자연어 요구사항의 상태차트 모델링*

김진현° 김창진 심재환 박승현 최진영
고려대학교
{jhkim°, cjkim, jhsim, shpark, choi}@formal.korea.ac.kr

Modeling Requirements in Natural Language with Statecharts

Jin-Hyun Kim , Chang-Jin Kim, Jae-Hwan Sim, Seung-Hyun Park, Jin-Young Choi
Dept. of Computer Science, Korea University

요 약

정형명세는 자연어의 모호함을 없는 명료한 시스템 설계를 가능하게 한다. 상태차트와 같은 정형명세 된 요구사항은 시뮬레이션이나 정형검증을 통해 요구사항을 실행하여 볼 수 있으며, 더 나아가 여러 가지 특성을 정형검증과 같은 검증 기법으로 검증 할 수 있다. 하지만 자연어 요구사항을 상태차트로 변환하여 다양한 요구사항의 특성을 기술하기 위해서는 상당한 노력과 경험이 필요로 하다. 본 논문에서는 자연어 요구사항을 상태차트로 직접 변환하는 기법을 제안한다. 이를 위해 본 논문에서는 기능적인 요구 사항의 자연어를 분석하고, 또한 소프트웨어 요구사항 기술에 적절하도록 상태차트 문법의 의미를 제안한다.

1. 서론

소프트웨어의 위기의 시대 도래하면서, 이러한 소프트웨어의 요구와 설계에 대한 기술과 기술된 명세의 검증을 통한 시스템의 완전하고 정확한 이해가 강조되었다. 이러한 완전하고 정확한 요구 및 설계 명세를 위해, 논리나 수학에 기반한 명세 언어 및 검증 기법이 개발되었는데, 이러한 언어가 바로 정형명세 언어이다. 하지만 정형명세[1] 언어는 수학이나 논리에 기반한 언어이기 때문에, 수학적 기본 지식에 대한 습득이나 훈련이 없이는 적용하기가 쉽지 않다.

상태차트[2,3]는 정형명세 언어의 하나로 시스템의 상태를 기반으로 시스템의 행위를 모델링 한다. 이러한 상태 기반의 모델은 시간에 따른 상태 변화에 따라 행위를 기술한다. 이러한 상태차트는 상태 다이어그램에 기반한 언어이다. 또한 대부분 상태 기반의 정형명세 언어는 어렵지 않게 오토마타로 변형되고, 이러한 오토마타는 또한 상태차트로 변형될 수 있다. 따라서 상태차트는 시스템을 행위 관점에서 기술하게 하는 고급 언어로 진화하고 있다.

하지만 이러한 상태차트는 자연어로부터 직접적으로 변형되기는 쉽지 않다. 특히 상태차트는 상태와 전이로 이루어져 있고, 조건과 이벤트가 이러한 상태간의 전이를 일으킨다. 따라서 자연어 내의 시스템의 상태와 전이 조건 등을 구분하기 쉽지 않다. 특히 상태 차트는 계층성과 직교성의 문법과 의미론을 지니고 있는데, 자연어 요구사항에서 이를 충분히 활용할 수 있는 방법이 없어 경험에 따라서 자연어를 상태차트로 변형할 수 밖에 없다.

본 논문에서는 자연어 기술된 소프트웨어 요구사항을 상태차트로 변형하는 규칙과 방법을 제시한다.

본 논문의 구성은 다음과 같다. 다음 절에서는 본 연구와 관련된 연구를 제시한다. 3절에서는 소프트웨어 요구사항의 특성과 요구사항을 제시한다. 또한 자연어로 기술된 기능적인 특성의 자연어 문장 특성을 구분하고, 상태차트의 문법 특성을 기술한다. 다음 4절에서는 자연어에서 어떻게 상태차트로 변형되는지 기술한다. 그리고 5절은 본 논문의 결론을 기술한다.

2. 관련연구

자연어 요구사항을 상태차트와 같은 행위언어로 바꾸는 대표적인 방법은 UML를 이용하는 것이다[2]. UML은 Use-case를 통해서 사용자의 요구사항을 분석한다. Use-case는 사용자가 요구한 기능이 어떠한 것이 있으며, 이러한 기능들 간의 포함관계 등을 기술하게 된다. 이러한 Use-case로 분석된 자연어는 다양한 클래스로 구별된다. 이러한 클래스는 시퀀스 다이어그램을 통해 필요한 오퍼레이션이 구분된다. 또한 이러한 오퍼레이션을 통해 클래스가 가진 데이터에 변형을 가하게 되고, 이러한 오퍼레이션은 클래스 다이어그램의 메소드로 변형하게 된다. 또한 클래스가 가진 데이터는 클래스 다이어그램의 데이터로 기술된다. 이 클래스 다이어그램은 다시 행위적인 명세인 상태차트를 통해, 클래스의 행위를 기술하게 된다. 이러한 상태차트의 행위 기술을 통해, 클래스가 객체로 객체화되고, 각각의 메소드가 언제 어떻게 호출되는지 그 조건을 기술하게 된다.

이러한 변형 이외에도, 자연어의 명세를 Z로 변형한다[6]. Z는 이산수학의 집합론과 술어 논리에 기반한 명세 언어로서, 변수의 타입과 이러한 변수가 특정 기능에서 어떻게 변하는지를 기술하면서, 상태를 기술한다. 이러한 기술을 통해 변수 집합의 상태를 구분하고 이를 기반으로 상태와 상태전이를 구분함으로 상태차

* 본 논문은 KNICS(Korea Nuclear Instrumentation & Control System)의 지원을 받아 수행하였음.

트를 기술하게 된다.
하지만 두 기법 모두 자연어를 직접적인 변형이 쉽지 않다. 본 논문에서는 자연어 명세의 문장의 특성을 구별하고, 이를 직접 상태차트로 기술한다. 다음 절에서는 소프트웨어의 요구사항의 특성을 기술한다.

3. 소프트웨어의 요구사항 특성

소프트웨어의 요구명세는 사용자의 소프트웨어에 대한 요구를 기술한다. 이러한 요구명세는 시스템이 만족해야 할 요구 조건을 기술하는데, 기능적인 요구조건과 비기능적인 요구조건을 명세한다. 이러한 기능적인 요구조건은 개발한 대상을 구분하고 그 대상이 무엇을 수행할 것인지를 기술한다. 비기능적인 요구조건은 시스템의 사용가능성(Usability), 효율성(Efficiency), 수행속도(Performance)와 자원의 사용, 신뢰성 등을 기술한다.
소프트웨어 요구 사항에 기술되어야 할 특성은 다음과 같다[7].

- **소프트웨어에 요구되는 기능에 대해 명세해야 한다.**
- **소프트웨어의 인터페이스에 대해 명세해야 한다.**
- **소프트웨어의 기능 수행 속도, 반응 시간 등 요구되는 performance에 대해 기술되어야 한다.**
- **정확성, 유지보수, 보안성 등 특성에 대해 기술되어야 한다.**
- **소프트웨어로 구현할 때 필요한 설계 제약 조건에 대해 기술되어야 한다. 즉, 구현하고자 하는 언어, 자원 제약, 동작 환경 등에 대해 기술해야 한다.**

그리고 이렇게 기술된 요구 사항은 다음과 같은 특성을 지니고 있어야 한다. 이러한 요구 사항은 다음과 같은 특성이 만족하는지 검증해야 한다.

- **소프트웨어 요건 명세는 상치되는 면 없이 정확해야 한다.**
- **소프트웨어 요건 명세는 모호하지 않아야 한다.**
- **소프트웨어 요건 명세는 완전해야 한다. 이를 위해서 중요한 모든 요구 조건들이 기술되어야 하고, 유효한 입력뿐 아니라 유효하지 않은 소프트웨어의 응답에 대해서도 기술되어야 한다.**
- **소프트웨어 요구 명세는 일관성을 갖추어야 한다.**
- **소프트웨어 요구 명세는 검증 가능해야 한다. 즉, 구현된 소프트웨어가 요구조건을 만족하는 지 확인할 수 있어야 하는데 이를 위해서는 모호한 요구조건이 없어야 한다.**
- **소프트웨어 요구 명세는 쉽게, 완전하게, 일관성 있게 수정 가능해야 한다.**
- **소프트웨어 요구 명세는 추적성이 있어야 한다.**

이러한 요구사항을 검증하기 위해 다양한 검증 기법이 적용된다. 특히 정형 검증 도구로는 완전성이나 정확성, 무모순성 등을 검증한다. 특히 I-Logix의 STATEMATE MAGNUM과 같은 도구는 직접 실행을 통해 요구 사항에 대한 이해를 즉시 해 볼 수 있다는 장점을 가지고 있다.

본 논문에서 상태차트로 변형되는 요구 사항은 기능에 대한 명세, 인터페이스 명세, 수행 및 반응 시간을 명세 한다. 자연어로 기술된 기능적인 요구명세는 다음을 명세 한다.

- **기능설명**
- **기능의 입력조건 및 출력조건**
- **수행 제어 및 수행 조건**
- **안전성(safety) 조건**

기능, 입,출력 조건, 수행 및 수행 조건은 시스템의 기능에 대한 기술이다. 기능설명은 시스템의 기능에 대한 개략적인 설명이다. 기능설명은 시간과는 관련 없이, 특정 입력에 대해 원하는 출력을 얻을 수 있는지에 대한 요구 사항이다. 수행 순서는 기능의 목적을 만족시키기 위해, 변수의 대입, 순환 등의 일련의 제어 순서를 가리킨다. 이러한 순서는 수행 조건에 의해 다이나믹 하게 바뀌고, 기능의 설명은 이러한 수행 조건에 의한 수행 순서의 제어를 통해 알고리즘이 기술되고, 이렇게 행위가 기술된다. 이러한 기능적 요구 사항 이외에, 비기능적인 요구 사항으로서 안전성(Safety)과 수행성(Performance) 요구 사항이 있다. 안전성 요구 사항은 특정 기능이 만족해야 하는 조건으로 기능이 이 조건을 만족하지 않을 경우 문제를 발생시킬 수 있음을 명시하는 것이다. 일례로 " 어떠한 일은 절대로 일어나지 말아야 한다." 라는 문장의 형태가 안전성 조건을 기술한다.
이러한 기능 요구 사항을 기술하기 위한 자연어의 형태는 다음과 같이 구분된다.

- **행위 요구 사항**: " ~을 해야 한다." " ~을 수행한다."
- **상태 요구 사항**: " ~한 상태가 되어야 한다."
- **조건 사항**: " ~할 때," " ~라면"
- **입, 출력 사항**: " ~가"

행위 요구 사항은 기능이 해야 하는 행위-알고리즘, 수행 절차 등-를 기술한다. 이에 반해 상태 요구 사항은 기능의 특정 행위의 결과를 표현한다. 특정한 출력 변수의 상태와 관련이 있다. 또한 시스템의 요구된 상태 혹은 의도되지 않은 상태를 기술할 때에도 상태 요구 사항 형태로 문장을 기술한다. 조건 명세는 행위를 일으키거나 특정 상태를 유도하는 조건을 가리킨다. 입력 변수의 값과 밀접한 관련을 가지고 있다. 다음 절에서는 이러한 자연어를 상태차트와 관련시키기 위한 상태차트의 문법을 기술한다.

3.1. 상태차트

상태차트는 상태 다이어그램의 확장된 언어로서 시스템의 계층성 및 동시성을 기술한다. 근래 들어 일반 프로그램 언어 수준의 언어적인 특성이 추가되어 모델링뿐 아니라 실제 구현이 가능한

정도까지 언어가 발전하고 있다. 그림 1에서는 I-Logix[5]의 STATEMATE MAGNUM의 상태차트 문법을 보여주고 있다.

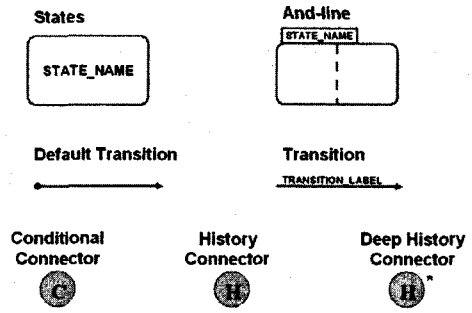


그림 1 상태차트의 도식적 문법

특히 TRANSITION_LABEL은 다음과 같은 형태의 조건을 갖는다.

E[c]/A

" E" 는 발생하는 이벤트를 가키킨다. " c" 는 이벤트가 전이를 발생시키기 위해 만족해야 할 조건문으로 []안의 조건이 만족해야만 전이를 일으킬 수 있다. A는 전이가 일어날 때 수행하는 액션을 가리킨다.

4. 자연어 요구 사항의 상태차트의 명세

4.1. 자연어를 위한 상태차트의 해석

소프트웨어 요구 사항의 자연어 명세를 해석하기 위해 상태차트의 각각의 문법에 소프트웨어의 행위적인 의미를 부여하면 다음과 같다.

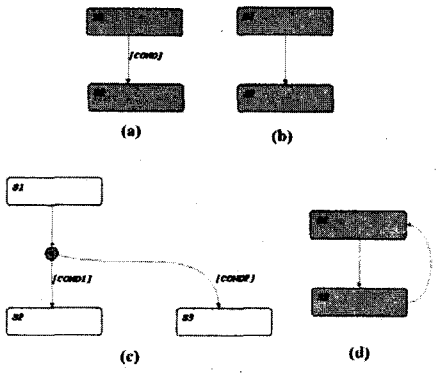


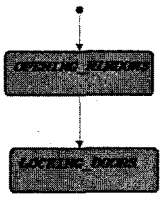
그림 2 상태 전이의 종류

자연어 요구 사항을 명세하기 위해 본 논문은 상태차트의 가장 기본적인 문법인 상태와 전이, 전이 조건을 사용한다. 상태 차트의 상태는 시스템의 특정 시간 내에서의 모드이다. 이 상태는 연결된 진입 전이와 진출 전이 지니고 있다. 소프트웨어의 행위를 해석할 수 있는 상태 차트의 상태 전이는 다음 다섯 가지 형태로 나눌 수 있다.

그림 2(a)는 특정 조건에 의존하는 순차적 행위를 의미한다. 이 상태차트에서 상태 S1은 두 가지 행위적인 의미를 부여할 수 있는데, 만약 [COND]를 만족시키는 COND가 외부 조건이라면 상태 S1은 시간을 소모하는 상태가 될 수 있다. 하지만 COND가 기능의 내부 조건이라면 S1은 시간을 소모하는 상태가 아니다. 그림 2(b)는 특정 행위의 순차적인 수행을 기술한다. 이때 상태 S1은 조건 없이 S2를 진행하기 때문에, S1은 기능의 수행을 추상화 한 것으로 의미를 부여할 수 있다. 즉 S1은 특정한 행위를 수행한 후, 바로 S2로 진행하는 것을 의미한다. 그림 2(c)는 조건 COND1과 COND2에 따라 수행 흐름이 달라지는 것을 의미한다. 그림 2(d)는 수행 순환을 의미한다. 그림 2의 상태의 의미에서 볼 수 있는 것처럼 상태차트의 상태는 전이하기 위해 특정 조건이 만족하기를 기다리는 상태와 특정 행위만을 표현하기 위한 상태로 나눌 수 있다. 이러한 상태를 본 논문은 다음과 같이 구분한다.

Def. 1 Hot state
상태의 집합 중, 진출 전이 조건이 항상 참인 상태.

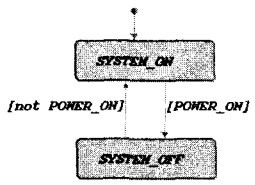
Hot 상태는 진출 전이가 즉시 일어나야 하는 상태이다. 이 상태는 소프트웨어 요구 사항의 기능의 행위를 기술할 때 사용한다. 상태차트의 표현은 상태에 행위를 대표하는 이름이 붙여진 형태로 기술된다. 예를 들면 다음과 같다.



Hot 상태의 이름은 " 행위+대상" 의 형태를 지니며, 이렇게 특정 대상에 대한 행위가 일어나고 있음을 기술한다. Hot 상태의 진출 전이 레이블은 항상 " 참" 이어야 한다. 만약 조건이 진출 전이 레이블이라면 조건이 참인 경우와 참이 아닌 경우를 모두 명시함으로 종료 이전까지의 기능 수행의 순서를 완전하게 기술해야 한다. 앞에서 언급한 **행위 요구 사항**을 기술할 때 사용된다.

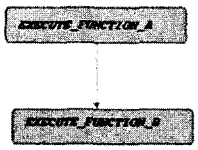
Def. 2 Code state
상태의 집합 중, 진출 전이 조건이 거짓인 경우도 있는 상태.

상태의 집합 중, 진출 전이 조건이 만족해야만 진출을 시도하는 전이를 가진 상태로서, 이 상태에서는 시스템의 다른 부분이 기능을 수행하여 진출 전이 조건을 만족시킬 때까지 전이의 수행이 정지해 있을 수 있다. 이 상태는 **상태 요구 사항**을 기술할 때 사용된다. 즉 특정 행위를 한 후의 시스템의 상태를 기술한다. 예를 들면 다음과 같다.

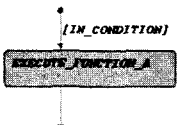


Cold 상태의 이름은 " 대상+상태" 형태를 지니고 있으며, 각각의 상태에서 상당한 시간을 보낼 수 있다. 앞서 언급한 요구 사항의 각각의 소프트웨어 요구 사항을 기술하도록 상태차트는 다음과 같은 형태를 지닌다.

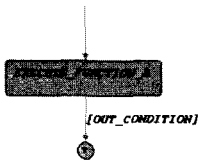
1) 기능 요구: 각각의 행위와 행위의 순서를 기술한다.



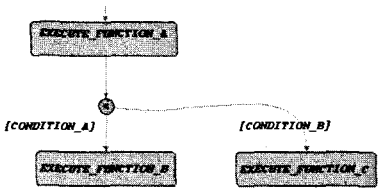
2) 기능의 입력 요구: 기능이 시작하게 될 조건을 초기 상태의 default 전이에 기술한다.



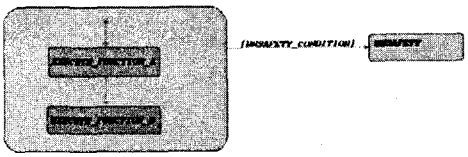
3) 기능의 출력 요구: 기능의 종료 조건을 기능의 종료 노드의 진입 레이블에 기술한다.



4) 수행 제어 및 수행 조건: 전이의 레이블에 수행 조건을 기술하며, 조건에 따른 제어를 기술한다.



5) 안전성 요구: 기능의 수행 도중, 기능이 항상 만족해야 하는 조건을 기술하는데, 상태차트에서는 이 요구를 만족시키지 못하는 경우의 조건을 다음 그림과 같이 기술함으로 안전성 요구사항을 기술한다.



안전성 조건은 시스템이 에러를 발생시킬 상황을 기술한다. 안전성 조건의 기술은 시스템이 수행 도 중, UNSAFETY_CONDITION이 만족을 하면 바로 상태 UNSAFETY로 진입하게 함으로 기술할 수 있다. 상태차트의 전이 우선 순위가 UNSAFETY_CONDITION이 만족할 때, 다른 정이보다 우선순위를 가짐으로 시스템의 안전성 조건을 기술할 수 있다.

4.2. 사례 연구

+ 모 듈 : 태스크 생성 서비스
+ 입력 조건 : 커널 혹은 태스크가 호출하며, 우선순위을 입력한다.
+ 출력 출력 : 태스크가 생성된다. 혹은 에러로 인해 종료된다.
+ 요구 사항 :
SWR20. pCOS는 태스크 또는 시작 소프트웨어가 태스크를 만드는 서비스를 제공한다.
SWR20.1. 이 서비스는 각각의 생성된 태스크는 유일한 우선순위 등급을 가지도록 한다.,
SWR20.2. 이 서비스는 새로운 우선순위가 이미 다른 태스크에게 할당되어있는지를 호출한 태스크나 시작코드에게 알려준다.
SWR20.3. 이 서비스는 명기된 우선순위를 사용할 수 없는지를 호출한 태스크에게 알려주고 종료한다.
SWR20.4. 이 서비스는 태스크를 생성하는데 필요한 자원들을 얻을 수 있는가를 호출한 태스크에게 알려주고 종료한다
SWR20.5. pCOS는 64개의 태스크를 만들 수 있도록 한다.
SWR20.6 운영체제가 수행이 시작한 후, 태스크가 생성되면, 재스케쥴링을 요청한다.
SWR20.7 운영체제가 수행 중이 아닌 때에, 태스크가 생성되면, 재스케쥴링을 요청하지 않는다.

그림 3 실시간 운영체제 태스크 생성 기능의 요구사항

다음 예제는 실제 원자력 발전소의 실시간 운영체제의 모듈에 대한 요구 사항이다. SWR20은 입력조건을 명세 한다. SWR20.1~3는 우선순위가 유일하지 않으면 에러를 발생시킬 것을 명세하고 있다. SWR20.4는 태스크 생성에 필요한 자원을 얻을 수 있는지를 확인한다. 자원을 얻을 수 없으면 호출한 태스크에게 알려준다. SWR20.1~3은 우선 순위 확인 기능을 기술한다. 이 때 제어 조건은 만약 " 우선순위가 이미 다른 태스크에게 할

당" 되었다면 태스크 생성을 요청한 기능에 알리고 태스크 생성 기능을 종료한다.
본 논문에서 제안하는 소프트웨어의 요구 사항의 상태차트 모델링 순서는 다음과 같다.

1) 자연어 기능 요구 사항 중, 행위 요구 사항, 상태 요구 사항, 입, 출력 사항 및 조건 사항 등을 구분한다.
2) 행위 요구 사항 및 상태 요구 사항을 상태차트의 상태로 연결시키고 이름을 명시한다.
3) 상태 차트 중에서, 중복되는 상태를 제거한다.
4) 입력 사항을 추출하여 상태차트의 default 전이의 조건으로 명시한다.
5) 출력 사항을 추출하여 상태차트의 종료노드의 진입 전이 레이블의 조건으로 명시한다.
6) 행위 요구 및 상태 요구 사항의 상태를 나열하고 전이 조건에 따라 이를 연결한다.
7) 안전성 요구 사항을 기술하기 위해, 상태 요구 사항을 표현하는 상태를 구분하고, 안전성을 위배하는 조건을 구분하여 명시한다.

5. 결론

본 논문에서는 상태차트를 이용하여 자연어로 기술된 소프트웨어 요구 사항을 해석하는 절차를 제공한다.
상태 차트는 다양한 도구를 이용하여 시뮬레이션과 검증이 가능하다. 하지만 자연어 명세를 직접 상태 차트에 적용시키기에는 무리가 있다. 본 논문에서는 이를 위해, 요구 사항의 기능 명세의 문장의 형태를 분석하고, 이를 상태 차트의 문법적인 요소와 연결시켜, 자연어 명세에서 직접 상태차트를 추출하는 방법과 예를 보였다.
이 방법은 현재 원자력 발전 실시간 운영체제를 상태차트로 기술하는 방법으로 이용되고 있다.
향후 연구로서는 이러한 요구 명세로부터 설계 명세를 작성하는 정형화 된 방법을 찾고, 요구 사항과 설계 사항의 부합성을 증명하는 방법을 찾는 것이 될 것이다.

참조문헌

[1] J.M. Wing, A Specifier's Introduction to Formal Methods. IEEE *Computer*, 23(9):8-24, September 1990.

[2] Bruce Power Douglass, " Real-time UML: Advances In The UML for Real-time Systems" , Addison-Wesley, 2004

[3] David Harel, STATECHART: A VISUAL FORMALISM FOR COMPLEX SYSTEMS, Science of Computer Programming 8 (1987) pp231-274

[3] David Harel and Ammon Naamad, *" The STATEMATE Semantics of Statecharts"* , ACM Trans. Soft. Eng. Method. Oct. 1996

[5] http://www.ilogix.com

[6] Hye Yeon Kim and Frederick T. Sheldon. *"Testing Software Requirements with Z and Statecharts Applied to an Embedded Control System,"* Software Quality Journal, 12, 231-264, 2004. Kluwer Academic Publishers. The Netherlands.

[7] IEEE 830-1998, Recommended Practice for Software Requirements Specifications Institute of Electrical and Electronics Engineers, May 1998,

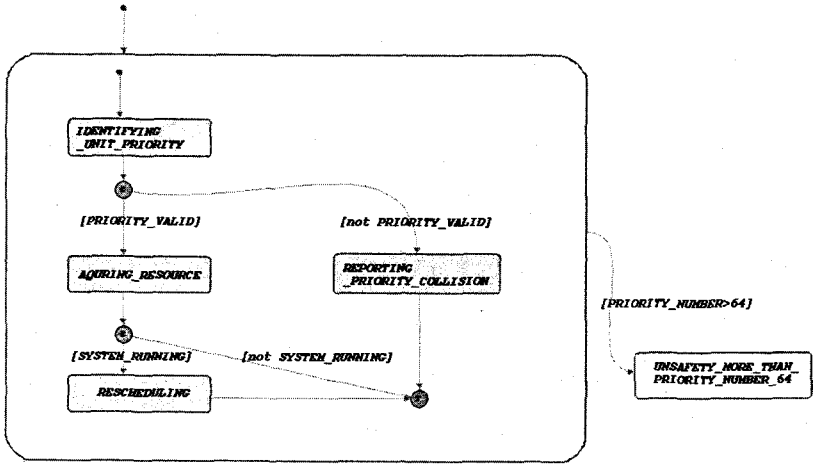


그림 4 요구 사항의 상태차트 모델