Modern Software Design Methods for Concurrent and Real-Time Systems

Hassan Gomaa

발표자 : 200811306 이해성
200511355 정용구
200511325 박찬석
200412349 임현웅

1. INRTODUCTION

- The massive reduction in the cost of micro-processor and semiconductor chips and the large increase in microprocessor performance
- So real-time and distributed real-time micro-computer-based systems are very cost-effective solution to many problems
- Commercial, industrial, military, medical, consumer products are microcomputer based
- Real-time systems is concurrent processing; activities occurring simultaneously
- Consequently, real-tome systems deal with con-current activities

1. INRTODUCTION

2. Concurrent Processing concepts
3. Run-time Support for Concurrent Tasks
4. Survey of Design Methods for Concurrent and Real-time Systems
5. A Modern Software Design Method for Concurrent and Real-time Systems
6. Performance Analysis of Real-time Designs
7. Conclusions

2. Concurrent processing concepts

2.1. Concurrent Tasks

- A concurrent task represents the execution of sequential program
- A concurrent system consists of several tasks executing in parallel
- Concurrency in software system is obtained by having multiple asynchronous tasks
- There are two typical way to deal with tasks which named ‘robot arms’ : mutual exclusion and task synchronization

2. Concurrent processing concepts

2.2. Mutual Exclusion

- Mutual exclusion is required when only one task at a time may have exclusive access to a resource
- To prevent the collision that could occur if more than one robot is allowed to move into the colli-sion zone at the same time, robot entry is made mutually exclusive

2. Concurrent processing concepts

2.2. Mutual Exclusion

EX)

Perform operations outside collision zone

P (Collision_Zone_Semaphore)

Perform mutually exclusive operation in collision zone

V(Collision_Zone_Semaphore)

Perform more operations outside collision zone

2. Concurrent processing concepts

2.3. Synchronization of Tasks

- Task synchronization is required when one task wishes to signal another to notify it that some event has occurred
- For example, the producer robot moves a part into position and then signals the event Part_Ready and the consumer robot, which is suspended waiting for the signal, is reactivated so that it can move to the part and pick it up

2. CONCURRENT PROCESSING CONCEPTS

2.3. Synchronization of Tasks

EX)

Robot A

WHILE Work_Available DO

Pick up part
Move Part to Workplace
Release part
Move to safe position
Signal (Part_Ready)
Wait (Part_Completed)
Pick up part
Remove from Workplace
END

Robot B

WHILE Work_Available DO

Wait (Part_Ready)
Move to workplace
Drill Four holes
Move to safe position
Signal (Part_Completed)
END

2. Concurrent processing concepts

2.4. Message Communication

- Message communication is used when data needs to be passed between two tasks
- For example, the producer sends a message to the consumer and consumer receive a message
- If a message is available, the consumer will receive it and continue processing; otherwise, the consumer is suspended until the message arrives
- Two types of message communication
  - Loosely couple message communication
  - Tightly couple message communication

2. Concurrent processing concepts

2.4. Message Communication

EX)

Vision System:

Wait (Car_arrived)

Take picture of car body

Identify car body

Determine location and orientation of car body

Send message (car model i.d., car body offset) to robot

Robot task:

Wait for message from vision system

Read message (car model i.d., car body offset)

Select welding program using offset for car position

Signal (Move_car)

3. Run-time support for concurrent tasks

Run-time support for concurrent processing may be provide by:

- Kernel of an operating system
  - This has the functionality to provide services for concurrent processing
- Run-time support system
  - For a concurrent language
- Thread package
  - Provides services for managing threads within heavyweight process

3. Run-time support for concurrent tasks

3.1. Language Support for Concurrent Tasks

- With sequential programming languages(C, C++, Pascal, and Fortran), there is no support for concurrent tasks
- So it is necessary to use a kernel or threads package
- With concurrent programming language(Ada and Java), the language supports constructs for task communication and synchronization
- So the language's runtime system provides the services and underlying mechanisms

3. Run-time support for concurrent tasks

3.2. Real-Time Operating Systems

A real-time operating system must:

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

4. Survey of design method for concurrent and real-time systems

- MASCOT notation [Simpson 1979]
- MASCOT design method [Simpson 1986]
- Software Cost Reduction Method [Parnas, Clements, and Weiss 1984]
- Real-Time Structured Analysis and Design (RTSAD) [Ward and Mellor 1985, Hatley and Pribhai 1988]
- Design Approach for Real-Time Systems (DARTS) [Gomma 1984]
- Jackson System Development (JSD) [Jackson 1983]
- Object Modeling Technique (OMT) [Rumbaugh 1991]
- Concurrent Design Approach for Real-Time Systems (CODARTS) [Gomma 1993]
- Octopus [Awad, Kuusela, and Ziegler 1996]
- Real-Time Object-Oriented Modeling (ROOM) [Selic, Guulekson and Ward 1994]
- Douglass [1999, 2004]

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.1. Introduction

- Most books on object-oriented analysis and design omit the important design issues that need to be addressed when designing real-time and distributed applications
- COMET method is for the designing real-time and distributed applications
- How to use the UML notation to address the design of large-scale concurrent, distributed, and real-time applications
- Example) Pump monitoring and control system

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.2. The COMET Method

- COMET is a concurrent object modeling and architectural design method for the development of concurrent applications
- Requirements modeling phase
  - Define the functional requirements of the system
- Analysis modeling phase
  - Static and dynamic models of the system are developed
- Design modeling phase
  - An architectural design model is developed

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.2. The COMET Method

Distinguishing features of the Comet method are the emphasis on:

- Structuring criteria to assist the designer at different stages of the analysis and design process: subsystems, objects, and concurrent tasks
- Dynamic modeling, both object collaboration and state charts, describing in detail how object collaborations and statecharts relate to each other
- Distributed application design, addressing the design of configurable distributed components and intercomponent message communication interfaces
- Concurrent design, addressing in detail task structuring and the design of task interfaces
- Performance analysis of real-time designs using real-time scheduling

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.3. Requirements Modeling with UML

- The use of actors in real-time application
- There are several variations on how actors are modeled
- An actor is very often a human user
- An actor can also be an external I/O device
- An actor can also be a timer
- An example of a use case model from the pump monitoring and control system is given in Figure 1

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.3. Requirements Modeling with UML

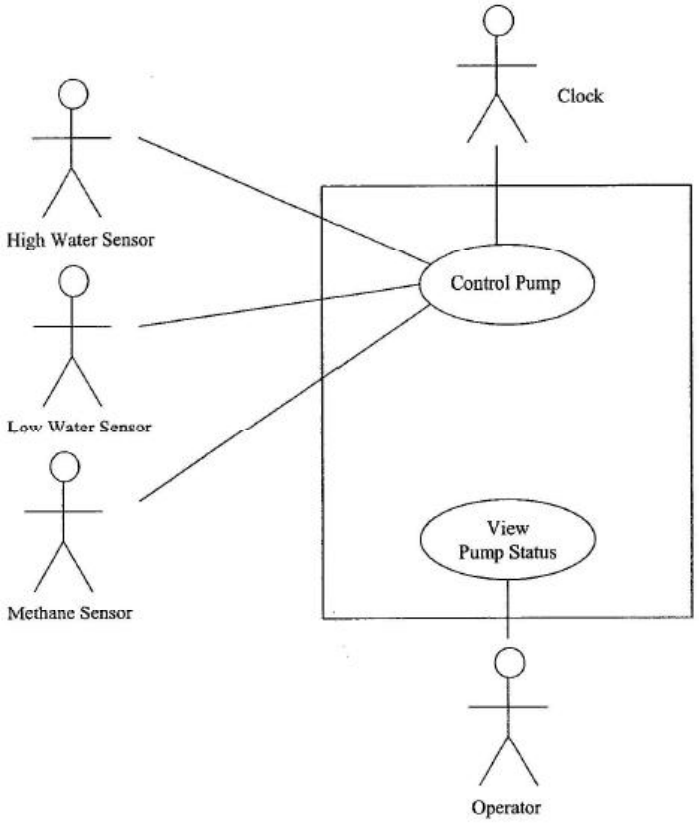

Figure 1. Use case model for pump monitoring and control system.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.1. Static Modeling

- The UML notation does not explicitly support a system context diagram
- A system context class diagram provide a more detailed view of the system boundary than a use case diagram
- Using the UML notation for the static model, the system context is depicted showing the system as an aggregate class with the stereotype <system>
- An external class can be an <external input device>, an<external output device>, an <external I/O device>, an <external user>, or an <external system>
- An example of a system context class diagram from the pump monitoring and control system is given in Figure 2

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.1. Static Modeling

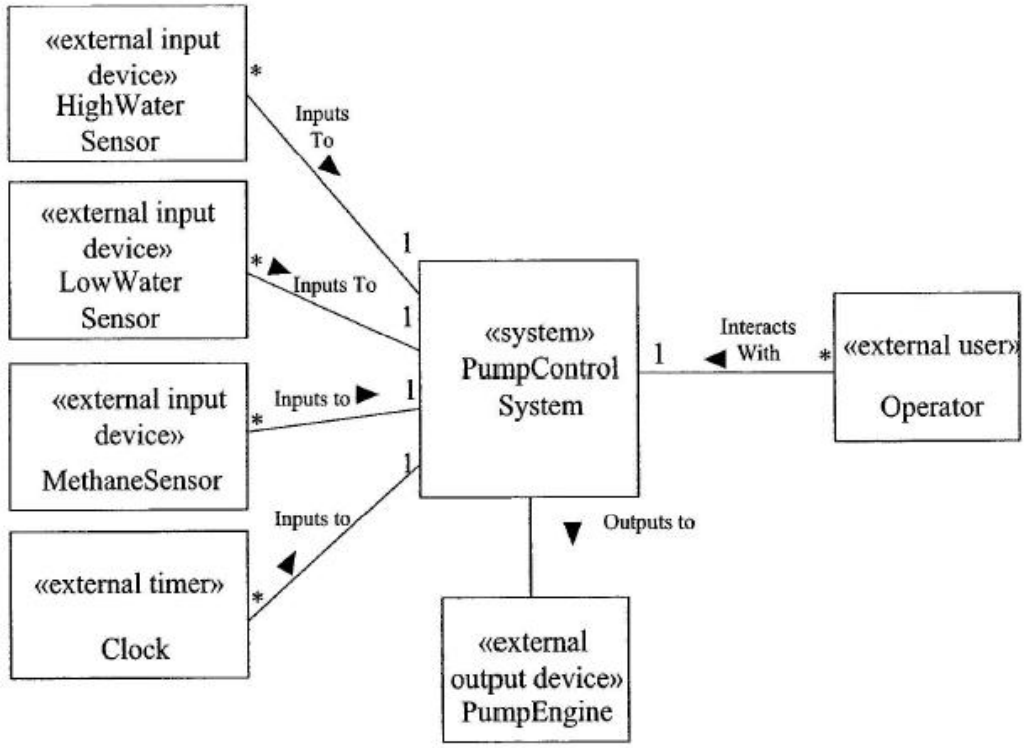

Figure 2. Pump monitoring and control system class context diagram.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.2. Object Structuring

- Several object-based and object-oriented analysis methods provide criteria for determining objects in the problem domain
- Object structuring is to categorize objects in order to group together with similar characteristics
- An application class can be categorized as an <entity> class, a <controll> class, an <interface> class, a <controll> class, or an <application logic> class
- Real-time systems will have many device interface classes to interface to the various sensors and actuators

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.3. Dynamic Modeling

- It is important to understand how the finite-state machine model, depicted using a state chart that is executed by a state-dependent control object, and relates to the interaction model, which depicts the interaction of this object with other objects
- State-dependent dynamic analysis addresses the interaction among objects that participate in state-dependent use cases
- The state chart needs to be considered in conjunction with the collaboration diagram
- An example of the collaboration diagram for the control pump use case is given in Figure 3, and the state chart for the Pump Control object is shown in Figure 4

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.3. Dynamic Modeling

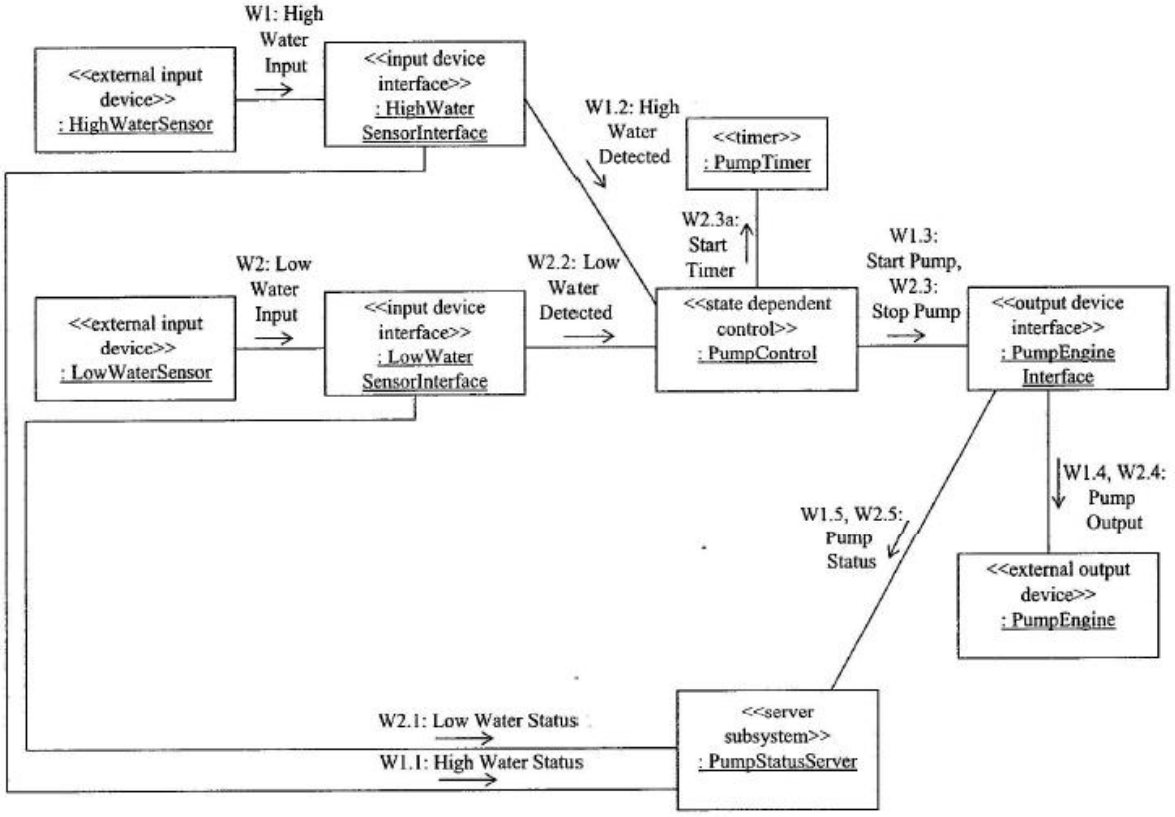

Figure 3. Collaboration diagram for control pump use case.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.4. Analysis Modeling with UML

5.4.3. Dynamic Modeling

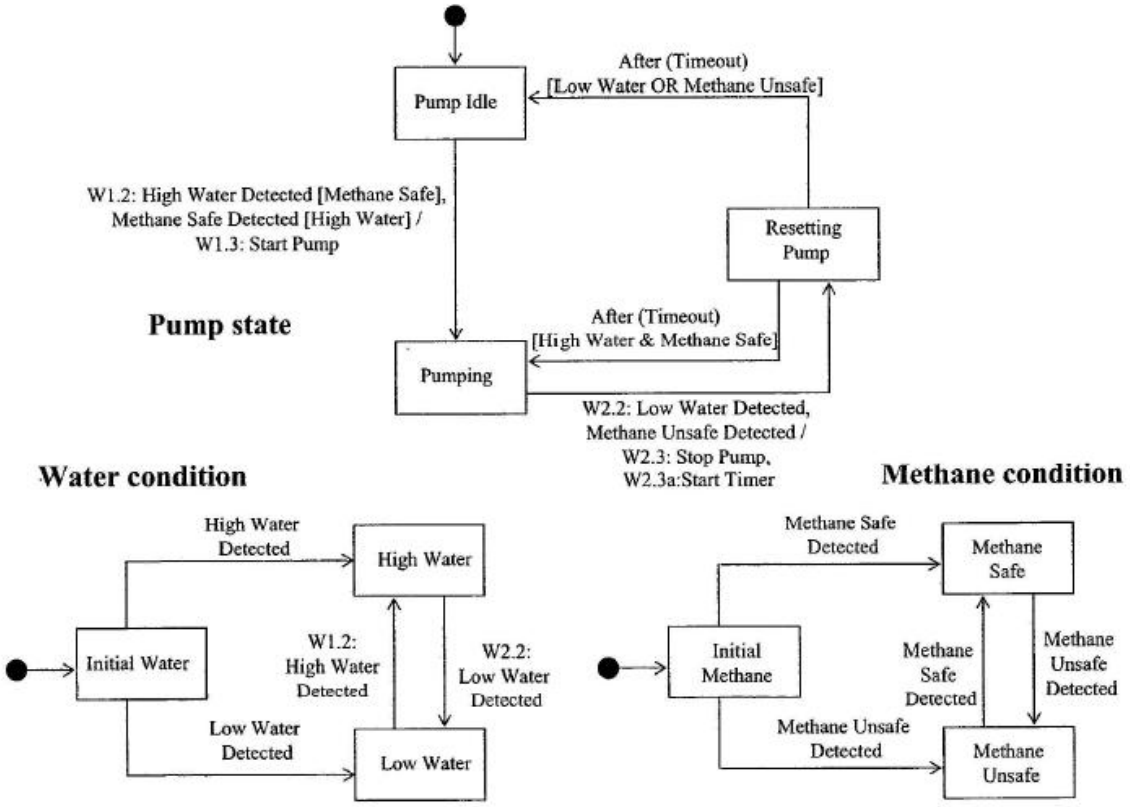

Figure 4. Pump control state chart.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.1. The Transition from Analysis to Design

- The consolidated collaboration diagram is a synthesis of all the collaboration diagrams developed to support the use cases
- The consolidated collaboration diagram, which depicts the objects and messages from all the use-case-based collaboration diagrams, can get very large for a large system and thus it may not be practical to show all the objects on one diagram
- So develop a higher-level subsystem collaboration diagram to show the dynamic interaction between subsystems on a subsystem collaboration diagram
- As shown in Figure 5

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.1. The Transition from Analysis to Design

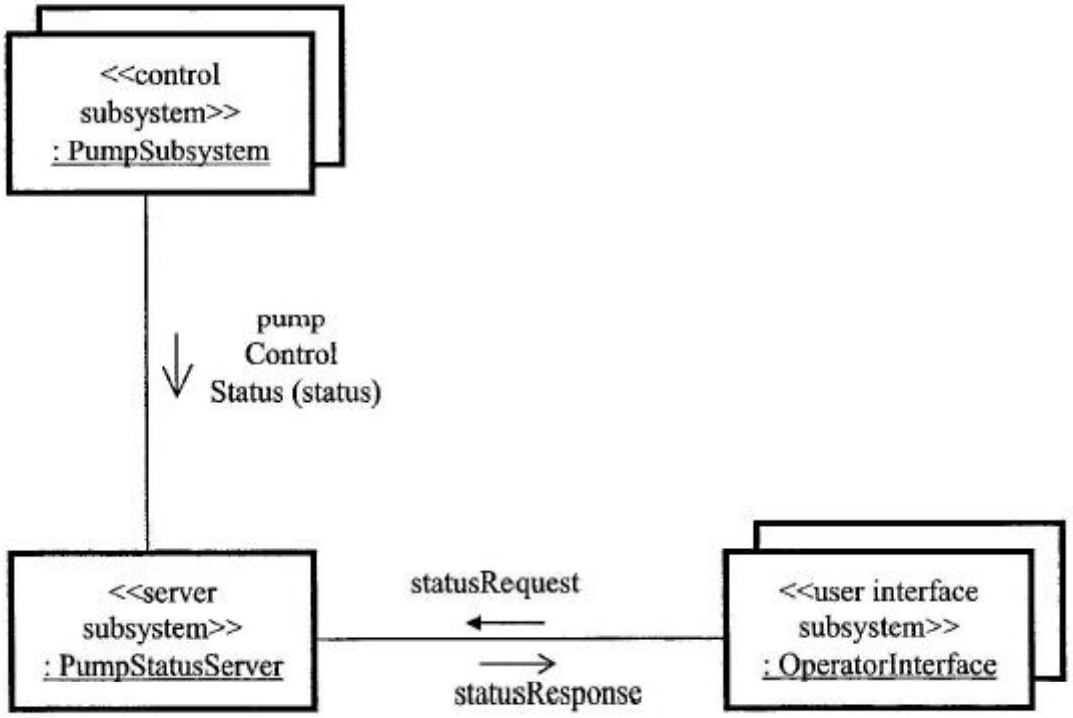

Figure 5. Distributed software architecture.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.2. Software Architecture Design

- During software architecture design, the system is broken down into subsystems and the interfaces between the subsystems are defined
- Software architecture design is to have objects with high coupling among each other in the same subsystem

5.5.3. Concurrent Collaboration Diagrams

- An active object has its own thread of control and executes concurrently with other objects
- A passive object only executes when another object invokes one of its operations
- On a concurrent collaboration diagram, an active object (task) is depicted as a box with thick black lines and a passive object is depicted as a box with thin black lines

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.4. Architectural Design of Distributed Real-Time Systems

- Distributed real-time system is structured into distributed subsystems, where a subsystem is designed as a config-urable component and corresponds to one logical node
- Task in different subsystems may communicate with each other using several different types of message communication including asynchronous communication, synchronous communication, client/server communication, group communication, brokerd communication, and negotiatated communication
- An example is shown in Figure 6

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.4. Architectural Design of Distributed Real-Time Systems

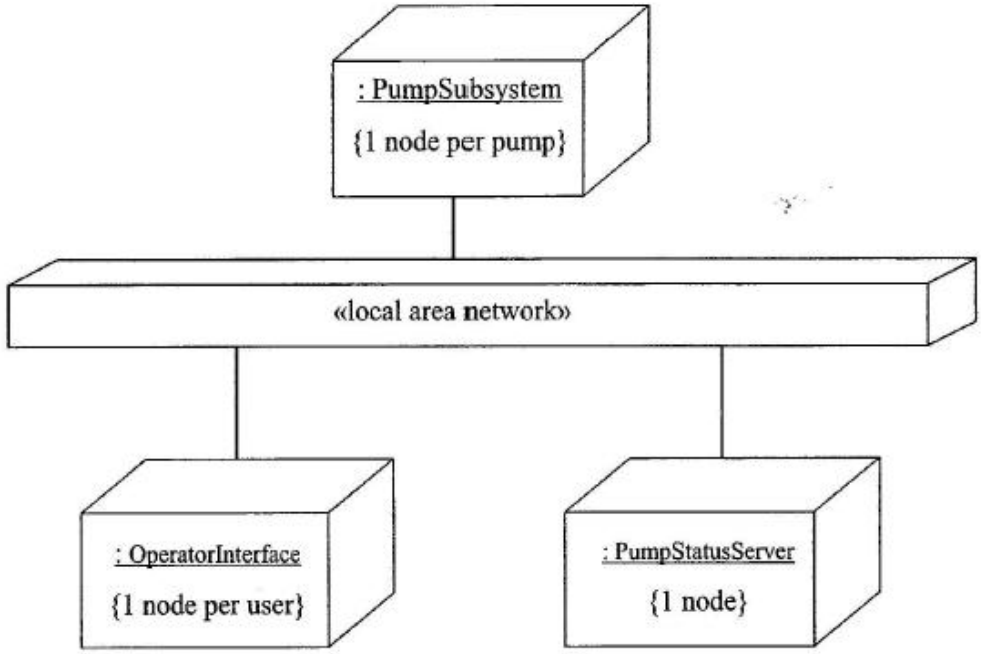

Figure 6. Distributed system configuration.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.5. Task Structuring

- Task structuring criteria are provided to assist in mapping an object-oriented analysis model of system to a concurrent tasking architecture
- During task structuring, if an object in the analysis model is determined to be active, then it is categorized further to show its task characteristics
- An example of a task architecture for the pump monitoring and control system is given in Figure 7

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.5. Task Structuring

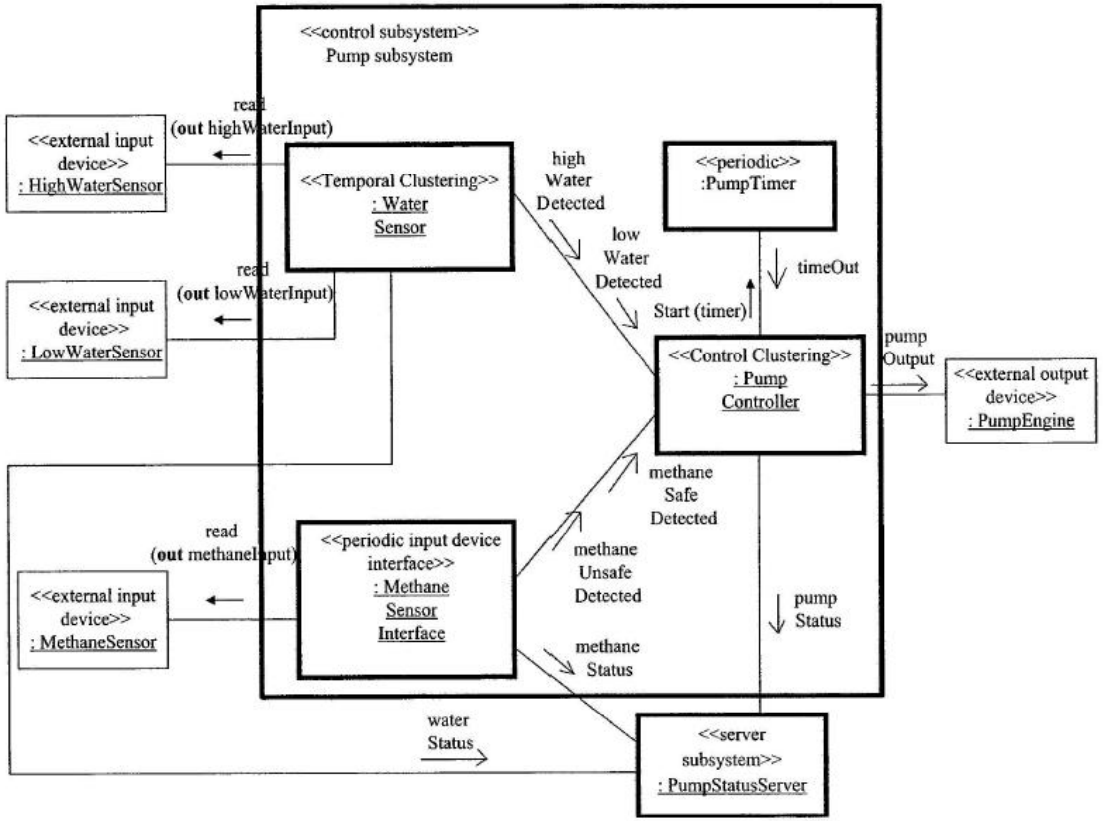

Figure 7. Pump subsystem, task architecture.

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.6. Detailed Software Design

- In this step, the internals of composite tasks that contain nested objects are designed, detailed task synchronization issues are addressed, connector classes are designed that encapsulate the details of intertask communication, and each task's internal event sequencing logic is defined
- An example of the detailed design of a composite task is given in Figure 8

5. A MODERN SOFTWARE DESIGN METHOD FOR CONCURRENT AND REAL-TIME SYSTEMS

5.5. Design Modeling

5.5.6. Detailed Software Design

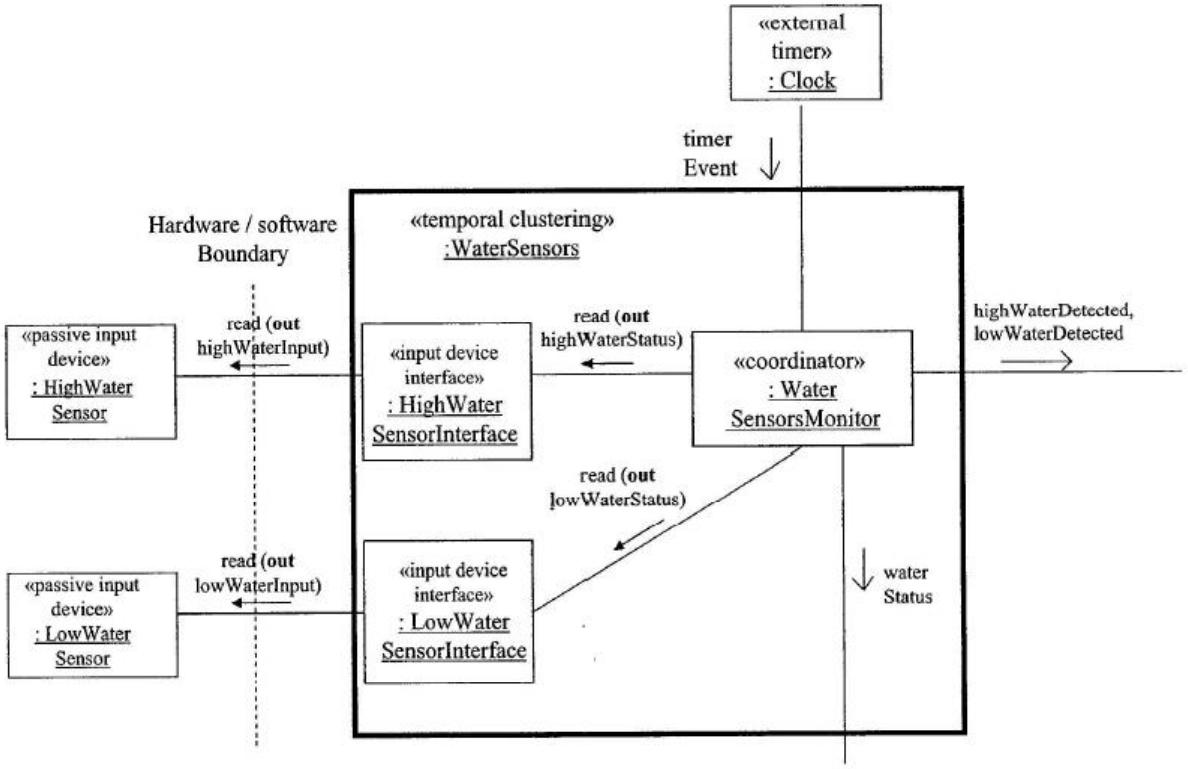

Figure 8. Water sensors, temporal clustering with nested device interface objects.

6. Performance analysis of real-time designs

- Performance analysis of software designs is particularly important for real-time systems
- Real-time scheduling is an approach that is particularly appropriate for hard real-time systems that have deadlines that must be met
- Event sequence analysis considers scenarios of task collaborations and annotates them with the timing parameters for each of the tasks participating in each collaboration

7. Conclusions

- When designing concurrent and real-time systems, it is essential to blend object-oriented concepts with the concepts of concurrent processing
- With the proliferation of low-cost workstations and personal computers operating, the concurrent systems is likely to grow rapidly in the next few years
- With the growing need for reusable design, design method for software product lines are likely to be of increasing importance for future real-time systems