Statechart 행위 모델 기반의 테스트 오러클 생성 지원 환경

신동익, 전태웅
고려대학교 전산학과
e-mail :{eastwing, jeon}@korea.ac.kr

The Support Environment for Constructing Test Oracles Based on the Behavioral Models Represented as Statecharts

Dong-ik Shin, Taewoong Jeon
Dept. of Computer Science, Korea University

요 약

소프트웨어 시험을 효과적으로 수행하기 위해서는 시험 대상 소프트웨어가 명시된 행위 명세를 올바로 준수하는지 점검할 수 있는 테스트 오러클의 개발이 요구된다. 본 논문에서는 Statechart로 표현된 행위 모델로부터 실행 가능한 테스트 오러클의 자동 생성을 지원하는 시험환경을 제안하고 설명한다.

1. 서론

소프트웨어 시험은 소프트웨어의 신뢰성에 직접적으로 영향을 미칠 수 있는 방법들 중의 하나이다. 그러나 소프트웨어 시험은 일반적으로 많은 비용과 노력이 드는 개발 단계이다. 따라서 경제적으로 신뢰성 높은 소프트웨어를 개발하기 위해서는 테스트케이스 생성기, 테스트 제어기, 테스트 감시기, 그리고 테스트 오러클과 같은 소프트웨어 시험성 강화 메커니즘들이 요구된다.

본 논문에서는 테스트 오러클에 초점을 둔다. 테스트 오러클은 테스트 실행에 대한 허용 가능한 행위를 규정하고, SUT(Software Under Test: 시험 대상 소프트웨어)가 명세에 명시된 기능을 올바로 수행하는지를 판단할 수 있는 수단을 제공하는 도구이다[1]. 신뢰성 있는 소프트웨어 시스템들을 효율적을 개발하기 위해서는 테스트 오러클의 생성 자동화를 지원할 수 있는 시험 환경이 요구된다.

기존의 테스트 오러클 생성 방법들은 대부분 수작업을 통한 개발 방법들이다[3, 4, 5]. 테스트 오러클을 수작업으로 개발한다면, 테스트 오러클의 개발 비용이 SUT자체 개발 비용 만큼이나 많이 들 수 있고 테스트 오러클 개발자의 오류로 인해 테스트 오러클이 SUT의 고장들을 찾지 못하거나 정상적인 결과를 고장으로 오인할 수 있다. 그리고 기존의 상태 의존적인 SUT에 대한 테스트 오러클들은 대부분 단순한 상태/전이 다이어그램 기반의 테스트 오러클들이다[3, 4, 6]. 단순 상태/전이 다이어그램은 상태들 간의 계층성, 동시성 등을 지원하지 못한다. 따라서 단순한 상태/전이 다이어그램 기반의 테스트 오러클은 상태들 간의 계층성과 동시성을 갖는 SUT의 행위에 대한 시험을 지원하지 못한다.

본 논문은 Statechart[2]로 기술된 SUT의 행위 모델로부터 테스트 오러클의 자동 생성을 지원하는 시험 환경을 제안하고 설명한다. 본 논문의 테스트 오러클 생성 환경은 계층성과 동시성을 지원하는 Statechart 행위 모델로부터 실행 가능한 테스트 오러클의 자동 생성을 지원한다.

본 논문의 구성은 다음과 같다. 2절에서는 Statechart 행위 모델 기반의 전반적인 SUT 시험 환경을 설명하고, 3절에서는 Statechart 행위 모델 기반의 테스트 오러클 생성 지원 환경에 대하여 기술한다. 마지막으로, 4절에서 결론 및 향후 연구를 기술한다.

2. Statechart 행위 모델 기반의 SUT 시험

그림 1은 Statechart 행위 모델 기반의 전반적인 시험 환경을 구성하는 요소들간의 데이터 흐름을 보여주는 블록 다이어그램이다.

시험자는 Statechart 테스트 오러클의 생성을 위해 시험자로부터 Statechart 행위 모델과 매핑 규칙 모델을 테스트 케이스 생성기와 테스트 오러클 생성기에 전달한다. Statechart 행위 모델은 Statechart로 명세된 SUT의 기대 행위 모델이다. 매핑 규칙 모델은 테스트 입력들에 대한 Statechart 테스트 오러클의 기대결과와 SUT의 실제결과 간의 매핑 관계를 정의한 모델이다.

테스트 케이스 생성기는 테스트 제어기로부터 전달받은 Statechart 행위 모델을 이용하여 SUT를 위한 테스트 스위트를 생성하고, 테스트 오러클 생성기는 Statechart 행위 모델 이용하여 SUT를 위한 Statechart

테스트 오러클을 생성한다.

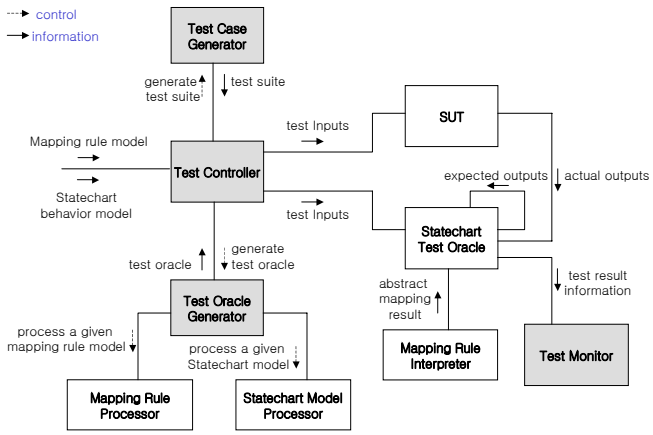

그림 1: Statechart 행위 모델 기반의 시험 환경 구성 요소들

테스트 제어기는 테스트 케이스 생성기에 의해 생성된 Statechart 기반의 테스트 스위트를 전달 받아 테스트 입력들을 SUT와 Statechart 테스트 오러클에 동시에 제공하여 실행시킨다. 테스트 입력에 대해, SUT는 시험 실행 결과를 생성하고 Statechart 테스트 오러클은 시험 기대 결과를 생성한다. SUT의 실행 결과는 SUT의 변수들에 저장되어 있는 현재 값(current value)들의 조합으로써 표현된다. Statechart 테스트 오러클의 기대 결과는 Statechart 행위 모델 상의 현재 상태(current state)로써 표현된다.

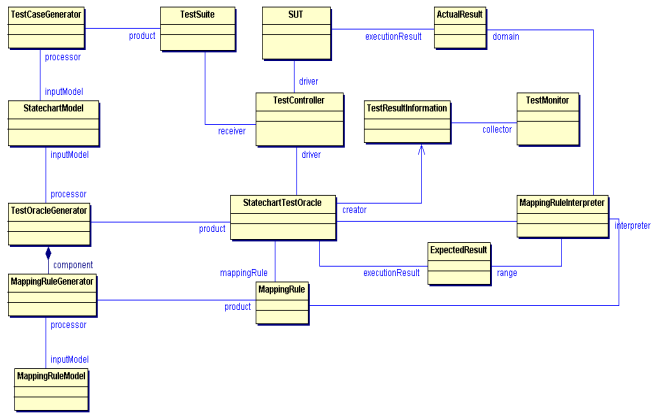

그림 2: Statechart 행위 모델 기반의 SUT 시험 환경

일반적으로 SUT의 값 공간(value space)은 테스트 오러클의 값 공간과 다르기 때문에, 이들 간의 관계를 매핑 규칙 모델로 정의하고 이를 매핑 규칙 인터프리터가 해석한다. 매핑 규칙 인터프리터는 매핑 규칙을 해석하여 SUT의 실제결과에 대응하는 테스트 오러클 상의 기대 결과를 도출한다. 매핑 규칙 인터프리터가 도출한 테스트 오러클의 기대 결과와 Statechart 테스트 오러클의 기대 결과는 일치해야 한다. 매핑 규칙 모델은 테이블 형태로 표현되고, Statechart 행위 모델 상의 한 추상 상태로 결정되기 위한 조건식과 이를 만족하는 추상 상태의 쌍들로 이루어진다.

Statechart 테스트 오러클은 또한 시험 결과 정보를 생성한다. 시험 결과 정보는 SUT의 시험 실행 결과, Statechart 테스트 오러클의 시험 기대 결과, 그리고 이들 결과들에 대한 비교 평가 결과들에 대한 정보들을 의미한다. Statechart 테스트 오러클에 의해 생성된 각 테스트 입력들에 대한 시험 결과 정보들은 테스트 감시기에 의해 수집 또는 저장된다. 테스트 제어기는 테스트 감시기에 의해 수집 또는 저장된 시험 결과 정보를 이용하여 동일 SUT의 재시험 또는 회귀 시험에 이용한다.

그림 2는 본 연구에서 제시한 Statechart 행위 모델 기반의 SUT 시험 환경에 대한 전체적인 관계를 표현한 클래스 다이어그램이다.

3. Statechart 테스트 오러클 생성 지원 환경

본 절에서는 2절에서 설명한 Statechart 행위 모델 기반 시험 환경 내의 테스트 오러클 생성 지원 환경에 초점을 맞춘다.

그림 3는 Statechart 테스트 오러클을 생성할 때 관련되는 구성 요소들과 이들 간의 제어 또는 정보 흐름을 나타내는 블록 다이어그램이다.

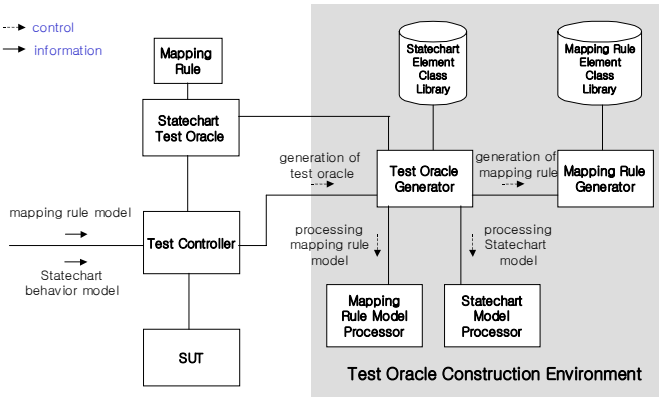

그림 3: 테스트 오러클 생성에 관련 있는 구성 요소들간의 데이터 흐름

Statechart 테스트 오러클 생성 지원 환경은 SUT의 Statechart 행위 모델과 매핑 규칙 모델을 이용하여 Statechart 테스트 오러클을 생성한다. 생성된 Statechart 테스트 오러클은 매핑 규칙을 구성 요소로 포함한다.

테스트 제어기로부터 오러클 생성 요청을 받으면, 테스트 오러클 생성기는 Statechart 행위 모델과 매핑 규칙 모델들에 대한 모델 검사를 Statechart 모델 처리기와 매핑 규칙 모델 처리기에게 각각 요청한다. 모델 처리 과정에는 문법 검사가 포함된다.

Statechart 모델 처리기와 매핑 규칙 모델 처리기의 산출물들은 Statechart 테스트 오러클 생성기와 매핑 규칙 생성기가 각각 실행 가능한 Statechart 테스트 오러클과 해석 가능한 매핑 규칙을 생성하는데 용이한 형태를 갖는다. 다시 말해, 두 모델 처리기는 실행 가능한 Statechart 테스트 오러클 및 매핑 규칙 생성을 위한 전처리기 역할을 수행한다. Statechart 모델 처리

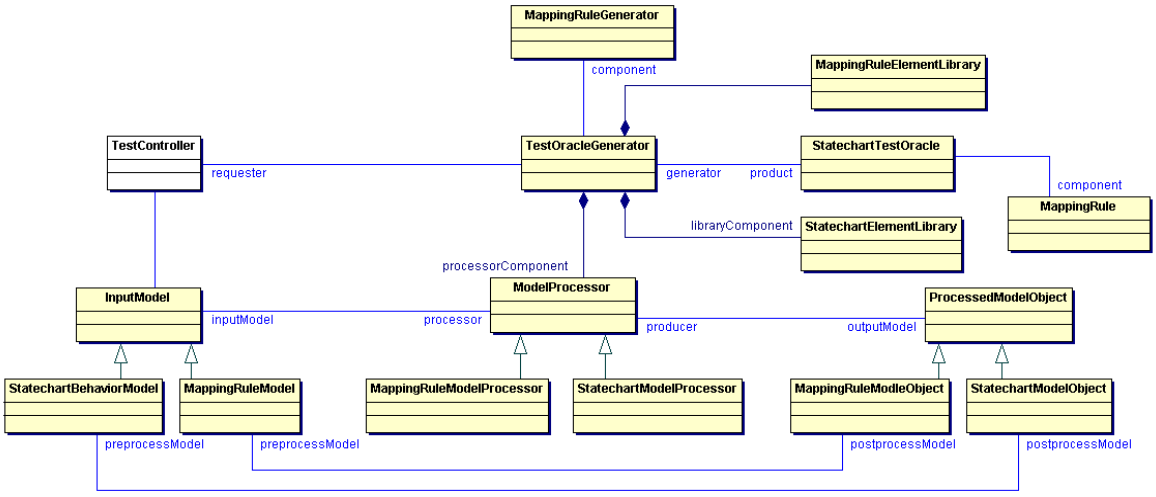

그림 4: 테스트 오러클 생성 관련 클래스 다이어그램

기와 매핑 규칙 모델 처리기에 의해 가공 처리된 Statechart 행위 모델과 매핑 규칙 모델은 각각 정보의 내부 표현 형태만 변환되어졌을 뿐 동일한 정보를 갖고 있는 객체 모델들이다.

그림 4는 테스트 제어기, 테스트 오러클을 생성하기 위해 입력된 모델들, 테스트 오러클 생성기와 모델 처리기들, 그리고 모델 처리기에 의해 생성되는 모델들 간의 관계에 초점을 맞추어 나타낸 클래스 다이어그램이다.

Statechart 행위 모델과 매핑 규칙 모델이 각각의 모델 처리기에 의해 가공된 후, 테스트 오러클 생성기는 우선 가공된 Statechart 행위 모델을 이용하여 실행 가능한 Statechart 테스트 오러클을 생성한다. 그 다음, 테스트 오러클 생성기는 매핑 규칙 모델 처리기 통해 가공된 매핑 규칙 모델을 인터프리터가 해석 가능한 매핑 규칙으로 생성한 후, 이를 Statechart 테스트 오러클의 컴포넌트로 끼워넣는다. 다시 말해, 생성된 Statechart 테스트 오러클은 매핑 규칙을 구성요소로 포함한다.

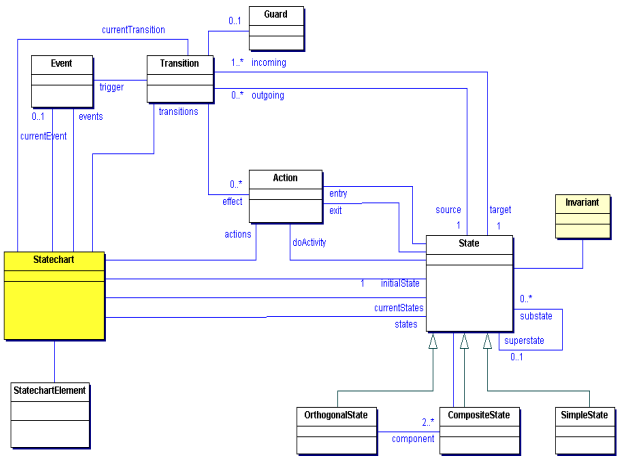

그림 5: Statechart 의미 메타모델 클래스 다이어그램

테스트 오러클 생성기는 Statechart 모델 검사기에 의해 가공 처리된 모델을 이용하여 실행 가능한 Statechart 테스트 오러클을 생성할 때 오러클 생성 환경의 구성 요소들 중의 하나인 Statechart Element Class Library를 이용한다. Statechart Element Class Library는 Statechart 의미 모델(그림 5)의 각 요소들을 구현한 클래스들이다. 따라서 Statechart 테스트 오러클은 Statechart Element Class Library에서 제공되는 클래스들의 객체들로 구성된다.

매핑 규칙 모델 처리기는 매핑 규칙 모델을 생성할 때 Mapping Rule Element Class Library를 이용한다. Mapping Rule Element Class Library는 매핑 규칙 모델의 의미 모델(그림 6)의 각 요소들을 구현한 클래스들이다.

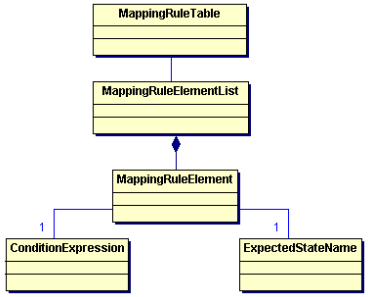

그림 6: 매핑 규칙 모델의 의미 메타모델 클래스 다이어그램

Statechart 모델 처리기와 매핑 규칙 모델 처리기에 의해 처리된 Statechart 모델(Processed Statechart model)과 매핑 규칙 모델(Processed mapping rule model)은 테스트 오러클 생성기와 매핑 규칙 생성기가 테스트 오러클 생성 및 매핑 규칙 생성할 때 필요한 모델 정보를 접근할 수 있는 인터페이스를 제공한다.

그림 7은 Statechart 행위 모델 기반의 테스트 오러클 생성 환경을 초점을 둔 SUT 시험 환경을 나타낸다. 그림 7의 음영 부분은 테스트 오러클을 생성을 지원하는 구성 요소들을 의미한다. 그림 7의 음영 밖에 표현된 클래스들은 테스트 오러클 생성 지원 환경에 의해 생성된 테스트 오러클을 이용하는 SUT 시험 환경을 간략하게 표현한 것이다.

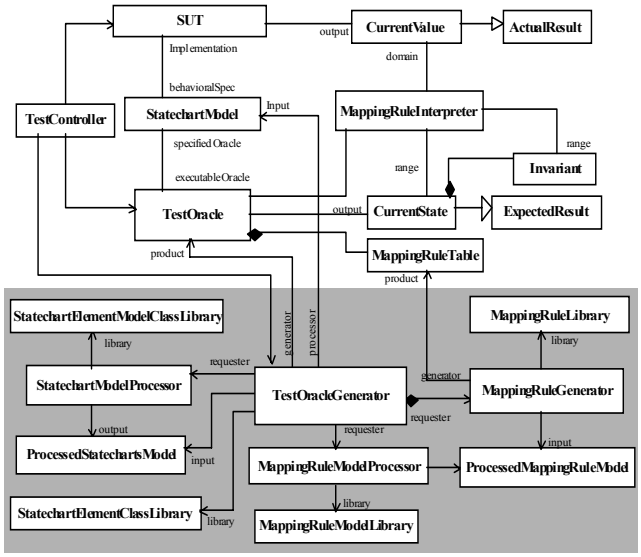

그림 7: Statechart 행위 모델 기반 테스트 오러클 생성

4. 결론 및 향후 연구

지금까지, 본 논문에서 Statechart로 기술된 행위 명세를 기반으로 하여 실행 가능한 테스트 오러클을 자동 생성할 수 있는 시험 환경을 제안하고 설명하였다. 본 논문의 제안한 테스트 오러클은 단순한 상태 의존적인 행위 뿐만 아니라 계층성과 동시성을 지닌 SUT의 시험에 사용될 수 있다.

본 논문에서는 설명하지 않았지만 SUT의 행위 명세에 사용되는 Statechart의 의미 모델을 Object-Z[7]로 별도로 정형 명세하였다. 이는 현재 테스트 오러클의 생성 규칙으로 사용되었지만, 향후에는 Statechart 행위 모델 인터프리터 개발의 토대로 이용할 것이다.

현재 본 논문의 Statechart 테스트 오러클은 SUT 시험의 전이 발생 시 이벤트에 의한 상태 변화만을 고려하였다. 전이 발생 시 수행되는 액션들에 대한 처리는 고려되지 않았다. 향후 이를 추가하여 보다 정밀하고 정확한 테스트 오러클로 확장해 나갈 계획이다. 또한 효과적인 SUT 시험을 지원하기 위해서, Statechart 행위 모델 기반의 테스트 오러클 뿐만 아니라 테스트케이스를 자동 생성을 지원하기 위한 방법에 대해 연구해 나갈 계획이다.

참고문헌

[1] D. J. Richardson, S. L. Aha, T. O. O'Malley, Specification-based Test Oracles for Reactive Systems," International Conference on Software Engineering, 1992, pp. 105 –118.

[2] D. Harel, Statecharts: A Visual Formalism for Complex Systems", Science of Programming 8, 1987, pp. 231-274.

[3] D. Hoffman and P. Strooper, "ClassBench: a Framework for Automated Class Testing," Software Maintenance: Practice and Experience, Vol. 27, No. 5, May 1997, pp. 573-597.

[4] J. McDonald, P. Strooper, "Translating Object-Z Specifications to Passive Test Oracles," The 2nd International Conference on Formal Engineering Methods, 1998, pp.165 -174.

[5] C. Hunter, P. Strooper, "Systematically Deriving Partial Oracles for Testing Concurrent Programs," 24th Australasian Computer Science Conference, 2001, pp. 83-91.

[6] S. Antoy and D. Hamlet, "Automatically Checking an Implementation against Its Formal Specification", IEEE Transactions on Software Engineering, Vol. 26, No. 1, January 2000, pp. 55-69.

[7] Graeme Smith, "The OBJECT-Z Specification Language. Advances in Formal Methods", Kluwer Academic Publisher, 2000.