链滴

软件工程形式化方法初学体会

作者：aopstudio

原文链接：https://ld246.com/article/1569238541461

来源网站：链滴

最近在上一门叫做新技术讲座的课，我估计好多同学是看到讲座这两个字感觉这课会很水才选的但实际上这门课讲的内容是非常硬核的软件工程形式化方法，而且授课老师也是国内第一批研究软件程形式化方法的教授，现任日本埼玉大学教授，且上课非常认真负责，每节课都有课堂练习。不过了了软件工程形式化的概念后，发现这个研究方向真的了不得，简直可以颠覆整个软件行业。
在早期的软件开发过程中，开发者们往往拿到问题后就直接开始编码，软件往往会出现各种各样问题，大部分软件的质量十分低下，被称为“软件危机”。为了解决软件质量低下的问题，人们提出软件工程化的思想，也就是目前主流的软件开发方式，引入了从需求分析到最后编码实现这一系列的程，以及需要遵守的一系列规范，并定义了如 CMMI 这样的规范性等级。然而软件工程里面有很多容的规范是非常主观的，没有一个确定的评定标准，另外很多步骤只能尽量争取正确，但根本无法确正确，比如对于软件的测试，在软件测试这门课中就了解到全覆盖基本上是不可能完成的，并提出了句覆盖、判定覆盖、条件覆盖等一系列只能尽量争取正确的测试覆盖。而且在执行的过程中，就算是MMI 5 级的开发单位也往往会为了追求进度而省去其中某些甚至很多步骤，导致最后开发出来的软根本不可用。可以说，软件工程化确实解决了一部分问题，但治标不治本，没有从根本上解决软件质低下的问题。
有没有一种方法可以从根本上解决问题呢？有，就是软件工程形式化方法。软件工程形式化的目简单来说就是把要解决的问题先用逻辑学和集合论等数学的方式把它抽象表示出来，也就是形式化，后就可以自动化的方式完成一些原本需要大量人工完成的步骤。它的效果是可以减少软件开发的成本同时大大减少软件中的错误。由于数学是稳定的、抽象的，可以应用到各个领域，可以作为一个相对确性的基础，因此我们就可以将其他事物的正确性建立在数学的正确性之上。另外，形式化方法可以大减少规格说明的二义性，防止出现客户和开发方对需求理解不一致的情况。我们可以这么理解，在件工程形式化方法中，只要规格说明(specification) 是正确的，那么最后的软产品一定是正确的，甚至连测试都不需要做。就如同建筑一样，只要图纸和设计方案正确，那么一个格的施工队一定能建造出正确的建筑，而且已经建造完毕的建筑也无法像软件产品一样测试之后发现误再进行修改。
为何建筑工程的精准性和正确性如此之高，其实就是引入了形式化方法的思想。在建筑工程设计，一定都会首先在图纸上对建筑进行力学分析和计算，确保最终建筑拥有完美的力学结构，或者至少容易塌。并且设计单位和施工单位往往不是同一家，但最后的施工成果依然能满足设计单位原先的要。而在软件工程中，对于数学的应用主要在算法、数据结构中，比较少地用来抽象具体事物以及约束们的关系。从这个角度看软件开发过程的先进性远远不如建筑工程的过程，尽管 IT 被认为是高新技。
在软件工程形式化方法中，规格说明的重要性大大提高了，同时在规格说明中，如何用数学语言不是自然语言抽象一个具体的事物是需要重点考虑的问题。可以看一下伦敦地铁线路图的抽象例子。br>
老图：<img src="https://ld246.com/images/img-loading.svg" alt="伦敦地铁原图.PNG" data-sr="https://b3logfile.com/file/2019/09/%E4%BC%A6%E6%95%A6%E5%9C%B0%E9%93%81%E%8E%9F%E5%9B%BE-47888a5e.PNG?imageView2/2/interlace/1/format/jpg">
新图：<img src="https://ld246.com/images/img-loading.svg" alt="伦敦地铁新图.PNG" data-sr="https://b3logfile.com/file/2019/09/%E4%BC%A6%E6%95%A6%E5%9C%B0%E9%93%81%E%96%B0%E5%9B%BE-1f0d6015.PNG?imageView2/2/interlace/1/format/jpg">
可以看到最后抽象的结果十分清晰明了、没有歧义、且具有可维护性，与之相比，老图尽管更精地反应了真实的地理位置，但却让读者感到混乱。目前几乎所有城市的地铁以及公交线路示意图都模了伦敦地铁图。软件工程形式化的规格说明也需要像这样排除无关事物的干扰，并且精准、无二义性比如，“令 x=3,y=x+1”就比“令 x 为一个存有 3 的内存单元，令 y 为存有 x+1 的值的内存单元”得多，尽管在计算机中实际运算的过程可能是后一种。
在软件工程形式化方法中，是不需要进行测试的，尽管利用形式化方法产生测试用例同样可以提效率，而且工业应用上也有相应的工具，但实际上并不推荐这样的用法。因为测试本质上就是效率低的，前面已经提到，测试根本无法保证所有可能的结果都正确。在软件工程形式化方法中，取代测试位的是证明(proof)。绝大多数关于软件工程形式化方法的文章中都提到了证明可见证明在软件工程形式化方法中的重要性。证明可以保证的正确性远远高于测试。工程师在构建抽模型的过程中，需要通过数学证明的方式逐步验证它。要证明的语句不需要工程师自己定义，而是由个叫做证明业务生成器的工具自动生成。为了生成证明语句，该工具将不断分抽象模型的各个精化形式。要做的证明需要关注两方面情况：1.抽象模型里的转换是否保持了不变式成立。这方面称为不变式保持证明。2.抽象模型的每个更准确的版本是否破坏了前一版中已经证明的质，这称为精化正确性证明。这些证明中的大多数都由一个叫做证明器的工具

动完成。
当然，任何方法都不是十全十美的，软件工程形式化方法也存在着一些问题以及推广上的障碍。管软件形式化方法总体上来说能够节省成本，但它在前期设计规格说明阶段投入较多，传统观念根深固的老板们可能这样做觉得不值得。对于软件工程师来说，习惯了传统的开发方式，而软件形式化方需要换一种思路去思考问题，可能前期不太适应，而且对于非数学专业的他们来说，对系统构造出清的数学关系模型可能稍显困难。另外，则是形式化方法中用到的自动化工具本身可能存在问题，无法证翻译器能够正确地将数学模型翻译成可执行代码，这样也就无法保证最终的正确性。
尽管如此，但从软件工程形式化目前已有的应用来看，它是相当成功的。最早的软件工程形式化法应用是在 IBM 的 CICS 系统中。当时 CICS 的复杂程度已经成了一个严重的问题，靠人力已经很难对这个系统进行维护，因此 IBM 想要重新设计这个系统。当时 CICS 的经理和牛津大学的一名研究形化方法的教授进行了沟通，并决定使用形式化方法来重新设计 CICS 系统，使用的方法叫做 notation。工程师们习惯了使用自然语言来描述 specification，一开始还担心用数学方来表示 specification 对他们来说很困难。但最后证明这并不是个问题，就连没有一些数学经验的程员也能够轻松运用。最终开发成本减少了百分之 9，用户报告的错误减少了 2.5 倍，而且都是轻微的误。巴黎地铁 14 号线和 Roissy 机场穿梭车的无人运行系统也是使用形式化方法开发的。它们使用的法叫做 B notation，比 Z notation 出现得更晚一些。这两个案例都没有执行单元测试，工程师没有改任何代码行，然而最终的运行结果仍然非常成功。
软件工程形式化方法的前景十分远大，一旦实现在工业中的广泛应用，将彻底颠覆整个软件行业大幅度提高生产效率和减少错误。非常有幸能在大学中了解到这么前沿的技术。