软件工程概述

软件

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。

软件的特点

1、 Logical rather than physical（软件是一种逻辑实体，而非具体的物理实体）

2、Developed or Engineered, not manufactured in the classical sense（在研制、开发活动中被创造出来，但不能按传统的生产含义加以理解）

3、Doesn't wear out（在软件的运行和使用期间，没有磨损、老化问题）

软件分类

按功能划分

-系统软件

-支撑软件

-应用软件

按规模划分——微型、小型、中型、大型、甚大型、极大型

类别	参加人员数	研制期限	产品规模（源程序行数）
微型	1	1-4 周	0.5k

小型	1	1-6 月	1k-2k
中型	2-5	1-2 年	5k-50k
大型	5-20	2-3 年	50k-100k
甚大型	100-1000	4-5 年	1M
极大型	2000-5000	5-10 年	1M-10M

按工作方式划分

-实时处理软件

-分时软件

-交互式软件

-批处理软件

按软件服务对象的范围划分

-项目软件（定制软件）

-产品软件

软件的发展

软件的发展可以分为这样三个阶段：

1、 程序设计阶段，约为 50 至 60 年代

2、 程序系统阶段，约为 60 至 70 年代

3、软件工程阶段，约为 70 年代以后

时期 特点	程序设计	程序系统	软件工程
软件所指	程序	程序及说明书	程序、文档及数据
主要程序设计语言	汇编及机器语言	高级语言	软件语言*
软件工作范围	程序编写	包括设计和测试	软件生存期
需求者	程序设计者本人	少数用户	市场用户
开发软件的组织	个人	开发小组	开发小组及大、中型软件开发机构
软件规模	小型	中、小型	大、中、小型
决定	个人程序设	小组技术水平	管理水平

质量的因素	计技术		
开发技术和手段	子程序、程序库	结构化程序设计	数据库，开发工具，开发环境，工程化开发方法，标准和规范，网络和分布式开发，面向对象技术，软件过程与过程改进
维护责任者	程序设计者	开发小组	专职维护人员
硬件特征	价格高，存储容量小，工作可靠性差	降价，速度、存储容量及工作可靠性有明显提高	向超高速、大容量、微型化及网络化方向发展
软件特征	完全不受重视	软件技术的发展不能满足需求，出现软件危机	开发技术有进步，但未获突破性进展，价格高，未完全摆脱软件危机

软件危机

*软件需求增长得不到满足；

*软件生产成本高，价格昂贵；

*软件生产进度无法控制；

*软件需求定义不准确，易偏离用户需求；

*软件质量不易保证；

*软件可维护性差等等

软件工程

采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，这就是软件工程。

Boehm 曾为软件工程下了定义：运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必需的相关文件资料。

Fritz Bauer: The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.软件工程是建立和使用一系列完善的工程化原则以便经济地获得能够在实际机器上有效运行的可靠软件。

1983 年，IEEE（Institute of Electrical & Electronic Engineers，电气和电子工程师学会）做出的定义是“软件工程是开发、运行、维护和修复软件的系统方法。”

它的核心内容是“以工程化的方式组织软件的开发”，其中涉及软件计划、开发和维护各个阶段。

软件工程包括三个要素：方法、工具和过程。

1） 软件工程方法为软件开发提供了“如何做”的技术。

2） 软件工具为软件工程方法提供自动的或半自动的软件支撑环境。

3） 软件工程过程指将软件工程的方法和工具综合起来，以达到合理、及时地进行计算机软件开发的目的。

软件工程过程（Software Engineering Process）

软件工程过程是指为了获得软件产品，在软件工具支持下由软件工程师采用软件工程方法完成的一系列软件工程活动。

软件工程的七条原理

著名的软件工程专家 B.W.Boehm 于 1983 年在一篇论文中提出了软件工程的七条基本原理。他认为这七条原理是确保软件产品质量和开发效率的原理的最小集合。

1） 用分阶段的生命周期计划严格管理

2） 坚持进行阶段评审

3） 实行严格的产品控制

4） 采用现代程序设计技术

5） 结果应能清楚地审查

6） 开发小组的人员应少而精

7） 承认不断改进软件工程实践的必要性

软件工程的目标

组织实施软件工程项目是为了获得项目的成功，即达到以下几个主要的目标：

1）付出较低的开发成本

2）达到预期的软件功能

3）取得较好的软件性能

4）使开发的软件易于移植

5）需要较低的维护费用

6）能按时完成开发工作，及时交付使用

概括地说，基本目标，四个字：优质、高效。即质量高、效率高。

软件工程的原则

软件工程基本目标适用于所有软件工程项目。为达到这些目标，在软件开发过程中必须遵循下列软件工程原则。

1） 抽象

2） 信息隐蔽

3） 模块化

4） 局部化

5） 确定性

6） 一致性

7） 完备性

8） 可验证性

使用一致性、完备性和可验证性的原则可以帮助开发者设计一个正确的系统。

软件生命周期（life cycle）

同其它事物一样，软件也有孕育、诞生、成长、成熟、衰亡的生存过程，称为软件的生命周期。包括六阶段内容：

1、 制定计划

2、 需求分析

3、 软件设计

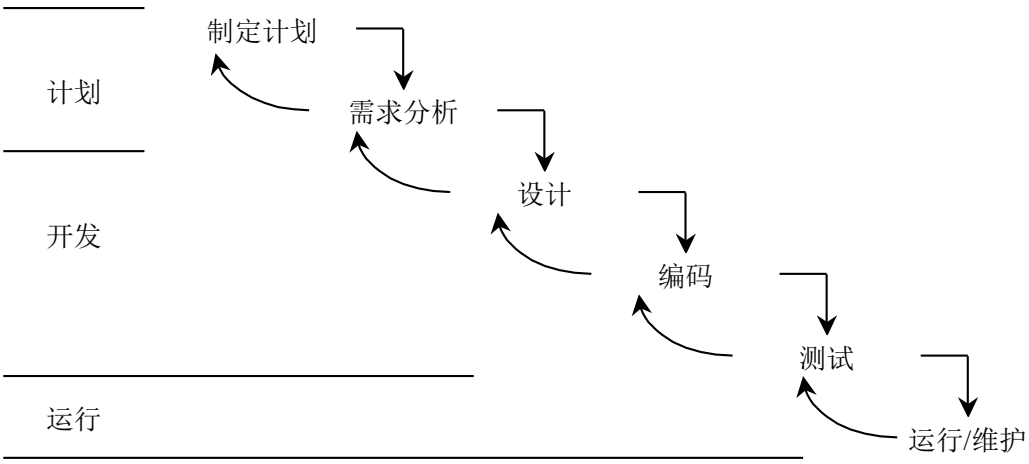
4、 程序编写

5、 软件测试

6、 运行和维护

软件生命周期经典模型——瀑布模型

软件生命周期模型是从软件项目需求定义直至软件经使用后废弃为止，跨越整个生存期的系统开发、运作和维护所实施的全部过程、活动和任务的结构框架。



软件开发模型

瀑布模型：按照软件生命周期经典模型-瀑布模型的各个阶段实施开发工作

优点：

1） 提供了软件开发的基本框架，优于“手工作坊”式的开发方法

2） 有利于大型软件开发过程中人员的组织、管理

3） 有利于软件开发方法和工具的研究与使用，从而提高了大型软件项目开发的质量和效率

缺点：

1） 在软件开发的初始阶段指明软件系统的全部需求是困难的，有时甚至是不现实的

2） 需求确定后，用户和软件项目负责人要等相当长的时间（经过设计、编码、测试、运行）才能得到一份软件的最初版本。如果用户对这个软件提出比较大的修改意见，那么整个软件项目将会蒙受巨大的人力、财力、时间方面的损失。

渐进模型（演化模型、原型模型）

原型模型在克服瀑布模型缺点、减少由于软件需求不明确而给开发工作带来风险方面，确实有着显著的效果

原型化方法是用户和软件开发人员之间进行的一种交互过程，适用于需求不确定性高的系统

螺旋模型

它是生命周期模型与原型模型的结合，不仅体现了两个模型的优点，还增加了新的成分——风险分析

螺旋模型是支持大型软件开发并具有广泛应用前景的模型

问题定义

关于问题性质、工程目标和规模的书面报告

可行性研究（也称可行性分析，Feasibility Analysis）

目的就是要用最小的代价在尽可能短的时间内确定问题是否能够解决。

具体任务：

1、 进一步分析和澄清问题定义。

2、 导出系统的高层逻辑模型。从系统逻辑模型出发，探索若干种可供选择的主要解法（即系统实现方案）。对每种解法都应该至少从下述几方面研究其可行性：

1） Economic feasibility，经济可行性。这个系统的经济效益能超过它的开发成本吗？

2） Technical feasibility，技术可行性。使用现有的技术能实现这个系统吗？

3） Legal feasibility，法律可行性。确定系统开发可能导致的任何侵权行为、妨碍性后果和责任。

4） Operational feasibility，操作可行性。系统的操作方式在这个用户组织内行得通吗？

3、对以后的行动方针提出建议。

基于计算机系统的成本—效益分析是可行性研究的重要内容，它用于评估计算机系统的经济合理性。给出系统开发的成本论证，并将估算的成本与预期的利润进行对比。

成本

软件开发成本主要表现为人力消耗（乘以平均工资则得到开发费用）。

一般来说，基于计算机系统的成本由四个部分组成：

1、 购置并安装软硬件及有关设备的费用

2、系统开发费用

3、系统安装、运行和维护费用

4、人员培训费用

1、代码行技术

代码行技术是比较简单的定量估算方法，也是一种自底向上的估算方法。它把开发每个软件功能的成本和实现这个功能需要用的源代码行数联系起来。通常根据经验和历史数据估计实现一个功能需要的源程序行数。

一旦估计出源代码行数以后，用每行代码的平均成本乘以行数即可确定软件的成本。每行代码的平均成本主要取决于软件的复杂程度和开发小组的工资水平。

大致分如下两步：

1、 对要求设计的系统进行功能分解，直到可以对为实现该功能所要求的源代码行数做出可靠的估算为止。根据经验和历史数据，对每个功能块估计一个最有利的、最可能的和最不利的 LOC 值。设最有利的 LOC 值为 a，最可能的 LOC 值为 m，最不利的 LOC 值为 b，则代码行的期望（平均）值 lc 和对期望值偏离的方差 ld 为

$$lc = \frac{(a+4m+b)}{6}$$

$$l_d = \sqrt{\sum_{i=1}^{n}(\frac{b-a}{6})^2}$$

2、 再根据历史数据和经验，选择每个软件功能块的 LOC 价格

计算每个功能块的价格及工作量，并确定该软件项目总的估算价格和工作量。

举例：CAD 软件，项目范围确定了其主要功能：

*用户接口控制（UIC） *二维几何图形分析（2DGA）

*三维几何图形分析（3DGA） *数据结构管理（DSM）

*图形显示（CGD） *外围设备控制（PC） * 设 计 分 析（DA）

功能	最有利	最可能	最不利	期望值	方差数	$/行	行/人月	人月	价格
UIC	1800	2400	2650	2340	140	14	315	7.4	32760
2DGA	4100	5200	7400	5380	550	20	220	24.4	107600
3DGA	4600	6900	8600	6800	670	20	220	30.9	136000
DSM	2950	3400	3600	3350	110	18	240	13.9	60300
CGD	4050	4900	6200	4950	360	22	200	24.7	108900
PC	2000	2100	2450	2140	75	28	140	15.2	59920
DA	6600	8500	9800	8400	540	18	300	28	151200
估算值				33360	1100			144.5	656680

（人月=期望值/行/人月，价格=$/行*期望值）

2、任务分解技术

首先把软件开发工程分解为若干个相对独立的任务，再分别估计每个单独的开发任务的成本，最后累加起来就得出软件开发工程的总成本。估计每个任务的成本时，通常先估计完成该项任务需要用的人力（以人月为单位），再乘以每人每月的平均工资而得出每个任务的成本。

最常用的方法是按开发阶段划分任务。如果软件系统很复杂，由若干个子系统组成，则可以把每个子系统再按开发阶段进一步划分为更小的任务。

功能	需求分析	设计	编码	测试	总计
UIC	1.0	2.0	0.5	3.5	7.0
2DGA	2.0	10.0	4.5	9.5	26.0
3DGA	2.5	12.0	6.0	11.0	31.5
DSM	2.0	6.0	3.0	4.0	15.0
CGD	1.5	11.0	4.0	10.0	27.0
PC	1.5	6.0	3.5	5.0	16.0
DA	4.0	14.0	5.0	7.0	30.0
总计	14.5	61.0	26.5	50.5	152.5
劳务费($/人月)	5200	4800	4250	4500	
成本	75400	292800	112625	227250	708075

用这种方法估算，CAD软件成本和工作量总计为708075元和152.5

人月。将这些数据与代码行的成本估算比较（分别为656680元和144.5人月），前者相差7%，后者相差5%，结果非常接近，可以接受。若相差较大，应该分析原因后再重新估算，结果应基本一致。

效益

系统效益包括经济效益和社会效益两部分

例 开发计算机辅助设计（CAD）系统取代当前的手工设计过程。系统分析员为当前的手工设计系统和CAD目标系统定义对应的可测试特征：

T：绘一幅图的平均时间，单位是小时/幅。

d：每小时绘图的成本，单位是元/小时。

n：每年绘图的数目，单位是幅。

r：用CAD系统绘图减少的绘图时间比例。

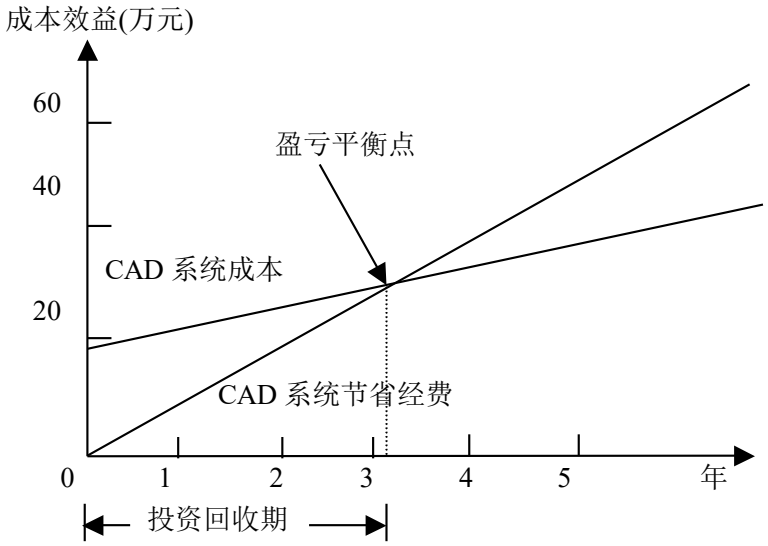
p：用CAD系统绘图的百分比。

于是，可用下式计算利用CAD系统绘图每年可以节省的经费

$$B = n \times T \times d \times p \times r$$

这样，当r =1/4，T=4小时/幅，n=8000幅，d=20元/小时，p=60%时，代入上式计算得B=96 000元，即用CAD系统绘图比用手工系统绘图平均每年约节省经费96 000元。

系统开发成本、节省的经费与时间的关系如图所示。盈亏平衡点对应



的时间坐标是 3.1 年，表示系统应用 3.1 年后可以收回系统成本。

实际上，投资是现在进行的，效益是将来获得的，不能简单地比较成本和效益，应当考虑货币的时间价值

1、货币的时间价值

通常用利率的形式来表示货币的时间价值。假设年利率为 i，如果现在存入 P 元，则 n 年后可以得到：

$$F = P(1+i)^n$$

这个 F 就是现在的 P 元钱在 n 年后的价值。

反之，如果 n 年后能收入 F 元，那么这些钱现在的价值就是

$$P = \frac{F}{(1+i)^n}$$

例如：修改一个已有的库存清单系统，使它能在每天送给采购员一份定货报表。修改已有的库存清单并且编写产生报表的程序，估计共需 5000 元；系统修改后，能及时定货将消除零件短缺影响生产的问题，估计因此每年可以节省 2500 元。五年共可以节省 12500 元。但不能简单地把现在的 5000 元与五年后的 12500 元相比。

假设年利率为 12%，可以算出修改库存清单系统后每年预计节省的钱的现在价值。

表 2-1　将来的收入折算成现在的价值

年	将来值（元）	$(1+i)^n$	现在值（元）	累计的现在值（元）
1	2500	1.12	2232.14	2232.14
2	2500	1.2544	1992.98	4225.12
3	2500	1.40493	1779.45	6004.57
4	2500	1.57352	1588.80	7593.37
5	2500	1.76234	1418.57	9011.94

2、投资回收期

通常用投资回收期衡量一项开发工程的价值。所谓投资回收期就是使累计的经济效益等于最初投资所需要的时间。

上例中，　2+（5000-4225.12）/1779.45=2.44

或者 3-（6004.57-5000）/1779.45=2.44

投资回收期仅仅是一项经济指标，为了衡量一项开发工程的价值，还应考虑其它的经济指标。

3、纯收入

衡量工程价值的另一项经济指标是工程的纯收入，也就是在整个生命周期之内系统的累计经济效益（折合成现在值）与投资之差。例，上述修改库存清单系统，工程的纯收入预计是：9011.94-5000=4011.94 元

4、投资回收率

设想把数量等于投资额的资金存入银行，每年年底从银行取回的钱等于系统每年预期可以获得的效益，在时间等于系统寿命时，正好把在银行中的存款全部取光，那么，年利率等于多少呢？这个假想的年利率就等于投资回收率。

已知现在的投资额 P，已估计出将来每年可以获得的经济效益 F_i，那么，在给定软件的使用寿命 n 年后，由 $P=\frac{F}{(1+i)^n}$，可列方程式：

$$P=\frac{F_1}{(1+j)}+\frac{F_2}{(1+j)^2}+\ldots\ldots+\frac{F_n}{(1+j)^n}$$

其中，P 是现在的投资额（即 5000 元），F_i 是第 I 年年底的效益（i=1，2……n）（本例中均为 2500 元）；n 是系统的使用寿命（5 年），j 是投资回收率

解这个高阶代数方程就可求出投资回收率

如上例，解出 j=41%-42%，远大于 12%，一般认为是值得投资的。

可行性论证的提纲

大致包括如下内容：

1、 背景情况。问题描述，市场需求等

2、 系统描述。简略的范围描述，计划目标和阶段目标等

3、 候选方案。候选方案的配置，选择最终方案的准则等

4、 价格利益分析。经费概算和预期经济效益

5、 技术冒险评价。包括技术实力、设备条件和已有工作基础

6、 操作可行性。用户组织对操作方式的希望

7、 法律可行性。系统开发可能导致的侵权、违法等

8、 其它与项目有关的问题。可能的未来变化

9、 结论。

需求分析

所谓软件需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。

需求分析具体任务：

1、 确定对系统的综合要求

1） 系统功能要求

2） 系统性能要求

3） 运行要求

4） 将来可能提出的要求

2、 分析系统的数据要求（需求分析的本质就是对数据和加工进行分析）

3、 导出系统的逻辑模型

4、 修正系统开发计划

5、 开发原型系统（可选）

需求获取技术

1、 访谈与会议

2、 书面调查法

3、 观察用户工作流程

4、 用户和开发人员共同组成联合小组

需求调查

主要包括四部分内容：

1、 组织概况

2、 组织的业务活动：

1）组织的业务状态

2） 业务的详细内容

3） 输入输出信息从六个方面着手：

*信息流向

*信息种类

*利用的目的

*信息的使用者和制造者

*输入和输出地点

*输入和输出信息量

3、 存在问题、约束条件

4、 未来要求

结构化分析方法（SA 方法）就是面向数据流自顶向下逐步求精进行需求分析的方法。

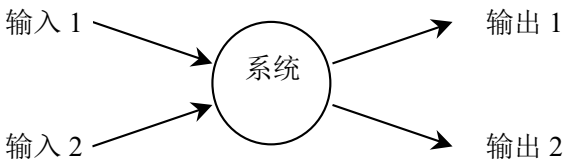
分析的过程：面向数据流，采用自顶向下、分支分层、逐步分解、逐步细化、逐步求精，最后落实到具体加工、基本加工、简单加工。用逐步求解的方法控制系统的复杂度，使得复杂系统简单化、抽象系统具体化。

数据流图（DFD）

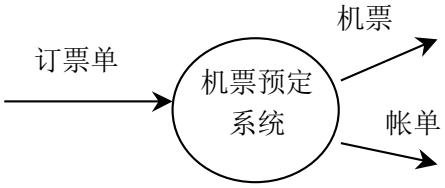
数据流图描绘系统的逻辑模型，或者说描绘信息在系统中流动和处理的情况。

基本系统模型

数据流分析将系统模型视作一种数据变换，它接受各种形式的输入，通过变换产生各种形式的输出。



例，下图是一个飞机机票预定系统的基本系统模型



数据流图的成分

下图是机票预定系统的 DFD。

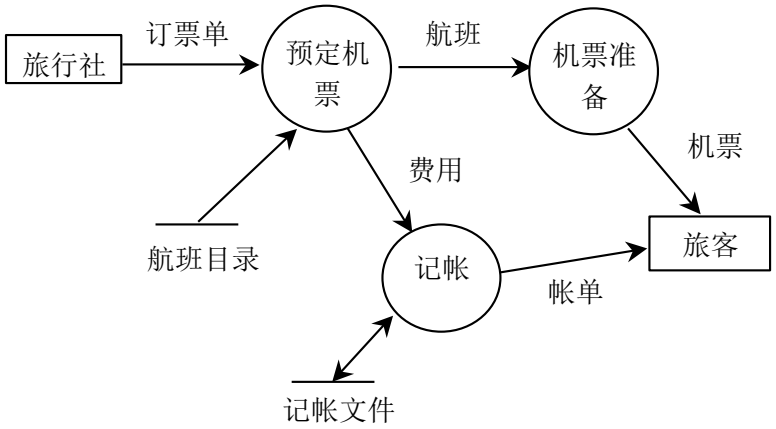
可以看出，数据流图是由以下四个基本成分组成的：

1、数据流（用箭头表示）

2、 加工（用圆圈表示）

3、 文件（用单线或双线表示）

4、 数据源点和终点（用方框表示）



几点说明：

1、 一个加工并不一定是一个程序。

2、 一个数据存储也不等同于一个文件。

3、 数据存储和数据流都是数据，但所处状态不同。

自顶向下逐层画 DFD 的步骤如下：

1） 把基本系统模型加上源点和终点作为顶层 DFD。

2） 画出各层 DFD，逐层画 DFD 的过程也就是逐层分解的过程。

为便于管理，要对分层 DFD 和图中的加工进行编号。

A、子图的图号就是分解的父图中相应加工的编号。

B、子图中的加工编号是由子图号、小数点、局部顺序号组成。

3） 画出总的 DFD。（可选）

画 DFD 时要注意的几个问题：

1、 画 DFD 不是画流程图。DFD 只描述“做什么”，不描述“怎

么做”和做的顺序。而流程图表示对数据进行加工的次序和细节。

2、 父图和子图的平衡。父图某加工的输入输出和分解这个加工的子图的输入输出必须完全一致。

3、 局部文件。文件总是局部于分层 DFD 的某一层或某几层。

4、 分解的深度与层次。逐层分解要求分解到加工成为足够简单、易于理解的基本加工为止。所谓基本加工就是加工的意义明确、操作单一。但是究竟分解的层次多少合适，应当根据问题的复杂程度来确定。一般来说，可参考以下准则：

1） 一个加工的分解，最多不要超过 7 个子加工。若超过 7 个时，可以用增加层次的办法来减少子加工数。

2） 分解在逻辑上应合理、自然，不能硬性分割。

3） 在保证数据流的易理解性的前提下，尽量少分解层次。这样可以减少层次的界面。

4） 分解要均匀。即在一张 DFD 中，不要有些加工已是基本加工，另一些加工还要分解好几层。绝对均匀不可能，但不要相差太大。

数据字典（DD）

数据字典是关于数据的信息的集合，也就是对数据流图中包含的系统元素（所有数据流、加工、文件）的定义的集合。

数据流图和数据字典是需求规格说明书的主要组成部分。

1、 数据流条目

数据流条目是定义数据流的。定义的方式一般是列出该数据流的各组成数据项。在定义数据流时，通常借助于一些简单的符号。如

= 等价于（定义为）

+ “与” x=a+b，x 由 a 和 b 组成

[…|…]，[…,…] “或” x=[… , …]，x=[a|b] x 由 a 或 b 组成

{…} 重复 x={a} x 由 0 个或多个 a 组成

m{…}n 重复 如 x=3{a}8 表示 x 中至少出现 3 次 a，至多出现 8 次 a

（） 可选（选择） 如 x=a+（b）+c b 可以在 x 中出现，也可以不出现

“…” 基本数据元素 x=“a” x 为取值为 a 的数据元素

.. 连接符 x=1..9 x 可取 1 至 9 中的任一值

如数据流“存折”可以定义：

存折={户名+所号+帐号+开户日+性质+（印密）+1{存取行}50}$_{00000000}^{99999999}$

户名=2{字母}24

所号=“001”…“999”

帐号=“00000001”…“99999999”

开户日=年+月+日

性质=“1”..“6”

印密=“0”

存取行=日期+（摘要）+支出+存入+余额+操作+复核

日期=年+月+日

年=“0001”..“9999”

月=“01”..“12”

日=“01”..“31”

摘要=1{字母}4

支出=金额

金额=“0000000.01”..“9999999.99”

操作=“00001”..“99999”

……

2、 数据文件条目

文件条目是定义文件的，一般只需列出文件的组成数据项，以及文件的组织方式

如数据文件（数据存储）“存折”可定义为

存折={户名+所号+帐号+开户日+性质+（印密）+1{存取行}50}$_{00000000}^{99999999}$

组织：按帐号次序从小至大排列

3、 加工条目（小说明）

小说明集中描述一个加工“做什么”，即加工逻辑。所谓加工逻辑

是指用户对这个加工的逻辑要求，即这个加工的输出数据流和输入数据流的逻辑关系。

目前用于写小说明（加工逻辑说明）的工具有结构化英语、判定表和判定树。

结构化英语

下面是商店业务处理系统中“检查发货单”的例子

IF the invoice exceeds $500 THEN (发货单金额超过$500)

IF the account has any invoice more than 60 days overdue THEN （欠款超过 60 天）

the confirmation pending resolution of the debt (在偿还欠款前不予批准)

ELSE (account is in good standing) (欠款未超期)

issue confirmation and invoice (发批准书及发货单)

ENDIF

ELSE (invoice $500 or less) (发货单金额未超过$500)

IF the account has any invoice more than 60 days overdue THEN (欠款超过 60 天)

issue confirmation, invoice and write message on credit action report (发批准书，发货单及赊欠报告)

ELSE (account is in good standing) （欠款未超期）

Issue confirmation and invoice (发批准书及发货单)

ENDIF

判定表

在某些数据处理问题中，某数据流图的加工需要依赖于多个逻辑条件的取值，就是说完成这一加工的一组动作是由于某一组条件取值的组合引发的。这时使用判定表来描述比较合适。下面以“检查发货单”为例，说明判定表的构成。

		1	2	3	4
条件	发货单金额	>$500	>$500	≤$500	≤$500
	赊欠情况	>60 天	≤60 天	>60 天	≤60 天
操作	不发出批准书	√			
	发出批准书		√	√	√
	发出发货单		√	√	√
	发出赊欠报告			√	

判定表由四个部分组成，虚线分割开的四部分是：

条件茬（Condition Stub）——左上部分

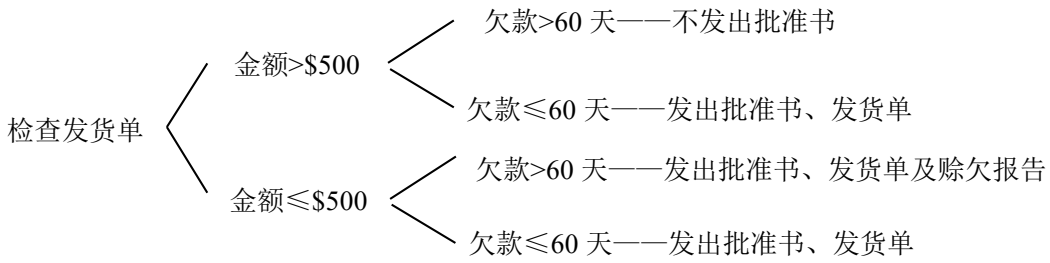
条件项（Condition Entry）——右上部分

动作茬（Action Stub）——左下部分

动作项（Action Entry）——右下部分

判定树

判定树也是用来表达加工逻辑的一种工具，有时它比判定表更直观。下面把前面的“检查发货单”的例子用判定树表示。



我们用 DFD 描述了系统中数据流动和加工的状况，用 DD 描述了其中的一些数据流、文件和加工，但是还不能满足需求分析阶段的要求。任何一个软件系统都可能有成千上万个数据项，仅仅描述它们是不够的，更重要的是如何把它们以最优的方式组织起来，以满足系统对数据的要求。

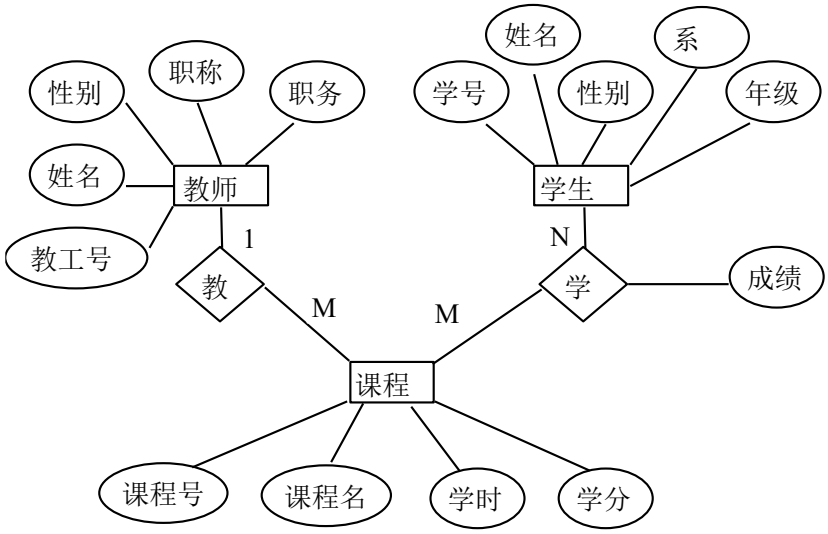
E-R 图（Entity-Relationship，实体-关系图）

1、实体

是现实世界中实体的数据侧面；或者说，数据对象是现实世界中省略了功能和行为的实体。

2、联系

客观世界中的事物彼此间往往是有联系的。同样，应用问题中的任何数据对象都不是孤立的，它们与其它数据对象一定存在各种形式的联系。



联系可分为三类：

1） 一对一联系（1：1）

2） 一对多联系（1：N）

3） 多对多联系（M：N）

3、属性

属性是实体或联系所具有的性质

软件需求规格说明

1. 引言。编写目的，项目背景等

2. 任务概述。目标，运行环境等

3. 数据描述。

4. 功能需求。

5. 性能需求。

6. 运行需求。

7. 其他需求。

一旦软件需求确认之后，就进入开发阶段。开发阶段由 3 个互相关联的步骤组成，即设计、实现（编码）和测试。

从工程管理的角度看，软件设计可分为概要设计/总体设计和详细设计两大步骤。

从技术角度来看，软件设计可分为数据设计、系统结构设计和过程设计。现在越来越多的人把界面设计也单独取出作为一个方面。

概要设计

基本目的是用比较抽象概括的方式确定系统如何完成预定的任务，即确定软件系统的总体结构，给出系统中各个组成模块的功能和模块间的接口。

涉及的概念主要有：模块化，抽象与逐步求精，信息隐藏，模块独立

在进行软件结构设计时应该遵循的最重要的原理是模块独立原理。

采用模块化原理的优点：

1） 可以使软件结构清晰，容易设计、容易阅读和理解、容易测试和调试。

2） 提高软件的可靠性。

3） 易于软件开发工程的组织管理。

所谓模块的独立性，是指软件系统中每个模块只涉及软件要求的具体的子功能，而和软件系统中其他模块的接口是简单的。

模块独立性可用两个定量准则来度量

耦合（coupling）：各个模块之间相互关联的度量

内聚（cohesion）：模块内部各元素之间相互关联的度量

耦合共分七级

非直接耦合、数据耦合、特征耦合/标记耦合、控制耦合、外部耦合、公共耦合、**内容耦合**

按上述顺序，耦合性逐渐升高，模块独立性逐渐降低。

内聚共分七级

偶然内聚、逻辑内聚、时间内聚、过程内聚、通信内聚、顺序内聚、功能内聚

按上述顺序，内聚性逐渐加强，功能趋于单一，模块独立性也逐渐加强。

软件概要设计的目标是力求增加模块的内聚，尽量减少模块间的耦合。但增加内聚比减少耦合更重要，应当把更多的注意力集中到提高模块的内聚程度上来。

系统结构设计的七条启发式原则：

1） 改进软件结构提高模块独立性

2） 模块规模应该适中

3） 深度、宽度、扇出和扇入都应适当

4） 模块的作用范围应在控制范围之内

5） 力争降低模块接口的复杂程度

6） 设计单入口单出口的模块

7） 模块功能应该可以预测

概要设计常用图形工具有：层次图、HIPO 图、结构图。

常见的软件概要设计方法有 3 大类：

1） 以数据流图为基础构造模块结构的结构化设计方法（SD，Structured Design）

2） 以数据结构为基础构造模块结构的 Jackson 方法和 LCP（Logical Construction of Programs）（Wanier）逻辑构造方法

3） 以对象、类、继承和通信为基础的面向对象的设计方法（OOD）

此外，以信息隐蔽为原则的 Parnas 方法虽然没有给出系统化的设计方法，但提出了一组原则。

结构化设计方法（Structured Design，SD）是基于模块化、自顶向下细化、结构化程序设计等程序设计技术基础而发展起来的。

概要设计的步骤：

1、 分析/重画 DFD

2、 确定 DFD 的特点及边界

3、 映射为软件结构

4、 细化后，得到初始结构图

5、 获得最终的软件结构图

变换流的 DFD 是一个线性结构，由输入、变换和输出三部分组成。变换是系统的变换中心，变换输入端的数据流为系统的逻辑输入，输出端为逻辑输出。而系统输入端的 DF 为物理输入，输出端为物理输出。如何做变换分析？

数据沿输入通路到达一个处理，这个处理根据输入数据的类型在若干个动作序列中选出一个来执行。这类数据流应该划为一类特殊的数据流，称为事务流。如何做事务分析？

概要设计说明书

1. 引言。编写目的，项目背景等

2. 任务概述。目标，运行环境，需求概述，条件与限制等

3. 总体设计。处理流程，系统总体结构等

4. 接口设计。外部接口，内部接口等

5. 数据结构设计

6. 运行设计

7. 出错处理设计

8. 其它问题