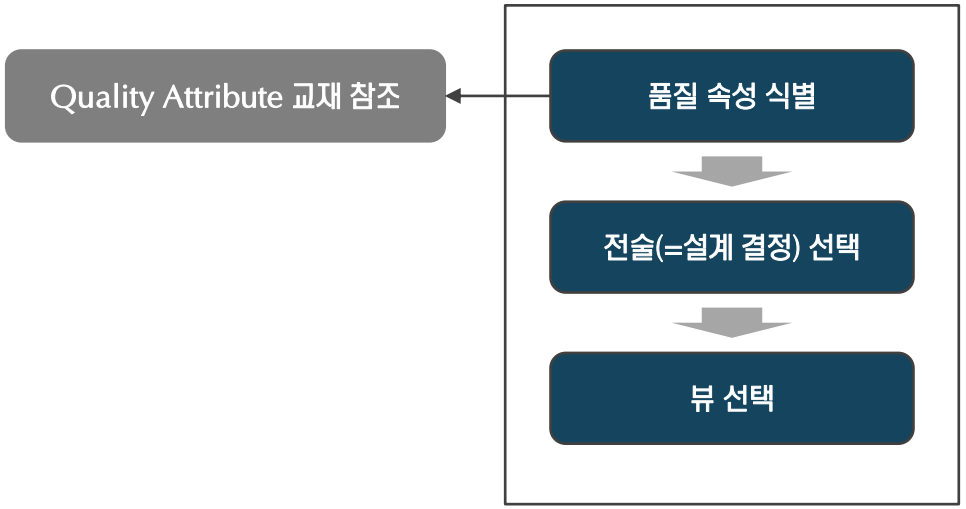
Architecting Process – SEI's ADD

목차

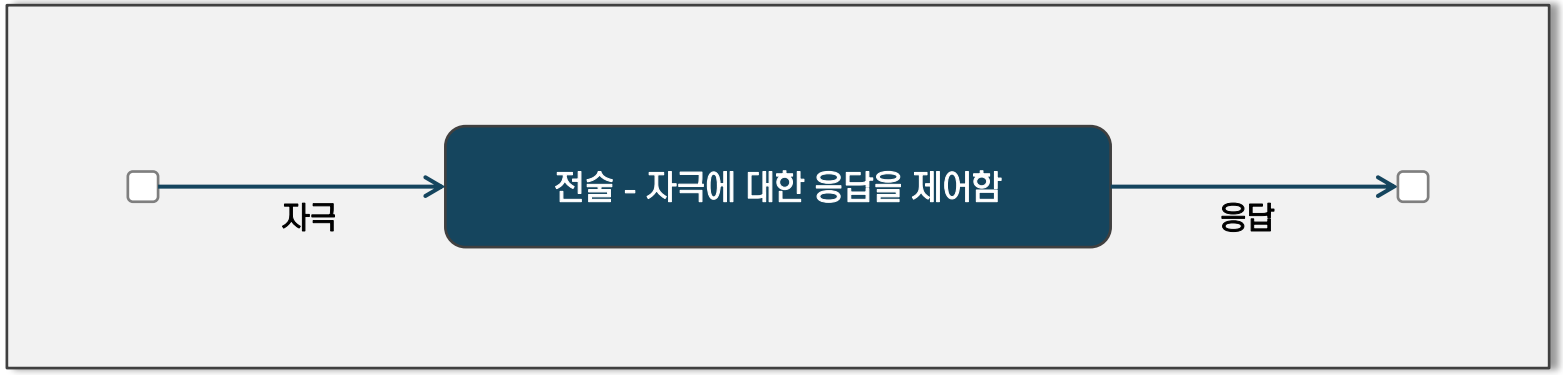
1. SEI 프로세스

- ✓ SEI에서 두 책(Software Architecture in Practice, Documenting Software Architecture) 을 통해 제시합니다.
- ✓ 품질속성으로부터 뷰 선택에 이르는 일련의 아키텍팅 활동에 필요한 지식을 정리합니다.
- ✓ SEI는 소규모, 비-객체지향 개발 환경에서 사용할 수 있는 기본 아키텍팅 프로세스 입니다.



2. 전술 (1/5)

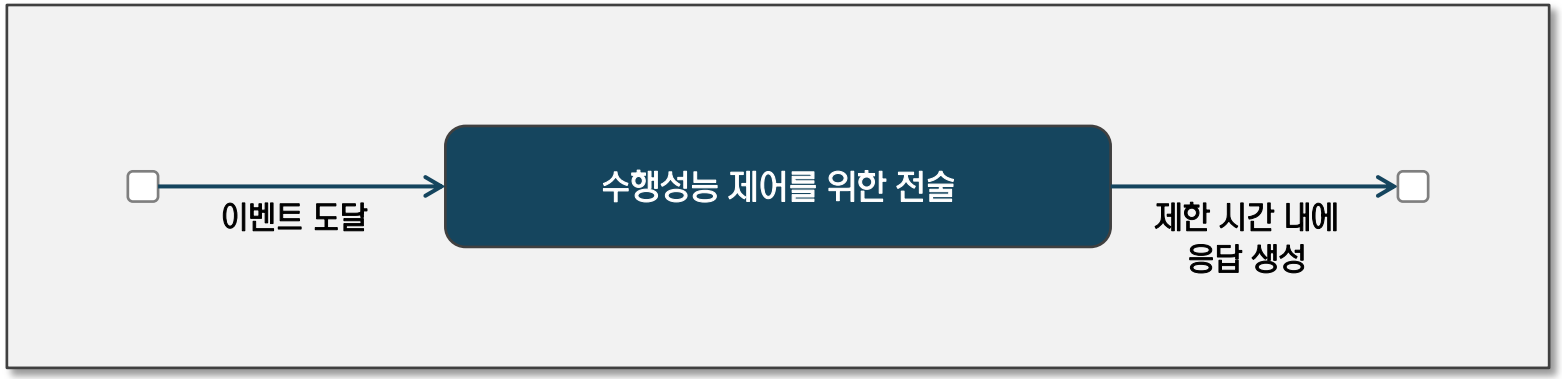
- ✓ 품질 목표 달성 여부는 아키텍처 전술에 의존함, 아키텍처 전술의 집합이 아키텍처 전략임
- ✓ 따라서, 각 전술은 결국 설계 옵션이며, 선택 간에 trade-off 이 존재함
- ✓ 예를 들면, 가용성을 높이기 위해 서버를 늘리면 동기화 문제가 발생함



2. 전술 (2/5) – 수행성능 (1/2)

✓ 수행성능(performance) 전술의 목표

- 주어진 시간 안에 시스템에 도달한 이벤트에 대한 응답을 생성함
- 이벤트 종류 – 메시지 도착, 만기 시간 도래, 상태 변화 감지
- 대기 시간(latency) – 이벤트 도달과 응답 생성 간의 시간

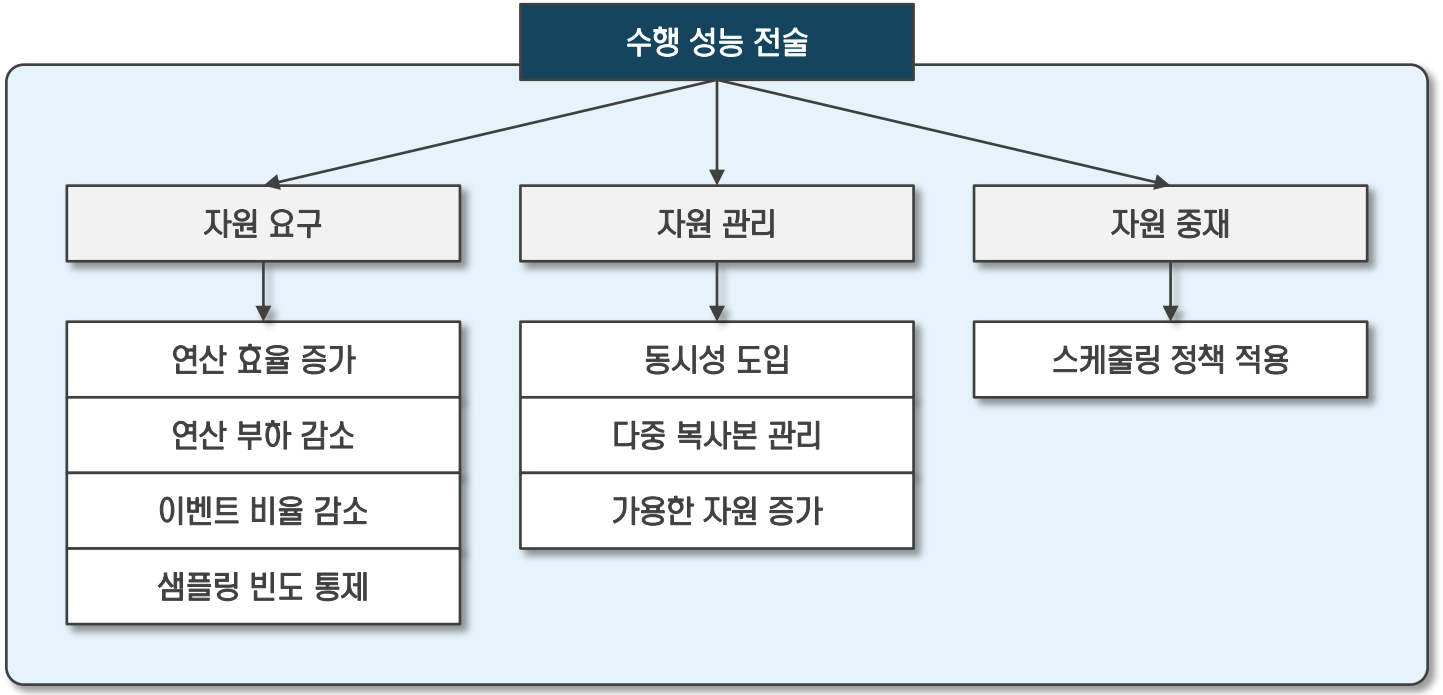


✓ 응답시간을 결정하는 요인

- 자원 소비
- 차단 시간 (blocked time) – 자원 경쟁, 자원 가용성 제한, 타 컴퓨팅 자원에 의존

2. 전술 (3/5) – 수행성능 (2/2)

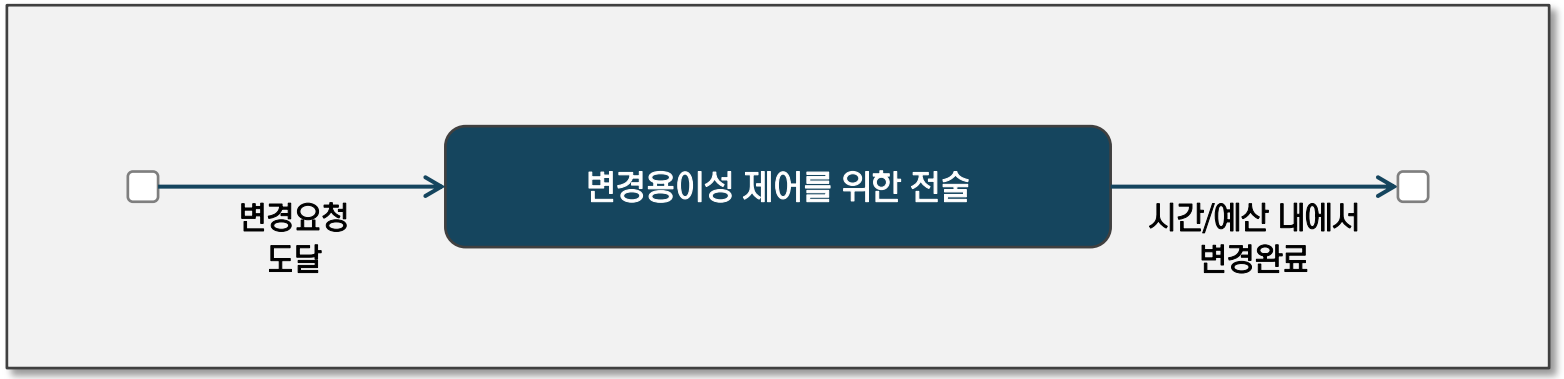
- ✓ 자원 요구 – 대기시간을 줄이기 위해 연산 효율을 높이거나, 처리할 이벤트를 줄이거나, 자원을 제한함
- ✓ 자원 관리 – 동시성을 도입하거나 컴퓨팅 자원의 여러 복사본을 유지하거나, 가용자원을 늘림
- ✓ 자원 중재 – 스케줄링을 도입하여 자원 경합을 피함, 다양한 우선순위(고정, 동적, 정적) 스케줄링을 할 수 있음



2. 전술 (4/5) – 변경용이성 (1/2)

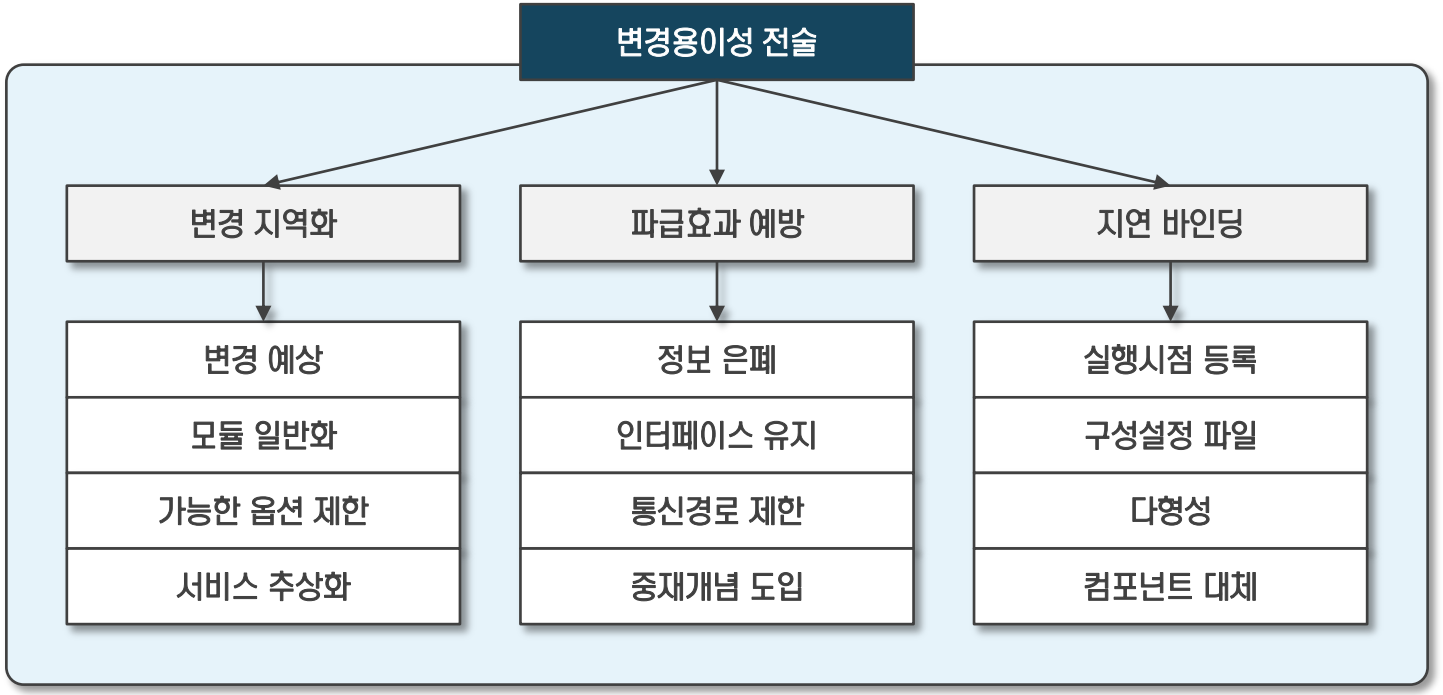
✓ 변경용이성(modifiability) 전술의 목표

- 설계, 구현, 테스트, 배치 변경으로 인한 시간과 비용을 제어



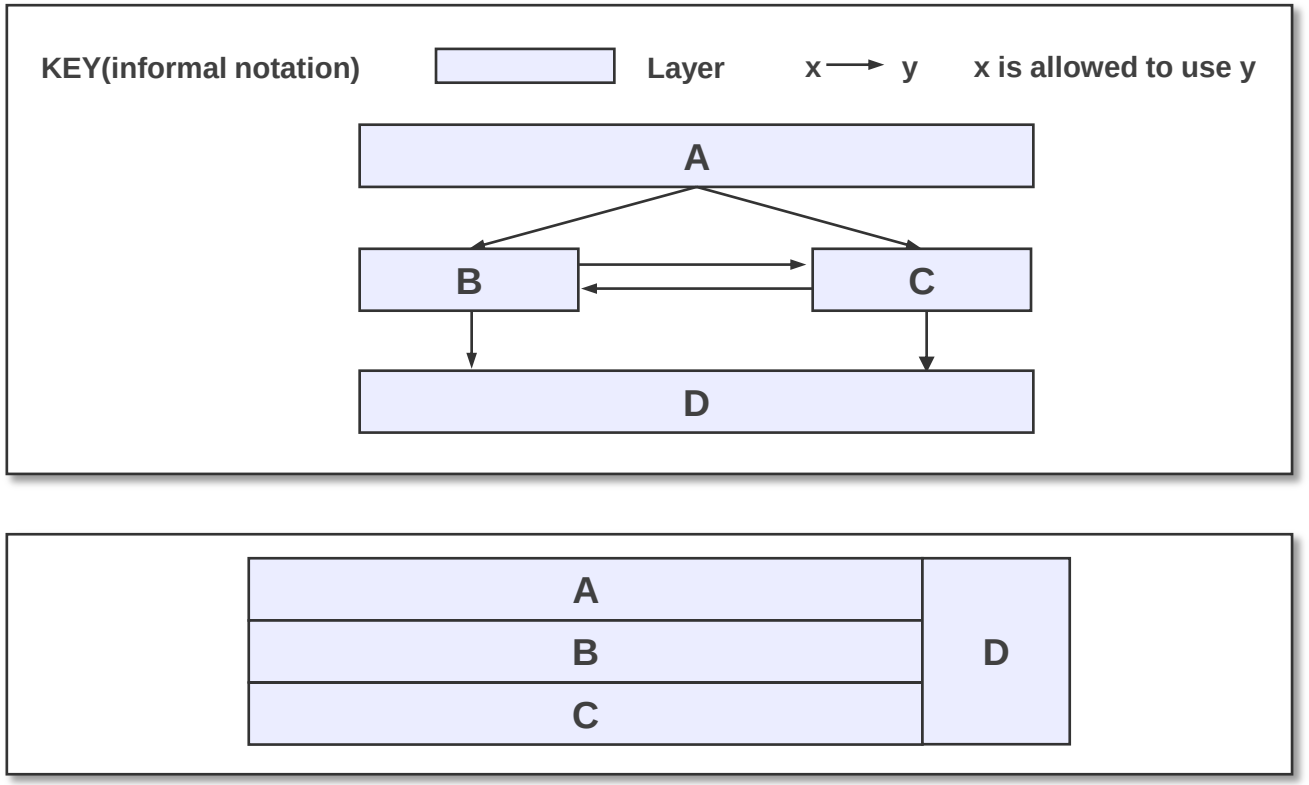
2. 전술 (5/5) – 변경용이성 (2/2)

- ✓ 변경의 지역화 - 변경에 직접 영향을 받는 모듈의 개수를 줄임
- ✓ 파급효과 예방 - 변경범위를 지역 모듈 안으로 제한
- ✓ 바인딩 연기 – 배치 시간과 비용을 통제



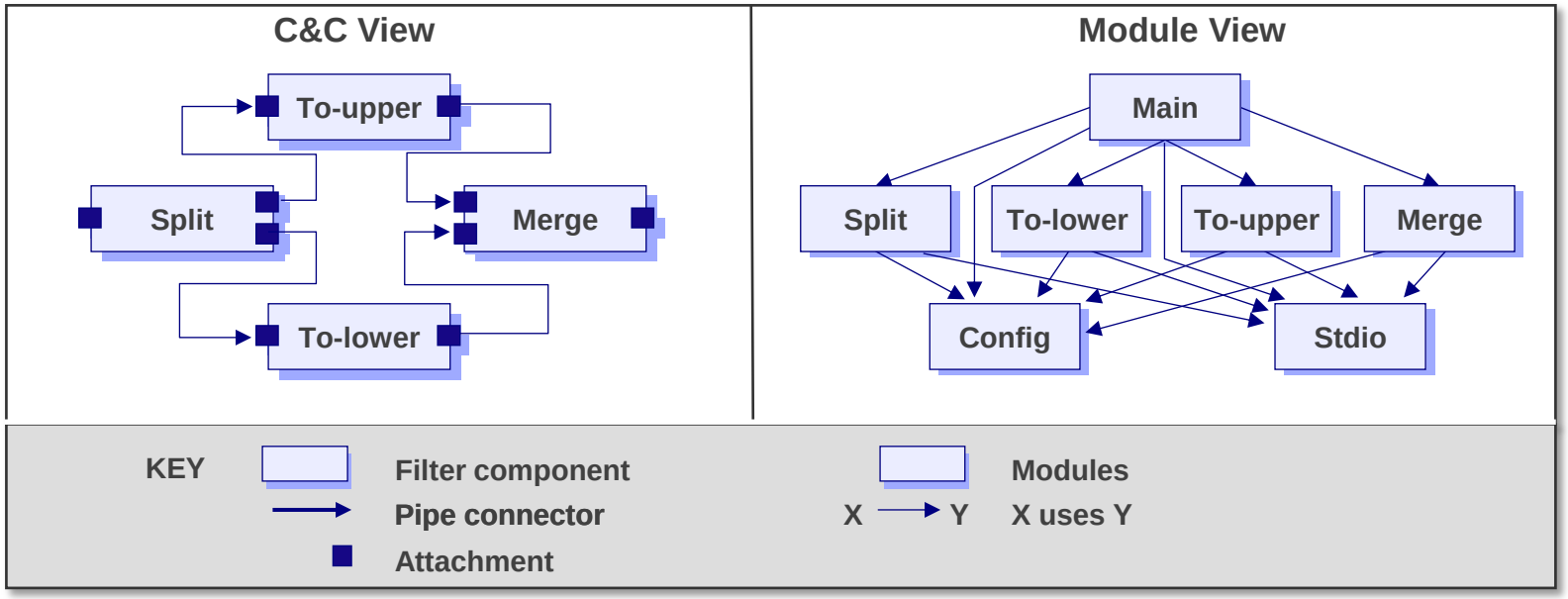
3. 뷰 (1/4) – 모듈 뷰타입

- ✓ 4가지 모듈 뷰타입 스타일 – 분할 스타일, 사용 스타일, 일반화 스타일, 레이어 스타일
- ✓ 모듈은 책임의 집합이며, 구현 단위이며, 기능의 묶음임
- ✓ 모듈은 세 가지 관계를 가지고 있음 – is-part-of 관계, depends-on 관계, is-a 관계



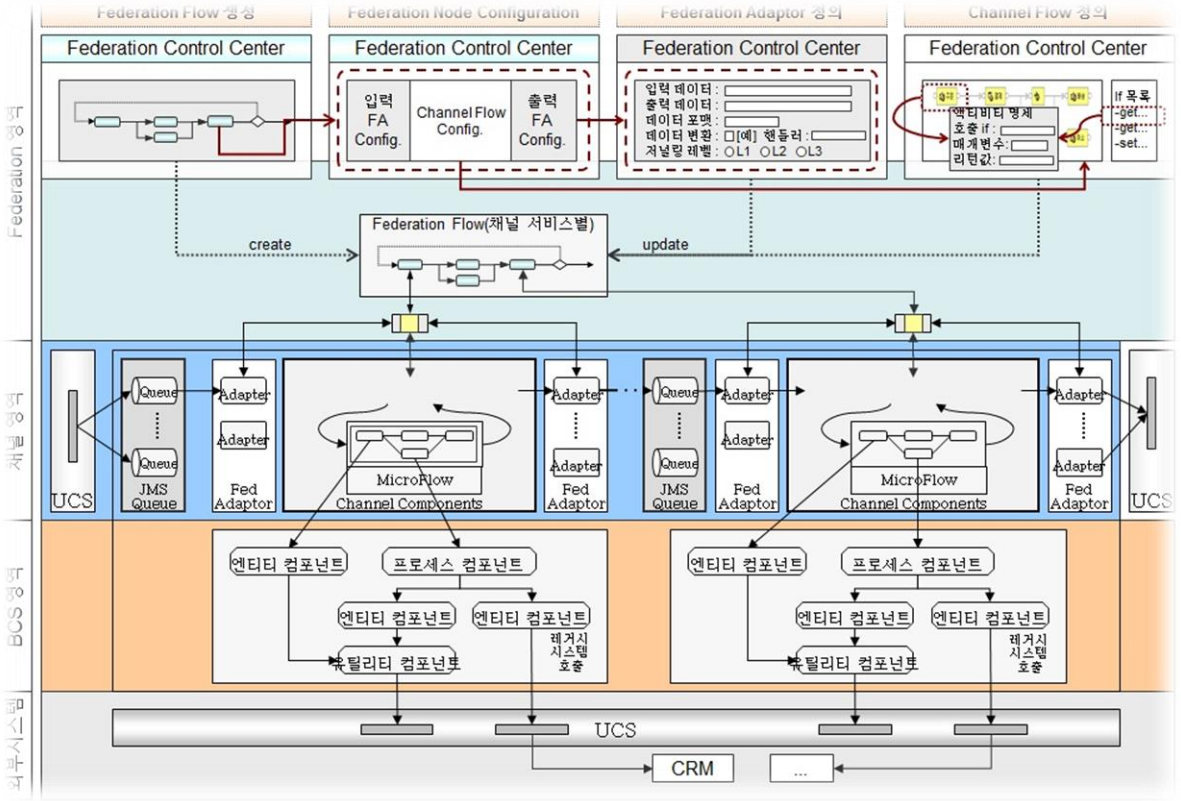
3. 뷰 (2/4) – C&C 뷰타입

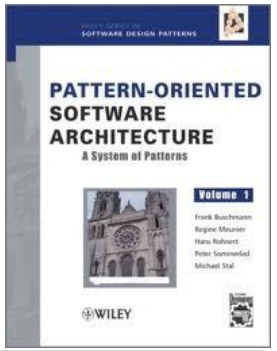
- ✓ 다양한 스타일 – 파이프-필터 스타일, 공유-데이터 스타일, 발행-구독 스타일, 클라이언트-서버 스타일, 피어-피어 스타일
- ✓ 컴포넌트는 실행되는 컴퓨팅 요소나 저장소이며, 커넥터는 여러 컴포넌트 간의 상호작용 경로임
- ✓ 컴포넌트와 커넥터는 부착(attachment) 관계 만 존재함



3. 뷰 (3/4) – C&C 뷰타입

- ✓ 파이프-필터 스타일을 적용한 채널 시스템 설계 사례,
- ✓ C&C 뷰타입의 스타일은 아키텍처 패턴이라고도 불리며, 현대의 소프트웨어 설계에 많이 응용하고 있음
- ✓ POSA(*) 패턴은 아키텍처 패턴의 주류를 이루고 있음

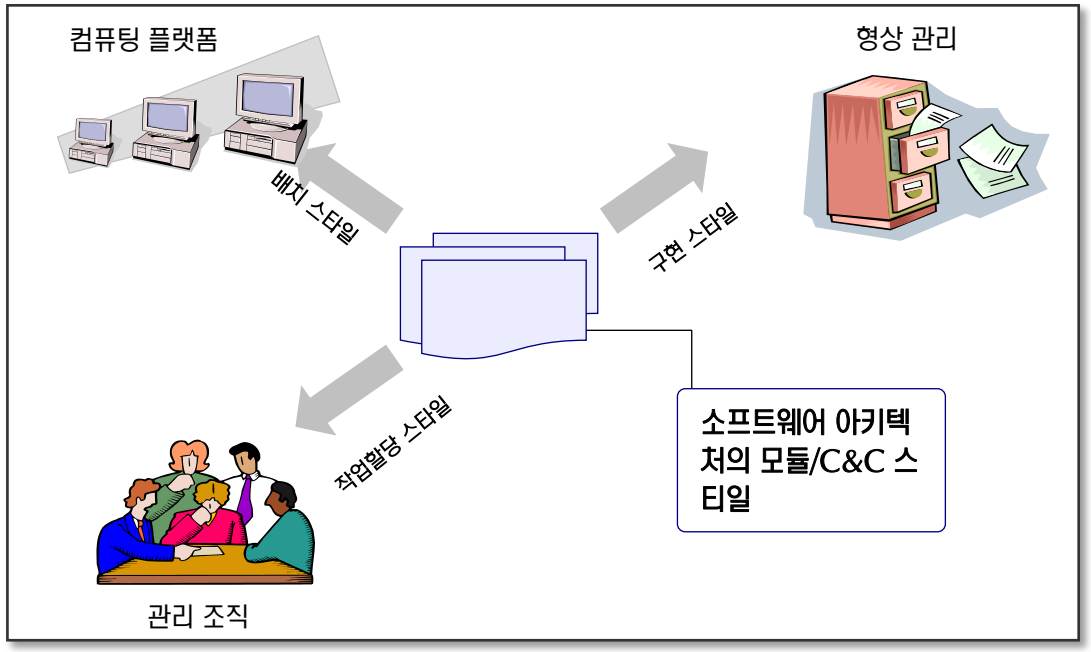


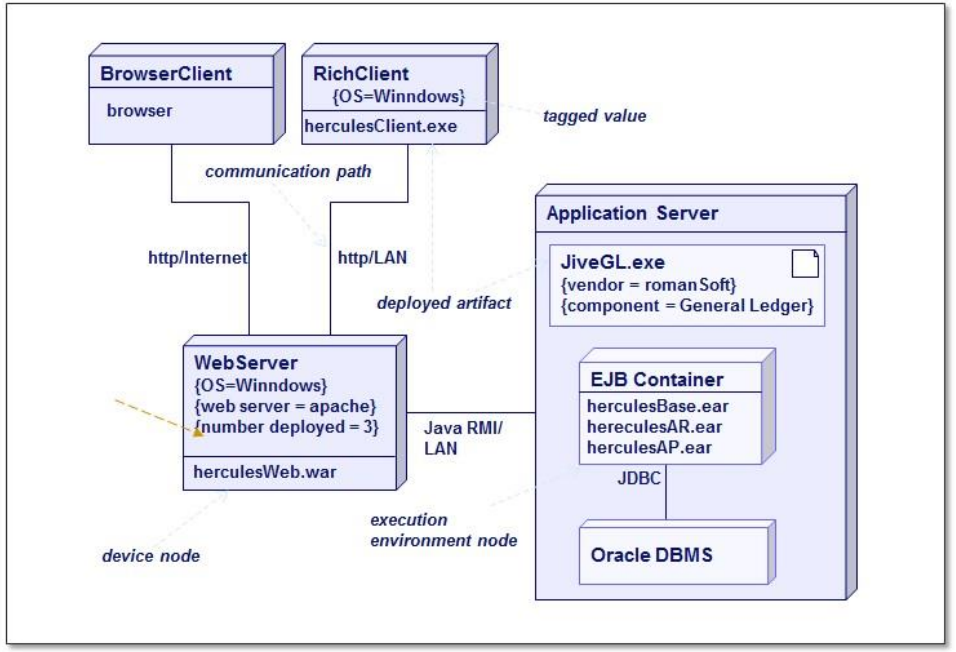


(*)POSA – Pattern Oriented Software Architecture

3. 뷰 (4/4) – 할당 뷰타입

- ✓ 할당 뷰타입 스타일은 배치 스타일, 구현 스타일, 작업할당 스타일 세 가지가 있음
- ✓ 할당 뷰타입은 소프트웨어 아키텍처와 외부 환경 간의 매핑을 표현함, 아키텍처는 HW, 시스템, 팀의 구조에 영향을 줌
- ✓ 소프트웨어 요소와 환경 요소로 구성되어 있으며, 할당(allocation) 관계를 가짐





4. 뷰선택 (1/6) – 이해관계자와 문서화

- ✓ 목표 시스템을 둘러싸고 있는 다양한 이해관계자를 식별하고 그들의 관심사를 확인함
- ✓ 관심의 정도에 따라 뷰와 스타일을 분류한 후, 제거할 것은 제거한 후, 통폐합을 함

이해관계자	모듈 뷰				C&C 뷰	할당 뷰			기타					
	분할	사용	일반화	레이어	Various	배치	구현	작업할당	인터페이스명세서	컨텍스트 다이어그램	뷰 간 매핑	다양성 가이드	분석 결과	타당성 제약조건
프로젝트 관리자	s	s		s		d		d		o				s
개발 팀원	d	d	d	d	d	s	s		d	d	d	d		s
테스터와 통합담당	d	d	d	d	s	s	s		d	d	s	d		s
다른 시스템 설계담당									d	o				
유지보수 담당	d	d	d	d	d	s	s		d	d	d	d		d
프로덕트라인 빌더	d	d	s	o	s	s	s		s	d	s	d		s
고객						o		o		o			s	
최종 사용자					s	o		o					s	
분석가	d	d	s	d	s	d			d	d		s	d	s
인프라 지원 담당	s	s				s	d					s		
신규 이해관계자	x	x	x	x	x	x	x	x	x	x	x	x	x	x
현재와 미래 아키텍트	d	d	d	d	d	d	s	s	d	d	d	d	d	d

Key: d=상세 정보(detailed information), s=일반 정보(some details), o=개요 정보(overview information), x=anything

4. 뷰선택 (2/6) – 절차

✓ 후보 뷰 리스트 생성
- 이해관계자/뷰 테이블 구축
- 아키텍처에 관심을 가진 이해관계자를 행에 나열
- 시스템에 적용될 뷰를 열에 나열
- 각 셀에 이해관계자가 각 뷰의 정보를 얼마나 많이 요구하는지 표시(none, overview only, moderate detail, high detail)

✓ 뷰 통합
- 관리 가능한 크기로 뷰 통합
- 개요만을 요구(overview only)하거나 적은 수의 이해관계자와 관련된 뷰 선택
- 통합 뷰를 위한 후보 선택

✓ 우선 순위 결정
- 먼저 수행할 뷰 선택, 프로젝트에 대한 상세 정도 기준
- 고려사항1: 다른 뷰를 시작하기 전에 하나의 뷰를 완결할 필요는 없음
- 고려사항2: 너비-우선(breadth-first) 접근방법 적용
- 고려사항3: 어떤 이해관계자의 관심이 다른 이해관계자의 관심을 우선함
- 고려사항4: 프로젝트 관리자나 회사 경영진이 더 일찍 그리고 더 자주 정보를 요구하는 경우
- 고려사항5: 아키텍처가 목적에 대한 적합성에 대해 평가되거나 확인되지 않은 경우 이 작업을 우선함

4. 뷰선택 (3/6) – 예제(A-7E) – 1. 후보 뷰 리스트 생성

✓ 이해관계자 목록
- 현재와 미래의 아키텍트(current and future architect)
- 프로젝트 관리자(project manager)
- 개발 팀 멤버(a member of the development team)
- 테스터와 통합자(testers and integraters)
- 유지보수자(maintainers)
- 프로젝트 투자 기관(funding agency)

✓ 품질 속성
- 수정 용이성(modifiability)
- 실시간 성능(real-time performance)
- 점증적 부분집합 배치 능력(ability to be fielded in incremental subset)

✓ 뷰 목록
- 모듈 뷰타입 - 분할 뷰, 사용 뷰, 레이어 뷰
- 컴포넌트-커넥터 뷰타입 - 통신-프로세스 뷰, 공유 데이터 뷰
- 할당 뷰 - 배치 뷰, 구현 뷰, 작업 할당 뷰

4. 뷰선택 (4/6) – 선택 예제(A-7E) – 1. 후보 뷰 리스트 생성

✓ 이해관계자와 아키텍처 뷰 테이블

이해관계자	모듈 뷰			C&C 뷰		할당 뷰		
	분할	사용	레이어	통신처리	데이터공유	배치	구현	작업할당
현재와 미래의 아키텍트	d	d	d	d	d	d	s	s
프로젝트 관리자	s	s	s		o	d	o	d
개발 팀원	d	d	d	d	d	s	s	d
테스터와 통합 담당		d		d	s	s	d	
유지보수 담당	d	d	d	d	d	s	s	s
수행성능 분석 담당	d	d	d	d	s	d		
변경 용이성 분석 담당	d	d	d	s	s	d	o	o
설정 용이성 분석 담당	d	d	d	s	s	d		o
펀딩 에이전시	o	o	o	d	d			

Key: d=상세 정보(detailed information), s=일반 정보(some details), o=개요 정보(overview information), x=anything

4. 뷰선택 (5/6) – 선택 예제(A-7E) – 2. 뷰통합

✓ **뷰 선택**

- 배치 뷰 제외 - 단일 프로세서 하드웨어 환경
- 작업 할당 뷰, 구현 뷰를 모듈 분할 뷰와 통합 - 모듈을 작업 할당과 개발 파일 구조의 기본 단위로 수용
- 컨설턴트들은 공유 데이터 뷰가 무용하다고 결론
- 성능 분석가들은 통신-프로세스 뷰를 사용하여 성능 모델 개발 가능
- 개발자, 테스터, 통합 담당자, 유지보수 담당자는 모듈 분할 뷰와 레이어 뷰 사용

✓ **최종 뷰 목록**

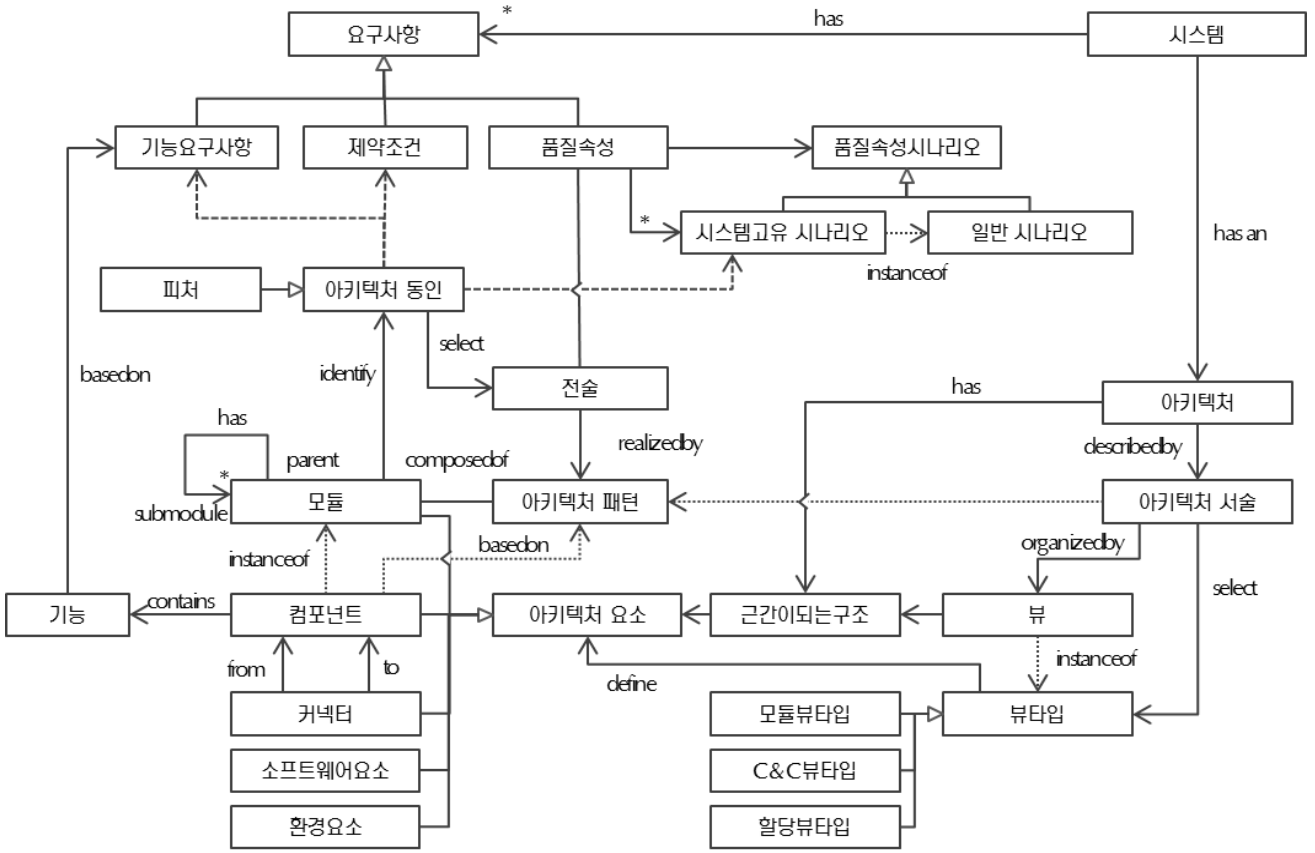
- 분할 뷰
- 사용 뷰
- 레이어 뷰
- 통신-프로세스 뷰

4. 뷰선택 (6/6) – 선택 예제(A-7E) – 3. 우선순위 결정

- ✓ 모듈 분할 뷰와 레이어 뷰
  - 대부분의 이해관계자는 모듈 분할 뷰와 관련
  - 모듈 분할 뷰는 작업 할당과 관련
  - 작업 할당을 위해 어떤 모듈 사용 가능한 지 알아야 함 - 레이어 뷰 우선 할당
- ✓ 통신-프로세스 뷰
  - 통신과 동기화가 결정되고 문서화될 정도로 모듈 분할이 완료된 후 작업
- ✓ 사용 뷰
  - 가장 낮은 우선순위
  - 레이어 뷰의 사용 가능 관계는 사용 뷰의 제약사항

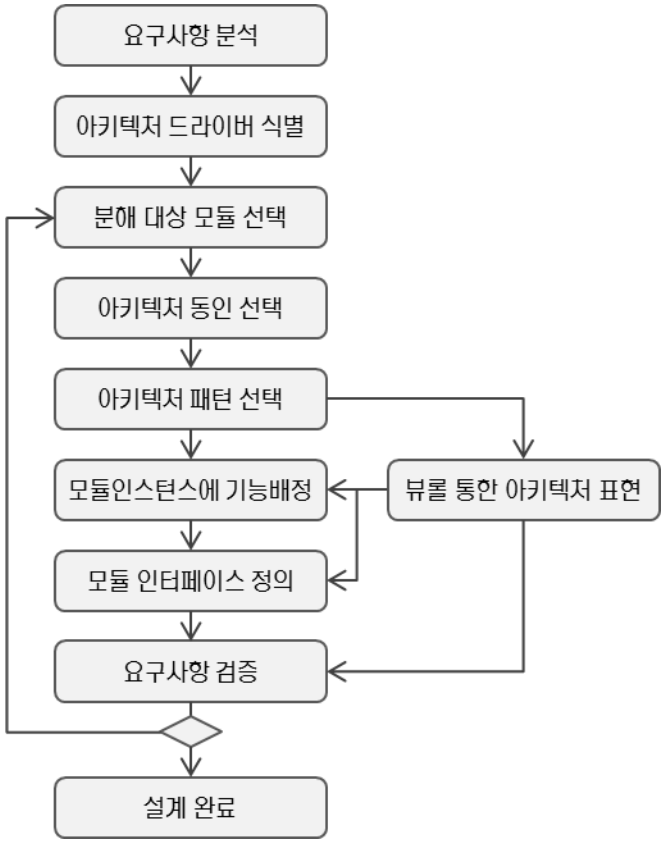
5. ADD 프로세스 (1/2)

- ✓ ADD – [Quality] Attribute Driven Design
- ✓ 아키텍팅 프로세스의 목표를 품질과 기능요구 만족으로 정함
- ✓ ADD input은 품질 속성 시나리오, 품질 목표 달성과 아키텍처 간의 관계에 대한 지식임



5. ADD 프로세스 (2/2)

- ✓ 품질 속성 상의 분해(decomposition) 절차를 기반으로 함
- ✓ 재귀적인 분해 프로세스
  - 각 단계에서, 품질 시나리오를 만족하기 위해 전술과 아키텍처 패턴을 선택
  - 패턴이 제공하는 모듈 타입을 인스턴스화하기 위해 기능을 배정
- ✓ 요구사항 분석 후 아키텍처 동인이 명확히 식별되었을 때 시작
- ✓ ADD 출력 :
  - 아키텍처에 대한 최초 몇 가지 모듈 분해 뷰들
  - 시스템은 기능과 기능들 간의 상호작용을 위한 컨테이너로 간주됨
  - 아키텍처에 대해 최초로 선을 그음 – 거친 입자 수준
  - 아키텍처 설계와 구현을 위한 상세 설계 간의 차별화



6. ADD 프로세스 – 예제 (1/9)

- ✓ 예제 – 홈 정보시스템의 차고 문 개폐를 위한 아키텍처
- ✓ ADD 입력
  - 기능 요구사항과 제약조건
  - 품질 요구사항을 시스템-고유의 품질 시나리오의 집합으로 표현
  - 일반 시나리오로부터 애플리케이션을 위해 시스템-고유 시나리오로 상세화
  - 차고 문을 위한 품질 시나리오
  - 시나리오1 - 문 개폐를 위한 장치와 제어기는 제품 라인에서 다양한 제품 유형별로 다름
  - 시나리오2 - 서로 다른 제품에 사용되는 프로세서는 서로 다름
  - 시나리오3 - 문을 내리는 중에 장애물(사람이나 물체)을 발견하면, 0.1초 내에 정지해야 함
  - 시나리오4 - 차고 문 개폐기는 제품-고유의 진단 프로토콜을 사용하는 홈 정보 시스템이 관리 및 진단을 위해 접근할 수 있어야 함
- ✓ ADD 시작
  - 아키텍처 동인(driver) 식별 완료
  - 요구 변경의 결과 또는 요구에 대한 보다 깊은 이해 후 아키텍처 동인은 변경 가능함
  - 동인이 되는 요구가 명확히 식별되면 ADD를 시작

6. ADD 프로세스 – 예제 (2/9)

- ✓ 예제 – 홈 정보시스템의 차고 문 개폐를 위한 아키텍처
- ✓ ADD 단계
  - 분해할 모듈 선택
  - 다음 단계에 따라 모듈 재정의

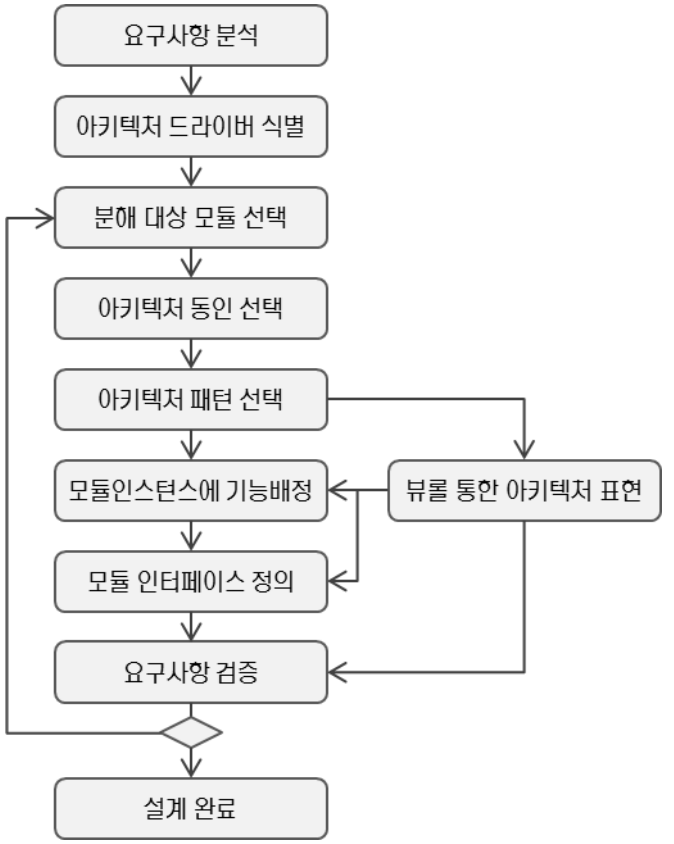
    단계1: 구체적인 품질 시나리오와 기능 요구로부터 아키텍처 동인 선택

    단계2: 아키텍처 동인을 만족하는 아키텍처 패턴 선택

    단계3: 모듈을 인스턴스화 하고 기능을 할당하며 여러 뷰를 사용하여 표현

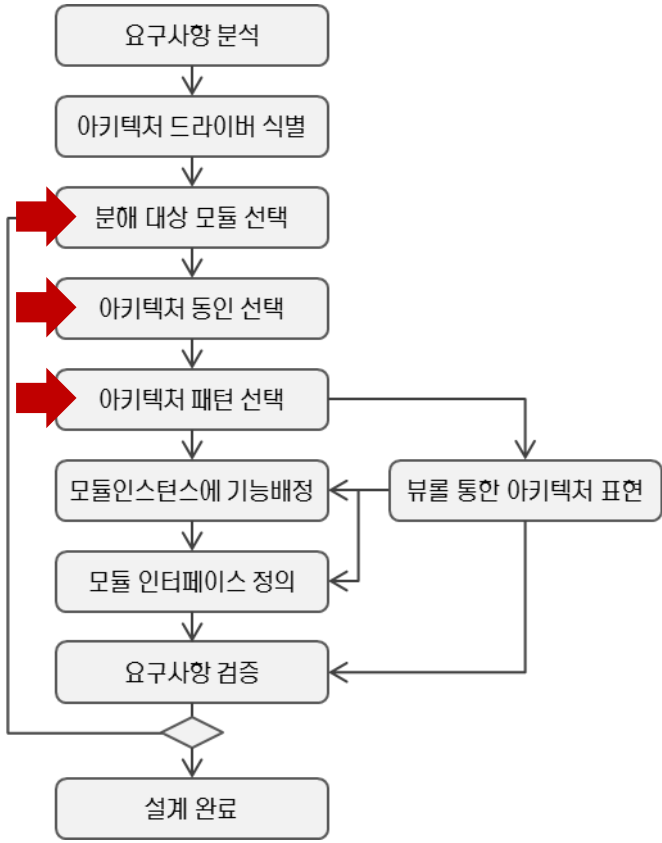
    단계4: 하위 모듈의 인터페이스 정의. 분해는 모듈 상호작용에 대한 모듈과 제약조건을 제공. 이 정보를 각 모듈을 위한 인터페이스 문서에 문서화

    단계5: 유스케이스와 품질 시나리오를 검증하고 정의하고 이 제약조건을 하위 모듈에도 적용
  - 추가 분해를 필요로 하는 각 모듈에 대해 위의 절차를 반복



6. ADD 프로세스 – 예제 (3/9)

- ✓ 분해할 모듈 선택
  - 시스템 -> 하위 시스템 -> 하위 모듈
  - 예제의 모듈 : 차고 개폐기(시스템)
  - 제약 조건 : 개폐기는 홈 정보 시스템과 상호작용해야 함
- ✓ 아키텍처 동인 선택
  - 아키텍처 동인 = 아키텍처를 이끄는 기능과 품질 요구사항의 조합
  - 예제의 요구 : 실시간 수행성능[0.1초], 제품 라인 지원을 위한 변경 용이성, 온라인 진단
  - 아키텍처 프로토타입 요구
- ✓ 아키텍처 패턴 선택
  - 각 품질속성 별 전술이 존재, 전술 구현을 위해 사용할 패턴 존재
  - 전술은 하나 이상의 품질에 영향을 주며, 패턴은 다른 품질 속성에 영향을 줌
  - 아키텍처 설계에서, 여러 품질 간의 균형을 위해 여러 전술의 조합을 사용 (계속)



6. ADD 프로세스 – 예제 (4/9)

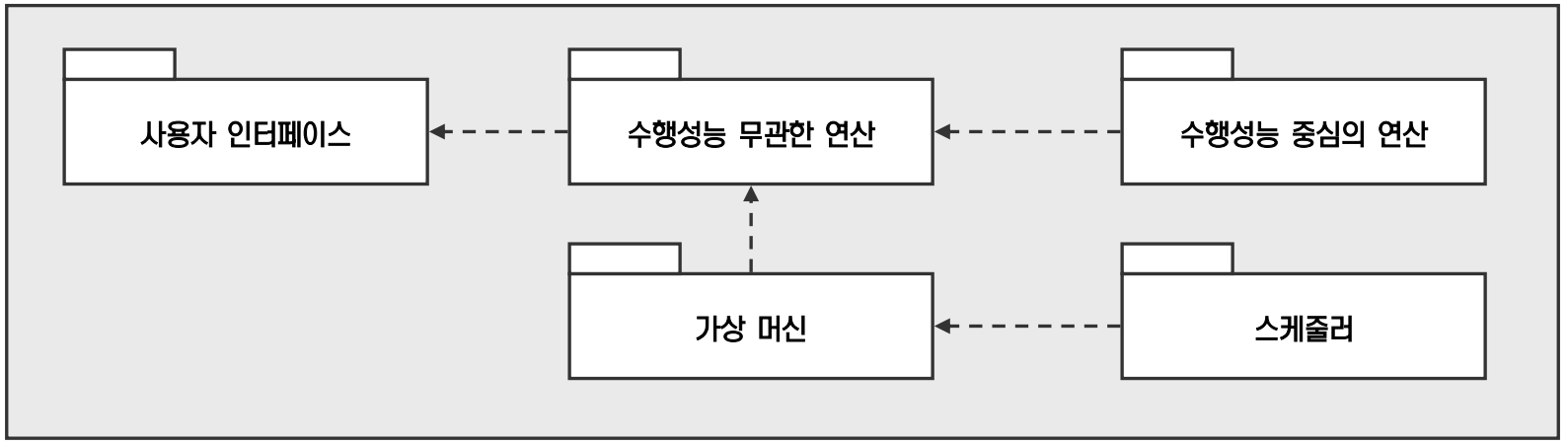
✓ 아키텍처 패턴 선택 (계속)

- 이 단계의 목표 : 모듈 타입을 구성하는 전체적인 아키텍처 패턴을 수립
- 전술 선택의 두 가지 가이드 : 동인, 적용된 패턴이 다른 품질에 주는 부작용(side effect), Side effect 의 예 : 변경 용이성을 위해 HTML 해석기를 사용, 이는 수행성능에 영향을 줌
- 예제(변경 용이성, 수행성능)를 위한 전술 세 가지

  전술1: 정보 은폐, 모듈의 분리 - 사용자 인터페이스, 통신, 진단, 센서를 다루는 모듈, 각 모듈을 가상 머신 이라 함

  전술2: 연산 효율성 증대

  전술3: 스케줄링 정책 사용

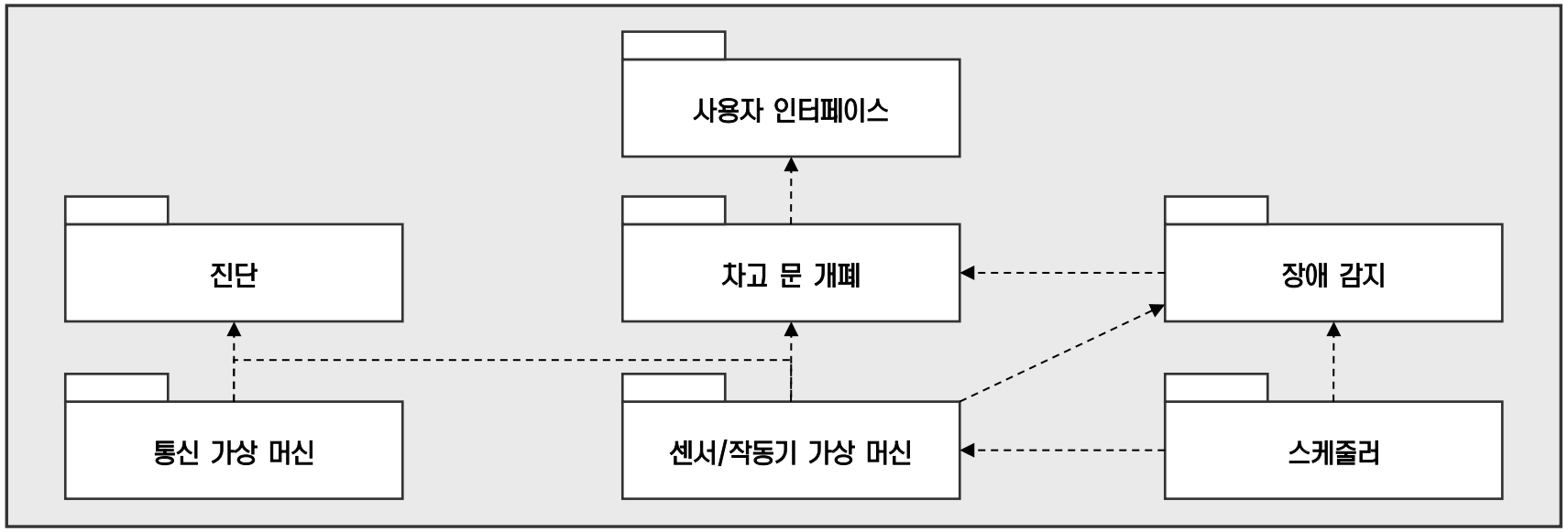


[차고 문 동인 달성을 위한 전술을 활용하는 아키텍처 패턴]

6. ADD 프로세스 – 예제 (5/9)

✓ 모듈 인스턴스화

- 수행 성능과 무관한 연산은 통신과 센서를 관리하는 가상 머신 위에서 실행
- 가상 머신 위에서 실행되는 소프트웨어 = 애플리케이션
- 실재 시스템에서 하나 이상의 모듈을 가지며, 기능의 각 “그룹”에 대해 하나의 모듈이 존재
- 예제를 위한 할당
- 장애물 감지 관리와 차고 문 정지 부분을 수행성능 중심 모듈에 할당
- 차고 문 일반 개폐는 수행성능 무관 모듈에 할당
- 진단 부분은 수행성능 무관 모듈에 할당



[차고 문 개폐기의 레벨 1 분해도]

6. ADD 프로세스 – 예제 (6/9)

✓ 기능 할당

- 상위 모듈에 관계된 유스케이스 적용은 기능 분산에 대한 이해를 높임
- 필요한 기능을 수행하기 위해 하위 모듈을 추가하거나 제거함
- 상위 모듈에 적용되는 유스케이스는 하위 모듈 안에서 일련의 책임으로 표현되어야 함
- 책임 할당을 통해 생산자/소비자 관계 형성
- 이 단계에서 정보 교환 방법, 메시지 타입 등은 설계 단계로 넘김
- 초점 : 정보 자체와 역할(생산자/소비자)
- 생산자와 소비자 간에 다양한 상호작용 패턴이 나타남(publish-subscribe)
- 품질 충족여부 확인을 위해 동적 실행시간 배치 정보 등이 추가로 필요함 (뷰를 통해 표현)

6. ADD 프로세스 – 예제 (7/9)

✓ 뷰를 통한 아키텍처 표현

- 세 가지 뷰로 충분 – 모듈 분해, 동시성, 배치
- 방법론 자체는 선택된 특정 뷰에 종속되지 않음, 따라서 필요한 뷰는 추가 가능
- 모듈 분해 뷰 – 각 모듈은 책임을 담는 컨테이너이며, 각 모듈 간의 데이터 흐름을 파악
- 동시성 뷰 – 병렬 활동과 동기화와 같은 시스템의 특성인 다음을 모델링 함, 자원 경쟁 문제, 데드락, 데이터 일치성
- 동시성 문제에 대한 고찰

  .두 사용자가 동시에 유사한 일을 함 - 자원 경쟁이나 데이터 일치성 문제를 다룸 [한 사람은 원격으로 문을 닫고, 다른 사람은 스위치를 이용하여 문을 염]

  .한 사용자가 동시에 여러 활동을 수행함 – 데이터 교환이나 활동 제어 문제를 다룸 [사용자가 문을 열면서 동시에 진단을 수행함]

  .시스템을 시작함 – 시스템 안에서 실행 활동에 대한 개요와 초기화 방법을 보여 줌 [차고 문 개폐 시스템은 홈 정보 시스템의 가용성에 의존하는가? 차고 문 개폐 시스템은 항상 작동하며 신호를 기다리는가?]

  .시스템을 종료함 – 일관된 시스템 상태의 달성이나 저장과 같은 마무리(cleaning up)문제를 다룸
- 배치 뷰 – 다중 프로세서와 특수한 하드웨어가 사용됨, 배치 뷰 사용을 통해 원하는 품질 달성을 지원하는 배치를 결정하고 설계하는데 도움을 줌

6. ADD 프로세스 – 예제 (8/9)

✓ 하위 모듈의 인터페이스 정의

- 모듈의 인터페이스는 필요한 서비스와 특성을 보여 줌
- 구조(모듈 분해 뷰), 동적인 특성(동시성 뷰), 실행시간(배치 뷰)의 관점에서 분해를 분석하고 문서화 함으로써 하위 모듈 간의 상호작용 가정을 보여 줌
- 모듈 뷰의 문서화 내용

  .정보의 생산자/소비자,

  .서비스를 제공하고 사용하기 위한 모듈을 필요로 하는 상호작용 패턴
- 동시성 뷰의 문서화 내용

  .서비스를 제공하고 사용하는 모듈에 대한 인터페이스를 리드하는 쓰레드 간의 상호작용

  .활동 컴포넌트에 대한 정보

  .동기화하고 직렬화 하는 컴포넌트에 대한 정보
- 배치 뷰의 문서화 내용

  .하드웨어 요구사항 – 특수 목적의 하드웨어

  .처리 성능 요구사항 – 프로세서의 처리 속도가 10 MIPS는 되어야 함

  .통신 요구사항 – 1초에 1회 이상 갱신되어야 함

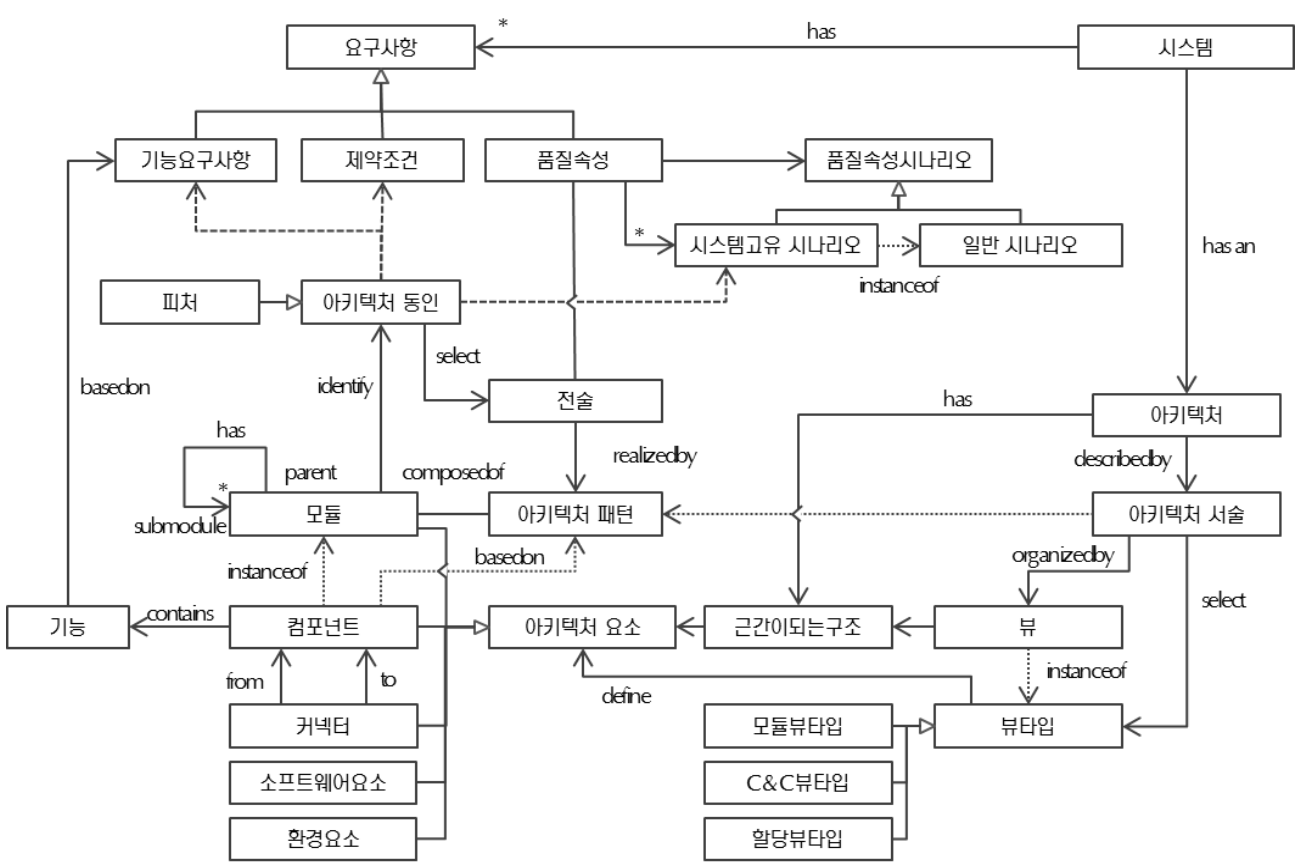
6. ADD 프로세스 – 예제 (9/9)

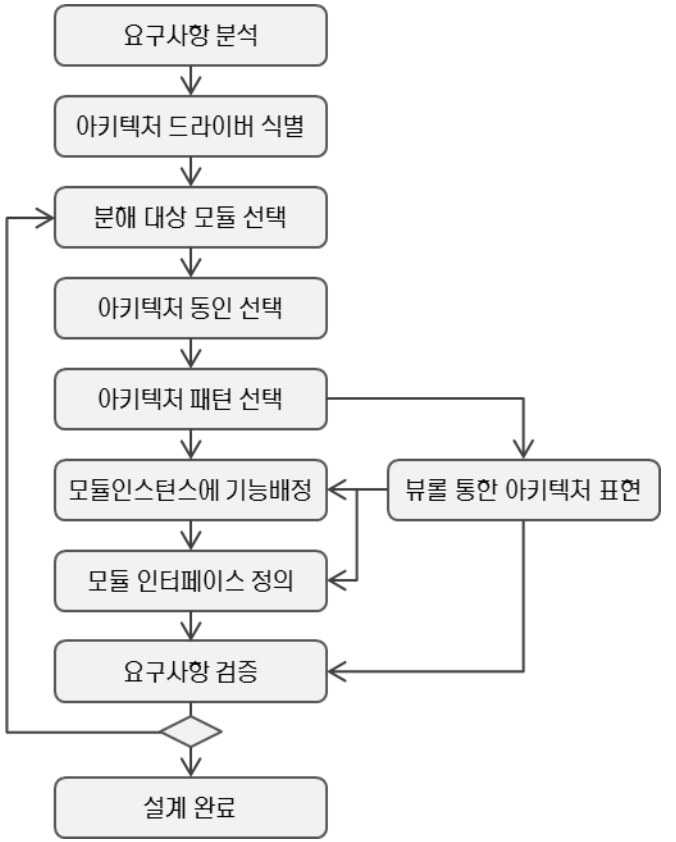
✓ **하위 모듈을 위한 제약 조건인 유스케이스와 품질 시나리오를 검증하고 재정의 함**

- **기능 요구**
  - .차고 문 개폐기는 요청에 따라 지역적으로 또는 원격으로 문을 열고 닫음
  - .장애가 감지되었을 경우 0.1초 이내에 멈추어야 함
  - .홈 정보 시스템과 상호작용하고 원격 진단을 지원하여야 함
- **책임은 다음의 모듈로 분해**
  - .사용자 인터페이스, 문 개폐 모듈, 장애물 감지
  - .통신 가상 머신, 센서/작동기(actuator) 가상 머신
  - .스케줄러, 진단
- **제약 조건**
  - .분해는 제약조건을 만족하여야 함
  - .제약조건은 단일 하위 모듈에서 만족
  - .제약 조건은 여러 하위 모듈에서 만족, 예, 홈정보 시스템과 통신이 유지되어야 함 – 통신가상머신은 통신 단절을 감지함
- **품질 시나리오**
  - .문을 열고 닫기 위한 장치와 제어기는 서로 다른 제품 라인에서 서로 달라야 함
  - .서로 다른 제품에서 사용되는 프로세서는 달라야 함
  - .차고 문을 내리는 동안 장애물을 감지하면, 문은 0.1초 이내에 멈추어야 함
  - .차고 문 개폐기는 홈 정보 시스템 안에서 제품-고유의 진단 프로토콜로 접근되어야 함

7. ADD 프로세스 요약

- ✓ ADD 프로세스는 현대의 다양한 아키텍팅 프로세스의 바탕이 되는 프로세스임
- ✓ 소규모 시스템 또는 Embedded 시스템 개발에서는 아직도 활발하게 사용되는 프로세스 임
- ✓ 프로젝트에서 개발하려는 목표 시스템과 개발팀의 특성에 맞추어 조정(tailoring) 하여야 사용함





8. 토의

- ✓ 질의 응답
- ✓ 토론

감사합니다....

❖넥스트리컨설팅(주)

❖CEO 송태국 / 대표 컨설턴트

❖tsong@nextree.co.kr