2021년 2학기

분산프로그래밍2

중간레포트

신동아화재 RFP 소프트웨어 아키텍처

제출일	2021-	**팀명**	귤비
과목	분산프로그래밍 2	**조원**	60191644 김규리
학기	2021-2 학기		60191677 이은비
담당교수	김정호 교수님		

Homework Assignment

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

학　　과 : 융합소프트웨어학부

과　　목 : 분산프로그래밍1

담당교수 : 김정호 교수님

학번 : 60191644

이름 : 김규리　　(서명)

Homework Assignment

보고서 및 논문 윤리 서약

5. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
6. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
7. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
8. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

학 과 : 융합소프트웨어학부

과 목 : 분산프로그래밍1

담당교수 : 김정호 교수님

학번 : 60191677

이름 : 이은비 (서명)

목 차

Ⅰ. 서론

시스템이란 요소가 모여서 하나 이상의 목적을 완전하게 달성하는 것으로, 우리가 개발하려는 시스템에는 항상 관심을 가지는 Stakeholder 들이 있다. 그렇게 모이는 관심을 concern 이라고 하며 이는 항상 모호성이 있다. 모호성이란 하나의 단어를 들었을 때 두 개 이상의 이미지가 떠오르는 것이다. 요구사항을 만들기 위해 Concern 을 분석해야 하는데 분석에서 중요한 것은 기능과 비기능을 Orthogonal 하게 나눠야 한다는 것이다.

분석을 통해 모호성을 없앤 요구사항을 코딩으로 구현하기에 앞서 구현하는 방법을 미리 정의해야 한다. 이를 설계라고 한다. 소프트웨어 설계는 눈에 보이지 않기 때문에 언어를 통해서 설계를 하며, 기능 요구사항의 설계 모델로 Class Diagram 을 사용할 것이다.

비기능은 입력값 대비 출력값의 변화를 제약하는 요소들을 의미하는데, 이는 품질이라고도 부른다. 본 레포트에서는 비기능을 품질이라 칭한다.

소프트웨어를 구성하기 위해서는 컴포넌트와 컴포넌트의 관계를 추상적인 구조로 나타내야 한다. 이때 품질이 보이는 구조를 아키텍처라 한다. 소프트웨어 아키텍처는 시스템을 운영하는 사람, 이해하는 사람, 관리하는 사람에 따라 다른 View Point 에 맞춰 각각 그려져야 한다.

본 레포트는 2021 년 2 학기 강의 '분산 프로그래밍 2' 중간고사 대체 레포트로, 우리는 현 강의에서 신동아화재 NSP TFT 의 NSP IT 시스템 구축 제안 요청서(RFP)를 기반으로 품질을 Runtime View, Module View, Allocation View 에 따라 세 개의 소프트웨어 아키텍처를 그리고 각각의 아키텍처에서 어떤 품질이 보이고, 그 품질을 나타내기 위해 어떤 전략과 패턴을 사용했는지에 대해 말해보고자 한다.

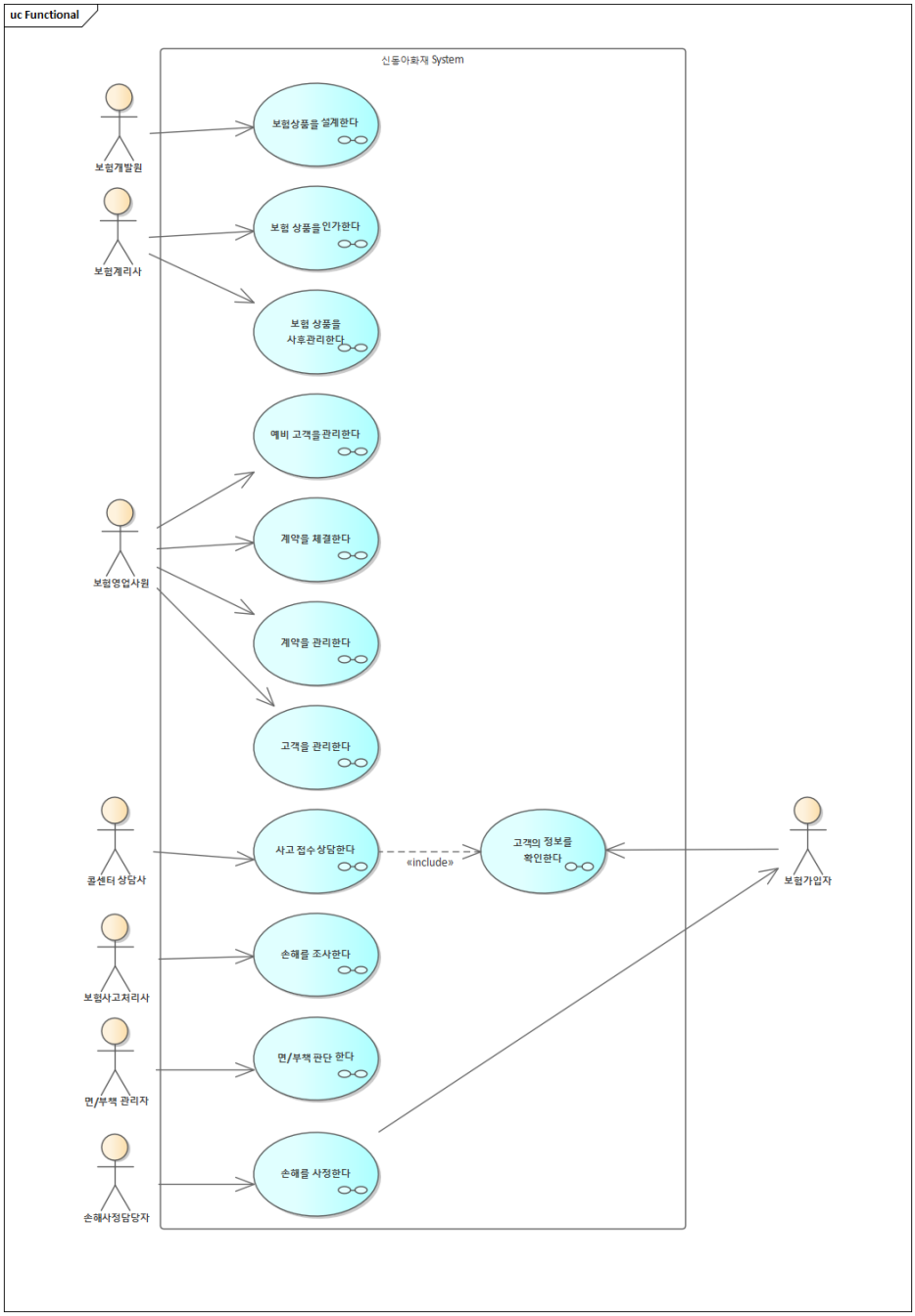
.

Ⅱ. 본론

1. Requirements

Requirements 분석을 통해 기능 요구사항을 Use Case Diagram 으로, 품질 요구사항을 QAS(Quality Attribute Scenarios)로 기술하였다.

A. Functional Requirements



위의 Use Case Diagram 은 신동아화재 NSP TFT 의 NSP IT 시스템 구축 제안 요청서(RFP)를 기반으로 작성되었으며, 구현하고자 하는 신동아화재 시스템에는 여러 Actor 가 존재하고, 이들에게는 고유의 Use Case 가 있다. 본 절에서는 각 Use Case 에 대한 간략한 설명만을 기술하였다.

1) 개발

Use Case	Description
보험 상품 설계	보험 개발원이 보험사 시스템을 통해 보험 상품을 설계한다. 설계된 상품은 인가를 받아야 한다.
보험 상품 인가	보험 계리사가 개발된 신규 보험 상품을 인가하여 통과된 상품은 영업 시 판매 가능한 상품으로 노출된다.
보험 상품 사후관리	보험 계리사가 인가 완료된 상품에 대해 사후 관리한다. 판매된 상품에 대해서는 판매 실적표를 작성할 수 있다.

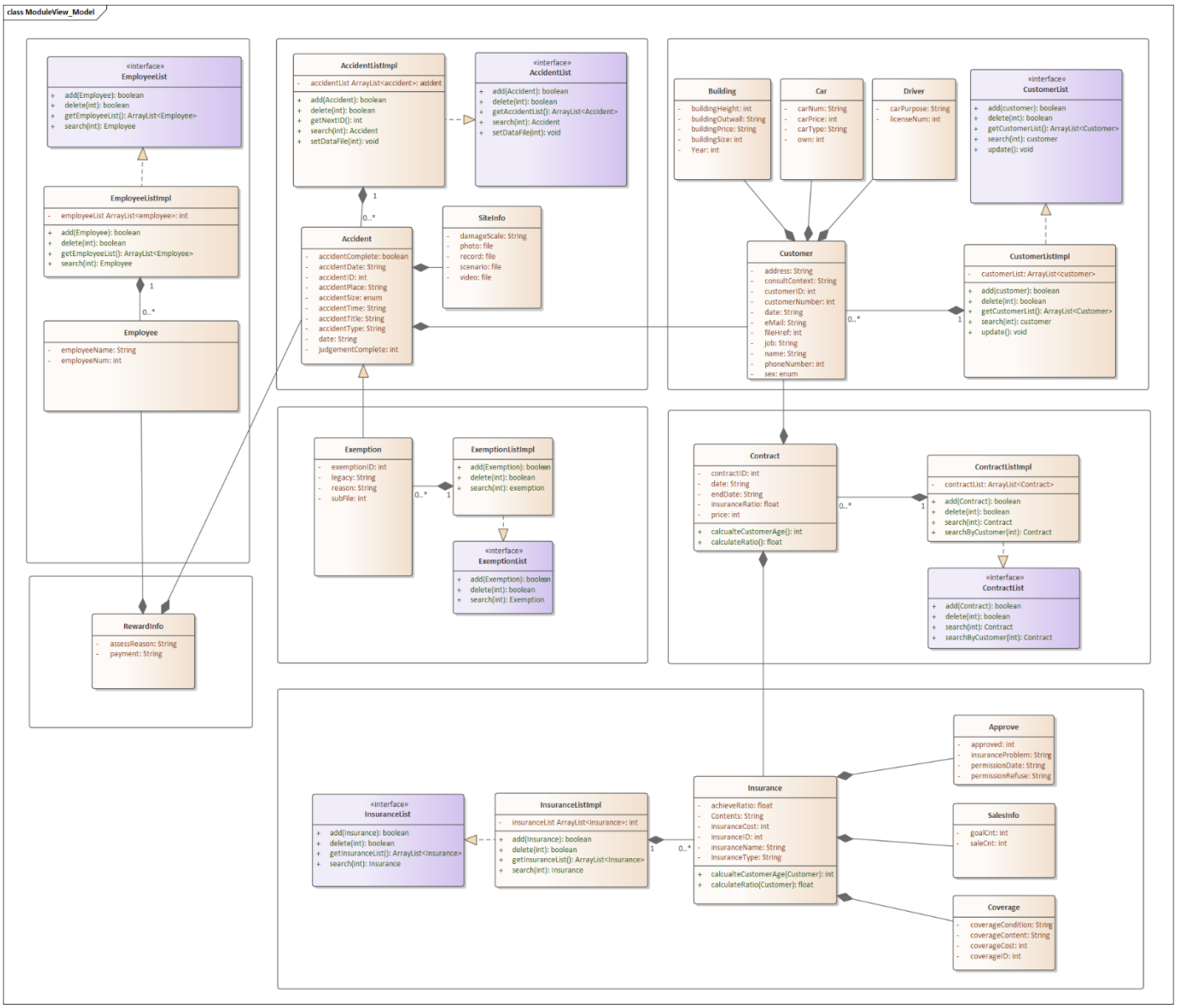
2) 영업

Use Case	Description
예비 고객 관리	보험 영업 사원이 보험사에 등록되지 않은 예비 고객을 관리한다. 예비 고객과의 상담을 통해 고객의 자사 사이트에 회원가입, 고객이 관심 있는 상품 정보를 자사 시스템에 저장한다.
계약 체결	보험 영업 사원이 자사 사이트에 회원 가입된 고객 중 계약을 진행하고자 하는 고객에 대해 계약을 체결하고, 이를 자사 시스템에 저장한다.
계약 관리	보험 영업 사원이 고객의 요청에 따라 이전에 체결한 계약에 대해 계약 기간 연장 혹은 신규 계약 체결을 진행한다.
고객 관리	보험 영업 사원이 맡은 고객의 정보에 변경사항이 있을 경우, 수정한다.

3) 보상

Use Case	Description
고객 정보 확인	상담원은 사고 접수하려는 고객이 자사와 계약한 이력이 있는지 확인 후, 있다면 사고 접수 상담을 진행하고 없다면 자사 사이트에 회원가입을 진행한다.
사고 접수 상담	상담원은 고객의 사고 정보를 자사 사이트에 저장한다.

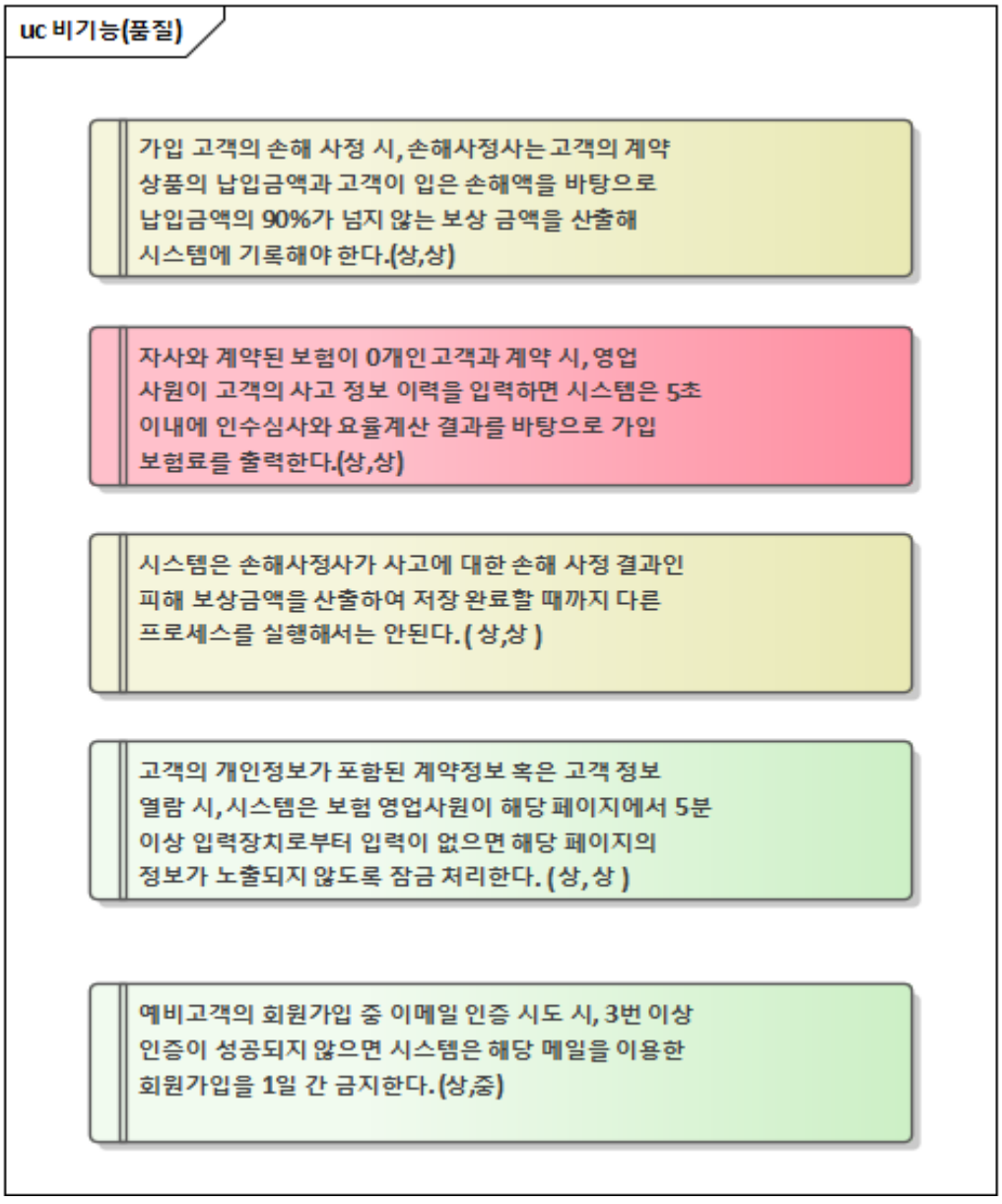
손해 조사	사고처리사가 사고 현장에서 고객에게 발생한 손해를 조사하고, 이를 자사 사이트에 저장한다.
면/부책 판단	면/부책 관리자가 손해 조사가 완료된 사고에 대해 고객의 면/부책을 판단하고 그 근거를 기록한다.
손해 사정	손해사정사가 면/부책 판단이 완료된 사고에 대해 판단의 근거를 기준으로 손해 사정 정보를 기록하고, 이를 고객에게도 전달한다.



위의 Class Diagram 은 Use Case Scenario 를 바탕으로 구현된 보험사 시스템의 구조를 보여준다.

B. QAS(Quality Attribute Scenario)

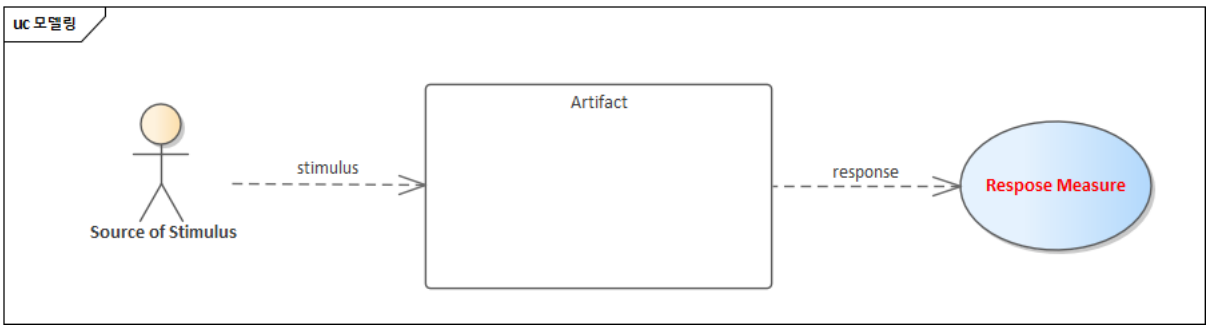
QAS 는 품질 속성을 보여주기 위한 시나리오다. 품질이 나쁘면 시장에서 외면 받기 때문에 QAS 작성은 중요하다.



위의 Diagram 은 신동아화재 NSP TFT 의 NSP IT 시스템 구축 제안 요청서(RFP)를 기반으로 하며, 우선순위와 개발난이도가 가장 높은 순으로 5 개를 선택하였다.

품질 속성	속성 상세화	시나리오	우선순위	
			중요성	난이도
안정성	계산 정확성	손해사정사가 처리하는 업무의 계산이 정확해야 한다.	H	H
	인터럽트 최소화	손해사정사의 업무로 인해 작업을 방해받지 않아야 한다.	H	H
성능	작업의 실시간성	영업 사원이 입력한 값의 결과 출력까지 5초 이내에 완료되어야 한다.	H	H
보안	고객 프라이버시 존중	고객의 개인 정보 열람의 경우, 해당 프로세스의 작업이 5분간 이루어지지 않으면 잠금 처리한다.	H	H
	올바른 사용자 확인	개인 정보 도용 방지를 위해 올바른 사용자가 아닐 경우, 회원 가입을 1일 간 금지한다.	H	M

1) 품질 속성에 따른 시나리오 작성 틀[1]



Stimulus	자극원에서 생성 혹은 도출되는 행위. 자극원에서 대상체에게 보내지거나 전달되는 이벤트나 명령
Stimulus Source	자극을 만들어내는 존재. 시나리오가 시작되는 행위의 주체
Environment	자극이 발생할 때의 상황이나 조건에 따른 시스템의 상태
Artifact	자극이 도달하는 최종 목적지. 자극을 받는 구성 요소.
Response	시스템이 반응하는 특정 행동
Response Measure	시나리오의 응답이 품질 속성을 어떻게 만족하는가

[1] 강승준,『소프트웨어 아키텍처 설계 가이드』프리렉 (2013), P.60-61

2) 안정성[2]

안정성은 시스템이 에러 없이 동작하는 것을 말한다. 시스템 실패와 이로부터 영향을 받는 다른 시스템의 요소들과 연관된 속성으로, 시스템의 중단 없이 서비스가 정상적으로 동작하는 것을 말한다.

- 계산 정확성

안정성	가입 고객의 손해 사정 시, 손해사정사는 고객의 계약 상품의 납입금액과 고객이 입은 손해액을 바탕으로 납입금액의 90%가 넘지 않는 보상금액을 산출해 시스템에 기록해야 한다.	상	상

Stimulus	고객의 계약 상품의 납입 금액과 고객이 입은 손해액을 바탕으로 고객의 보상 금액을 기록한다.
Stimulus Source	손해사정사
Environment	가입 고객의 손해를 사정하는 상황
Artifact	보험사 DB
Response	손해사정사가 입력한 정보를 바탕으로 시스템이 자동으로 보상 금액 계산한다.
Response Measure	보상금액이 납입금액의 90%를 넘지 않았는가?

- 인터럽트 최소화[3]

[2] 강승준,『소프트웨어 아키텍처 설계 가이드』프리렉 (2013), p.35

[3] https://www.geeksforgeeks.org/interrupts/

안정성	시스템은 손해사정사가 사고에 대한 손해 사정 결과인 피해 보상금액을 산출하여 저장 완료할 때까지 다른 프로세스를 실행해서는 안된다.	상	상

Stimulus	손해 사정 결과(피해 보상금액)를 산출하여 저장한다.
Stimulus Source	시스템
Environment	손해 사정 결과를 산출해서 저장하는 상황
Artifact	보험사 DB
Response	작업에 방해되지 않게, 또다른 손해사정 결과 산출 및 저장을 막는다.
Response Measure	작업 도중, 또다른 손해 사정이 진행되었는가?

인터럽트란 프로세스 또는 이벤트에 즉각적인 반응이 필요할 때, 하드웨어 또는 소프트웨어에서 발생하는 신호이다. 현재 진행중인 작업 프로세스를 중단해야 하는 우선순위가 높은 프로세스에 대해 프로세서에 경고한다.

3) 성능

성능이란 시스템이 얼마나 바르고 효율적으로 원하는 기능을 수행하는 가를 말한다.

- 작업의 실시간성

성능	자사와 계약된 보험이 0개인 고객과 계약 시, 영업 사원이 고객의 사고 정보 이력을 입력하면 시스템은 5초 이내에 인수심사와 요율 계산 결과를 바탕으로 가입 보험료를 출력해야 한다.	상	상

Stimulus	영업 사원이 고객의 사고 정보 이력을 입력한다.
Stimulus Source	영업 사원
Environment	자사와 계약된 보험이 0개인 고객과의 계약 상황
Artifact	영업 사원의 Web Browser
Response	고객 정보에 대한 인수심사와 요율 계산 결과를 바탕으로 가입 보험료 산출한다.
Response Measure	시나리오의 응답이 품질 속성을 어떻게 만족하는가.

4) 보안

보안[4]이란 적법한 사용자에 대해서는 서비스를 제공하고, 권한이 없는 사용자에 대해서는 서비스를 거부하는 시스템을 의미한다. 보안에서 고려해야 할 요소는 인증, 권한, 기밀성 등이 있다.

- 고객 프라이버시 존중

보안	고객의 개인정보가 포함된 계약정보 혹은 고객 정보 열람 시, 시스템은 보험 영업사원이 해당 페이지에서 5분 이상 입력 장치로부터 입력이 없으면 해당 페이지의 정보가 노출되지 않도록 잠금 처리한다.	상	상

Stimulus	영업 사원이 고객 개인 정보를 열람한다.
Stimulus Source	영업 사원
Environment	고객의 개인 정보가 담긴 페이지에서 5분 이상 입력 장치로부터 입력이 없는 상황
Artifact	개인 정보가 담긴 페이지
Response	해당 페이지의 정보 노출을 잠금 처리한다.
Response Measure	5분 이상 입력 장치로부터 입력이 없는 경우, 해당 페이지를 잠갔는가.

- 올바른 사용자 확인

보안	예비고객의 회원가입 중 이메일 인증 시도 시, 3번 이상 인증이 성공되지 않으면 시스템은 해당 메일을 이용한 회원가입을 1일 간 금지한다.	상	중

Stimulus	예비고객의 이메일 인증 성공을 기다린다.
Stimulus Source	시스템
Environment	3회 이상 이메일 인증 실패 상황

[4] 강승준,『소프트웨어 아키텍처 설계 가이드』프리렉 (2013), p.36

Artifact	예비고객의 이메일
Response	시스템은 해당 이메일을 이용한 회원가입을 1일 간 금지한다.
Response Measure	3회 이상 인증 실패한 이메일로 24시간이 지나기 전에 회원가입에 성공하였는가.

2. Strategy for Quality Attributes

1) 안정성 – 계산 정확성[5]

시나리오	손해사정사가 처리하는 업무의 계산이 정확해야 한다.
품질 속성	안정성
환경	가입 고객의 손해를 사정하는 상황
자극	고객의 계약 상품의 납입 금액과 고객이 입은 손해액을 바탕으로 고객의 보상 금액을 기록한다.
응답	손해사정사가 입력한 정보를 바탕으로 시스템이 자동으로 보상 금액 계산한다.
아키텍처	Exception (결함 탐지)
설명	계산 Logic을 수행하는 서비스가 응답이 없을 경우, Exception을 throw한다.
아키텍처	Active Redundancy (결함 복구)
설명	계산 Logic을 수행하는 서비스를 처리하는 Thread를 하나 더 준비한다. 같은 서비스를 처리하는 2개의 Thread 중 이벤트 발생 시, 가장 먼저 반응하는 하나의 Thread만 남기고 다른 하나는 버려진다. 만일, active Thread에 결함이 발생하면 다른 Thread가 대치한다.

- Exception: 결함이 발생하면, 적절한 예외 처리를 수행한다. 예외는 누락, 정지, 타이밍, 응답 중 하나가 인식되는 상황에서 발생한다. (하나의 프로세스 내에서 사용한다.)
- Active Redundancy: 특정 이벤트에 동시에 같이 반응하는 중복 컴포넌트를 준비한다. 이때 각 컴포넌트는 같은 상태를 갖는다. 이벤트가 발생하면 이에 가장 먼저 반응하는 하나의 컴포넌트 응답만이 사용되고 나머지는 버려진다. 이 경우 한 컴포넌트에 결함이 생기면 같은 다른 컴포넌트를 대치함으로써 시스템 다운 시간을 최소화할 수 있다.

[5] 강승준,『소프트웨어 아키텍처 설계 가이드』프리렉 (2013), P.97-100

2) 안정성 – 인터럽트 최소화[6]

시나리오	손해사정사의 업무로 인해 작업을 방해받지 않아야 한다.
품질 속성	안정성
환경	손해 사정 결과를 산출해서 저장하는 상황
자극	손해 사정 결과(피해 보상금액)를 산출하여 저장한다.
응답	작업에 방해되지 않게, 또다른 손해사정 결과 산출 및 저장을 막는다.
아키텍처	Thread Pooling
설명	Task Queue에 Task를 쌓아 1개의 Task에 1개의 Thread가 배정되고, Task가 완료되면 해당 Thread에 다음 Task를 배정한다.

- Thread Pooling: n개의 Thread를 미리 준비해두고, 요청이 들어왔을 때 하나의 Thread와 매칭시킨다. 이때, 작업을 진행 중인 Thread에 오류가 생기면 대기하고 있는 다른 Thread에 매칭해 손해사정사의 업무에 지장이 생기지 않도록 한다.

3) 성능 – 작업의 실시간성

시나리오	영업 사원이 입력한 값의 결과 출력까지 5초 이내에 완료되어야 한다.
품질 속성	성능
환경	자사와 계약된 보험이 0개인 고객과의 계약 상황
자극	영업 사원이 고객의 사고 정보 이력을 입력한다.
응답	고객 정보에 대한 인수심사와 요율 계산 결과를 바탕으로 가입 보험료 산출한다.
아키텍처	Thread Pooling
설명	Task Queue에 Task를 쌓아 1개의 Task에 1개의 Thread가 배정되고, Task가 완료되면 해당 Thread에 다음 Task를 배정한다. 실시간성으로 Task를 완료하기 위해서, 클라이언트-서버 간 요청과 응답에 1초를 가정할 때, Task-Thread 배정과 Task 완료가 4초 이내에 이루어져야 한다.

[6] 멀티 프로세스 vs. 멀티 쓰레드
https://wooody92.github.io/os/%EB%A9%80%ED%8B%B0-%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4%EC%99%80-%EB%A9%80%ED%8B%B0-%EC%8A%A4%EB%A0%88%EB%93%9C/

- Thread Pooling: n 개의 Thread 를 미리 준비해두고, 요청이 들어왔을 때 하나의 Thread 와 매칭시킨다. Task Queue 에 쌓인 Task 를 다 처리할 때까지, 놀고 있는 Thread 가 발생하지 않도록 하여 작업의 실시간성을 보장한다.

4) 보안 – 고객 프라이버시 존중[7]

시나리오	고객의 개인 정보 열람의 경우, 해당 프로세스의 작업이 5분간 이루어지지 않으면 잠금 처리한다.
품질 속성	보안성
환경	고객의 개인 정보가 담긴 페이지에서 5분 이상 입력 장치로부터 입력이 없는 상황
자극	영업 사원이 고객 개인 정보를 열람한다.
응답	해당 페이지의 정보 노출을 잠금 처리한다.
아키텍처	Limit Exposure (노출 제한)
설명	호스트가 일정 시간 작업을 하지 않을 경우, 화면에 노출되는 정보를 잠금 처리한다.

- Thread 에서 호스트가 일정 시간 입력 장치로부터 동작을 하지 않는 것을 감지하고 화면에 노출되는 고객의 개인정보 보호를 위해 페이지 자체를 잠금처리한다.
- Limit Exposure : 시스템의 사용량을 제안하여 과도한 시스템 사용을 줄여 리소스 손실을 낮추기 위한 아키텍처이다. 페이지를 잠금으로써, 고객의 정보 보호와 동시에 시스템 사용량을 줄여 시스템에 과부하가 가는 것을 막을 수 있다.

5) 보안 – 올바른 사용자 확인[8]

시나리오	개인 정보 도용 방지를 위해 올바른 사용자가 아닐 경우, 회원 가입을 1일 간 금지한다.
품질 속성	보안성
환경	3회 이상 이메일 인증 실패 상황
자극	예비고객의 이메일 인증 성공을 기다린다.
응답	시스템은 해당 이메일을 이용한 회원가입을 1일 간 금지한다.
아키텍처	Authenticate Users (사용자 인증)
설명	이메일 인증으로 사용자가 누구인지 확실하게 보장한다.

[7] 강승준,『소프트웨어 아키텍처 설계 가이드』프리렉 (2013), P.106-108

[8] https://to-dy.tistory.com/92

	(*AuthenticationCredentialNotFoundException* 3회 이상 인증 실패 시, Exception 던지기.)
아키텍처	Authorize Users (사용자 인가)
설명	인증된 사용자에게 데이터나 서비스에 접근/수정하기 위한 권리를 사용자에 따라 권한을 부여한다.

- Authenticate Users: 이메일 인증, 아이디, 비밀번호 인증, 생체 정보 인증 등 다양한 인증 방식으로 사용자가 누구인지 확실하게 보장한다.
- Java Spring Framework *AuthenticationCredentialNotFoundException* 예외: 인증 요구가 거부됐을 때 던지는 예외이다.
- Authorize Users: 사용자 인증이 된 사용자에게 시스템의 데이터나 서비스에 접근 혹은 수정을 할 수 있는 권리를 부여한다.

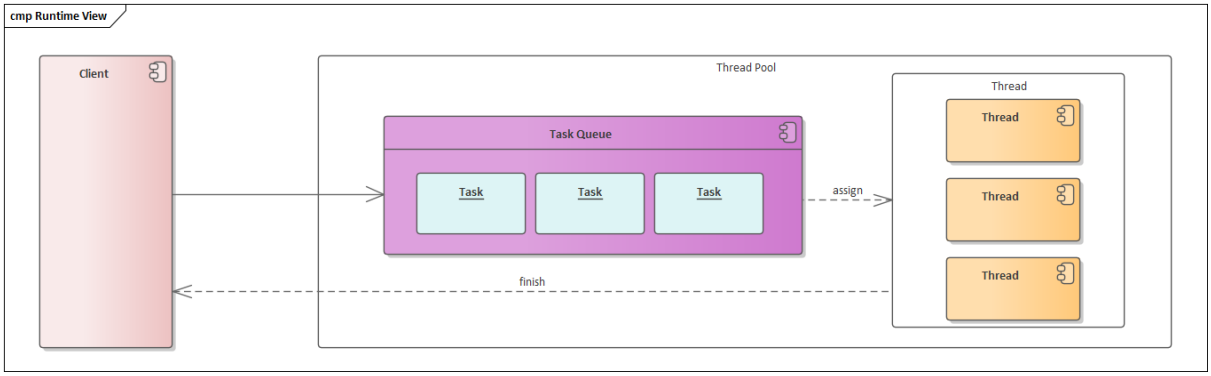
---

3. View Point

A. Runtime View

Runtime View 에 적용한 스타일은 아래와 같이 설명할 수 있다.

(1) Thread Pooling

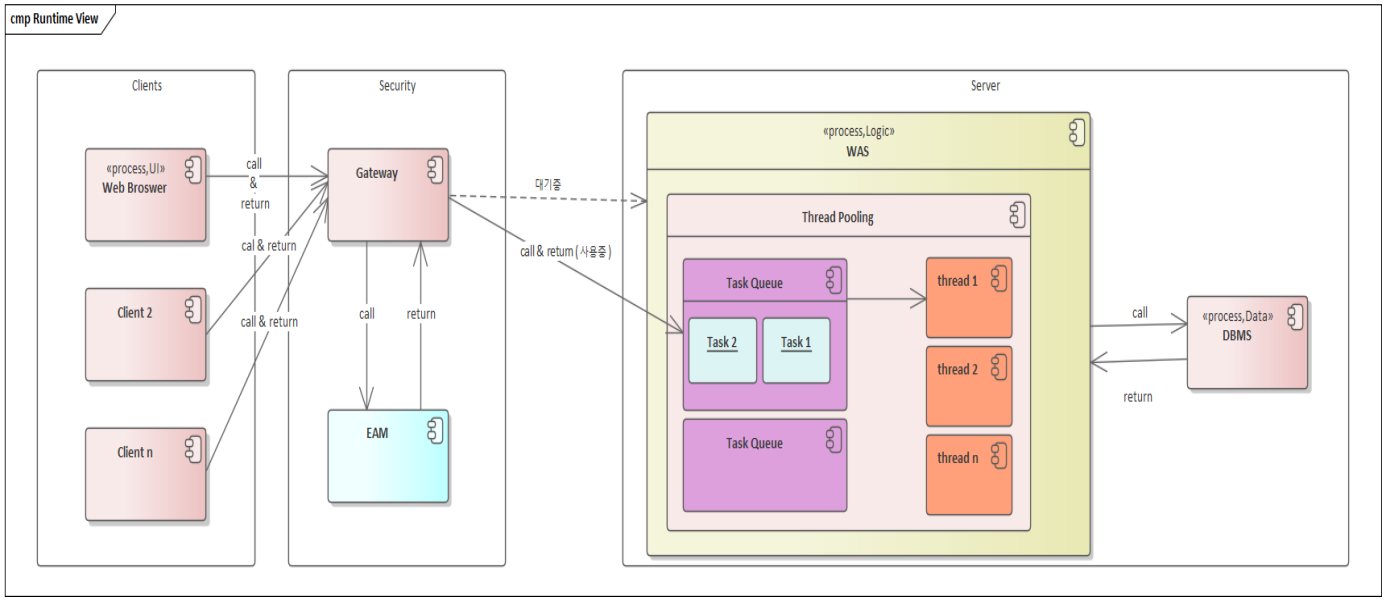


N 명의 Client 가 Server 에 접근하여 시스템의 데이터나 서비스를 접근 혹은 수정하려고 하는 것을 Task 라고 할 때, 1 명의 Client 에는 WAS 내부의 1 개의 Task Queue 와 매핑된다. 1 명의 Client 가 N 개의 Task 를 처리하려고 할 때, N 개의 Task 는 Task

Queue 에 쌓이고, 이를 Thread 에 하나씩 매핑하여 Task 를 처리한다. Task 가 완료되면, Queue 내부에 다른 Task 를 매핑한다.

Thread Pool 을 생성하여 병렬 처리를 할 수 있도록 해주는 것이 Java 의 `java.util.concurrent.Executors` 와 `java.util.concurrent.ExecutorService` 이다.[9] 위의 Diagram 처럼 Thred 를 미리 만들어두고, Client 가 Task 를 요청할 때마다 Thread 에 Task 를 할당하게 해준다.

(2) Call & Return



N 명의 Client 는 Server 와의 통신에서 Call & Return 방식으로 데이터나 서비스를 받는다. Client 는 Server 에 접근하기 위해서, Security Bondury 에 있는 Gateway 에 먼저 call 한다. Gateway 는 Client 가 고객, 상담원, 상담원을 제외한 보험사 직원 중 누구에 해당하는지 EAM 에 call 한다. EAM 은 Client 가 누구인지 판별해서 Gateway 에 return 한다. Gateway 는 return 된 정보를 갖고 WAS 에 Client 에 맞는 View 를 call 한다.

Client 가 누구인지에 상관없이 WAS 에 로그인 없이 접근을 시도하면, 고객 View 를 보여준다. 만일 직원이 로그인하여 WAS 에 접근하면, 그에 맞는 View 를 보여준다.

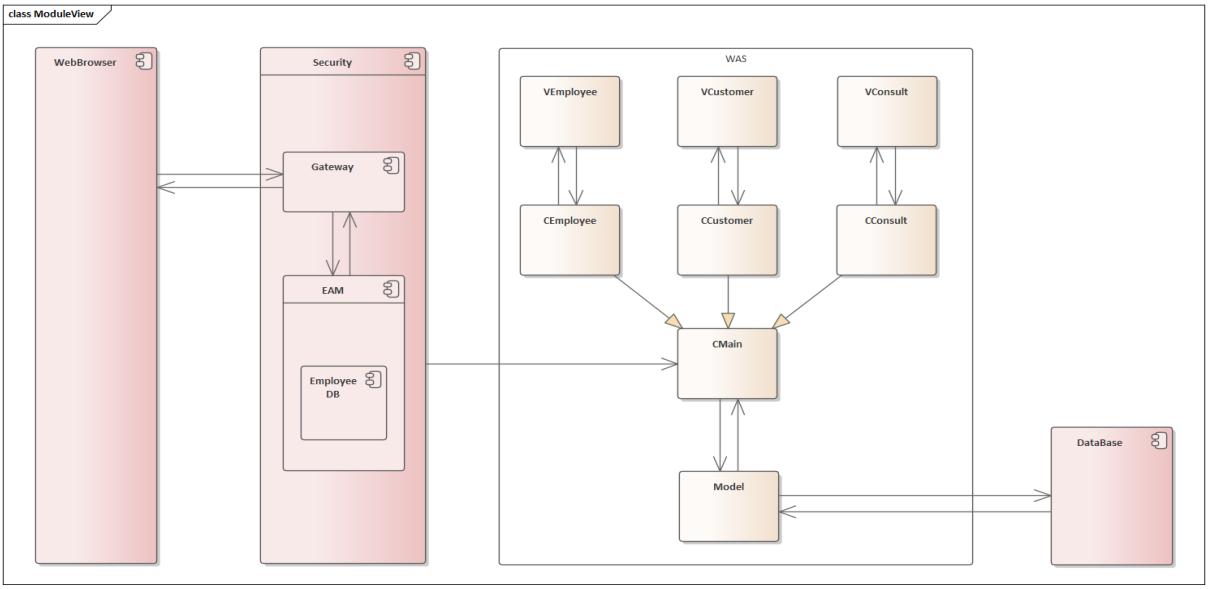
요소	설명
Web Browser	Client 에게 보여지는 화면이다. Default 값은 고객에게 보여지는 VIew 이다.

[9] https://codechacha.com/ko/java-executors/

GateWay	Client 가 Server 에 접근하기 위해 거치는 관문으로, Client 가 누구인지 판별하기 위해 EAM 에 Client 정보를 Call 한다. EAM 으로부터 Return 받은 Client 정보(누구인가에 대한 정보)를 Server 에 call 한다.
EAM	Server 에 접근하려는 Client 가 누구인지 판별하는 시스템으로, Gateway 로부터 받은 Client 정보와 보험사인적 DB 의 정보가 일치하면, 그에 맞는 View 를 보여주도록 Gateway 를 거쳐 Server 에 요청한다. 만일 보험사 인적 DB 에 일치하는 정보가 없으면 고객으로 간주하고 Gateway 를 거쳐 Server 에 고객 view 를 보여주도록 요청한다.
WAS	보험사 시스템 Server 로, Client 가 시스템의 데이터나 서비스에 접근하려고 call 하면 그에 맞는 응답을 return 해준다.
Task Queue	Client 가 Server 에 데이터나 서비스에 접근 혹은 수정하고자 하는 것을 하나의 Task 라고 한다. 1 명의 Client 에 1 개의 Task Queue 를 매칭하여, Client 가 하려는 Task 를 Queue 에 쌓는다.
Thread	Client 가 하려는 Task 를 처리하는 하나의 서비스이다.
Thread Pooling	Task Queue 에 쌓인 Task 와 Thread 를 일대일 매칭하여, 놀고 있는 Thread 가 발생하지 않도록 한다.
DBMS	보험사에서 발생하는 모든 데이터를 저장 및 관리하는 시스템이다.

B. Module View

Module View에 적용한 스타일은 아래와 같다.



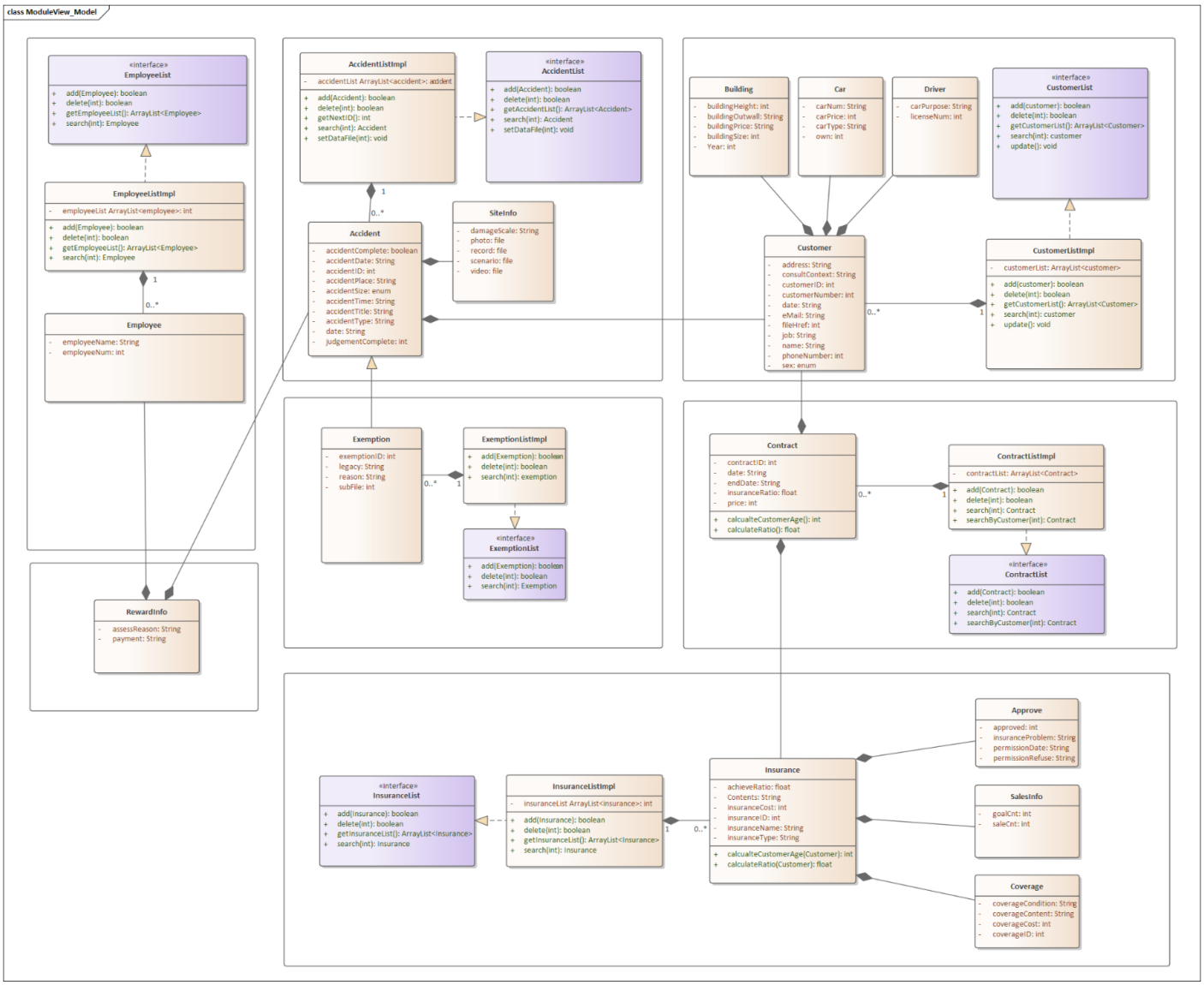
(1) MVC 패턴

데이터를 관리하는 Model과 UI를 관리하는 View, 제어를 담당하는 Controller로 분리한다. View와

Model의 연결을 Controller가 관리해 View의 추가, 변경, 삭제 등은 Model에 영향을 미치지 않고 Model의 변화 또한 View에 영향을 미치지 않게 하는 패턴이다. MVC가 각각 별도의 컴포넌트로 분리되어 있어 서로에게 영향을 받지 않고 작업을 수행해 재사용성과 유지보수의 편리성을 높여 준다.

(2) Polymorphism

Model과 View를 관리하는 Controller를 Polymorphism을 적용해 여러 타입의 객체들을 하나의 타입으로 관리해 Controller의 유지보수성 뿐만 아니라, 전체 MVC 패턴 구조의 유지보수성을 높인다.

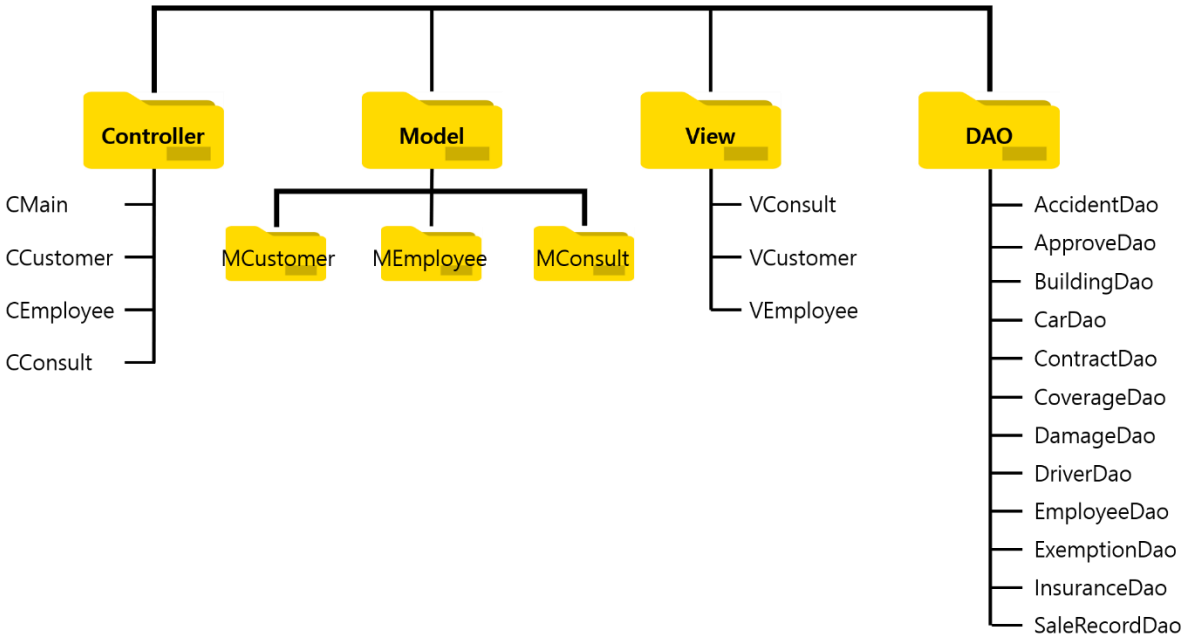


(3) Association & Aggregation & Generalization

Association은 요소끼리 연결이 되어 있어 서로 지속적으로 영향을 주는 관계이다. 클래스끼리 관계를 맺을 때 Association이 많으면 유지보수성이 좋지 않게 된다. 이를 해결하기 위해서는 Associatino 대신 Aggregation과 Generalization의 비율을 높여야 한다.

위의 Module View를 살펴보면, 큰 관점에서 Contorller 끼리의 연결을 제외한 나머지는 Asssociation으로 연결되어 있지만 MVC 패턴의 내부를 자세히 살펴보면 Association은 보이지 않고 Aggregation과 Generalization으로 요소들이 붙어 있는 것을 확인할 수 있다. 이를 통해 이 아키텍처는 유지보수성이 좋을 것을 예상할 수 있다

C. Allocation View (Implementation Style)



보험사 시스템을 구성하기 위해 Runtime View 와 Module View 에서 도출한 컴포넌트와 모듈을 바탕으로 각 실제 구현 파일로 역할을 할당해보면 위의 그림과 같이 나타난다.

Ⅲ. 결론

시스템이란 요소가 모여서 하나 이상의 목적을 완전히 달성하는 것을 의미한다. 지금까지 우리는 보험사 시스템 개발을 위해 고객의 모호한 Concern 을 분석하고 기능 요구사항을 만들어 Use case Diagram 과 Class Diagram 을 만들어보았다. 이렇게 완성된 시스템은 철저히 기능 요구사항만 충족하게 되는데, 이런 시스템은 시장에서 외면 받는다는 문제점이 있다. 시장에서 시스템이 살아남기 위해서는 품질이 좋아야 한다.

우리는 본 레포트를 통하여 우리가 만든 보험사 시스템에 필요한 품질 정의해보았고, 정의한 품질들을 높이기 위해 목적에 맞는 아키텍처 패턴과 전략을 선택하고 이를 아키텍처로 그려보았다.

신동아화재 NSP TFT 의 NSP IT 시스템 구축 제안 요청서(RFP)를 기반으로 구현된 보험사 시스템의 품질을 높이기 위해 QAS 를 작성하고, 우선순위와 개발 난이도가 높은 5 개의 시나리오로부터 안정성, 성능, 보안 품질 속성을 도출했다.

안정성, 성능, 보안 품질 속성을 높이기 위해 우리가 선택한 아키텍처 전략은 다음과 같다.

1) 안정성을 보여주기 위해 Thread pool 로 하나의 Queue 에 Task 를 쌓고, 이를 미리 할당한 Thread 에 일대일 매칭으로 Task 의 중단이 발생하지 않도록 한다. 만약 Task 를 처리하던 Thread 에 이상이 생길 경우, Exception 으로 결함을 탐지해서 알린다. 그리고 결함을 복구하기 위해 Active Redundancy 로 같은 서비스를 하는 Thread 를 하나 더 만들어 두고, 이것으로 대체하여 Task 의 정확성을 높인다.

2) 성능을 보여주기 위해 call & return (Client-Server) 패턴으로 통신한다. Thread pool 로 미리 할당한 Thread 중 노는 Thread 가 발생하지 않도록 Task Queue 의 Task 와 일대일 매칭을 한다. Task 가 완료되면 다음 Task 를 바로 배정하여 Task 처리를 실시간으로 하도록 한다.

3) 보안을 보여주기 위해 Client-Server 패턴에 Gateway-EAM 을 추가하였다. Client 가 Server 에 접근하기 위해서는 Gateway 를 거쳐 Gateway 가 EAM 으로부터 Client 를 누구인지 확인한다. EAM 에서 긍정응답이 온다면 그에 맞는 View 를 보여주도록 Server 에 요청하고, 부정응답이 온다면 잘못된 접근이라 판단해 튕겨낸다. Thread 에서는 입력 장치로부터 일정 시간 입력이 없을 경우, 고객의 개인정보 보호를 위해 해당 페이지를 잠근 처리한다.

시스템의 가장 중요한 품질 속성은 유지보수성이다. 이를 보여주기 위해 MVC 패턴과 Polymorphism 을 사용하였다. Client 가 시스템을 통해 받길 원하는 서비스는 모두 다르고 그에 사용되는 데이터 역시 다르기 때문에, Sever 는 1 명의 Client 에 각각 MVC 패턴을 적용하여 View,

Controller, Model 이 View 와 Model 의 변화가 다른 컴포넌트에 영향을 미치지 않도록 기능하게 하였다. Controller 는 Polymorphism 을 적용해, 여러 타입의 객체를 하나의 타입으로 관리하도록 하였다.

시스템의 품질을 높이기 위해서는 QAS 를 작성해 최우선이 되어야 할 품질이 무엇인지 결정하고, 이를 바탕으로 각 품질에 맞는 아키텍처를 선택하여 어떤 관점에서 보아도 그 품질이 보이도록 해야 한다. 본 레포트를 통해 결정한 아키텍처들이 실제 구동이 되었을 때 어떤 효과를 보일지 기대해본다.