모델 기반 임베디드 소프트웨어 개발 방법론

2006. 2. 14

한국정보과학회 프로그래밍언어연구회 2006 겨울학교

홍 장 의 / selab.chungbuk.ac.kr

Microwave Oven

일반적인 가정용 전자제품

- 음식을 조리하기 위해 사용하는 단순하고 간편한 가정용 전자제품
- 마이크로 오븐을 제어하기 위하여 내장된 소프트웨어의 실행을 통해 오븐을 동작시킴
- 사용자가 버튼의 입력을 통해 오븐의 기능을 수행시키고, 간단한 정보를 보여줌.

전자레인지 외관도

Microwave Oven

마이크로 오븐 개발시 요구사항들

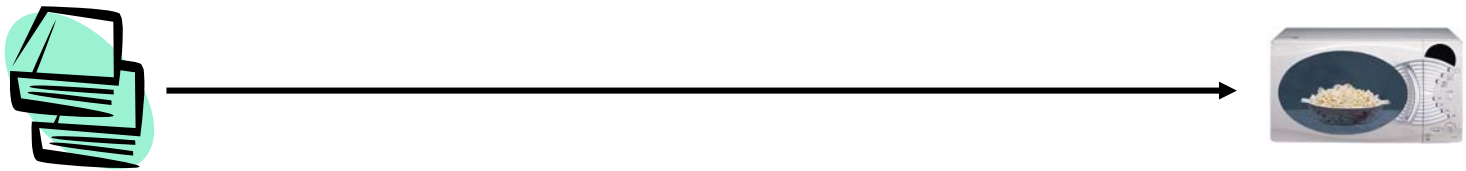
- 마이크로 오븐은 버튼을 통해 동작한다. Cook 버튼, Cancel 버튼 등의 다양한 버튼이 존재한다.
- 오븐이 동작할 때, 오븐 내부의 램프가 점등된다.
- 오븐의 조리 동작은 Cook 버튼이 눌려졌을 때 시작된다. 오븐의 동작은 자동 조리와 수동 조리 모드가 지원된다.
- 수동 조리모드에서 오븐의 동작은 다음과 같이 발생한다.
  - 버튼을 1회 누루면 1분간 동작한다.
  - 버튼을 누른 횟수만큼 1분씩 증가한다.
- 동작중 오븐의 도어가 열리면 조리가 종료된다.
- 동작중 Cancel 버튼이 눌리면 조리가 종료된다.
- 조리가 완료되면, Power tube와 램프가 정지하고, Beep 소리를 울린다.

소프트웨어 개발

Software Development

Requirements

Final System



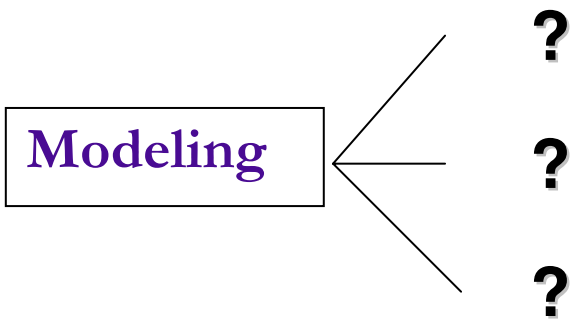
Requirements = Final System

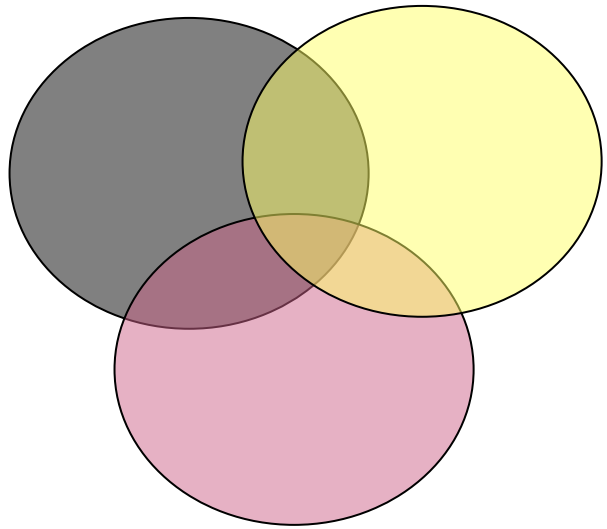
Software Development is Continuous Modeling Activities

소프트웨어 모델링

What is Modeling ?

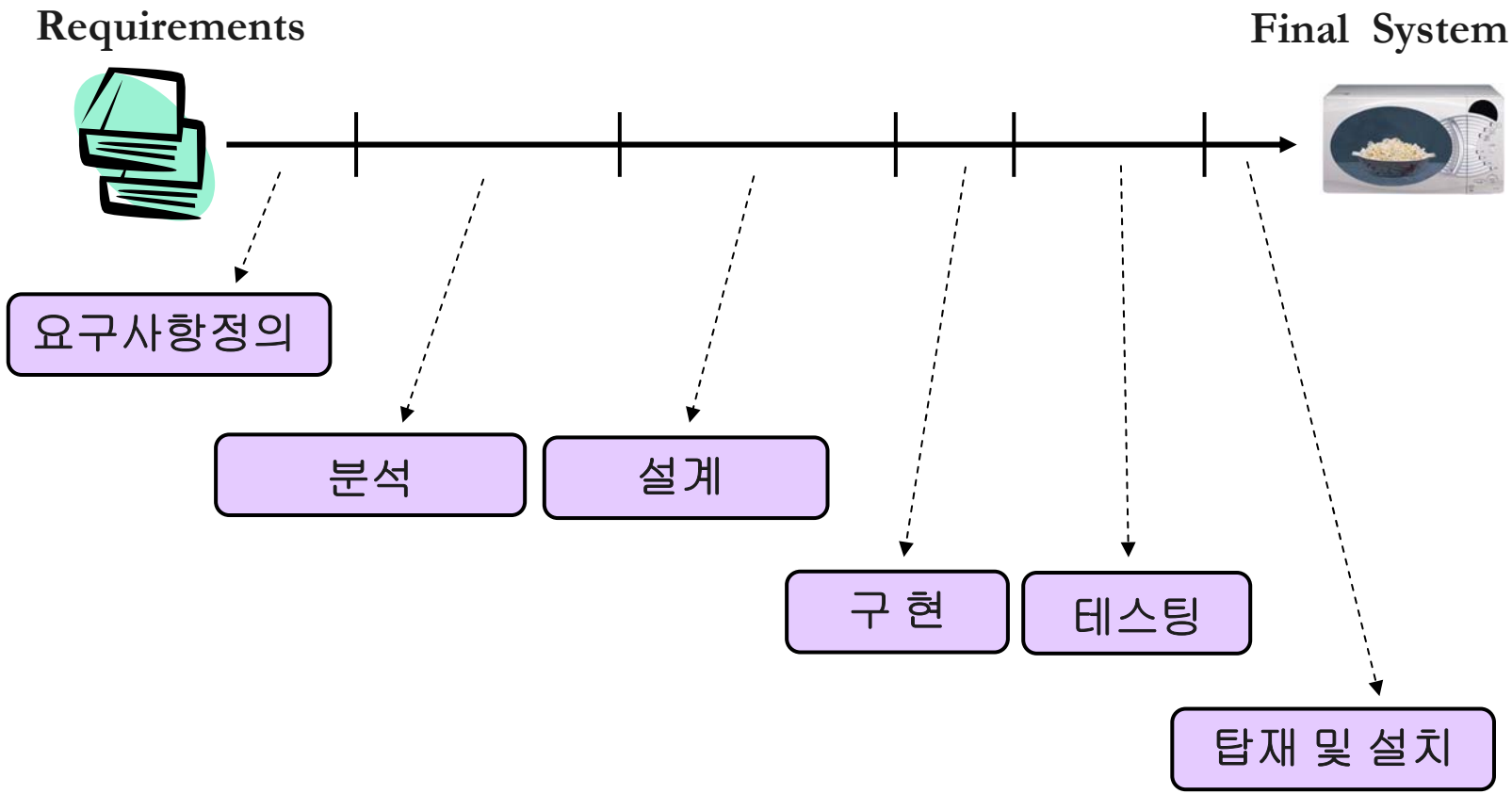
What is Disciplined Approach for Modeling ?





개발 절차

Software Development Process

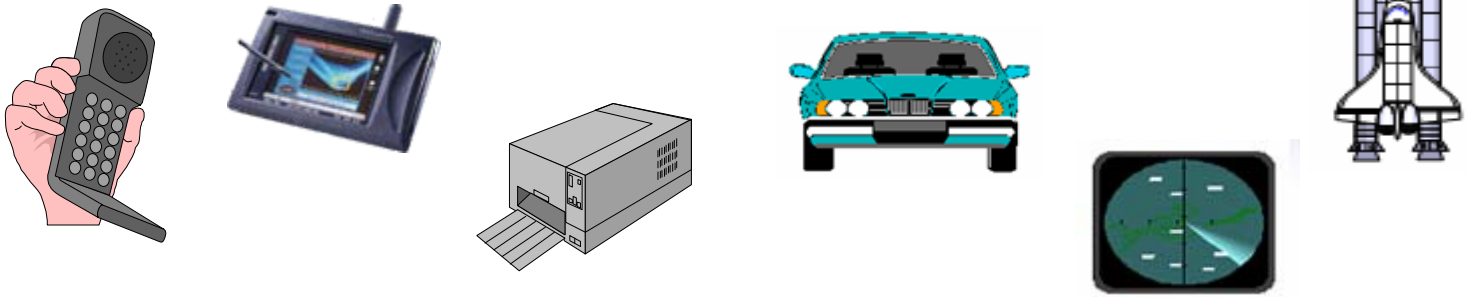


목 차

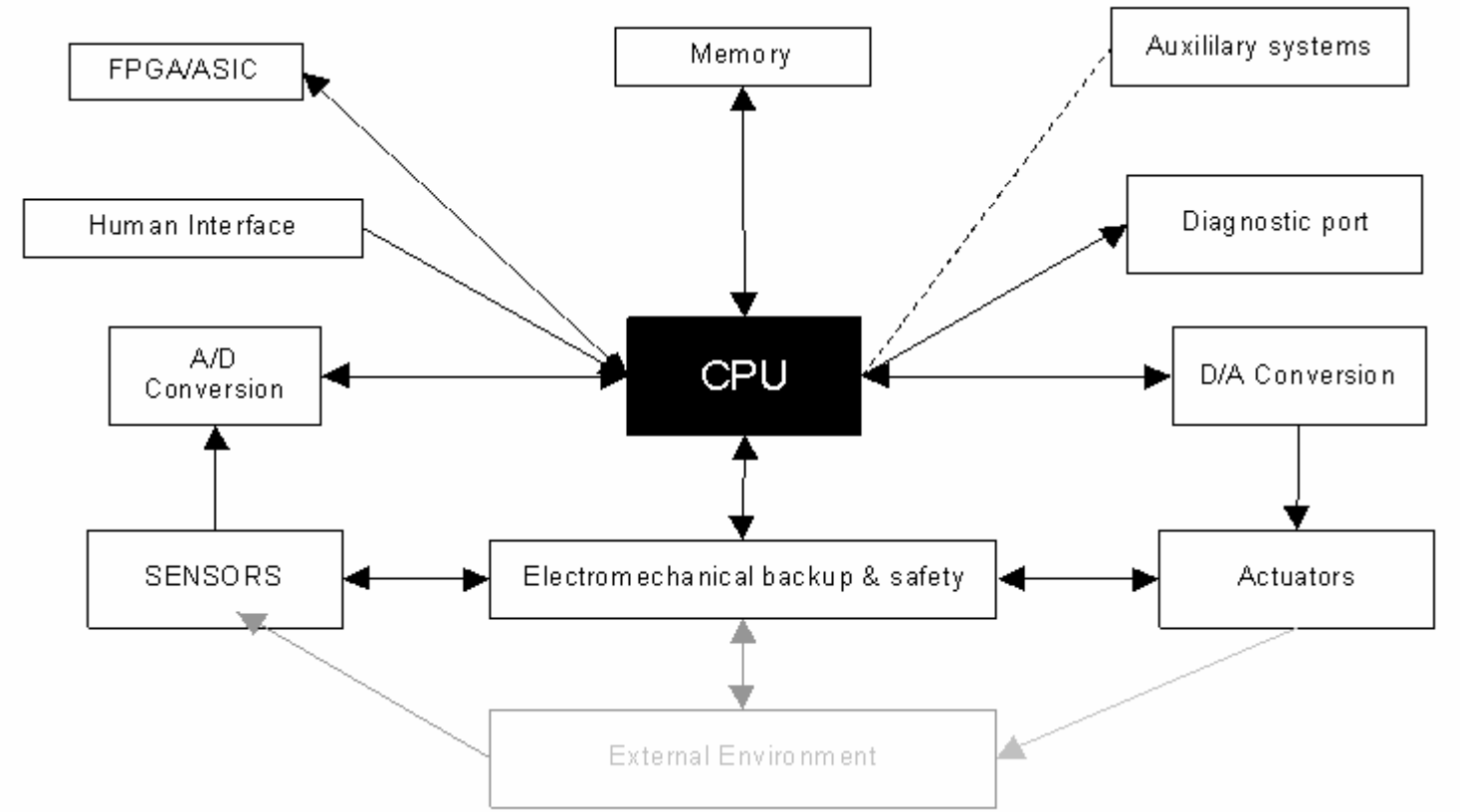
임베디드 시스템 개요

임베디드 시스템

- a computer system that is placed inside a product
- 특정한 기능을 수행하기 위한 하드웨어와 소프트웨어의 조합
- Reactive와 Time-constraints 환경에서 동작
- 외부 환경을 모니터링 하거나, 상호 작용(interaction) 또는 제어하는 시스템



임베디드 시스템



Embedded Systems are quite diverse, no one statement describes them all

Embedded vs. Conventional S/W

논리적, 시간적 정확성

- 소프트웨어는 상대시간 또는 절대 시간을 만족시켜야 함.
- Hard real-time, Soft real-time, and Firm real-time

내재된 물리적 동시성

- 외부로부터 입력되는 시그널(입력)의 동시성
- 시스템 설계의 복잡도를 증가시키는 요인

신뢰성 및 결함 허용 이슈

- 결함에 대처하기 위한 높은 신뢰성이 요구
- Self-recoverability / Home Property

Embedded vs. Conventional S/W

도메인 또는 응용 중심

- Application-specific software
- Stand-alone
- HMI (Human-Machine Interface) 설계가 중요

테스팅 및 검증의 어려움

- 시뮬레이터 및 다양한 디버거 이용
- 정형적인 명세 및 검증 기술

임베디드 시스템 설계 이슈

왜 임베디드 소프트웨어는 테스크탑 소프트웨어의 설계 방법과 다른 것인가 ?

임베디드 컴퓨터를 설계하는 것이 아니라 임베디드 시스템을 설계하는 것이다.

이슈들(Issues)

- 컴퓨터 설계 (Computer design)
- 시스템 설계 (System design)
- 수명주기 지원 (Lifecycle support)
- 비즈니스 모델 (Business model)
- 설계 문화 (Design culture)

이슈 : 컴퓨터 설계

Real-time and reactive operation

- Hard, Soft, or Firm real-time
- 동시성을 가질 수 있는 외부 이벤트에 대한 응답처리

Small size, low weight

- 외관의 형태와 에너지 요구에 따라 결정됨

Safety and reliability

- 결함의 발생에 의한 치명성
- 다중 시스템 구조 또는 분산 처리 컴퓨팅

Harsh environment

- 열, 진동, 충격, 물(water) 등에 의한 기능 손실

Cost sensitivity

- 개발 비용에 의존적인 컴퓨터 시스템 설계

이슈 : 시스템 레벨 설계

End-product utility

- 사용된 CPU가 무엇인가가 아니라 제공되는 기능이 어떠한가?
- 소프트웨어에 의해 유용성이 결정됨

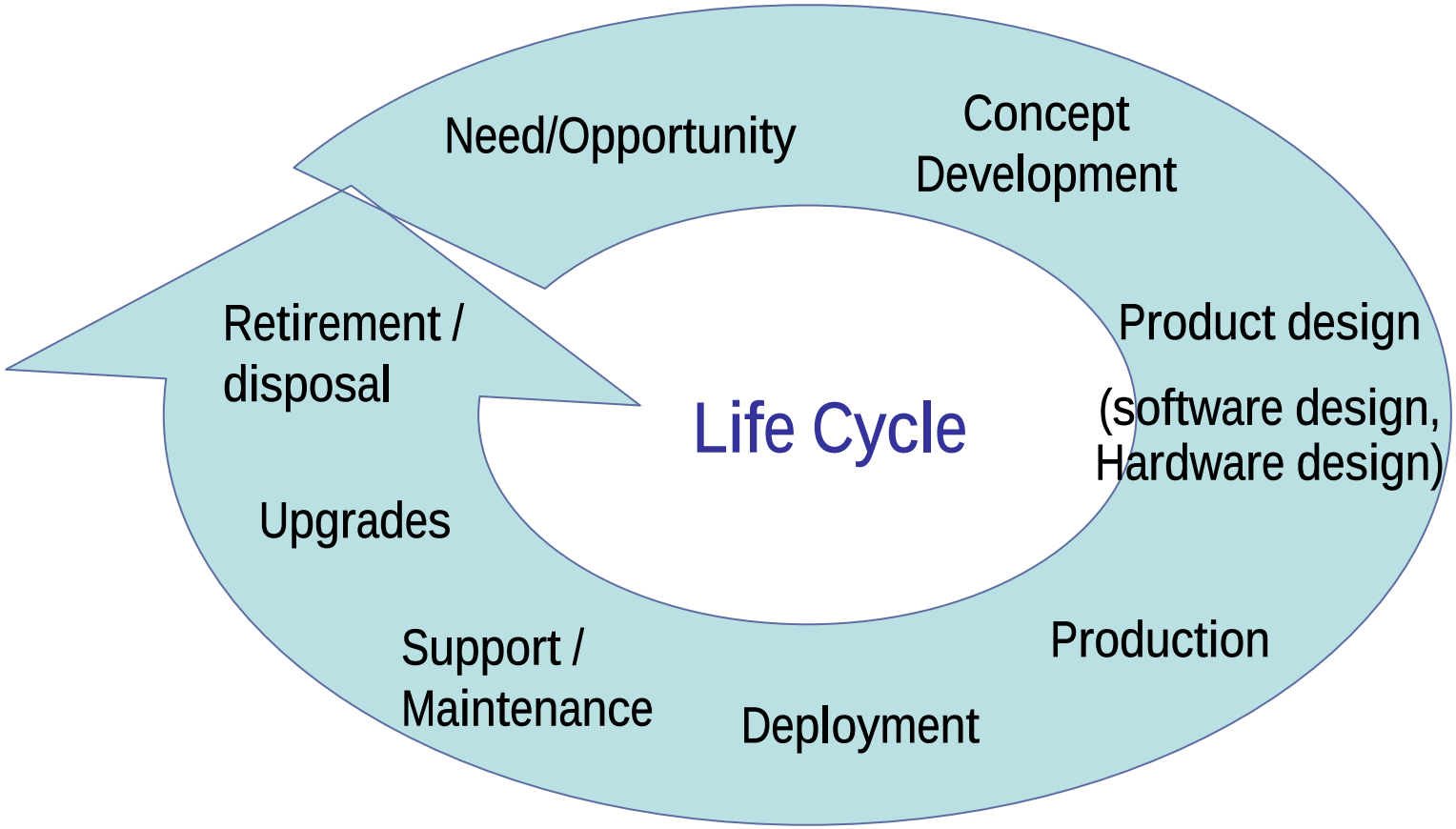
System safety and reliability

- 시스템 전체적인 이슈
  - 하드웨어 : Mechanical 방법에 의해 셧 다운 동작
  - 소프트웨어 : 인터럽트 또는 예외 처리

Control of physical systems

- 주변 장치들에 대한 제어 및 상호 작용

이슈 : 수명주기 지원 (1)


Need/Opportunity
Concept Development
Product design
(software design, Hardware design)
Production
Deployment
Support / Maintenance
Upgrades
Retirement / disposal
Life Cycle

이슈 : 수명주기 지원 (2)

컴포넌트 획득 및 사용

- 컴포넌트 재사용을 통한 개발 및 비용 효율성 증대

시스템 인증 및 확인

- 변경 발생에 대한 확인 및 검토
- 산출물 확인을 통한 설계 품질의 개선

유지보수 지원

- Repairability and Evolvability
- 로그 (log)에 대한 기록 및 수집

개선 및 가용성

- 문서화 되지 못한(undocumented) 행위에 대한 설명
- 특성화된 컴포넌트의 가용성 확보

이슈 : 비즈니스 모델

설계 및 양산 비용

- 소량의 시스템 생산 : 설계 비용의 최소화
- 다량의 시스템 생산 : 양산 비용의 최소화

Cycle Time

- 설계 사이클을 결정하는 요인들에 대한 고려가 필요
  - 개발 절차(반복 개발), 변경 반영 절차 등

Product Families

- 유사한 동종의 소프트웨어를 개발하기 위한 전략
- 범용 솔루션과 최적화된 설계간의 Trade-Off

이슈 : 설계 문화

설계 문화의 차이점

- 컴퓨터 소프트웨어 문화 : 설계 도구 >> 프로토타입 시뮬레이션
- Mechanical 설계 문화 : 프로토타입 >> 선행 분석 설계

하드웨어 엔지니어 vs. 소프트웨어 엔지니어

Hardware	Software
• Timing : a system clock signal	• Timing : logical time unit
• bitwise & continuous data flow	• bitwise & discrete data flow
• serial data streams	• parallel data streams
• a pulse	• a bit
• a signal	• data
• hardwired design	• CTRL + ALT + DEL

모델 기반 임베디드 소프트웨어 개발 방법론

대표적인 개발 방법론(1)

1. CODART/RTSA 방법론 (H. Gomaa)
   - COncurrent Design Approach for Real-Time system
2. PeaCE 기반 방법론 (S. Ha)
   - Ptolemy Extension As Codesign Environment
3. COMET 방법론 (H. Gomaa)
   - Concurrent Object Modeling & architecture design mEThod)
4. OCTOPUS 방법론 (M. Awad)
   - OMT (Object Modeling Technique) + Fusion
5. BridgePoint 방법론 (S. Mellor)
   - Excutable and Translatable UML (xtUML)

대표적인 개발 방법론 (2)

6. ROPES 방법론, Real-Time UML (B. Douglass)
   - Rapid Object-oriented Process
7. MoBIES 방법론 (R. Alur)
   - Model-Based Integration of Embedded Software
8. MaRMI IV 방법론(ETRI)
   - Embedded software development based on product lines
9. OMEGA 방법론 (S. Graf)
   - UML based modeling of real-time embedded systems
10. ESUML 방법론 (Hong & Bae)
   - Embedded Software modeling with UML2.0

ESUML 모델링 방법론

ESUML : Principles

Embedded Software modeling with UML 2.0
An Embedded Software Development Methodology

Concepts

Real-Time Embedded Software
Interaction-based Behavior Modeling
Light-Weight Methodology
Model-driven Approach

Process

Requirement Capturing
PIM and PSM modeling
Verification & Validation
Code Generation

ESUML

Notations

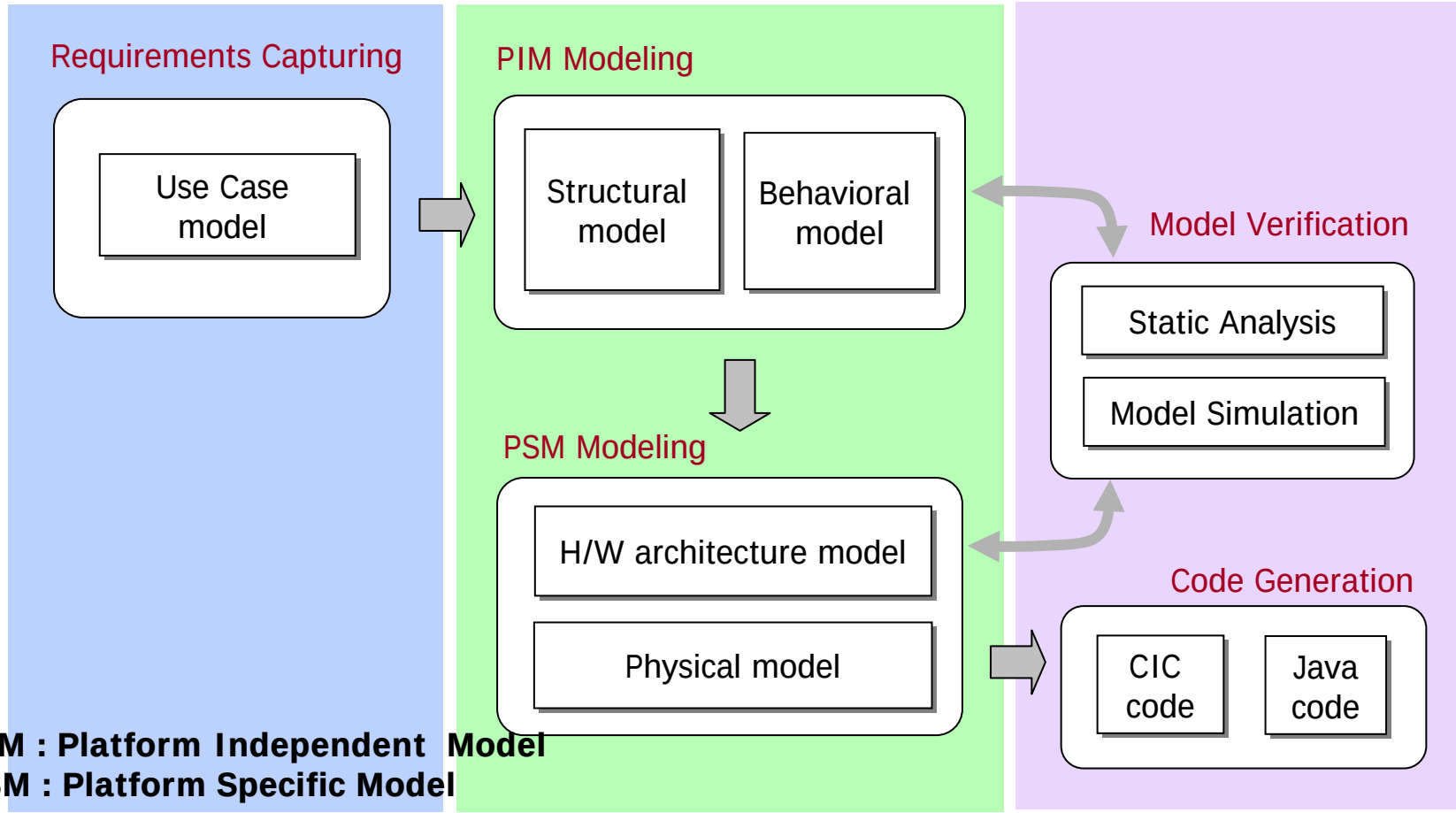
Use Case Diagram
Class Diagram
Interaction Overview Diagram
Sequence Diagram
Hardware Architecture Diagram

Rules

Event-Based Modeling
Hierarchical Decomposition
Modeling Guidelines and Tips

ESUML : Overall Process

Model-Driven Application Development



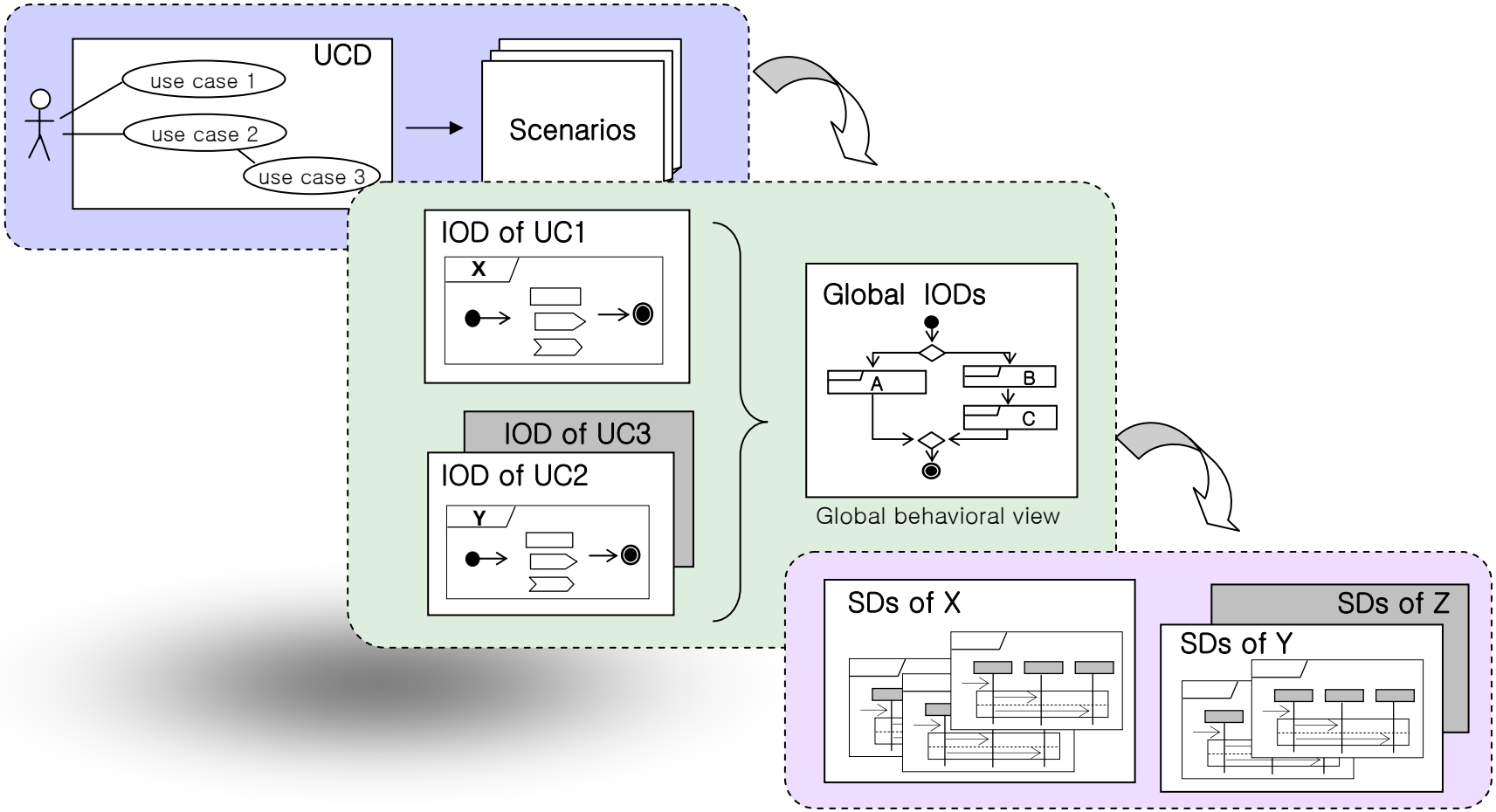
ESUML : Artifacts

개발 단계별 산출물

단계	산출물	설 명
Requirements Capturing	Use case diagram Use case sheet	요구사항 추출 및 정의
PIM Modeling	Class diagram Operation sheet Interaction overview diagram Sequence diagram	소프트웨어 정적 구조 및 동적 행위 모델링
PSM Modeling	Hardware architecture diagram Physical software model	하드웨어 플랫폼 정의 및 물리적 소프트웨어 모델
Model Verification	Verified models	정적 분석과 시뮬레이션을 통한 모델 분석 및 검증
Code Generation	Java, Common Intermediate Code	타겟 / 중간 코드 생성

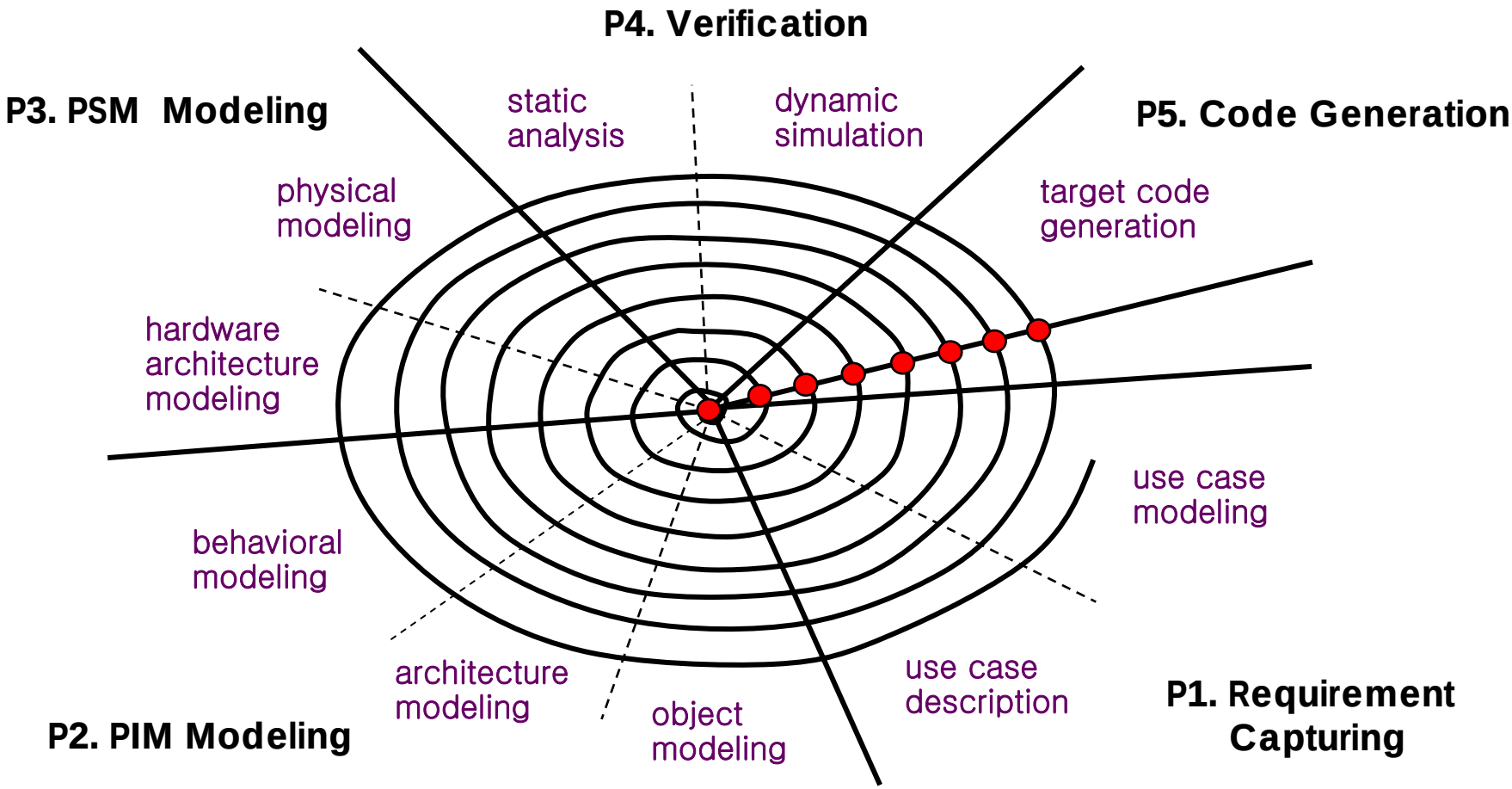
ESUML : Modeling Approach

Interaction-based Behavior Modeling



ESUML : Modeling Approach

Iterative Modeling Approach



P1. 요구사항 획득단계

요구사항 명세

- 개발하고자 하는 소프트웨어 제품에 대한 기술
  - Functional requirements
  - Non-functional requirements
- 개발 대상 시스템의 소프트웨어 경계 설정

요구사항 명세를 위한 주요 수단

- use case diagram
  - 시스템과 시스템 외부와의 상호 작용을 표현하는 방법
  - 시스템 외부에서 가시적인 행위들에 대한 표현
- use case sheets & event table

Use Case 와 Actor

Use Case

- 주어진 목적(goal)을 만족하기 위해 시스템을 사용하는 방법

Starting point: 시스템과 상호작용하는 외부의 에이전트

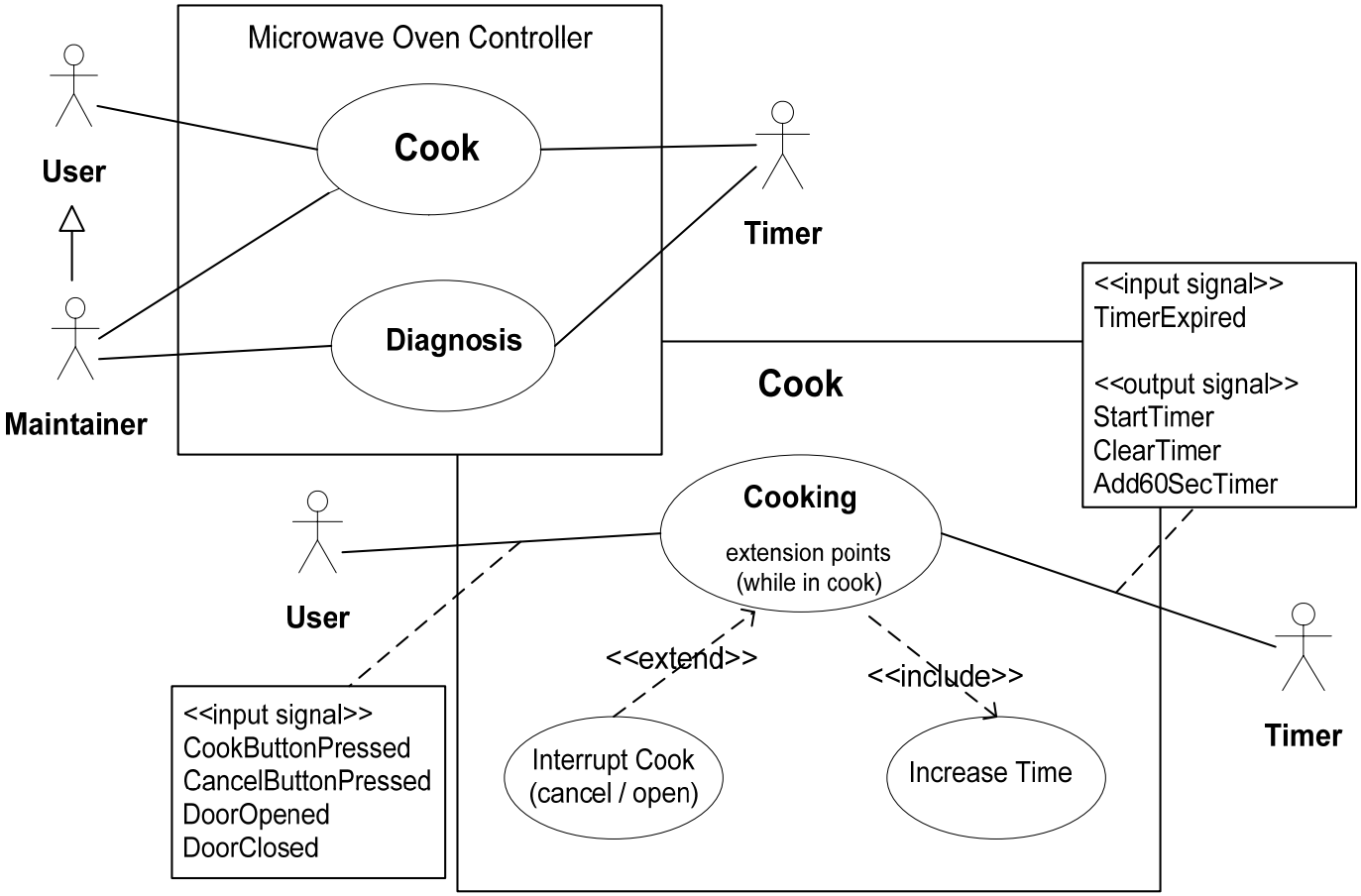
- 에이전트(agent) : 다수의 역할을 가질 수 있음
- 에이전트 + 역할 = 액터 (actor)

Actors

- Active: initiate a use case
- Passive: participate but do not initiate
- Client: use the system for a certain purpose
- Nonclient: just affect the system
- Primary: use the system
- Secondary: exist so that the primary actors can use the system

Use Case diagram

Event-Based Requirements Capturing



Use Case Sheet

Use Case를 문서화하기 위해 사용하는 틀(template)

Use Case	(U1) Cook	
UC 설명	...	
참여 액터	User, Timer	
선행 조건	전원인가 및 정상 초기화 상태	
시나리오	1. 사용자가 오븐의 문을 연다.	DoorOpened
	2. 오븐 내부의 램프가 점등한다.	
	3. 사용자가 오븐의 문을 닫는다.	DoorClosed
	4. 사용자가 Cook 버튼을 누른다.	CookButtonPressed
	5. Beep	
	6.	
	:	
	14. 램프가 점멸하고 3번의 알림 신호를 발생한다.	
예외 상황	U1.2	
후행 조건	정상 초기화 상태	

Event Table

시스템과 액터간에 존재하는 입출력 이벤트의 정의

	DoorOpened
Intent	Notify that the oven door is opened
Direction	input
Content	True, False
Cycle	Episodic
Response Time	N/A

	CookButtonPressed
Intent	Notify that the Cook button is pressed
Direction	input
Content	True, False
Cycle	Episodic
Response Time	N/A

P2. PIM 모델링 단계

1. Object Modeling

소프트웨어의 정적 구조를 모델링

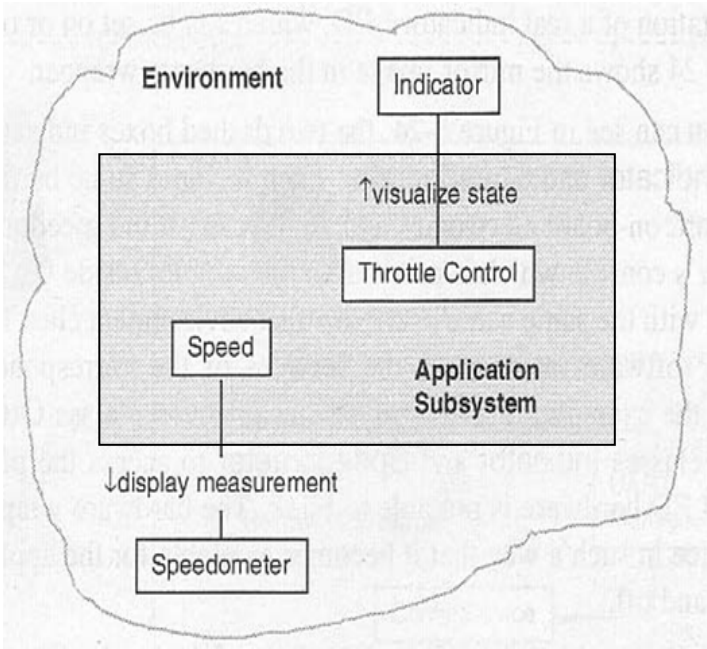
Objects / Classes 의 식별

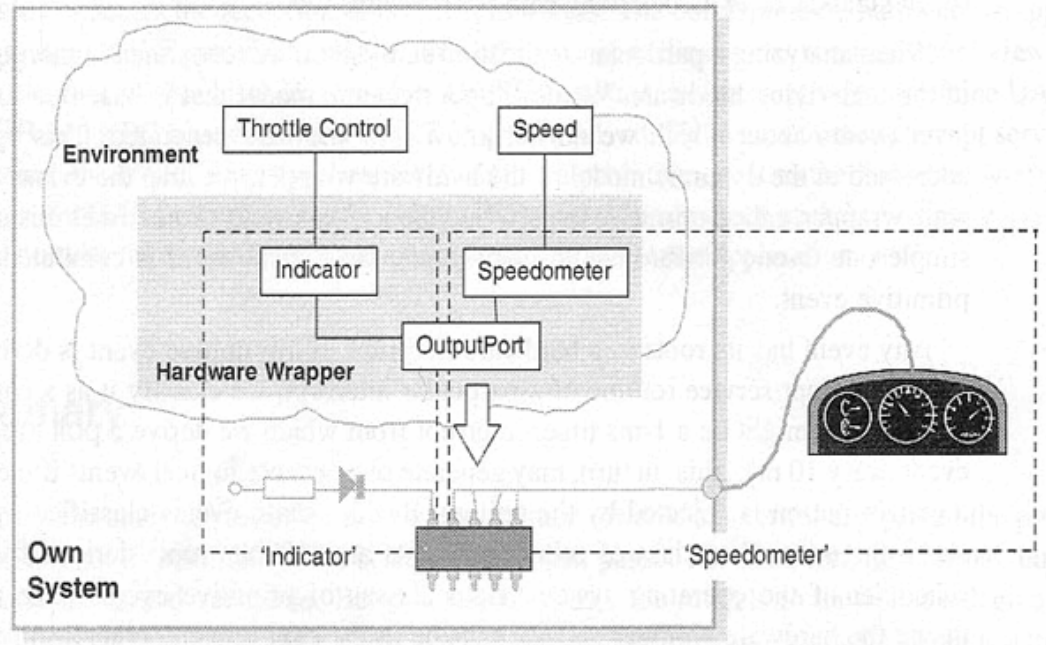
- 요구사항에 포함된 모든 명사 또는 명사구
- 일반적인 에이전트
  - for example, floor, elevator, door, button, request, etc
- 수동적 제어나 데이터 관리와 같은 서비스
- 물리적인 장치들
- 도메인에서 식별되는 추상화된 개체
  - window, scroll bar, cursor, icon, message, packet, etc
- 트랜잭션 분석을 통한 객체 식별

하드웨어 랩퍼 클래스

Wrapper Class

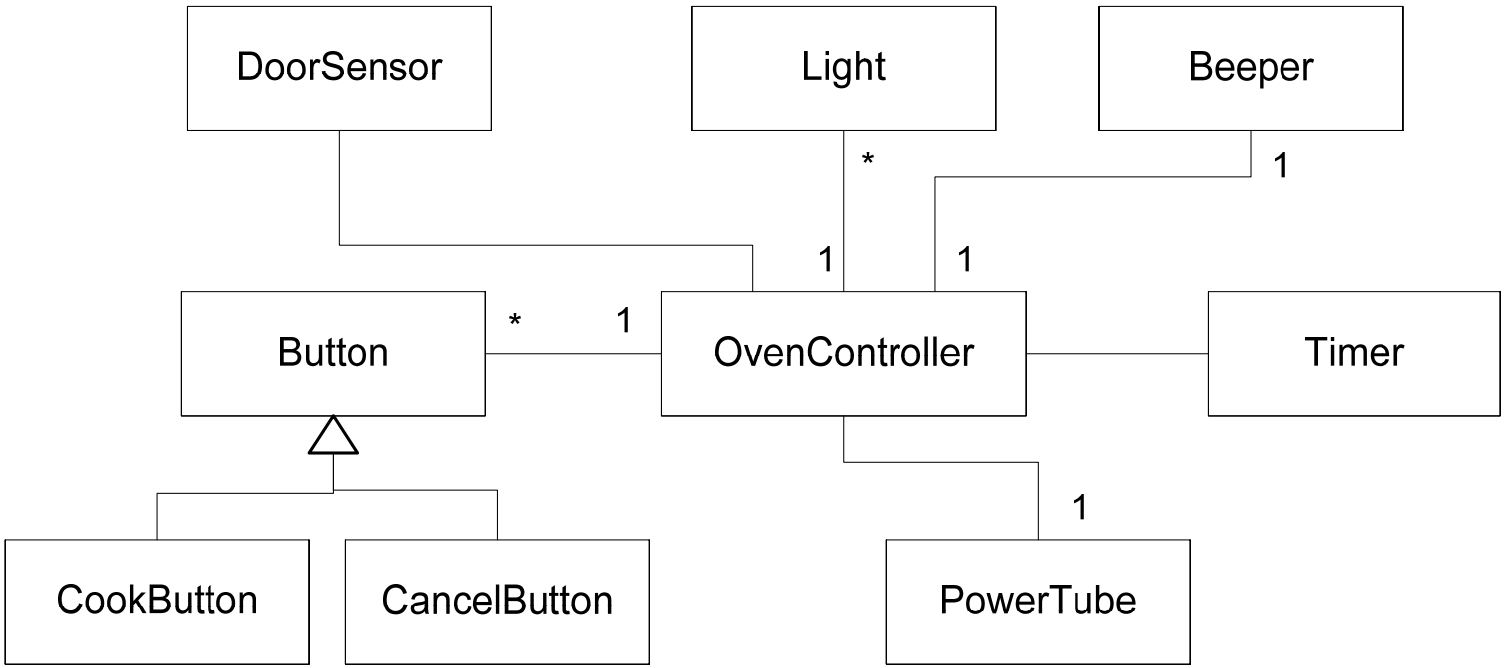
- 외부에 존재하며, 소프트웨어와 상호작용하는 컴포넌트
- 하드웨어 컴포넌트의 동작과 소프트웨어 동작을 식별





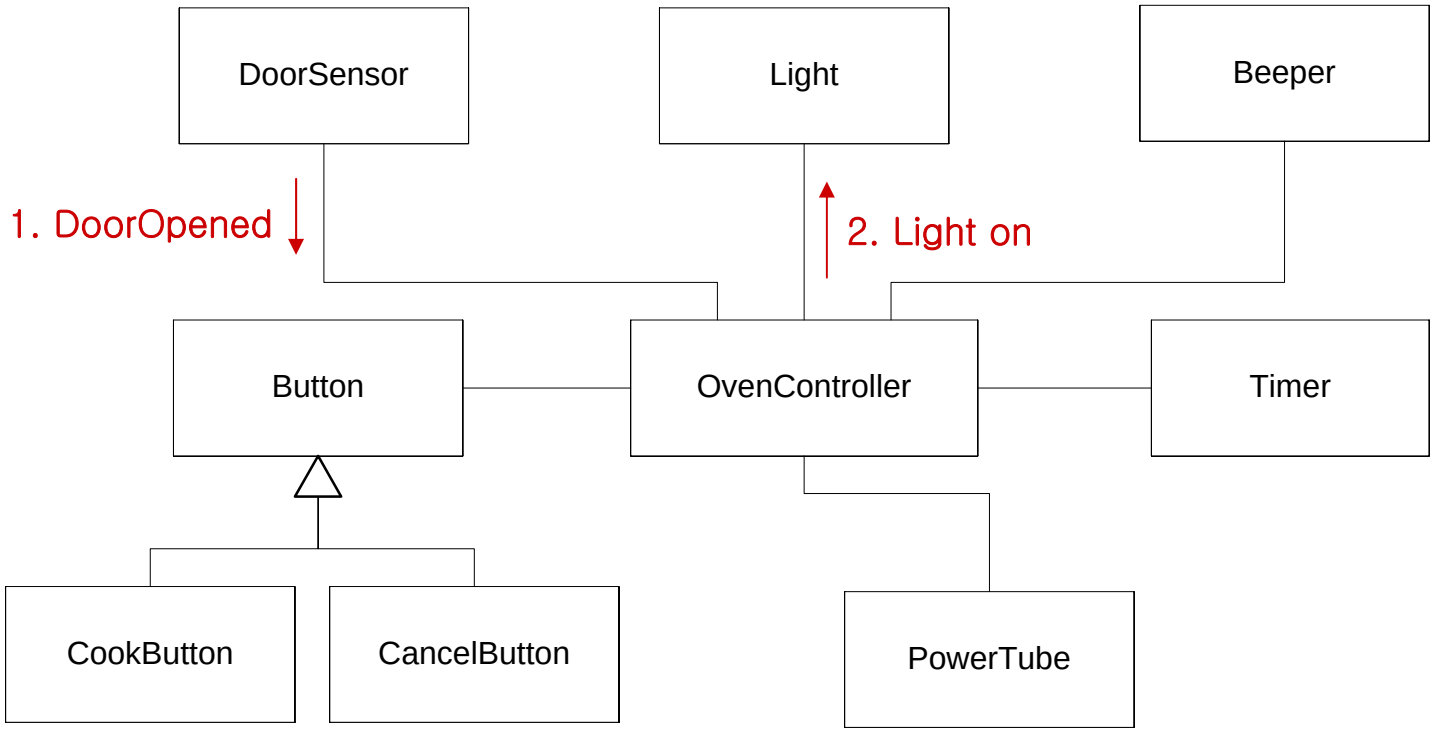
Object modeling (1)

클래스 다이어그램 : Microwave Oven 예제



Object modeling (2)

Object Diagram : 클래스 다이어그램의 인스턴스



Operations sheet

클래스(객체) 연산 설명서

- 주어진 입력에 대하여 객체가 수행하는 함수
- 추후 행위 모델링의 근거 자료

연산	button input handling
설명	버튼으로부터 입력되는 이벤트에 대한 처리를 담당한다.
관련 객체	OvenController
입력	CookButton, CancelButton
출력	CookBtnPressed, CancelBtnPressd
처리 로직	[CookBtnPressed] Timer += 60; PowertubeOn = true; LightOn = true; :

2. Architecture Modeling

시스템 아키텍처

- 하드웨어 아키텍처와 소프트웨어 아키텍처로 구성

아키텍처 설계시 고려 사항

- 하드웨어 아키텍처 + 소프트웨어 아키텍처 : Hw/Sw codesign
- 일반적으로 하드웨어 아키텍처가 주어지며, 이는 소프트웨어의 제약 사항으로 고려됨
- 하드웨어 아키텍처와 소프트웨어 아키텍처가 분리되어 고려되어야 하나, 그렇지 못한 경우
  - 하드웨어 : 추상화(hardware wrapper)

소프트웨어 아키텍처

- 소프트웨어의 모듈화된 분할과 모듈간의 관련성을 정의한 형상

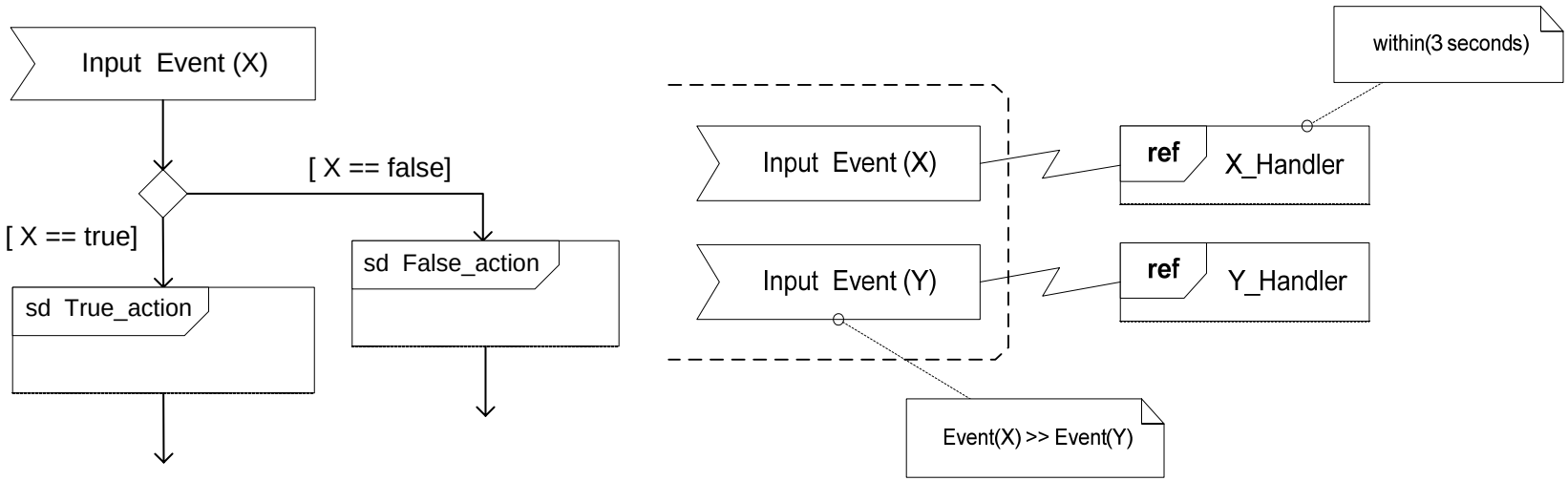
ESUML 아키텍처

Operational Architecture

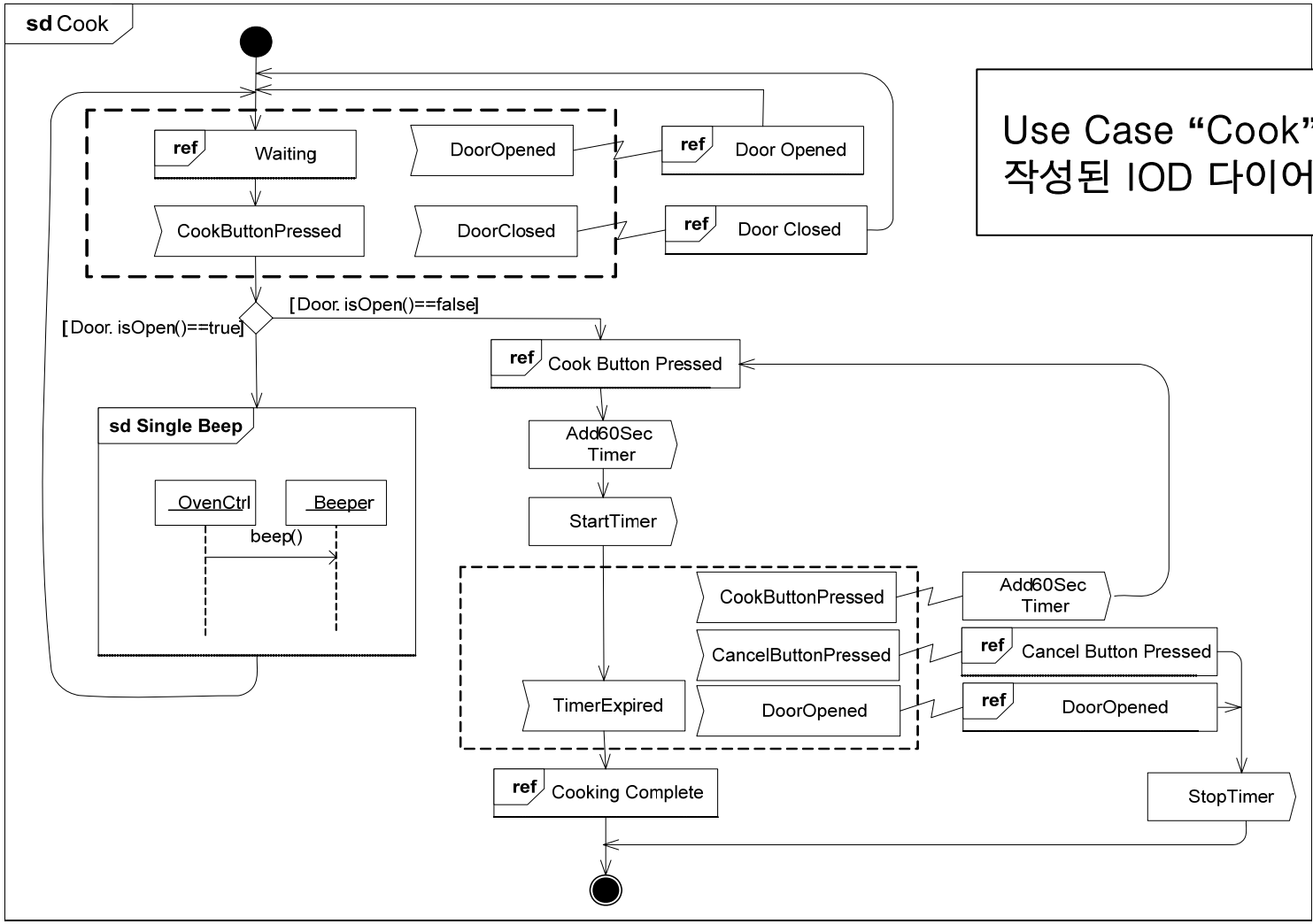
- 시스템의 동작 행위를 중심으로 모델링
- Use Case Sheet로부터 생성됨

표현 도구

- UML2.0, Interaction Overview Diagram (IOD)



ESUML 아키텍처 예제

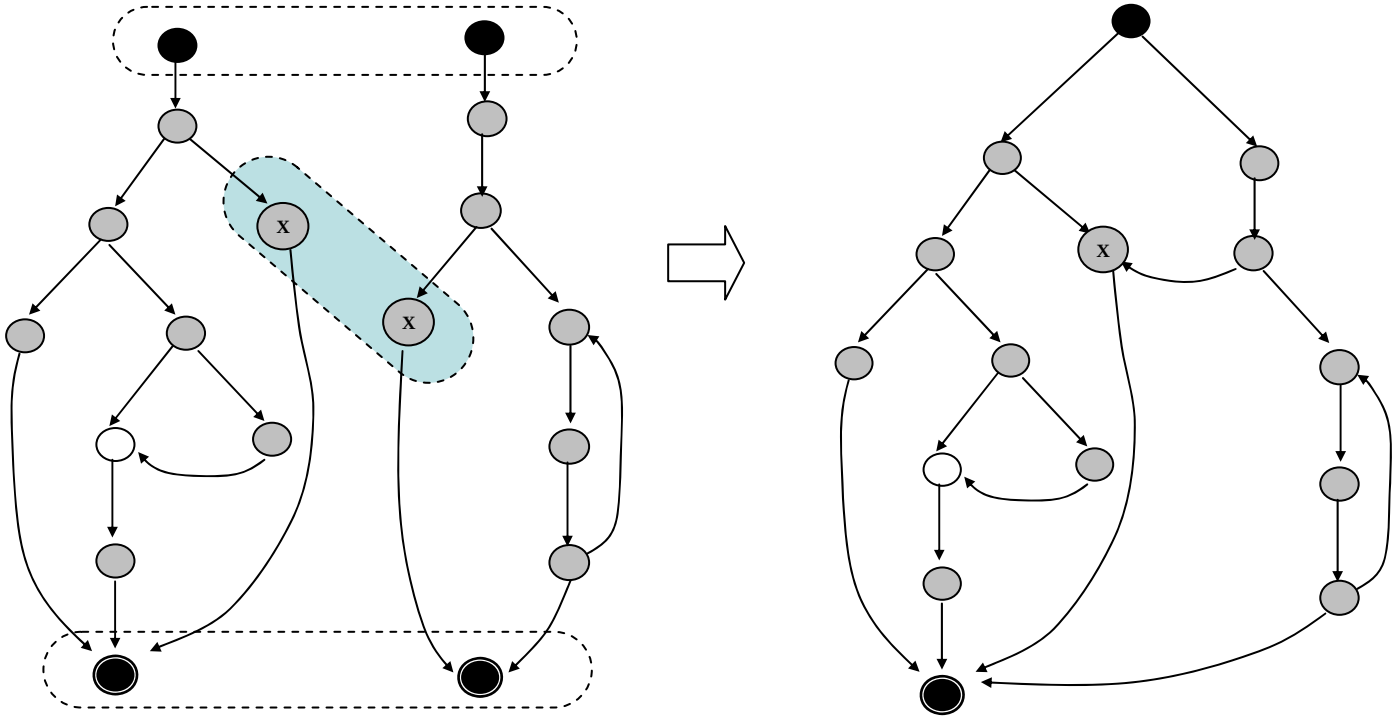


Use Case “Cook”으로부터 작성된 IOD 다이어그램

ESUML 아키텍처

System Global View

- Use case별 IOD 다이어그램의 합성
- 그래프 기반 시나리오 합성 방법 이용 [11,12]

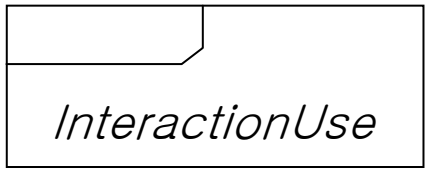


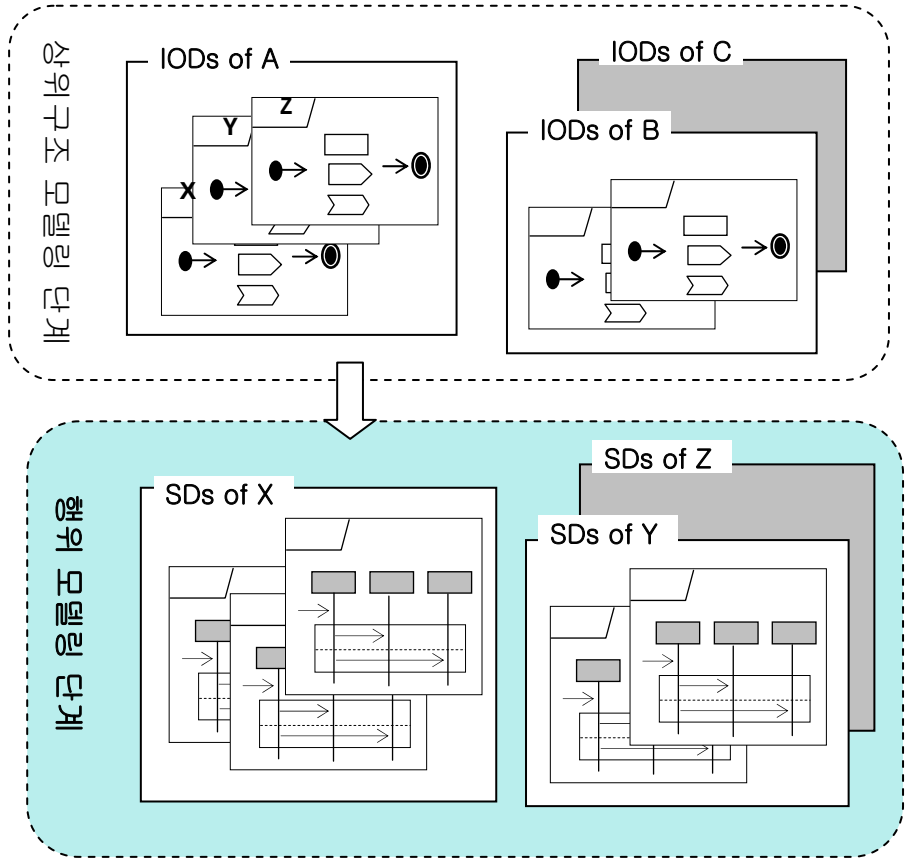
3. Behavioral Modeling

소프트웨어 행위의 모델링

- Sequence Diagram
- Action Language

상위 수준의 IOD에 나타난 “interactionUse”에 대하여 상세화

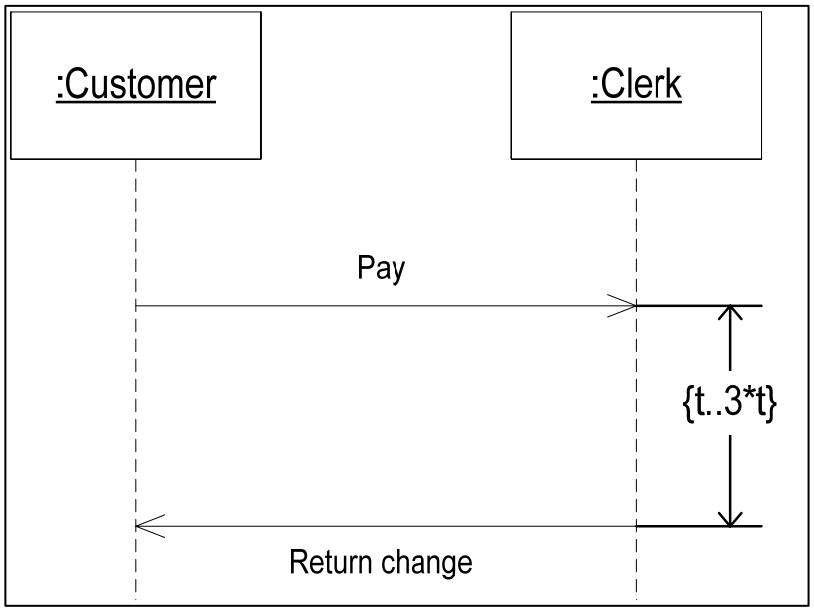




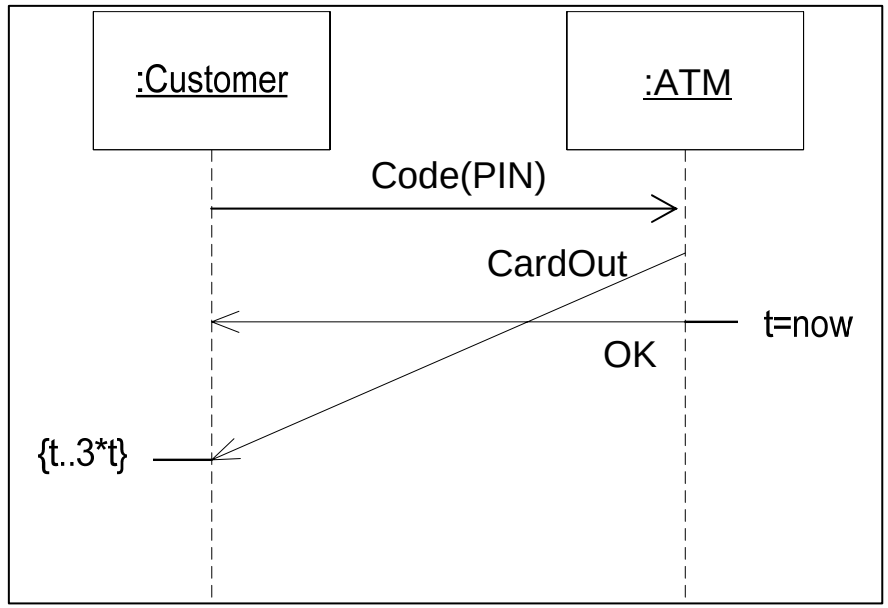
Behavioral Modeling(1)

Real-Time Constraints

Duration Constraints

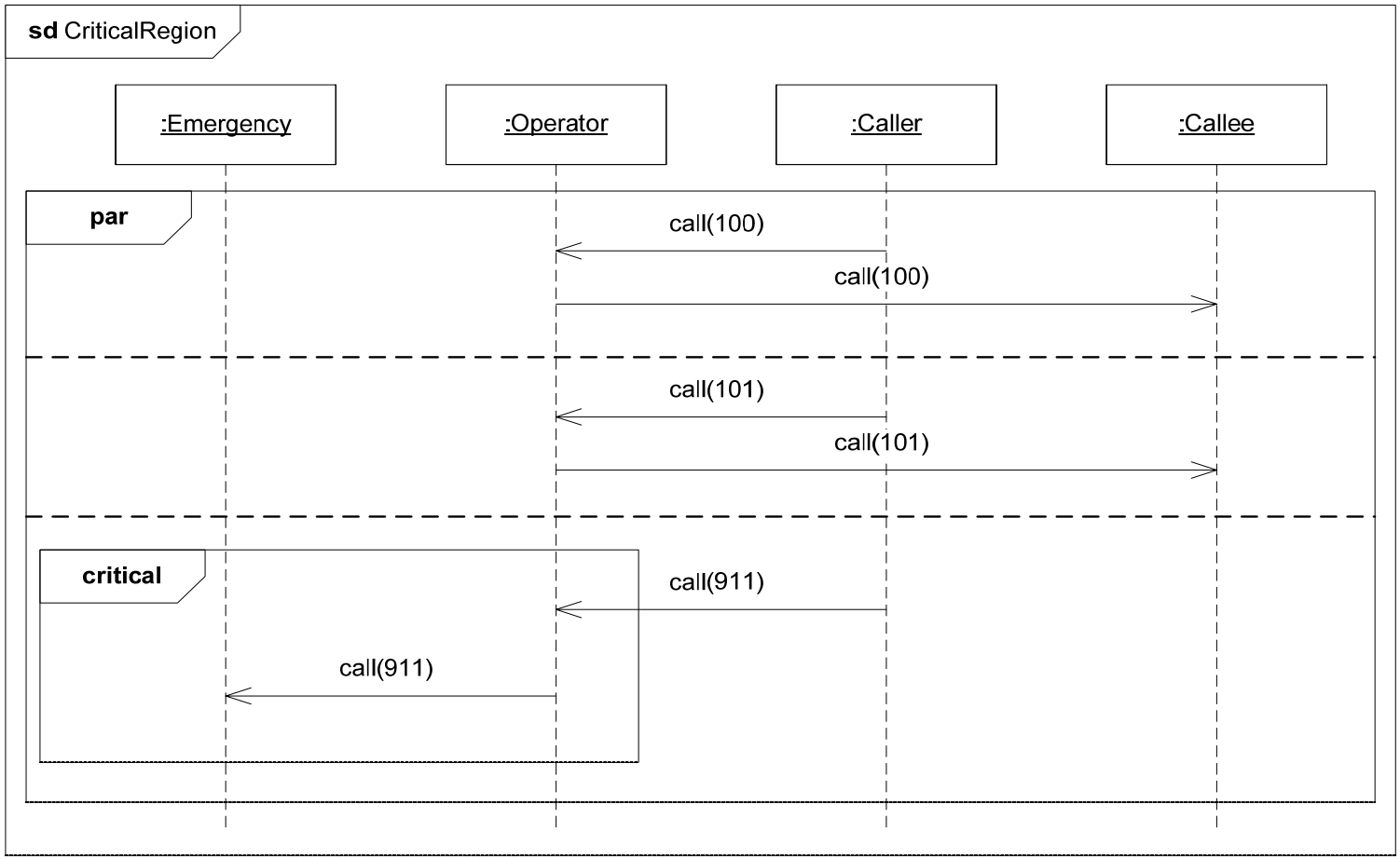


Time Constraints



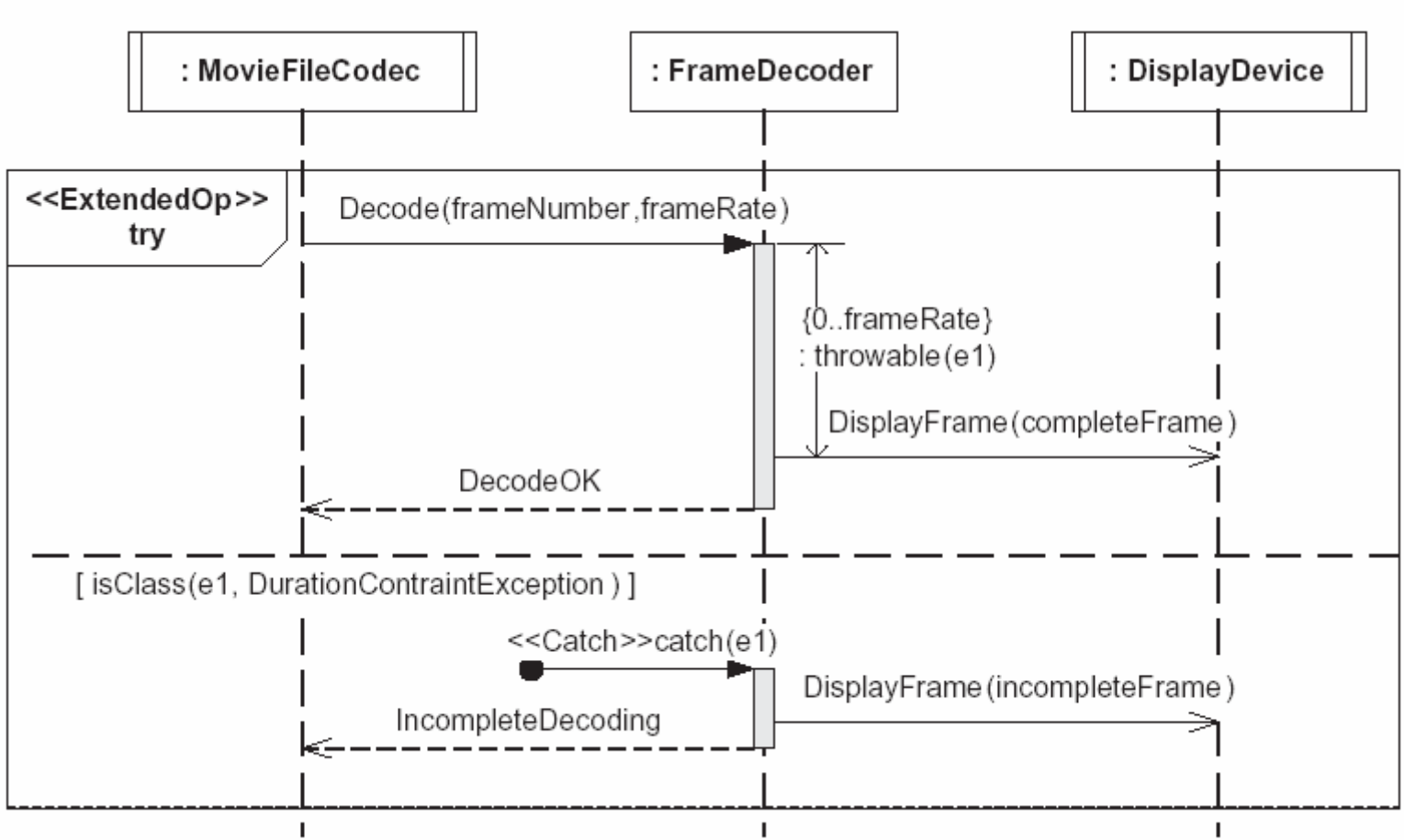
Behavioral Modeling(2)

Concurrency & Synchronization



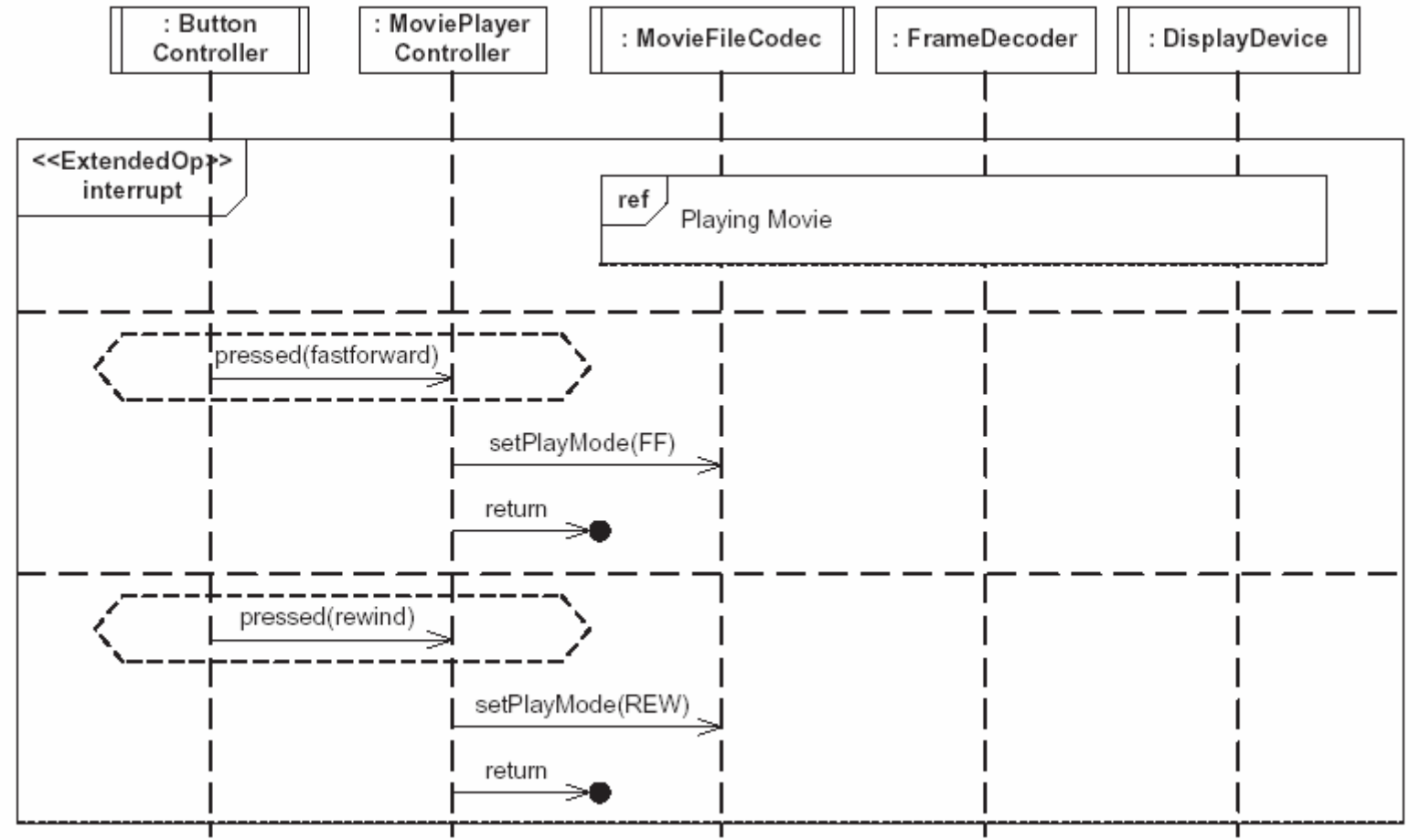
Behavioral Modeling(3)

Exception Handling



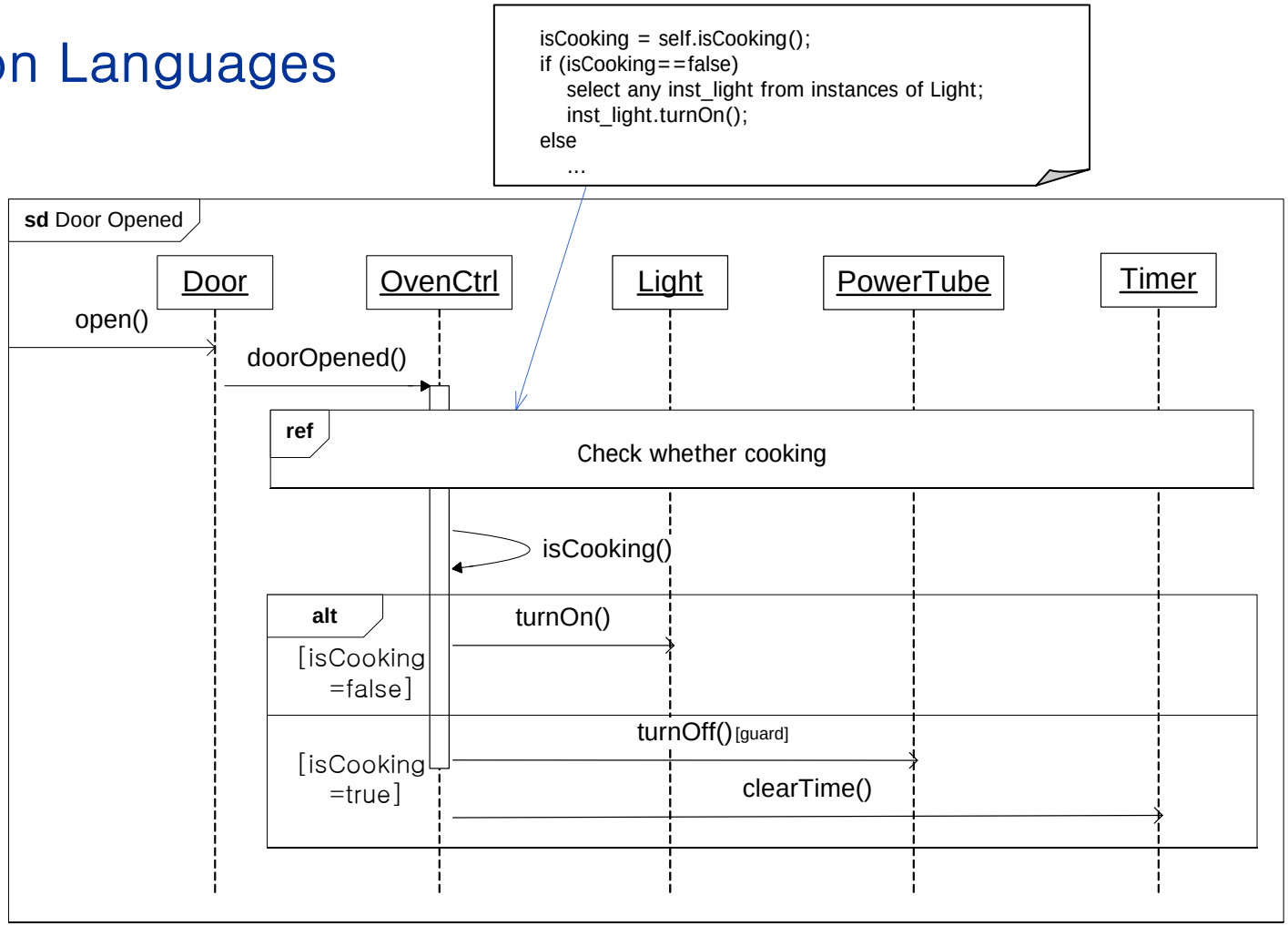
Behavioral Modeling(2)

Interrupt Handling



Behavioral Modeling(4)

Action Languages



P3. PSM 모델링 단계

Hardware architecture modeling

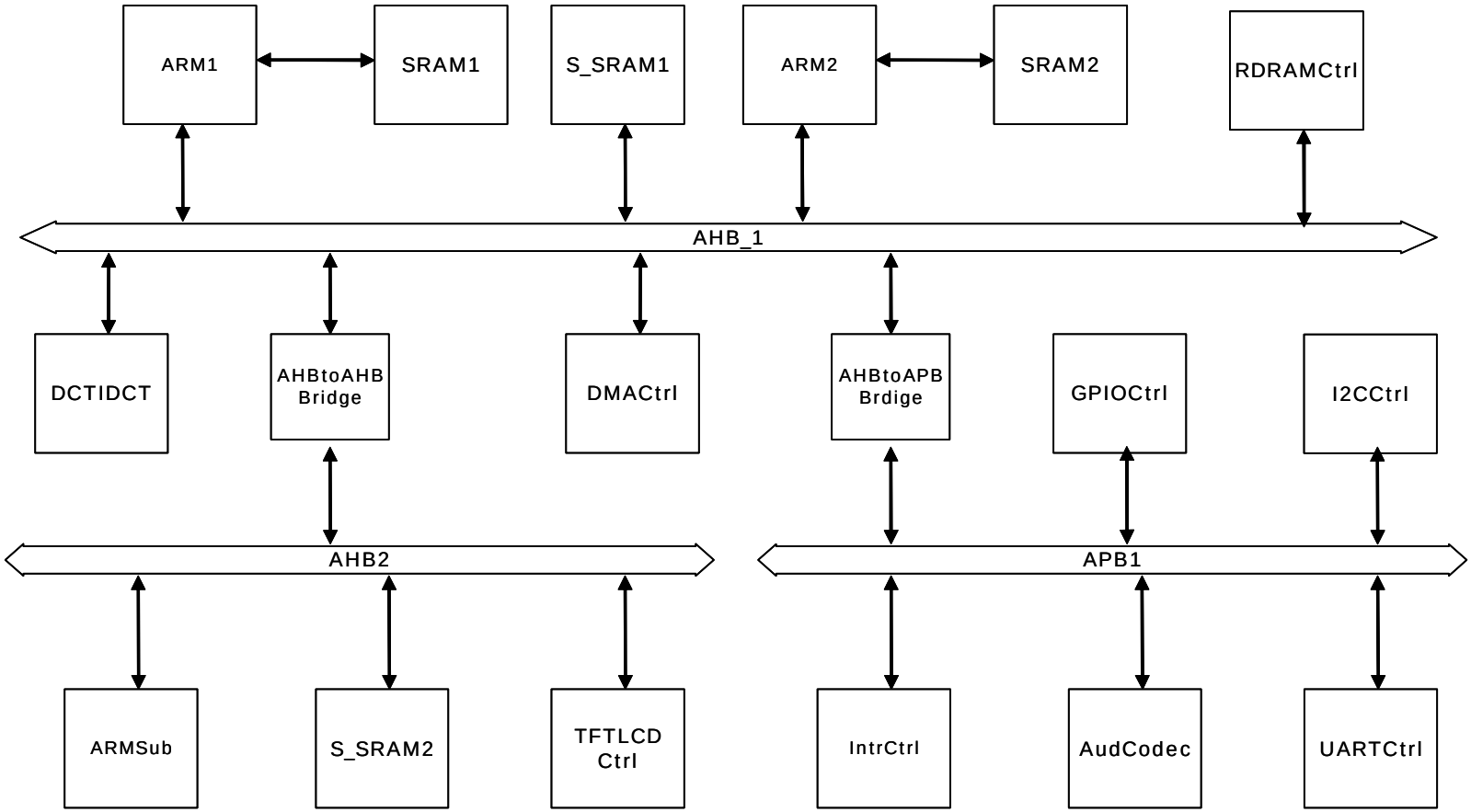
- 하드웨어를 구성하는 컴포넌트의 식별
- 하드웨어 컴포넌트에 대한 세부 사양 정의
- 컴포넌트간 상호 연결 관계 정의

상용 하드웨어 컴포넌트에 대한 명세 라이브러리 구축 활용

컴포넌트	속성변수	속성 값
프로세서, ARM 1	타입	ARM926EJS
	캐쉬	IDCache
	처리속도	600MHz
	운영체제(OS)	Velos
	캐쉬 사이즈	0x10000
	속도	400
	포트 1 타입 / 크기	AHB_M_Port / 32 bits
	:	

PSM 모델링 단계

Hardware architecture model : 예제



PSM 모델링 단계 (1)

Physical Software Model

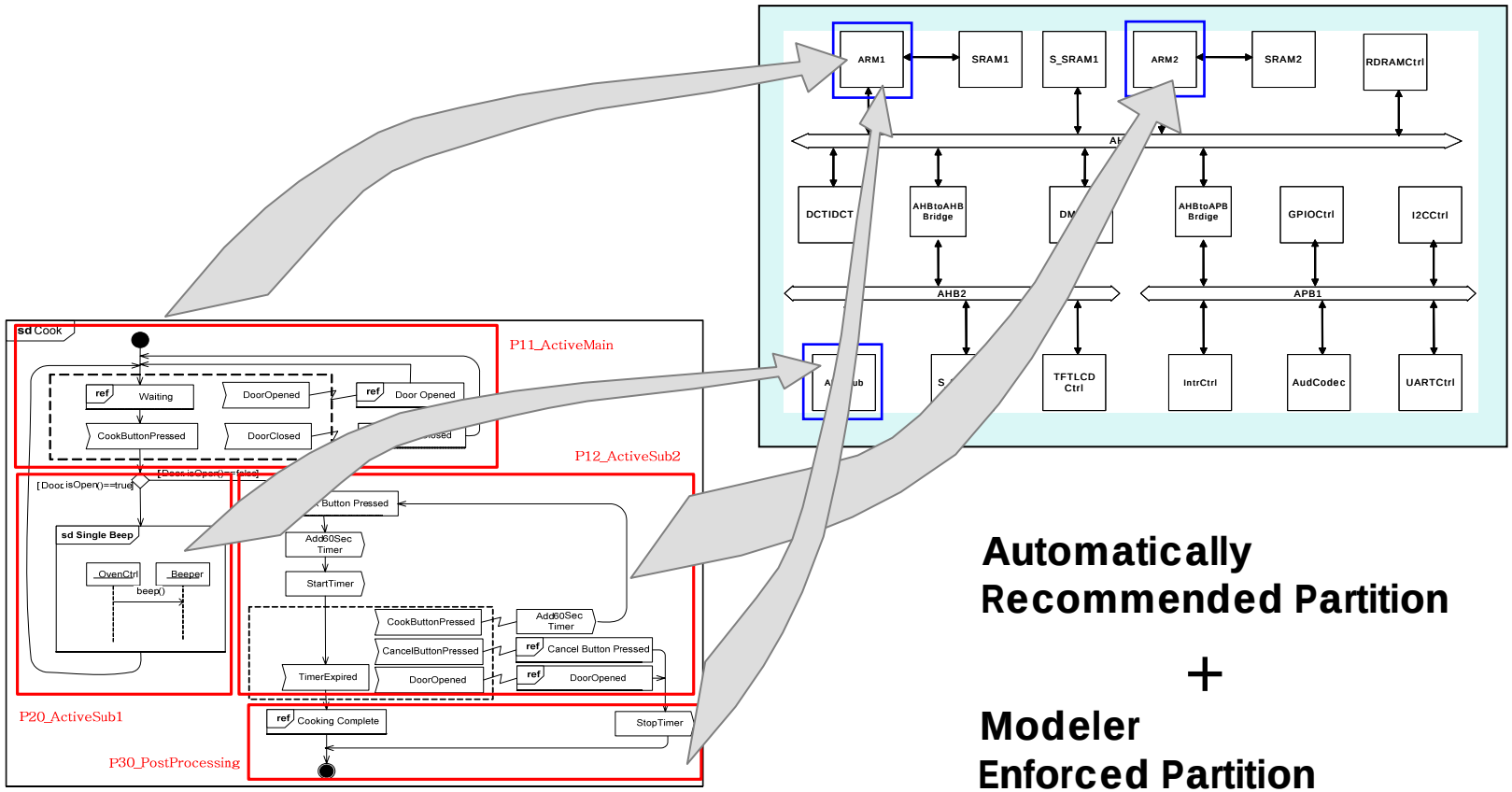
- 하드웨어 플랫폼에 의존적인 소프트웨어 모델 생성
- 소프트웨어 모델의 분할 과정을 통해 생성됨

소프트웨어 모델의 분할

- 입력 : IOD 다이어그램, Sequence Diagram
- 분할 방법
  - 자동적으로 분할된 물리적 모델
    : 모델에 나타난 병렬 식별자 : Fork/Join, Par fragment
  - 모델러에 의한 수동 분할
    : 프로세서 수를 고려하는 소프트웨어 태스크 분리

PSM 모델링 단계 (2)

컴포넌트 매핑 : 소프트웨어 컴포넌트 vs. 하드웨어 컴포넌트

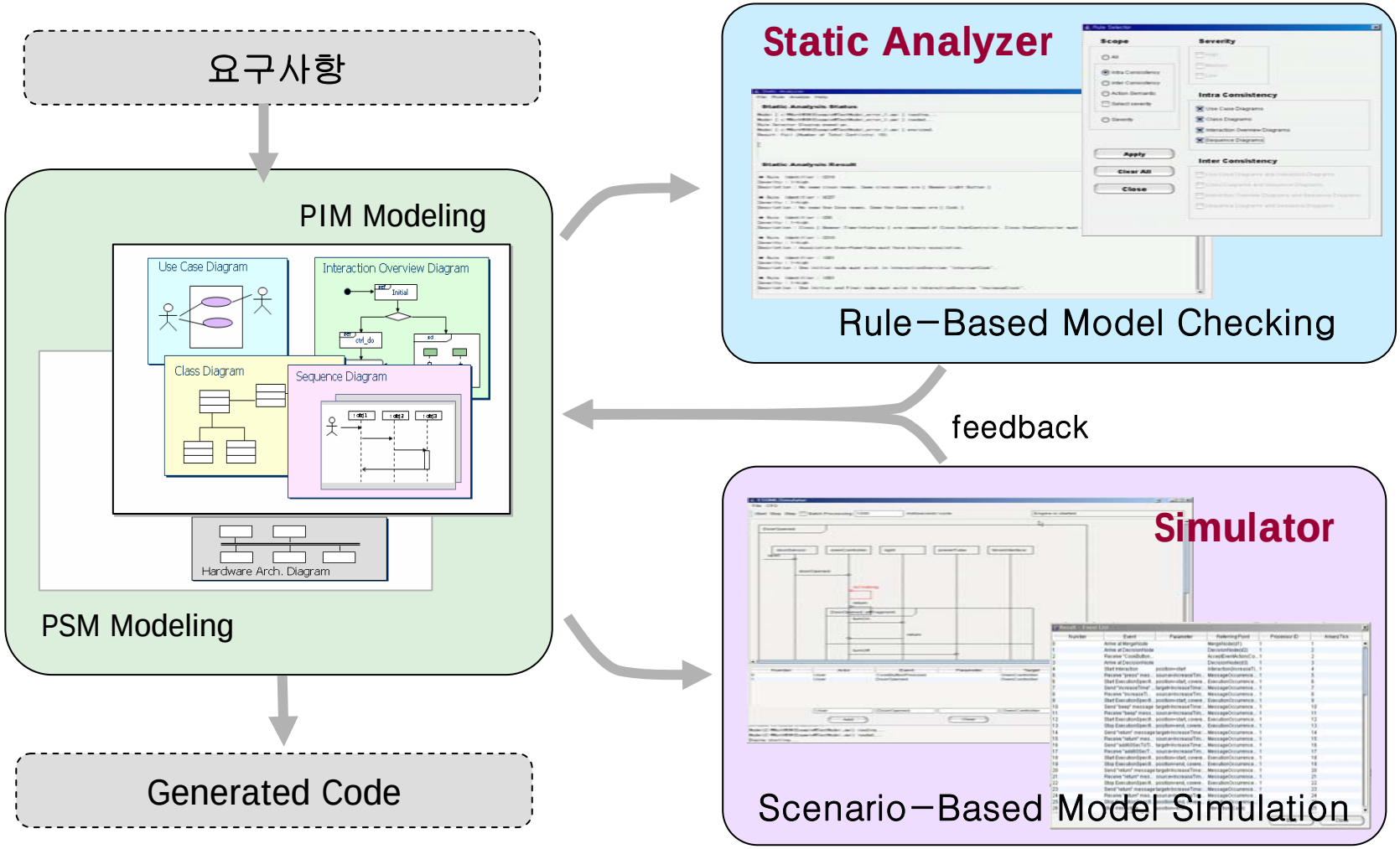


Automatically Recommended Partition

+

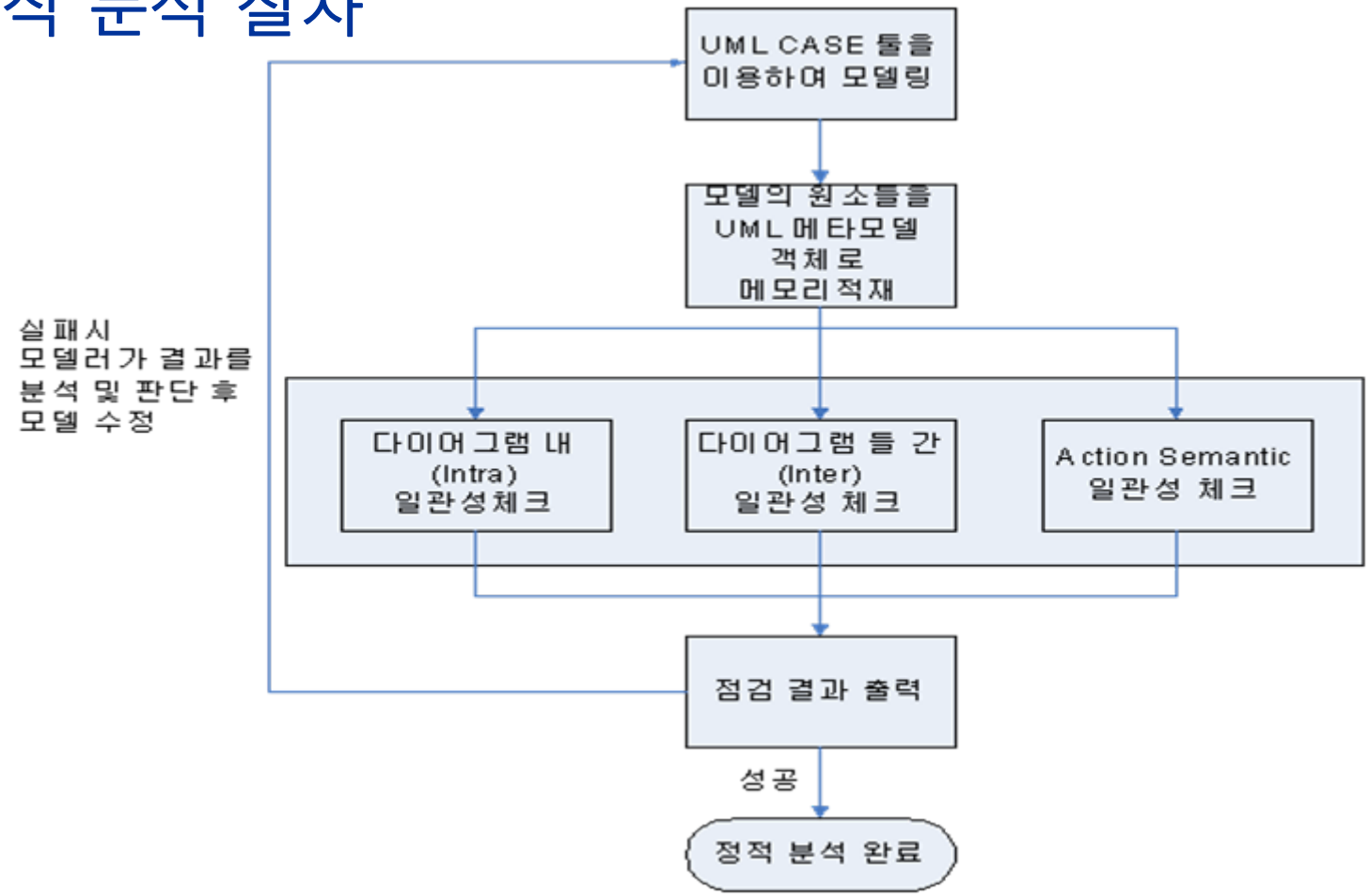
Modeler Enforced Partition

P4. 검증 단계


요구사항
PIM Modeling
Use Case Diagram
Interaction Overview Diagram
Class Diagram
Sequence Diagram
Hardware Arch. Diagram
PSM Modeling
Generated Code
Static Analyzer
Rule-Based Model Checking
feedback
Simulator
Scenario-Based Model Simulation

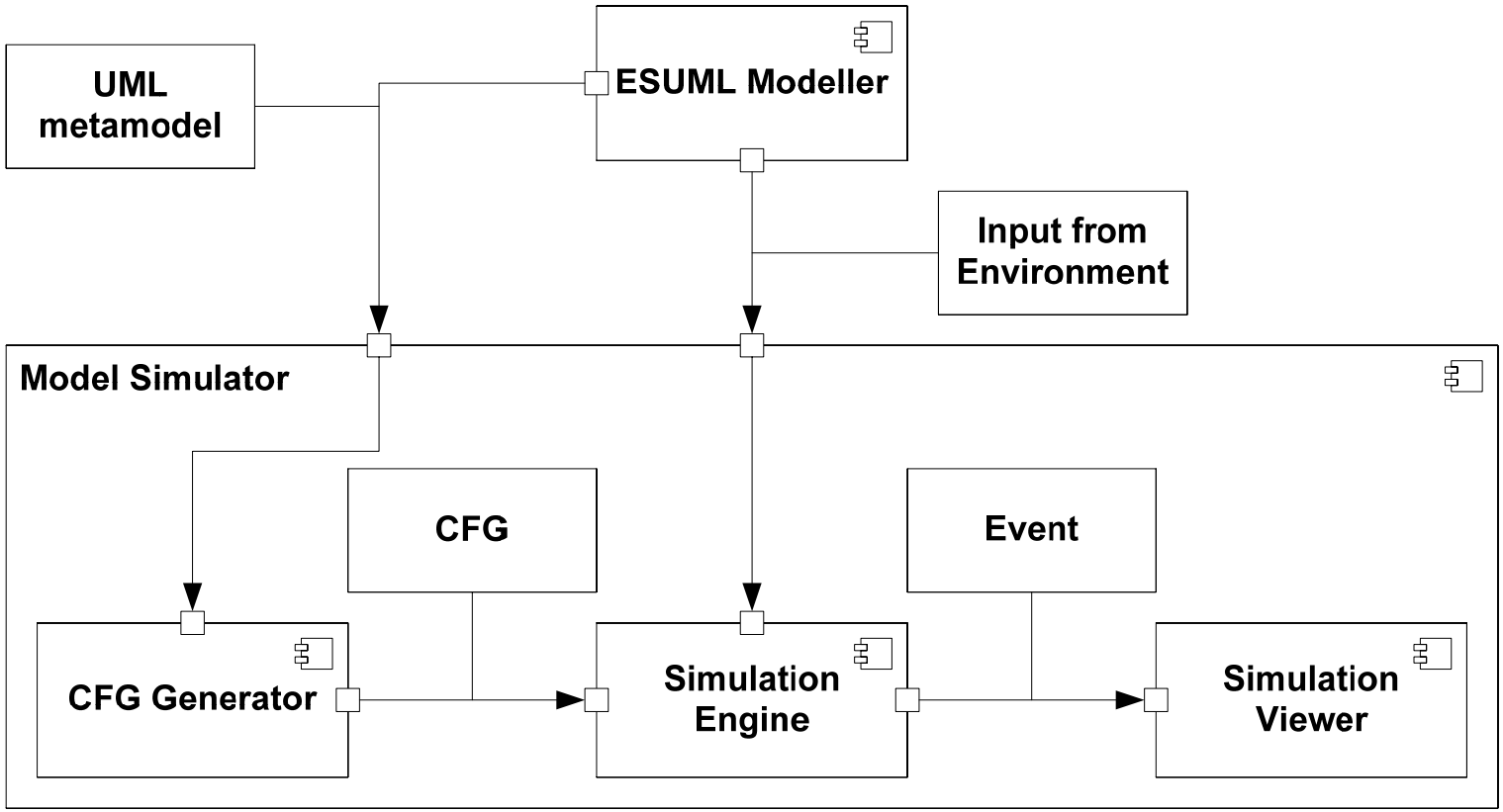
규칙 기반의 정적 분석

정적 분석 절차



시나리오 기반의 동적 시뮬레이션

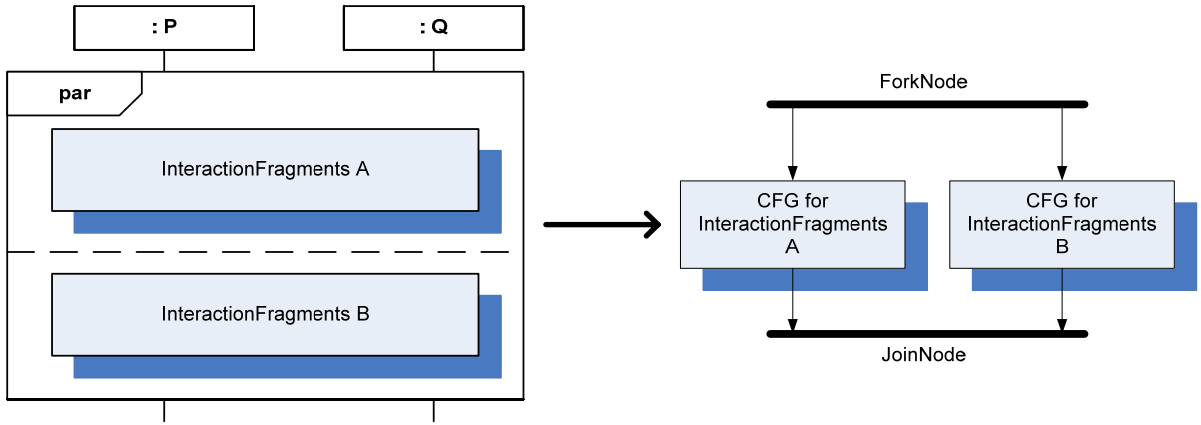
시뮬레이션 수행 구조



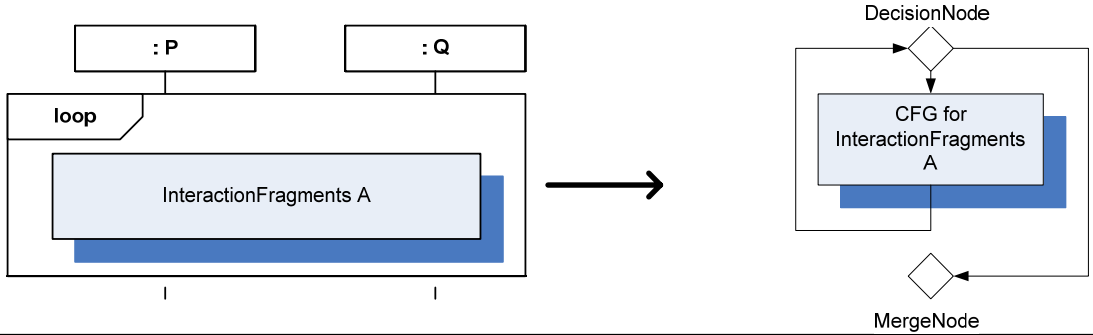
시뮬레이션 - CFG 생성

Control Flow Graph 생성 예제

- Parallel fragment



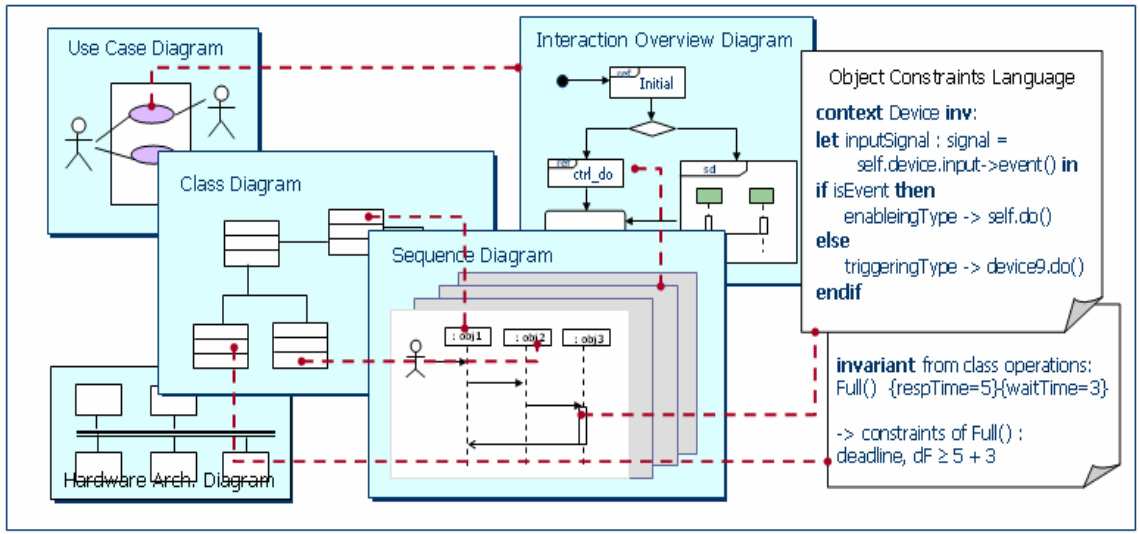
- Loop fragment

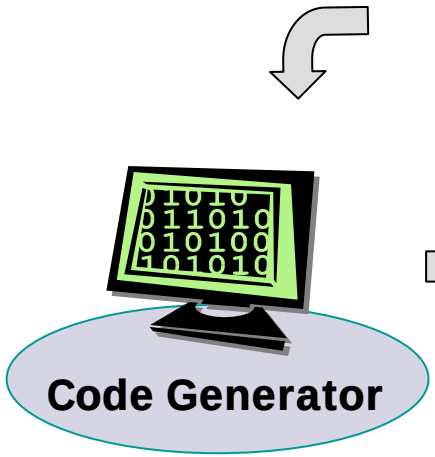


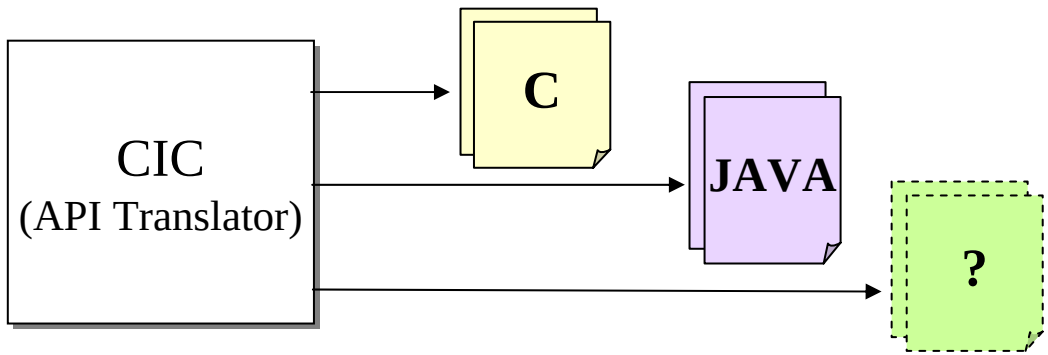
P5. 코드 생성 단계

Target Code Generation

PIM/PSM Models



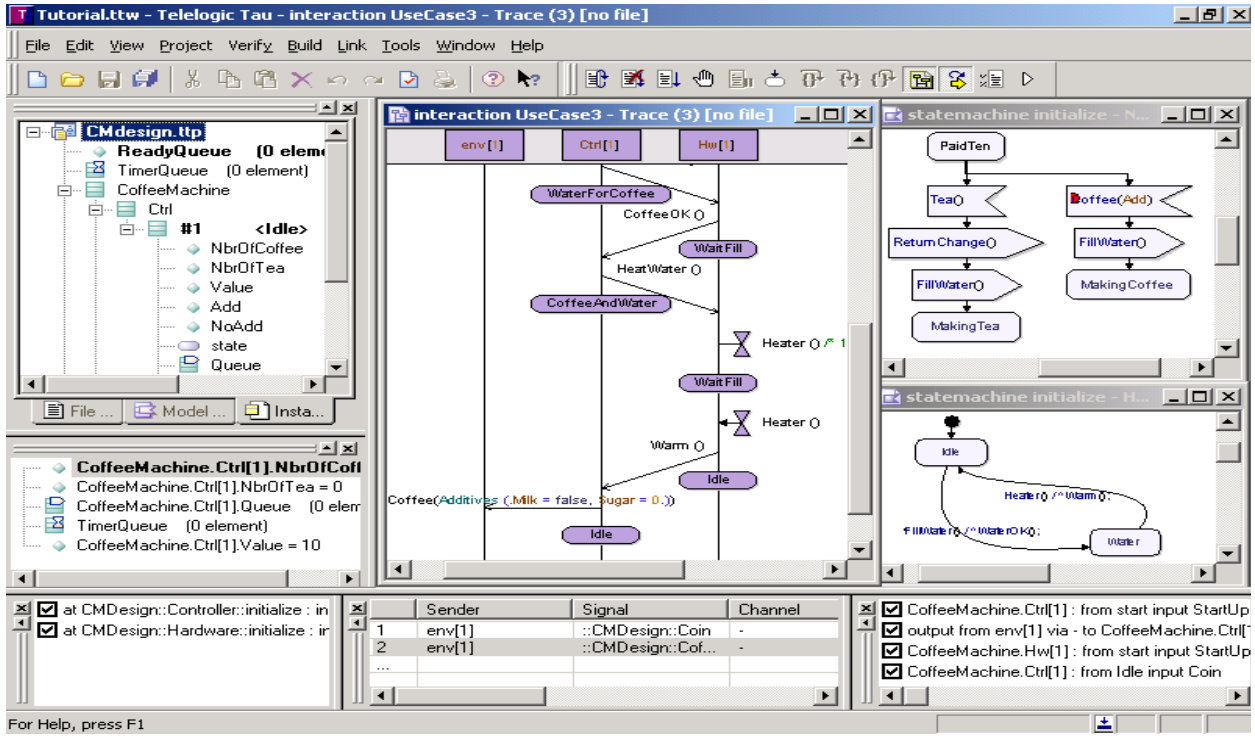




자동화 지원 도구

임베디드 소프트웨어 모델링 도구들

- IBM, Rose Technical Developer
- TeleLogic, Tau G2
- I-logix, Rhapsody



결 론

임베디드 소프트웨어 개발 방법론

- 기존의 소프트웨어 개발 방법과의 차이점은 크지 않음
- 개발상에서 고려되어야 하는 관심사를 표현하기 위한 활동이 더 필요한 방법
  - Time-Constraints, Resource, etc
  - Model Verification
  - Simulation 등
- 임베디드 소프트웨어의 응용에 적합한 방법의 구성이 중요

References

1. Accelerated Technology. Advanced Embedded System Programming using xtUML
2. P. Koopman, *Embedded System Design Issues – The Rest of the Story,* Proc. of the 1996 International Conference on Computer Design, Austin, October 7–9 1996.
3. H. Gomaa, Designing Concurrent, Distributed, and Real–Time Application with UML, Addison–Wesley, 2000
4. H. Gomma, Software Design Methods for Concurrent Real–Time Systems, Addison–Wesley, 1993
5. B.P. Douglass, ROPES: Rapid Object–Oriented Process for Embedded Systems, i–Logix, 1999
6. M. Awad, Object–Oriented Technology for Real–Time Systems, Prentice–Hall, 1996
7. S. Graf, UML based modeling of real–time and embedded systems and formal validation of timed systems with the IF environment, FMCO, 2004 (OMEGA project)

References

8. R. Alur, et al, Hierarchical Hybrid Modeling of Embedded Systems, First Workshop on Embedded Software, 2001 (MoBIES Project)
9. S. Jeon, J. Hong and D. Bae, Interaction-based Behavior Modeling of Embedded Software using UML2.0, ISORC 2006
10. OMG, UML2.0 Superstructure Specification, 2005
11. W. Lee, et al, Integration and Analysis of Use Cases Using Modular Petri Nets in Requirements Engineering," IEEE TOSE, Vol.24, No. 12, Dec. 1998
12. J. Hong and D. Bae, Incremental Scenario Modeling Using Hierarchical Object-Oriented Petri Net," IJSEKE, 11(3) pp.357-386, 2001