병렬 태스크의 행동 특성 추출에 관한 연구

유창문 김규년
울산대학교 컴퓨터 정보통신 공학과
window@cic.ulsan.ac.kr knkim@uou.ulsan.ac.kr

A Study on the Extraction of Behavior Characteristic for Concurrent Task

Chang-Moon Yu Kyu-Nyun Kim
School of Computer Engineering & Information Technology

요 약

COncurrent Design Approach for Real-Time System(CODARTS) 방법론은 Gomaa에 의해 제안된 실시간 소프트웨어 설계 방법론으로서 분석 단계에서 COBRA방법을 사용하여 시스템의 객체와 함수들을 식별하고 행동 모델을 개발한다. 그리고 설계 단계에서는 병렬 태스크 구조화 지침 및 정보 은닉 모듈 구조화 지침을 적용하여 행동 모델의 객체와 함수들을 병렬 태스크와 정보 은닉 모듈들로 구조화한다. 마지막으로 병렬 태스크와 정보 은닉 모듈을 결합하여 소프트웨어 구조를 개발하고 구현 단계를 수행한다.

소프트웨어 구조를 개발하고 구현 단계를 수행하기 위해서는 병렬 태스크의 행동 특성이나 태스크간의 인터페이스가 정확히 명시되어야 한다. 이는 분석 단계에서 식별된 객체와 함수들에서 태스크에 대한 정보를 추출함으로써 이루어질 수 있다.

본 논문에서는 행동 모델의 객체와 함수들로부터 병렬 태스크에 대한 행동 특성 정보 추출 방법을 제안하고 태스크 사이의 인터페이스를 결정하는 방법을 보인다.

1. 서론

실시간 시스템의 설계에서 병렬 태스크를 적용하는 것은 그 표현 방법이 실세계와 유사하고, 태스크의 역할이 분명해지므로 전체 시스템의 수행 시간 감소와 시스템 초기 단계에서 스케쥴링과 성능 분석을 할 수 있는 장점을 가진다[1]. 실시간 시스템 설계 방법론 중에서 이러한 태스크 구성을 강조한 대표적인 것이 CODARTS(COncurrent Design Approach for Real-Time Systes) 방법론이다.

CODARTS방법론은 분석단계에서 시스템을 병렬 수행 가능성을 가진 객체와 함수로 분해하여 행동 모델을 개발하고, 설계단계에서 그러한 객체와 함수들에 병렬 태스크 구성 지침을 적용하여 병렬 태스크로 구성한다. 하지만 CODARTS에서는 행동 모델의 객체와 함수로부터 태스크의 구체적인 행동 특성을 추출하는 방법을 제공하지 않는다. 이는 설계를 마친 시스템을 구현하고자 할 때 모호성을 제공하는 원인이 된다.

본 논문에서는 행동 모델의 객체와 함수들로부터 병렬 태스크에 대한 행동 특성 정보를 추출하는 방법을 제안하고 태스크간의 인터페이스를 결정하는 방법을 보인다.

2. CODARTS 방법론의 수행 절차

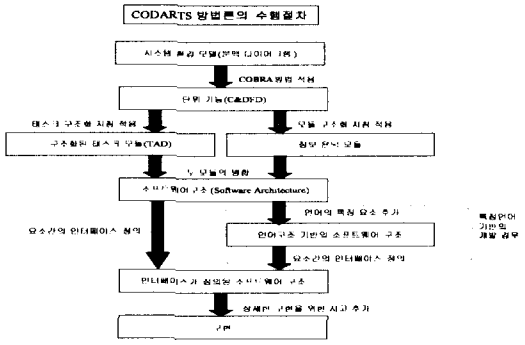
우선 분석단계에서 Concurrent Object-Based Real-Time Analysis(COBRA) 방법을 사용하여 시스템을 병렬 수행 가능성을 가진 객체와 함수들로 분해하고 행동 모델을 개발한다. 행동 모델의 표기법으로는 Control and Data Flow Diagram(C&DFD)를 사용한다. 식별된 객체와 함수들은 C&DFD에서 transformation으로 표현되며, 설계 단계에서 각 transformation에 태스크 구조화 지침[2]과 정보 은닉 모듈 구조화 지침[1]을 적용하여 병렬 태스크와 정보 은닉 모듈들로 구조화한다. 마지막으로 병렬 태스크와 정보 은닉 모듈을 결합하여 소프트웨어 구조를 개발하고 구현단계를 수행한다.

위에서 설명한 CODARTS방법론의 수행 절차를 도식화하면 그림 1과 같다.

3. 태스크 행동 특성 정보 추출

3.1 STS(Specification for Task Structuring)

COBRA 행동 모델에 태스크 구성 지침을 적용하여 병렬 태스크를 구성하기 위해서는 C&DFD에 명시된 각 transformation의 정보를 기술하기 위한 구조가 필요하다. 본 논문에서는 이를 위해 STS[3]를 사용한다.



[그림 1] CODARTS방법론의 수행절차

STS는 태스크 구성에 필요한 C&DFD의 정보를 기록하는 형식이다. 이 장에서는 태스크 정보 추출에 필요한 transformation의 STS에 대해서만 언급하기로 한다. transformation에 대한 STS는 표 1과 같다.

속 성	설 명
Transformation Type	Transformation의 종류. IODTr(입출력 특성), INTr(내부 특성), COTr(제어 특성)
Activation Type	Transformation이 활성화되는 형식. Asynchronous(비동기적), Periodic(주기적)
Frequency	Activation Type이 Periodic일 경우의 실행 빈도.
Priority	Transformation의 우선순위. TC(Time-Critical), NTC(Non-Time Critical)

[표 1] transformation에 대한 STS

3.2 태스크 구성 정보

표 1은 태스크가 가지는 정보를 나타내며 각 속성은 다음과 같이 기술한다.

속 성	설 명
Task ID	태스크 식별 ID
Task Name	태스크 이름
Task Description	태스크에 대한 설명
Task Type	태스크의 유형. ADIOTask, PIOTask, RMTask, PTask, ATask, CTRLTask, URTask
Frequency	주기적 실행 태스크의 실행 빈도
Priority	태스크의 우선순위 TC(Time Critical), NTC(Non Time Critical)

[표 2] 태스크 구성 정보

1) Task ID, Name, Description

Task에 대한 식별 ID와 이름, 설명을 지정한다.

2) Task Type

CODARTS에서는 태스크에 대한 유형을 7가지로 분류해 놓았다. 각 태스크의 유형은 다음과 같다.

Type	해당 태스크
ADIOTask	Asynchronous Device I/O Task
PIOTask	Periodic I/O Task
RMTask	Resource Monitor Task
PTask	Periodic Task
ATask	Asynchronous Task
CtrlTask	Control Task
URTask	User Role Task

[표 3] 태스크의 유형

각 태스크 유형에 대한 구조화 방법은 3.3에 기술한다.

3) Frequency

주기적 실행 특성을 가지는 태스크의 경우, 그 실행 빈도를 나타낸다. 주기적 성질을 가지지 않는 비동기 태스크의 경우에는 Frequency값을 가지지 않는다.

4) Priority

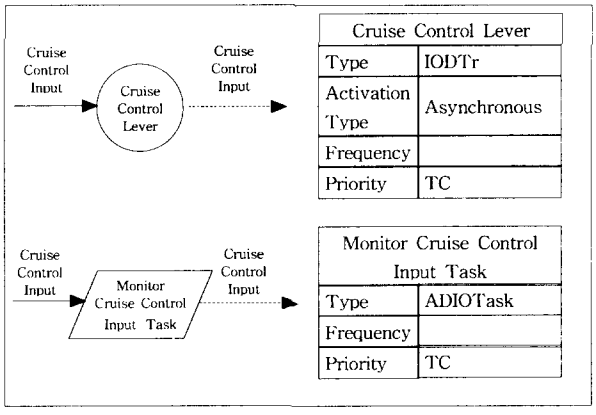
태스크의 type이 ADIOTask, PIOTask, CTRLTask인 경우는 외부 장치와 직접 입출력을 수행하거나 프로그램 흐름을 제어하므로 TC(Time-Critical)값을 가지고 PTask, ATask, URTask인 경우는 계산 위주의 작업을 수행하므로 NTC(None-Time-Critical)값을 가진다. Priority는 태스크간 인터페이스에서 약한 결합 메시지의 Queue사용 방식 결정에 사용된다.

3.3 태스크 정보의 추출

먼저 태스크 유형을 결정한 다음, 필요한 속성들의 값을 구성한다.

1) Asynchronous Device I/O Task

행동 모델의 transformation 활성화 형태가 Asynchronous이고 type이 IODTr인 경우 Asynchronous I/O Device Task로 구조화된다. 이러한 태스크는 device 입력의 인터럽트에 의해 활성화되며 주기적 성질을 가지지 않으므로 frequency값을 가지지 않는다.



[그림 2] Asynchronous Device I/O Task 구조화 예

2) Periodic I/O Task

Transformation의 활성화 형태가 Periodic이고 type이 IODTr이면 Periodic I/O Task로 구조화된다. 이 때 transformation의 실행 빈도가 태스크 frequency 값이 된다. 만약 시간 응집도가 적용된 경우라면 가장 낮은 실행 빈도를 가진 transformation의 frequency 값이 해당 태스크의 frequency값으로 지정된다.

3) Resource Monitor Task

여러 개의 장치들로부터 입력을 받아들이거나 출력을 내보내는 transformation의 경우, 데이터 무결성을 유지하거나 데이터 값이 손실되는 것을 막기 위해 Resource Monitor Task로 구조화된다.

4) Periodic Task

활성화 형태가 Periodic이고 type이 INTr인 transformation은 Periodic Task로 구조화되며 transformation의 frequency가 태스크 frequency값으로 지정된다. Periodic I/O task와 마찬가지로 시간 응집도가 적용된 경우라면 가장 낮은 실행 빈도를 가진 transformation의 frequency가 태스크의 frequency값이 된다.

5) Asynchronous Task

Transformation의 활성화 형태가 Asynchronous이고 type이 INTr이면 Asynchronous Task로 구조화된다. Asynchronous I/O Task가 외부 장치의 인터럽트에 의해 활성화되는데 반해, Asynchronous Task는 다른 태스크에 의해 전송된 내부 이벤트 또는 메시지에 의해 활성화된다.

6) Control Task

Transformation의 type이 COTr이라면 Control Task로 구조화된다.

7) User Role Task

사용자가 직접 수행하는 작업을 나타내는 transformation의 경우 User Role Task로 구조화된다. 하지만 STS에는 이 태스크를 위한 정보가 존재하지 않으므로 Transformation type에 URTr(User Role Transformation)특성을 추가해야 한다.

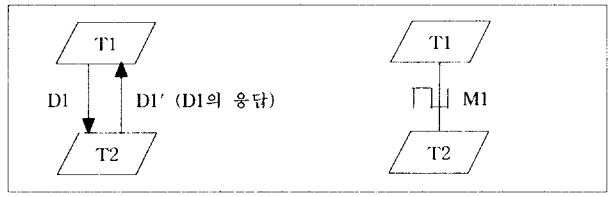
4. 태스크 인터페이스 구성

태스크 구조화가 완료고 나면, 태스크 인터페이스가 정의되어야 한다. 구성된 태스크는 먼저 data flow와 event flow 인터페이스를 가지는 preliminary TAD에 표현되며, 메시지 통신과 이벤트로 표현되는 태스크 인터페이스로의 변환을 거쳐 TAD에 표현된다.

4.1 메시지 통신

Data flow는 두 태스크간의 메시지 통신으로 변환될 수 있다. 메시지 통신은 약한 결합 또는 강한 결합 형태를 가진다. 약한 결합 메시지 통신은 송신 태스크에서 메시지를 보낸 후 수신 태스크의 응답을 기다리지 않거나, 응답이 도착하기 전에 계속적으로 다른 작업을 수행하는 경우에 사용된다. 강한 결합 메시지 통신은 송신 태스크에서 메시지를 보낸 후 수신 태스크로부터 응답 메시지가 도착할 때까지 블록 상태로 대기하는 경우 사용된다. Data flow를 메시지 통신으로 변환하기 위해서는 그의 응답에 해당하는 data flow를 명시해야 한다.

두 태스크 사이에 연결된 data flow에 대한 응답 data flow가 있을 경우 두 태스크간의 인터페이스는 강한 결합 메시지 통신으로 변환된다.



[그림 3] 강한 결합 메시지 통신 구성 예

만약 응답 data flow가 존재하지 않는다면 약한 결합 메시지 통신으로 변환된다. 약한 결합 메시지 통신으로 변환될 경우, 메시지를 보내는 태스크가 여러 개 있고 그 태스크 중 Priority가 TC인 태스크가 있다면, 메시지 처리에 Priority Queue가 사용될 수 있다. 그렇지 않다면 FIFO Queue가 사용될 수 있다.

4.2 이벤트

이벤트는 태스크를 활성화시키는 방법으로 사용되며 메시지와는 다르게 데이터를 가지지 않는다.

event flow의 source와 destination이 모두 task라면 Internal event로 변환되고 source가 만약 외부 입출력 장치와 같은 외부 개체라면 external event로 변환된다. 태스크의 type이 Periodic이라면 Timer event가 자체적으로 그 태스크에 입력되게 된다.

5. 결론 및 향후 연구 방향

본 논문에서는 태스크의 행동 특성을 표현할 수 있는 구조와 행동 모델로부터 그러한 정보를 추출하는 방법을 제안하였다. 이렇게 추출된 정보를 이용하면 마지막 구현단계를 쉽게 수행할 수 있을 것이다.

향후 연구 방향은 COBRA 방법론에 의한 행동 모델로부터 병렬 태스크를 자동으로 구성하는 CASE 도구를 설계하고 구현하는 것이다.

6. 참고 문헌

[1] Gomaa. H., Software Design Methods for Concurrent and Real-Time System, ADDISON-WESLEY, 1993.

[2] Gomaa, H., "Structuring Criteria for Real-time System Design" Proceedings of the Eleventh International Conference on Software Engineering, pp.290-301, 1989.

[3] 엄진아, 김규년, "CODARTS 방법론의 태스크 구성 지침을 적용한 태스크 자동 구성", 울산대학교 공학 연구 논문집, 1997