3. 아키텍처 설계 문서화 사례

> "좋은 사례가 주는 성가심보다 더 참기 힘든 것은 없다" [1] – Mark Twain

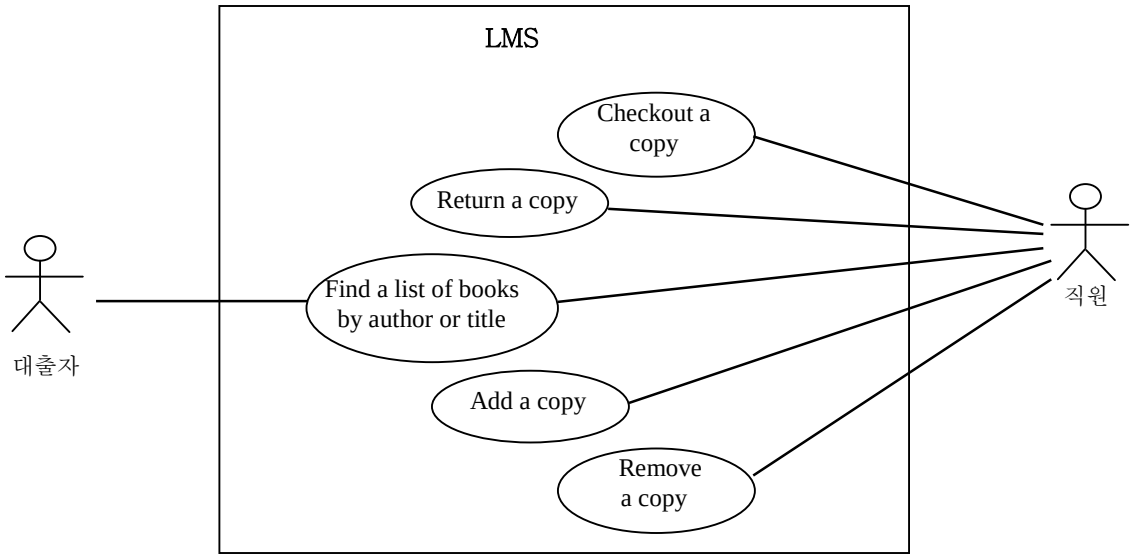
이 장에서는 도서관 관리 시스템(LMS: Library Management System) 아키텍처 설계 사례를 소개한다. 작은 설계 사례를 통하여 이 책에서 소개한 기본적인 원리의 적용을 볼 수 있고 또한 그림 1-1의 아키텍처 설계 문서 목차를 따르는 문서화의 사례를 보인다.

A. 시스템 개요

LMS는 도서관이 소장하고 있는 책을 관리하는 시스템으로 도서의 대출, 반납, 검색, 새 도서의 추가 및 삭제기능을 갖는 시스템으로 시작하여 점차 고도화된 기능을 갖도록 진화할 계획이다. 관리대상이 되는 정보는 데이터베이스로 관리한다. 시스템은 최초에 100명의 사용자를 지원할 수 있고 향후 사용자 수가 늘어날 경우 확장 가능해야 한다.

B. 시스템 요구사항

아래 그림은 LMS 시스템의 Use Case 다이어그램이다. 행위자로 직원과 대출자가 있다.



[1] "Few things are harder to put up with than the annoyance of a good example."

B.1 기능요구사항

LMS의 사용 시나리오:

ID	UC01
이름	Checkout a copy
행위자	직원
설명	대출자의 도서 대출을 처리한다.
선행조건	해당 도서가 도서관에 있다.
절차	1. 도서ID, 도서명, 대출자명, 날짜를 입력 받는다. 2. 해당 도서의 대출요청을 처리한다.

ID	UC02
이름	Return a copy
행위자	직원
설명	대출자의 도서 반납을 처리한다.
선행조건	해당 도서가 대출된 상태이다.
절차	1. 도서ID를 입력 받는다. 2. 해당 도서의 반납요청을 처리한다.

ID	UC03
이름	Find a list of books by author or title
행위자	대출자, 직원
설명	저자 혹은 제목으로 도서를 검색한다.
선행조건	N/A
절차	1. 저자 혹은 제목을 입력 받는다. 2. 해당 도서목록을 출력한다

ID	UC04
이름	Add a copy
행위자	직원
설명	도서를 등록한다.
선행조건	해당 도서가 등록되어 있지 않다.
절차	1. 도서ID, 도서명, 출판사, 저자를 입력 받는다. 2. 해당 도서를 등록한다.

ID	UC05
이름	Remove a copy
행위자	직원
설명	도서를 삭제한다.
선행조건	해당 도서가 등록되어 있다.
절차	1. 도서ID를 입력 받는다. 2. 해당 도서 정보를 삭제한다.

B.2 품질요구사항

품질 속성 트리를 사용한 LMS의 품질요구사항 기술

품질 속성	속성 상세화	ID	시나리오	우선순위	
				중요성	난이도
QA1. 사용용이성	인터페이스 적응	QAS1-1	처음으로 시스템을 사용하는 직원도 30분 이내에 모든 기능을 수행할 수 있다. 처음 사용하는 대출자는 바로 사용 가능해야 한다.	M	M
QA2. 성능	응답속도	QAS2-1	최대 부하*가 걸린 상태에서 사용자의 요청에 0.1초 내에 응답을 완료한다.	M	M
	처리량	QAS2-2	최대 부하*가 걸린 상태에서 초당 100개의 요청에 대한 응답을 완료한다.	H	H
	수용량	QAS2-3	최초 100명의 사용자를 지원할 수 있어야 한다	H	L
QA3. 확장용이성	규모확장성 (scalability)	QAS3-1	1년 후, 200명의 사용자를 지원할 수 있어야 한다.	H	H
	확장개발성 (extendibility)	QAS3-2	새로운 기능 추가 시, 5개 이내의 모듈만 영향을 받는다.	H	M

* 100명의 사용자가 접속했을 때 H: High, M: Medium, L: Low

B.3 제약사항

ID	제약사항
CR1	Java ORB 미들웨어를 사용한다.
CR2	오픈 소스 데이터베이스를 사용한다.
CR3	시스템은 x86기반의 리눅스(커널 3.5.x)서버에서 운용된다.

C. 아키텍처 문제 분석

C.1 선정된 아키텍처 드라이버

아키텍처 드라이버 ID	요구사항 ID	선정이유
AD1	CR1	시스템에 직접 관련된 제약사항
AD2	CR2	시스템에 직접 관련된 제약사항
AD3	QAS2-2	우선 순위(H, H)
AD4	QAS3-1	우선 순위(H, H)
AD5	QAS3-2	우선 순위(H, M)

Note) 아키텍처 드라이버의 우선 순위는 미정

C.2 아키텍처 문제 분석표

설계문제	요구사항 (아키텍처 드라이버)	설계전략	이유 (이득/비용/위험요인/불확실성 포함)	관련사항 (관련전략/파급효과/완화전략)
Issue 0. 제약사항의 구현	AD1, AD2	AS00. Java ORB 1.6 미들웨어를 사용한다. AS01. Mysql 오픈소스 데이터 베이스를 사용한다.	- 제약사항이기 때문에 준수되어야 한다. - Mysql은 CR3에서 요구된 리눅스 환경에 최적화되어 있다.	
Issue 1. 사용자 증가에 대비	AD4	AS10. 서버 측의 부하를 최소하기 위하여 Multi-threading을 사용한다. AS11. 클라이언트-서버 아키텍처 스타일을 적용한다.	-기존 시스템에 대한 변경 없이 향후 사용자 증가에 대비할 수 있다. -서버의 성능이 어느 정도로 보장되어야 하는지에 대한 지표가 없다.	
Issue 2. 향후 계속적인 기능 확장	AD5	AS20. 시스템을 계층화한다. AS21. 모듈별로 캡슐화하고, 기능별로 메소드를 나눈다.	-각각의 컴포넌트가 독립성을 갖도록 하여 한 컴포넌트의 변경이 다른 컴포넌트에 영향을 주지 않도록 한다. -코드 변경 시에. 수정이 용이해진다.	
Issue 3. 성능	AD3, AD4	AS30. Multi-threading를 이용하여 개발한다.	- Multi-threading를 이용하면 오버헤드를 최소화 하여, 요청 작업들을 동시에 처리할 수 있다.	

QR: Quality Requirement, CR: Constraint Requirement

AS: Architecture Strategy, DR: Design Rational

D. 아키텍처 설계 절차

D.1 아키텍처 스타일

여러 명의 대출자 혹은 직원이 서버에 DB의 도서정보에 대한 저장, 삭제, 수정, 검색을 요청하면 이에 대해서 서버가 응답하는 방식으로 진행된다. 이는 대출자 혹은 직원이라는 클라이언트가 서버와 커뮤니케이션 하는 관계이고 Issue 1에 대한 적절한 아키텍처 설계전략이므로, 요청/응답 커넥션을 사용하는 클라이언트-서버 아키텍처 스타일을 선정한다. (AS11. 클라이언트-서버 아키텍처 스타일을 적용한다.)

D.2 아키텍처 관점체계

제 II부의 참조 관점체계의 여덟 개의 뷰 중 개념 뷰를 제외한 일곱 개의 뷰를 다음 순서로 도출한다:

(1) 시나리오뷰
(2) 논리뷰
(3) 모듈뷰
(4) 실행뷰
(5) 실행요소 배치뷰
(6) 코드뷰
(7) 코드 배치뷰

D.3 아키텍처 설계 절차의 정의

시스템의 규모가 작기 때문에 최상위와 차상위의 두 개 수준의 설계로 충분하며, 다음의 순서로 진행된다:

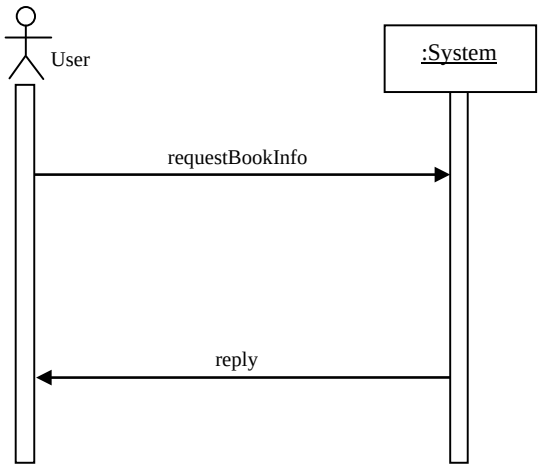
(1) D.2절의 관점체계의 각 뷰에 대한 Level-1(최상위) 설계를 수행
(2) D.2절의 관점체계의 각 뷰에 대한 Level-2(차상위) 설계를 수행

E. 아키텍처 설계결과

E.1 Level – 1 설계

E.1.1 시나리오뷰

여기에서는 시스템의 여러 가지 사용 시나리오 중 하나의 대표적인 시나리오만을 보인다.



E.1.2 논리뷰

AS11. 클라이언트-서버 아키텍처 스타일을 적용한다.

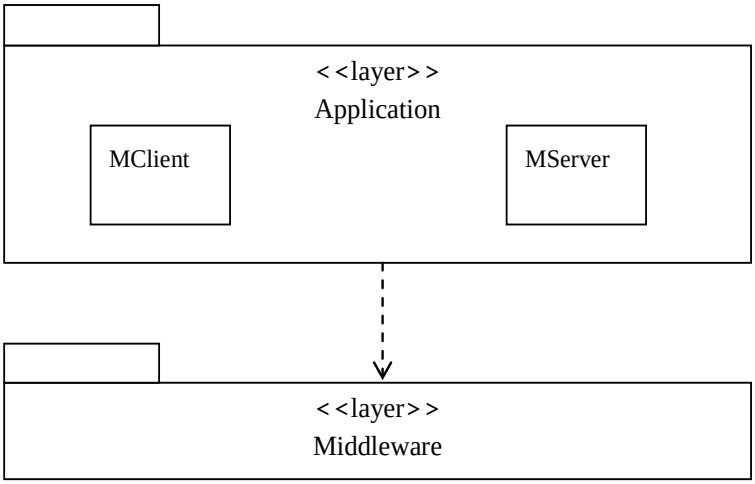
시나리오뷰에서 논리뷰로의 맵핑

시나리오뷰		논리뷰	
이름	종류	이름	종류
System	객체	LClient	컴포넌트
		LServer	컴포넌트

E.1.3 모듈뷰

AS00. Java-ORB 플랫폼을 사용한다.

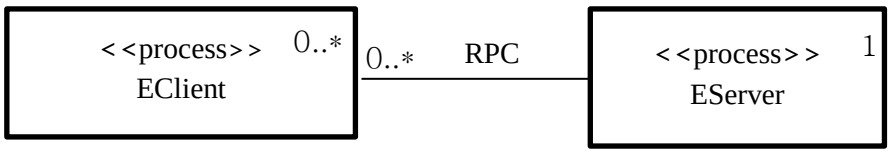
AS20. 시스템을 계층화한다.


<<layer>>
Application
MClient
MServer
<<layer>>
Middleware

논리뷰에서 모듈뷰로의 맵핑

논리뷰		모듈뷰	
이름	종류	이름	종류
LClient	컴포넌트	MClient MClientManager	모듈
LServer	컴포넌트	MServer MServerImpl	모듈

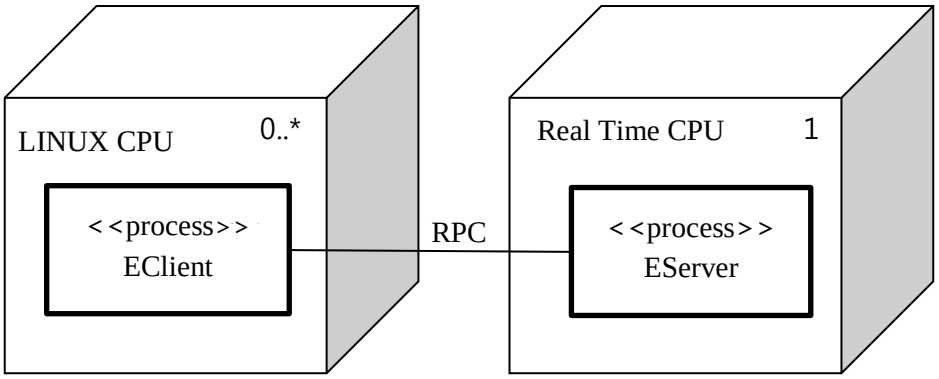
E.1.4 실행뷰



모듈뷰에서 실행뷰로의 맵핑

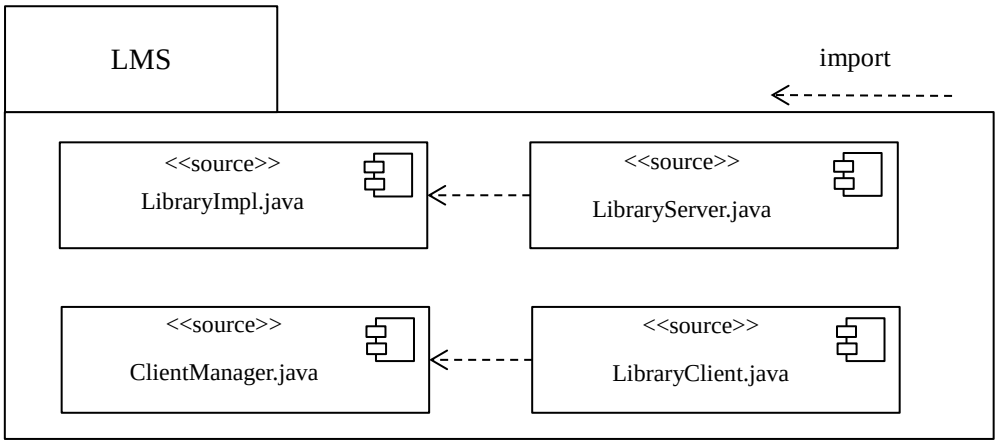
모듈뷰		실행뷰	
이름	종류	이름	종류
Client-side Subsystem			
MClient MClientManager	모듈	EClient	프로세스
Server-side Subsystem			
MServer MServerImpl	모듈	EServer	프로세스

E.1.5 실행요소 배치뷰



E.1.6 코드뷰

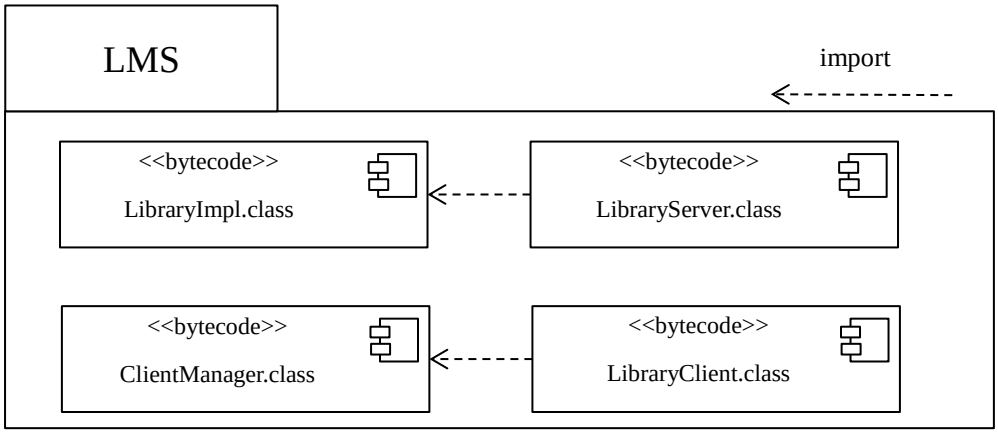
소스코드 코드뷰



실행뷰에서 소스코드 코드뷰로의 맵핑

실행뷰		코드뷰		설명
이름	종류	디렉토리	파일	
EClient	프로세스		LibraryClient.java ClientManager.java	LibraryClient.java 파일이 ClientManager.java를 import
EServer	프로세스		LibraryServer.java LibraryImpl.java	LibraryServer.java 파일이 LibraryImpl.java를 import

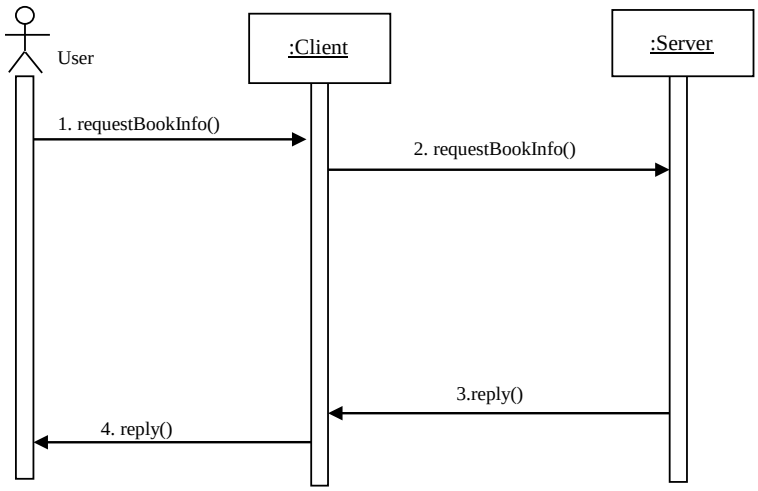
바이트코드 코드뷰



E.1.7 코드 배치뷰

E.2 Level – 2 설계

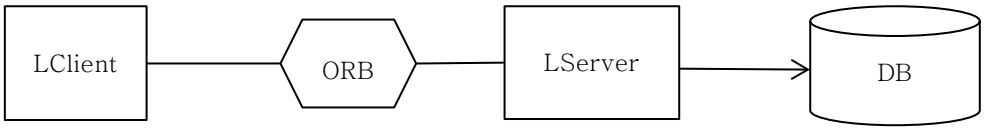
E.2.1 시나리오뷰



E.2.2 논리뷰

AS00. Java ORB 미들웨어를 사용한다.

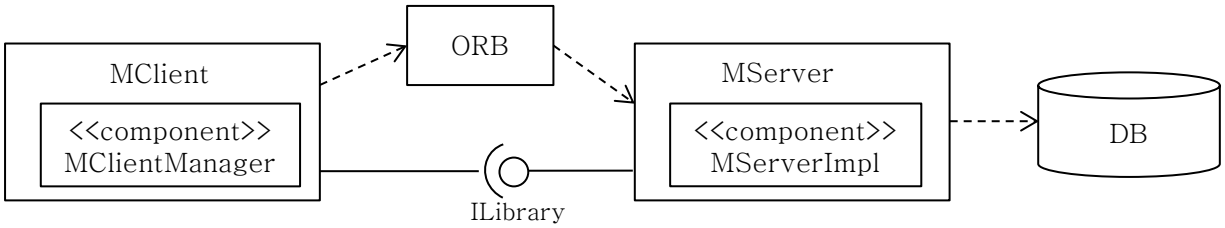
AS01. 오픈 소스 DB를 사용한다.

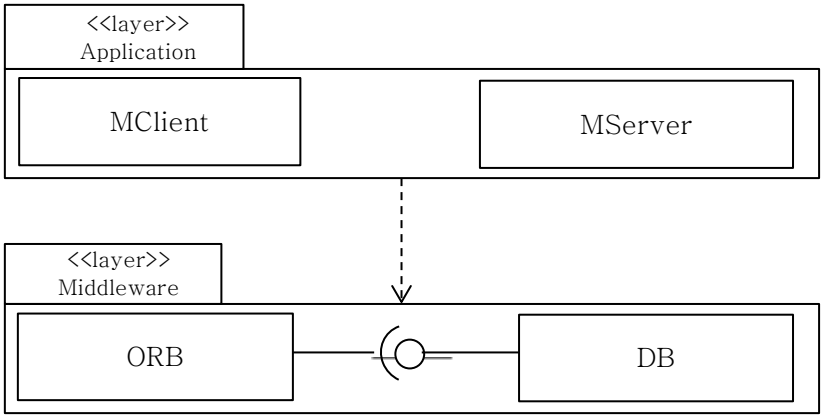


시나리오뷰의 논리뷰로의 맵핑

시나리오뷰		논리뷰	
이름	종류	이름	종류
LClient	컴포넌트	LClient	컴포넌트
		ORB	컴포넌트
LServer	컴포넌트	LServer	컴포넌트
		DB	컴포넌트

E.2.3 모듈뷰





논리뷰에서 모듈뷰로의 맵핑

논리뷰		모듈뷰	
이름	종류	이름	종류
LClient	컴포넌트	MClient MClientManager	모듈
LServer	컴포넌트	MServer MServerImpl	모듈
ORB	컴포넌트	ORB	모듈
DB	컴포넌트	DB	모듈
		ILibrary	인터페이스

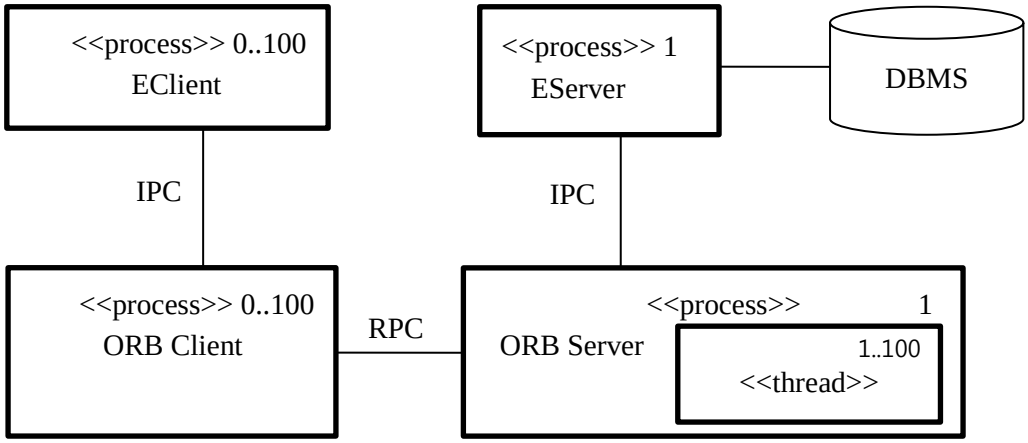
인터페이스 기술

AS21. 모듈별로 캡슐화하고, 기능별로 메소드를 나눈다,

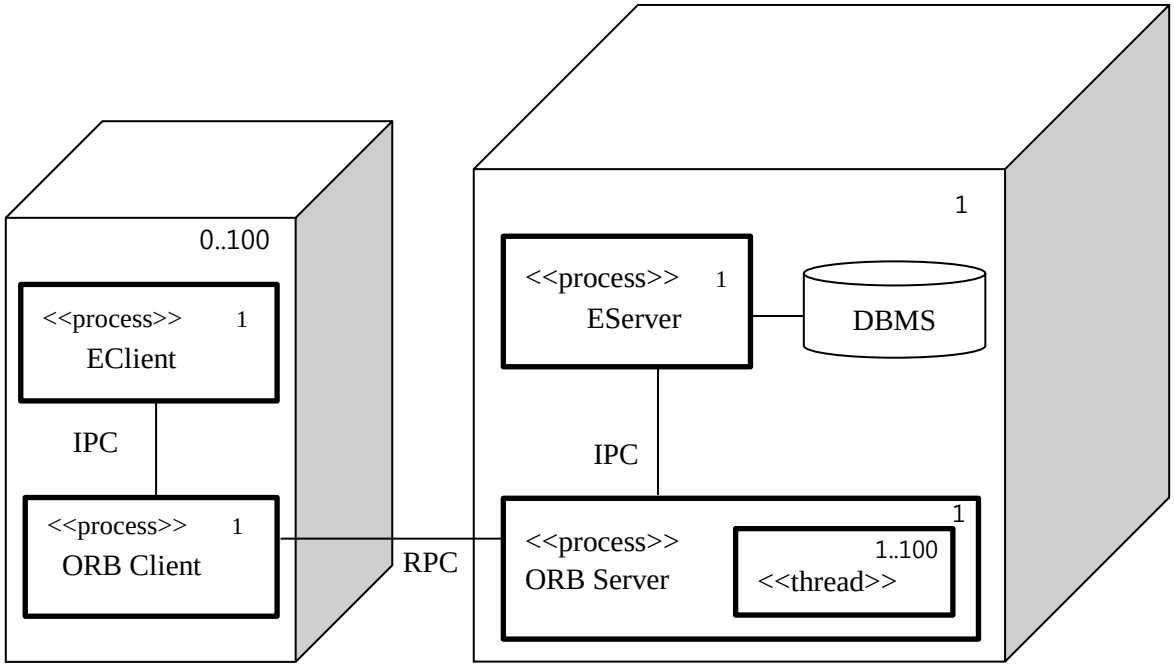
ILibrary 인터페이스		
소개		LMS시스템에 대한 기능을 제공한다.
경로		Library\library.idl
제공 메소드 리스트		
메소드 이름		checkoutCopy
	소개	도서를 대출한다.
	시그너처	in string id, in string userName, in string userId
	파라미터	id : 도서 ID, username : 사용자명, userId : 사용자 주민등록번호
	사전조건	현재 대출중인 도서가 아니어야 한다.
	사후조건	해당 사항 없음.
	예외처리	현재 다른 사용자에 의해 대출된 도서이면 대출을 허용하지 않는다.
메소드 이름		returnCopy
	소개	도서를 반납한다.
	시그너처	in string userId, in string bookId
	파라미터	userId : 사용자 주민등록번호, bookId : 도서 ID
	사전조건	userId를 가진 사용자가 bookId를 가진 도서를 현재 대출 중이다.
	사후조건	해당 사항 없음.
	예외처리	userId를 가진 사용자가 현재 bookId를 가진 도서를 현재 대출 중이지 않다면 반납이 불가능하다.
메소드 이름		findBooksByAuthor
	소개	저자를 검색어로 하여 일치하는 도서를 검색한다.
	시그너처	in string author, out short count
	파라미터	author : 도서의 저자, count : 검색된 도서의 개수
	사전조건	등록된 도서목록에 포함되어 있는 도서에 한해 검색된다.
	사후조건	해당 사항 없음.
	예외처리	등록되지 않은 도서는 출력하지 않는다.
메소드 이름		findBooksByTitle
	소개	도서제목을 검색어로 하여 일치하는 도서를 검색한다.
	시그너처	in string title, out short count
	파라미터	title : 도서제목, count : 검색된 도서의 개수
	사전조건	등록된 도서목록에 포함되어 있는 도서에 한해 검색된다.
	사후조건	해당 사항 없음.
	예외처리	등록되지 않은 도서는 출력하지 않는다.
메소드 이름		addCopy
	소개	도서를 리스트에 등록한다.
	시그너처	in string ISBN, in string title, in string id, in string author
	파라미터	ISBN : 도서의 ISBN, title : 도서제목, id : 도서 ID, author : 도서저자
	사전조건	같은 ID를 가진 도서가 등록되어 있지 않아야 한다.
	사후조건	해당 사항 없음.
	예외처리	ID가 같은 도서가 이미 등록되어 있을 경우, 등록이 불가능하다.
메소드 이름		removeCopy
	소개	도서를 리스트에서 삭제한다.
	시그너처	in string id
	파라미터	id : 도서 ID
	사전조건	리스트에 등록된 도서이어야 한다.
	사후조건	해당 사항 없음.
	예외처리	리스트에 등록되지 않은 도서는 삭제할 수 없다.

E.2.4 실행뷰

AS10, AS30. Multi-threading을 적용한다.



E.2.5 실행요소 배치뷰

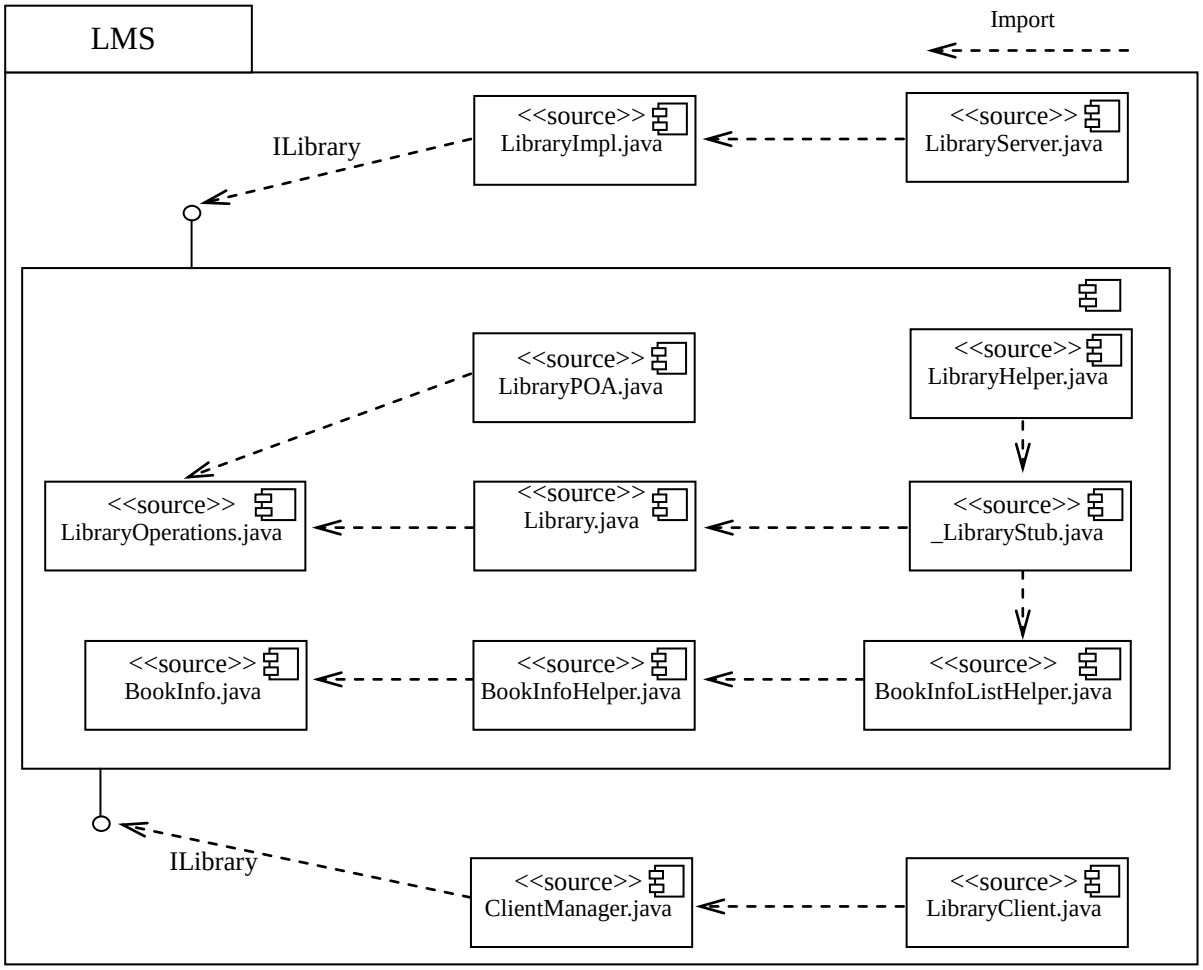


모듈뷰에서 실행뷰로의 맵핑

모듈뷰		실행뷰	
이름	종류	이름	종류
Client-side Subsystem			
MClient MClientManager	모듈	EClient	프로세스
Server-side Subsystem			
MServer MServerImpl	모듈	EServer	프로세스
Middleware Subsystem			
ORB	모듈	Client ORB Server ORB	프로세스
DB	모듈	DBMS	프로세스
ILibrary	인터페이스		

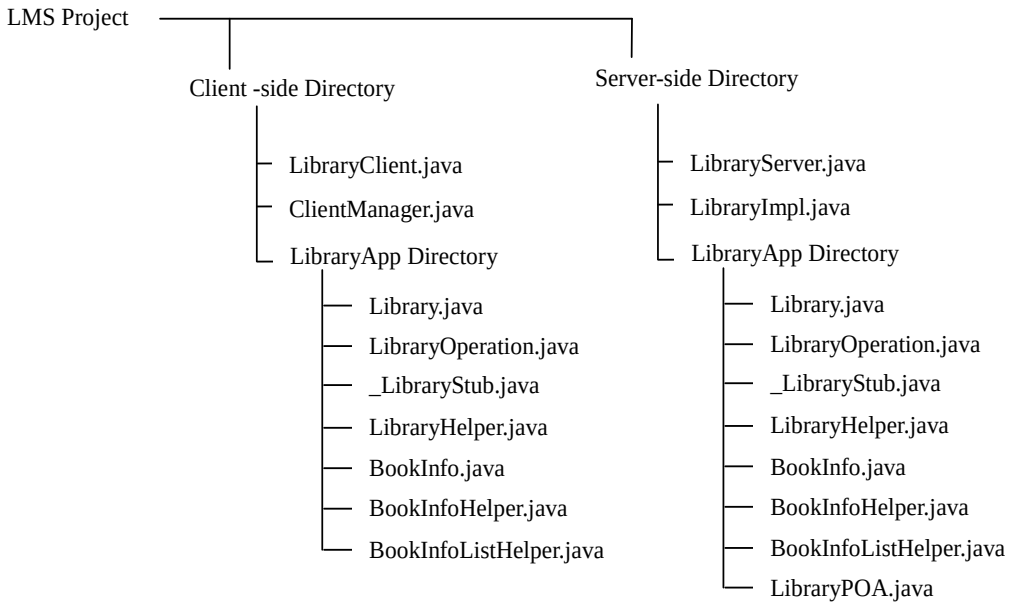
E.2.6 코드뷰

소스코드 코드뷰



코드 조직도

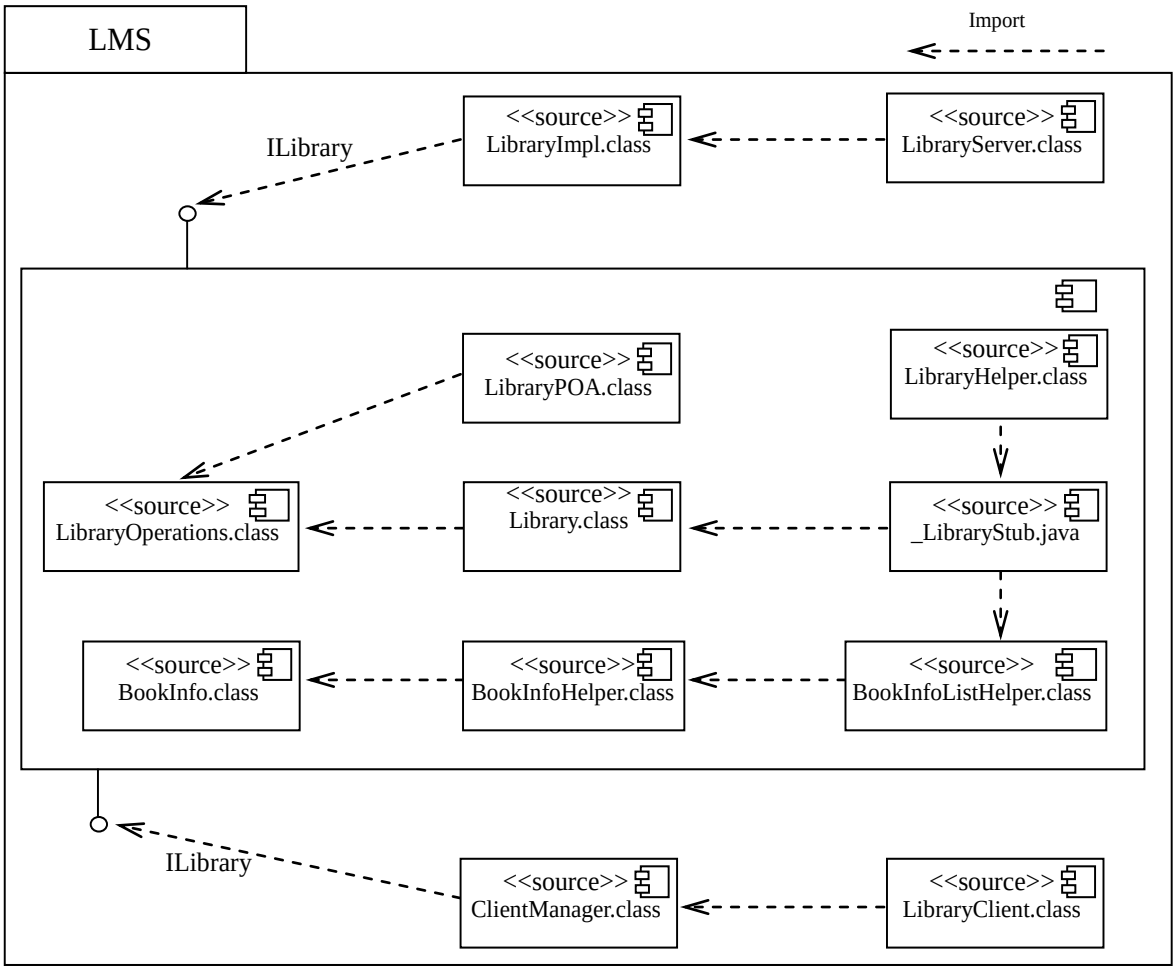
AS21. 모듈별로 캡슐화하고, 기능별로 메소드를 나눈다.



실행뷰에서 소스코드 코드뷰로의 맵핑

실행뷰		코드뷰		설명
이름	종류	디렉토리	파일	
EClient	프로세스		LibraryClient.java ClientManager.java	LibraryClient.java 파일이 ClientManager.java를 import
EServer	프로세스		LibraryServer.java LibraryImpl.java	LibraryServer.java 파일이 LibraryImpl.java를 import
ORB Client	프로세스	LibraryApp		Server-side와 Client-side 모두 LibraryApp을 필요로 함
ORB Server	프로세스	LibraryApp		Server-side와 Client-side 모두 LibraryApp을 필요로 함

바이트코드 코드뷰



배치 산출물의 정의

Directory	File
Client-side Directory	LibraryClient.jar
Server-side Directory	LibraryServer.jar

E.2.7 코드 배치뷰

F. 아키텍처 설계결과의 분석

F.1 아키텍처 추가 문제 분석표

추가 문제 분석 내용 없음.

F.2 아키텍처 설계 결정표

설계결정	해결조건/방법	설계전략	적용점
ADD1	서버와 클라이언트 개발이 독립적으로 이루어져, 보다 쉬운 개발이 이루어 질 수 있도록 한다.	AS00	E.2.2 논리뷰
ADD2	개발자들 사이에서 많이 알려져 있으며, 사용하기 쉽고 편리한 오픈소스 DBMS를 사용해야 한다.	AS01	E.2.2 논리뷰
ADD3	성능을 높이기 위해 Multi-threading 기법을 사용한다.	AS10	E.2.4 실행뷰
ADD4	클라이언트-서버 아키텍처 스타일을 적용한다.	AS11	E.1.2 논리뷰
ADD5	추후 기능확장을 고려하여 시스템을 계층화한다	AS20	E.1.3 모듈뷰
ADD6	모듈별로 캡슐화하고 기능별로 메소드를 구분한다.	AS21	E.2.3 모듈뷰

ADD: Architecture Design Decision

Note) AS10과 AS30는 동일한 전략이므로 둘 중에 한 가지만 표시하였다.

F.3 아키텍처 분석관점 및 분석결과

분석방법: 골격시스템(skeletal system)을 구현하여 각 분석관점을 시험한다.

분석관점		분석결과
사용용이성	처음으로 시스템을 사용하는 사용자도 30분 이내에 가능한 모든 기능을 수행 해 볼 수 있는가?	메뉴식 인터페이스의 경우 30분 이내 모든 기능을 수행 할 수 있다.
성능	100명의 사용자 접속 시, 사용자의 요청에 0.1초 이내로 응답이 오는가?	100명의 사용자 접속 시 요청에 대한 응답이 0.0007초가 걸린다.
	100명의 사용자 접속 시, 초당 100개의 요청에 대한 응답이 가능한가?	가능하다.
	100명의 사용자 접속 시, 원활히 동작하는가?	원활히 동작한다.
확장용이성	동시 접속 사용자의 최대 허용범위를 늘리는 과정에서 영향을 받는 모듈은 몇 개인가?	ORB library가 동시접속기능을 담당하고 있기 때문에 ORB 데몬을 사용하는 서버측의 모듈 1개만 수정하면 된다.
	200명의 사용자로 늘어날 경우, 수정해야 하는 모듈은 몇 개인가?	사용자의 수가 늘어나도 수정할 모듈은 없다.
	기능을 추가할 때, 영향이 미치는 모듈은 몇 개인가?	새로운 기능을 추가할 경우, 인터페이스와 해당 기능들을 관리하는 두 개의 파일만 수정하면 된다.

F.4 민감점과 상충점

분석항목	분석결과
민감점	전체적인 성능은 client 또는 server의 상태보다는 네트워크 상태에 더 민감하다.
상충점	시스템은 동시 접속한 사용자의 수를 제한하지 않기 때문에, 사용자 수와 응답시간은 상충관계에 있다.

F.5 위험요인

ID	위험요인/전략	확률	영향	완화전략
AS11: 클라이언트-서버 아키텍처 스타일을 적용한다.				
R1	클라이언트 컴퓨터에서 해킹을 통하여 시스템 서비스를 사용할 가능성이 있다.	낮음	적음 (인가되지 않은 사용자에 의한 시스템 서비스의 사용이 도서관 서비스에 큰 영향을 주지 않는다.)	없음(시스템에 높은 보안성이 요구되지 않는다.)

G. 아키텍처 평가

G.1 평가방법

E절에서 도출한 아키텍처를 평가하기 위하여 CBAM을 응용한 아키텍처 평가를 수행한다.

G.2 아키텍처 평가 결과

시나리오 수집 및 정제

시나리오	시나리오 출처
1	UC01
2	UC03
3	QAS2-2
4	QAS3-1
5	QAS3-2

시나리오 우선순위 결정

시나리오	가중치	응답 수준		
		최악의 경우	바람직한 경우	최상의 경우
1	5	구현됨	구현됨	구현됨
2	5	구현됨	구현됨	구현됨
3	25	50 개	100 개	150 개
4	35	110 명	200 명	210 명
5	30	10 개	5 개	3 개
합계	100			

Note) 도서관 관리 시스템은 새로이 개발되는 시스템이므로 "현재의 경우"는 고려되지 않았다.

응답수준별 효용값 부여

시나리오	가중치	효용값		
		최악의 경우	바람직한 경우	최상의 경우
1	5	100	100	100
2	5	100	100	100
3	25	30	90	100
4	35	10	90	100
5	30	20	90	100
합계	100			

아키텍처 전략별 관련 시나리오의 기대수준

전략	영향받는 시나리오	기대 수준
AS00	1	구현됨
	2	구현됨
AS01	해당없음	해당없음
AS10	3	100개
	4	200명
AS11	4	150명
AS20	5	5개
AS21	5	5개

AS01은 아키텍처 설계에 적용되어도 시나리오 1~5에 영향을 주지 않기 때문에 위의 표에서 기대 수준과 더불어 "해당없음"으로 표시하였다.

아키텍처 전략별 관련 시나리오의 기대효용값

전략	영향받는 시나리오	기대 효용값
AS00	1	100
	2	100
AS01	해당없음	0
AS10	3	90
	4	90
AS11	4	50
AS20	5	90
AS21	5	90

아키텍처 전략별 총 이득

전략	영향받는 시나리오	가중치	이득 원시데이터	정규화된 이득	총 이득
AS00	1	5	100	500	1,000
	2	5	100	500	
AS01	해당없음				0
AS10	3	25	90	2250	5,400
	4	35	90	3150	
AS11	4	35	50	1750	1750
AS20	5	30	90	2700	2,700
AS21	5	30	90	2700	2,700

Note) AS00 과 AS10 는 가중치가 각각 5 와 35 이지만, 영향받는 시나리오 – 즉 AS00 은 시나리오 1 과 2, AS10 은 시나리오 3 과 4 – 각각이 전략의 가충치에 비례하는 이득을 발생시킨다고 가정하였다.

아키텍처 전략별 비용대비 총 전략 이득

전략	비용	총 전략 이득
AS00	0	1,000
AS01	0	0
AS10	3,400	5,400
AS11	2,100	1,750
AS20	1,000	2,700
AS21	1,500	2,700
합계	8,000	13,550

E 절의 아키텍처 설계에 위의 모든 전략이 선택되었으므로, ROI = 13,550/8,000 ≈ 1.69 이다.

G.3 대안 아키텍처 및 평가

E절에서 도출한 선택된 아키텍처와 동일하게, B절의 시스템 요구사항과 C.1절의 아키텍처 드라이버에서부터 대안 아키텍처가 설계된다.

G.3.1 대안 아키텍처 문제 분석표

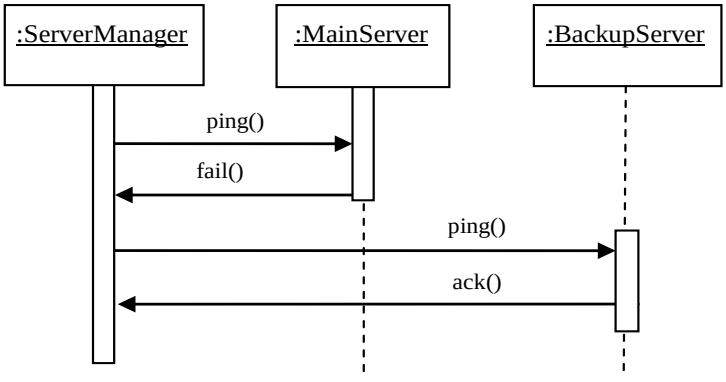
대안 아키텍처 설계를 위하여 C.2절의 아키텍처 분석에 추가적으로 다음의 설계문제가 고려되었다.

설계문제	요구사항	설계전략	이유 (이득/비용/위험요인/불확실성 포함)	관련사항 (관련전략/파급효과/완화전략)
Issue 4 신뢰성	신뢰성은 보편적인 요구사항이므로 명시적인 요구사항은 없음	AS40, 백업서버를 두어서 서버가 동작하지 않을 경우에도 백업서버로부터 기능을 사용 할 수 있도록 한다.	R2. 서버의 동작 중단 가능성	

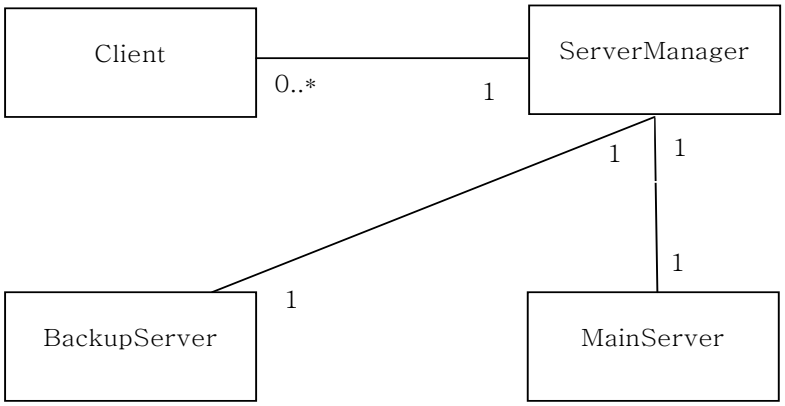
G.3.2 대안 아키텍처

AS40. 서버관리자 컴포넌트를 두어 주서버와 백업서버가 동작하는지를 주기적으로 확인하고 주서버가 동작하지 않을 경우 서비스 처리요청을 백업서버로 보낸다.

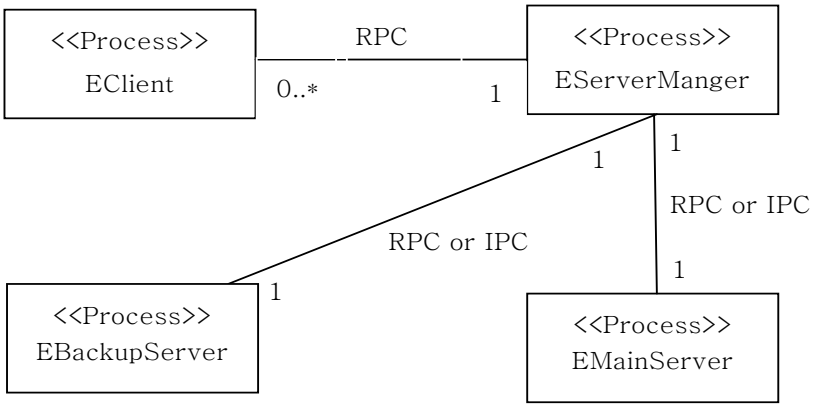
시나리오뷰



논리뷰



실행뷰



G.3.3 대안 아키텍처 설계 결정표

F.2 절의 아키텍처 설계결정 ADD1~ADD6 외에 설계전략 AS40 를 추가적으로 적용하는 결정을 통하여 대안 아키텍처를 얻는다.

설계결정	해결조건/방법	설계전략	적용점
ADD7	백업서버를 두어서 서버가 동작하지 않을 경우에도 백업서버로부터 기능을 사용 할 수 있도록 한다.	AS40	E.2.2 논리뷰

G.3.4 대안 아키텍처 평가

시나리오 우선순위 결정

시나리오	가중치	응답 수준		
		최악의 경우	바람직한 경우	최상의 경우
1	5	구현됨	구현됨	구현됨
2	5	구현됨	구현됨	구현됨
3	25	50 개	100 개	150 개
4	35	110 명	200 명	210 명
5	30	13 개	7 개	4 개
합계	100			

응답수준별 효용값 부여

시나리오	가중치	효용값		
		최악의 경우	바람직한 경우	최상의 경우
1	5	100	100	100
2	5	100	100	100
3	25	30	90	100
4	35	10	90	100
5	30	20	90	100
합계	100			

아키텍처 전략별 관련 시나리오의 기대수준

번호	영향받는 시나리오	기대 수준
AS00	1	구현됨
	2	구현됨
AS01	해당없음	해당없음
AS10	3	100 개
	4	200 명
AS11	4	150 명
AS20	5	7 개
AS21	5	7 개
AS40	1	구현됨
	2	구현됨

아키텍처 전략별 관련 시나리오의 기대효용값

번호	영향받는 시나리오	기대 효용값
AS00	1	100
	2	100
AS01	해당없음	0
AS10	3	90
	4	90
AS11	4	50
AS20	5	90
AS21	5	90
AS40	1	100
	2	100

아키텍처 전략별 총 이득

전략	영향받는 시나리오	가중치	이득 원시데이터	정규화된 이득	총 이득
AS00	1	5	100	500	1,000
	2	5	100	500	
AS01	해당없음				0
AS10	3	25	90	2,250	5,400
	4	35	90	3,150	
AS11	4	35	50	1,750	1,750
AS20	5	30	90	2,700	2,700
AS21	5	30	90	2,700	2,700
AS40	1	5	100	500	1,000
	2	5	100	500	

아키텍처 설계 비교

전략	선택된 아키텍처		대안 아키텍처	
	비용	총 전략 이득	비용	총 전략 이득
AS00	0	1,000	0	1,000
AS01	0	0	0	0
AS10	3,400	5,400	3,400	5,400
AS11	2,100	1.750	2,100	1,750
AS20	1,000	2,700	1,000	2,700
AS21	1,500	2,700	1,500	2,700
AS40	해당없음	해당없음	2,000	1,000
합계	8,000	13,550	10,000	14,550

선택된 아키텍처의 ROI = 13,550/8,000 ≈ 1,69 이고 대안 아키텍처의 ROI = 14,550/10,000 ≈ 1.46 이므로 선택된 아키텍처의 ROI 가 더 높다. 따라서 선택된 아키텍처가 더 나은 설계 결과로 평가된다.