아키텍팅, 원점에서 다시 보기

1. 출발 - 대상의 본질

시작은 가볍게...

구구단(TimesTable)을 출력하세요. →

```
2 x 1 =  2
2 x 2 =  4
2 x 3 =  6
2 x 4 =  8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
-----------
3 x 1 =  3
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
-----------
4 x 1 =  4
```

아주 간단하군요 !!

```
public void showTable() {
    //
    System.out.println(" Times table.");
    System.out.println(" ------------");

    for (int i = 2; i<=9; i++) {
        for (int j = 1; j <=9; j++) {
            System.out.format(" %2d x %d = %2d%n", i, j, (i*j));
        }

        System.out.println(" ------------");
    }
}
```

배려가 있는 코딩 !!

```
public void showTable() {
    //
    System.out.println("\n Times table.");
    System.out.println(" ------------");

    for (int leftNumber = 2; leftNumber<=9; leftNumber++) {
        for (int rightNumber = 1; rightNumber <=9; rightNumber++) {
            System.out.format(" %2d x %d = %2d%n",
                              leftNumber,
                              rightNumber,
                              (leftNumber*rightNumber));
        }

        System.out.println(" ------------");
    }
}
```

사랑이 변하듯 요건도 변하죠...

변경해 주세요. →

```
2 x 1 = 2     3 x 1 = 3     4 x 1 = 4     5 x 1 = 5
2 x 2 = 4     3 x 2 = 6     4 x 2 = 8     5 x 2 = 10
2 x 3 = 6     3 x 3 = 9     4 x 3 = 12    5 x 3 = 15
2 x 4 = 8     3 x 4 = 12    4 x 4 = 16    5 x 4 = 20
2 x 5 = 10    3 x 5 = 15    4 x 5 = 20    5 x 5 = 25
2 x 6 = 12    3 x 6 = 18    4 x 6 = 24    5 x 6 = 30
2 x 7 = 14    3 x 7 = 21    4 x 7 = 28    5 x 7 = 35
2 x 8 = 16    3 x 8 = 24    4 x 8 = 32    5 x 8 = 40
2 x 9 = 18    3 x 9 = 27    4 x 9 = 36    5 x 9 = 45

6 x 1 = 6     7 x 1 = 7     8 x 1 = 8     9 x 1 = 9
6 x 2 = 12    7 x 2 = 14    8 x 2 = 16    9 x 2 = 18
6 x 3 = 18    7 x 3 = 21    8 x 3 = 24    9 x 3 = 27
6 x 4 = 24    7 x 4 = 28    8 x 4 = 32    9 x 4 = 36
6 x 5 = 30    7 x 5 = 35    8 x 5 = 40    9 x 5 = 45
6 x 6 = 36    7 x 6 = 42    8 x 6 = 48    9 x 6 = 54
6 x 7 = 42    7 x 7 = 49    8 x 7 = 56    9 x 7 = 63
6 x 8 = 48    7 x 8 = 56    8 x 8 = 64    9 x 8 = 72
6 x 9 = 54    7 x 9 = 63    8 x 9 = 72    9 x 9 = 81
```

이것도 간단하긴 한데...

```
private static void printTimesTable(int columnCount) {
    //
    int loopCount = (8 / columnCount) + 1;

    int leftNumberOffset = 2;
    for(int i=0; i<loopCount; i++) {
        for(int rightNumber=1; rightNumber<=9; rightNumber++) {
            for(int count = 0; count < columnCount; count++) {
                int leftNumber = leftNumberOffset + count;
                if (leftNumber > 9) {
                    break;
                }
                printUnit(leftNumber, rightNumber);
            }
            System.out.println("");
        }
        leftNumberOffset += columnCount;
        System.out.println("");
    }
}

private static void printUnit(int leftNumber, int rightNumber) {
    System.out.print("\t" + leftNumber + " x " + rightNumber + " = " + (leftNumber*rightNumber));
}
```

잠시만, 이것도 하나 추가해 주세요.

죄송한데요... →

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81


9 18 27 36 45 54 63 72 81
8 16 24 32 40 48 56 64 72
7 14 21 28 35 42 49 56 63
6 12 18 24 30 36 42 48 54
5 10 15 20 25 30 35 40 45
4  8 12 16 20 24 28 32 36
3  6  9 12 15 18 21 24 27
2  4  6  8 10 12 14 16 18
1  2  3  4  5  6  7  8  9
```

흠, 이건 이야기가 다른데...

```
public void showLineTable(SortOrder sortOrder) {
    //
    int leftNumber = 1;     /// 1 또는 2
    int addNumber = 1;

    if(SortOrder.Descending.equals(sortOrder)) {
        leftNumber = MaxTimes;
        addNumber = -1;
    }
    System.out.println();

    while(true) {
        System.out.print(" ");
        for(int rightNumber = 1; rightNumber <= MaxTimes; rightNumber++) {
            System.out.print(getColumn(leftNumber, rightNumber));
        }
        System.out.println();

        leftNumber += addNumber;
        if(leftNumber < 2 || leftNumber > MaxTimes) {
            break;
        }
    }
    System.out.println();
}
```

Cyclometic Complexity

```
 2 x 1 =  2
 2 x 2 =  4
 2 x 3 =  6
 2 x 4 =  8
 2 x 5 = 10
 2 x 6 = 12
 2 x 7 = 14
 2 x 8 = 16
 2 x 9 = 18
------------
 3 x 1 =  3
 3 x 2 =  6
 3 x 3 =  9
 3 x 4 = 12
 3 x 5 = 15
 3 x 6 = 18
 3 x 7 = 21
 3 x 8 = 24
 3 x 9 = 27
------------
 4 x 1 =  4
```

```java
public void showTable() {
    //
    System.out.println("\n Times table.");
    System.out.println(" ------------");

    for (int leftNumber = 2; leftNumber<=9; leftNumber++) {
        for (int rightNumber = 1; rightNumber <=9; rightNumber++) {
            System.out.format(" %2d x %d = %2d%n",
                              leftNumber,
                              rightNumber,
                              (leftNumber*rightNumber));
        }

        System.out.println(" ------------");
    }
}
```

3

5

12

```java
public void showLineTable(SortOrder sortO
    //
    int leftNumber = 1;      /// 1 또는 2
    int addNumber = 1;

    if(SortOrder.Descending.equals(sortOr
        leftNumber = MaxTimes;
        addNumber = -1;
    }
    System.out.println();

    while(true) {
        System.out.print(" ");
        for(int rightNumber = 1; rightNumber <= Max
            System.out.print(getColumn(leftNumber,
        }
        System.out.println();

        leftNumber += addNumber;
        if(leftNumber < 2 || leftNumber > MaxTimes)
            break;
        }
    }
    System.out.println();
}
```

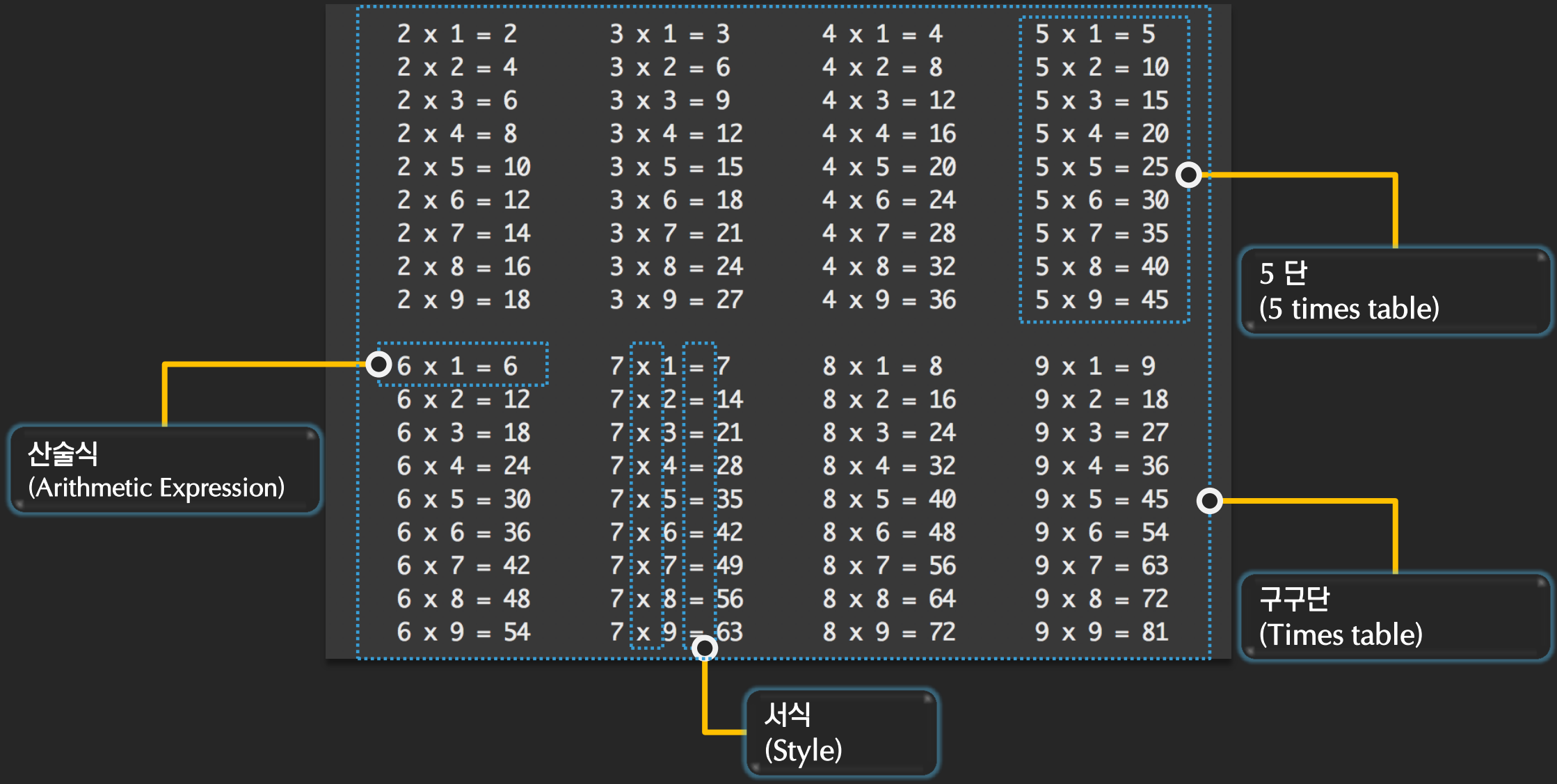
```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81

 9 18 27 36 45 54 63 72 81
 8 16 24 32 40 48 56 64 72
 7 14 21 28 35 42 49 56 63
 6 12 18 24 30 36 42 48 54
 5 10 15 20 25 30 35 40 45
 4  8 12 16 20 24 28 32 36
 3  6  9 12 15 18 21 24 27
 2  4  6  8 10 12 14 16 18
```

```
2 x 1 = 2    3 x 1 = 3    4 x 1 = 4    5 x 1 = 5
2 x 2 = 4    3 x 2 = 6    4 x 2 = 8    5 x 2 = 10
2 x 3 = 6    3 x 3 = 9    4 x 3 = 12   5 x 3 = 15
2 x 4 = 8    3 x 4 = 12   4 x 4 = 16   5 x 4 = 20
2 x 5 = 10   3 x 5 = 15   4 x 5 = 20   5 x 5 = 25
2 x 6 = 12   3 x 6 = 18   4 x 6 = 24   5 x 6 = 30
2 x 7 = 14   3 x 7 = 21   4 x 7 = 28   5 x 7 = 35
2 x 8 = 16   3 x 8 = 24   4 x 8 = 32   5 x 8 = 40
2 x 9 = 18   3 x 9 = 27   4 x 9 = 36   5 x 9 = 45

6 x 1 = 6    7 x 1 = 7    8 x 1 = 8    9 x 1 = 9
6 x 2 = 12   7 x 2 = 14   8 x 2 = 16   9 x 2 = 18
6 x 3 = 18   7 x 3 = 21   8 x 3 = 24   9 x 3 = 27
6 x 4 = 24   7 x 4 = 28   8 x 4 = 32   9 x 4 = 36
6 x 5 = 30   7 x 5 = 35   8 x 5 = 40   9 x 5 = 45
6 x 6 = 36   7 x 6 = 42   8 x 6 = 48   9 x 6 = 54
6 x 7 = 42   7 x 7 = 49   8 x 7 = 56   9 x 7 = 63
6 x 8 = 48   7 x 8 = 56   8 x 8 = 64   9 x 8 = 72
6 x 9 = 54   7 x 9 = 63   8 x 9 = 72   9 x 9 = 81
```

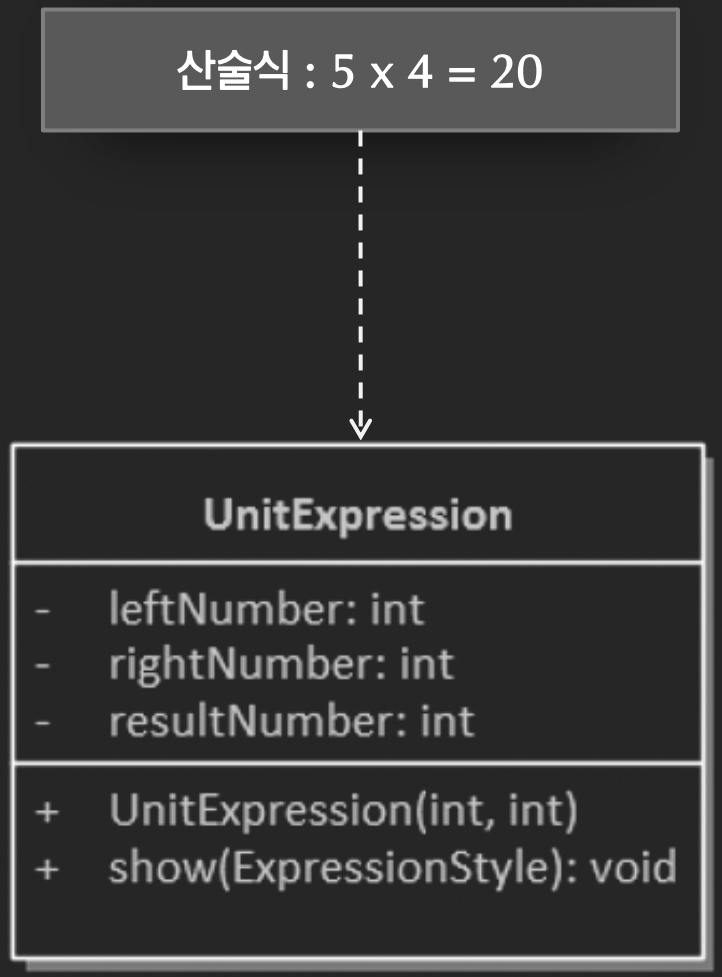
구구단(Times table)은 어디에...

```
private static void printTimesTable(int columnCount) {
    //
    int loopCount = (8 / columnCount) + 1;

    int leftNumberOffset = 2;
    for(int i=0; i<loopCount; i++) {
        for(int rightNumber=1; rightNumber<=9; rightNumber++) {
            for(int count = 0; count < columnCount; count++) {
                int leftNumber = leftNumberOffset + count;
                if (leftNumber > 9) {
                    break;
                }
                printUnit(leftNumber, rightNumber);
            }
            System.out.println("");
        }
        leftNumberOffset += columnCount;
        System.out.println("");
    }
}

private static void printUnit(int leftNumber, int rightNumber) {
    System.out.print("\t" + leftNumber + " x " + rightNumber + " = " + (leftNumber*rightNumber));
}
```

구구단(Times table)을 구성하는 개념들

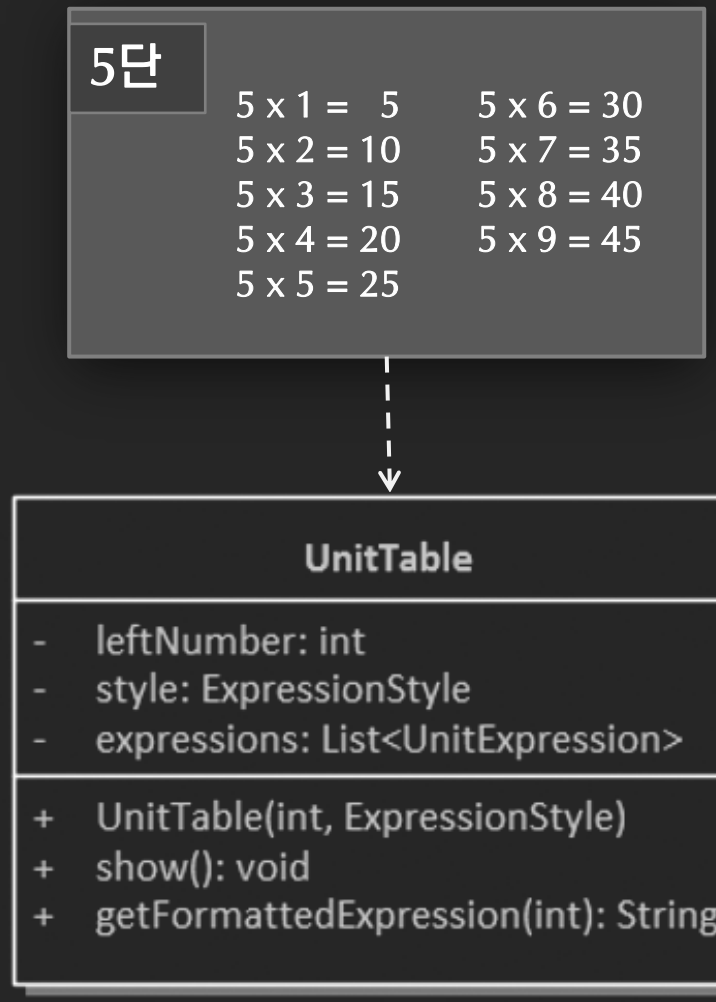


도메인 모델링 - 개념 1 : 산술식(Arithmetic Expression)




```java
public class UnitExpression {
    //
    private int leftNumber;
    private int rightNumber;
    private int resultNumber;

    public UnitExpression(int leftNumber, int rightNumber) {
        //
        this.leftNumber = leftNumber;
        this.rightNumber = rightNumber;
        this.resultNumber = leftNumber * rightNumber;
    }

    public String toString() {

    public void show(ExpressionStyle style) {
        //
        System.out.format(style.formatStrln(),leftNumber, rightN
    }

    public static UnitExpression getSample() {
```

도메인 모델링 - 개념 2 : 각 단(Each Multiplication Table)




```java
public class UnitTable {
    //
    private int leftNumber;
    private ExpressionStyle style;
    private List<UnitExpression> expressions;

    public UnitTable(int leftNumber) {
        //
        this(leftNumber, ExpressionStyle.InEnglish);
    }

    public UnitTable(int leftNumber, ExpressionStyle style) {

    public String toString() {

    private void init() {
```

도메인 모델링 - 개념 3 : 구구단(Times table)


```java
public class TimesTable {
    //
    private boolean basicTable;
    private List<UnitTable> unitTables;

    public TimesTable() {
        //
        this.basicTable = false;
        this.unitTables = new ArrayList<>();
    }

    public void clear() {
        this.unitTables.clear();
    }

    public boolean isBasicTable() {
        return basicTable;
    }

    public List<UnitTable> getUnitTables() {
        //
        if (basicTable) {
```

도메인 모델링 - 개념 4 : 서식(Style for the expression)

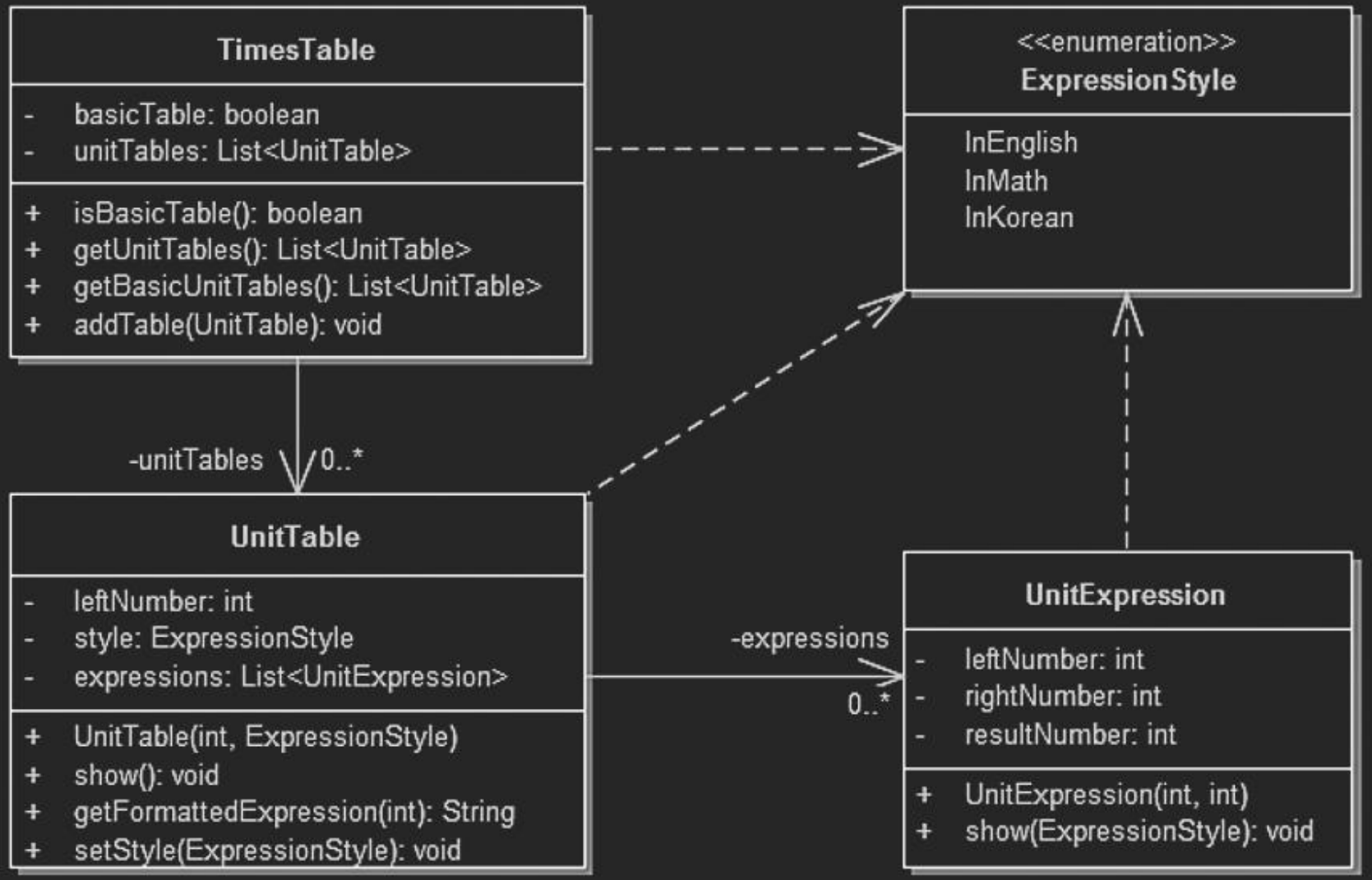
5 times 4 is 20

5 x 4 = 20

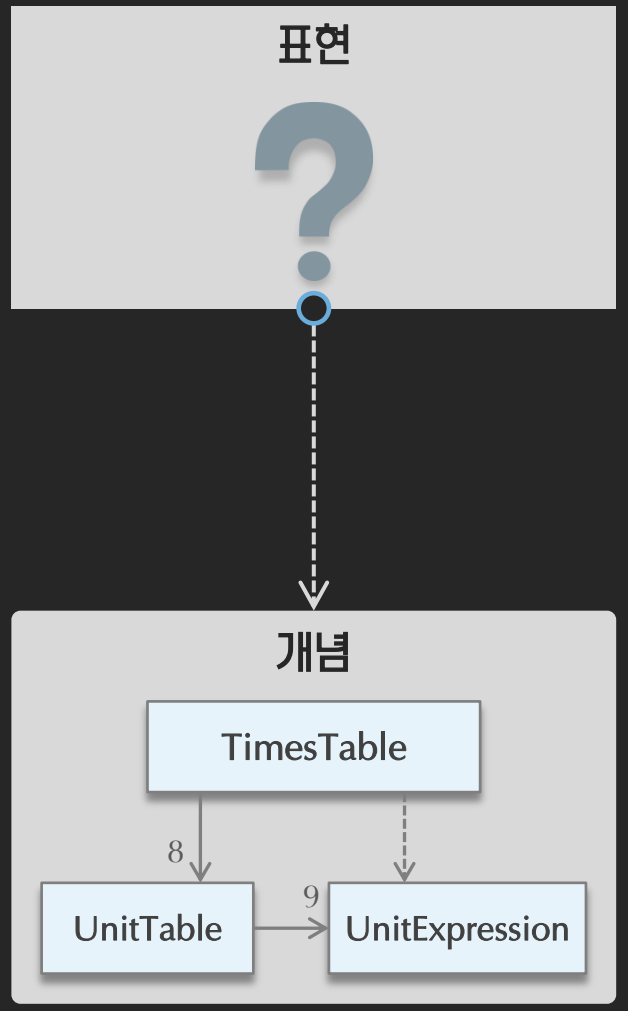
5 곱하기 4 는 20

```
public enum ExpressionStyle {
    //
    InEnglish(" %d times %d is %2d"),
    InJavaCode(" %d * %d = %2d"),
    InMath(" %d x %d = %2d"),
    InKorean(" %d 곱하기 %d 는 %2d");

    private String formatStr;

    private ExpressionStyle(String style) {
        this.formatStr = style;
    }
```

대상(TimesTable)의 본질



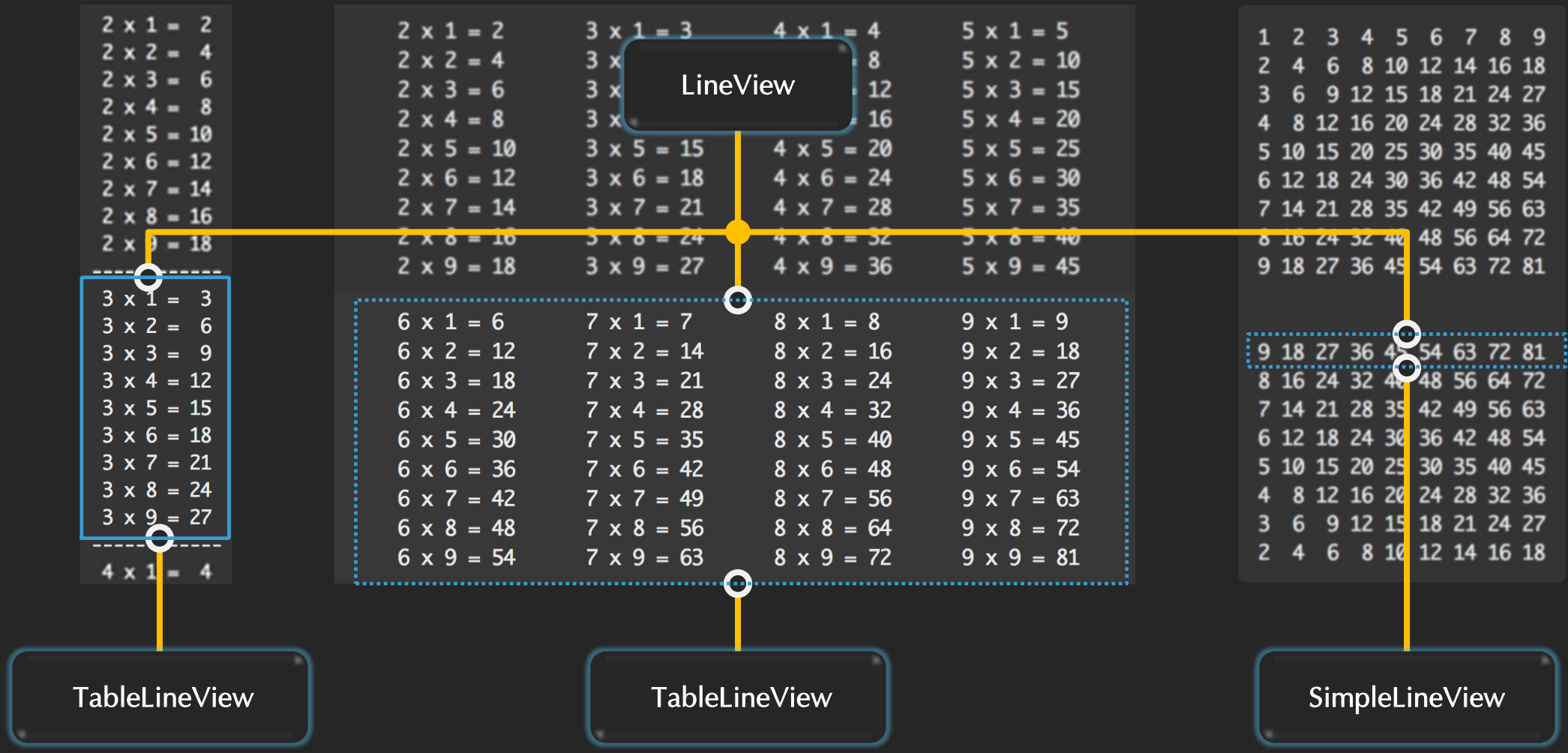
본질은 원래 추상적...



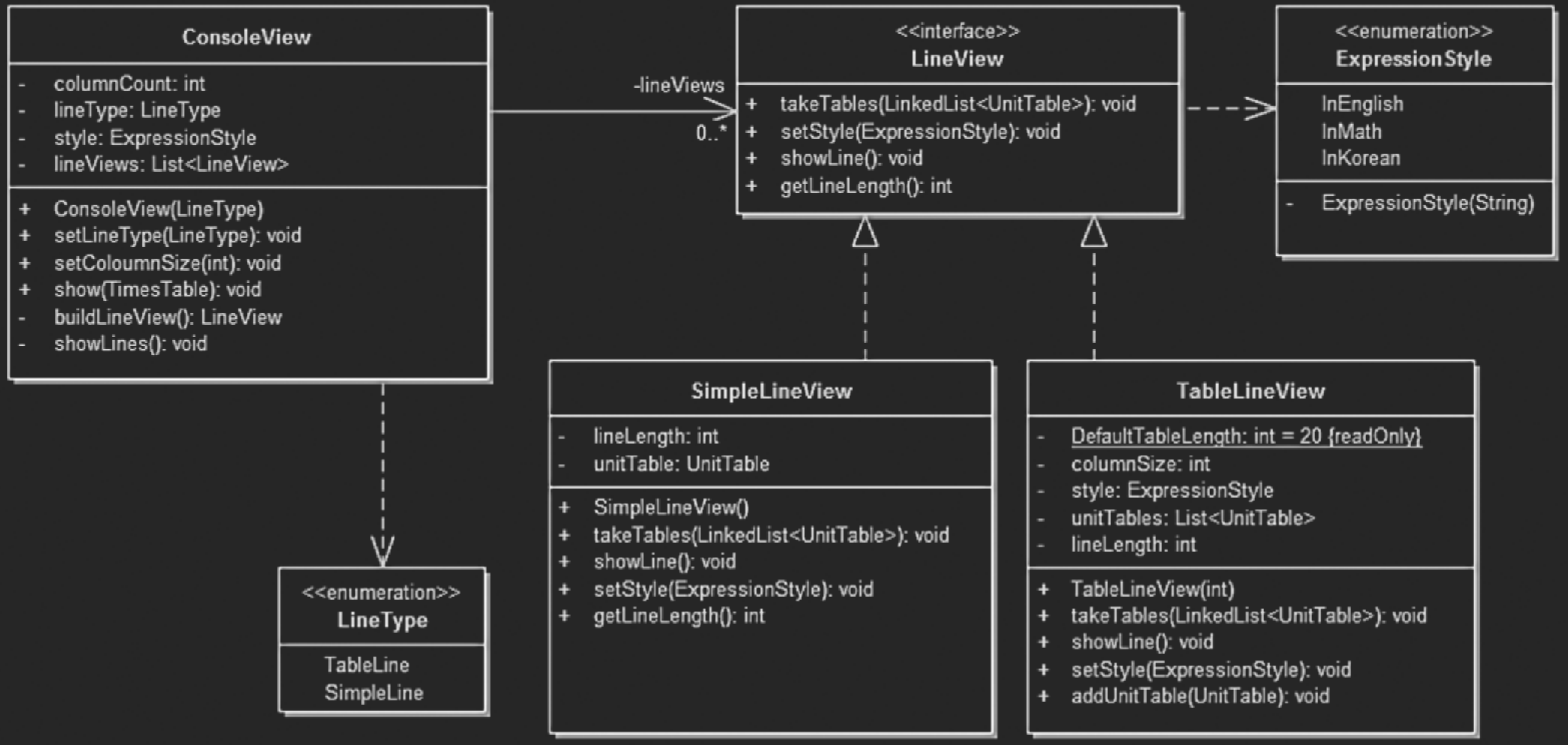
← 표현에도 개념이 있다...

← 도메인 모델

추상화(Abstraction)로 변화하는 대상에 질서를...

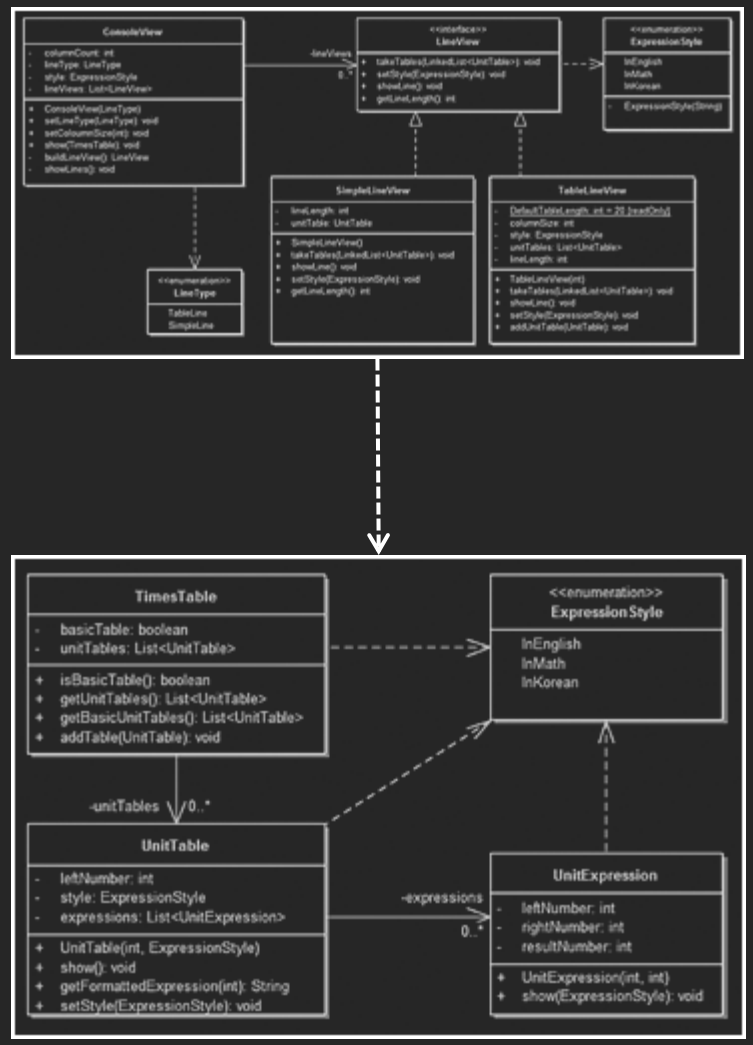


표현(Representation)을 위한 개념들



SOLID – SRP, OCP, LSP, DIP, except for ISP

개념은 안정적이고, 표현은 유동적이다.



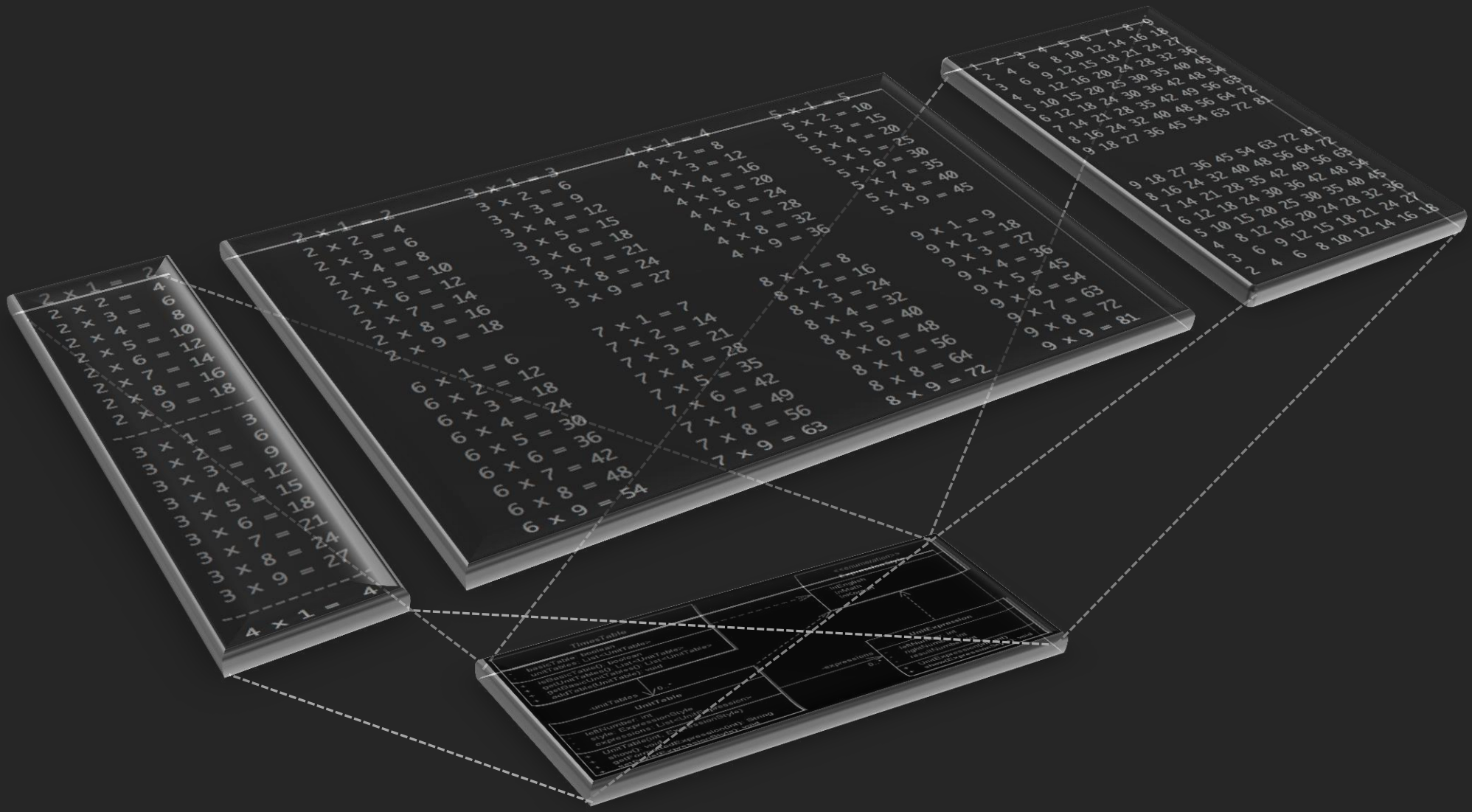
1열 4열 정방형

← 표현 (representation)
← Dynamic

← 도메인 (로직과 개념)
← Stable

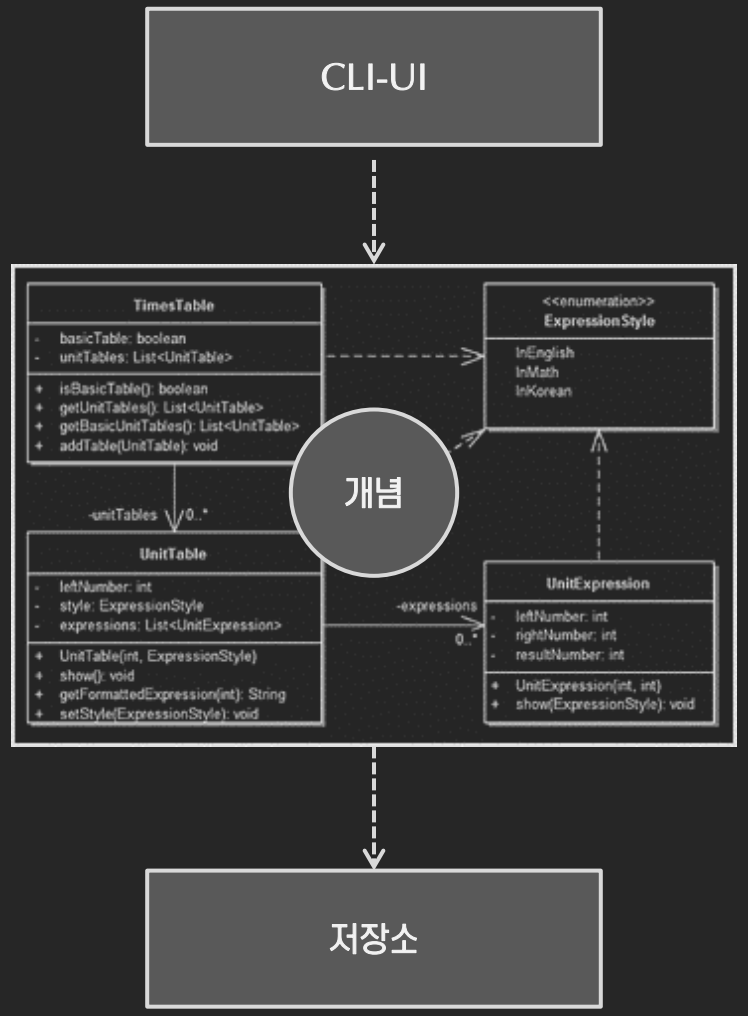
```java
public TimesTableDemo() {
    //
    this.timesTable = new TimesTable();
    this.timesTable.initialize();
}

public void showTableLineDemo() {
    //
    ConsoleView consoleView = new ConsoleView(LineType.TableLine);
    consoleView.setColoumnSize(4);
    consoleView.setStyle(ExpressionStyle.InMath);
    consoleView.show(this.timesTable);
    consoleView.showLineSeparator();
}

public void showSimpleLineDemo() {
    //
    ConsoleView consoleView = new ConsoleView(LineType.SimpleLine);
    consoleView.setColoumnSize(4);      // ignored
    consoleView.show(this.timesTable);
    consoleView.showLineSeparator();
}


public static void main(String[] args) {
    //
    TimesTableDemo demo = new TimesTableDemo();
    demo.showTableLineDemo();
    demo.showSimpleLineDemo();
}
```

본질은 다양한 형상으로 그 모습을 드러낸다.

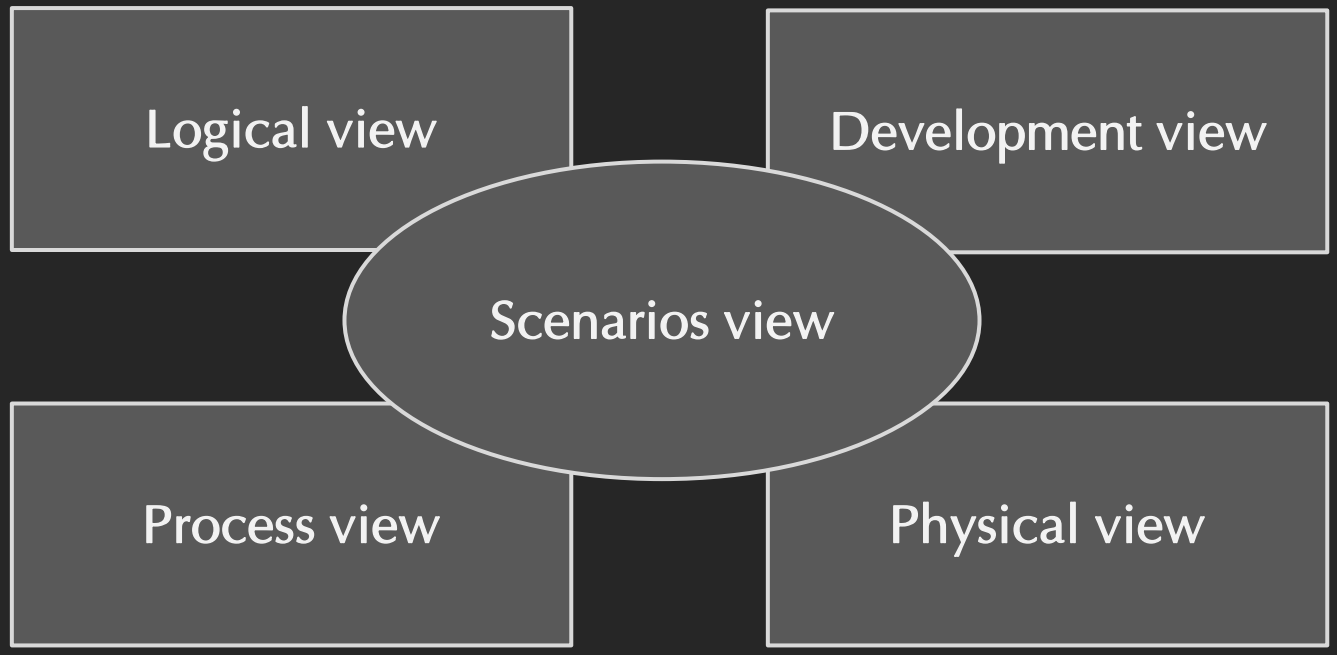


2. 확장 - 구조와 뷰

시스템으로써 틀을 갖추려면...

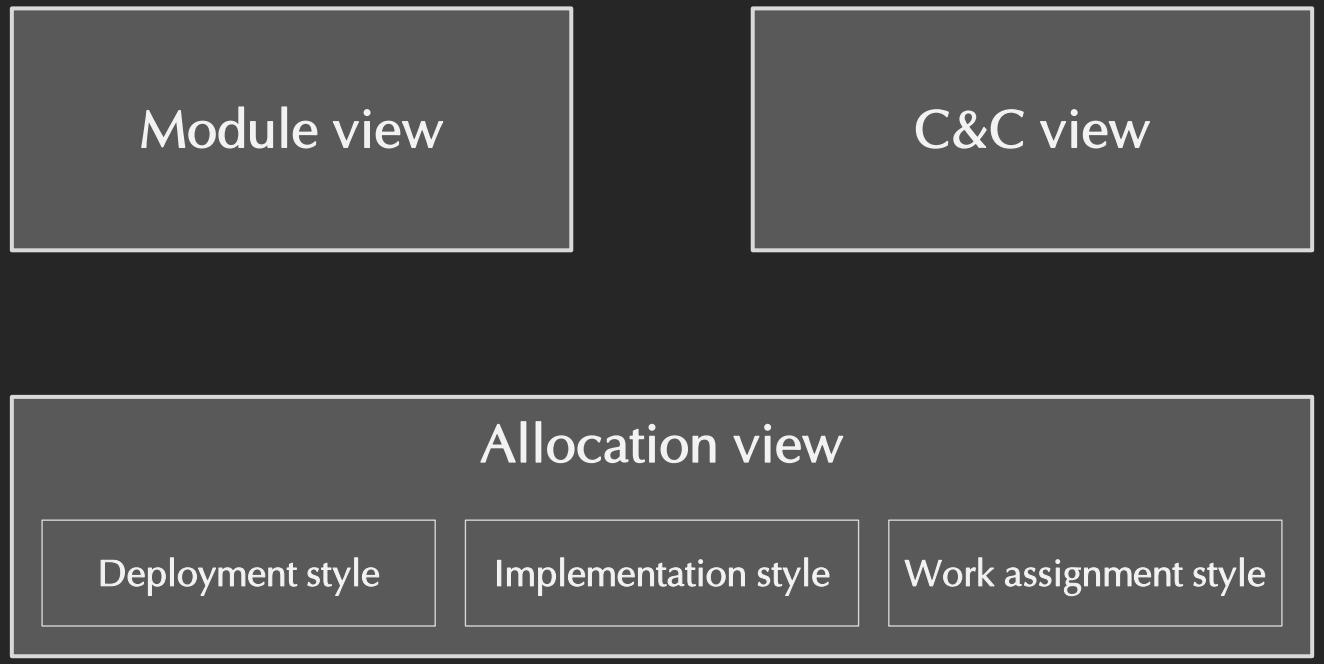


4 + 1 Views (as an architecture framework)



1995, Philippe Kruchten

3 Views (from CMU SEI)



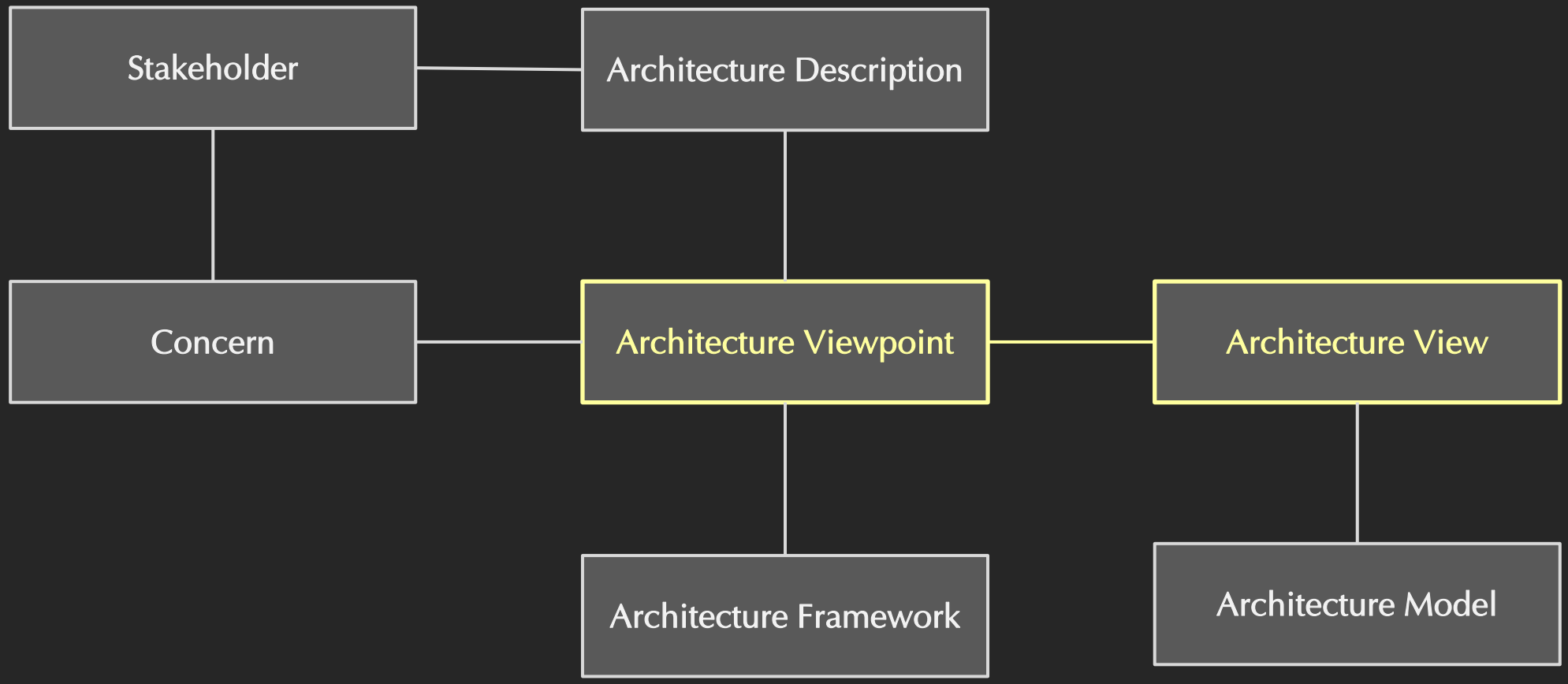
3 Perspectives (from ACDM by Lattanze)

Static perspective

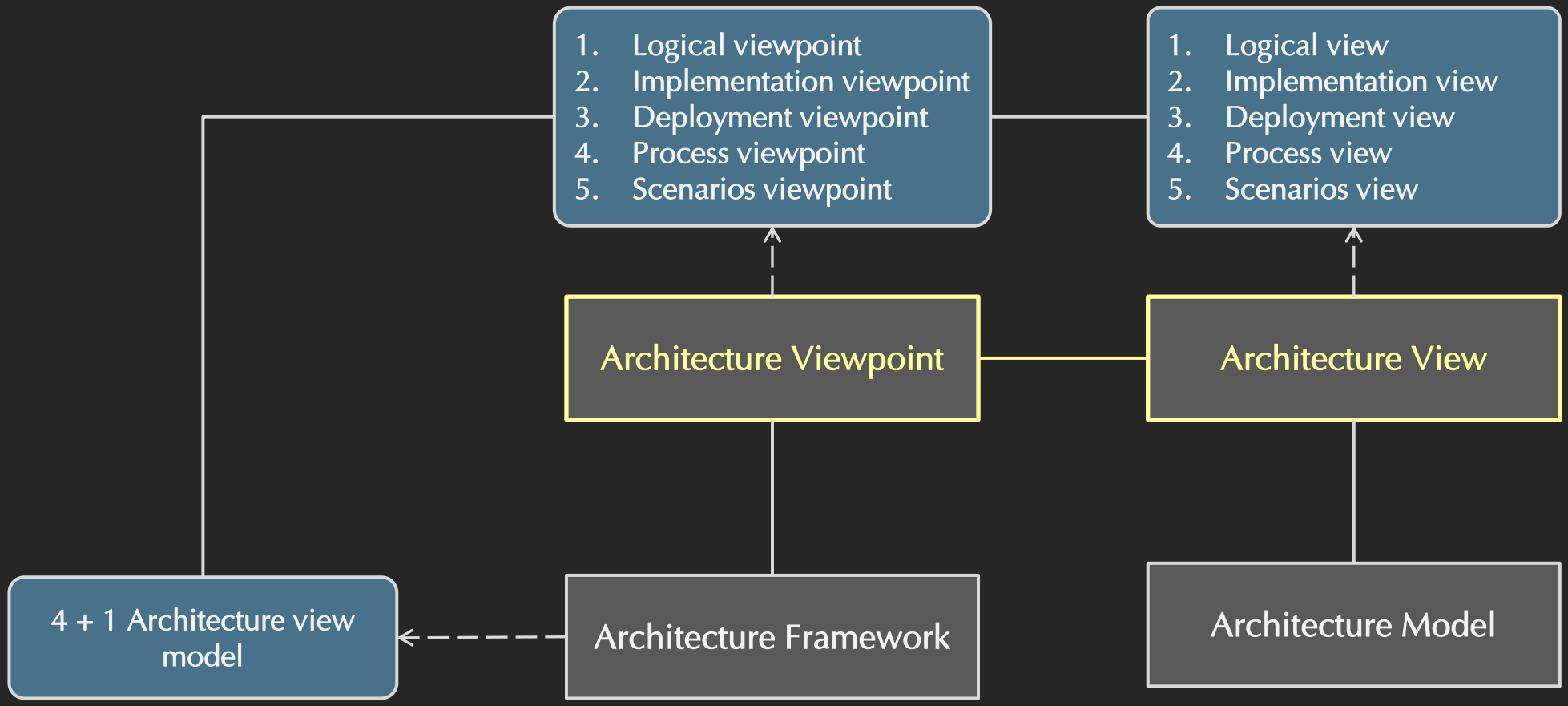
Dynamic perspective

Physical perspective

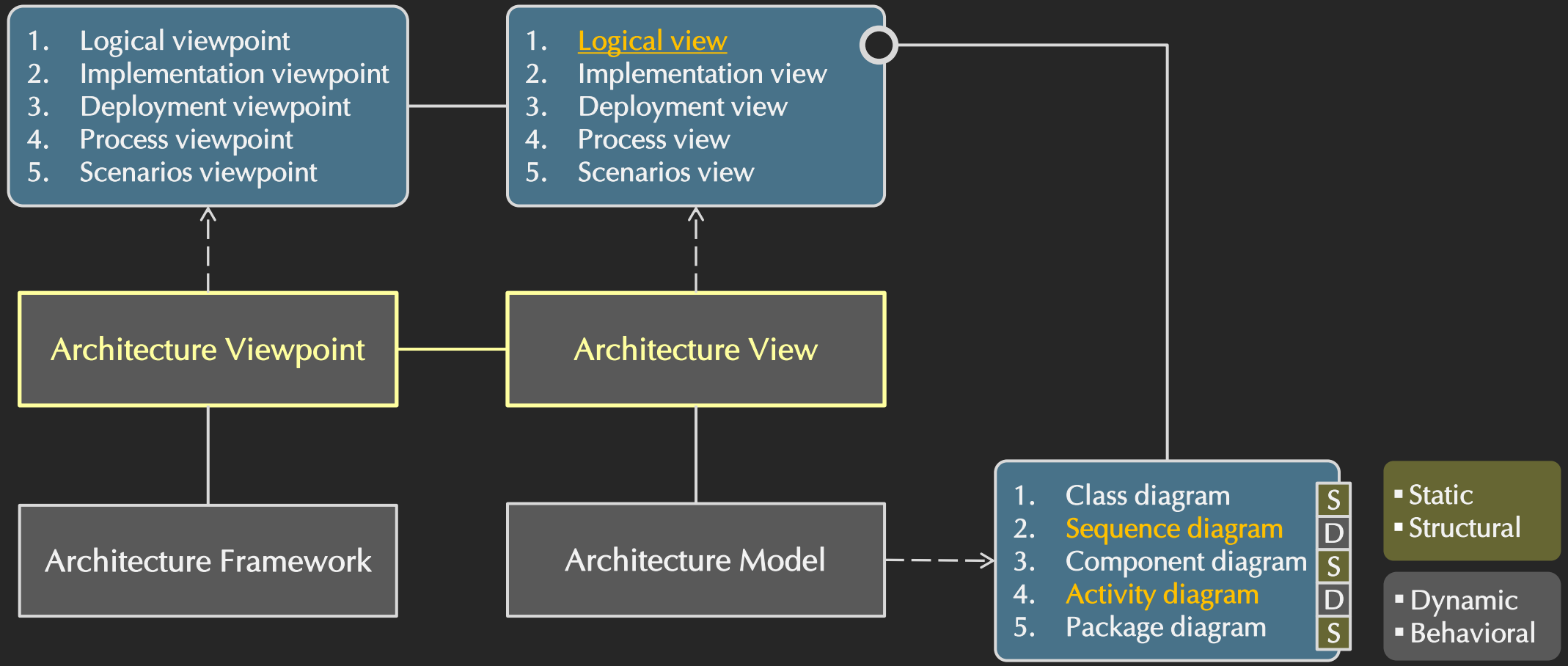
ISO/IEC/IEEE 42010:2011



Architecture framework – Viewpoint : View



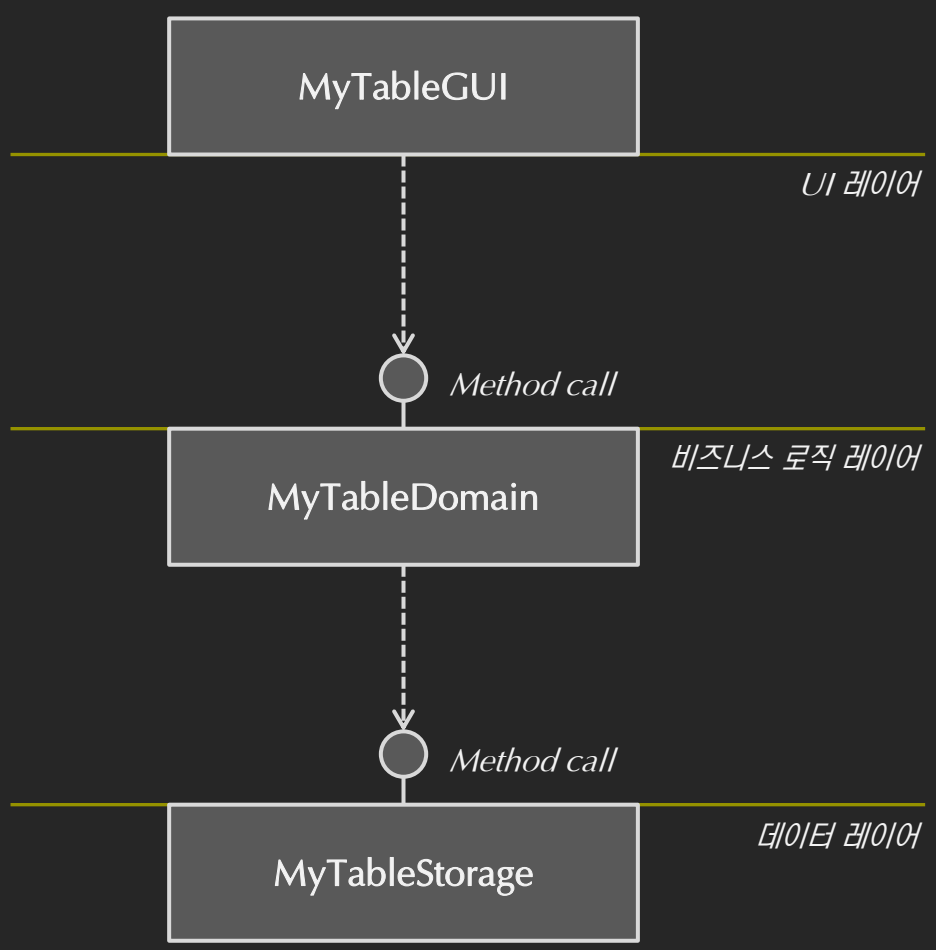
View : Model(static, dynamic)



SDLC와 Views


Requirement
Analysis
Design
Implements
Deployment
Runtime
Functional perspective
기능 요건
도메인 모델/UI 모델/데이터 모델
분석 수준 모델
설계 수준 모델
비즈니스 컴포넌트 프로그램 또는 프로그램
배포 비즈니스 컴포넌트, 라이브러리
프로그램 실행
Non-functional perspective
비기능 요건
아키텍처 모델
패턴/스타일
반복적인 분해 모델
아키텍처 요소 프로그램
배포 아키텍처 요소
프로그램 실행
개발 모델
배포 모델
런타임 모델
아키텍처 프레임워크
4 + 1 뷰
Use Case view (or Scenarios view)
Logical view
Development view (or Implementation view)
Physical view (or Deployment view)
Process view (or Runtime view)
CMU SEI
N/A
Module view
Allocation view
Allocation view
C&C view
ACDM
N/A
Static perspective
Physical perspective
Dynamic perspective

독립형(stand-alone) 시스템

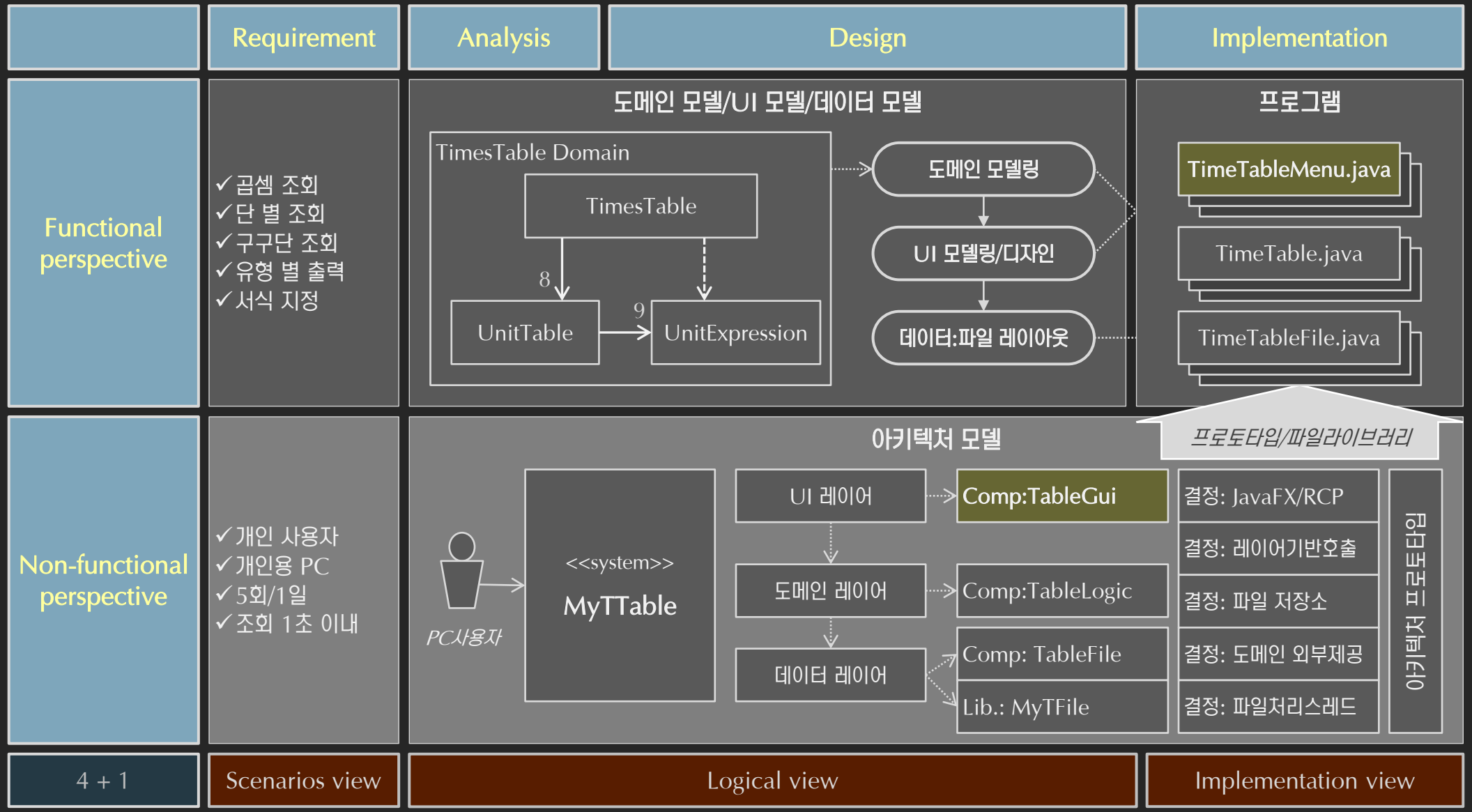


Logical View (모듈/설계 컴포넌트)

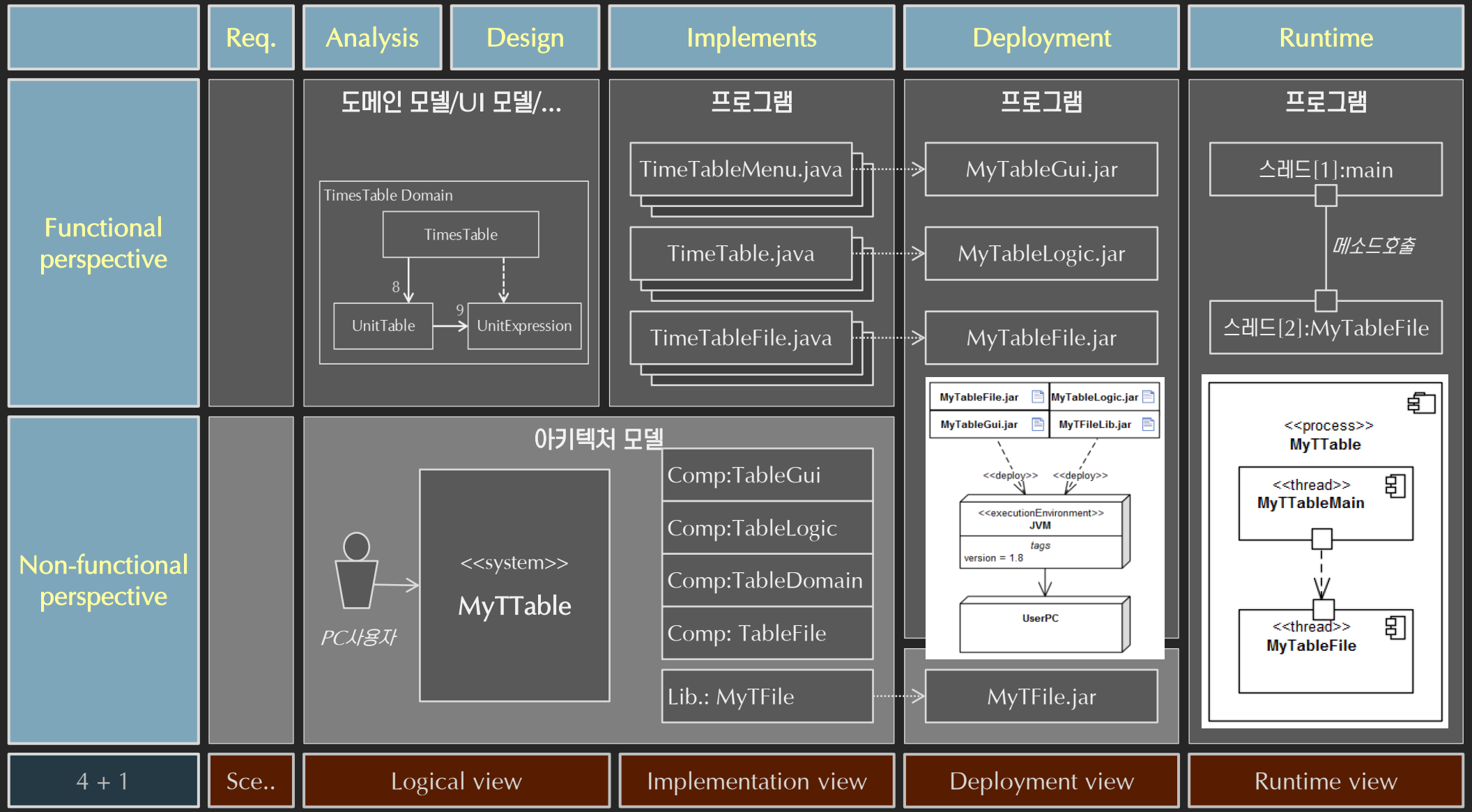
Deployment View

Process(Runtime)View

독립형(stand-alone) 시스템



독립형(stand-alone) 시스템



아키텍처 중심 또는 도메인 중심 ?

	소규모 도메인	중규모 도메인	대규모 도메인
Functional perspective	상대적으로 작은 규모 도메인	도메인	상대적으로 큰 규모 도메인
Non-functional perspective	상대적으로 큰 아키텍처	아키텍처	상대적으로 작은 아키텍처

Architecture centric

- ✓ embedded 시스템
- ✓ 플랫폼, 프레임워크, 프로덕트 라인
- ✓ 제어 시스템
- ✓ 독립형(stand-alone) 시스템

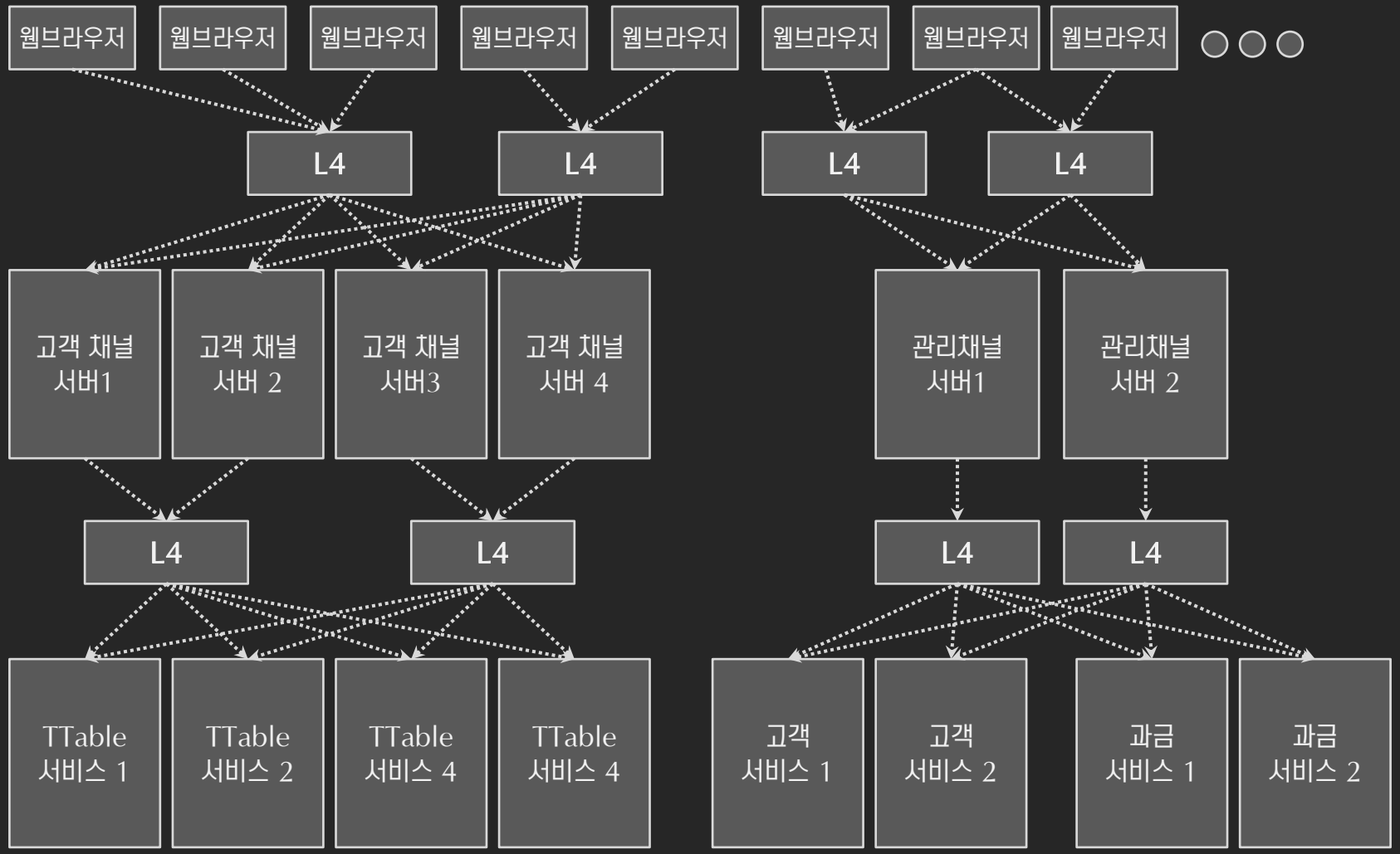
Domain centric

- ✓ 기업용 시스템
- ✓ 인터넷 서비스 시스템
- ✓ 솔루션 시스템
- ✓ 오피스용 시스템

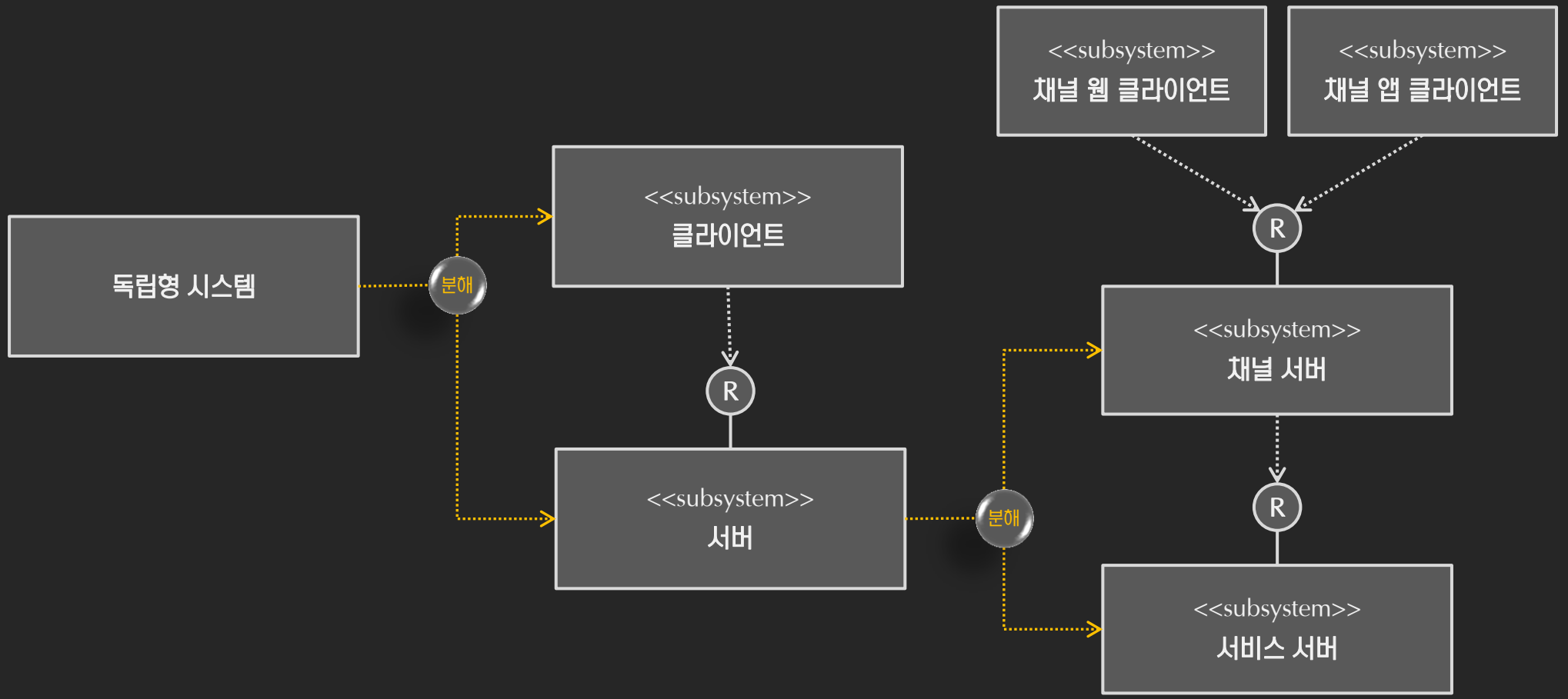
독립 시스템에서 서비스 시스템으로 확장

- Scalability →1000 만명 회원
- Availability → 24 x 369, 서비스 정지 없음
- Performance →모든 서비스 요청은 1초 이내
- Easy of use →누구나 서비스를 쉽게 사용할 수 있도록
- Extendability →스마트 폰, 와치 등으로 디바이스 확장
- And more...

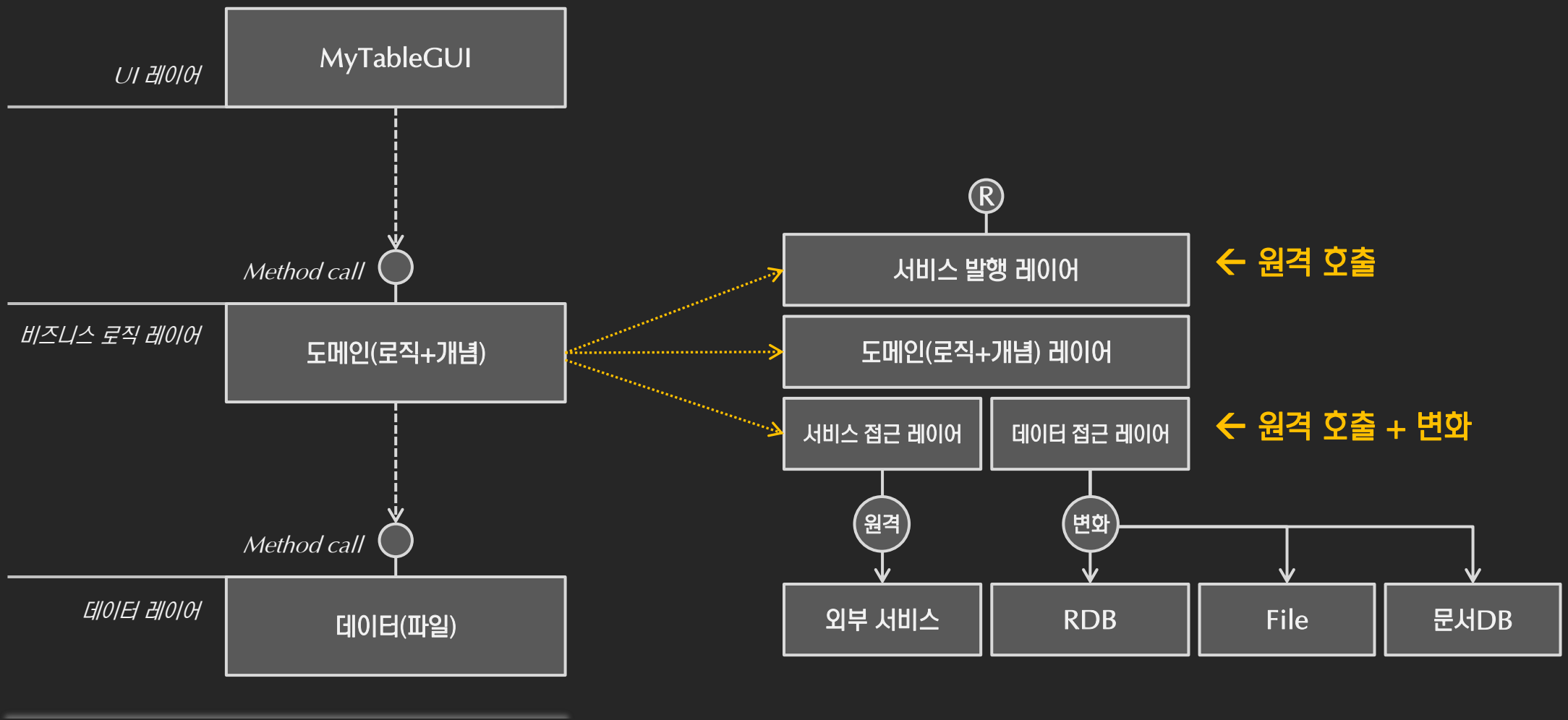
예상 Deployment view



Logical view: 서브-시스템 식별

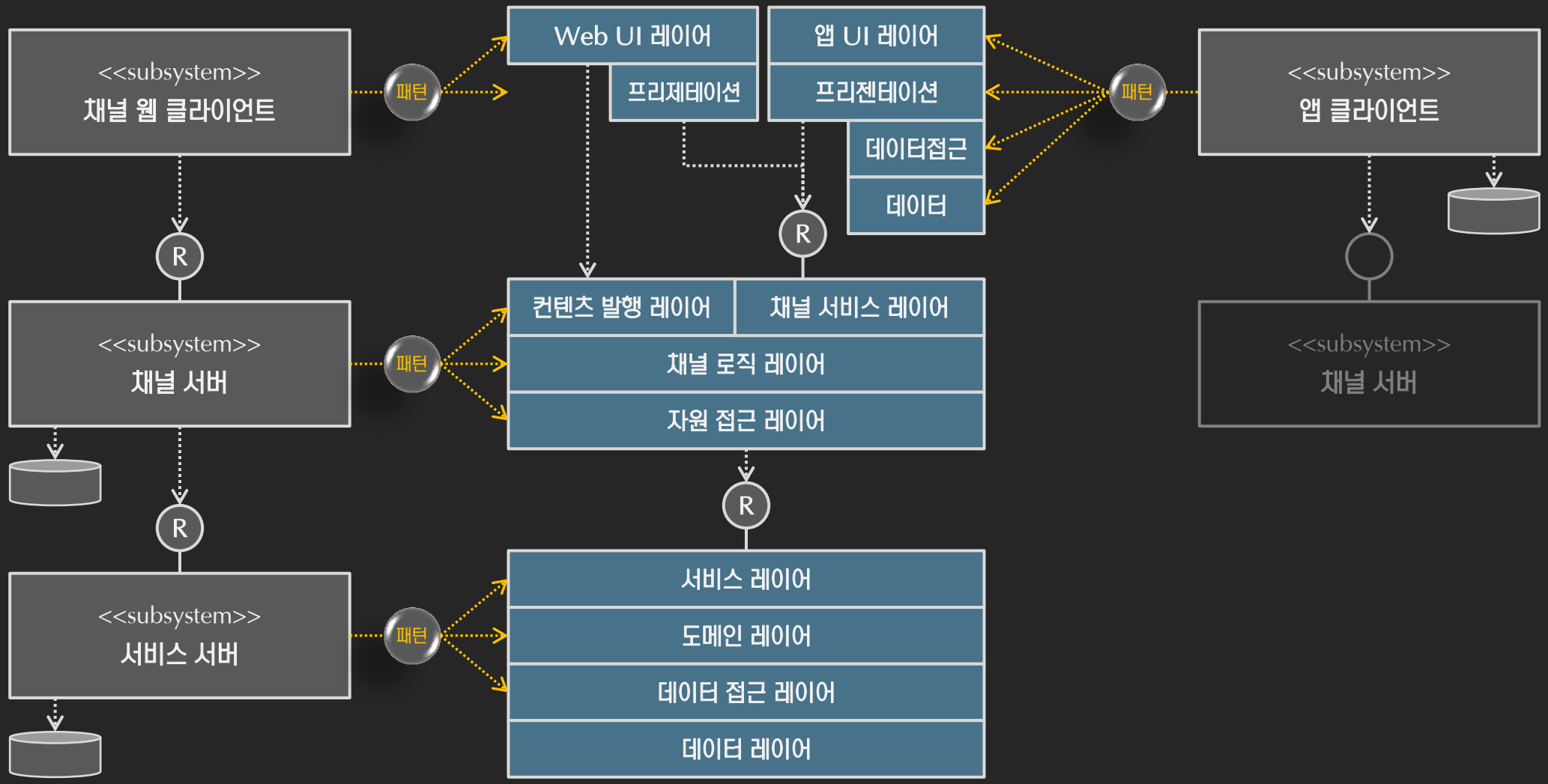


Logical view: 레이어 확장

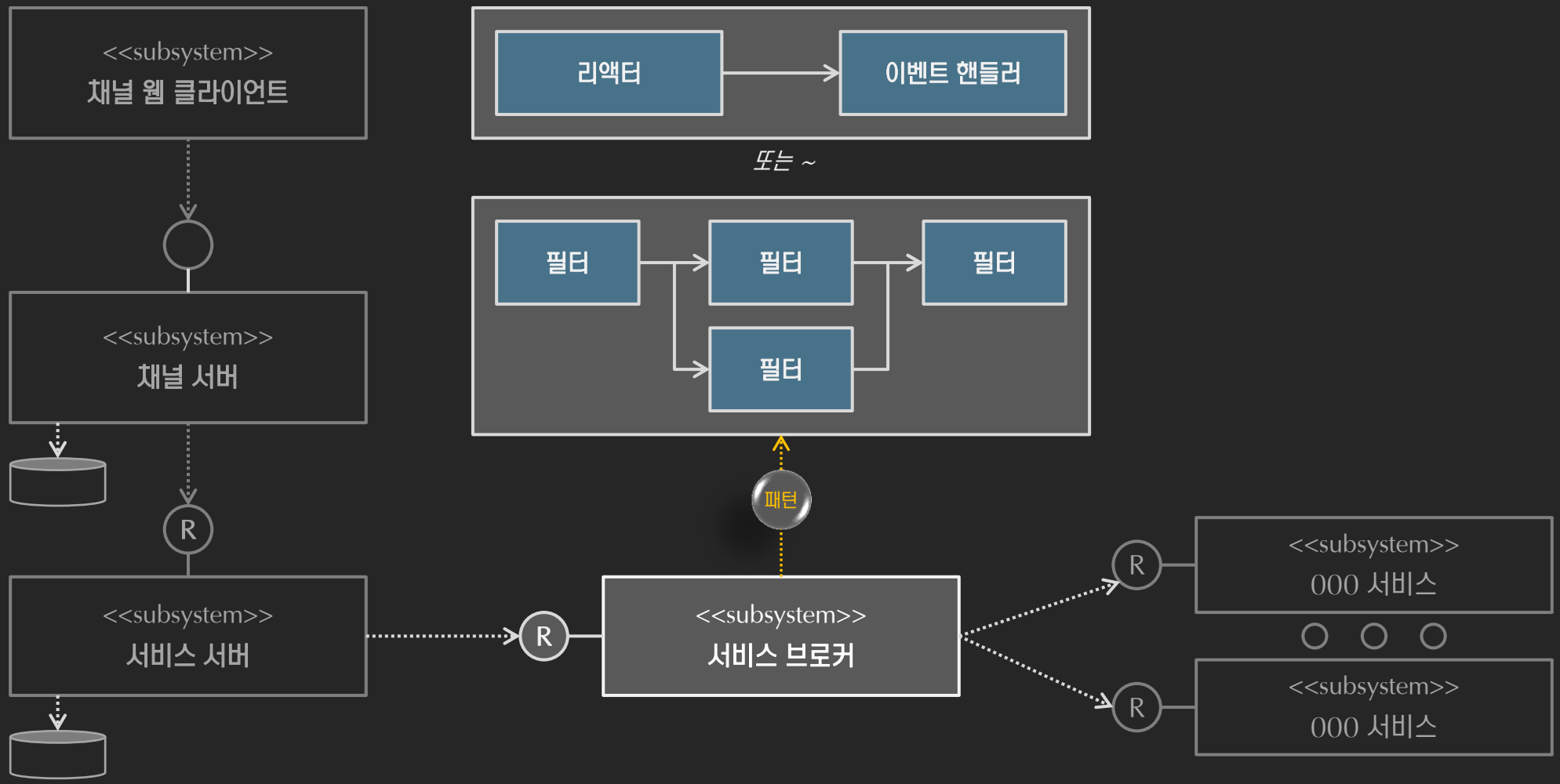


Logical View (모듈/설계 컴포넌트)

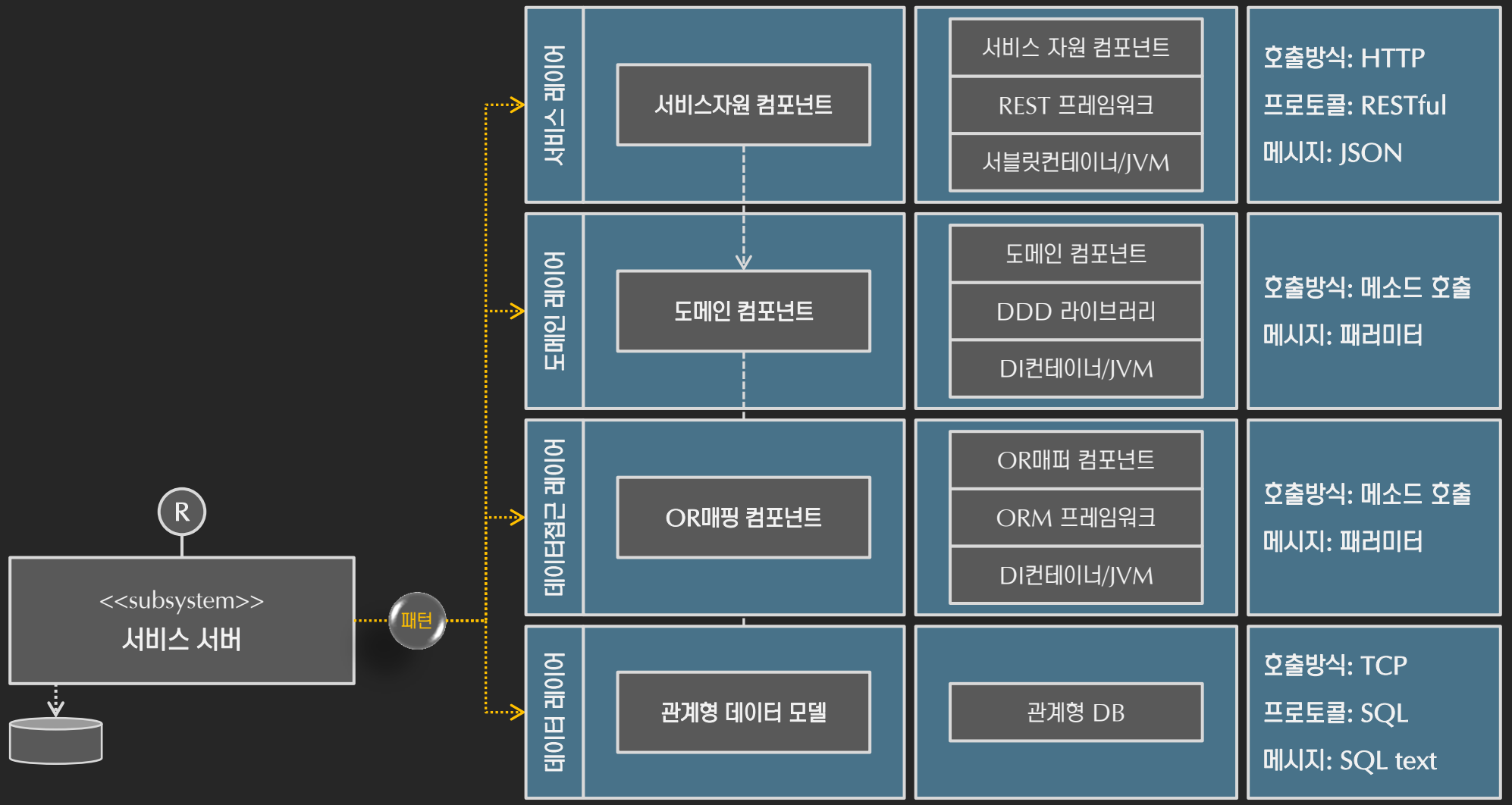
Logical view: 레이어 설계



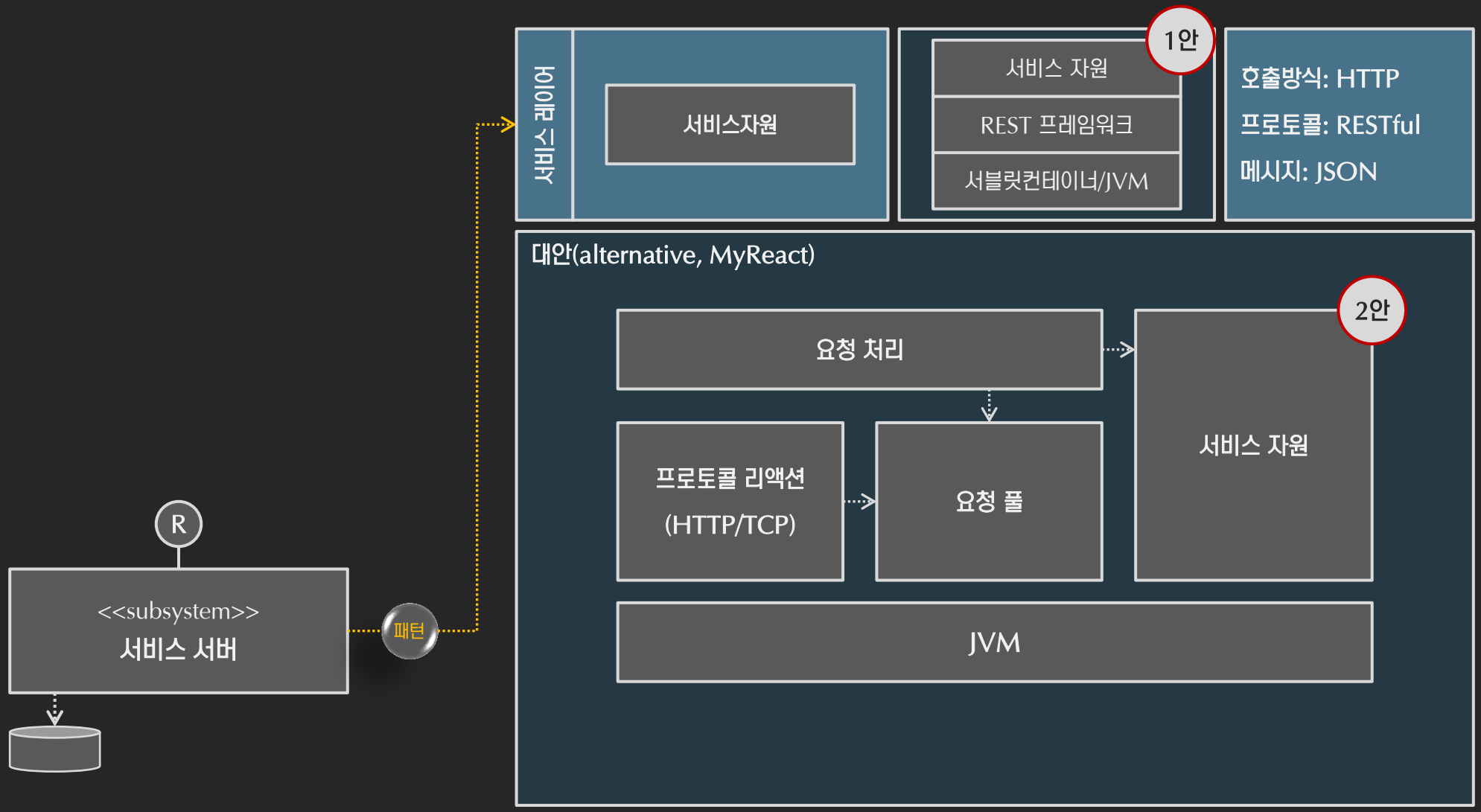
Logical view: 다른 아키텍처 패턴의 예



Logical view: 레이어 구성 모듈



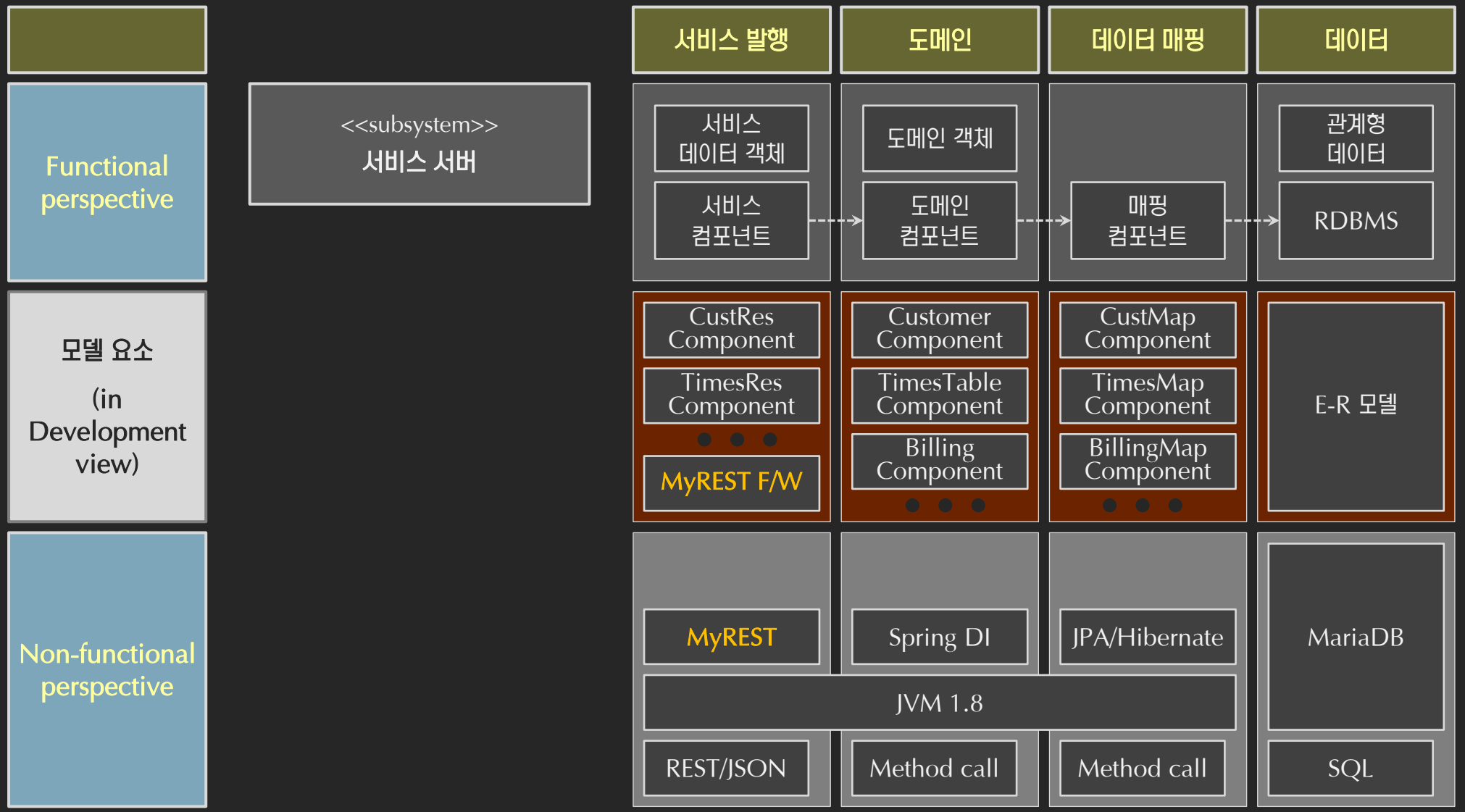
Logical view: 레이어 구성 – 설계 대안



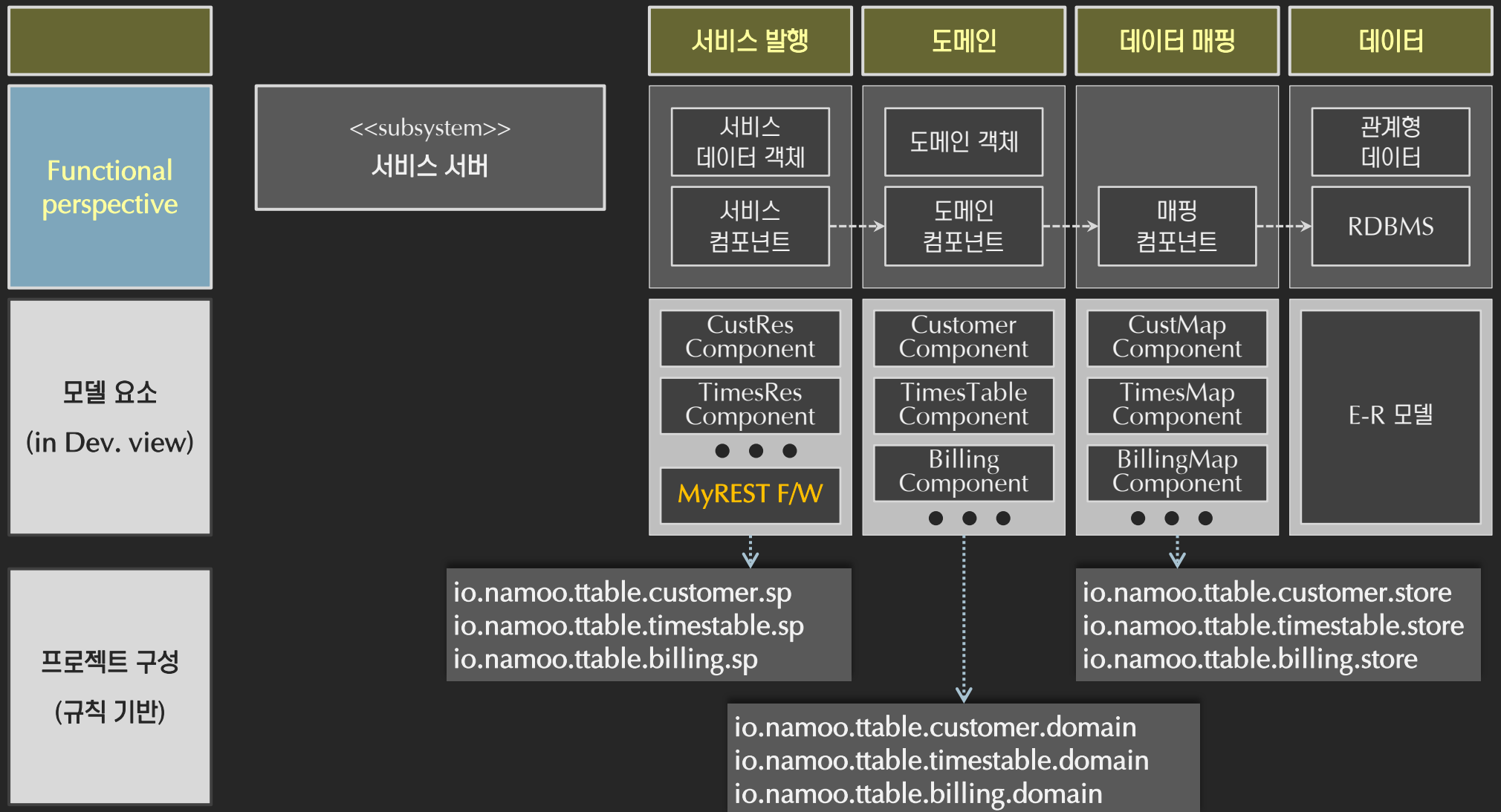
Logical view: 레이어 구성 요소 - 상세

Logical view: 레이어 구성 요소 – 서비스 서버

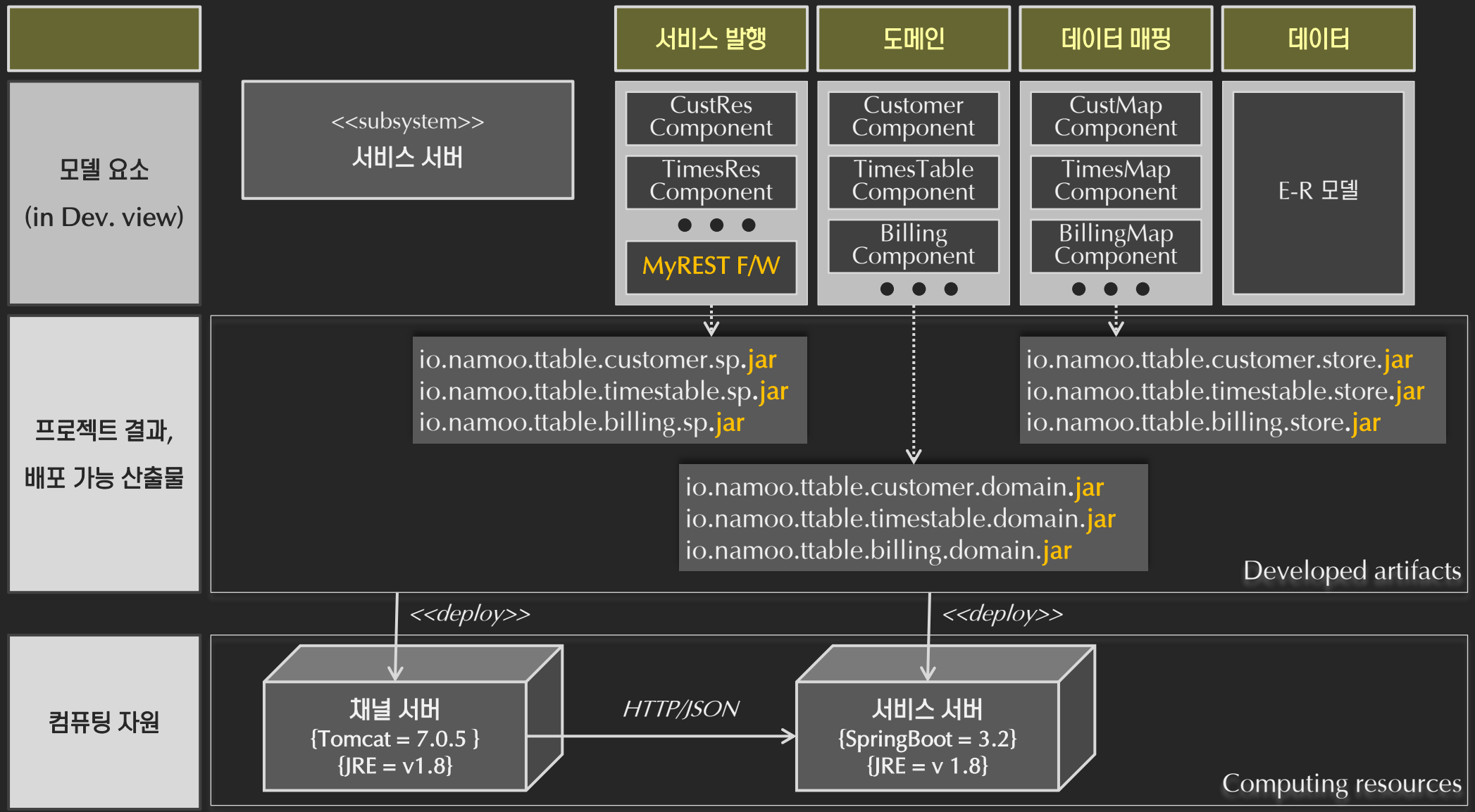
Development view



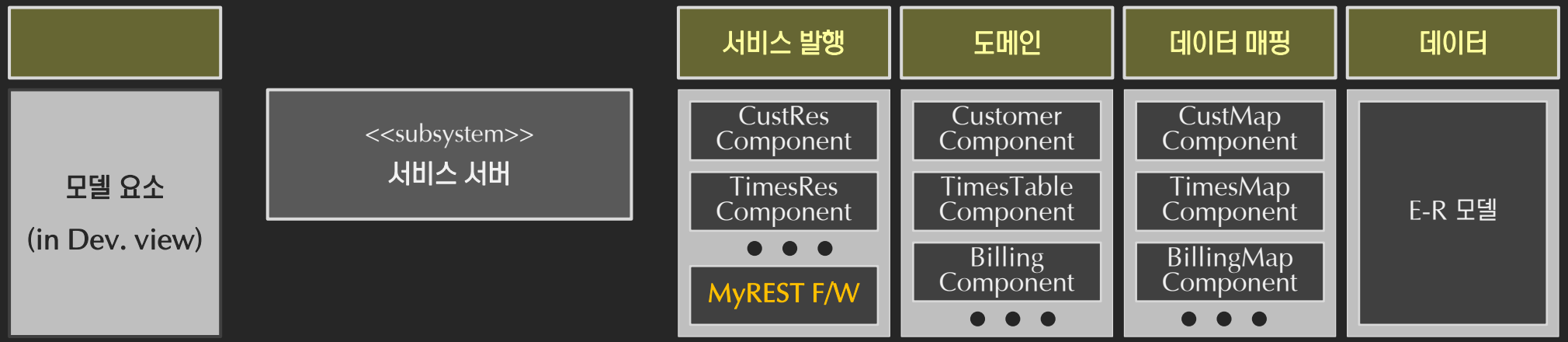
Development view



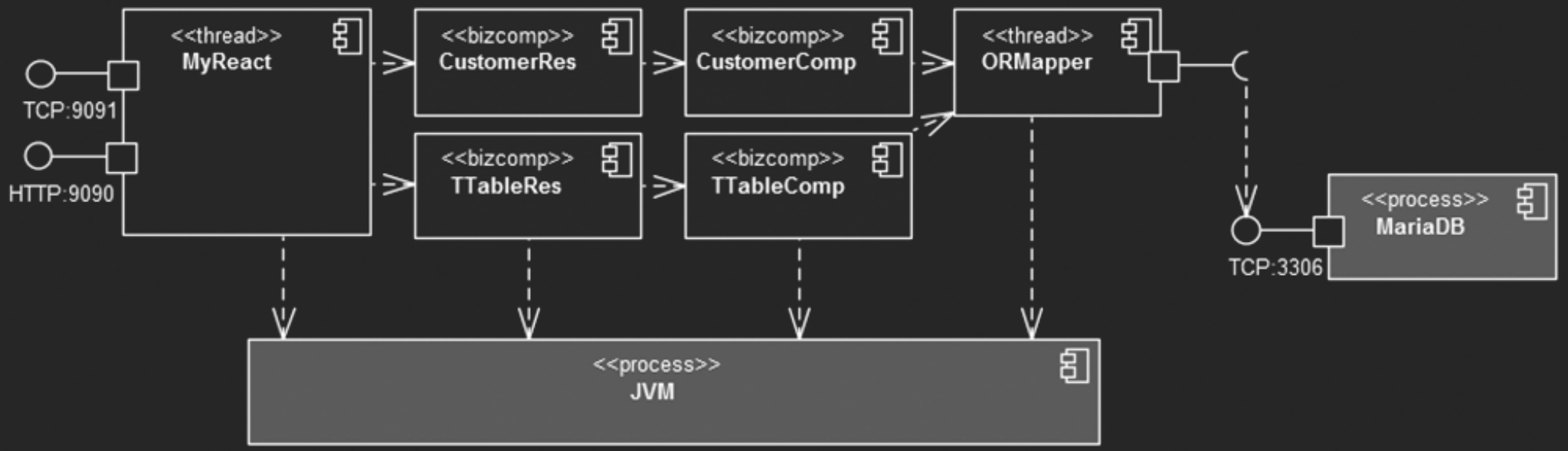
Deployment view – static only



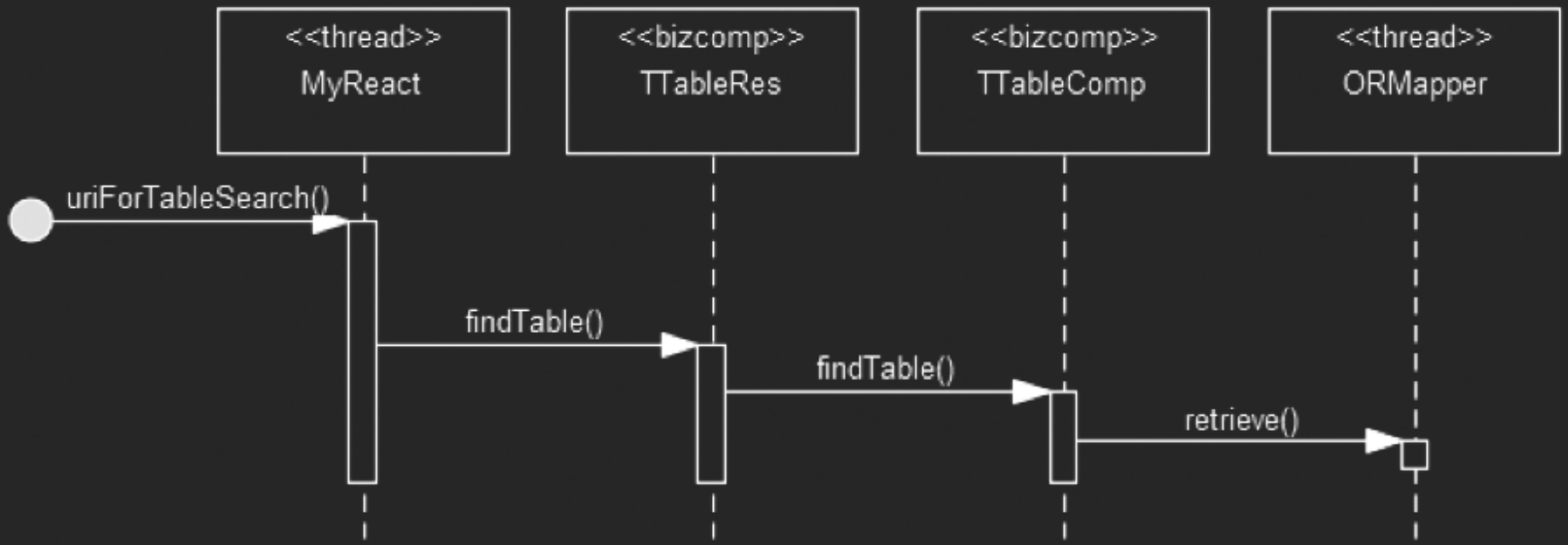
Process(Runtime) view – Level 1



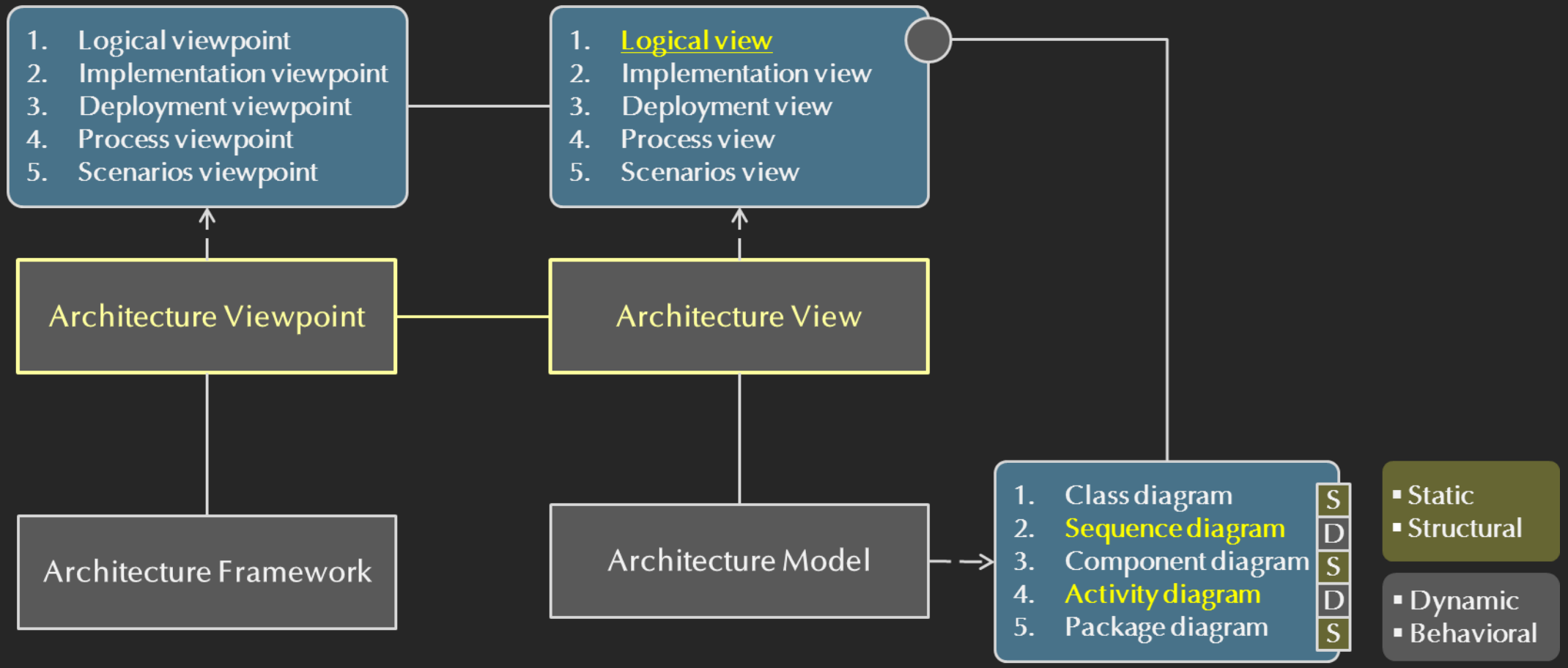
Process(Runtime) view – level 2 - static



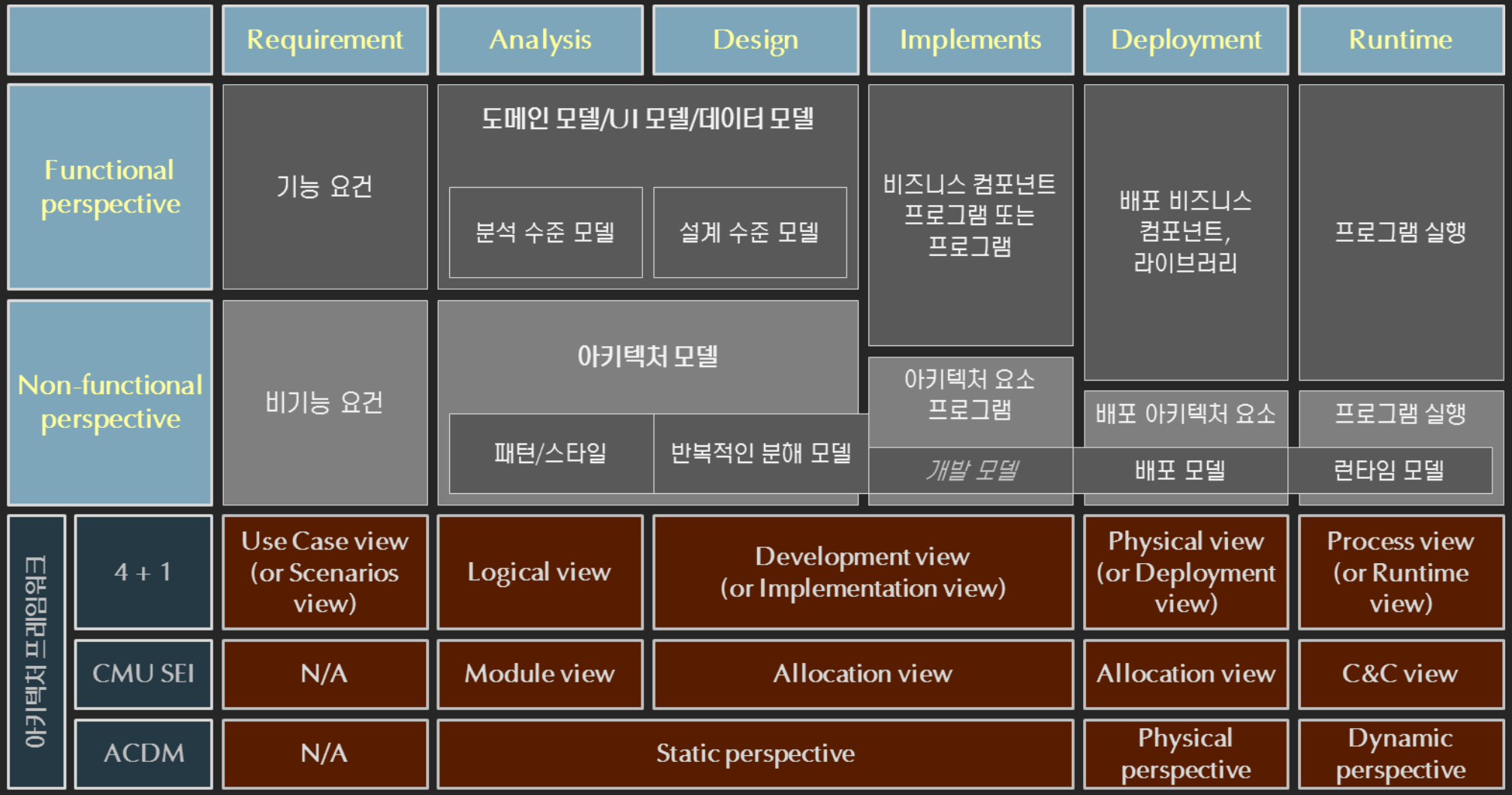
Process(Runtime) view – level 2 - dynamic



요약 1


1. Logical viewpoint
2. Implementation viewpoint
3. Deployment viewpoint
4. Process viewpoint
5. Scenarios viewpoint
1. Logical view
2. Implementation view
3. Deployment view
4. Process view
5. Scenarios view
Architecture Viewpoint
Architecture View
Architecture Framework
Architecture Model
1. Class diagram
2. Sequence diagram
3. Component diagram
4. Activity diagram
5. Package diagram
S
D
S
D
S
▪ Static
▪ Structural
▪ Dynamic
▪ Behavioral

요약 2

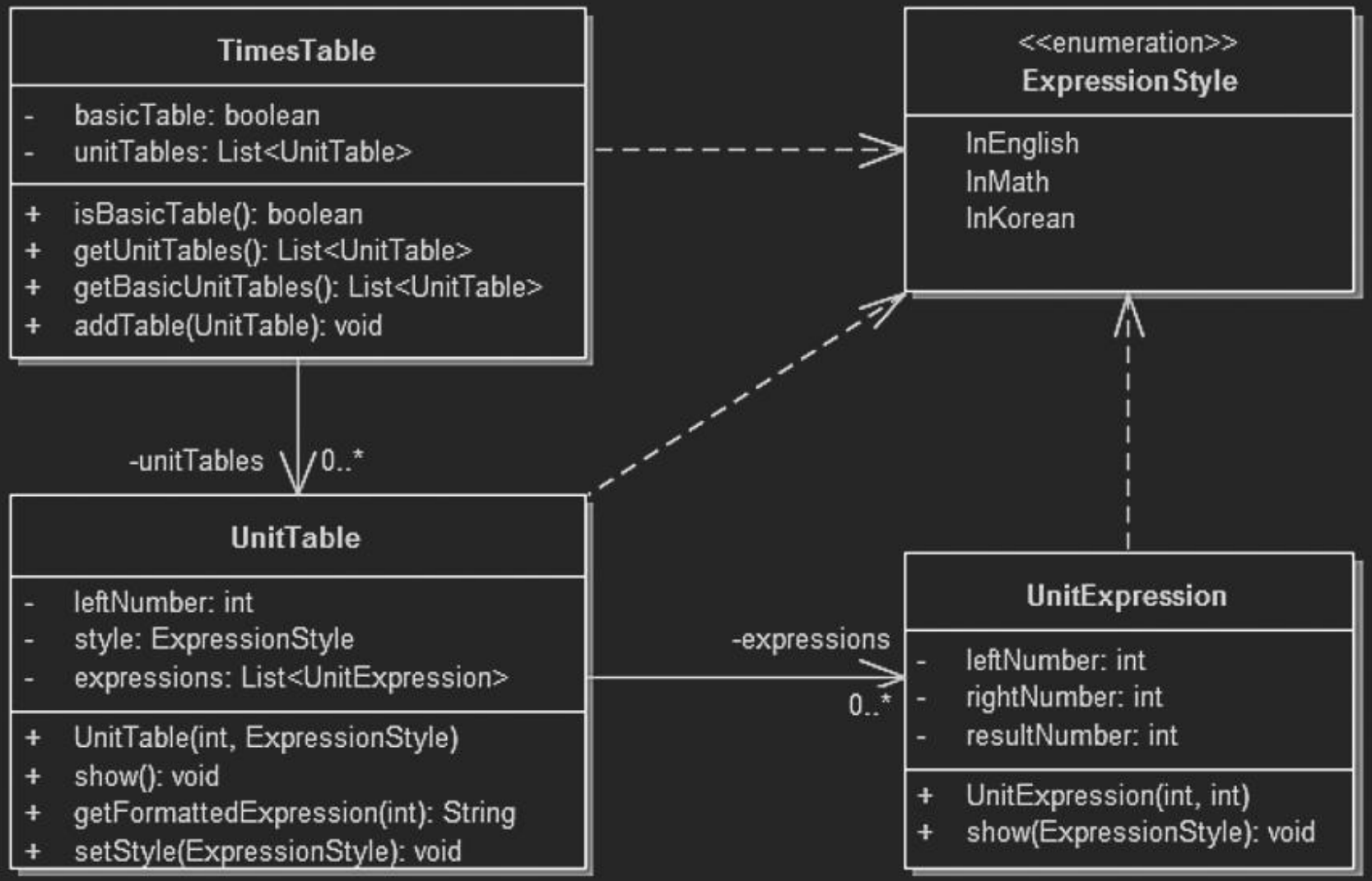

Requirement
Analysis
Design
Implements
Deployment
Runtime
Functional perspective
기능 요건
도메인 모델/UI 모델/데이터 모델
분석 수준 모델
설계 수준 모델
비즈니스 컴포넌트 프로그램 또는 프로그램
배포 비즈니스 컴포넌트, 라이브러리
프로그램 실행
Non-functional perspective
비기능 요건
아키텍처 모델
패턴/스타일
반복적인 분해 모델
아키텍처 요소 프로그램
배포 아키텍처 요소
프로그램 실행
개발 모델
배포 모델
런타임 모델
아키텍처 프레임워크
4 + 1
Use Case view (or Scenarios view)
Logical view
Development view (or Implementation view)
Physical view (or Deployment view)
Process view (or Runtime view)
CMU SEI
N/A
Module view
Allocation view
Allocation view
C&C view
ACDM
N/A
Static perspective
Physical perspective
Dynamic perspective

3. 분리 – 아키텍처와 도메인 모델

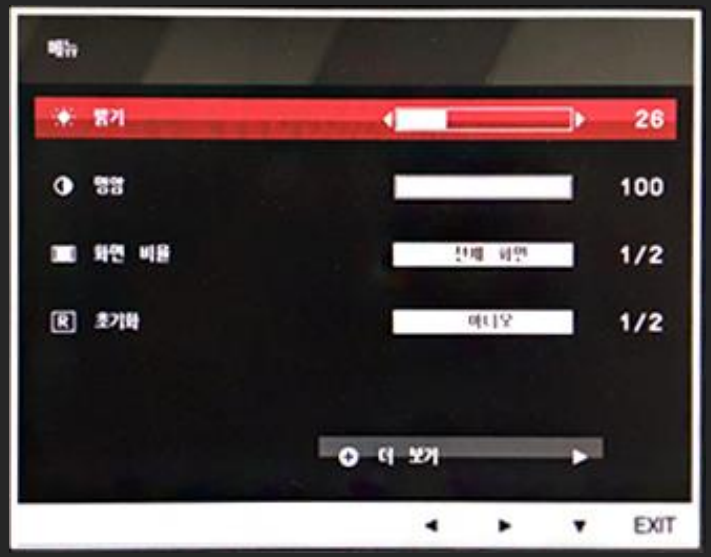
도메인 객체와 메커니즘


UI
프리젠테이션
서비스 발행
도메인
데이터 매핑
데이터
Functional perspective
UI 데이터 객체
서비스 데이터 객체
도메인 객체
관계형 데이터
UI 컴포넌트
프리젠테이션 컴포넌트
서비스 컴포넌트
도메인 컴포넌트
매핑 컴포넌트
RDBMS
Non-functional perspective
Shiro library
Jersey
JQuery /BluePrint
Play framework
Tomcat
Spring DI
JPA/Hibernate
MariaDB
웹브라우저
JVM 1.8
JVM 1.8
HTTP/HTML
REST/JSON
Method call
Method call
SQL
메커니즘

도메인 모델의 가치


TimesTable
- basicTable: boolean
- unitTables: List<UnitTable>
+ isBasicTable(): boolean
+ getUnitTables(): List<UnitTable>
+ getBasicUnitTables(): List<UnitTable>
+ addTable(UnitTable): void
<<enumeration>>
ExpressionStyle
InEnglish
InMath
InKorean
-unitTables
0..*
UnitTable
- leftNumber: int
- style: ExpressionStyle
- expressions: List<UnitExpression>
+ UnitTable(int, ExpressionStyle)
+ show(): void
+ getFormattedExpression(int): String
+ setStyle(ExpressionStyle): void
-expressions
0..*
UnitExpression
- leftNumber: int
- rightNumber: int
- resultNumber: int
+ UnitExpression(int, int)
+ show(ExpressionStyle): void

도메인과 아키텍처를 섞으면...

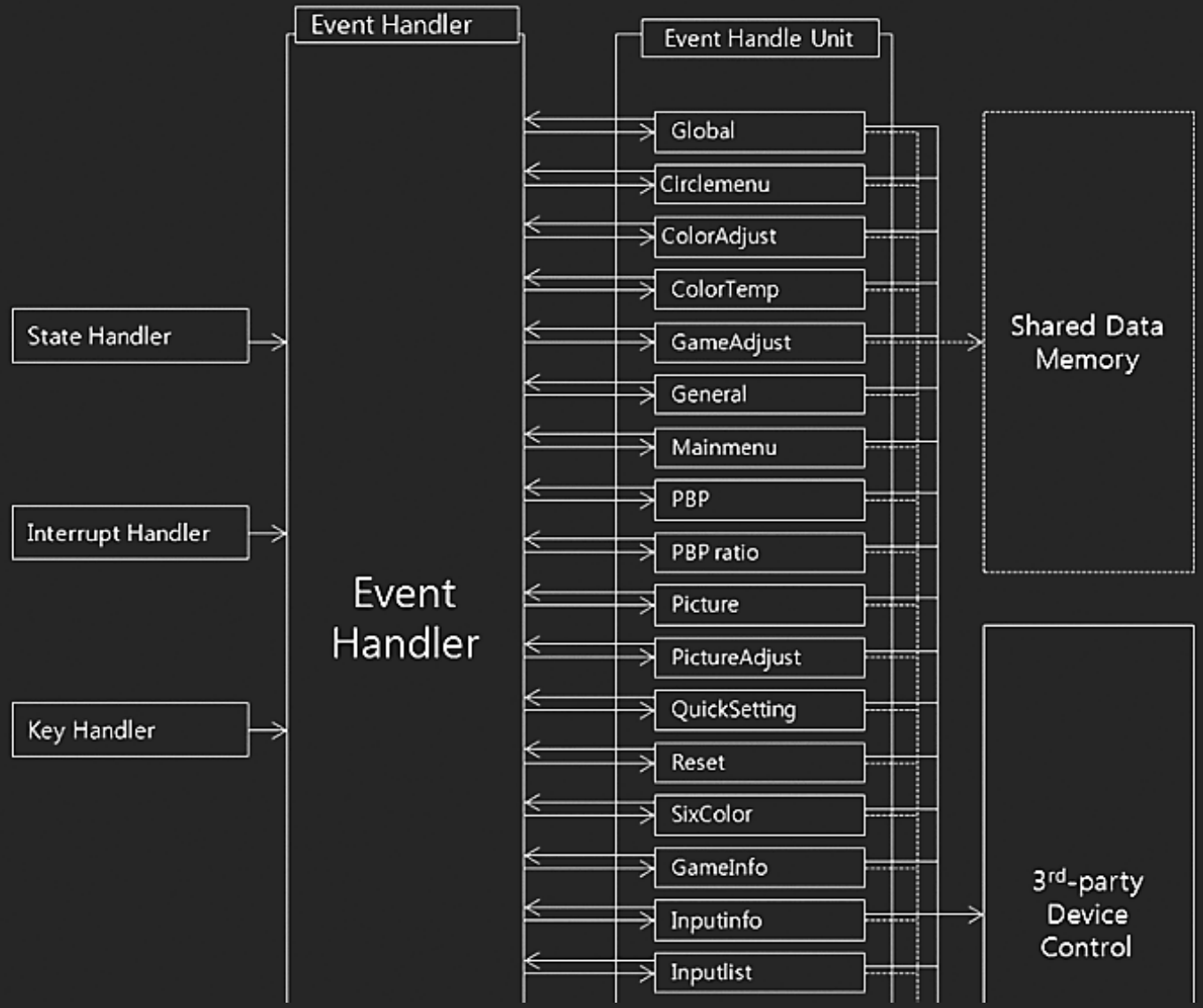


Functional perspective

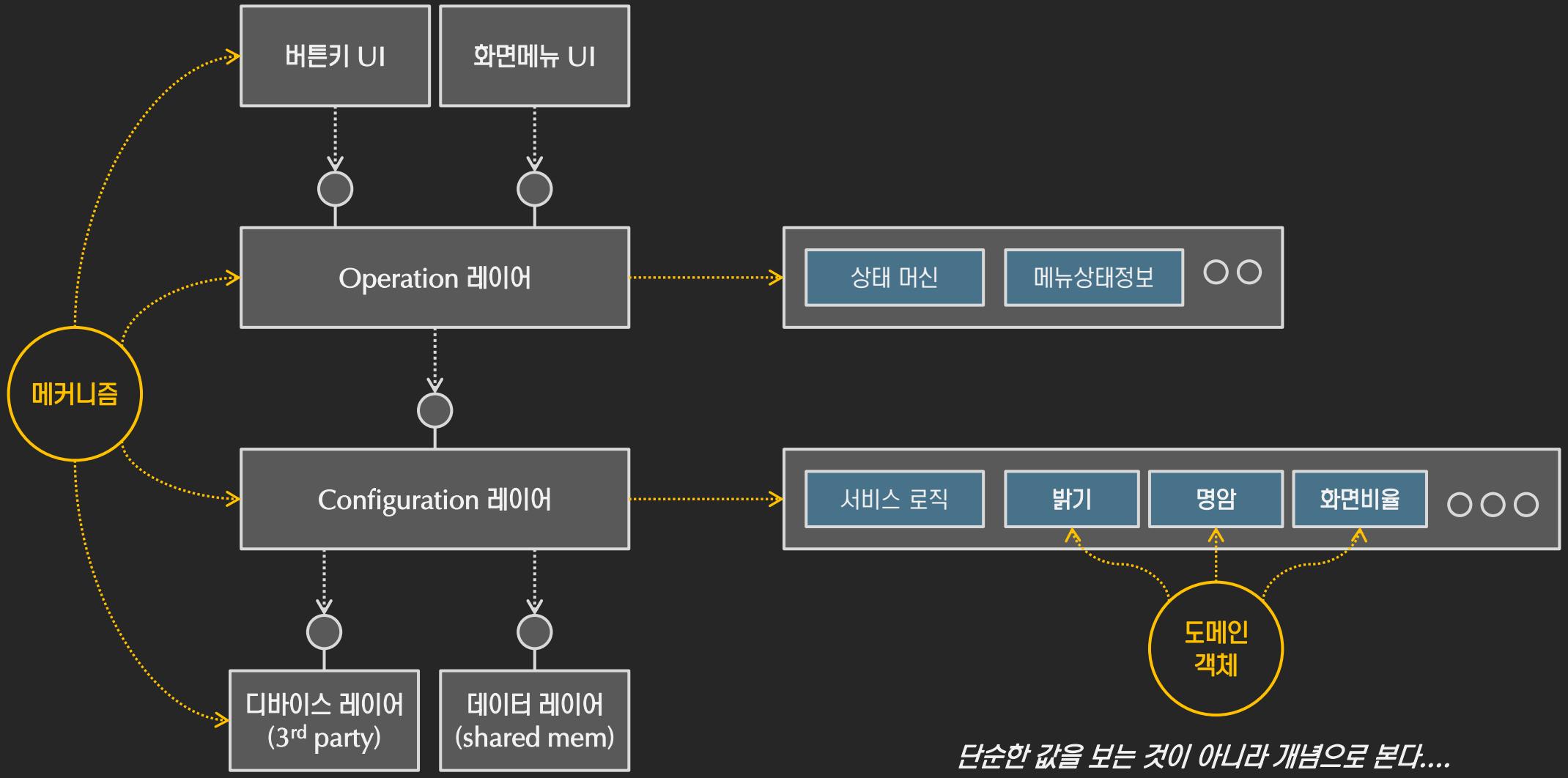
상대적으로 작은 규모 도메인

Non-functional perspective

상대적으로 큰 아키텍처



도메인과 아키텍처를 나누면...



단순한 값을 보는 것이 아니라 개념으로 본다....

무엇을 분리하는가?

도메인 모델 vs 아키텍처 모델

서비스 컨텍스트 vs 사용 컨텍스트

개념 vs 영속성

개념 vs 사용자 인터페이스

관심사의 분리

관심사의 분리(Separation of concern)는 아키텍팅 활동의 Best practice이며 원칙입니다. SoC의 다른 표현은 Loose coupling, High cohesion 입니다. 이런 원칙을 지키며 설계하는 이유는 단순함을 지켜 주기 때문입니다. 결국 기본으로 돌아가서 원칙을 준수하는 설계를 하면, 좋은 소프트웨어를 개발할 수 있습니다.

요약

1. 시스템의 본질을 도메인에 담아야 합니다.

2. 아키텍처는 기준 뷰를 갖고 분해하며 설계합니다.

3. 도메인 모델과 아키텍처 모델은 분리합니다.

감사합니다. !!

Nextree Consulting, Taegook Song