Modern Software Design Methods for Concurrent and Real-Time Systems

-By Hassan Gomaa

발표자 : 박홍제(200511327)

CONTENTS

- Introduction
- Concurrent processing concepts
- Run-Time support
- Survey of Design methods
- A modern software design method
- Performance analysis
- Conclusions
- Reference

Introduction

- 비용효과가 큰 Real-Time Systems
  -반도체와 microprocessor의 가격인하, 수행능력 향상
- Real-Time Systems의 특징
  - concurrent processing

CONCURRENT PROCESSING CONCEPTS

- Concurrent Tasks
- Mutual Exclusion
- Synchronization of Tasks
- Message Communication

Concurrent Tasks

- Execution of sequential component of a concurrent program
- Multiple asynchronous tasks running at different speeds
- Need to coordinate
  - Use mutual exclusion, task synchronization

Mutual Exclusion

- Required when only one task at a time may have exclusive access to a resource
- Critical section
  - Ex>robots
- Solved by using semaphores
  - P(s) – potential wait
  - V(s) – leaving the critical section

Synchronization of Tasks(1/2)

- Signal(Producer와 Consumer사이)
  - Producer task perform
    - Signal(E)
    - event(E)
  - Consumer task perform
    - Wait(E) – Suspend task until event has been signaled by producer

Synchronization of Tasks(2/2)

	Robot A	Robot B
1	Pick up part	Wait (Part_Ready)
2	Move Part to Workplace	
3	Release part	
4	Move to safe position	
5	Signal (Part_Ready)	
6	Wait (Part_Completed)	Move to Workplace
7		Drill Four holes
8		Move to safe position
9		Signal (Part_Completed)
10	Pick up part	
11	Remove from Workplace	

Message Communication(1/2)

- Communicate with each other using messages
- Used when data needs to be passed between two tasks
- Loosely coupled message communication(Asynchronous)
- Tightly coupled message communication(Synchronous)

Message Communication(2/2)

	Vision System	Robot task
1	Wait (Car_arrived)	Wait for message form vision system
2	Take picture of car body	
3	Identify car body	
4	Determine location and orientation of car body	
5	Send message (car model i.d., car body offset)to robot	
6		Read message (car model i.d., car body offset)
7		Select welding program for car model
8		Execute welding program using offset for car position
9		Signal (Move_car)

RUN-TIME SUPPORT

- Provided by
  - Kernel of an operation system
    - Provide functionality
  - Run-time support system
  - Threads package
    - Managing threads within heavyweight processes

Language Support

- Concurrent programming languages
  - Ada, Java
  - Supports constructs for task communication and synchronization
- No support
  - C, C++, Pascal, Fortran
  - Use a kernel or threads package

Real-Time Operating Systems

- Special needs
  - Support multitasking
  - Support priority preemption scheduling
  - Provide task synchronization and communication mechanisms
  - Provide a memory-locking capability
  - Provide a mechanism for priority inheritance
  - Have a predictable behavior

SURVEY OF DESIGN METHODS

- MASCOT
- RTSAD(Real-Time Structured Analysis and Design)
- DARTS(Design Approach for Real-Time Systems)
- JSD(Jackson System Development)
- OMD(Object Modeling Technique)
- CODARTS(Concurrent Design Approach for Real-Time Systems)
- Octopus
- ROOM(Real-Time Object-Oriented Modeling)

A MODERN SOFTWARE DESIGN METHOD

- Gomaa, Bacon
  - Blend object-oriented concepts with concepts of concurrent processing
- COMET
  - Integrates object-oriented and concurrent processing concept
  - Uses UML (Unified Modeling Language) notation
  - Describes decision made on how to use the UML notation

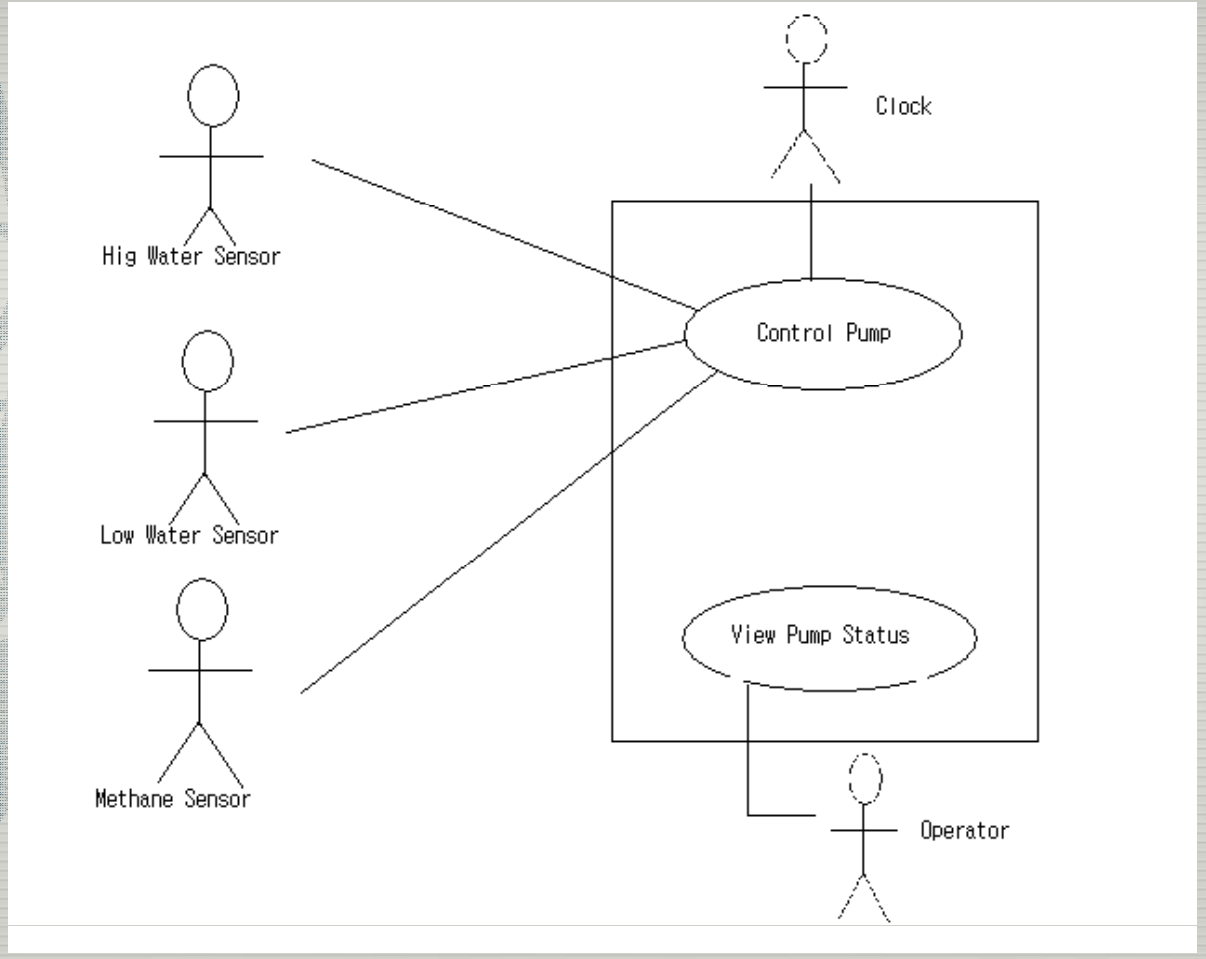
The COMET Method

- Concurrent object modeling and architectural design method
- Iterative life cycle
  - Requirements modeling
  - Analysis modeling
  - Design modeling

REQUIREMENTS MODELING WITH UML

- System을 입출력만 있는 black box로 생각하여 기능적인 관점에서 파악
- 유용한 모델링객체 : Use case model is developed
- Actor
  - Human user
  - External system
  - I/O devices
  - Timer

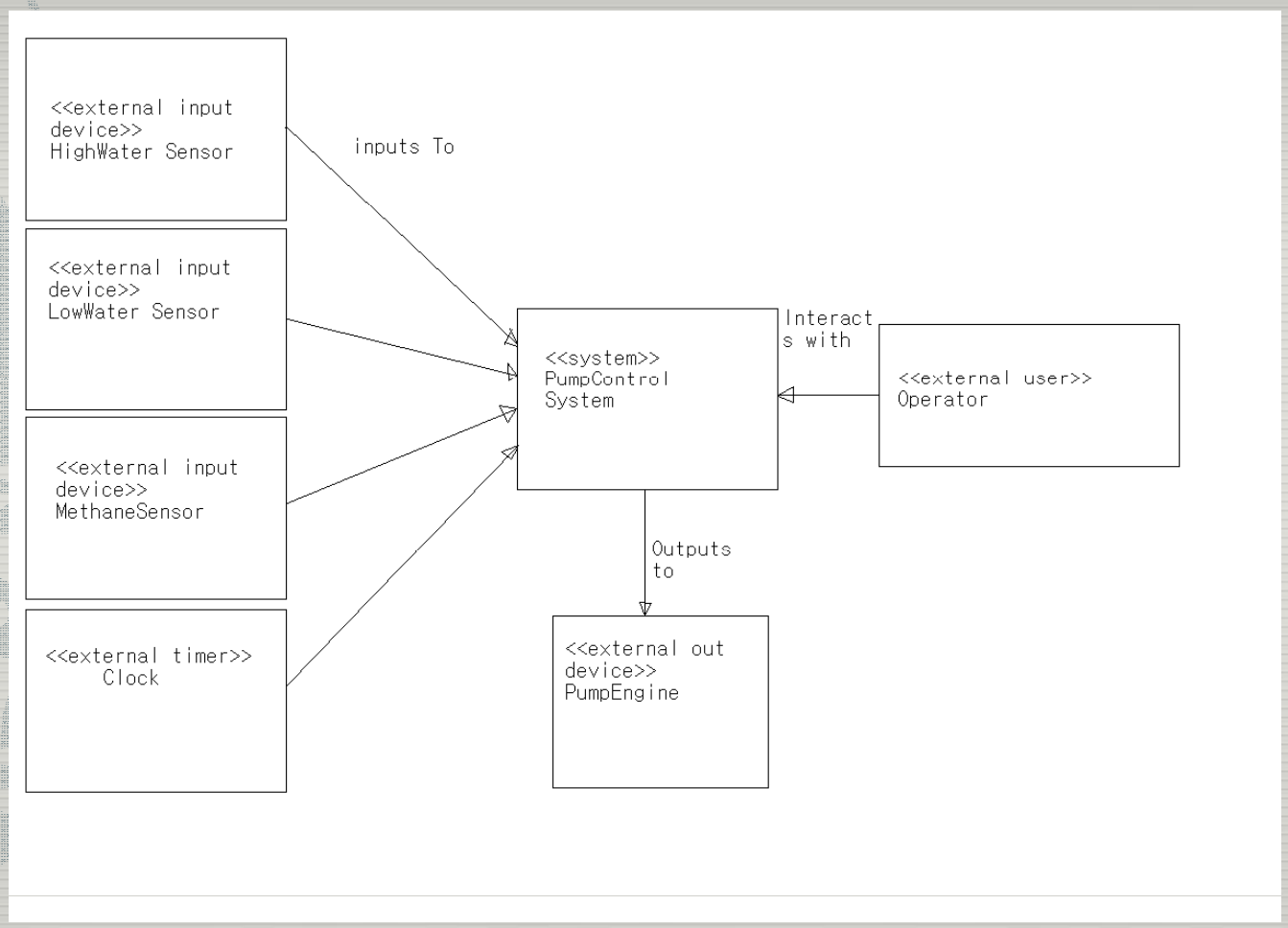
Use case model for pump monitoring and control system



ANALYSIS MODELING WITH UML -Static Modeling

- System context
  - Understand interface between system and external environment
- UML notation does not support system context diagram, use
  - Static model
  - Collaboration model

Pump monitoring and control system class context diagram



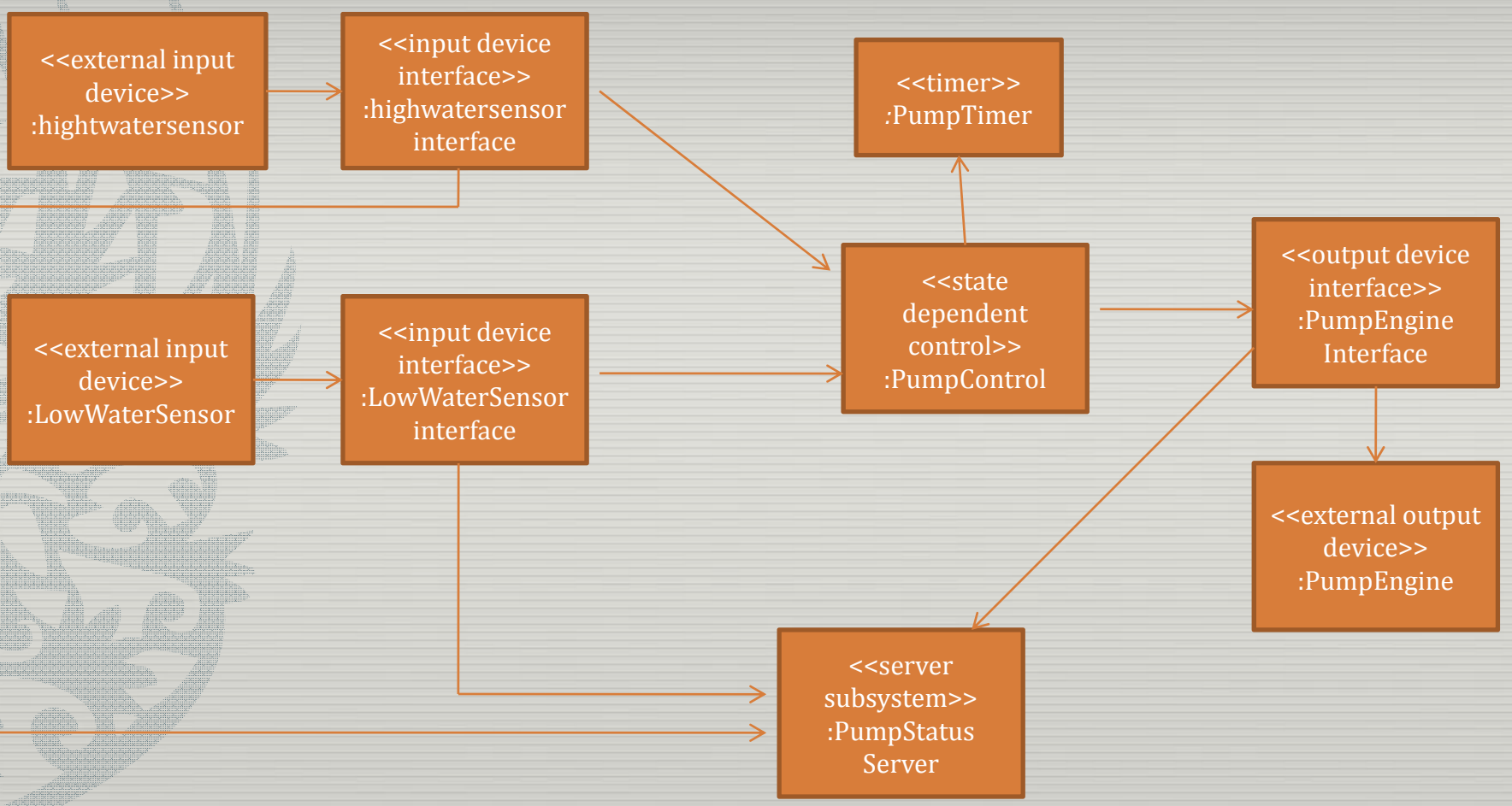
ANALYSIS MODELING WITH UML
-Object Structuring

- Assist designer
- Goal – 유사한 특징을 가진 object들로 Group을 구분시키는것
- UML stereotypes are used
  - Stereotype – A subclass of an existing modeling element
- Can be
  - <<entity>>class
  - <<interface>>class
  - <<control>>class
  - <<application logic>>class

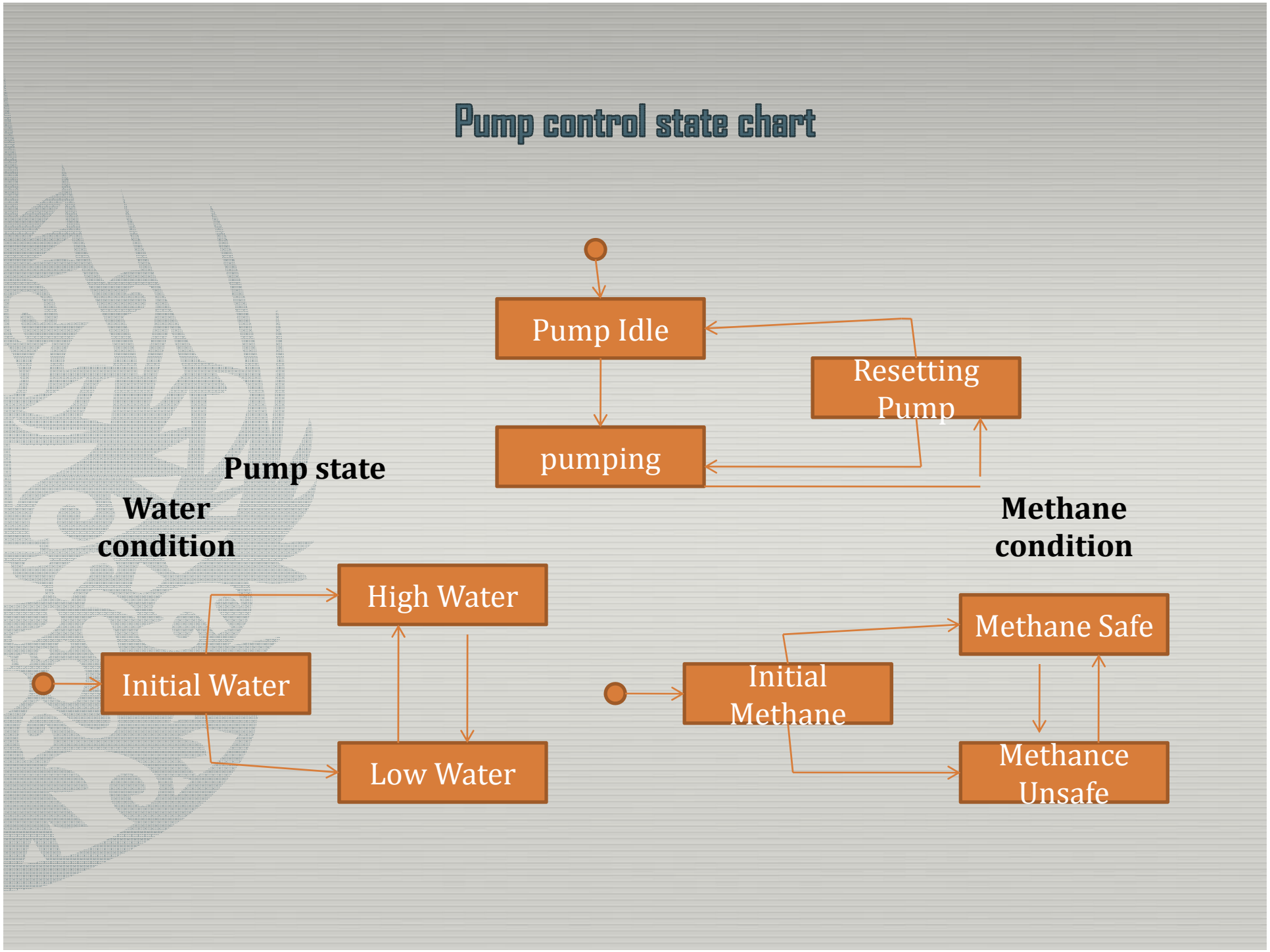
ANALYSIS MODELING WITH UML -DYNAMIC MODELING

- Understand finite-state machine model
  - State chart
- State-dependent dynamic analysis – object간의 interaction에 초점.
  - Use collaboration or sequence diagram
- need conjunction
  - Output event of each diagram must be consistent

Collaboration diagram for control pump use case



Pump control state chart



DESIGN MODELING
-Transition from Analysis to Design

- Synthesize
- Consolidated Collaboration diagram
  - Synthesis of all the collaboration
  - Can get very large

DESIGN MODELING
-SOFTWARE ARCHITECTURE DESIGN

- System is broken down into subsystem
- Goal – to have objects with highly coupling in the same subsystem, weakly coupling in the different subsystems.
- Subsystem – composite object

DESIGN MODELING
-CONCURRENT COLLABORATION DIAGRAMS

- Active object
  - Own thread of control
  - Task : box with thick black lines
- Passive object
  - Execute when another object invokes
  - Passive object : box with thin black line

DESIGN MODELING -ARCHITECTURAL DESIGN

- Distributed real-time systems은 분산된 nodes에서 실행된다.
- Subsystem component는 logical node에서 동시에 실행되는 일의 집합이다.

Distributed software architecture

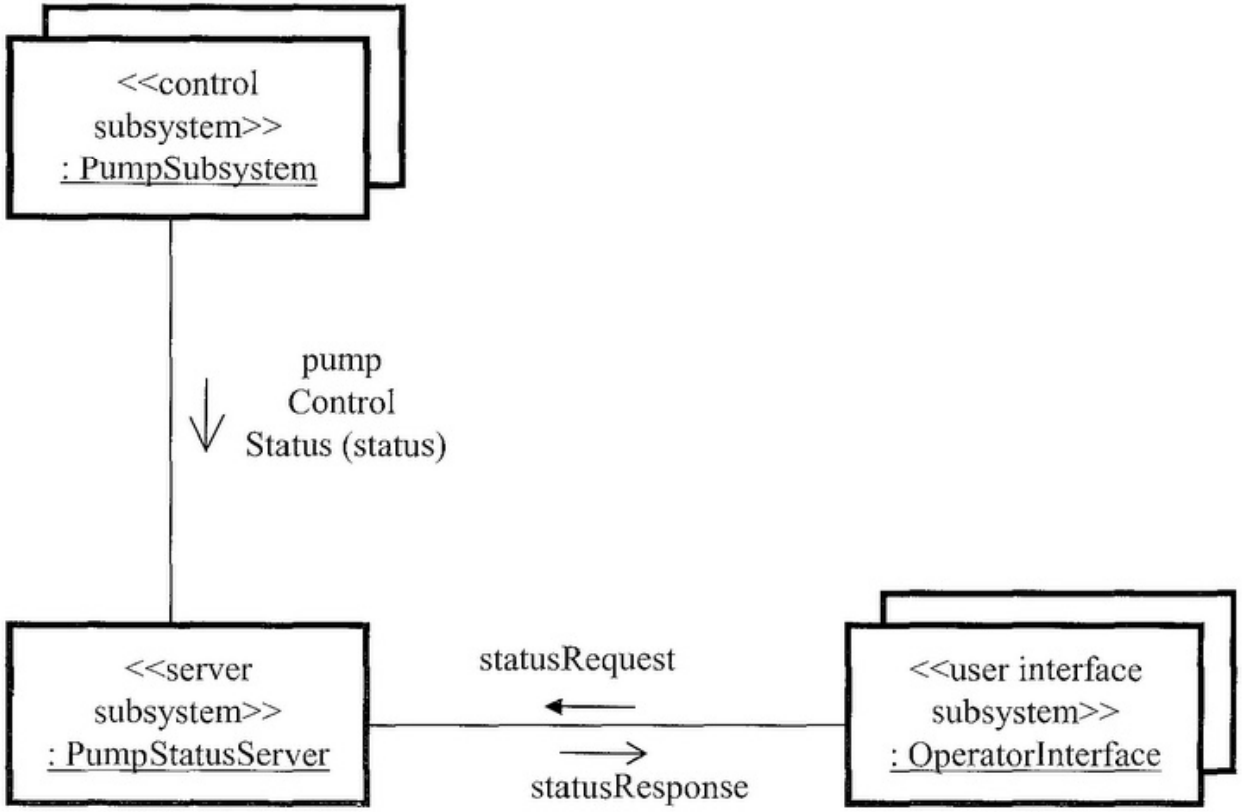
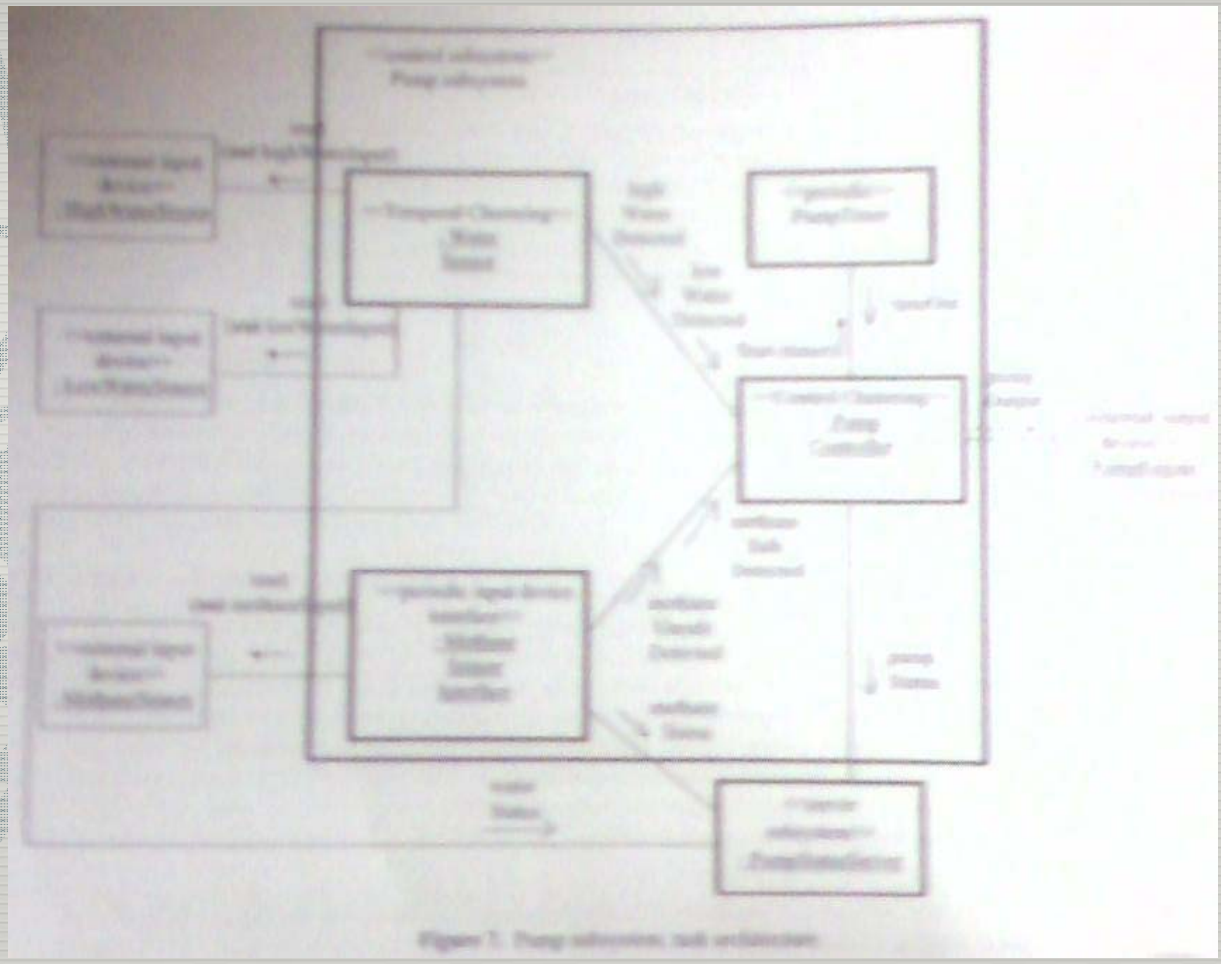


Figure 5. Distributed software architecture.

DESIGN MODELING
-TASK STRUCTURING

- Each subsystem is structured into concurrent tasks and the task interfaces are defined.
- Stereotypes : Depict different kinds of devices
- Object : determine active, categorized->특성부각

Pump subsystem, task architecture



DESIGN MODELING
-DETAILDE SOFTWARE DESIGN

- Composite task internals designed
- Detailed task synchronization issues
- Connector classes are designed
- Task’s internal event sequencing logic is defined

Water sensors, temporal clustering with nested device interface objects

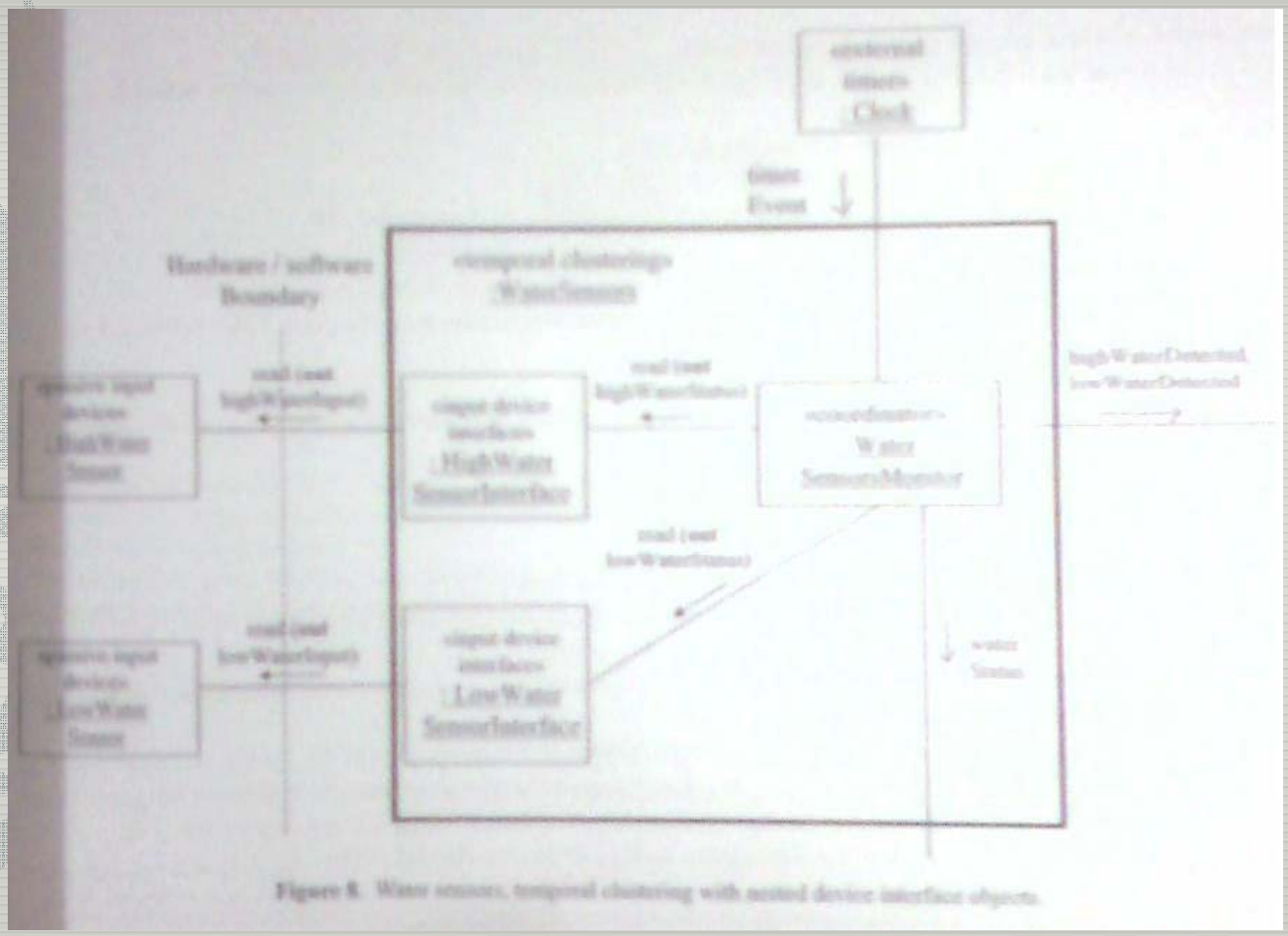


Figure 8. Water sensors, temporal clustering with nested device interface objects.

PERFORMANCE ANALYSIS

- Important for real-time systems
- Achieved by realtime scheduling
  - Determine whether it can meet its deadlines.
- Use event sequence analysis

CONCLUSIONS

- Blend object-oriented concepts with concurrent processing concepts
- COMET method integrates object-oriented and concurrent processing concepts and uses UML notation
- Increasing importance for future real-time systems

REFERENCES

- COMET 객체지향 설계방법론을 이용한 차량용 소프트웨어의 설계-한국자동차공학회 워크숍 및 심포지엄 논문집 (김세화, 임진택, 유우석, 홍성수)