|  |
| --- |
| ADL FRAMEWORK |
| 구조설계서 |
|  |
|  |
| **2021-03-15** |
| **홍덕기(duki.hong@samsung.com)** |

이 문서는 **ADL FRAMEWORK** 개발을 위한 구조설계서이다.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0.1 | 2021-03-15 | <<홍덕기>> | 초기 문서 생성 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1. 시스템 개요 3](#_Toc479150191)

[2. 요구사항 4](#_Toc479150192)

[2.1. 기능적 요구사항 4](#_Toc479150193)

[2.2. 비기능적 요구사항 4](#_Toc479150194)

[2.3. 품질 속성 5](#_Toc479150195)

[3. 시스템 구조 6](#_Toc479150196)

[4. 컴포넌트 사양 7](#_Toc479150197)

[부록 8](#_Toc479150198)

[A. 도메인 모델 9](#_Toc479150199)

[B. 품질 시나리오 10](#_Toc479150200)

[C. 품질 시나리오 분석 11](#_Toc479150201)

[D. 후보 구조 12](#_Toc479150202)

[E. 후보 구조 평가 13](#_Toc479150203)

[F. 최종 구조 설계 14](#_Toc479150204)

[G. 최종 구조 평가(ATAM) 15](#_Toc479150205)

# 시스템 개요

## 비즈니스 환경

현대의 거의 모든 산업 분야에서 IT가 차지하는 비중이 나날이 높아지고 있다. 이중 핵심이라 할 수 있는 S/W에 관심이 높아지고 있으며, 이런 S/W의 Quality에 큰 영향을 끼치는 S/W Architecture의 중요성이 대두되고 있다.

하지만 S/W Architecture 설계를 위한 공식 Tool이 존재하지 않고, 몇 안되는 S/W(그림1. AcmeStudio) 들도 그 사용법이 까다로워서 Architect들은 PPT나 종이에 직접 그리는 것을 보다 선호하는 상황이다.

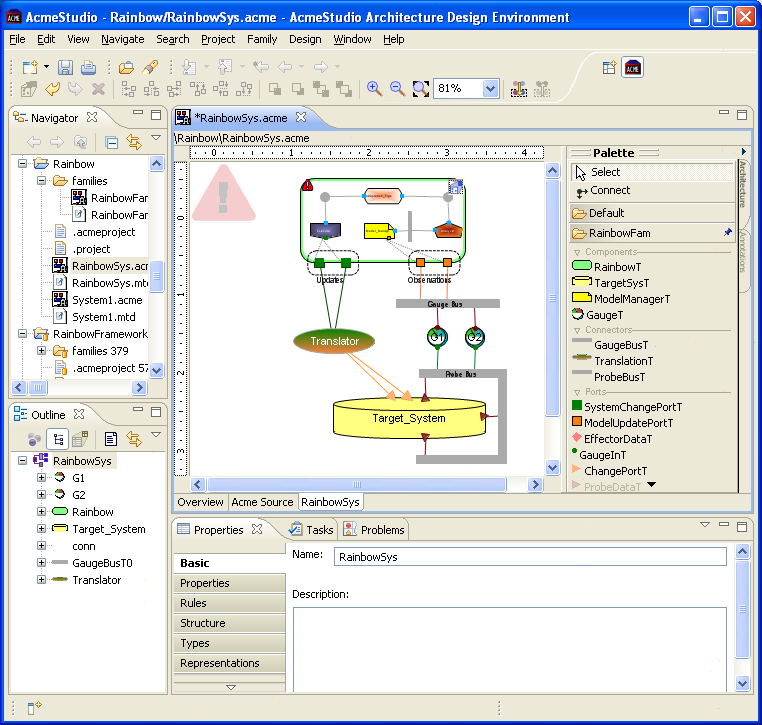


그림 . ADL 도구 예제(AcmeStudio)

이에 따라, System에 대한 기능적, 비기능적 품질 요구사항을 정의하고 이에 대한 다양한 구조를 설계할 때 이를 보다 손쉽게 실행과 개발 관점에서 도식화해서 나타낼 수 있는 도구를 개발하고자 한다.

## 시스템의 정의

ADL(Architecture description Language)은 소프트웨어 구조를 명세하기 위한 언어 또는 개념적인 모델이다. 본 시스템은 사용자가 ADL을 사용해서 구조를 설계하고 검토하고 분석할 수 있도록 하며 정의는 아래와 같다.

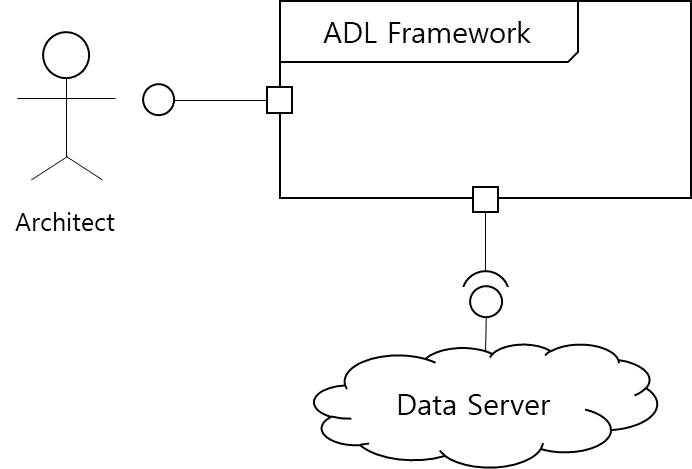


그림 . 시스템 정의

아직까지 소프트웨어 구조를 명세하기 위한 표준 모델링 언어는 존재하지 않는다. 이 과제 에서는 소프트웨어 구조를 명세하기 위한 ADL을 정의하고, 정의된 ADL을 사용해서 소프트웨어 구조를 설계하기 위한 설계 도구를 개발한다.

# 요구사항

## 기능적 요구사항

ADL Framework의 Use Case는 다음과 같다.



그림 . Use case Diagram

ADL Framework은 새로운 Project를 생성(UC\_01) 및 저장(UC\_02), 저장된 Project을 불러올(UC\_03) 수 있으며, Project에 시스템의 구조를 명세하기 위한 Component & Connector View(UC\_05), 각 Component와 Connector를 어떻게 컴퓨팅 환경에 배치/할당하는지를 명세하기 위한 Deployment View(UC\_06), 개발해야 할 모듈의 관계를 명세하기 위한 Module View(UC\_07) 및 누가 개발/담당할 지를 할당하는 Work Assignment View(UC\_08)를 추가할 수 있다. 또한 생성된 View를 바탕으로 Skeleton Code를 생성(UC\_09)을 할 수 있다. 또한 동작 중 Process Fatal이 발생하는 경우 서버로 관련 Log를 Report(UC\_10)할 수 있다.

|  |  |
| --- | --- |
| **UC\_01** | **Project를 생성한다.** |
| 설명 | 각종 View를 포함할 수 있는 새로운 Project를 생성한다. |
| 행위자 | 사용자 |
| 선행조건 | ADL Framework 구동상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 Project 메뉴를 연다. 2. 사용자가 Project 생성 메뉴를 선택한다. 3. System은 Project Name을 설정할 수 있는 화면을 표시한다. 4. 사용자는 Project Name을 입력한다. 5. System은 Project Explorer Window에 새로 생성한 Project를 표시한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_02** | **Project를 저장한다.** |
| 설명 | 각종 View를 포함하는 생성된 Project를 저장한다. |
| 행위자 | 사용자 |
| 선행조건 | Project가 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 Project 메뉴를 연다. 2. 사용자가 Project 저장 메뉴를 선택한다. 3. System은 Project를 Project Name으로 저장한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_03** | **Project를 불러온다.** |
| 설명 | 저장되어 있는 Project를 불러온다. |
| 행위자 | 사용자 |
| 선행조건 | 저장된 Project가 존재하는 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 Project 메뉴를 연다. 2. 사용자가 Project 불러오기 메뉴를 선택한다. 3. System은 Project Load한다.    1. 저장되어 있는 View들을 Load한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_04** | **View를 생성한다.** |
| 설명 | 모든 View의 기본이 되는 View를 생성한다. |
| 행위자 | 사용자 |
| 선행조건 | Project 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 View 메뉴를 연다. 2. 사용자가 Component & Connector View 또는 Deployment View 또는 Module View 또는 Work Assignment View 메뉴를 선택한다. 3. System은 View를 Project에 추가한다.    1. System은 그릴 수 있는 Canvas를 생성한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_05** | **Component & Connector View를 그린다.** |
| 설명 | System의 구조를 명세하기 위한 Component & Connector View를 그린다. |
| 행위자 | 사용자 |
| 선행조건 | Project 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 View 메뉴를 연다. 2. 사용자가 Component & Connector View 메뉴를 선택한다. 3. System은 View를 Project에 추가한다.    1. System은 그릴 수 있는 Canvas를 생성한다.    2. System은 Palate에 Component와 Connector를 표시한다. 4. 사용자가 Component를 Canvas에 배치한다. 5. System은 배치된 Component를 표시한다. 6. 사용자가 Component간의 관계를 Connector로 연결한다. 7. System은 Component간의 Connector를 표시한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_06** | **Deployment View를 그린다.** |
| 설명 | Component와 Connector를 어떻게 컴퓨팅 환경에 배치/할당하는지를 명세하기 위한 Deployment View를 그린다. |
| 행위자 | 사용자 |
| 선행조건 | Project 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 View 메뉴를 연다. 2. 사용자가 Deployment View 메뉴를 선택한다. 3. System은 View를 Project에 추가한다.    1. System은 그릴 수 있는 Canvas를 생성한다.    2. System은 Palate에 Component와 Connector를 표시한다. 4. 사용자가 Component를 Canvas에 배치한다. 5. System은 배치된 Component를 표시한다. 6. 사용자가 Component간의 관계를 Connector로 연결한다. 7. System은 Component간의 Connector를 표시한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_07** | **Module View를 그린다.** |
| 설명 | 개발해야 할 모듈의 관계를 명세하기 위한 Module View를 그린다. |
| 행위자 | 사용자 |
| 선행조건 | Project 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 View 메뉴를 연다. 2. 사용자가 Module View 메뉴를 선택한다. 3. System은 View를 Project에 추가한다.    1. System은 그릴 수 있는 Canvas를 생성한다.    2. System은 Palate에 Module를 표시한다. 4. 사용자가 Module를 Canvas에 배치한다. 5. System은 배치된 Module를 표시한다. 6. 사용자가 Module간 관계를 연결한다. 7. System은 Module간 관계를 표시한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_08** | **Work Assignment View를 그린다.** |
| 설명 | 누가 개발/담당할 지를 할당하는 Work Assignment View를 그린다. |
| 행위자 | 사용자 |
| 선행조건 | Project 생성 / 불러온 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 View 메뉴를 연다. 2. 사용자가 Work Assignment View 메뉴를 선택한다. 3. System은 View를 Project에 추가한다.    1. System은 그릴 수 있는 Canvas를 생성한다.    2. System은 Palate에 Module를 표시한다. 4. 사용자가 Module를 Canvas에 배치한다. 5. System은 배치된 Module를 표시한다. 6. 사용자가 Module간 관계를 연결한다. 7. System은 Module간 관계를 표시한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_09** | **Code를 생성한다.** |
| 설명 | 그려진 Module View를 기준으로 Java Code를 생성한다. |
| 행위자 | 사용자 |
| 선행조건 | Module View가 만들어진 상태 |
| 후행조건 |  |
| 기본 동작 | 1. 사용자가 Tool 메뉴를 연다. 2. 사용자가 Generate Code 메뉴를 선택한다. 3. System은 Module View를 찾는다. 4. System은 Module View에 명시된 Element를 list up 한다.    1. Module Name으로 Class를 만들고 Stereo type을 주석으로 남긴다. 5. System은 Module View에 명시된 Relation을 list up한다.    1. Relation에 맞게 각 Class에 Field를 추가한다. |
| 추가 동작 |  |

|  |  |
| --- | --- |
| **UC\_10** | **시스템 Fatal을 서버로 전송한다.** |
| 설명 | 시스템 동작 중에 Process Fatal이 발생한 경우 Cause 및 Call Stack을 Data Server로 전송한다. |
| 행위자 | 시스템 |
| 선행조건 | 시스템 동작 중에 Process Fatal이 발생한다. |
| 후행조건 |  |
| 기본 동작 | 1. System은 Fatal Cause 및 Call stack을 수집한다. 2. System은 수집한 정보를 Data Server로 전송한다. |
| 추가 동작 |  |

## 비기능적 요구사항

|  |  |  |
| --- | --- | --- |
| **NFR\_01** | **Reliability** | **저장하지 않은 Project 변경사항을 복구할 수 있어야 한다.** |
| 설명 | 컴퓨터 시스템 Power off등의 이유로 Project 변경사항을 저장하지 않은 상태에서 시스템이 종료된 경우에 다시 시작할 때 이전 수정 사항을 보여줘야한다. | |
| 환경 | Project의 변경 후 저장하지 않고 종료한 상황 | |
| 자극 | 시스템을 시작 | |
| 반응 | 종료하기 10초전의 Project 상태를 Load한다. | |
| 측정 | [Project 종료 10초전 상태] | |
| **제약** | 시스템을 시작할 때 [Project 종료 10초전 상태]를 보여줘야 한다. | |

|  |  |  |
| --- | --- | --- |
| **NFR\_02** | **Modifiability** | **시스템 하위 호환성** |
| 설명 | 시스템 Revision이 올라갈 때 기 제공하던 시스템 API를 사용해서 개발된 3rd party plugin들이 정상적으로 동작해야 한다. | |
| 환경 | 3rd party plugin이 개발되어 있는 상태 | |
| 자극 | 시스템 Revision update | |
| 반응 | 3rd party plugin이 update된 Revision에서도 정상적으로 동작 한다. | |
| 측정 | [비정상 동작 Plugin 수] = [기존 Revision에서 정상 동작하는 Plugin 수] – [Update된 Revision에서 정상 동작하는 Plugin 수] | |
| **제약** | [비정상 동작 Plugin 수]가 0개여야 한다. | |

## 품질 속성

|  |  |  |
| --- | --- | --- |
| **QA\_01** | **Availability** | **시스템에 오류가 발생시 복구시간이 빨라야 한다.** |
| 설명 | 시스템에 오류가 발생해서 동작하지 않는 경우, 시스템을 재시작 해서 정상적으로 복구한다. | |
| 환경 | 시스템이 정상적으로 동작중인 상태이다. | |
| 자극 | 시스템에 오류가 발생하여 정상적으로 동작하지 못한다. | |
| 반응 | 시스템을 재시작해서 복구한다. | |
| 측정 | [시스템 복구 시간] = [시스템 재시작 완료 시간] – [시스템 오류 발생 시간] | |

|  |  |  |
| --- | --- | --- |
| **QA\_02** | **Performance** | **Project load시 View 표시가 빠를수록 좋다.** |
| 설명 | 다수의 Element 및 Relation을 가진 View를 load시 빠르게 화면에 표시한다. | |
| 환경 | 시스템이 정상적으로 동작중인 상태이다. | |
| 자극 | Project를 Load한다. | |
| 반응 | Project load하며 View를 표시한다. | |
| 측정 | [View 표시시간] = [View가 화면에 나타난 시간] – [Project Load한 시간]  [View 표시시간]이 작을수록 좋다. | |

|  |  |  |
| --- | --- | --- |
| **QA\_03** | **Usability** | **직관적으로 쉽고 빠르게 View를 그릴 수 있어야한다.** |
| 설명 | 처음 시스템을 사용하는 사용자도 쉽고 빠르게 View를 그릴 수 있어야한다. | |
| 환경 | S/W Architecture에 대한 지식이 있는 사용자 | |
| 자극 | Project를 시작한다. | |
| 반응 | Element와 Relation이 있는 View를 생성한다. | |
| 측정 | [View 완성시간] = [View에 Element와 Relation을 추가] – [Project 시작 시간] | |

|  |  |  |
| --- | --- | --- |
| **QA\_04** | **Efficiency** | **메모리 사용량을 최소화해야 한다.** |
| 설명 | View, Element, Relation들이 추가될 때 메모리 증가분이 적어야 한다. | |
| 환경 | System 사용 중 | |
| 자극 | View, Element, Relation을 추가한다. | |
| 반응 | 추가된 View, Element, Relation을 표시한다. | |
| 측정 | [Memory 증가분] = [View, Element, Relation 추가 후 Memory 사용량] – [View, Element, Relation 추가 전 Memory 사용량] | |

|  |  |  |
| --- | --- | --- |
| **QA\_05** | **Interoperability** | **3rd party plugin의 에러발생이 최소화되야 한다.** |
| 설명 | 시스템에서 제공하는 API를 사용해서 개발된 3rd party plugin들이 Error 없이 동작해야 한다. | |
| 환경 | 3rd party plugin이 추가된 상태 | |
| 자극 | Plugin의 기능을 수행한다. | |
| 반응 | Plugin의 기능이 Error 없이 수행된다. | |
| 측정 | [Plugin Error율] = [정상 수행된 설치된 Plugin의 각 기능 수] / [(설치된 Plugin 수) X (각 Plugin 별 기능 수)] | |

|  |  |  |
| --- | --- | --- |
| **QA\_06** | **Extendibility** | **Code 생성시 쉽게 Language가 확장될 수 있어야한다.** |
| 설명 | Code 생성시 지원하는 Programing Language가 쉽게 추가될 수 있어야 한다. | |
| 환경 | 시스템 Maintenance 상황 | |
| 자극 | 새로운 Programing Language 추가 Requirement 발생 | |
| 반응 | 새로운 Programing Language로 Code 생성할 수 있도록 시스템 수정 | |
| 측정 | [변경 Code 라인 수] = [새로운 Programing Language로 Code 생성 기능 추가 후 Code ]와 [기존 Code] 간에 다른 Code Line 수 | |

|  |  |  |
| --- | --- | --- |
| **QA\_06** | **Extendibility** | **Code 생성시 쉽게 Language가 확장될 수 있어야한다.** |
| 설명 | Code 생성시 지원하는 Programing Language가 쉽게 추가될 수 있어야 한다. | |
| 환경 | 시스템 Maintenance 상황 | |
| 자극 | 새로운 Programing Language 추가 Requirement 발생 | |
| 반응 | 새로운 Programing Language로 Code 생성할 수 있도록 시스템 수정 | |
| 측정 | [변경 Code 라인 수] = [새로운 Programing Language로 Code 생성 기능 추가 후 Code ]와 [기존 Code] 간에 다른 Code Line 수 | |

|  |  |  |
| --- | --- | --- |
| **QA\_07** | **Modifiability** | **UI 변경에 의한 서비스 로직의 영향이 최소화되야 한다.** |
| 설명 | UI 변경이 필요한 상황에서 서비스 로직의 변경을 최소화하며 쉽게 변경이 이루어져야 한다. | |
| 환경 | UI 수정사항 발생 | |
| 자극 | UI 모듈 업데이트 | |
| 반응 | 변경된 UI 적용 | |
| 측정 | [변경된 Class 수] = [새로운 UI 적용 후 변경된 서비스 모듈 class 수] + [새로운 UI 적용 후 추가된 서비스 모듈 class 수] + [새로운 UI 적용 후 삭제된 서비스 모듈 class 수] | |

|  |  |  |
| --- | --- | --- |
| **QA\_08** | **Accountability** | **시스템 Error상황에서의 Bug report는 많을 수록 좋다.** |
| 설명 | 시스템 Error 상황에 DataServer로 보내는 Log 정보가 많을수록 좋다. | |
| 환경 | 시스템 정상동작 상황 | |
| 자극 | Process Fatal | |
| 반응 | Bug report | |
| 측정 | [총 line 수] = [Fatal Cause Line 수] + [Fatal Call stack line 수] + [RAM 사용량 line 수] + [Fatal 이전 Scenario log line 수] | |

# 시스템 구조

// 활동8. 구조 명세

// 점검8-1. 프로세스 등의 Allocation이 적절한가?

// 점검8-2. 컴포넌트 측면에서 Grouping이 적절한가?

// 점검8-3. 시스템의 동작 특성에 대한 설명이 적절한가?

# 모듈 사양

// 활동9. 모듈 명세

// 점검9-1. 컴포넌트 명세가 충분한가? (개발 가능)

// 점검9-2. 모듈 측면에서 Grouping이 적절한가?

// 점검9-3. Work Assignment가 적절한가?

// 점검9-4. 개발 측면에서 구조적 특징에 대한 설명이 적절한가?

부록

[A. 도메인 모델 9](#_Toc479150206)

[B. 품질 시나리오 10](#_Toc479150207)

[C. 품질 시나리오 분석 11](#_Toc479150208)

[D. 후보 구조 12](#_Toc479150209)

[E. 후보 구조 평가 13](#_Toc479150210)

[F. 최종 구조 설계 14](#_Toc479150211)

[G. 최종 구조 평가(ATAM) 15](#_Toc479150212)

1. 도메인 모델

본 과제의 도메인 모델은 아래 그림과 같다.

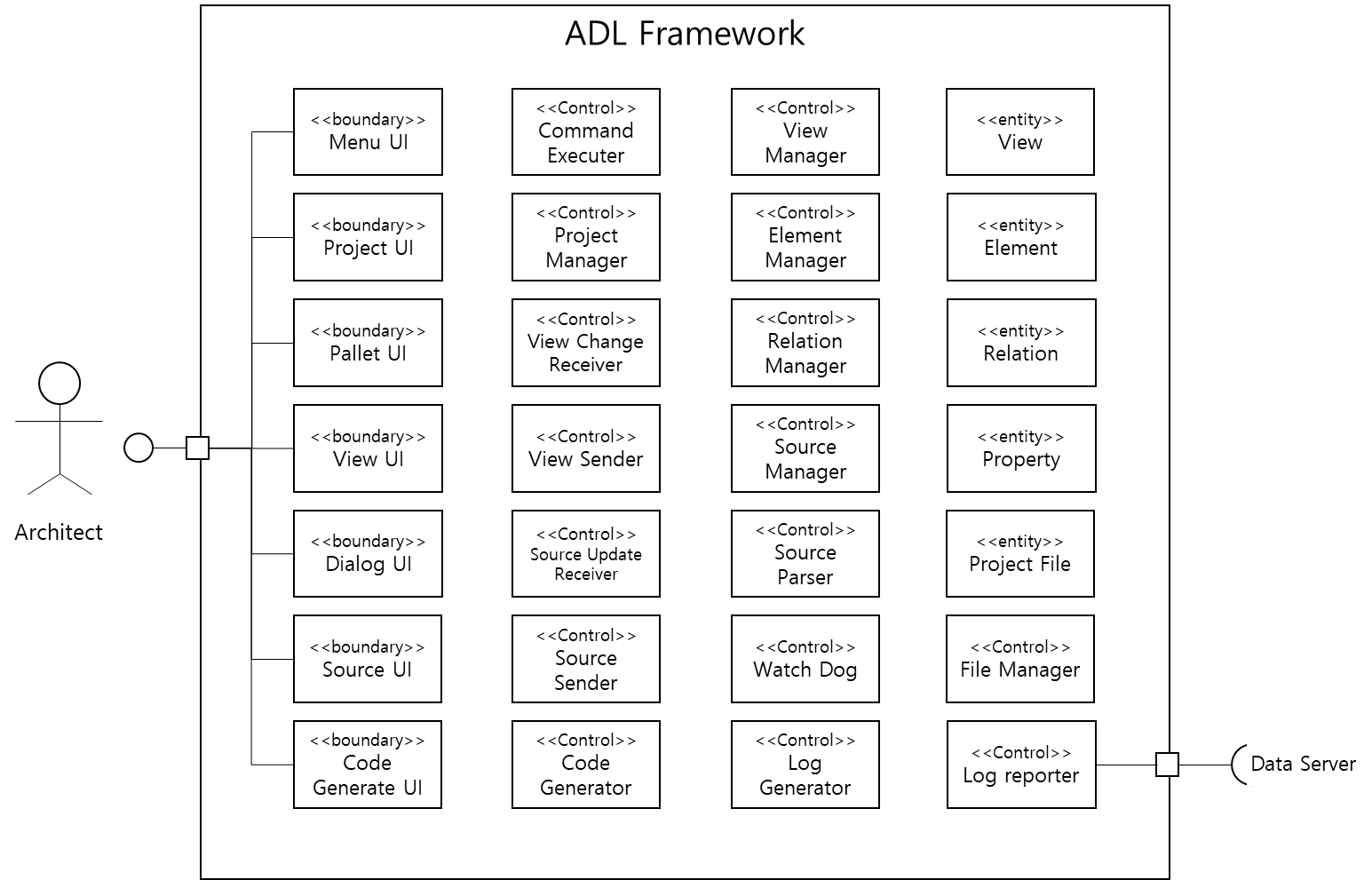


그림 . Domain Model

각 Use Case에 대한 세부적인 Sequence Diagram은 아래와 같다.

그림 . UC\_01 Project를 생성한다.



그림 . UC\_02 Project를 저장한다.



그림 . UC\_03 Project를 불러온다.



그림 . UC\_04 View를 생성한다.



그림 . UC\_05 Component & Connector view를 그린다.



그림 . UC\_06 Deployment View를 그린다.



그림 . UC\_07 Module View를 그린다.



그림 . UC\_08 Work Assignment View를 그린다.



그림 . UC\_09 Code를 생성한다.



그림 . UC\_10 시스템 Fatal을 서버로 Report 한다.

1. 품질 시나리오

본 시스템의 품질 시나리오 들은 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **QS\_01** | **Availability** | **시스템 오류 발생시 복구시간** |
| 설명 | 시스템에 오류가 발생해서 동작하지 않는 경우, 시스템을 재시작 해서 정상적으로 복구하는 시간이 빠를수록 좋다. | |
| 측정 | [시스템 복구 시간] = [시스템 재시작 완료 시간] – [시스템 오류 발생 시간] | |
| **QS\_02** | **Performance** | **Project load시 View 표시 시간** |
| 설명 | 다수의 Element 및 Relation을 가진 View를 load시 화면에 표시가 빠를수록 좋다. | |
| 측정 | [View 표시시간] = [View가 화면에 나타난 시간] – [Project Load한 시간]  [View 표시시간]이 작을수록 좋다. | |
| **QS\_03** | **Performance** | **Element 추가 시 rendering 시간** |
| 설명 | View에 Element 추가 시 화면에 표시가 빠를수록 좋다. | |
| 측정 | [Element 표시시간] = [View에 Element가 보이는 시간] – [Element 추가한 시간] | |
| **QS\_04** | **Usability** | **시스템 사용의 직관성** |
| 설명 | 처음 시스템을 사용하는 사용자도 쉽고 빠르게 View를 그릴 수 있어야한다. | |
| 측정 | [View 완성시간] = [View에 Element와 Relation을 추가] – [Project 시작 시간] | |
| **QS\_05** | **Efficiency** | **메모리 사용량을 최소화** |
| 설명 | View, Element, Relation들이 추가될 때 메모리 증가분이 적어야 한다. | |
| 측정 | [Memory 증가분] = [View, Element, Relation 추가 후 Memory 사용량] – [View, Element, Relation 추가 전 Memory 사용량] | |
| **QS\_06** | **Interoperability** | **3rd party plugin의 에러발생 최소화** |
| 설명 | 시스템에서 제공하는 API를 사용해서 개발된 3rd party plugin들이 Error 없이 동작해야 한다. | |
| 측정 | [Plugin Error율] = [정상 수행된 설치된 Plugin의 각 기능 수] / [(설치된 Plugin 수) X (각 Plugin 별 기능 수)] | |
| **QS\_07** | **Compatibility** | **기존 ADL Tool과의 호환성** |
| 설명 | Acme Studio등과 같은 기존 ADL tool에서 인식할 수 있도록 file export할 수 있어야 하고 해당 Tool에서 작성된 file도 import 할 수 있어야 한다. | |
| 측정 | [Import / Export 성공률] = [Import / Export 성공 Tool수] / [Total ADL Tool수] | |
| **QS\_08** | **Extendibility** | **Code 생성 Language 확장성** |
| 설명 | Code Generate시 지원하는 Programing Language가 쉽게 추가될 수 있어야 한다. | |
| 측정 | [변경 Code 라인 수] = [새로운 Programing Language로 Code 생성 기능 추가 후 Code ]와 [기존 Code] 간에 다른 Code Line 수 | |
| **QS\_09** | **Modifiability** | **UI 변경에 의한 서비스 로직의 영향이 최소화** |
| 설명 | UI 변경이 필요한 상황에서 서비스 로직의 변경을 최소화하며 쉽게 변경이 이루어져야 한다. | |
| 측정 | [변경된 Class 수] = [새로운 UI 적용 후 변경된 서비스 모듈 class 수] + [새로운 UI 적용 후 추가된 서비스 모듈 class 수] + [새로운 UI 적용 후 삭제된 서비스 모듈 class 수] | |
| **QS\_10** | **Testability** | **Unit test Coverage 수치** |
| 설명 | 시스템의 Unit Test Code Coverage가 높아야 한다. | |
| 측정 | [Code Coverage] = [Unit Test Code Coverage lines] / [Total Code Lines] | |
| **QS\_11** | **Accountability** | **시스템 Error상황에서의 Bug report 양** |
| 설명 | 시스템 Error 상황에 Data Server로 보내는 Log 정보가 많을수록 좋다. | |
| 측정 | [총 line 수] = [Fatal Cause Line 수] + [Fatal Call stack line 수] + [RAM 사용량 line 수] + [Fatal 이전 Scenario log line 수] | |
| **QS\_12** | **Reliability** | **저장하지 않은 Project 변경사항 복구** |
| 설명 | 컴퓨터 시스템 Power off등의 이유로 Project 변경사항을 저장하지 않은 상태에서 시스템이 종료된 경우에 다시 시작할 때 이전 수정 사항을 보여줘야한다. | |
| 측정 | [변경사항 소실률] = [Project 종료 10초전 상태]와 [재시작후 Project 상태] 사이에 다른 Source line 수 | |
| **QS\_13** | **Modifiability** | **시스템 하위 호환성** |
| 설명 | 시스템 Revision이 올라갈 때 기 제공하던 시스템 API를 사용해서 개발된 3rd party plugin들이 정상적으로 동작해야 한다. | |
| 측정 | [비정상 동작 Plugin 수] = [기존 Revision에서 정상 동작하는 Plugin 수] – [Update된 Revision에서 정상 동작하는 Plugin 수] | |

1. 품질 시나리오 분석

우리 시스템의 Business Driver는 아래와 같다.

1. 원하는 구조도를 쉽게 그릴 수 있어야 한다.
2. 코드 생성, 시뮬레이션, 성능 및 품질 분석 등 추가 기능의 확장이 용이할수록 좋다.
3. 성능과 변경 용이성이 높을수록 좋다.

이를 고려해서 중요도와 복잡도에 따라 최종 선택된 NFR(Non Functional Requirements), QA(Quality Attribute) 그리고 QS(Quality Scenario)의 매핑은 아래 표와 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 구분 | QS\_ID | 항목 | 중요도 | 복잡도 | 속성 |
| Modifiability | QS\_09 | UI 변경에 의한 서비스 로직의 영향이 최소화 | H | H | QA\_07 |
| QS\_13 | 시스템 하위 호환성 | H | L | NFR\_02 |
| Extendibility | QS\_08 | Code 생성 Language 확장성 | H | H | QA\_06 |
| Performance | QS\_03 | Element 추가 시 Rendering 시간 | L | L |  |
| Availability | QS\_01 | 시스템 오류 발생시 복구시간 | M | H | QA\_01 |
| Performance | QS\_02 | Project load시 View 표시 시간 | M | H | QA\_02 |
| Usability | QS\_04 | 시스템 사용의 직관성 | H | L | QA\_03 |
| Compatibility | QS\_07 | 기존 ADL Tool과의 호환성 | L | H |  |
| Efficiency | QS\_05 | 메모리 사용량을 최소화 | M | M | QA\_04 |
| Interoperability | QS\_06 | 3rd party plugin의 에러발생 최소화 | H | M | QA\_05 |
| Testability | QS\_10 | Unit Test Code Coverage 수치 | L | M |  |
| Accountability | QS\_11 | 시스템 Error상황에서의 Bug report 양 | M | H | QA\_08 |
| Reliability | QS\_12 | 저장하지 않은 Project 변경사항 복구 | M | H | NFR\_09 |

본 시스템은 Architect들이 별도의 학습없이 Architecture를 보다 손쉽게 설계할 수 있게 하는데 그 목적이 있다. 이에 따라 **QS\_04. 시스템 사용의 직관성**은 높은 중요도를 가지며 **QA\_03**로 선정하였다.

또한 시스템의 변경 용이성도 중용한 Business Driver인데 이에 따라 **QS\_09. UI 변경에 의한 서비스 로직의 영향 최소화**를 높은 중요도와 함께 **QA\_07**로 선정하였다. 그리고 본 시스템은 그 자체로 Tool이자 Framework이기에 3rd party에서도 본 시스템을 대상으로 추가기능 개발이 가능해야 하는데, 이때 **QS\_13. 시스템 하위 호환성** 및 **QS\_06. 3rd party plugin의 에러발생 최소화**는 필수적인 부분이 여서 높은 중요도를 가지며, **NFR02**와 **QA\_05**로 선정하였다.

확장의 용이성 또한 대표적인 Business Driver이며 **QS\_08. Code생성 Language 확장성**을 높은 중요도와 함께 **QA\_06**로 선정하였다.

하지만 **QS\_07. 기존 ADL과의 호환성**은 해당 속성을 지원하기 위한 투자대비 이를 통해서 얻을 수 있는 이점이 미미해서 품질속성에서 제외하였다. ADL은 표준이 없는 상황이어서 굳이 기존 Tool과의 호환성을 가져갈 필요는 없겠다.

Availability와 Performance 및 Reliability는 대부분의 System에서 중요한 부분이며 **QS.01 시스템 오류 발생시 복구시간**과 **QS\_12. 저장하지 않은 project 변경사항 복구**, **QS.02 Project Load시 View 표시시간**을 각 **QA\_01**과 **NFR\_09**, **QA\_02**로 선정하였다.

하지만 **QS\_03 Element 추가시의 Rendering 시간**은 시스템의 구조에 따라 크게 영향을 받는 부분이 아니고, 사용자가 인지할 만큼 큰 차이가 발생하지 않는 부분으로 판단해서 품질속성에서 제외하였으며, **QS\_10 Unit Test Code Coverage 수치**도 구조에 직접적인 영향을 끼치지 않아서 품질 속성에서 제외하였다.

시스템이 확장돼서 복잡해지게 되면 Error의 원인을 알기 어렵다. 이에 따라 가능한 많은 양의 Error Logging이 필요하며 **QS\_08 시스템 Error상황에서의 Bug report양**을 **QA\_08**로 선정하였다.

1. 후보 구조

// 활동6. 후보 구조 설계

// 점검6-1. 품질에 대한 분석과 후보 구조가 적절한가?

// 점검6-2. 성능에 대한 분석과 후보 구조가 적절한가?

// 점검6-3. 변경용이성/확장성에 대한 분석과 후보 구조가 적절한가?

1. 후보 구조 평가

// 활동7. 최종 구조 설계

// 점검7-1. 충돌되는 후보 구조의 비교/분석이 적절한가? (근거)

// 점검7-2. 선정된 후보 구조의 단점/RISK 분석이 명확한가?

1. 최종 구조 설계

// 활동7. 최종 구조 설계

// 점검7-3. 최종 구조로의 통합 과정에 대한 설명이 적절한가?

// 점검7-4. 최종 구조의 단점/RISK 관리가 적절한가? (개선)

1. 최종 구조 평가(ATAM)

// 활동10. 최종 구조 평가

// 점검10-1. 구조에 영향을 미치는 품질에 대한 검토가 충분한가?

// 점검10-2. 설계 결정사항의 식별이 충분한가?

// 점검10-3. 설계 결정사항의 분석이 적절한가? (근거)

// 점검10-4. 최종 구조의 평가가 적절한가? (위험)