실시간 시스템의 순환 공학을 위한 정형 기법 : 추상 시간 기계

노경주° 박지연 이문근
전북대학교 컴퓨터 과학과
{kjuno, jypark, mklee}@cs.chonbuk.ac.kr

A Formal Method for Round-Trip Engineering of Real-Time System : Abstract Timed Machine

Kyoung-Ju Noh° Ji-Yeon Park Moon-Kun Lee
Dep. of Computer-Science, Chonbuk National University

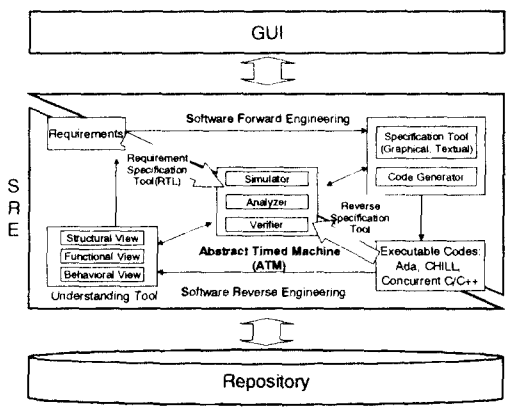
요약

본 논문은 ATM(*Abstract Timed Machine*)에 대해서 기술한다. ATM 은 임무 위급 시스템과 같은 실시간 시스템을 명세, 분석 및 검증할 수 있는 정형 기법이다. ATM 은 모드(*mode*), 전이(*transition*), 포트(*port*)로 구성된다. 모드는 머신의 압축된 상태를 의미하고 전이는 모드와 모드 사이의 변화를 말한다. 포트는 ATM 들 사이의 상호작용을 위한 진입을 나타내기 위한 것이다. ATM 은 소프트웨어 순환 공학을 위해 디자인 되었다. ATM 은 재/역공학적 측면에서 계산 이론과 더불어 기존의 실시간 시스템의 소스코드에 대한 디자인 및 환경 정보를 나타낸다. 본 논문은 병렬성, 병렬적으로 동작하는 엔터티들 사이의 제어 정보의 흐름, ATM 타입과 클래스로부터의 인스턴스, 비/동기적 이벤트, 포트와 이벤트 타입, 포트의 타입, 통신, 입/출력, 예외처리, 시간에 관한 요구사항, 다수를 대상으로 하는 통신, 주기적 작업등과 같은 ATM 의 여러 개념을 기술하고 이러한 속성들을 Producer-Buffer-Consumer 예제로 살펴본다.

1. 서론

실시간 소프트웨어는 크기 면에서 대우 방대하고, 병렬성, 동기/비동기, 통신, IO, 시간과 같은 복잡한 속성 외에도 분산성, 형상성 등과 같은 환경적 성질들을 가지고 있다. 이러한 크기와 복잡성 때문에 실시간 시스템을 명세 하는 기존의 정형기법들[1,2,3,5,7]을 통하여 역공학 과정에서 소프트웨어의 동적 행위를 이해하는 것은 매우 어렵다.

위와 같은 기존 정형명세 기법의 단점을 보완하여 임무 위급 시스템과 같은 실시간 시스템의 여러 속성이나 동적 행위에 관한 정보를 명확하게 표현하고, 명세된 시스템에 대한 신뢰할 수 있는 검증 방법을 제공하기 위해 ATM을 개발하게 되었다.



<그림 1> 상태도에 기반을 둔 SEE/SRE 통합 순환 공학 환경

<그림 1>에서 보여주는 바와 같이 ATM 은 SEE(Software Engineering Environment)/SRE (Software Reverse Engineering Environment), 즉 순공학과 역공학이 하나로 통합된 순환공학(Round-Trip Engineering)환경을 구축하기 위한 정형기법이다.

본 논문의 1 장에서는 ATM 개발 동기, 2 장에서는 관련 연구, 3 장에서는 ATM 기본 구성 및 속성, 4 장에서는 예제, 5 장에서는 ATM 구조에서 제공되는 뷰(view), 6 장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

정형 기법의 대표적인 기법인 LTS(*Labeled Transition System*)인 Statechart[3]와 Modechart[7]는 모듈 구조를 통해 시스템 상태간의 전이를 표현하는 상태 기반 기법이다. 캡슐화를 통해 계층성을 표현하고 각 상태나 모드 사이의 병렬성을 나타낸다. 이와 같은 계층성은 한 시점에서의 시스템의 다중 계층 관계나 상태와 외부 상태와의 상호작용을 명시적으로 표현하기 어렵고, 또 이들에서 사용되는 광역 통신(broad casting) 방법은 각 기계사이 또는 상태, 모드 사이의 동기화가 중요시 되는 실시간 시스템을 정확히 명세하기 어렵다.

CRSM(*Communicating Real-Time State Machines*)[1]은 실시간 시스템의 요구사항을 명세하기 위한 목적으로 개발된 정형 기법으로 병렬성, 통신, 동기화, 시간에 대한 표현을 제공하지만 마찬가지로 시스템의 계층성을 명시적으로 표현하지 못한다.

Statechart와 Modechart, CRSM 등과 같은 상태기반 정형기법의 첫번째는 문제점은 상태 폭발(state explosion) 가능성이고, 또 다른 문제로는 이러한 정형 기법들은 역공학 과정에서 요구되는 원시 소프트웨어의 구조 정보를 충분히 제공하기 어렵다는 것이다.

3. ATM

ATM은 소프트웨어 역공학 과정에서 원시 소프트웨어에 대한 명세를 위해 개발된 LTS이다.

이장에서는 ATM의 구성요소 및 ATM이 정의하고 있는 여러 속성을 기술한다.

본 연구는 한국과학재단 특정기초연구 (과제번호 1999-2-203-003-3) 지원으로 수행되었음.

3.1 ATM 구성요소

ATM은 모드(mode)의 집합, 가드(guard)된 전이(transition)의 집합, 포트(port), 실행 시작점과 실행의 종료점으로 구성된다.

머신은 ATM의 기본 단위로 내부에 모드를 포함한다. 일반적으로 태스크, 프로시저와 같은 독립적 프로그램 블록 단위를 표현 한다.

모드는 시스템의 상태를 나타내기 위한 것으로 모드의 역할에 따라 각 모드는 타입화 되어 이름, 아이디, 타입, 시간 제약 등 자신을 대표할 수 있는 속성을 갖는다. 모드의 종류로는 계산 모드, 추상화 모드, 주제 모드가 있다. 계산 모드는 실행 문을 내부에 직접적으로 포함하고 있어 자체의 실행에 의해 다음전이를 유발한다. 추상화 모드는 소프트웨어 구조 계층의 하위 단계의 머신을 현재단계에서 추상화 시킨다. 주제 모드는 소프트웨어의 특정 기능을 수행하거나 특정 기능에 영향을 받는 모드들을 표현하는 모드이다.

전이는 이벤트, 조건, 시간제약으로 구성된 레이블을 갖고, 포트는 머신 간의 통신을 위한 것으로 머신의 활성/비활성을 위한 활성화(activation) 포트, 엔트리(entry) 포트, 대체(substitute) 포트가 있다.

ATM은 1) 해석 (Interpretation), 2) 재구성 (Restructuring), 3) 명세언어로의 표현의 단계를 거친다. 해석 단계에서는 소스코드로의 블록 단위에 따른 머신과 모드에 대한 타입 ATM이 생성된다. 재구성 단계에서는 시스템의 동적 행위에 따라 타입 ATM의 재구성이 이루어지고, 마지막 단계에서 ATM으로 표현된다.

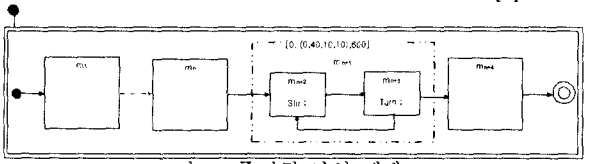
3.2 ATM의 속성

ATM에서의 인스턴스된 모든 머신은 병렬 수행된다. 병렬 실행되는 머신은 머신 내부에서 추상화 시키고 있는 아키텍쳐 계층의 하위 단계의 머신의 병렬 실행도 잠재적으로 포함한다. 이러한 추상화 모드는 소스코드의 블록 구조를 아키텍쳐 계층으로 표현 가능하게 한다.

시간에 대한 요구 사항을 모드나 전이상에 레이블 형태로 명시적으로 표현한다. 특히 일정한 간격을 두고 준비되고 실행되는 주기적 작업의 대한 준비 시간, 주기, 실행 시간, 데드라인등의 시간 제약이 주제모드에 레이블로 명시적으로 의해 표현된다. 예제는 <그림2, 3>에서 보인다.

```
from now to now+10 min every 40 sec
execute 10 sec within 10 sec do
    Sir;
    Turn;
end;
```

<그림 2> 주기적 작업을 나타내는 CRSM[8]

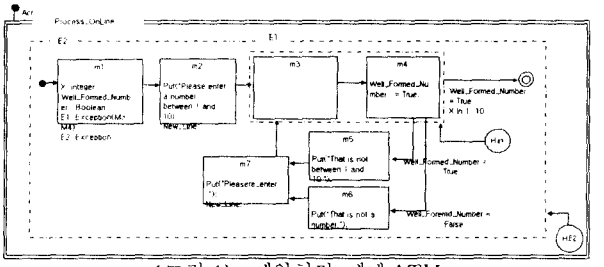


<그림 3> 주기적 작업 예제 ATM

ATM 에서는 통신을 위해 송신 기호로 '!'을, 수신 기호의 '?'로 표현하고 여기에 확장하여 RPC 와 같이 동기적 통신의 표현을 위해 '!?'을 사용한다. '!?'의 의미는 어떠한 시그널이나 이벤트를 원격 상대 머신에 보내고, 이에 따른 대답이 수신될 때까지 기다림을 의미한다

유한한 자원에 대한 자원경쟁을 해결 하게 위한 정책으로 작업 대기 큐를 사용한다. 큐는 FIFO(*First In First Out*)방식을 채택하며, 대기하는 머신의 순서는 머신이 가진 우선순위에 의해 결정된다.

ATM 의 예외처리는 예외처리 핸들러가 머신 내에서 모드 단위로 적용되는 내부예외(Internal Exception), 머신 간의 동작이나 머신 내부의 오류등에서 발생하는 머신 외부의 외부예외(External Exception)가 있다. 각 머신의 선언부에서 각 예외 처리 핸들러가 처리할 수 있는 모드들이 표현되고 각 예외처리 핸들러의 범위는 주제 모드에 의해 표현된다. 해당 예외처리 핸들러 머신이 종료 한 후 시스템 제어는 예외가 발생한 머신에 속한 히스토리(history)를 사용하여 복귀한다. 히스토리는 머신의 흐름 정보를 저장하고 있어서 예외 발생시의 상태를 기록하고 있으므로 히스토리의 마지막 상태로 복귀하면 된다.



<그림 4> 예외처리 예제 ATM

<그림 4>에서는 *m1*모드에 모드 *m3*와 *m4*의 범위를 가진 지역 예외처리 머신 *E1* 과, 머신 전체에 걸쳐 적용되는 전역 예외처리 *E2* 가 선언되었다. 예외처리 주제 모드는 점선으로 예외처리의 범위를 나타낸다. 예외가 발생할 경우 해당 예외처리 머신이 인스턴스 되어 동작한 후 해당 주제모드에 속해 있는 히스토리를 이용하여 제어가 복귀한다.

4. ATM 예제

Ada 코드인 Producer-Buffer-Consumer (PBC) 예제를 ATM으로 표현한 것이다. 먼저 해석 단계와 재구성 단계를 거쳐 타입 ATM이 생성된 후 인스턴스 된 ATM이 표현된다.

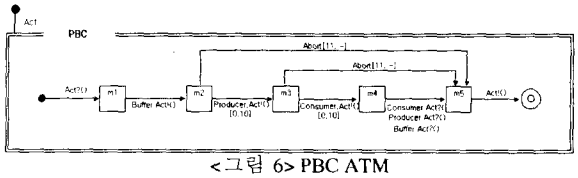
PBC는 protected 타입의 Buffer와 Producer, Consumer 프로시져로 구성되어 있다. Producer는 인스턴스 된 후, 사용자로부터 값을 입력 받아 5단위 시간 내에 버퍼에 기록한다. Consumer는 버퍼로부터 5단위 시간 내에 메시지 읽어 들여 출력한다.

```
Procedure PBC;
procedure PBC is
 protected Buffer is
  entry Read (C : out Character);
  entry Write(C : in Character);
 private
  Pool   : String(1..100);
  Count  : Natual := 0;
  In_Index : Positive := 1;
  Out_Index : Positive := 1;
 end Buffer;

 protected body Buffer is
  entry Write(C : in Character)
   when Count < Pool'Length is
  begin
   Pool(In_Index) := C;
   In_Index:=( In_Index mod Pool'Length) + 1;
   Count := Count + 1;
  end Write;

  entry Read(C : out Character)
   when Count > 0 is
  begin
   C := Pool(Out_Index);
   Out_Index:=(Out_Index mod Pool'Length) + 1;
   Count := Count - 1;
  end Read;
 end Buffer;

 task type Producer_Type;
 task body Producer_Type is
  Msg : character;
 Begin
  Loop
   get(Msg);
   select
    Buffer.Write(Msg);
    Exit when Msg = ASCII.EOT;
   or
    delay 5.00;
```

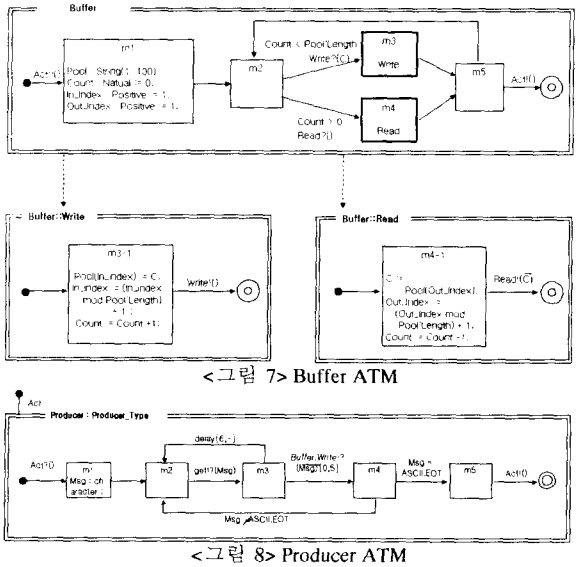
```
    end select;
   end while;
 end Producer_Type;

 task type Consumer_Type;
 task body Consumer_Type is
  Msg : Character;
 Begin
   Loop
    Select
     Buffer.Read(Msg);
     Put(Msg);
     Exit when Msg = ASCII.EOT;
    or
     delay 5.00;
    end select;
   end while;
 end Consumer_Type;

 type Producer_Ptr is access Producer_Type;
 type Consumer_Ptr is access Consumer_Type;

 Producer : Producer_Ptr;
 Consumer : Consumer_Ptr;

Begin
 Select
  delay 10.00;
 then abort
  Producer := new Producer_Type;
 end select;

 select
  delay 10.00;
 then abort
  Consumer := new Consumer_Type;
 End select;
end PBC;
```

<그림 5> Producer-Buffer-Consumer(PBC) 예제

다음은 인스턴스 된 PBC와, Buffer, Producer, Consumer를 보인다. 원시 코드의 각 타입, 프로시저, 함수 등의 내부의 제어의 흐름, 계층 관계가 표현된다. <그림 7>의 Buffer를 보면 추상화 모드에 의해서 블록 구조에 따른 아키텍쳐를 표현하고 있다.

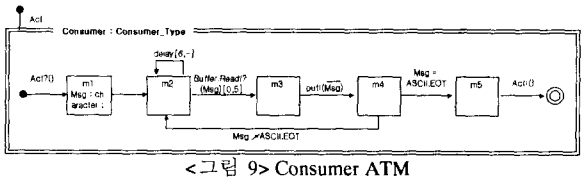


<그림 6> PBC ATM



<그림 7> Buffer ATM

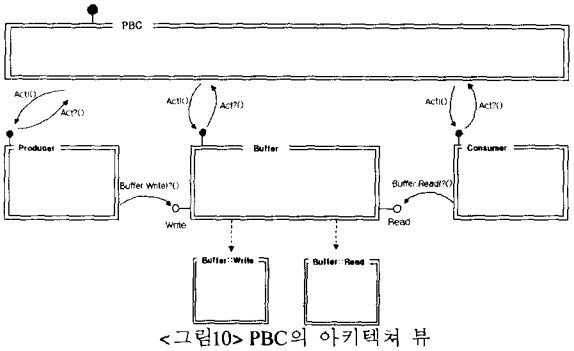
<그림 8> Producer ATM



<그림 9> Consumer ATM

5. 뷰(View)

전역 상호 작용 뷰는 머신 간의 통신에 의한 상호 작용을 거시적 관점에서 보여준다. 아키텍쳐 뷰는 전역 상호 작용 뷰에 추가하여 추상화 모드에 의한 추상화 된 머신 까지의 상호작용을 보여준다.



<그림10> PBC의 아키텍쳐 뷰

6. 결론 및 향후연구

현재 ATM은 소프트웨어가 가질 수 있는 다양한 속성을 정의하는 단계에 있으며 분산성이나 다수에 대한 동기화와 같이 아직 정의되지 않은 속성이 존재한다. 또 설계 단계에서 밝혀지지 않은 새로운 정의를 필요로 하는 여러 속성에 대한 연구도 지속해야 한다.

향후 연구는 정의 되지 않는 속성을 정의하고 ATM으로 명세된 시스템에 대한 검증에 관한 연구를 지속하는 것이다. 또한 ATM이 정의된 속성을 표현하고 검증 할 수 있는 자동화 도구로 개발하는 것이다.

참고문헌

[1] A. Shaw, "*Communicating Real-Time State Machines,*" IEE Transactions on Software Engineering, Vol. 18, No. 9, pp. 805816 September 1992.

[2] C.A.R.Hoare, "*Communicating Sequential Processes,* Prentice-Hall MD.

[3] D. Harel, "Statecharts: A Visual Formalism for Compl System," Science of *Computer Programming.* Vol. 8. 1987, p 231-274.

[4] Feldman and Koffman,"Ada95," Addison-Wesley, 1996.

[5] I. Kang and I. Lee, "State Minimization for Concurrent Syste Analysis Based on State Space Exploration," *Proceedings Conference on Computer Assurance,* Gaithersburg MD, June 199

[6] Moon Lee, "*An Environment for Understanding of Real-tin Systems,*" Ph.D. Thesis The University of Pennsylvania, 1995.

[7] S. jahanian and A. Mok,"Modechart : *A Specification Language f Real Time Systems,*" IBM Technical Report: RC 15140, Novemt 1989.

[8] Insup Lee, "Language constructsfor Distributed Real-Tin Programming," Dec, 1985