DSM（领域定义建模）和MDA（模型驱动架构）[1] 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_DSM_EF_BC_88_E9_A2_86_E5_c29_141301.htm 模型在软件开发中的角色 当今信息系统的开发越来越复杂，而且所涉及到的领域也越来越广，开发者必须掌握许多不同的技术，包括流行的面向对象技术，XML，脚本语言，接口定义语言，过程定义语言，数据库定义和查询等等。要把来自于问题领域的需求转换成解决方案需要对许多架构和协议的深刻理解。再者，最终用户常常期望结果是高运行效率的，易用的，易扩展的，而且对于不可知且不可靠的网络连接是安全的，这可是件苦差事。 在软件开发之外的一些领域，例如电子产品（电视机，HiFi音响，照相机）等等，我们可以看到低成本和高可靠性的情况。在过去的几十年里，制造行业一直采用这样的流程：通过一连串复杂的步骤来制造一台电视机或汽车，其中有很多步骤是完全自动化的。 我们会喜欢使用相同的原理来构筑软件，不同的是我们没有开发出能够允许有效分离软件中关注点的软件说明语言。尽管我们使用不同的程序开发语言来书写应用逻辑，来完成不同的开发任务。例如：使用XML在应用组件中传递数据，使用SQL存取数据，使用WSDL来说明面向Web应用的组件的接口等等，但是它们中没有一个直接针对最终用户所面对的业务问题。 本文将要介绍的软件构筑技术是domain-specific languages（领域定义语言，简称DSL）的开发。DSL被设计为直接面向它所要解决的问题领域。在某种程度上，它能够代替编码，数据交换，配置等工作，我们常把这类语言称为建模语言。我们使用这些语

言来针对问题领域进行建模。 模型里的每个元素都映射到现实领域中的一个概念，很多年以来，模型对于定义IT系统如何来保存数据一直是很重要的，现在，模型的应用更广泛，例如对业务过程建模，服务的部署，数据中心等等。模型受欢迎是因为它能够很好地表述问题从而避免陷入技术细节中。当技术变得越来越复杂的时候，模型是提高生产力的必须手段。模型的另一个好处是可以让程序员和问题领域的专家使用同样的表述方法，这有助于团队成员间的沟通。我们也可以把使用模型看作弥合技术和业务之间缝隙的方法。 来源：www.examda.com 在过去的几年里，新的建模方法开始合并，特别是在MDA的旗帜和能够提高软件开发生产力及软件可移植性的承诺下，OMG对UML和一些相关技术进行了大力的推广。但是我们必须看到80年代的CASE工具的结果，显然，CASE已经无法兑现当初的诺言，对于MDA我们仍然十分怀疑，除非能够证明它对于我们所面对的问题找到了新的道路，不再重蹈CASE工具的覆辙。在我看来只有模型，模式，框架等技术结合在一起才能够避免象CASE工具那样的失败。Domain-Specific Languages 如果我们想通过运用模型来使领域专家能更容易地解决问题，那么模型就必须能够清晰地描述问题领域。对于建模语言，就是指用来针对问题领域建模的标记和关系的定义。典型的，但不是必须的，建模语言可以是一组图释，一组由线条连接起来的节点，也可以是流程图或者实体－关系图。 模型通常被标记和文本元素所修饰，开发者需要细心地审视，理解掩盖在这些修饰下的模型真正要表达的信息。所以要能够进行建模就必须明白每个元素相关细节。这意味着一旦成为具有专门的建模技能的人才就会带

来丰厚的回报。 有些人可能会认为正确的观点是应该定义一个通用的建模语言，使用它来对所有的问题领域建模，同时要教会那些领域专家们学会使用这个通用建模语言。从UML得到的经验来看，这样作并不成功，后面我们将会讨论UML。 现在，我们把那些被设计成用来对特定问题领域建模的建模语言称作domain-specific languages（领域建模语言）。领域建模语言可以针对很多问题领域创建，例如：通讯，银行业务，空间勘测等等。 无论如何，设计和使用领域建模语言只是模型被用作辅助软件开发的很小一部分。模型可以被分析和验证，转换，通过很多步骤，被部署并执行软件。这个过程包括开发，分析，验证模型，在很多不同领域中，通过工具将模型进行转换，直到部署系统完成。 在软件系统的构筑中，模型的一个对应物是框架，框架是适用于整个领域的代码实现框架，并且给多个系统间在相同领域的不同元素提供了扩展点。有很多框架的例子：从GUI到ERP的基础结构和算法。在所有的情况下，模型的角色是使用框架，给特定的应用定义扩展点。在这个层面上，我们可以认为建模语言是定义扩展点，使其契合到框架上，来适应用户对问题的理解的一种方法。 另一个重要的对应物是模式，一个模式本质上是一个有很多小孔的模型，和如何将这些小孔用其他模型来填充的规则。有一种高效使用模式的方法：使用一个由许多小的模型组成的大的模型，或一个由其他类型的模型组装起来的模型。 在软件开发过程中运用模型，模式，框架，代码的至关重要的一点就是最终的结果必须是“敏捷“的。在代码和模型间必须不存在任何不可逆性和不连续性，在整个开发过程中必须能够对可见的各种因素作出快速的反映，变化后

重新产生最终结果。CASE的错误在于没有针对问题领域使用框架，而是使用了庞大的，不可逆的代码生成过程，这样使得开发者无法修改生成的代码，从而使整个方法完全失效。只有将模型，模式，框架结合在一起，并使它们无缝的整合进一个敏捷的开发过程中，才可以避免出现CASE方法那样的缺陷。