지별로 동작이 다르게 됩니다.

* 현대 Project 의 경우에는 USB 연결 인식 > Android 단말인지 확인

> AndroidAuto 연결 요청

* VW Project 의 경우에는 USB 연결 인식 > Android 단말인지 확인 > AndroidAuto 연결 요청

> 10초 대기 > 10초 대기 동안 BT HFP 가 붙지 않는지 확인

> 붙지 않은 경우 BT HFP 를 연결 요청

위와 같이 목표는 AndroidAuto phone projection 을 사용하기 위해 AndroidAuto 연결 요청이지만, 사업자마다 요구사항이 달라서 이 부분에 대한 처리가 필요합니다.

Alternative #1. Task 방식으로 수행 (Maintain Task Model)

변동 부에서 Task 를 구성하고, 공용부에서 구성된 Task 를 수행

|  |
| --- |
| Code sample |
| 공용 부  변동 부에서 사전에 구성되어 올라오는 Task 를 수행  현대)  Task 1 : AndroidAuto 연결  폭스바겐)  Task 1 : AndroidAuto 연결  Task 2 : 10초 대기  Task 3 : HFP 미연결시 HFP 연결 요청 |

Alternative #2. Feature define 으로 수행 (Toggle feature)

변동 부 코드들을 if define 등으로 featuring 하여 수행함

|  |
| --- |
| Code sample |
| 1. AndroidAuto 연결  if defined(폭스바겐)  2. 10초 대기  3. 만약 10초 동안 HFP 연결이 되지 않았다면 HFP 연결 진행  endif |

### 설계 결정 및 근거 : Maintain Task Model Tactics

* 설계 결정 안 : **Maintain Task Model Tactics**
  + Technical constraints 인 6개 project 동시 진행해야 하는 상황에서 적합한 선택임
  + Runtime 시에 project 별 차이 나는 부분들에 대해서 task (context 정보) 를 생성하여 대응 가능함

Table 3‑11 Dynamic view - Design 2 (Task Manager)

|  |  |  |
| --- | --- | --- |
| **Tactics** | **Task Manager** | **Task (Queue)** |
| Maintain Task Model | 1. Task Manager 는 Task (context 정보) 들을 포함하고 있으며, Task 들은 Task Manager 내부에 dequeue 으로 보관하고 있음  2. 순차적으로 Project 별로 구성된 Task 들을 수행하는 역할을 함  3. Task Manager 에서는 각각의 Task 생성 및 관리하는 역할을 함 | 1. Task 는 작은 일거리이며, Context 정보를 포함함  2. 각각의 Project 마다 Task 들이 다를 수 있으며, 해당 Task 들은 Task Operator 내의 dequeue 에 쌓여서 순차적으로 처리하게 됨  예) 특정 연결 동작 진행 시  Project 현대 5세대 : Task A - Task B -Task C 수행  Project 현대 6세대 : Task B - Task C -Task D 수행 |

> 각 Project 별로 수행해야 하는 일거리들을 작은 일거리 단위 (Task) 로 구성한 후

해당 Task 들을 수행하게 하면, Project 별로 차이 나는 부분들을 지원할 수 있습니다.

6개의 현대 Project 들을 동시에 진행해야 하며, 현대 Project 내에서도 차이가 나는

부분들이 있어서 이러한 diff 부분들에 대해서 Task list 구성을 변경함으로 대응할 수

있습니다.

### Risk 식별 및 개선 전략 : Maintain Task Model Tactics

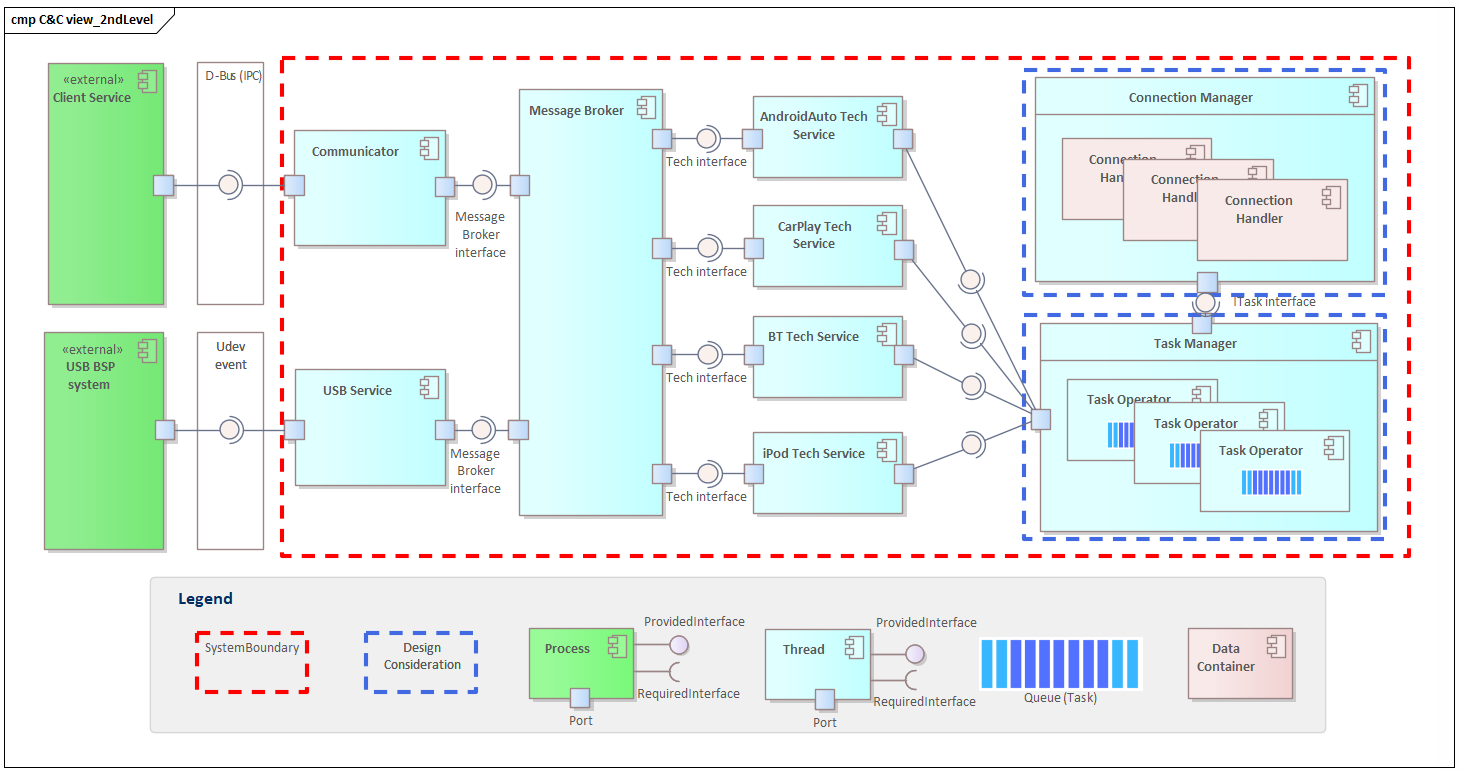
* **Risk**   
   다양한 Project 가 진행되어야 하는 상황에서 Maintain Task Model Tactics 를

적용하고자 합니다.  
 Risk 식별시 Task 를 구성하는 구성 단위를 판별하는 데 어려움이 발생하였습니다.  
 Task 단위로 Project 들의 업무를 쪼개야 하였고, 해당 업무들의 작은 단위들로  
 Task 를 구성해야 하기에 이에 대한 판단이 어려웠습니다.  
 Task 의 묶음을 큰 단위로 할 경우 Project 마다 달라지는 부분에 대한 처리가  
 어려웠으며, Task 의 묶음을 작은 단위로 할 경우 수많은 Task 들의 수행이 필요한  
 Trade-off 지점이 발생하였습니다.

* **개선 전략**  
   Task 의 기준점이 필요하였습니다.  
   Task 단위의 기준점을 아래와 같이 정의하였습니다.  
   1) Client 에서 요청하는 Job 단위로 1차 Task 분류  
   2) 해당 Task 내에서 Project 별로 공통적으로 수행해야 하는 부분과  
   Project 별로 차이가 나는 부분에 대한 2차 Task 분류  
   3) 2차 Task 분류 중 Client 와 통신이 필요한 부분의 경우 해당 단위로 재 분류

위와 같이 총 세 차례의 Task 분류 과정을 통해 적당한 Job 들의 구성으로  
 분류가 완성되었으며, 해당 진행 과정으로 Risk 를 보완하였습니다.

### 2nd Level Decomposition – C&C View



**Design 4**

**Design 3**

Figure 3‑9 Dynamic view - 2nd decomposition

* 2nd Level Decomposition C&C View

: 앞장에서 도식화된 Tech Service, Task Manager, Connection Manager 를

세분화하여 표시함

* + Design 3. Task Manager
  + Design 4. Connection Manager
  + 단, Communicator 와 USB Service 는 기존 양산된 Project 코드 재활용 및  
    Refactoring 진행으로 설계 명시하지 않았습니다.

### 품질 속성 : QA.07 Usability 공용 부와 변동 부의 분리 (2nd Level)

1st Decomposition 단계에서 QA.07 을 만족하기 위해 Maintain Task Model (TaskManager)를

선정하였습니다. 2nd Level 에서는 해당 QA 요구사항을 구체화하기 위해 Task Manager   
내부를 설계하였습니다.

### 설계 대안 : Task + Event driven vs Task + Thread wait

* **Task + Event driven** vs **Task + Thread wait (TaskOperator)**

Task Operator 내의 Task 진행 과정의 순차 처리를 위한 설계  
 - Queue : 처리해야 할 Task 들의 자료 구조

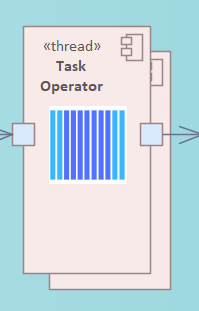
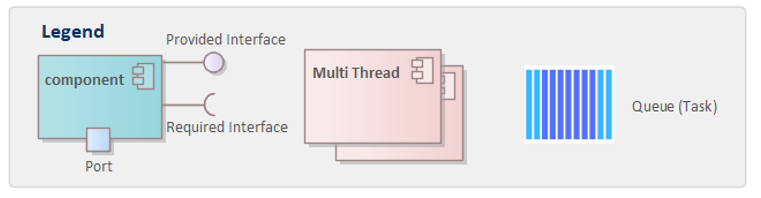
 

Figure 3‑10 Dynamic view - Task Operator

Table 3‑12 Dynamic View - Design 3 (Task Manager)

|  |  |  |
| --- | --- | --- |
|  | **Task + Event driven** | **Task + Thread wait (TaskOperator)** |
| Pros | 1. Client 에서 보내는 Event 발생시마다 해당 event 에 따라 순차 처리를 함  2. Event 누적된 순차적으로 처리하기에 Event 순서에 신경 쓸 필요 없음 | 1. Task list 를 각각의 TaskOperator 가 가지고 있기에 Task 의 순서대로 연결 진행을 하면 되기에 명료함  2. Event 도착시 다음 수행해야 하는 일들을 if 조건 등으로 찾을 필요가 없어서 코드 간소화됨  3. 여러 요구사항들도 병렬 진행이 가능함 |
| Cons | 1. Event 도착시 다음 동작에 대한 고민이 필요함  2. 1번 과정으로 인해 여러 상황들에 대한 if 조건들이 필요함 | 1. Task 처리 / Thread 처리 등으로 인해 구현이 복잡함  2. Task 라는 작은 일거리 (Context) 로 정의 및 수행하도록 분배가 필요함 |

Task 로 작은 일거리를 나눈 이후에 해당 Task list 를 처리해야 하는 상황입니다.

하나의 Task list 로 관리할 경우 여러 개의 연결 진행 상황을 병렬로 수행 할 수가 없습니다.

또한 Task 완료 결과가 도착 시 다음 Task 를 꺼내어 일거리를 진행해야 하는 데,

이에 대한 판단의 Business logic 이 들어가게 됩니다.

그러므로, Event 처리 방식으로는 Task 관리에 어려움이 있습니다.

그 대안으로 Task Operator 가 설계되었습니다.

Task Operator 하나는 내부에 수행해야 하는 Task list 를 갖고 있으며,

Task actor 로부터 수행 결과를 받으면, Task Operator 내부의 다음 Task 를 꺼내어 수행하면 됩니다.

Task Operator 내부의 Task list 는 요청 목표를 달성하기 위해 구성된 Task 들이므로 모두 수행되어야 하는 Task 들입니다.

그리고, 각각의 Task Operator들은 각각의 요청 사항들을 수행하는 Operator 들이며, 모두 Thread 로 구성되어 각각 다중 요청 사항들을 처리하게 설계되었습니다.

즉, A phone 의 Android Auto 를 연결하면서, B phone 의 iPod wireless 를 연결할 수 있습니다.

서로 영향을 주지 않는 연결 과정이므로 각각 Task Operator 가 생성되며,

각각의 Task Operator 내에는 각각 Android Auto 연결을 진행하기 위한 Task List,

iPod wireless 연결을 진행하기 위한 Task List 들을 관리하게 됩니다.

Task Operator 는 하나씩 Task 를 꺼내어 Task actor 에게 수행하는 역할을 하게 됩니다.

이 과정을 통해 QA. 07 의 요구 사항을 달성할 수 있었습니다.

### 설계 결정 및 근거 : Task + Thread wait

* 설계 결정 안 : **Task + Thread wait (TaskOperator)**
  + Task Operator 안에서 각각의 Task 들을 순차적으로 처리하며,  
    Client 쪽 응답을 기다려야 하는 경우  
    Thread wait 처리로 간단하게 대응 가능함 **(QA 07 Usability 대응)**
  + **예)** 현대 Project : Connect AA Task 만 존재하며, 해당 Task 만 수행  
     VW Project : Connect AA Task 수행 이후에 Thread Wait 으로 일정  
     시간 대기 후 Connect HFP Task 수행
  + 동일한 AndroidAuto 연결 과정에서 Project (현대향/ VW 향 등) 의  
    요구 사항을 모두 만족시킬 수 있음

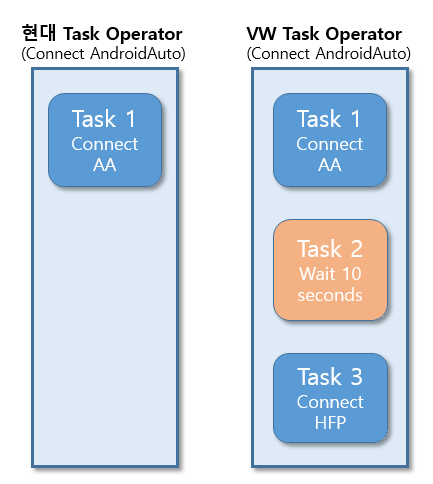


Figure 3‑11 Dynamic view - Task Operator (Tasks)

위와 같이 현대와 폭스바겐 모두 AndroidAuto 연결 완료라는 목표를 가지고 있습니다.

그러나, 현대의 경우에는 AndroidAuto 연결까지만 진행하면 되고,

폭스바겐의 경우 AndroidAuto 연결 이후에 10초 대기/ BT HFP 까지 연결을 진행해야 하는

일이 있습니다. 즉, 같은 목표지만 동작은 Project 마다 다른 상황입니다.

이에 각각의 일들을 작게 쪼개고 Task 라는 개념으로 일거리를 부여한 후

Task Operator 에게 각각의 Task 를 꺼내어 수행할 수 있도록 역할을 부여하였습니다.

Task Operator 는 Thread 형태로 돌며, 각각의 Task 를 꺼내어 Task Actor 에게

Task 수행을 요청하도록 설계하였습니다.

그 결과 Project 별 diff 나는 부분들을 원만히 적용할 수 있었고,

QA.07 SDCM은 공용 부 소스를 수정하지 않고, 새로운 Project 가 추가되더라도 변경 사항은

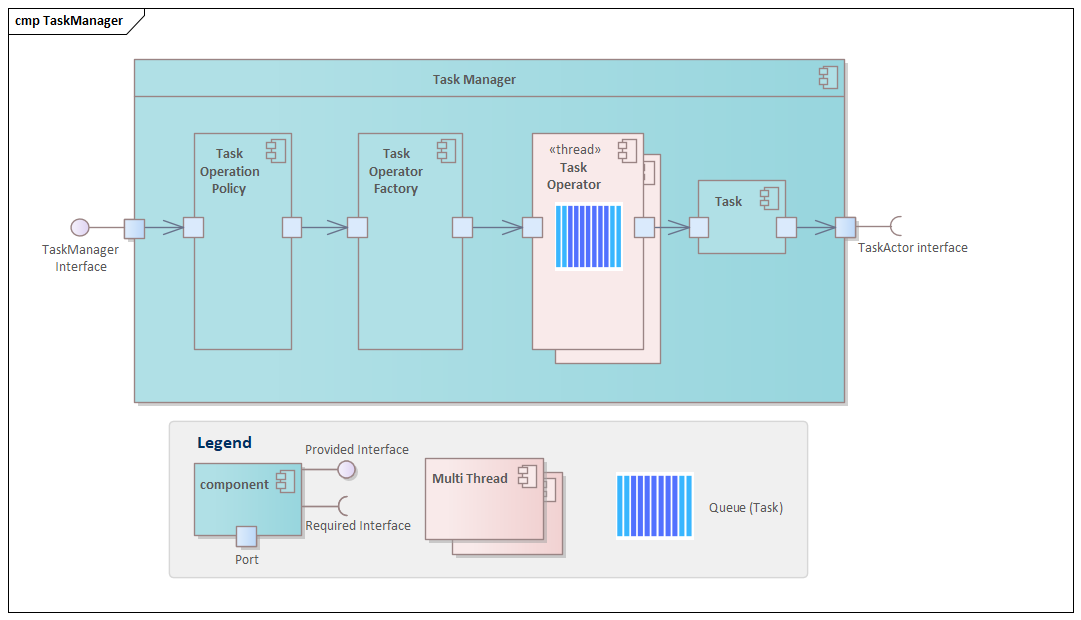
100% 변동 부에서의 변경만 발생해야 합니다.

를 달성할 수 있었습니다.

### Risk 식별 및 개선 전략 : Task + Thread wait

* **Risk**  
   Task 를 수행하면서 외부와의 통신을 하기 위해 대기해야 하는 경우들이  
   발생하였습니다.  
   이에 대한 후보안 검토 결과 Thread wait 처리에 대한 안이 최적 안으로 판단되어  
   해당 안으로 진행하기로 결정하였습니다.  
   그러나, Thread wait 처리시 일반적으로 발생하는 동기화 처리에 대한 Risk 가  
   존재하였습니다.  
   하나의 main thread 에서 수행하는 경우 해당 thread 내에서만 동작하기에  
   동기화 Issue 들이 발생하지 않지만, 각각의 Thread 로 구동되는 Task Operator 내에서  
   진행되는 작업들은 동기화를 반드시 고려하여야 하였습니다.
* **개선 전략**  
   기존에 Thread 동기화를 크게 신경 쓰지 않았던 코드들 모두 동기화  
   검토가 필요하였습니다.  
   단순히 queue 로만 쓰던 것은 동기화를 지원하는 concurrent queue 등으로 대체가  
   필요하였고, thread safe 하지 않는 data container 들은 모두 mutex 처리 등의  
   동기화 처리가 필요하였습니다.  
   동기화 처리로 인한 속도 저하가 있을 수 있으며, Trade off 발생 지점이었습니다.  
   그러나, 향후 신규 기능 확장성을 고려하여 Task + Thread wait 처리가 필요하였으며,  
   실제 성능 측정 결과 별 차이 없음을 확인하여 Task + Thread wait|  
   처리로 진행하였습니다.

### 세부 설계 QA.07 Design 3. Task Manager - Task Operator



Design 3

Figure 3‑12 Dynamic view - 2nd level (Task Manager)

* Task manager 의 2nd Level Decomposition C&C View
* SDCM 으로 다양한 연결 요청 및 SDCM 의 동작 요구 사항들이 인입 됨
* 해당 요청 사항들에 대해서 다중으로 처리되어야 하며, 외부 Client 의 응답을  
  받기 위해 대기도 필요함  
  > 외부 Client 의 응답은 Core 등으로 연결 요청을 보내는 경우 Core 의 연결   
   완료까지 대기해야 하는 상황들이 발생합니다.
* 다중의 요청을 동시에 처리해야 하며, 요청 중간 중간 멈춰서 Client 의 대기를  
  기다려야 하는 설계가 필요함

> Project 별로 다양한 Phone projection (AndroidAuto, CarPlay, CarLife 등) 을

지원하며, 동시에 관련 연결 동작들이 진행해야 하는 상황이 발생할 수 있습니다.  
 이러한 상황들을 순차 처리가 아닌 동시에 상황 처리를 해야 합니다.

이를 지원하기 위해 TaskOperator 를 다중으로 생성할 수 있도록 설계하였습니다.

TaskOperator 하나는 하나의 요청 사항에 대한 처리입니다.

TaskOperator 여러 개는 여러 개의 요청 사항에 대한 다중 처리입니다.

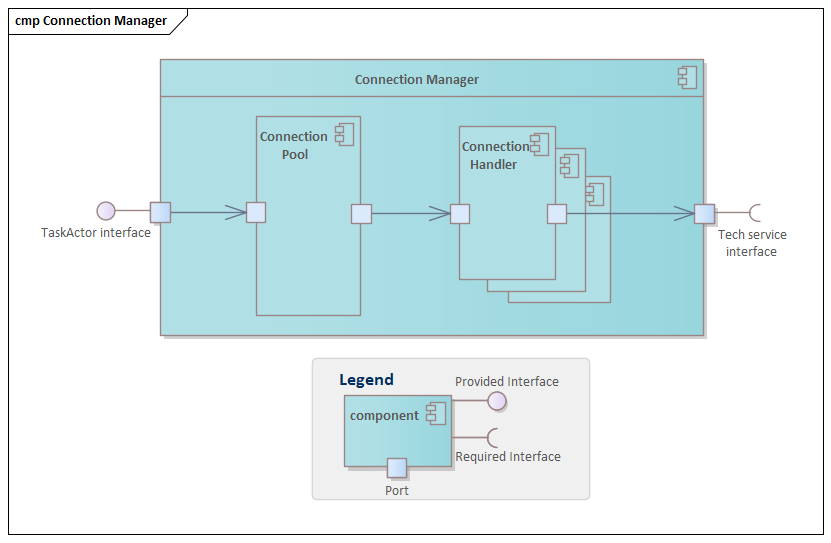
TaskOperator 는 Thread 로 동작 처리할 수 있기에 각각의 요청들의 처리 대기   
 또한 수행 할 수 있습니다.

### 품질 속성 : QA.06 Maintainability 연결 관리의 독립성 보장

* Legacy SDCM 에서는 Client type (Phone Projection / Media / BT 등) 별로 진행 사항들을  
  관리하기 위해 각각 진행 관리 class 를 만들었습니다.  
  SDCM 초기 설계 시에는 SRP 의 role 을 Client 단위로 규정하였었습니다.  
  이로 인해 AndroidAuto / CarPlay 등도 같은 Phone Projection 종류이므로,  
  하나의 DcmHandlerCoreEngine 이라는 class 내에서 공통 관리하였었습니다.  
  이로 인해서 다양한 문제들이 발생하였습니다.  
    
  AndroidAuto 연결 진행 중에 iPod Wireless 라는 기능이 진행되는 경우  
  해당 class 내에서 어떤 Technology 가 진행 중인지 매번 진행 상황마다 체크해야 하는  
  Issue 가 발생하기 시작했습니다. 해당 class 내에서는 여러 개의 Technology 들이  
  동시 다발적으로 진행할 수 있으며, 처리를 해야 하는 Role 이 부여되고 관리되었었습니다.  
    
  가장 큰 문제점은 신규 기능이 적용될 때 기존 기능들에 영향을 준다는 점이었습니다.  
  신규 기능 코드 적용 시 마다 기존 동작에 영향이 없는지 재 검증을 해야 했습니다.  
  이번 Wireless phone projection 신규 기능을 적용하면서 이러한 부분들을 제거하고자  
  아래와 같이 설계 하였습니다.

### 설계 대안 : Client type 별로 관리 vs Technology 진행 상황 별로 관리

* Phone Projection 연결 과정에서 발생하는 정보 관리
  + **SRP 원칙**의 재 해석
    - 1) Phone projection core (AndroidAuto/ CarPlay) 에서  
       발생하는 정보들 통합 관리
    - 2) Device 별 / Core 별로 각각의 정보 관리



Design 4

Figure 3‑13 Dynamic view - Connection Manager

Table 3‑13 Dynamic view - Connection manager

|  |  |  |
| --- | --- | --- |
|  | **Core 발생 정보 통합 관리** | **각각 Device 별/ Core 별로 관리** |
| Pros | 1. 한군데에서 통합 관리하므로 중복되는 정보들에 대해서 처리가 쉬움  2. 중복 코드 방지 | 1. Device 별 / Core 별로 관리 되다 보니 다른 class 에 영향을 받지 않음  2. 신규 기능 추가 시 기존 기능 검증 필요 없음 |
| Cons | 1. 통합 관리 (AndroidAuto, CarPlay) 되다보니 각각의 event 에 대해서 어떤 연결 과정 진행되는지 매번 확인 필요 (AndroidAuto or CarPlay)  2. 신규 기능 추가 시 기존 코드 전체 재 검증 필요 | 1. 통합 관리되던 방식에 비해 중복 코드 발생 가능성 있음  2. 여러 Handler 를 생성하기에 메모리 사용량 증가함 |

기 양산된 SDCM 에서는 SRP 의 역할을 Client 의 종류로 잡았었습니다.

### 설계 결정 및 근거 : Technology 진행 상황 별로 관리

기존 Legacy SDCM 의 경우

Media 기능은 DcmMediaHandler, Core (Phone Projection 기능) 은 DcmCoreEngineHandler,

BT 기능은 DcmBTHandler 등 Client 의 종류로 기능을 응집하였고, 이에 따라

각각의 Handler 에서는 각각의 Client 에 대응하는 역할을 수행하고 있었습니다.

이렇다보니 Core (Phone Projection 기능) 의 신규 PP 를 추가 시 문제가 발생하였습니다.

Core (Phone Projection 기능) 에 신규 PP 기능을 추가 하기 위해서

DcmCoreEngineHandler 를 수정하여 기능을 추가하여야 하였습니다.

이에 아래와 같은 형태의 코드로 진행되었습니다.

|  |
| --- |
| Code sample (DcmCoreEngineHandler) |
| if (AndroidAuto) {  …  } else if (CarPlay) {  …  } else if (CarLife) {  …  } |

만약 신규 기능 AirPlay 를 추가한다고 가정하면 아래와 같이 추가해야만 했습니다.

|  |
| --- |
| Code sample (DcmCoreEngineHandler) |
| if (AndroidAuto) {  …  } else if (CarPlay) {  …  } else if (CarLife) {  …  } else if (AirPlay) {  …  } |

위와 같이 붉은 색 부분의 if 문이 하나 더 들어가야 합니다.

DcmCoreEngineHandler class 를 구성하면서 Core engine 에서 발생하는 이벤트들을

통합 관리하여 SRP 를 부여했다고 생각했지만, 오히려 신규 기능 추가 시에는

기존 기능들에 영향을 주고, 신규 코드 작성 시마다 기존 코드 검증이 지속 발생하였습니다.

### Risk 식별 및 개선 전략 : Connection Handler

* **Risk**

SRP 재해석으로 기존에 client 별로 큰 묶음으로 되어 있던 코드들을  
 refactoring 및 Technology 별로 분리하는 처리로 설계하였습니다.  
 분리 과정에서 중복 코드들의 발생하기 시작하였습니다.  
 큰 묶음으로 합쳐져 있을 때는 모두 같은 method 들을 이용할 수 있었다고 하면,  
 물리적인 분리로 인해서 중복으로 해당 method 들을 가지고 가야 하는 문제점이  
 발생하였습니다.

* **개선 전략**

큰 묶음에서 작은 묶음 단위 (기능별) 로 class 분리를 하면서, 공통적으로  
 사용하는 method 들을 식별하였습니다.  
 기존에는 큰 묶음으로 되어 있기에 특별히 신경 쓰지 않고, 해당 class 안에 구현을  
 했다면, 현재는 공통적인 사용 부분과 특화된 기능 내의 사용 코드를 분류하여  
 구현을 해야 하도록 분류해 놓았습니다.  
 세부 기능들은 공통부분의 부모 class 를 상속 받아 사용하도록 변경 처리하였습니다.

### 세부 설계 QA.06 Design 4. Connection Handler

* 개선 안 : Technology 진행 상황 별로 관리

이번 Wireless PP 기능을 추가하면서 이 부분에 대해서 개선하였습니다.

* As-Is : DcmCoreEngineHandler 로 Core engine 정보를 모두 관리
* To-Be : Device 별 / Technology 별로 Core engine 코드를 모두 분리

DcmCoreEngineHandler 에서 다양한 Technology (AndroidAuto / CarPlay / CarLife 등) 의

정보들을 통합 관리한다면, 이번 설계에서는 각각 분리하여 아래와 같이 물리적으로

(별도 class) 로 분리시켰습니다.

* DcmAndroidAutoWiredHandler / DcmAndroidAutoWirelessHandler
* DcmCarPlayWiredHandler / DcmCarPlayWirelessHandler
* DcmCarLifeWiredHandler / DcmCarLifeWirelessHandler 등

>모든 Technology 공통인 것은 DcmWiredHandler / DcmWirelessHandler 에 구현함

위와 같이 물리적으로 코드를 분리시켰기에 신규 기능 추가 시 신규 class 파일을

생성하면 되며, 신규 기능을 구현하더라도 기존 기능 class 에는 영향을 미치지 않게

되었습니다. (물리적으로 분리 시켜 놓음)

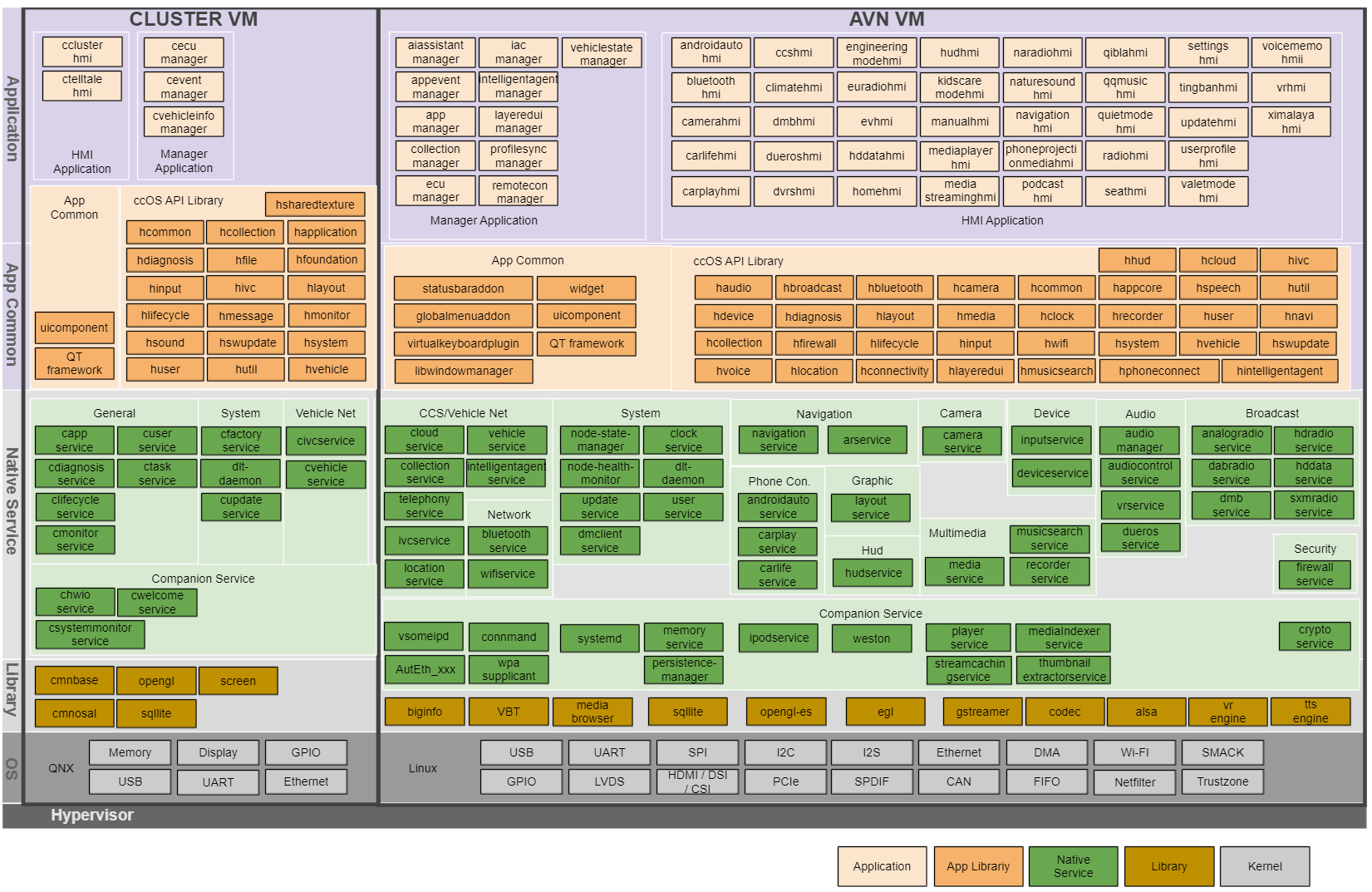
이로써 QA.06 연결 관리의 유연성과 독립성을 위해 각 장치의 상태 정보를 효과적으로 캡슐화하고 관리해야 하며, 각각의 연결 과정이 100% 독립적으로 운용되어야 합니다.

를 만족하였습니다.

## Allocation View

### Allocation Style

SDCM 은 Head Unit 내의 Middle ware software 로 탑재되어 있고, 현대 ccOS interface 와 연계되어 구동되도록 되어 있습니다.



Legend

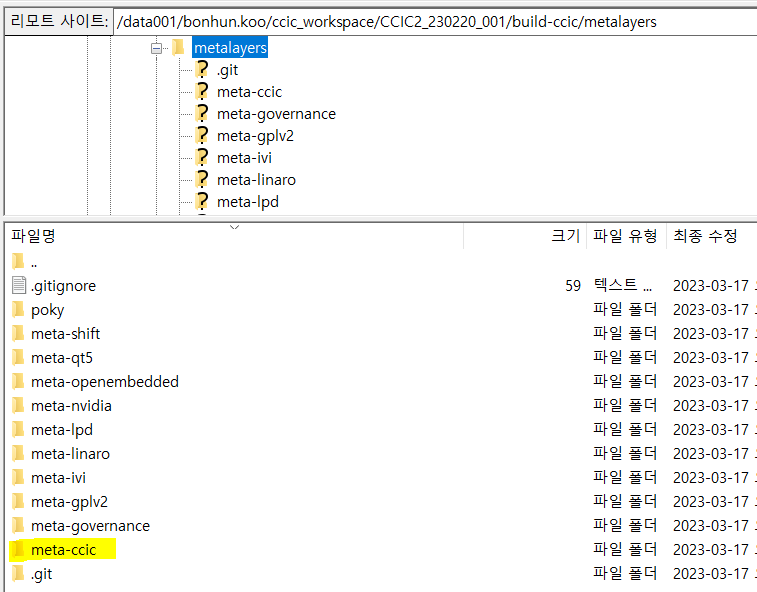
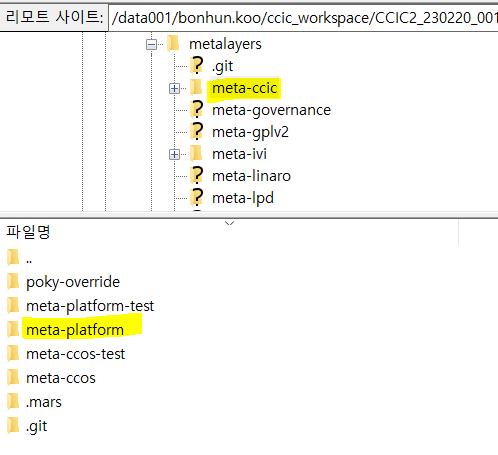
Figure 3‑14 AVN 구성요소

현대 AVN Software 내에서 deviceservice 라는 모듈로 구성되어 동작하고 있습니다.  
(붉은색 상자)

### Implementation Style

현대 AVN 은 yocto 기반의 software 입니다.

Yocto 기반의 빌드 시스템이 구성되어 있으며, sdcm 은 하기 위치에 위치해 있습니다.

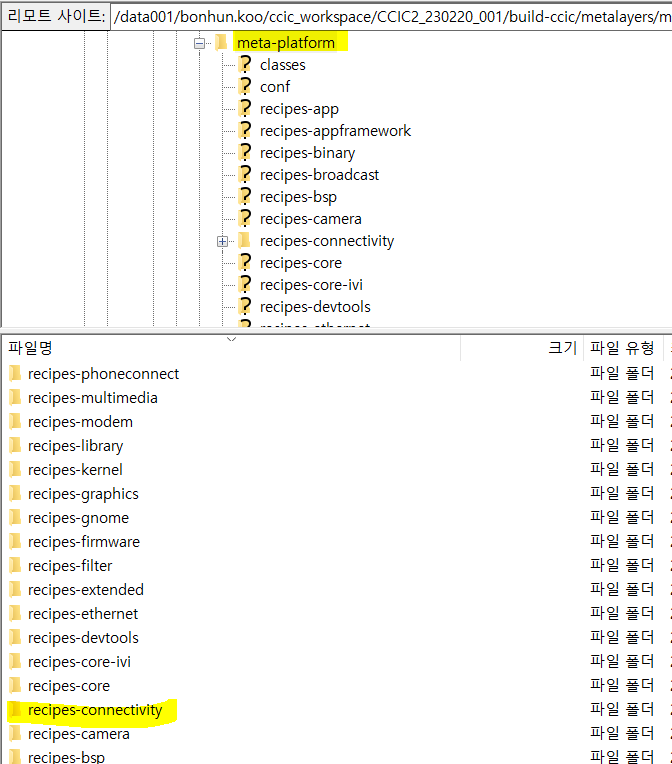
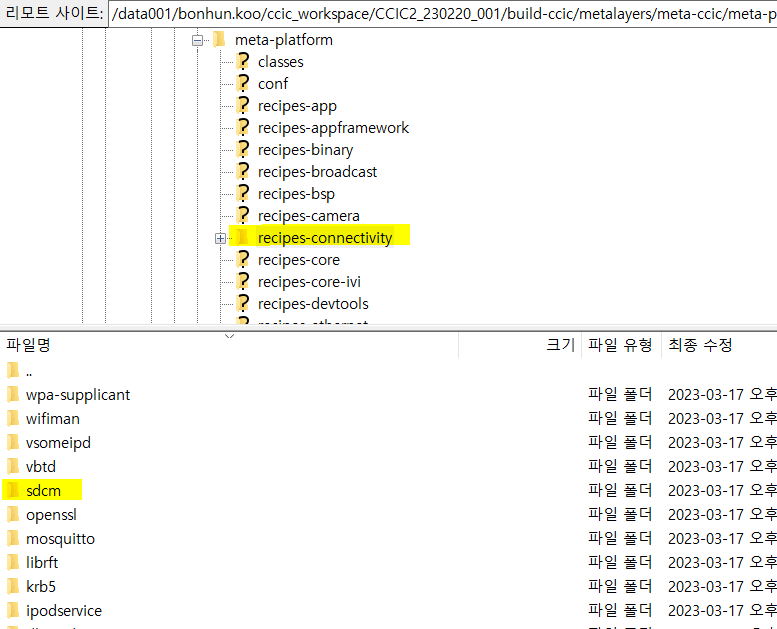
 

Figure 3‑15 Implementation Style

## Traceability summary

SDCM 서비스 시스템의 설계 결정 사항과 관련 QA를 정리하였습니다.

Table 3‑14 Traceability summary

|  |  |  |
| --- | --- | --- |
| **Design Element** | **Quality Attribute** | **Architectural Decision** |
| Message Broker | QA-09 Project 별 기능 추가 시 공용 부의 독립성 보장 | Publish-Subscribe pattern 을 적용하여 Event 발생 부와 수신 부를 분리시켰습니다. 이로 인해 신규 기능 추가 시 공용 부는 변경 없이 변동 부 추가만으로 신규 기능 구현이 가능해졌습니다. |
| Tech Service | QA-09 Project 별 기능 추가 시 공용 부의 독립성 보장 | Technology 별 (AndroidAuto/ CarPlay 등) 로 각각 Service 를 분리하였기에 신규 기능 추가 시 별도의 신규 Tech service 를 추가하면 됩니다. 이로 인해 공용 부의 수정 없이 변동 부 추가만으로 신규 기능 구현이 가능해졌습니다. |
| Task Manager | QA-07 공용 부와 변동 부의 분리 | Project 별로 차이 나는 부분들을 작은 Task 단위로 분리하였으며, 이러한 Task 를 수행하는 Task Operator 와 Task Manager 를 이용하여 지원할 수 있게 되었습니다. |
| Task Operator | QA-07 공용 부와 변동 부의 분리 | 변동 부에서 설정된 Task list 들을 Task Operator 가 하나씩 Task 들을 꺼내어 수행 가능 해 짐으로 인해 Project 별 diff 나는 부분들을 공용 부 수정 없이 변동 부의 Task 추가로 지원할 수 있게 되었습니다. |
| Connection Manager | QA-06 연결 관리의 독립성 보장 | Connection Handler 들을 통합 관리하는 Manager 역할로 각각의 Device 별, Technology 별로 Connection Handler 를 생성할 수 있어서 독립성을 보장하게 되었습니다. |
| Connection Handler | QA-06 연결 관리의 독립성 보장 | 기존에 통합된 Handler 에서 관리되던 연결 정보들을 각각의 Technology 별로 분리하였습니다. 물리적으로 CarPlay / AndroidAuto / CarLife 등을 분리하였기에 서로 연결 정보들이 영향을 받지 않고 독립적으로 연결 정보 관리가 될 수 있게 되었습니다. |

# Architectural Validation

## Maintainability QA06

* QA06 : 연결 관리의 유연성과 독립성을 위해 각 장치의 상태 정보를 **효과적으로 캡슐화하고** 관리해야 하며, **각각의 연결 과정이 100% 독립적으로 운용**되어야 합니다.  
  + 연관 설계 모듈 : **Connection Handler**
  + Maintainability Validation 실험  
    - 실험 조건 : 연결 과정에서 발생되는 로그로 판단  
    - 실험 기대 결과 : 분리된 각각의 Connection Handler 에서 로그 출력을 기대  
    - 실험 수행 결과 :  
     Legacy SDCM : 하나의 통합된 Core handler 에서 통합되어 로그 출력함  
     신규 설계 SDCM : **분리된 Connection Handler 에서 각각 연결 진행   
     상황 로그 출력함**
  + Maintainability Validation 실험 결과 QA06 요구 사항을 만족함을 확인함

## Usability QA07

* QA07 : SDCM은 공용 부 소스를 수정하지 않고, 새로운 Project 가 추가되더라도 변경 사항은  
   **100% 변동 부에서의 변경만 발생**해야 합니다.  
  + 연관 설계 모듈 : **Task Manager / Task Operator**
  + Usability Validation 실험  
    - 실험 조건 : 신규 Project 에 따른 연결 과정상 특이 동작 추가 가정하여  
     실제 구현 진행  
    - 실험 기대 결과 : 공용부의 수정 없이 변동 부에서만 영향을 받아 수정 진행  
    - 실험 수행 결과 :  
     Legacy SDCM : 공용부에도 해당 Project 추가를 위해 #ifdef (신규 Project)   
     등의 조건 필요함  
     신규 설계 SDCM : **Task Operator 내에 특이 동작 (Task) 를 추가함으로써  
     공용 부 영향 없이 추가 가능**
  + Usability Validation 실험 결과 QA07 요구 사항을 만족함을 확인함

## Modifiability QA09

* QA09 : 신규 Phone Projection 기능을 추가 적용 할 때 공용부의 **기존 구조 변경 없이**   
   신규 기능 추가가 가능해야 합니다.  
  + 연관 설계 모듈 : **Message Broker / Connection Handler**
  + Modifiability Validation 실험  
    - 실험 조건 : 신규 Phone Projection 기능 가정하여 실제 구현 진행  
    - 실험 기대 결과 : 공용부의 최소한의 수정으로 신규 기능 추가  
    - 실험 수행 결과 :  
     Legacy SDCM : Core 응답부터 연결 과정의 정보 처리까지 모두 신규 기능에   
     영향을 받음 (기존 코드 전체 재 검증 필요)  
     신규 설계 SDCM : **1차적으로 Message Broker 에서 정보의 분리가 되었으며  
     (Topic/ Subject), 2차적으로 각각의 Connection Handler 에서  
     AA,CP, 신규 기능별로 분리 (물리적으로 코드 분리) 가 되어  
     기존 코드에 영향 없이 신규 기능 추가 가능함을 확인**
  + Modifiability Validation 실험 결과 QA09 요구 사항을 만족함을 확인함

# Wrap up

## 사업적 성과와 기여

**현대 Project 신규 기능 (Wireless Phone Projection) 23년 내 대응 기여**

* 23년 내 신규 요구사항인 현대 모델 (5개 Project 동시 개발) 에 Wireless Phone Projection 기능 추가 요청에 대해서 대응 가능함
* 현대 측에서 지속적인 일정 단축과 안정화에 대한 요구에 대응 가능함  
  - Legacy SDCM : 각각의 모델 branch 와 공용 부에 적용된 소스 안정화 등에 의해 일정   
   만족은 요원하였던 상황임  
  - 신규 설계 SDCM : Task Manager 등 다양한 project 의 요구 사항에 대비할 수 있도록   
   설계되어, 하나의 소스와 하나의 git 에서 다양한 사업자 및 Project   
   대응이 가능하도록 설계됨  
  - 예상 M/M : 기존 10 M/M 소요되던 것을 5 M/M 이하로 인원 효율화 가능함   
   (50% 인력 효율 개선)  
   \* 10 M/M 산정 기준 : 현대 Project 5개 Project 에서 각각의 project 당 2 M/M 소요함   
   (신규 기능 추가 시)

**Linux platform 에서의 SDCM 인력 효율화 기여**

* 각각의 다양한 Project, Branch 등으로 인원 파편화가 되어 있던 부분을 인력 효율화 가능한  
  상황이 되어, 향후 적은 인원으로도 공통 대응이 가능함

## Lessons Learned

**여러 마디의 말보다 설계 문서를 보여주는 게 빠르다.**

* 이전에는 주로 언어로 소통하고, 이해하려고 하였으나 현재는 가능한 간단한 Sequence   
  diagram 이라도 그려서 보여주고 이해 시키는 것이 빠르다는 것을 느꼈습니다.
* UML 이라는 도구를 통해 서로 의사소통 할 때 시간이 절약되는 것을 경험하였습니다.

**급할수록 돌아가라는 말이 있듯이 급한 일정이라도 합리적인 설계가 밑바탕이 되어야 추후 Issue 를 줄일 수 있다.**

* 보통 일정이 급하면, 설계는 미뤄놓고 구현부터 시작하는 경우가 많았습니다.  
  설계는 간단하게 그림을 고치면 되지만, 구현은 처음부터 다시 짜야 하는 경우도 있다는   
  것을 경험하였습니다.  
  급하더라도 설계로 충분한 검증을 하고 구현을 할 수 있도록 노력이 필요합니다.

**Architect 교육 과정에서의 경험은 개발자 인생에 있어서 최고였다.**

* 사전 과정 포함하여 CMU Architect 교육은 인생에 있어서 최고의 교육 시간이었습니다.   
  후배 연구원들께 누구나 한번 즈음은 꼭 경험해 볼 수 있으면 좋겠다고 말합니다.
* 22년 CMU 교육에서 1팀 (one team)의 팀장을 맡았었고, 1팀이 Team Project 1등을 한 것이   
  기억에 남습니다.

## Question and Answer