# 我对软件开发环境与工具的理解

十五次课，总共33.75小时、2025分钟、121500秒的学习，我对软件开发环境与工具有了深刻的认识。时间虽然不多，就十几次课，但就是这么十几次课，让我接触到了另外一个五彩的世界——工具的世界。在这说长也不长，说短也不短的时间内，几乎每节课都要接触好几个新的工具，这些新工具无一不是有用的工具。在后面几次的实验课上，自己又动手去使用了好几款对于自己来说崭新的工具，这简直太棒了！这个课让我意识到软件开发工具是多么重要且有意义的。虽然有些工具操作复杂，甚至还不开源、收取高额费用，但是它们真的是无可替代的软件产品，有了它们使得我在软件开发的每一步都得心应手。接下来，谈谈，对这个课的内容的理解：

1. 软件开发环境与工具概述
2. 方法与工具

这两个东西我们应该是相当的熟悉，方法的选择能直接决定一件事的成败，而工具的选择则能决定一件事的容易程度，甚至关乎到是否能完成任务。这两者相互对一件事造成影响，似乎两者之间并没有什么关联。以上是在未上这个课之前对方法与工具的理解。上了这个课之后，我发现完全不一样，它们在这个课上的关系是那么的不一样。

在软件工程学中，方法和工具是同一个问题的两个不同的方面，方法是工具的先导，工具是方法的实在体现，软件工程方法的研究成果只有最终实现为软件工具和系统，才能充分发挥软件工程方法在软件开发中的作用。

上面这段话，是书上的原话，很抽象吧，有点摸不到头脑，尤其是那句“方法和工具是同一个问题的两个方面”，刚开始接触到这句话，我也是这样的，表示很困惑，这说的是啥啊？经过这么久时间的学习，再加上自身的经历，现在的我真正地理解了这句话。“同一个问题”，“两个方面”，举个例子就能很清楚地认识到。面向对象，这是一种开发软件的方法，而面向对象的开发工具有很多——vs，eclipse，idea等等。这些工具和这个方法都在解决一个问题——开发软件，一个从理论上解决问题，一个在实际行动上解决问题，这就是“同一个问题”、“两个方面”。然后就是那句“方法是工具的先导，工具是方法的实在体现”，从这句话的字面意思，就能理解到，方法先于工具，工具是为了方法更好的实现。还是上面个面向对象的例子，在面向对象的发展历程中，是先有的方法，然后才出现的那些面向对象的工具，有了这些工具之后，越来越容易使用面向对象的方法来开发软件，进一步使面向对象走向更成熟、更加美好的道路。后面的话，就是一种陈述了，比较容易理解，就不做过多解释了。

1. 软件开发环境

对于软件开发环境的定义，书上有很多很多，我觉得最易理解，最能体现出它与软件开发工具关系的定义如下：“软件开发环境是相关的一组软件工具集合，它支持一定的软件开发方法或按照一定的软件开发模型组织而成”。这句话明显地揭示出了软件开发环境和软件开发工具之间的密切关系，在软件开发环境中，软件开发工具以一种综合、一致和整体连贯的形态来支持软件的开发。

软件开发环境有四层：宿主层、核心层、基本层、应用层。各层的详细介绍如下：

* 宿主层：它包括基本宿主硬件和基本宿主软件。
* 核心层：一般包括工具组、环境数据库和会话系统。
* 基本层：一般包括最少限度的一组工具，如编译工具、编译程序、调试程序、连接程序和装配程序等。这些工具都是由核心层来支援。
* 应用层：以特定的基本层为基础，但可包括一些补充工具，用于更好地支援各种应用软件的研制。

1. 软件开发环境的发展、要求与特性

软件开发工具的发展随同软件方法的发展，所以软件开发环境也就随着它们一起发展，历经三个阶段。

第一个阶段是20世纪70年代，此时，兴起的是结构化设计方法，在这个时代，发展出了结构化分析技术的一整套的相互衔接的SA-AD的方法学，为之后的技术发展奠定了基础。

第二个阶段是20世纪80年代中期与后期，主要涉及到的有实时系统设计方法和面向对象的分析和设计方法的发展，这些克服了结构化技术的缺点，进一步加强了对软件的开发。

第三个阶段在20世纪90年代，主要进行系统集成方法和集成系统的研究。这个时代产生的CASE工具，确保了软件的研发成功，而且还提高了软件质量，降低了投资成本和开发风险。

从这些历程中，可看出对软件开发环境的要求越来越高。主要的要求有：

* 集成，高度集成的一体化系统。
* 通用，高度的通用性。
* 适应性、灵活性。
* 性能高、易使用、高效。
* 有辅助开发到半自动开发再到自动开发的转变趋势。

这些要求也决定了，软件开发环境特性的某些必然性，如下：

* 可用性。响应快，便于用户使用。
* 自动化程度。让用户更加容易地开发应用。
* 公共性。覆盖各种类型用户、各种软件活动。
* 集成化程度。要求软件开发过程的高度集成。
* 适应性。符合用户要求。
* 价值。收益、成本之间的比值。

1. 软件开发工具

什么是软件开发工具呢？说简单点，就是一套程序系统，一套能帮你更简单地开发出其它软件系统的软件系统，软件开发工具需具备如下五个基本功能：

* 提供描述软件状况及其开发过程的概念模式，协助开发人员认识软件工作的环境与要求、管理软件开发的过程。
* 提供存储和管理有关信息的机制与手段。
* 帮助使用者编制、生成和修改各种文档，包括文字材料和各种表格、图像等等。
* 生成代码，即帮助使用者编写程序代码，使用户能在短时间内半自动生成所需要的代码段落，进行测试和修改。
* 对历史信息进行跨生命周期的管理，即将项目运行与版本更新的有关信息联合管理。

软件开发工具一般四大技术要素构成：

* 总控部分及人机交互界面。使用者与工具之间交流信息的桥梁。
* 信息库及其管理。用来存储软件开发过程中所涉及到的所有信息。
* 文档生成与代码生成。除了交互之外，另一重要的功能，帮助使用者半自动开发系统。
* 项目管理与版本管理。这一部分的关键是对历史信息的管理。

1. 软件开发环境与工具的分类
2. 软件开发环境的分类

分类方法有很多，现在按解决的问题、软件开发环境演变趋向与集成化程度进行分类。

1. 按解决的问题分类

* 程序设计环境：支持某种或某几种特定的变成语言。
* 系统合成环境：子系统集成为一个大系统，是基于组件的开发。
* 项目管理环境：主要解决由于软件产品的规模大、生存周期长、人们交往多而造成的问题。

1. 按软件开发环境的演变趋向分类

* 以语言为中心的环境：围绕着一门或几门语言展开的开发环境。
* 工具箱环境：一整套工具的集成，各种各样的工具供开发者人员使用。
* 基于方法的环境：支持某种或某些特定的软件开发方法。

1. 按集成化程度分类

* 第一代，直接建立在操作系统上的软件开发环境。
* 第二代，指的是拥有真正数据库的软件开发环境，而不是只靠文件系统。
* 第三代，建立在知识库系统（智能数据库系统）之上的，出现集成化工具集。

1. 软件开发工具分类

从不同角度，可有四种不同的分类方式：

1. 基于工作阶段划分

* 需求分析工具：ERwin、BPwin、StarUML等等
* 设计工具：数据库设计工具、HIPO图工具、PDL支持工具等等。
* 编码工具：众多DBMS和程序开发工具。
* 测试工具：SQASuite、Rational Robot、WINRUNNER等等。
* 运行维护工具：Nagios、Zabbix、Open-falcon等等。
* 项目管理工具：P3、ProjectScheduler、Microsoft‘s Project等等。

1. 基于集成化程度划分

集成化是一个相当难的是，目前，开发与应用集成化的软件开发工具，还处于研究与探索阶段，故用的，也应充分利用的还是各种专用型软件开发工具。

1. 基于软件、硬件的关系划分的工具

* 依赖于特定计算机或特定软件：例如基于Windows的软件开发工具，便依赖于Windows操作系统，在其它系统上是没办法使用的。
* 独立于硬件与其它软件的软件开发工具：例如分析工具和计划工具。

1. 基于应用领域划分的工具

* 可视化编程工具：eclipse、visual studio，Power Build等等。
* 数据库管理系统开发工具及语言：dBASE 5.0 for Windows，Oracle Ingress等等。
* MIS（管理信息系统）生成工具：雅奇MIS、王特绿MIS、QuickPro等等。
* 多媒体工具：Flash、3DMAX、Maya等等。

1. 软件开发工具对软件工程的支持

软件开发工具的存在使得软件开发，真正成为了一项工程，并使得这项工程变得更加容易且规范。而这些具体的辅助软件开发的工具就组成了CASE，这是我起初的认识，发现这其中有些地方是不太对的，下面就来详细说说这个支持软件工程的CASE。

1. CASE概述
2. 定义

CASE的英文全称为Computer-Aided Software Engineering，其中文名称叫做计算机辅助软件工程，其是一组工具和方法的集合（不仅仅只是工具），可以辅助软件开发生命周期各阶段进行软件开发。

1. CASE三大作用——根本上改变软件系统开发方式

* 一个具有快速响应、专用资源和早期查错功能的交互式开发环境。
* 对软件的开发和维护过程中的许多环节实现了自动化。
* 通过一个强有力的图形接口，实现直观的程序设计。

1. 集成化的CASE环境

首先理解一下集成这个词，想简单一点，这个集成其实就是，把相关的东西放在一起，然后相互达成某种协议，使得它们之间相互协作。官方解释就是：集成指一个环境下的工具的合作协议，包括数据格式、一致的用户界面、功能部件组合控制和过程模型。集成主要有四个方面的集成如下：

* 界面集成：达成界面的风格一致，目的是减轻用户认知负担，从而提高用户使用环境的效率和效果。
* 数据集成：以某种协议使不同数据是一个整体，目的是确认环境中的所有信息都必须作为一个整体数据能被各部分工具进行操作或转换。
* 控制集成：两个方面供给——一个工具服务提供服务给另外一个方面、使用——一个工具使用另外一个工具的服务，目的是为了能让工具共享功能。
* 过程集成：工具适应不同过程和方法的潜在能力，目的是工具更好地被使用。

理解了集成化，那么接下来，就需要去了解了解，怎么做到集成化？CASE集成化环境的最终目的在于支持与软件有关的所有过程和方法，而一个环境，是由许多工具和工具集组成，不同环境集成的策略是不同的，所以解决集成问题，首先必须明确环境，故以下给出环境的4个广泛分类：

* 以语言为中心的环境，用一个特定的语言全面支持编程。
* 面向结构的环境，通过提供的交互式机制全面地支持编程，使用户可以独立于特定的语言而直接对结构化对象进行加工。
* 基于方法的环境，由一组支持特定过程或方法的工具所组成。
* 工具箱式环境，它由一套通常独立于语言的工具所组成。

以上几种环境的集成，多采用传统的基于知识的CASE技术，或采用统一的用户界面，或采用共同的数据交换格式，来支持软件开发的方法和过程模型，目前基于概念模型和信息库的环境设计和集成方法，也渐渐开始流行了，并取得了一些可喜的成绩。

1. 软件开发工具举例

通过这门课程的学习，真接触了许多工具，现在结合自己经验，按照软件生命周期，来介绍一些软件。

1. 需求阶段

这个阶段我要介绍的是一个在线作图网站processOn，直接百度它就能找到这个网站。使用简单，且只需注册一次，便可终身免费使用所有服务，但是了，免费的版本智能存九张图片，这点虽然有影响，对于大的项目，但是依旧不能撼动它的地位。在这个网站上能画各种图，包括ER图、类图、用例图、数据流图等等，这些图其它很多软件或网站也能提供，所以并不是它的特色。它最重要的一个功能，也是我介绍它的理由，那就是协作，这个网站能协作完成一个设计，不仅能同一张图被多个人修改，也能多个人同时修改。

1. 设计阶段

设计分两种，界面设计和程序结构设计。界面设计我将要介绍的是墨刀，不要看名字，其实这是一个界面设计工具。它也是一款在线的工具，也支持多人协同设计，但是很遗憾的是他并不开源，若要使用它所有功能，就得去购买。现在说说介绍它的理由：第一，项目云端存放，这一点的好处在于你的修改能够及时保存，而且当你电脑出现问题时不会导致这部分文件的丢失，也方便移机操作；第二，可创建项目类型多、所包含的组件多，磨刀能创建的项目类型包括：手机、平板、网页、电视、手表等等，还可以自定义。而其中每个类型项目中所有的组件也是非常齐全——按钮、文本框、形状、滑条、图标等等；第三，组件的修饰容易、能修饰的样式也很多，颜色、字体、边框，大小、形状、图片均能轻松的修改；第四，能模拟简单的用户操作，比如点击按钮跳转，文本输入等等；第五，导出方便，易展示，它能将项目导成离线演示html，还能导成安卓安装包apk，在手机上演示，当然也能导成图片，静态演示。

1. 编码阶段

对于编码不得不介绍一款开源，且及其好用的软件——eclipse，这个软件都很熟悉的，不过，相信大多数人都只是用它来进行开发Java程序。它支持用户自定义风格，例如改变字的颜色、关键字的颜色、选中的颜色等扽等个。以上，都是蛮普通的，但我这里还要说的是，它不止能开发Java应用程序，这个软件最强大的地方在于它的插件，只要装了插件和相应开发所需要环境，那么它还能支持开发C/C++、python、PHP等等语言的开发。Eclipse中的插件也不止是在扩展其对语言的支持，还能干很多其它的事情，例如装了插件你能话UML图，甚至还能从代码直接导出UML图，装了插件你也能进行项目管理，装了插件你还能更好地发布项目……这些都足以证明它是有多么的强大！

1. 测试阶段

测试工具很多类别，有白盒测试工具、黑盒测试工具、功能测试工具、性能测试工具、单元测试工具、集成测试工具等等。由于，基本没对自己开发的软件或系统做过测试，没有足够经验，所以这里就简单介绍一个功能测试工具——WinRunner。WinRunner用于检验一个应用程序能否如期运行，是一个商业化的测试软件。它通过自动捕获、检测和模拟用户交互操作，WinRunner能识别出绝大多数的软件功能缺陷问题，从而保证那些跨越了多个功能点和数据库的应用程序在发布时尽量不出现功能性故障。它的特点是快速、批量完成功能测试点，消除人工测试所带来的理解错误。WinRunner的工作流程可大致分为六步：

* 识别应用程序GUI，WinRunner使用GUI　Spy识别各种GUI对象并存储到GUI　Map　File中。
* 建立测试脚本，建立之前先录制——记录用户操作，之后再形成的脚本中再手工加入需要的TSL。
* 对脚本除错，调试，利用Debug　Toolbar进行调试，排除脚本错误。
* 在新版应用程序中执行测试脚本，修改之后，直接用旧脚本即可。
* 分析测试结果，最重要的一步，通过分析找出程序的功能性缺陷。
* 回报缺陷，分析之后，按流程回报应用程序的各种缺陷，然后将这些缺陷发给指定的人，以修改和维护。

1. 维护阶段

这个阶段，就更加的不熟悉了，就简单介绍一款版本管理软件吧。先解释一下，这个版本控制软件为什么会在维护阶段起作用。当程序发布之后，进入维护阶段，会不间断对其进行内容上的更新，有时甚至功能上的更新，如果在某次的更新操作中出现了问题，又无法解决，那么首要就要保证用户正常使用，最简单、最有效的方式就是回退，退到上一版本，这就需要版本控制软件。接下来，我要介绍的软件是SVN，全称Apache　Subversion，这个软件开放源代码。

这个软件有两端，一是服务器端，而是客户端，标准的C/S架构开发。服务端主要就是用来存放版本数据，而客户端，就取出、修改、再提交数据。此外，SVN的速度是比较快的，支持回滚机制，就是当操作中出现问题，未完成却已修改某些数据的时候，会回到上一正确的状态。SVN的事务原则是“零或一”，要么事务全做，要么不做，这一点很利于并发处理，是SVN的一大优势。SVN的优点有以下几个：

* 管理方便，逻辑明确，符合一般人思维习惯。
* 易于管理，集中式服务器更能保证安全性。
* 代码一致性非常高。
* 适合开发人数不多的项目开发。

以上，就是我总结的全部，最后一句话概括，这个可让我认识到了很多软件工具，真的很有用，使得我以后的软件开发，更加有方向、更加容易。

参考文献：

基于UML的集成化软件开发环境的研究与实现

“软件开发环境与工具”课程教学研究与探讨

《软件开发工具与环境》课程的考核方案设计

软件工具与软件开发环境

《软件开发环境与工具教程》

面向协作的软件开发环境及其构造方法