软件开发和运营的建模 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/141/2021_2022__E8_BD_AF_E4_BB_B6_E5_BC_80_E5_c29_141527.htm 摘要： 我们用许多模型来设计、开发、部署和管理技术解决方案。其中一些模型包括商业案例（business cases）、用例（use case）图解、实体关系模型、对象模型、代码、测试套件、开发计划、逻辑数据中心模型和异常管理计划。软件生命周期方法已经努力发展成可以自动化的从“ 上游 ”模型产生“ 下游 ”模型，甚至可以在不同的模型间保持同步。 本文简要描述了建模空间，概览了当前的发展，这些都可以在整个技术生命周期中提高模型的使用效率。 模型是真实世界中对象和系统的“ 简化 ”抽象,可忽略大小、细节和外观(例如，一个关注成本或持久性，而忽略不相关系统因素的模型)。 在信息技术解决方案的开发中，模型被用来控制复杂度，并在客户、解决方案和系统架构师、开发者和运营人员之间传达系统需求。 以工程项目大量使用的模型为例：用例模型可表达系统的高级功能需求和需要被支持的角色；风险模型可通过尽早地在项目周期中规避风险来优化工作；实体关系模型捕捉被解决方案管理的基本信息，并且提供将其适当分解为表格、对象和服务的建议；逻辑系统模型组成了开发和运营之间的通信基础；逻辑系统模型可以进行需求和解决方案策略的早期确认，等等。 所有这些模型都使项目参与者能在最终系统的开发和管理中发挥其专长。所有这些模型都通过尽早避免误解和疏忽来减少项目的代价和风险。建模工具和框架通过在各个级别的方案设计中提供更高级别的追溯性、可视性和说明性促进了

商业部门和信息技术团体的合作。 在整个的软件生命周期-从开发概念形成到运营和管理中，微软关注如何使模型更有使用价值。由于历史原因，在生命周期中通行的仅有的模型是含义模糊、冗余和很不完善的模型，而且是使用系统源代码所描述。系统模型之间的不协调经常在组织内部和其间造成理解和沟通错误，导致项目和系统的失败。 微软的目标是让所有的项目参与者--技术人员，专家和管理者都能够获取公司组织系统最适时、精确和有效的描述，并表达为各自熟悉的语言。微软的意图是使所有的项目参与者--从商业分析员到数据架构师，还有安全专家和网络工程师，在解决方案的开发过程中尽可能的发挥他们的专长，尽量减少信息和知识的损失。 某些模型力图在多个抽象层面上描述一个完全的系统。另外的模型则着重于系统的某些特定方面，诸如怎样保持安全性，或者从系统的各个方面跟踪性能表现。某些模型与解决的创建和开发过程有关，还有一些则预测并分析其在运营中的行为。 用例图有可能是软件工程中的最简单的“ 完全 ”模型：系统的用例集合建立了对系统功能的展望，为系统用户确定角色，并且提出了对运营的需求。用例可以发展成商务过程模型，派生出详尽的需求模型，数据模型，对象模型，和最终的可编辑模型。 这些模型的层叠使人们联想起软件工程中熟悉的“ 瀑布 ”的概念。 系统建模的一个更好的概念是一系列的锁定。不是随随便便的，被工具支持的系统化过程应该在良好的管理下从一个模型递交给下一个模型。递交过程应该是平滑可预见的。 最重要的是，这些递交应该是双向的。用例应可以被转换成为商业过程模型，而商业过程模型也应该是能转换成用例的(虽然由于信息损失，你不能

全保真的完成从客观到抽象再到客观的完整转换周期)。理想情况下，这种映射的阶梯流要允许所有描述系统的模型之间的同步。 在项目中支持这些相互映射模型的集合，将产生模型驱动 (model-driven development MDD)的开发模式。
在MDD中，模型成为开发过程中的最重要的原材料之一。 当一个系统架构模型允许或禁止横跨企业数据中心安全区域的信息路径时，这个模型就指导和约束了服务模型，很有可能影响服务设计。当数据架构模型确定数据访问方法，工具就产生代理代码和语言捆绑。MDD并不是一种全新的想法；成功的实现了模型同步的例子包括，多视图数据建模工具，它可以使图例化模型，基于实体的模型，和相关的数据架构都保持同步，还有就是可以在开发中保持外观和代码模型一致性的集成开发环境设计器。 我们可以从这些案例中汲取成功经验以大大扩宽MDD的应用范围。 