공학 연구논문집 제28권 제2호 pp. 93~116, 울산대학교.
Journal of Engineering Research Vol. 28, No. 2, pp. 93~116, 1997
University of Ulsan.

CODARTS 방법론의 태스크 구성 지침을 적용한 태스크 자동 구성

엄진아 · 김규년
컴퓨터 · 정보통신공학부

<요 약>

Gomaa에 의해 제안된 CODARTS(COncurrent Design Approach for Real-Time System) 방법론은 COBRA(Concurrent Object-based Real-Time Analysis)를 사용하여 실시간 시스템의 행동 모델을 개발하고 행동 모델에서 병렬 태스크 구조로 변환하는데 도움을 주는 태스크 구성 지침을 제공한다. CODARTS에서는 태스크 구성 지침을 RTSA 표기법으로 표현된 행동 모델에 적용하여 태스크를 구성한다. 그러나 RTSA 표기법에는 태스크를 구성하는데 필요한 실행 성질 및 주기, 이벤트의 병렬성 및 순서와 같은 정보들을 기술하지 않기 때문에 직접 태스크 구성 지침을 적용하는 것은 어렵다. 본 논문에서는 CODARTS 방법론의 분석 단계에서 태스크 구성에 필요한 정보를 통합적으로 나타낼 수 있는 STS(Specification for Task Struturing)를 제안하고 이를 이용하여 태스크를 자동적으로 구성할 수 있는 알고리즘과 구현 예제를 보인다.

Automatic Task Structuring using Task Structuring Criterias of CODARTS Methodology

Eom, Jin-Ah . Kim, Kyoo-Nyun
School of Computer Engineering and Information Technology

<Abstract>

CODARTS(COncurrent Design Approach for Real-Time System) made by Gomaa models behavioral model of real-time system using COBRA(Concurrent Object-based

Real-Time Analysis) and provides task structuring criteria that assists in mapping a behavioral model of the system into a concurrent-tasking architecture. In CODARTS, task structuring starts with a behavioral model represented using RTSA notation. But applying directly task structuring criteria to RTSA notaion is difficult because RTSA notation doesn't represent activation characteristic and frequency of transformation, concurrency and sequence of event/data flow as information needed task structuring. In this study, we propose STS(Specification for Task Structuring) that can represent all informations for task structuring in analyzing steps of CODARTS and show an algorithm and example using STS for automatic task structuring.

1. 서 론

실시간 시스템은 embedded system, 외부 환경과의 상호작용, 시간제한, 실시간 제어, reactive system과 같은 성질을 가지고 있어서 소프트웨어 설계에서 concurrent task를 적용하면 그 모델은 실세계와 유사하고, 각 태스크의 역할이 분명해지고, 전체 시스템의 실행시간 감소, 시스템 개발의 초기 단계에서 스케쥴링과 성능 분석을 할 수 있다. 실시간 시스템의 설계 방법론 중에서 태스크 구성을 강조하는 CODARTS 방법론이 있다.

CODARTS(COncurrent Design Approach for Real-Time System)는 COBRA(Concurrent Object-based Real-Time Analysis)의 시스템 분석 결과인 data/control flow diagram[1][2]에서 태스크 구성 지침(task structuring criteria)[1][4]을 적용하여 병렬 태스크로 구성한다. 태스크 구성의 핵심은 시스템의 동작 상황을 분석해서 실행의 병렬성을 찾고 병렬 실행이 필요한 것은 태스크로 구성하고 병렬성을 가지지 않는 함수나 객체는 하나로 묶어야 한다. 그러나, 시스템 분석의 결과인 data/control flow diagram에서 이러한 병렬성을 찾아낼 수 없다.

본 논문에서는 태스크 구성을 자동적으로 할 수 있도록 system context diagram, state transition diagram, data/control flow diagram에서 태스크 구성 지침의 적용에 필요한 실행 성질 및 주기, 이벤트의 병렬성 및 순서와 같은 정보들을 추출한 분석자료, STS(Specification for Task Struturing)를 제안하고 이를 사용하여 태스크를 자동적으로 구성할 수 있는 태스크 구성 알고리즘과 구현 예제를 보인다.

2. CODARTS 방법론

2.1 COBRA

CODARTS는 COBRA를 사용하여 시스템과 상호작용하는 외부 환경과 시스템의 행동을 분석하여 RTSA(Real-Time System Analysis) 표기법으로 표현하고 data/control flow diagram에서 태스크 구성 지침을 사용하여 병렬 태스크로 구성한다.

특히, COBRA는 부 시스템 구성 지침(Subsystem structuring criteria), 객체 구성 지침(Object structuring criteria), 함수 구성 지침(Function structuring criteria)을 제공하여 문제 영역에서 시스템을 부 시스템으로 나누고, 부 시스템에서 객체와 함수를 결정하는데 도움을 준다. 이러한 구성 지침들에 의하여 태스크 구성에 필요한 transformation의 기능적 연관성과 실행 성질에 관한 정보를 분석 단계에서 얻을 수 있다.

Criteria	성 질	표 현
External Device I/O Object	SCD에서 I/O device로 표현되는 terminator에 대한 객체	Data Transformation
User Role Object	SCD에서 시스템과 상호작용하는 사용자의 종류에 따른 서비스를 제공하는 객체	〃
Control Object	여러 state를 가지고 다른 object와 function들의 행동을 제어하는 active abstract 객체	Control Transformation
Data Abstraction Object	데이터를 캡슐화한 passive 객체	Data Transformation
Algorithm Object	알고리즘을 캡슐화한 객체	〃

[표 2.1] Object Structuring Criteria

Criteria	Activate	표 현
Function Supported by object	object	Data Transformation
Asynchronous Function	obejct function	〃
Asynchronous State Dependent Function	control obejct	〃
Periodic Function	timer event	〃
Periodic State-Depedent Function	control object	〃

[표 2.2] Function Structuring Criteria

2.2 태스크 구성 지침

태스크 구성은 시스템을 병렬 태스크로 구성하고 그들의 인터페이스를 정의하는 것이다. CODARTS에서는 COBRA의 행동 모델에서 객체나 함수로 표현되는 data/control transformation을 분석하여 병렬성을 가지는 것을 병렬 태스크로 구성한다. 병렬 시스템에서 태스크의 수가 많으면 태스크간의 통신과 동기화에 추가 부담이 발생함으로 너무 많은 태스크가 생기지 않도록 조절할 필요가 있다. Gomaa[1]는 너무 많은 태스크가 생기지 않도록 시스템의 행동 모델에서 병렬 태스크 구조로 사상시키도록 태스크 구성 지침을 제안하였다. 태스크 구성 지침에는 4 개의 범주가 있다.

1. I/O 태스크 구성 지침
2. 내부 태스크 구성 지침
3. 태스크 응집도에 대한 구성 지침
4. 태스크 우선 순위에 대한 구성 지침

2.2.1. I/O 태스크 구성 지침

I/O 태스크 구성 지침에는 3 가지의 세부 구성 지침이 있다. 시스템과 상호 작용하는 I/O 디바이스 객체에 대하여 디바이스의 성질에 따라 디바이스당 하나의 태스크를 구성한다. 디바이스가 인터럽트에 의해 비동기적으로 실행된다면 asynchronous device I/O 태스크로 구성하고 타이머 이벤트에 의해 활성화되면 periodic I/O 태스크로 구성한다. I/O 디바이스가 여러개의 태스크로부터 요구를 받아 처리한다면 이러한 요구의 순서를 조절하여 데이터의 손실을 막아주는 특별한 태스크가 필요하다. 이러한 태스크를 resource monitor 태스크로 구성한다.

지침	활성화 조건	적용 대상
Asynchronous Device I/O Task	인터럽트	I/O 디바이스 객체
Periodic I/O Task	타이머	〃
Resource Monitor Task	여러 곳에서 요구를 받아 처리할 경우	〃

[표 2.3] I/O 태스크 구성 지침

2.2.2. 내부 태스크 구성 지침

내부 태스크 구성 지침에는 4 가지의 세부 구성 지침이 있다. I/O 디바이스 객체를 제외한 모든 객체나 함수에 Internal task structuring criteria를 적용하여 태스크로 구성한다. 활성화 이벤트의 성격에 따라 periodic 태스크와 asynchronous 태스크로 구성하고 행동 모델의 제어 객체당 하나의 control 태스크를 할당한다. user role 객체는 user role 태스크로 구성한다.

지침	활성화 조건	적용 대상
Periodic Task	타이머	internal object, function
Asynchronous Task	내부 메시지, 이벤트	"
Control Task		control object
User Role Task		user role object
Multiple tasks of same type		object

[표 2.4] 내부 태스크 구성 지침

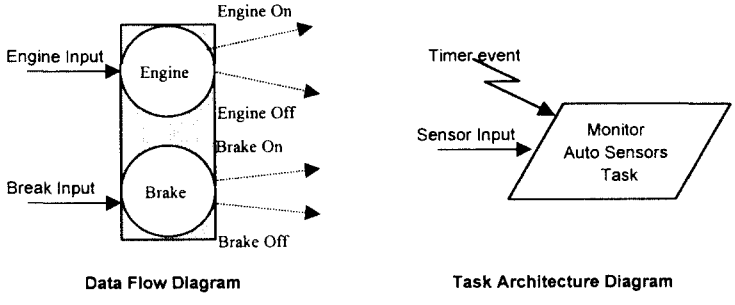
2.2.3. 태스크 응집도에 대한 구성 지침

CODARTS에서 응집도는 일반적인 모듈내의 함수의 응집도와는 다르게 행동 모델에서 tranformation으로 표현되는 어떤 객체나 함수들을 하나의 태스크로 묶어야 할지를 결정하는데 사용된다. 태스크 응집도에 관한 세부 지침으로 3 가지가 있다.

시간 응집도 지침(Temporal cohesion criteria)

같은 이벤트에 의해 활성화되는 2 개이상의 transformation이 순차적으로 실행되지 않는다면 이들을 하나의 태스크로 묶는다. 특히, 타이머 이벤트에 의해 활성화되는 periodic transformation 중에서 주기가 같은 것을 하나의 태스크로 구성한다.

자동 속도 조정 시스템에서, Engine과 Brake는 주기적 입력 디바이스로 실행 빈도가 100 ms로 같으므로 디바이스당 하나의 태스크로 구성하지 않고 temporal cohesion criteria

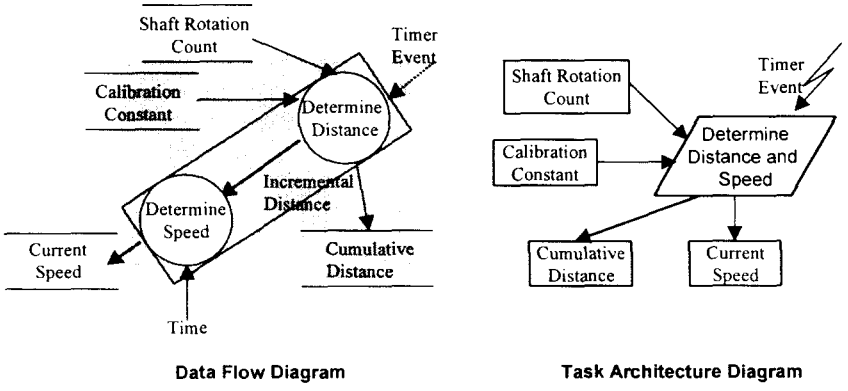


[그림 2.1] 시간 응집도를 적용한 예제

를 적용하여 함께 묶어서 하나의 태스크로 구성한다.

순서 응집도 지침(Sequential cohesion criteria)

비동기적 또는 주기적인 이벤트에 의해 활성화된 transformation부터 순차적으로s 실행



[그림 2.2] 순서 응집도를 적용한 예제

되는 모든 transformation을 하나의 태스크로 묶는다. 그룹화시킬 태스크에 data store에 데이터를 저장하는 transformation을 마지막으로 포함시킨다. 만약 다음 transformation이 우선순위가 높거나 다른 transformation의 이벤트에 의해 활성화될 가능성이 있다면 같이 묶지 않는다.

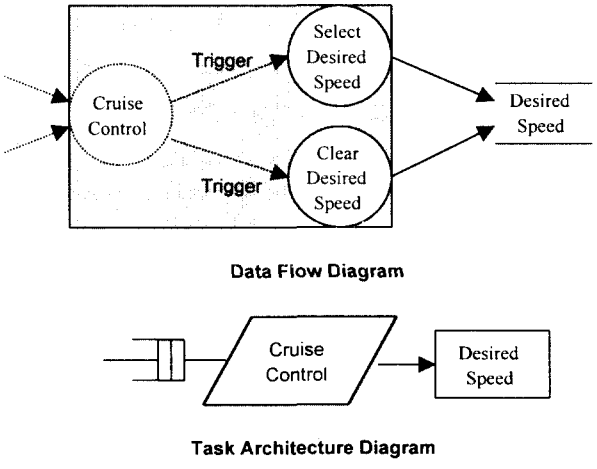
자동 속도 조정 시스템에서, Determine Speed는 Determine Distance로부터 전달받은 데이터를 사용하여 Current Speed를 계산하여 Current Speed data store에 저장한다. Determine Distance와 Determine Speed는 sequential cohesion criteria를 적용하여 하나의 태스크로 묶는다.

제어 응집도 지침(Control cohesion criteria)

제어 객체는 하나의 태스크로 구성하고 그것에 의해 동작하는 함수나 객체중에서 병렬성을 가지지 않는 것은 하나의 태스크로 묶는다. 제어 객체의 상태 의존적인 data transformation 중에서 trigger 이벤트에 의해 활성화되는 것은 하나의 태스크로 묶고, enable/disable 이벤트에 의해 활성화되는 것은 다른 태스크로 구성한다. trigger 이벤트에 의해 활성화되는 transformation은 상태전이 동안에 실행을 시작, 완료함으로 control transformation과 병렬적으로 실행될 필요가 없음으로 함께 묶을 수 있다. 반면에, enable/disable 이벤트에 의해 실행되는 transformation은 전이된 상태동안 계속해서 실행함으로 control transformation과 병렬적으로 실행함으로 다른 태스크로 구성한다.

대 상	조 건	적 용
State dependent data transformation	trigger 이벤트에 의해 활성화되고 상태전이 동안 실행하는 경우	control transformation과 같이 하나의 태스크로 구성
State dependent data transformation	trigger 이벤트에 의해 활성화되고 한 상태 동안 실행하는 경우	다른 태스크로 구성
State dependent data transformation	enabled/disable 이벤트에 의해 활성화되는 경우	다른 태스크로 구성
Non-state dependent data transformation	control transformation에 상태전이를 일으키는 유일한 이벤트를 보내는 경우	control transformation과 같이 하나의 태스크로 구성

[표 2.5] 제어 응집도 지침



[그림 2.3] 제어 응집도를 적용한 예제

자동 속도 조정 시스템에서, Select Desired Speed와 Clear Desired Speed는 상태 의존적인 함수로 상태 전이가 일어나는 동안에 시작하여 실행을 완료함으로 Cruise Control과 병렬적으로 실행될 필요가 없으므로 묶어서 하나의 태스크로 구성한다.

기능 응집도 지침(Functional cohesion criteria)

기능적으로 관련이 있는 함수나 객체를 묶어 하나의 태스크로 구성하는 functional cohesion criteria는 태스크 응집도가 가장 낮은 구성지침으로 다른 cohesion criteria와 함께 적용된다.

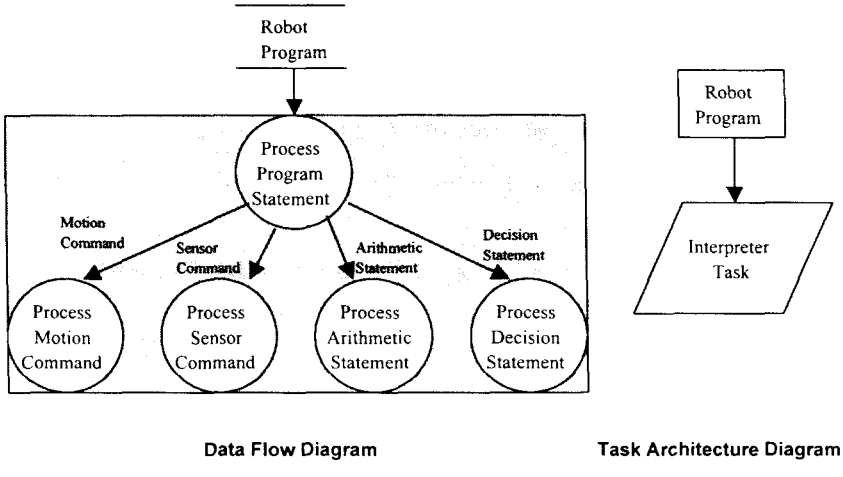
구성 지침	조 건	적용 대상
sequentially related functional cohesion	두 함수간의 데이터 교환이 많은 경우	data transformation
clustered functional cohesion	같은 데이터와 I/O 디바이스를 엑세스하는 경우(모듈 구성화에 사용)	
temporally related functional cohesion	temporal and functional cohesion	

[표 2.6] 기능 응집도 지침

로봇 제어 시스템에서, Process Program Statement는 로봇 프로그램을 읽어 명령문의 종류에 따라 적당한 인터프리터(Process Motion Command, Process Sensor Command, Process Arithmetic Statement, Process Decision Statement)에 전달한다. 이러한 5 개의 transformation은 차례대로 실행하면서 기능적 관련성도 있으므로 sequentially related functional cohesion criteria를 적용하여 Interpreter 태스크로 구성한다.

2.2.4. 태스크 우선순위에 대한 구성 지침

우선순위는 개발단계에서 고려하지만 태스크 구성단계에서는 time critical 태스크와 non-time critical한 것으로 구분하여 같이 묶지 않는다. I/O 인터럽트를 다루거나 마감시간안에 실행을 완료해야만 하는 것은 time critical한 태스크로 분류한다.



[그림 2.4] 기능 응집도를 적용한 예제

2.3. 태스크 구성 방법

2.2 절의 태스크 구성 지침을 다음과 같은 순서에 따라 COBRA의 행동 모델의 data/control transformation에 적용하여 태스크로 구성한다.

1. I/O transformation에 대한 태스크 구성
   - I/O 태스크 구성 지침에 따라 비동기 I/O 디바이스, 주기적 I/O 디바이스, 자원 감독자당 하나의 태스크를 구성한다. 주기적 I/O 디바이스중에서 실행 빈도가 같은 것들은 temporal cohesion criteria를 적용하여 하나의 태스크로 구성한다.
2. Control transformation에 대한 태스크 구성
   - 각 control transformation을 하나의 태스크로 구성하고 control transformation과 연관된 data transformation을 분석하여 control cohesion criteria를 적용하여 같은 태스크로 묶는다.
3. Periodic transformation에 대한 태스크 구성
   - Periodic transformation당 하나의 태스크를 구성하고 이들 중에서 실행빈도가 같은 것은 temporal cohesion criteria를 적용하여 묶는다. 실행빈도가 다른 transformation은 sequential cohesion criteria를 적용하여 하나의 태스크로 묶는다.
4. 나머지 내부 transformation에 대한 태스크 구성
   - 나머지 내부 transformation에 대하여 temporal, sequential, functional cohesion criteria를 적용하여 하나의 태스크로 묶는다.

3. STS(Specification for Task Structuring)

이 장에서는, 태스크 구성 지침을 적용하는데 필요한 정보를 기술하는 STS(Specification for Task Structuring)를 정의하고 이것을 사용하여 태스크를 자동적으로 구성할 수 있는 알고리즘을 제안한다.

3.1 STS의 개요

태스크 구성 지침을 적용하기 위해서는 data/control flow diagram의 각 transformation들의 실행 성질이 주기적인지 비주기적인지에 대한 정보, 실행 빈도, 우선 순위, 실행 순서와 데이터 의존도 등의 정보가 필요하다. 그러나, RTSA 표기법에서는 이러한 정보를 표현하는 다이아그램이나 결과물이 없다.

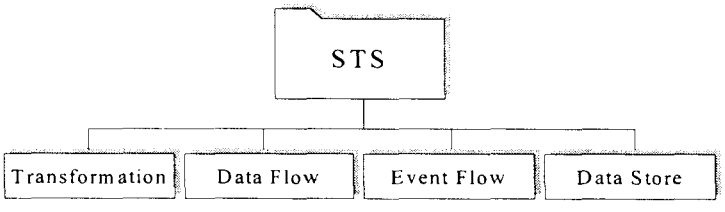
STS는 태스크 구성에 필요한 정보들을 기술하는 파일로서 COBRA의 각 단계의 결과물에서 얻을 수 있는 정보[표 3-1]를 추출하여 STS에 통합적으로 나타낼 수 있고 이것을 바탕으로 자동적으로 태스크로 구성할 수 있다. [표 3-1]은 COBRA의 각 분석 결과물에서 찾을 수 있는 태스크 구성 정보를 나타낸 것이다.

다이아그램	태스크 구성 정보
System Context Diagram	I/O device의 동작 성질, 실행 빈도, 외부 입력과 출력 event
State Transition Diagram	control object와 state-dependent function들의 실행 순서, event 발생의 병렬성, event 발생 순서
Data/Control flow Diagram	각 transformation들의 동작 성질, data dependency, sequential dependency
Event Sequence Diagram	각 event들의 발생 순서

[표 3-1] COBRA의 결과물에서 추출할 수 있는 태스크 구성 정보

3.2 STS의 정의

[그림 3.1]과 같이 STS는 data/control flow diagram의 transformation, event flow, data flow, data store에 대한 정보를 가진다. STS는 각 구성 요소마다 태스크 구성에 필요한 속성과 속성값으로 정의할 수 있다.



[그림 3.1] STS의 구성 요소

3.2.1 Transformation

속 성	설 명
Transformation ID	transformation의 이름
Transformation description	transformation에 대한 설명
Transforamtion type	transforamtion의 종류 (IDTr, ODTr, SDTr, INTr, COTr)
Activation type	transforamtion의 동작 방식 (Asynchronous/Periodic)
Frequency	주기적 I/O 디비이스의 실행 빈도
Execution time	transformation의 실행 시간
Priority	transformation의 우선순위(TC/NTC)
Activating input	transformation의 실행을 활성화시키는 input event 또는 data flow ID
Input event flows	input event flow 집합
Input data flows	input data flow 집합
Output event flows	output event flow 집합
Output data flows	output data flow 집합

[표 3-2] Transformation의 STS 정보

[표 3-2]는 transformation에 해당하는 STS의 정보를 나타내고 각 속성은 다음과 같이 기술한다.

Transformation ID, Transformation Description

이름과 transformation에 대한 설명, 실행 방법, 관련된 data들을 자유형식으로 기술한다. 태스크 구성 지침을 직접 적용하는데 사용되지 않지만 transformation을 이해하는데 도움을 준다.

Transformation type

Transformation type은 태스크로 구성할 수 있는 기본 transformation을 5 가지로 분류한 것이다. 이렇게 분류한 기본 transforation부터 태스크로 구성하고 관련된 transformation들은 응집도에 대한 구성지침을 적용하여 묶어서 하나의 태스크로 구성할 수 있다.

system context diagram에서 외부 입력 I/O 디바이스 개체를 표현되는 data

transformation을 IDTr(Input Device Transformation)과 ODTr(Output Device Transformation), control transformation을 COTr(Control Transformation), control transformation이 발생시키는 event에 의해 실행되는 data transformation을 SDTr(State dependent Transformation), 나머지 data transformation을 INTr(Internal Transformation)로 분류한다.

Activation type

Activation type 속성은 transformation의 실행 성질에 따라 A(asynchronous)와 P(periodic) 두 가지로 구분하여 정의한다. 태스크 구성 지침에서는 주기적 transformation을 하나의 태스크로 구성함으로 동작 성질을 분류하는 것이 구성 지침을 적용하기가 쉽다. 센서를 나타내는 I/O 디바이스 transformation이나 타이머 이벤트에 의해 실행되는 transformation의 Activation type 속성값은 P가 된다.

Frequency, Execution time

실행 빈도와 실행 시간을 나타내는 것으로 Frequency는 주기적 transformation에 대하여 temporal cohesion criteria를 적용하는데 필요하고 Execution time은 태스크를 구성하는데 직접 사용되지 않지만 태스크를 구성한 후, 성능 분석에 사용된다. IDTr이나 ODTr의 경우는 외부 I/O device 하드웨어의 성질에 따라, SDTr과 INTr의 경우는 Activating input의 성질에 따라 transformation의 동작을 분석하여 기술한다.

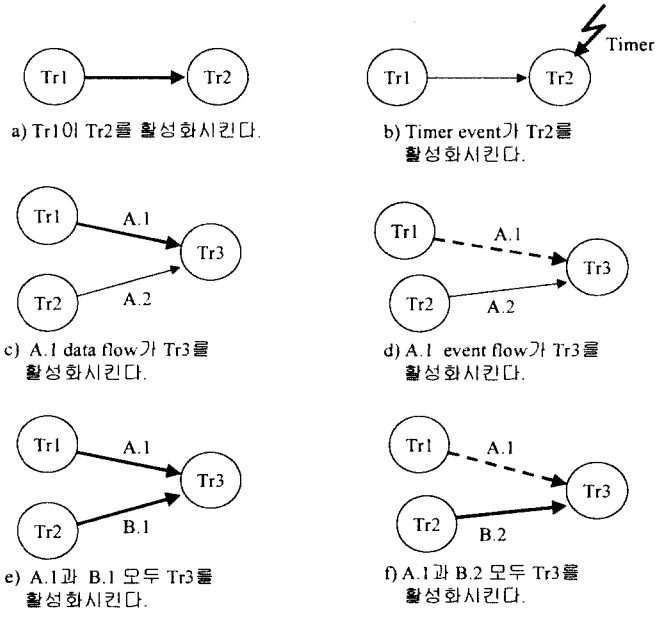
Priority

설계 단계에서 Priority는 TC(time-critical)/NTC(non-time-critical) 두 가지로 구분하여 정의한다. COTr, IDTr, ODTr은 TC를, 계산 위주인 SDTr이나 INTr에는 NTC을 지정한다. 각 transformation을 우선순위면에서 분류하여 우선 순위가 높은 것을 우선 순위가 낮은 것과 함께 태스크로 묶지 않는데 사용되는 정보이다.

Activating input

Activating input은 transformation의 입력중에서 transformation을 실행시키는 event/data flow이다. Activating input은 INTr의 동작 성질을 분석하고 sequential cohesin criteria을 적용하는데 도움을 준다. sequential cohesion criteria에서 순차 실행 의존도를 가지는 transformation을 찾을 때, Activating input에 의해 실행하는 transformation부터 순차적 실행 순서에 먼저 포함시킨다.

[그림 3-2]은 Activating input 속성값을 결정하는 방법을 보여준다. b)경우는 transformation의 Activation type이 periodic이면 Activating input은 timer event이다. 한 개의 input만을 가진다면 그 input을 Activating input으로 정한다. 다수의 input이 존재한다면 sequence number에서 외부 이벤트가 같은 것 중에서 event flow를, 모두 data flow



[그림 3.2] ***Activating input*** 속성 결정방법

이면 실행순서가 빠른 것을 Activating input으로 정한다. e)의 경우, A.1과 B.1은 실행 순서는 1이지만 외부 이벤트가 A와 B로 달라서 다른 순차적 실행 순서를 가지므로 A.1과 B.1은 모두 Tr3의 Activating input이 된다.

Input/Output Data Flow, Input/Output Event Flow

입/출력 data/event flow들의 ID를 기술함으로써 control/data flow diagram에서 표현된 transformation과 data/event 사이의 관계를 알 수 있다.

3.2.2 Event flow

속 성	설 명
Event flow ID	event flowID
Event flow description	event flow에 대한 설명
Event flow type	event의 종류 Trigger(TREv), Enable(ENEv), Disable(DSEv), Internal(INEv), Timer(TMEv)
Sequence number	event flow의 발생 순서
Source	event를 보내는 transformation ID
Destination	event를 받는 transformation ID

[표 3-3] Event flow의 STS 속성 정보

event flow는 control/data flow diagram에서 데이터를 가지지 않는 신호로 표현된다. event flow의 성격이나 발생 순서와 같은 정보는 event flow를 받거나 생성시키는 transformation의 동작 성질을 정하거나 sequential cohesion criteria를 적용하는데 도움을 준다. [표 3.3]는 event flow에 필요한 STS 속성을 나타낸다.

Event flow ID, Event flow Description

이름과 event flow에 대한 설명을 기술한다. Event flow ID는 transformation의 STS에서 Input/Ouput Event Flow의 속성을 지정할 때 사용된다.

Event flow type

Event flow type의 속성값으로 TREv(Trigger), ENEv(Enable), DSEv(Disable), INEv(Internal), TMEv(Timer)를 지정할 수 있다. Event flow type 속성값은 제어 응집도 지침나 순서 응집도 지침을 적용하는데 사용된다.

Sequence number

Sequence number는 이벤트의 발생 순서를 나타내는 번호이다. sequence number는 시스템의 실행 과정을 이해하거나 순서 응집도 지침을 적용하여 차례대로 실행되는 transformation을 찾는데 도움을 주는 정보이다. STS의 Sequence number는 이벤트의 발생 성질과 순서를 함께 나타낼 수 있는 번호 지정법을 3.3 절에서 소개하고 3.4 절에서는 이를 사용하여 순서 응집도를 측정하는 방법을 설명한다.

Source, Destination

Source와 Destination 속성값으로 data/control flow diagram의 정보를 바탕으로 이벤트의 근원지와 목적지의 transformation ID를 지정한다.

3.2.3 Data flow

data/control flow diagram에서 data flow는 transformation의 데이터 처리과정이나 순서 응집도 지침을 적용하는데 도움을 준다. [표 3-4]는 data flow의 STS로 event flow의 STS 속성이 같지만 다른 속성값을 가진다.

속 성	설 명
Data flow ID	data flow ID
Data flow description	data flow에 대한 설명
Data flow type	data flow의 종류, Input Device(IDDa), Output Device(ODDa), Data store(DSDa), Transformation(TRDa), External(EXDa)
Sequence number	data flow의 발생 순서
Source	transformation/data store ID
Destination	transformation/data store ID

[표 3-4] Data flow의 STS 정보

Data flow ID, Data flow Description, Data flow type

Data flow ID와 Data flow Description은 event flow의 STS 정보와 같다. Data flow type은 event flow와 다르게 자료의 목적지에 따라 분류한다. 목적지가 입력 디바이스이면 IDDa, 출력 디바이스이면 ODDa, transformation이면 TRDa, 데이터 저장소이면 DSDa로 정한다.

Sequence number

sequence number도 event flow의 STS와 같이 data의 발생 성질과 순서를 함께 나타낼 수 있는 번호 지정법을 3.3 절에서 소개하고 3.4 절에서는 이를 사용하여 순서 응집도를 측정하는 방법을 설명한다.

Source, Destination

Source와 Destination 속성값으로 data/control flow diagram의 정보를 바탕으로 자료의 근원지와 목적지를 기술하는데 event flow와 다르게 data store ID와 transformation ID를 지정할 수 있다.

3.2.4 Data store

data store와 관련된 정보는 태스크를 구성하는데 직접적으로 사용되지 않으므로 [표 3-5]와 같이 STS 정보를 간단하게 기술한다. Data Store description에는 자료 저장소에 대한 설명과 내부 자료 구조나 파일에 대한 정보를 기술한다.

속 성	설 명
Data Store ID	data store ID
Data Store description	data store에 대한 설명

[표 3-5] Data store의 STS 정보

3.3 Sequence numbering 방법

CODARTS에서는 data/control flow diagram의 event/data flow에 시스템이 외부 이벤트에 대하여 어떻게 동작하는지를 알 수 있도록 외부 이벤트에 반응하는 내부 이벤트에 순서를 부여한다. event flow나 data flow의 STS에서 Sequence number는 event sequence diagram에서 부여한 번호를 사용하지 않고 이벤트가 병렬적으로 발생하는지 순차적으로 발생하는지를 쉽게 알 수 있도록 새로운 번호 부여방법을 소개한다. 이렇게 부여한 번호는 각 transformarion들의 순서 실행 정도를 조사하여 태스크를 구성할 때 순서 응집도 지침과 제어 응집도 지침을 적용하는데 도움을 준다.

STS에서 사용하는 sequence number는 Activating_number와 Action_number로 구분한다. Activating_number는 한 transformaion을 활성화시키는 Activating input의 sequence number를 가르키고 Action_number는 Activating input에 의해 발생하는 출력 이벤트의 성질에 따라 순차적으로 발생하면 숫자번호를, 선택적으로 발생하면 알파벳 번호를 부여한다.

Sequence number : [Activating_number].[Action_number]

시스템에 들어오는 외부 이벤트부터 차례대로 다음 3가지 방법에 따라 data flow나 event flow에 Sequence number를 부여한다.

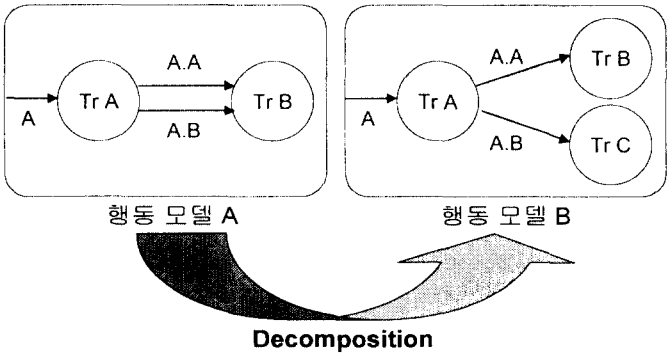
1. 시스템이 경험할 수 있는 외부 이벤트는 [Action_number]에 알파벳 번호를 부여한다. 실시간 시스템에서 외부 이벤트는 실세계의 I/O 디바이스 개체로부터 발생함으로 각 이벤트들은 독립적으로 발생하고 동시에 발생할 수 있다. 예) A.(Cruise Control Input), B.(Engin Input)

2. 이벤트 A의 부이벤트는 [Activating_number]에 A를, [Action_number]에 알파벳 번호를 부여한다. 부이벤트 A.A는 발생지가 이벤트 A와 같은 부분적인 내부 이벤트이다. A의 부이벤트인 A.A와 A.B는 선택 관계로 두 이벤트는 동시에 발생하지 않는다. 선택 관계에 있는 부이벤트는 조건에 따라서 하나의 부이벤트만 발생한다.

3. 활성화 이벤트에 의해 발생하는 출력 이벤트는 [Activating_number]에 활성화 이벤트의 번호를, [Action_number]에는 발생 순서를 따라 숫자 번호를 부여한다. 활성화 이벤트 A의 출력 이벤트인 A.1과 A.2에서 A.1이 A.2 보다 먼저 발생한다.

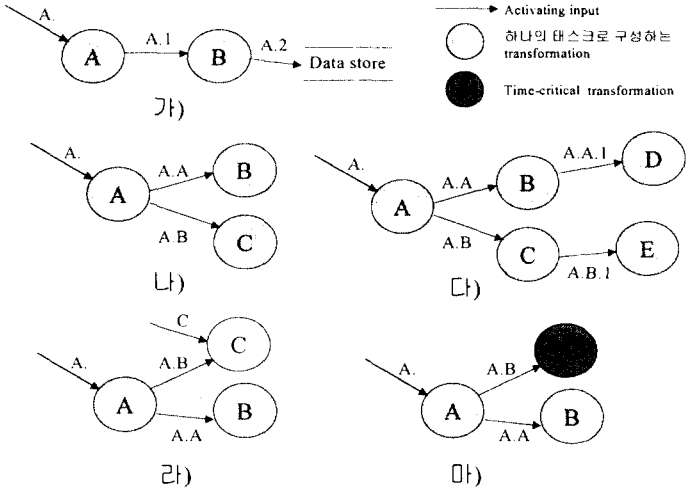
Sequence number를 지정하는 동안 [그림 3-2]와 같이 transformaion A에서

transformation B로 가는 event flow/data flow가 동시 발생 가능성이 있다면 transformation B내에서 병렬 실행이 발견됨으로 transformation B를 하위의 transformation으로 더 분해한다. 시스템의 행동 모델에서 가장 하위 data/control transformation들은 순차적으로 실행되는 객체나 함수로 표현되므로 한 transformation내에서 병렬 실행성이 존재하면 행동 모델을 다시 분석하여 하위 transformation으로 분해해야 한다.



[그림 3.3] transformation 분해

3.4 순서 응집도 측정 방법



[그림 3.4] 순서 응집도 적용 예제

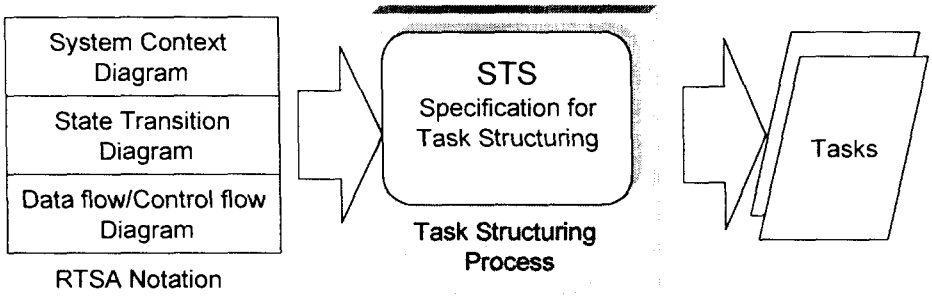
event/data flow의 sequence number 속성 값을 조사하여 event/data flow와 관련된 transformation들 중에서 순서 응집도를 가지는 것을 추출할 수 있다. transformation의 Activating input의 Sequence number가 [그림 3.4]과 같은 유형을 가지면 순서 응집도를

가진다.

가)와 나)에서와 같이 transformaion A의 출력 event/data flow의 Sequence number에서 [Activating_number]가 A로 같은 event/data flow의 목적 transformation들은 모두 순서 응집도를 가지므로 함께 묶을 수 있다. Sequence number의 [Action_number]가 숫자이면 순차적으로 실행되고 알파벳 문자이면 선택적으로 실행됨으로 병렬성을 가지지 않으므로 Sequence number에서 [Activating_number]가 같은 모든 transformation들은 함께 묶을 수 있다.

다)에서는 부이벤트 A.A와 A.B에 의해 활성화되어 순차적으로 실행하는 모든 transformation들은 병렬적으로 실행하지 않으므로 순차적 응집도를 가진다. 그러나, 라)처럼 Sequence number에서 같은 [Activating_number]를 가지지 않는 다른 event/data flow에 의해 활성화되는 transformation은 함께 묶지 않는다. 또, 순차적으로 실행되지만 마)처럼 다음 transformation의 [Priority]가 TC(time_critical)이면 함께 묶지 않는다.

3.5 STS를 사용한 태스크 구성 알고리즘



[그림 3.5] STS를 사용한 태스크 구성

data/control flow diagram의 transformation과 event/data flow에 관한 STS의 속성을 작성한 후, 다음의 태스크 구성 지침을 차례로 적용하여 병렬 태스크로 구성한다.

[1]. data/control flow diagram의 모든 Tr에 대해서 STS의 [Transformation type]의 속성값이 IDTr이거나 ODTr인 것을 대상으로 다음과 같이 병렬 태스크로 구성한다.

[1.1]. [1]에서 추출된 Tr에서 STS의 [Activation type] 속성값이 Asychronous인 것을 하나의 태스크로 구성한다.

[1.2]. [1]에서 추출된 Tr에서 STS의 [Activation type] 속성값이 Periodic인 Tr들을 찾는다.

[1.2.1]. [1.2]에서 추출된 Tr에서 STS의 [Frequency] 속성값이 같은 Tr들을 묶어서 하나의 태스크로 구성한다.

[1.2.2]. [1.2]에서 추출된 Tr에서 STS의 [Frequency] 속성값이 다른 각 Tr을 하나의 태스크로 구성한다.

[2]. data/control flow diagram의 모든 Tr에 대해서 STS의 [Transformation type]의 속성값이 COTr인 것을 하나의 control task T[i]로 구성한 후, 다음의 제어 응집도 지침을 만족하면 이 control task T[i]와 함께 묶는다.

[2.1]. STS의 [Transformation type] 속성값이 SDTr이고, [Activating input]의 속성값이 A 라면, event A의 [Event flow type] 속성값이 TREv인 모든 Tr에 대하여 [2.1.1]과 [2.1.2]을 따른다.

[2.1.1]. [2.1]의 Tr의 [Execution time]이 state transition 동안일 때, [2]의 control task T[i]에 결합시킨다.

[2.1.2]. [2.1]의 Tr의 [Execution time]이 state 동안 계속 실행될 때, [2]의 control task T[i]에 결합시키지 않는다. 왜냐하면 control task T[i]와 [2.1.2] 조건을 만족하는 Tr은 병렬하게 실행되기 때문에 다른 태스크로 구성해야 한다.

[2.2]. STS의 [Transformation type] 속성값이 SDTr이고, [Activating input]의 속성값이 A 라면, event A의 [Event flow type] 속성값이 ENEv이거나 DSEv인 모든 Tr에 대하여 control task T[i]와 다른 task T[j]로 구성한다.

[2.3]. [2.2]의 조건을 만족하는 task T[j] 중에서 같은 상태에서 실행되지 않는 task T[j]들을 묶어서 하나의 task로 구성한다.

[3]. STS의 [Transformation type] 속성값이 INTr이고, [Activation type]의 속성값이 Periodic인 모든 Tr에 대하여 [3.1]과 [3.2]의 지침을 따른다.

[3.1]. [3]의 조건을 만족하는 Tr의 [Frequency]의 속성값이 같으면, 하나의 task T[i]로 구성한다.

[3.1.1]. [3.1]에서 구성한 task T[i]에 속한 Tr들의 집합을 TrS로 한다. TrS[j]의 각 Tr에서 순서 응집도를 가지는 Tr을 task T[i]와 함께 묶는다.

[3.2]. [3.1]의 조건을 만족하지 않는 Tr을 독립적인 task T[i]로 구성하고, 이 Tr과 순서 응집도를 가지는 다른 Tr[j]이 있다면, task T[i]와 함께 묶는다.

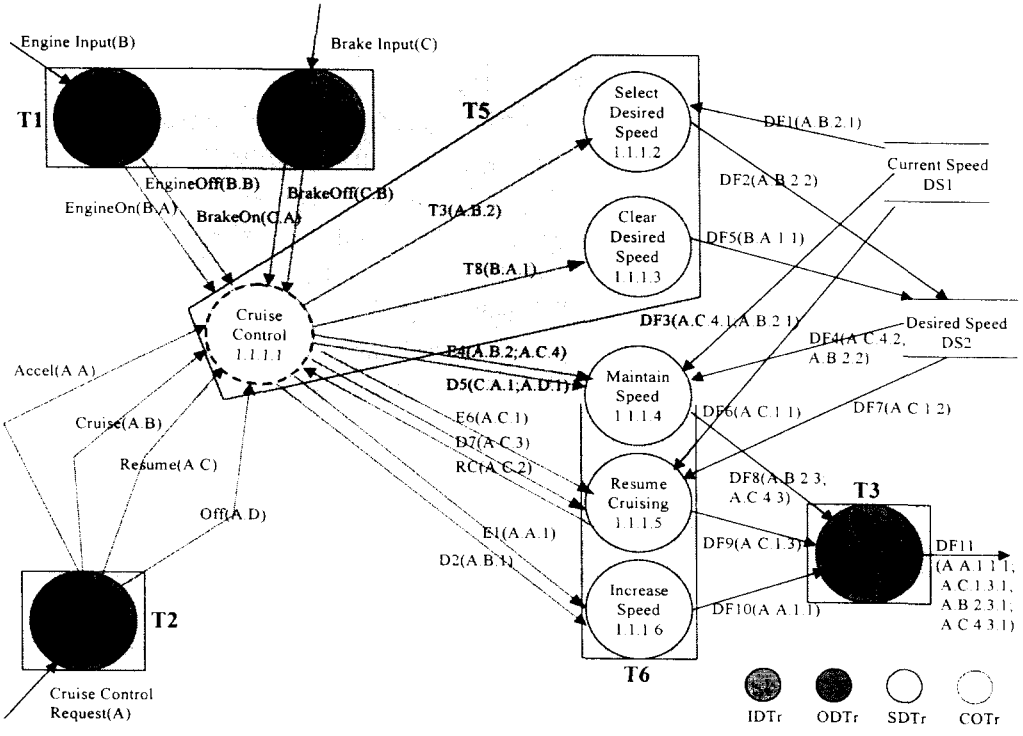
[4]. STS의 [Transformation type] 속성값이 INTr이고, [Activation type]의 속성값이 Asynchronous인 모든 Tr에 대하여 [4.1]과 [4.2]의 지침을 따른다.

[4.1]. [4]의 조건을 만족하는 Tr_P에 대하여 [Activating input] 속성값이 event A라면, event A의 [Source Transformation] 속성값 Tr_S의 [Transformation type] 속성값이 IDTr이거나 ODTr 이면, 이러한 Tr_P와 같은 [Activating input] 속성값을 가지는 모든 Tr을 Tr_S와 함께 묶어서 하나의 태스크로 구성한다.

[4.2]. 순서 응집도를 가지면, 하나의 태스크로 결합시킨다.

4. STS 적용 예제(자동 속도 조절 시스템)

이 장에서는 자동 속도 조절 시스템[1]의 행동 모델에서 태스크 구성에 필요한 정보들을 STS에 기술하고, 3.5 장에서 소개한 알고리즘을 적용하여 태스크를 구성한다.



[그림 4.1] STS를 적용한 태스크 구성 예제

[그림 4.1]은 STS에 따라 각 [Transformation type]과 data/event flow의 [Sequence number]를 data flow/control flow diagram에 나타낸 것이다. [Transformation type]이 IDTr인 Cruise Control Lever는 알고리즘 1.1에 의해 태스크 T2로, Engine, Brake는 알고리즘 1.2.1에 의해 태스크 T1으로 구성된다. Cruise Control을 알고리즘 2에 의해 태스크 T5 태스크로 구성한 후, Select Desired Speed와 Clear Desired Speed를 알고리즘 2.1.1에 의해 포함시켰다. Maintain Speed, Resume Cruising, Increase Speed는 알고리즘 2.2에 의해 각각 다른 태스크로 구성하지만 [Activating input]의 [Sequence number]를 조사해보면 A의 부이벤트인 A.A, A.B, A.C는 동시에 발생하지 않으므로 세 transformation이 병렬적으로 실행될 필요가 없기 때문에 알고리즘 2.3을 적용하여 하나의 태스크로 구성한다.

Event flow STS																	
Event flow ID	BO	BF	EO	EF	Accel	Cruise	Resume	Off	E1	D2	T3	E4	D5	E6	D7	T8	RC
Event flow description																	
Event flow Type	INEv	INEv	INEv	INEv	INEv	INEv	INEv	INEv	ENEv	DSEv	TREv	ENEv	DSEv	ENEv	DSEv	TREv	INEv
Sequence number	C.A	C.B	B.A	B.B	A.A	A.B	A.C	A.D	A.A.1	A.B.1	A.B.2	A.B.3 A.C.4	A.D.1 C.A.1	A.C.1	A.C.4	B.A.1	A.C.3
Source	1.1.3	1.1.3	1.1.2	1.1.2	1.1.4	1.1.4	1.1.4	1.1.4	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1
Destination	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.6	1.1.6	1.1.2	1.1.4	1.1.4	1.1.5	1.1.5	1.1.3	1.1.6

[표 4.1] Speed Control의 Event flow STS

Data flow STS														
Data flow ID	Df1	Df2	Df3	Df4	Df5	Df6	Df7	Df8	Df9	Df10	Df11	EI	BI	CCR
Data flow description														
Data flow Type	TRDa	DSDa	TRDa	TRDa	DSDa	TRDa	TRDa	TRDa	ODDa	ODDa	EXDa	IDDa	IDDa	IDDa
Sequence number	A.B.2.1	A.B.2.2	A.C.5.1	A.C.4.1 A.B.2.1	B.A.1.1	A.C.1.1	A.C.1.2	A.B.2.3 A.C.4.3	A.C.1.3	A.A.1.1	A.A.1.1 A.C.1.3.1 A.B.2.3.1 A.C.4.3.1	B	C	A
Source	DS1	1.1.1.2	DS1	DS1	1.1.1.3	DS2	DS2	1.1.1.4	1.1.1.5	1.1.1.6	1.1.5			
Destination	1.1.1.2	DS2	1.1.1.4	1.1.1.5	DS2	1.1.1.4	1.1.1.5	1.1.5	1.1.5	1.1.5		1.1.2	1.1.3	1.1.4

[표 4.2] Speed Control의 Data flow STS

Transformation STS										
Transformation ID	1.1.2	1.1.3	1.1.4	1.1.5	1.1.1.1	1.1.1.2	1.1.1.3	1.1.1.4	1.1.1.5	1.1.1.6
Transformation description	Engine	Brake	Cruise Control Lever	Throttle	Cruise Control	Select Desired Speed	Clear Desired Speed	Maintain Speed	Resume Speed	Increase Speed
Transformation Type	IDTr	IDTr	IDTr	ODTr	COTr	SDTr	SDTr	SDTr	SDTr	SDTr
Activation type	P	P	A	P	A	A	A	A	A	A
Frequency	100	100		100						
Execution time						ST	ST	S	S	S
Priority	TC	TC	TC	TC	TC	NTC	NTC	NTC	NTC	NTC
Activating input	DF12	DF13	DF14	DF8 DF9 DF10	Accel Cruise Resume Off	T3	T8	E4	E6	E1
Input event flows					Accel Cruise Resume Off EO, EF BO, BF RC	T3	T8	E4 D5	E6 D7	E1 D2
Input data flows	DF12	DF13	DF14	DF8 DF9 DF10		DF1		DF3 DF6	DF4 DF7	
Output event flows	EO EF	BO BF	Accel Cruise Resume Off		E1, D2 T3, E4 D5, E6 D7, T8				RC	
Output data flows				DF11		DF2	DF5	DF8	DF9	DF10

[표 4.3] Speed Control의 Transformation STS

[표 4.1]과 [표 4.2]는 자동 속도 조절 시스템의 Speed Control 부 시스템의 event/data flow의 STS를 나타내고 [표 4.3]은 transformation의 STS를 나타낸 것이다. [표 4.4]는 Speed Control 부시스템의 각 transformaion에 적용한 태스크 구성 알고리즘을 요약한 것이다.

알고리즘 \ Tr	1.1.2	1.1.3	1.1.4	1.1.5	1.1.1.1	1.1.1.2	1.1.1.3	1.1.1.4	1.1.1.5	1.1.1.6
1.1 단계			○							
1.2.1 단계	○	○								
1.2.2 단계				○						
2 단계					○					
2.1.1 단계						○	○			
2.1.2 단계										
2.2 단계										
2.3 단계								○	○	○
3.1 단계										
3.1.1 단계										
3.2 단계										
4.1 단계										
4.2 단계										

○ 한 태스크로 구성함

[표 4.4] Speed Control의 알고리즘 적용표

5. 결론 및 향후 연구

본 연구에서 COBRA의 분석 결과와 여러 다이아그램에서 나타내지 못했던 태스크 구성에 필요한 정보들을 통합적으로 나타낼 수 있는 STS를 제안하고 STS를 이용하여 자동으로 태스크를 구성할 수 있는 알고리즘과 구현 예제를 보였다. STS는 그것을 이용한 태스크 구성이 용이하고 설계자가 문제영역에서 명확한 행동 모델을 개발할 수 있도록 도와주는 지침 역할한다. 또, 태스크 구성 전에 행동 모델의 분석단계에서 행동 모델이 태스크 구성에 적합한지를 검증할 수 있고, 가장 하위 data/control transformation이 순차적인 객체나 함수로 명확하게 분해되었는지를 알 수 있다. 그러나, STS를 이용한 태스크 구성은 Gomaa의 태스크 구성 지침을 여전히 따르므로 태스크를 구성하는데 한계가 있다.

추후 연구 계획은 CODARTS 방법론의 COBRA 분석 단계에서 태스크 구성 단계까지 지원하는 자동 태스크 구성 도구를 설계, 구현한다. 이러한 자동 태스크 구성 도구는 RTSA notation에서 본 연구를 바탕으로 자동으로 태스크를 구성한 후 태스크 구성 결과를 TAD(Task Architecture Diagram)로 나타내도록 하여 분석 단계에서 태스크 구성까지 자동화할 수 있도록 한다.

참 고 문 헌

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

[2] Roger S. P, "Software Engineering A Practitioner's Approach", third edition, McGRAW-HILL, 1992

[3] Douglas Niehaus, "A Real-Time System Description Language", Real-time technology and applications symposium, IEEE computer society Press, 1995

[4] 이종구, 김규년, "확장 방향성 그래프를 이용한 실시간 병렬 태스크의 단계적인 구성", 울산대학교 공학 연구논문집, vol. 27, no. 2, 1996

[5] Maher A, Juha K, Jurgen Z, "Object-Oriented Technology for Real-Time Systems", Prentice-Hall, 1996