第 19 章：数据设计　　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据设计的步骤和原则

第19章　数据设计

数据设计是软件设计中最重要的活动，而数据结构对程序结构和过程复杂性的影响使得数据设计会对软件质量产生不容忽视的影响。

19.1　数据设计的步骤和原则

数据设计并不是一拍脑袋就能想出来的，我们必须遵循相应的规则和[illegible]。

上一节　本书简介　下一节

第 19 章：数据设计　　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据设计步骤

软件工程专家P.S.Pressman把数据设计的过程概括成以下两步：

为在需求分析阶段所确定的数据对象选择逻辑表示，需要对不同结构进行算法分析，以便选择一个最有效的设计方案。或者说确定一种结构，设计对于这种逻辑数据结构的一组操作，以实现各种所期望的运算。这里，选择逻辑表示的过程就是要确定软件的逻辑数据结构的过程。

确定对逻辑数据结构所必需的那些操作的程序模块（软件包），以便限制或确定各个数据设计决策的影响范围。

无论采取什么样的设计方法，如果数据设计得好，往往能产生很好的软件体系结构，具有很强的模块独立性和较低的程序复杂性。

上一节　本书简介　下一节

第 19 章：数据设计　　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据设计原则

Pressman提出了一组原则，用来定义和设计数据。实际上，在进行需求分析时往往就开始了数据设计。需要注意，需求分析与设计往往是交叉进行的。用于数据规格说明的原则如下：

用于软件的系统化方法也适用于数据。在导出、评审和定义软件的需求和软件系统结构时，必须定义和评审其中所用到的数据流、数据对象及数据结构的表示。应当考虑几种不同的数据组织方案，还应当分析数据设计给软件设计带来的影响。

要确定所有的数据结构和在每种数据结构上施加的操作。设计有效的数据结构，必须考虑要对

该数据结构进行的各种操作。如果定义了一个有多个不同类型数据元素的复杂数据结构，它会涉及到软件中若干个功能的实现处理。再考虑对这种数据结构进行的操作时，可以为它定义一个抽象数据类型，以便在今后的软件设计中使用它。抽象数据类型的规格说明可以大大简化软件设计。

应当建立一个数据词典并用它来定义数据和软件的实现。数据词典清楚地说明了各个数据之间的关系和对数据结构内各个数据元素的约束。如果有一个类似于数据词典的数据规格说明，一些必须涉及到数据之间某种具体关系的算法就很容易确定。

低层数据设计的决策应推迟到设计过程的后期进行。可以将逐步细化的方法用于数据设计。在进行需求分析时确定的总体数据组织，应在概要设计阶段加以细化，而在详细设计阶段才规定具体的细节。这种自顶向下进行数据设计的方法首先设计和分析主要的结构特性，具有与自顶向下进行软件设计的方法相类似的优点。

数据结构的表示限定于那些必须直接使用该数据结构内数据的模块才能知道。此原则就是信息隐蔽和与此相关的耦合性原则，并把数据对象的逻辑表示与物理表示分开。

应当建立一个存放有效数据结构及相关操作的库。数据结构和它的相关操作可以看做是软件设计的资源。数据结构应当设计成为可复用的。建立一个存有各种可复用的数据结构模型的部件库，可以减少数据的规格说明和设计两方面的工作量。

软件设计和程序设计语言应当支持抽象数据类型的定义和实现。如果没有直接定义某种复杂数据结构的手段，这种结构的设计和实现往往是很困难的。

上一节　本书简介　下一节

作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据字典

一个好的软件设计，数据的设计必然是合乎逻辑的、最优化的、最符合现实的。而要做好数据设计，数据字典是我们必须要搞清楚的一个概念。

数据处理流程图反映系统的全貌，它将各种信息流之间错综复杂的联系有机地统一在一张图上。数据流图元素如下：

图形元素的名字：某一词条的名字。

别名或编号。

分类：加工、数据流、数据文件、数据元素、数据源、汇点等。

描述：功能、特点。

定义：该词条的名称、数据结构等。

位置：数据流的来源，去处。

加工框的编号、输入、输出。

数据元素在哪个数据结构中等。

但是为了更准确地反映数据流图上元素的具体定义，应对数据流图中的3个元素：数据流、数据存储和数据处理，进一步进行描述。

综上所述，数据字典（Data Dictionary,DD）就是用来定义数据流图中出现的所有名字（数据

流、数据存储和数据处理），是对数据流图必要的补充。数据流图和数据字典是需求规格说明书的主要组成部分，只有同时有数据流图和数据字典才算完整地描述了一个系统。

第 19 章：数据设计　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据字典设计

数据字典的设计包括：数据流设计、数据元素字典设计、数据处理字典设计、数据结构字典设计和数据存储设计。这些设计涵盖了数据的采集、范围的确定，等等。

第 19 章：数据设计　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据流设计

一般，数据流设计需要考虑以下几点要素。

数据流编号：数据流编号是数据流的唯一标识。一般可以用字母和数字组成的字符串表示，而且需要遵守一定的编码规则。

数据流名称：数据流名称一般遵照用户的习惯，采用容易理解、有一定含义的名称。

简述：数据流的简要说明一般采用用户容易理解的、通俗的自然语言（汉语、英语等）描述。

数据流来源：发出数据流的数据处理名或外部实体名。

数据流去向：接受数据流的数据处理名或外部实体名。

数据流的组成：数据流的组成元素集合，它是数据流最核心的内容。

流通量：表示数据流在单位时间内的流量，是一个可选项，对将来的数据库设计或变量设计等有意义。

峰值：表示数据流在单位时间内的流量，可以写也可以不写，对将来的数据库设计或变量设计等有意义。

我们以"订书单"为例进行数据设计。它的来源是外部实体"图书馆采购部",接受数据流的是数据加工"采购验收".数据流"订书单"包括：订单编号、日期、书商、发票内图书种数、发票内图书册数、发票内图书金额、实际图书种数、实际图书册数、实际金额、退货图书种数、退货图书册数、退货金额、退款凭证、付款凭证、付款方式、付款人、货币类型、兑换率、修改等数据元素。这些是将来设计数据变量或数据库的重要依据。

确定各项后，填表如下，见表19-1.

表19-1　数据流设计

　作者：希赛教育软考学院　来源：希赛网　2014年01月27日

数据元素字典设计

数据元素是不能再进一步分解的数据组成的基本项。它直接反映事物的某一特征。当然，不可分解只是一个相对的概念。例如，一本书是一个数据元素，但是在图书馆处理时，还要分为书名、索书号、单价、分类号、著者号、出版社等几个数据项。

设计数据字典的基本内容有：数据元素编号、名称及其含义；数据类型和长度；合理取值。

一般按表19-2填写。

表19-2　数据字典

数据项编号		数据项名	
数据项别名		数据类型	
长度		取值范围	
取值含义		与其他数据项的逻辑关系	
数据项含义说明			

例如，以"文献分类号"为例，填写表19-3.

表19-3　文献分类号

数据项编号	文献 326	数据项名	文献分类号
数据项别名	文献资料	数据类型	字符类型
长度	双字节	取值范围	A～Z
取值含义	分别对应： A 马列主义、毛泽东思想 B 哲学 C 社会科学总论 D 政治、法律 E 军事 F 经济 G 文化、科学、教育、体育 H 语言、文字 I 文学 J 艺术 K 历史、地理 N 自然科学总论 O 数理科学、化学 P 天文学 Q 生物学 R 医药、卫生 S 农业科学 T 工业技术 U 交通运输 V 航空、航天 X 环境科学、劳动保护（安全科学） Z 综合性图书	与其他数据项的逻辑关系	分类图书
数据项含义说明	书的分类方法		

上一节　本书简介　下一节

数据处理字典设计

光有数据元素字典是不能满足软件设计的要求的，我们仍然不清楚数据处理的因果关系。顾名思义，数据处理字典就是让我们更清楚地了解数据流图中的数据处理会生成什么样的数据。

设计数据处理字典内容包括：数据过程编号、输入/输出。填写格式见表19-4.

表19-4 数据处理字典内容

处理过程编号		处理过程名称	
输入		输出	
处理说明			

例如，以"处理订书单"为例填表19-5.

表19-5 处理订书单

处理过程编号	DS865	处理过程名称	处理订书单
输入	定单	输出	1 合格订单，去处：处理逻辑“确定订单” 2 不合格订单，去处：外部实体“图书馆采购部”
处理说明			

上一节　本书简介　下一节

数据结构字典设计

数据结构是单个数据元素之间逻辑关系的表示，因为信息结构必将会影响最终的过程设计。数据结构的设计决定了数据的组织、访问方法、关联程度和可替换的处理方法。

1.数据结构字典格式

一般而言，数据结构字典可以由若干个数据项组成，也可以由若干个数据结构组成，或由若干个数据项和数据结构混合组成。填写格式见表19-6.

表19-6 数据结构字典

数据结构编号		数据结构名称	
含义说明			
组成			

2.数据结构符号约定

在数据设计中，数据必须以清晰、准确、无二义的方式来描述数据结构。其中，巴科斯-瑙尔范式（BNF BACKUS-NAUR FORM）是一种严格的描述数据结构方式。

表19-7给出了数据词典的定义式中常用的一些符号。

表19-7 数据词典定义中的符号

数 据 结 构	记　　号	意　　义
顺序	+	和
选择	[]	或
重复	{}n () **	n次重复 可选择的数据 限定的注释

上一节　本书简介　下一节

数据存储设计

数据存储设计是设计数据停留或保存的地方，也是确定数据流的来源和去向的过程。

一般按表19-8填写数据的存储过程。

表19-8 数据的存储过程

数据存储编号		数据存储名称	
流入数据流		流出数据流	
数据量		存取方式	
组成			
说明			

上一节　本书简介　下一节

设计数据的逻辑描述

前面所讲到的数据元素字典仅仅说明了如何描述数据流图中的数据格式，而对数据的逻辑性未加以详细的说明。下面我们进一步分析数据的逻辑关系。

一般来说，对数据进行逻辑描述时可把数据分为动态数据和静态数据。

静态数据是在运行过程中主要作为参考的数据，它们在很长的一段时间内不会变化，一般不随运行而改变。

动态数据包括所有在运行中要发生变化的数据及在运行中要输入、输出的数据。进行描述时应把各数据元素逻辑地分成若干组，例如函数、源数据或对于其应用更为恰当的逻辑分组。给出每一数据元素的名称（包括缩写和代码）、定义（或物理意义）、度量单位、值域、格式和类型等有关信息。

静态数据：

列出所有作为控制或参考用的静态数据元素。

动态输入数据：列出动态输入数据元素（包括在常规运行中或联机操作中要改变的数据）。

动态输出数据：列出动态输出数据元素（包括在常规运行中或联机操作中要改变的数据）。

内部生成数据：列出向用户或开发单位中的维护调试人员提供的内部生成数据。

数据约定：说明对数据要求的制约。逐条列出对进一步扩充或使用方面的考虑而提出的对数据要求的限制（容量、文卷、记录和数据元素的个数的最大值）。对于在设计和开发中确定是临界性的限制更要明确指出。

上一节　本书简介　下一节

数据设计的逻辑分析工具

在数据设计时，光用数据字典的数据说明有时候不直观，很多定义含糊不清，这往往会让程序员在阅读时感到无所适从，所以我们常常借助一些逻辑表达工具。常用的逻辑表达工具有结构化语言、判定表、判定树等。

上一节　本书简介　下一节

结构化语言

结构化语言是介于自然语言和形式语言之间的一种半形式语言，是在自然语言基础之上加了一些限度，使用有限的词汇和有限的语句来描述加工逻辑。

结构化语言的结构分为外层和内层。

外层有严格的语法，用一组符号IF-ELSE、WHILE、REPEAT-UNTIL、CASE-ON等描述控制结

构，采用顺序、选择和重复3种基本结构。

内层无严格的语法，采用祈使语句描述处理。其中用到的名词一般都是数据字典中定义了的。

下面是图书管理系统中"检查还书"的例子，用户要求的自然语言含义为：

如果学生所借的所有图书未超过90天，那么可以借阅图书；否则按每本书超期一天罚款1角钱处理，并标记为不可以借书；如果本科生所借图书未达到5本，研究生所借图书未达到15本，教师所借图书未达到20本，则可以借阅，否则不可以借阅。借书数目不限制。

上述需求用结构化语言表示如下：

```
WHILE（用户借书表未到达最后一条记录）{
IF 某一条借书记录未超过90天
{
指针移动到该用户借书表的下一条借书记录；
BREAK;
}
ELSE
{
按每本书超期一天罚款1角钱登记到罚款单；
标记该读者不可以借书；
指针转移到该用户借书表的下一条记录；
}
}
SWITCH（读者种类）
{
CASE"本科生"
判断用户是否达到借书上限，并标记是否可以借书；
CASE"研究生"
判断用户是否达到借书上限，并标记是否可以借书；
CASE"教师"
判断用户是否达到借书上限，并标记是否可以借书；
}
IF（该用户可以借书）
借书
ELSE
不允许借书
END
```

判定表

判定表由4个部分构成，见表19-9.

表19-9 判定表的构成

条件茬	条件项
动作茬	动作项

条件茬（Condition Stub）：左上部分。列出了各种可能的条件。通常判定表中条件茬所列条件的先后次序无关。

动作茬（Action Stub）：左下部分。列出了可能采用的动作。

条件项（Condition Entry）：右上部分。针对各种条件给出多组条件取值的组合。

动作项（Action Entry）：右下部分。指出在条件项各组取值的组合情况下应采取的动作。

下面同样以借书（仅包含本科生和老师）为例子，说明判定表的构成。见表19-10.

表19-10 判定表的例子

		1	2	3	4	5	6	7	8
条件	借书本数	<5	⩾5	<5	⩾5	<20	⩾20	<20	⩾20
	读者种类	本科	本科	本科	本科	老师	老师	老师	老师
	是否有超期图书	是	是	否	否	是	是	否	否
行动	开罚款单	√	√			√	√		
	允许借书			√				√	

一般而言，绘制判断表遵循以下6个步骤：

①提取问题中的条件。

②确定条件的取值，并指定各取值的代表符。

③计算所有条件取值的组合数。

④提取可能采取的动作。

⑤制作判断表。

⑥完善判断表（主要进行情况合并）。

算法中包含有多重嵌套的条件选择，用程序流程图、盒图、PAD图及过程设计语言（PDL）都不易清楚地描述时，此时用判定表则比较容易描述。

判定树

判定树本质上同判定表是一样的，当用户不易接受判定表这种描述方式时，我们可以用判定树的形式。判定树是一种图形表示，更易被用户理解。

下面同样以借书（仅包含本科生和老师）为例子，说明判定树的构成。如图19-1所示。

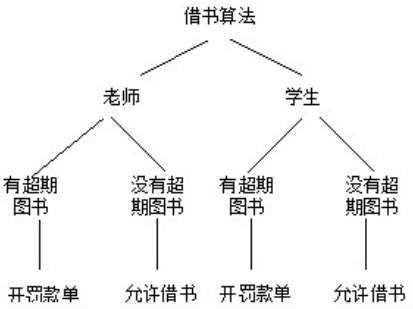

图19-1 判定树的例子

上一节 本书简介 下一节

 作者：希赛教育软考学院 来源：希赛网 2014年01月27日

数据保护性设计

作为软件设计的一部分，设计的数据还必须有一定的容错、纠错、容灾的功能，所以在数据设计中加入保护性设计是必要的。

一般数据保护性设计分为以下3种。

防卫性设计：即在软件设计中就插入自动检错、报错和纠错的功能。

一致性设计：在软件设计过程中，保证软件运行过程中所使用的数据的类型和取值范围不变，并且在并发处理过程中使用封锁和解除封锁机制保持数据不被破坏。

冗余性设计：针对同一问题，由两个开发者采用不同的程序设计风格不同的算法设计软件，当两者运行结果之差不在允许范围内时，利用检错系统予以纠正，或使用表决技术决定一个正确结果。

上一节 本书简介 下一节

 作者：希赛教育软考学院 来源：希赛网 2014年01月27日

软件测试概述

第20章 测试用例设计

在软件工程过程中，软件测试占据着十分重要的地位，按照经典软件工程理论，软件测试占整个开发过程时间的40%.而软件测试是否充分，是否能达到预定目标，测试用例的设计举足轻重。

20.1 软件测试概述

1.软件测试的定义

在引入软件测试的概念之前，先阐述与软件测试密切相关的几个术语。

错误：人类会犯错误，人们在编写代码时会出现过错，这种过错就是我们通常所说的bug,错误很可能扩散，需求错误在设计期间有可能被放大，在编写代码时还会进一步扩大。

缺陷：是错误的结果，或者说缺陷是错误的表现，而表现是表示的模式，例如叙述性文字、数据流程图、层次结构图、源代码等。缺陷分为过错缺陷和遗漏缺陷，如果把某些信息输入到不正确的表示中，就是过错缺陷；如果没有输入正确信息，就是遗漏缺陷；遗漏缺陷比过错缺陷更难检测和解决。

失效：当缺陷执行时会发生失效。失效只出现在可执行的表现中，通常是源代码，或更精确地说是被装载的目标代码；失效通常只与过错缺陷有关。

事故：当出现失效时，可能会也可能不会呈现给用户（或客户或测试人员）。事故说明出现了与失效类似的情况，警告用户注意所出现的失效。

软件测试需要处理与之相关的错误、缺陷、失效和事故。软件测试是为了发现错误而执行程序的过程；软件测试是根据程序开发阶段的规格说明及程序内部结构而精心设计的一批测试用例（输入数据及其预期结果的集合），并利用这些测试用例去运行程序，以发现程序错误的过程。

2.软件测试的目的

从软件开发者的角度出发，希望软件测试成为表明软件产品中不存在错误的过程，验证该软件已正确地实现了用户的要求，确立人们对软件质量的信心。

从用户的角度出发，普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否可接受该产品。

Meyers软件测试目的：

测试是执行程序的过程，目的在于发现错误。

一个好的测试用例在于能发现至今未发现的错误。

一个成功的测试是发现了至今未发现的错误的测试。

换言之，测试的目的是：

想以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。如果我们成功地实施了测试，我们就能够发现软件中的错误。

测试的附带收获是，它能够证明软件的功能和性能与需求说明相符合。

测试过程中收集到的测试结果数据为可靠性分析提供了依据。

测试不能表明软件中不存在错误，它只能说明软件中存在错误。

3.软件测试的原则

应当把"尽早地和不断地进行软件测试"作为软件开发者的座右铭。测试用例应当由测试输入数据和对应的预期输出结果这两部分组成。程序员应避免检查自己的程序。在设计测试用例时，应包括合理的输入条件和不合理的输入条件。充分注意测试中的群集现象。经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目成正比。严格执行测试计划，排除测试的随意性；应当对每一个测试结果做全面检查；妥善保存测试计划、测试用例、出错统计和最终分析报告，为软件维护提供方便。

软件测试并不等于程序测试。软件测试应贯穿于软件定义与开发的整个期间。需求分析、概要设计、详细设计及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计规格说明、详细设计规格说明及源程序，都应成为软件测试的对象。

4.测试用例设计

测试用例是为特定目标开发的测试输入、执行条件和预期结果的集合。

测试用例应该包含如下信息：测试用例ID、目的、前提、输入、预期输出、执行结果、执行历

史，以及日期、版本、执行人等信息。

输入包括两种类型：前提（在测试用例执行之前已经存在的环境）和由某种测试方法所标识的实际输入。预期输出也有两种类型：后果和实际输出。通常需要判断被执行的一组测试用例的输出是否可接受。测试活动要建立必要的前提条件，提供测试用例输入，观察输出，然后将这些输出与预期输出进行比较，以确定该测试是否通过。其他的测试用例信息主要支持测试管理。测试用例应该拥有一个标识和一个原因。记录测试用例的执行历史也是很有用的，包括测试用例是什么时候由谁运行的，每次执行的通过/失败记录，测试用例测试的软件版本等。测试用例需要被开发、评审、使用、管理和保存。

软件测试的中心是测试用例设计和执行。测试过程可以细分为独立的步骤：测试计划、测试用例开发、运行测试用例及评估测试结果。

设计测试用例通常有两种常用的测试方法：黑盒测试和白盒测试。

5.黑盒测试和白盒测试

黑盒测试把测试对象看做一个空盒子，不考虑程序的内部逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明，又称为功能测试或数据驱动测试。

白盒测试把测试对象看做一个透明的盒子，它允许测试人员利用程序内部的逻辑结构和有关信息，设计或选择测试用例，对程序所有逻辑路径进行测试，通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致，又称为结构测试或逻辑驱动测试。

6.逻辑覆盖

逻辑覆盖是以程序内部的逻辑结构为基础的设计用例的技术。它属于白盒测试，包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖等。