软件体系结构

作者：安德烈·阿德里安

版本：23jul2005

目录

绪论...........................................................................................................................1

螺旋模型....................................................................................................................2

分级Decomposition................................................................................................2

自下而上development............................................................................................3

模型视图Controller.................................................................................................4

Callback..................................................................................................................6

消息序列Chart........................................................................................................7

状态转换diagram....................................................................................................8

系统架构Steps........................................................................................................9

螺旋Model...............................................................................................................9

高等教育层次分解和自底向上的发展..................................................................... ..9

模型视图Controller...................................................................................................9

消息序列图和状态转换图..........................................................................................9

结论...........................................................................................................................9

介绍

什么是软件架构？这些规则，模式，有助于帮助构建并且严格的测试软件。它从实践经验上升为抽象理论。软件体系结构始于尼克劳斯·沃思的“逐步求精”、 地瑞莫和克朗的“大规模软件”，甚至在此之前。它是建立在系统工程和科学方法之上并且由伽利略确立所的。

伽利略：你知道测量什么，然而你却不知如何测量。与实验想比是更重要的则是教育。

有关软件体系结构的信息都在互联网上。本文将尽可能的引用和使用。这样做的目的是汇集信息而不是重新表述的他人已有的措辞。

与往常一样，真理是人类无法轻易求取到的。但是，像在积分中创立微积分，我们可以创建一系列相似的。

我并不是想卖CASE工具给你 - 其他人已经做到了市场化。我想向你推出一些新的思想。你可以把这些新的思想用到你编写的任何一个程序上，无论使用哪个语言开发的。我想给你非常好的工具。具有计算机研究20年经验的我说：工具固然不错，但理解

更加重要。

使用大型开发工具不仅是软件架构的话题。它与统一建模语言具有相同的目标。但建模是一个广义的术语。一些学者甚至使用13种不同的UML类图型来表示同一个的话题。

螺旋模型

http://en.wikipedia.org/wiki/Spiral\_model

螺旋模型是一个正在发展的模式，它结合了两种设计元素。原型模型阶段，它同时结合了自上而下和自下而上的优点。螺旋模型是巴里伯姆从1986年在他软件的螺旋模型文章中所定义的。第一模型所讨论的不是发展和加强的这种模式，它们被迭代所取代。因而第一模型解释了为什么把迭代作为第一模型。

层次分解

大多数读者认为70年代由Yourdon的和康斯坦丁所推广结构化设计（SD）是从一个过时的设计方法。但是，为了促进SD发展，将它与面向对象结合在一起很好很强大的想法。层次分解也是HIPO（分层输入处理输出）的一部分- 甚至更早于SD。

一个重要的事实是：层次分解是非常有价值的，但HIPO和SD今天仍然具有价值。现在的OO PARADIGMA也成了其中的一部分。

一个大的系统必须分为管理子系统（或组件或模块或对象）在多个层中。层次分解使程序变得简单，但这并不是主要的目的。首先，系统分为顶层子系统。然后每个顶层系统分为二级子系统。该系统是以一个分层的编号。一个名叫2子系统。是一个一级子系统，子系统命名2.3.1。

上层子系统没有任何对象。最低层次的子系统，应设计为对象。没有层次的系统分解更难成为“物化”系统。

通常，系统在设计时考虑到硬件设备。例如移动电话（用户终端）和基站在蜂窝两个子系统。

层次分解允许进入子系统。由于设计深度的不同，在不同的时间上设计相对于整个系统会显得松散。上层子系统应包含所有包括较低层次的子系统的细节。这对于输入输出子系统是重要的。通常在分层分解的输入和输出的上层子系统得到更新（增强），由于新的需求，设计者被要求设计低层次的子系统。

自下而上发展

自下而上的发展（也被称为敏捷软件开发的或极端的编程）是用于对系统算法的螺旋模型。自下而上的。开发放式与自上而下的层次分解的相反。你可以采用分层分解创建你的程序，但你也应该使用自下而上的发展来作为一个开发周期。

根据不同的系统，算法是最简单、最艰难的。你可以很容易使用烹调书式的算法。如果你要开发新的算法，那么就发展一个迭代算法：自底向上的方法。建议你实现要先从你的工程分解的重要子系统，那么你的首要这部分系统的完整的系统的几个周期。这是很重要的请记住。部分系统做不到抛弃式原型，在迭代自底向上的时候，你必须使用相同的编程语言和相同的系统开发环境。目标是建立模拟UPS之上，程序从早期检查和早期用户反馈得到修正。

因为只有几周甚至几天的周期，在用户更多地参与并之前。而不是等待2年的软件团队进行首次发布，每个开发周期用户都要知道其进展。因为软件开发人员，测试经理会在最后一个周期得到新的计划。开发周期必须适应这种新的情况。最后一个周期工作通常的子层次分解。

这关系到所有子系统水平到用户的计划的改变，这是顶层的系统规划。

从团队建设的角度出发，就把所有的参与者（用户，开发人员，测试仪）为一个团队。但是由于这种方法具有社会学等缺点：非团队成员可以妥协一个拥有自己团队观的系统，这是为达到系统的要求同时这也是由所有用户所达到的期望，而不仅是在软件开发团队的用户。

为了使审计不是一场闹剧，因而在每个周期要涉及用户和审计。测试参与者需要提供必要的知识，以打消公众的异质性，形成一个整体的团队。

[http://www.dmreview.com/article\_sub.cfm？articleId=1025869](http://www.dmreview.com/article_sub.cfm？时1025869)

你可能会遇到一些关于管理方面的挑战，你在软件开发的周期是，有些事情是需要注意的：

·由于一些问题延缓软件开发的周期。不要延迟开发周期仅仅是发现某些问题。把问题推到下一个开发的周期或暂时忽略它。我们的目标是创建完整的开发周期，我们主要得知我们的软件开发的发展状况。如果因为你的迟疑，整个过程的完成将使无法预知的。

·改变太多。根据你的介绍变化，特别是在开发周期（它是在稳定的周期假设）。数据模型的变化就是一个很好的例子。主要图式返工只是结果相同的工作。一遍又一遍的进行。打造“幸福”的周期，开发人员可以专注开发本身，不断地在前进，即使数据模型有一些缺陷。只要不改变我们的目标。任何开发任务都是未知的。只要仔细思考，即使有些问题暂时不明白，在后期开发中逐渐就会有头绪的。

模型-视图-控制器

http://ootips.org/mvc-pattern.html

MVC模式是一种方法使应用程序，甚至只是一个应用程序的界面，分为三个部分：模型，视图和控制器。MVC最初开发分为传统的输入，处理，输出的。开发者进入到GUI的开发模式：

输入——处理——> >输出

——>——>视图控制器模型

<http://www.cs.indiana.edu/~cbaray/projects/mvc.html>

MVC模式是如此的简单，的确令人难以置信。从本质上讲，模式迫使人们考虑的如何把现实问题转化为这三个模块中的应用—

·模型：应用的核心。它的作用是让应用程序呈现出的状态和数据。当模型发生明显的变化，它可以随时更新所有的数据。

·控制器：在操作系统中将用户界面呈现给用户。

·视图：在用户界面显示的信息。关于使用模型的用户。任何对象使用的信息模式只需要一个。

注册视图与模型。

http://wiki.tcl.tk/6225

模型-视图-控制器[ 1 ]（通常简称MVC）是一种工作方式和图形用户界面，它提供访问复杂的数据的服务，它似乎做得很好。你通过分离的代码来实现模型（基本数据，说，可更新列表或只读树）从（你的数据看起来时画的在屏幕上）和控制器（你的模型如何响应更新的视图）

虽然经常把视图和控制器连接在一起。

TK的一些特点，这样操作；特别是变量选择的入口，- textvariable选项标签，按钮等，和非listvariable选项列表。

设计你的界面，以及休息是明智的。跟踪命令（特别是[跟踪变量]）实现触发更新是非常有用的，看来当底层模型的变化。bindtags命令是有用的。

将一个控件与一个或多个控制器以外，TK的标准

提供；偶尔，你也需要借助于重载控件的命令。对于

工作的例子，看网页题为，“文本变量文本控件”。在那

例如，控件命令超载通知模型当用户

试图改变文本内容。跟踪通知视图的模型时，

更新。一个绑定的标签也可以用来使一切变得清洁起来正确时

窗口被销毁。

软件体系结构安德烈阿德里安4页9

http://wiki.tcl.tk/9219

如果0 {李察suchenwirth 2003-06-27 -另一个试图教编程

孩子，这个小玩意儿（在代码和屏幕长度）做

在你进入的因素两项飞乘。

所有的通信是通过textvariables和痕迹。我不确定这是否

已经作为一个小的模型/视图/控制器实例：

变量A，B，C模型

标签。C是在结果期，C

条目。，。B是用户改变模型控制器

但无论如何，这是一个很小的例子”（也为花式的组件名称-我没有

相信他们能。或。=以前我试过……）}

程序主要{ } {

全球A B C

进入。一个宽5 textvariable一

标签。\* \*

入口。B宽度5 textvariable B

标签文本=。=

标签。C宽度10 textvariable C

eval包[ WINFO孩子。]侧左

{一} {每个因子B

跟踪变量$因子重新计算W { }

}

}

如果在这个过程中，试图重新计算0 {（可能条目是否已经失败

清除），我们忽略的三个观点，一个痕迹会自动加入我们

知道我们想要什么：}

proc { - } { -重新计算

全球A B C

抓住{集C [ expr { $a b }】}

}

#最后，走吧（这是一个很好的方式开始和结束过程主要，调用它。）

主要

软件体系结构安德烈阿德里安5页9

回调

回调的话题（事件处理程序，软件中断）GUI和真实是很重要的—

时间规划。在MVC控制器模型之间的连接是一个回调，

模型和视图之间的连接可以是一个普通的函数调用。

回调函数是用来连接外部世界与程序。的

事件是异步。该计划预计这些事件，但事件顺序

确切的时间或事件发生之前不知道。

第一个动作是注册回调函数（设置事件处理程序）。运行时

程序的环境存储回调的细节。如果条件（触发）

该回调了，回调被称为。对于guiprograms运行时环境

是典型的LIBS像XT（X窗口系统工具包）或Tcl/Tk运行

lib。RT POSIX系统调用程序select()通常回调注册

和通话功能。操作系统的运行环境。

GUI和RT编程可以看作是写作扩展的GUI或RT运行。

扩展的回调。