DSM（领域定义建模）和MDA（模型驱动架构）[2] 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_141302.htm Model Driven Architecture 在IT界，术语MDA一般是指在软件开发过程中使用模型。但事实上，OMG把这个术语注册为商标，并将其引申为特殊的使用OMG的建模技术进行模型驱动开发的概念。使用的建模技术的核心是UML和MOF（Meta-Object Facility 元对象设施），本文的这部分将简要讨论MDA，然后将关注MDA中所包含的建模技术，特别是UML和MOF，还将讨论MDA中和我们相关的方法学。 MDA的本质就是区别Platform IndependentModels (PIMs) 和 Platform Specific Models (PSMs)。当使用MDA开发应用程序时，必须首先创建PIM（平台无关模型），然后使用标准映射，转换到PSM（平台定义模型），最后，映射生成最终程序代码，依照OMG的MDA的FAQ：http://www.omg.org/mda “ UML是MDA所使用的关键技术，任何使用MDA创建的应用程序都基于标准化的，平台无关的UML模型。 ”这样，就意味着应用程序的被定义为平台无关的，这样应用程序就是可移植的。这很容易让人回想其Java所宣称的“ write once run anywhere ”，试图去构建一个平台无关的框架，诸如Swing UI库，必须在性能和平台集成上作出折衷，在过去，这种折衷是很多产品失败的根源，因为这些失败，业界仍然非常怀疑MDA的宣言，在OOPSLA 2003上MDA的session就是佐证。 不过，MDA的探索在某些应用程序方面是有帮助的，一些厂商已经向我们展示了基于J2EE的Web应用，创建包含了数据实体，组件的UML模型，再映射到各

种J2EE应用。但是无论如何，就象前面所提到的，这对开发者意味着全面转向敏捷开发，而且不能引起不必要的转变和障碍，例如不可逆的代码生成过程和调试上的问题。 来源：www.examda.com 扩展MDA到其他领域很困难，OMG所定义的“ 平台 ”的概念很模糊，真正的例子也就是J2EE。在软件开发过程中使用模型的道路上，使用模型来创建J2EE应用是有效且使用的。事实上，几乎没有关于平台无关模型和平台依赖模型间的映射标准，现存的惟一一个也是针对Java平台的，尽管还有很多非标准的，开发社区的实现声称支持其他平台。 总而言之，MDA起错了名字，它不是体系结构，它是基于对相似平台的抽象的模型驱动开发标准。OMG在向业界推动MDA的时候，并没有采纳关于整和模型，框架，模式和工具来支持软件产品线的建议，而且，我们将看到，MDA所基于的UML和MOF规约将会限制它的用途。 The Unified ModelingLanguage UML是一种通用建模语言，它开发于90年代早期，由Grady Booch, JamesRumbaugh和Ivar Jacobson合并成一个统一的图形表示法。第一次标准化在1997年，经过了多次修订，最近正在开发第二个版本，这个版本已经接近完成。 UML是庞大且难于理解的，版本2更是如此，要向深入的理解UML必须先理解它怎样被使用。我们借用Martin Flower在《UML Distilled》一书中分类，Marting把UML的使用分为:用作草图，用作Blueprint，用作程序语言。 把UML当作草图使用非常流行，很多项目都在白板上使用UML画草图。 把UML作为草图使用的另一个含义是把试图从面向对象设计中生成结构化的文档被看作是不妥当的。在这种情况下，UML是非常成功的，它完全达到了消除了面向对象设计和

图解表示的不一致问题的目的。 把UML作为Blueprint使用提升了门槛，这时的目标是在开发过程中把多种UML模型结合起来。对于任何改动和自动化，都向系统地将UML模型转换到源代码，这也就意味这UML模型必须包含足够的信息，才能保证转换是有效且完整的。 当我们尝试这样作的时候，会很快发现UML的问题，因为它不能很直接的转换到我们所使用的技术，例如：一个UML类不能直接用来描述一个C#类，因为UML类并不能描述C#中的属性的概念。类似的，一个UML接口不能直接用来描述一个Java接口，因为UML不包括Java中的静态字段的概念。从这一点来看，把UML作为草图使用时，没有任何问题，但是，当UML被用作开发一个类的制品时，要么违反标准，要么引入一些不和谐因素来修补这些不匹配的问题。 将UML作为程序语言由一些社区支持，但是他们不喜欢走到商业化的路上，在这里我们不作讨论。让我们再来观察这些UML的主要使用方法：作为草图和Blueprint。把标准表述为一组灵活的，可扩展的图释，并无缝地映射到开发所使用的技术，而且不存在任何不匹配的描述说明，这是非常有用的。只有从模型所表述的概念进行无缝且可逆的映射才会被大多数开发者所接收。“ UML轮廓 ”采用允许有限度的对语言扩展来使蓝图具有可扩展性。但是实践证明这种可扩展性是非常有限的，并且还不能提供无缝的从UML到时下流行技术的映射。 在微软，我们的途径是通过我们的建模工具，使用一些易识别的扩展的表示法来表示概念。同时，我们发现UML表示法还不够清晰，我们对其进行了增补，例如：我们使.NET的类可视化，可以包含更多信息，更容易使用，而且使得图释的元素更精确。这保证

了.NET中的术语和概念能够在图释中清晰地表达。我们从客户那里得到的反馈是压倒性的支持这种作法。尽管我们的图释法不是标准的UML，但是它所表达的含义对任何人而言都是非常容易理解的。