제 9 장. 모델 검사하기

이 장에서는 소프트웨어 모델의 검사에 관해서 설명합니다. 검사 방법 및 검사에 사용되는 규칙들을 설명합니다.

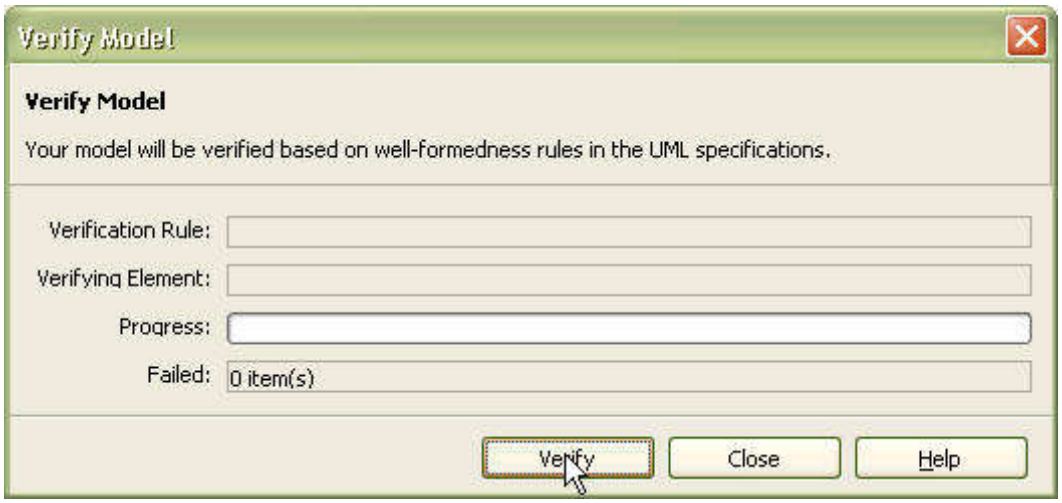
- 모델 검사하기
- 기본적인 검사 규칙들

모델 검사하기

UML로 소프트웨어를 모델링하다 보면 수 많은 실수를 하게 됩니다. 그러나, 그런 사소한 실수들이 시간이 지난 후 치명적인 오류를 유발할 수도 있으므로 사전에 그런 가능성을 미리 찾아내는 것이 중요합니다. StarUML™에서는 UML의 기본적인 규칙들을 적용하여 소프트웨어 모델을 검사할 수 있습니다.

모델을 검사하는 방법:

1. **[Model]->[Verify Model]** 메뉴를 선택합니다.
2. 모델 검사 대화상자가 나타나면 **[Verify]** 버튼을 누릅니다.



3. 검사가 끝나면 정보 영역의 **[Message]** 부분에 검사를 통과하지 못한 요소들과 오류 내용이 나타나고, 그것을 더블 클릭하면 해당 요소를 찾아갑니다.

기본적인 검사규칙들

모델을 검사하기 위한 38가지의 규칙들이 정의되어 있습니다. 이 정의들은 주로 UML 명세의 Well-formedness Rule들에 따라 정의되었습니다.

모델 검사 규칙 목록

번호	규칙내용	Elements Applied
1	연관(Association)내의 연관끝(AssociationEnd)들은 서로 유일한 이름을 가져야 합니다.	Association
2	집합(Aggregation) 혹은 합성(Composition)인 연관끝(AssociationEnd)은 하나의 연관(Association) 내에서 두 개 이상 존재할 수 없습니다.	Association
3	파라미터(Parameter)는 서로 유일한 이름을 가져야 합니다.	BehavioralFeature
4	분류자(Classifier) 내에서는 동일한 이름의 속성(Attribute)이 존재할 수 없습니다.	Classifier
5	반대측 연관끝(AssociationEnd)들의 이름들은 서로 유일해야 합니다.	Classifier
6	속성(Attribute)의 이름은 반대측의 연관끝(AssociationEnd) 혹은 분류자(Classifier)에 포함된 요소의 이름과 동일할 수 없습니다.	Classifier
7	반대측 연관끝(AssociationEnd)의 이름은 분류자(Classifier)에 포함된 요소나 그것의 속성(Attribute)의 이름과 동일할 수 없습니다.	Classifier
8	루트(Root) 요소는 더 일반화된 요소를 가질 수 없습니다.	GeneralizableElement

9	말단(Leaf) 요소는 더 특수화된 요소를 가질 수 없습니다.	GeneralizableElement
10	순환적인 상속(Inheritance) 구조는 허용되지 않습니다.	GeneralizableElement
11	인터페이스(Interface)의 모든 특징(Feature)들은 공개(Public) 이어야 합니다.	Interface
12	컴포넌트 인스턴스(ComponentInstance)는 자신의 원시(origin)로써 정확히 하나의 컴포넌트(Component)를 지정해야 합니다.	ComponentInstance
13	노드 인스턴스(NodeInstance)는 자신의 원시(origin)로써 정확히 하나의 노드(Node)를 지정해야 합니다.	NodeInstance
14	연관끝-역할(AssociationEndRole)은 역할(ClassifierRole)로 연결되어야 합니다	AssociationEndRole
15	역할(ClassifierRole)은 자신만의 특징(Feature)을 가질 수 없습니다.	ClassifierRole
16	역할(ClassifierRole)은 다른 어떤 역할(ClassifierRole)의 역할이 될 수 없습니다.	ClassifierRole
17	메시지(Message)의 송신자(sender)와 수신자(receiver)는 반드시 해당 인터랙션(Interaction)의 문맥인 협동(Collaboration)에 참여하는 것이어야 합니다.	Message
18	액터(Actor)는 유스케이스(UseCase), 클래스(Class) 혹은 서브시스템(Subsystem)으로 연결되는 연관(Association)만을 가질 수 있습니다.	Actor
19	복합상태(CompositeState)는 최대 1개의 초기 상태(Initial state)만을 가질 수 있습니다.	CompositeState
20	복합상태(CompositeState)는 최대 1개의 깊은 이력(Deep history) 만을 가질 수 있습니다.	CompositeState
21	복합상태(CompositeState)는 최대 1개의 얕은 이력(Shallow history) 만을 가질 수 있습니다.	CompositeState
22	동시성 복합상태(concurrent composite state)는 최소한 2개 이상의 복합상태를 포함해야 합니다.	CompositeState
23	동시성(concurrent) 상태는 하위상태(sub state)로써 복합상태(composite state)만을 가질 수 있습니다.	CompositeState
24	최종상태(Final state)는 나가는 전이(transition)를 가질 수 없습니다.	FinalState
25	초기상태(Initial state)는 나가는 전이(transition)를 최대 1개를 가질 수 있고, 들어오는 전이(transition)는 가질 수 없습니다.	Pseudostate
26	이력(History) 상태들은 최대 1개의 나가는 전이(transition)를 가질 수 있습니다	Pseudostate
27	점합점(junction vertex)는 최소한 들어오는 전이(transition) 1개, 그리고 나가는 전이(transition) 1개는 가져야 합니다.	Pseudostate
28	선택점(choice vertex)는 최소한 들어오는 전이(transition) 1개, 그리고 나가는 전이(transition) 1개는 가져야 합니다.	Pseudostate
29	상태머신(StateMachine)은 분류자(Classifier) 혹은 행위적 특징(BehavioralFeature) 둘 중 하나에 결합될 수 있습니다.	StateMachine
30	최상위 상태(top state)는 항상 복합상태(composite)이어야 합니다.	StateMachine
31	최상위 상태(top state)를 포함하는 상태는 존재할 수 없습니다.	StateMachine
32	최상위 상태(top state)로부터 나가는 전이(transition)는 존재할 수 없습니다.	StateMachine
33	하위-머신상태(SubmachineState)는 동시성을 가질 수 없습니다.	SubmachineState
34	의사상태(pseudostate)로 향하는 전이(transition)는 트리거(Trigger)를 가질 수 없습니다.	Transition
35	활동그래프(ActivityGraph)는 패키지(Package), 분류자(Classifier) 혹은 행위적 특징(BehavioralFeature) 중 하나의 동적행위를 묘사할 수 있습니다.	ActivityGraph
36	활동상태(ActionState)는 내부전이(internal transition), 퇴거-행위(exit action), 주-행위(do activity)를 가질 수 없습니다.	ActionState
37	활동상태(ActionState)으로부터 나오는 전이(transition)은 트리거 이벤트(trigger event)를 가질 수 없습니다.	ActionState
38	하위-활동상태(SubactivityState)는 활동그래프(ActivityGraph)로의 연결을 가져야 합니다.	SubactivityState