Detecting Model Inconsistency through Operation-Based Model Construction

MF1233055 周昊一

大型的软件系统的开发往往牵涉到众多的开发人员以及各种不同的模型.确保这些模型之间的一致性是一件非常重要的事,因为即便是微小的不一致也可能导致整个系统的严重错误,甚至导致项目的失败.

文章的作者提出了一个观点,即结构和方法上的不一致检测应该被统一地支持.方法上的规则限制所有的模型构建过程,而结构上的规则限制了这个过程最后获得的模型.从模型的生存周期的角度来讲,结构上的规则制约模型的状态,而方法上的规则制约模型的变化.

进一步地讲,不论一个模型的元模型是什么,不一致性检测都应该被统一地支持.为了对大型系统提供良好的支持,有的规则可能需要是面向多个元模型的.

在这篇文章中,作者提出了一个用于统一地检测模型结构以及方法上的不一致性的方法.这个方法并不检测模型中的元素,而是检测模型构建过程中的操作序列.

模型可以表示为一系列模型元素以及对其他模型的引用的集合.同时它也可以看作是由许多列基本构建操作的序列构成的.在此作者定义了四种基本操作:

1. create(me,mc),根据meta-class mc创建一个模型元素me的实例.只有当模型中没有me的时候,它才能被创建.
2. delete(me),删除一个模型元素,只有当它存在,并且不被任何其他元素所引用时,才能被删除.
3. setProperty(me,p,Values),将me中属性p的值设为Values中对应的值.只有当me存在,p有定义,并且值的类型以及数量和Values中的相同时,才能被设置.
4. setReference(me,r,References),将me中的引用r设为References中对应的值.只有当me存在,r有定义,并且r中值的类型以及和References中的相同时,才能被设置.

在这四个基本操作的基础上,作者使用谓词逻辑来定义这四个操作的不同组合所引起的结果.再用这些谓词来定义一系列的不一致行为或者状态.比如ActorsDoNotOwnUseCase表示在某个me的最后一次创建后,并没有被引用.

在实现部分,作者用prolog完成了一个系统的具体实现,并且可以用于Eclipse EMF与Rational Software Architect.在这个基础上,进行了一些实验,表明这个理论是有效的.