项目可行性研究

导读

做任何软件项目前，都必须进行可行性研究，即确定项目是否能开发、是否值得开发等。可行性研究是在项目建议书被批准后，对项目在技术上和经济上是否可行所进行的科学分析和论证。因为项目开发是一个综合技术、经济和管理等活动的过程，涉及多方面的因素与利益，后期维护比较频繁。所以，在软件项目开发前，必须对其进行全面分析与研究，最终以最小的代价在尽可能短的时间内达到用户满意、开发方赢利的双赢目的。

可行性研究主要从技术、经济、社会以及操作等方面出发，对以后的行动方针提出建议。技术上的研究，需要分析能否使用现有的技术实现项目；经济上的研究，需要分析开发项目获取的经济效益是否超过开发成本；社会上的研究，则应该从法律、社会效益等更广泛的方面考虑各种解决方案的可行性；操作上的研究，需要解决项目的操作方式能否在这个用户组织内行得通。

软件项目可行性研究意义重大，它既是项目投资决策的基础与依据，同时也是项目开发筹集资金和资源的重要依据；可行性研究是项目考核和之后评估的重要依据。此外，项目的管理者会根据可行性分析编制每个阶段的执行计划。

问题定义是指在对拟研发软件进行可行性分析和立项之前，对相关主要需求问题进行初步调研、确认和描述的过程。主要包括：提出问题、初步调研、定义问题、完成“问题定义报告”等。其目的是弄清用户需要计算机解决的问题根本所在，以及项目所需的资源和经费。

对于拟研发的新软件，输入（准备/基础/要求）是经过初步调研之后形成的一系列软件问题要求（业务处理等具体需求）和软件的结构框架等描述，以及预期软件支持业务过程的说明，最后的输出（完成结果）是“问题定义报告”。

“问题定义报告”主要包括：软件（项目）名称、项目提出的背景、软件目标、项目性质、软件服务范围、基本需求、软件环境、主要技术、基础条件等。

“对于问题定义阶段所确定的问题有行得通的解决方案吗?”，为了回答这个问题，必须对系统进行可行性研究，因此，可行性研究实质上是要进行一次大大压缩简化了的系统分析和设计过程。

3.1 可行性研究任务

并非任何问题都有简单、明显的解决办法。事实上，许多问题不可能在预定的系统规模或时间实现内得到解决。如果问题没有可行的解决方案，那么花费在软件项目上的时间、人力、资源和经费都是无谓的浪费。

可行性研究的目的，就是用最小的代价在尽可能短的时间内确定问题是否得到解决。因此，可行性研究之后必须下一个结论，即软件项目是否值得开发。

可行性研究需要分析几种主要的可能解决方法的利弊，从而判定原定的系统模型和目标是否现实，系统完成后所带来的效益是否值得投资开发这个系统。因此，可行性研究的实质是在较高层次上以比较抽象的方式进行系统分析和设计过程。

概括而言，可行性研究的任务是：首先，进一步分析和澄清问题定义，导出系统的逻辑模型；然后，从系统逻辑模型出发，探索若干种可供选择的主要解决方法（即系统实现方案）；最后，研究每种解法的可行性。主要从以下几个方面研究每种解法的可行性。

（1）技术可行性。

对要开发软件项目的功能、限制条件和技术特点进行分析。确定在现有的资源条件下，项目是否能够开发。现有资源包括硬件和软件资源、技术人员和管理人员水平等。主要从以下几个方面判断技术可行性：

① 开发风险。在已有的条件下，能否实现软件项目的功能和性能。

② 资源风险。现有的资源是否满足软件项目开发的需求。

③ 技术风险。开发过程中新技术的引进是否可以达到软件项目的需要。

（2）经济可行性。

对软件项目开发的成本以及效益进行估算，确定项目是否值得开发。对于大多数项目，衡量经济可行性，需要考虑一个“底线”，权衡公司长期经营策略及潜在市场前景等因素。

（3）社会可行性（法律可行性）。

应考虑项目是否存在任何侵权、责任等问题，考虑在现有的法规、制度下是否行得通，

包括责任、法律和合同等多种因素。

必要时,可行性研究也要从操作可行性和方案选择等方面研究每种解法的可行性。

(4) 操作可行性。

操作可行性是指软件项目在具体实施前及过程中的组织管理程序、方法在运用起来是否好用、是否流畅,以至于最后项目得以实现。操作可行性更侧重于组织管理。

(5) 开发方案可行性。

开发方案可行性是指对备选的解决问题的方案排队、筛选,劣中选好,好中选优,最后对要投入实施的方案进行决断的过程。

分析师应该为每个可行的解法制定一个粗略的实现进度。如果问题没有可行的解,分析师应该建议停止这项开发工程,以避免时间、资源、人力和金钱的浪费;如果问题值得解,系统分析师应该推荐一个较好的解决方案,并且为工程制订一个初步的计划。

可行性研究需要的时间取决于工程的规模。一般来说,可行性研究的成本只是预期工程总成本的5%～10%。

3.2 可行性研究过程

怎样进行可行性研究呢? 典型的可行性研究过程步骤如下。

(1) 复查系统规模和目标。

对问题定义阶段书写的关于规模和目标的报告书进一步复查确认。

(2) 研究目前正在使用的系统。

新目标系统必须也能完成旧系统的基本功能;新系统必须能解决旧系统中存在的问题。

(3) 导出新系统的高层逻辑模型。

优秀的设计过程通常从现有的物理系统出发,导出现有系统的逻辑模型;再参考现有系统的逻辑模型设想目标系统的逻辑模型;最后根据目标系统的逻辑模型建造新的物理系统。

通常,使用数据流图和数据字典能共同定义新系统的逻辑模型,再从这个逻辑模型出发设计新系统。

(4) 进一步定义问题。

分析师应该和用户一起再次复查问题定义、工程规模和目标。可行性研究的前4个步骤实质上构成一个循环。

(5) 导出和评价供选择的解法。

首先,从技术角度出发排除不可行方案;其次,考虑操作可行性,放弃用户不能接受的方案;接下来,考虑经济可行性,估计余下的每个可能的系统的开发成本和运行费用,进行成本/效益分析;最后,为每个在各方面都可行的系统制订实现进度表。

(6) 推荐行动方针。

根据可行性研究的结果应该做出一个关键性决定,是否继续进行这项开发工程。若继续开发,则选择一种最好的解法,说明选择解决方案的理由。

(7) 草拟开发计划。

分析师应该为所推荐的方案草拟一份开发计划,制订工程进度表、估计对各类开发人

员和各种资源的需要情况、估计系统生命周期每个阶段的成本、给出下一个阶段(需求分析)的详细进度表和成本估计。

(8) 书写文档提交审查。

把可行性研究各个步骤的工作结果写成清晰的文档,请用户、客户组织的负责人及评审组审查,以决定是否继续这项工程,以及是否接受分析员推荐的方案。

3.3 数据流图和数据字典

3.3.1 数据流图

数据流图或数据流程图(Data Flow Diagram,DFD)是描述系统中数据流程的一种图形化技术,它描绘信息流和数据从输入移动到输出过程中所经受的变换。在数据流图中没有任何具体的物理部件,它只是描绘数据在软件中流动和被处理的逻辑过程,描述的是业务数据的来龙去脉及加工规则。

数据流图是系统逻辑功能的图形表示,即使不是专业的计算机技术人员,也容易理解它,因此,数据流图是系统分析师和用户之间极好的沟通工具。此外,设计数据流图时只需要考虑必须完成的基本逻辑功能,完全不需要考虑如何具体实现这些功能,所以,数据流图也是软件设计的基础。

在结构化开发方法中,数据流图是需求分析阶段产生的结果。根据具体的转换方式,可以将数据流图转化为软件结构图,因此,分层数据流图是结构化开发方法的核心和基础。

1. 数据流图的基本符号

数据流程图有以下几种基本符号,如图 3.1 所示。

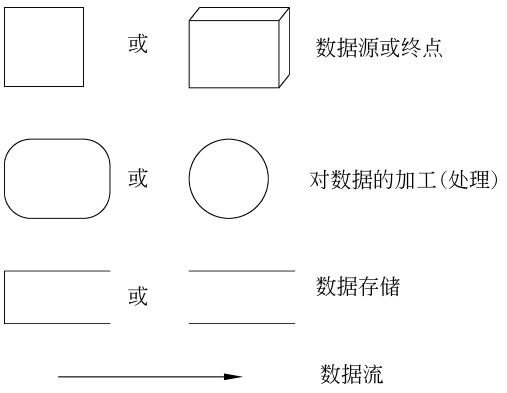

图 3.1 数据流图基本符号的含义

(1) 数据流。

数据流是数据在系统内传播的路径,因此,数据流由一组成分固定的数据组成,如订票单由旅客姓名、身份证号、航班号、日期、出发地和目的地等数据项组成。由于数据流是

流动中的数据，所以，数据流必须有流向，除与数据存储之间的数据流不用命名外，数据流应该用名词或名词短语命名。

(2) 数据源或终点。

数据源或终点代表系统之外的实体，可以是人、物或其他软件系统(不能是自身系统)。

(3) 对数据的加工(处理)。

加工就是对数据进行处理，它接收一定的数据输入，对其进行处理，并产生输出。加工不一定是一个程序，可以代表一系列程序、单个程序或某个模块。通常，在数据流图中忽略出错处理，基本要点是描绘“做什么”，而不是“怎么做”。

(4) 数据存储。

数据存储表示信息的静态存储，可以代表文件、文件的一部分、数据库的元素等。

注意，数据流图与程序流程图中用箭头表示的控制流有本质区别。在数据流图中应该描述所有可能的数据流向，而不应该描绘出现某个数据流的条件。

除了上述 4 种基本符号外，有时也使用几种附加符号，如图 3.2 所示。

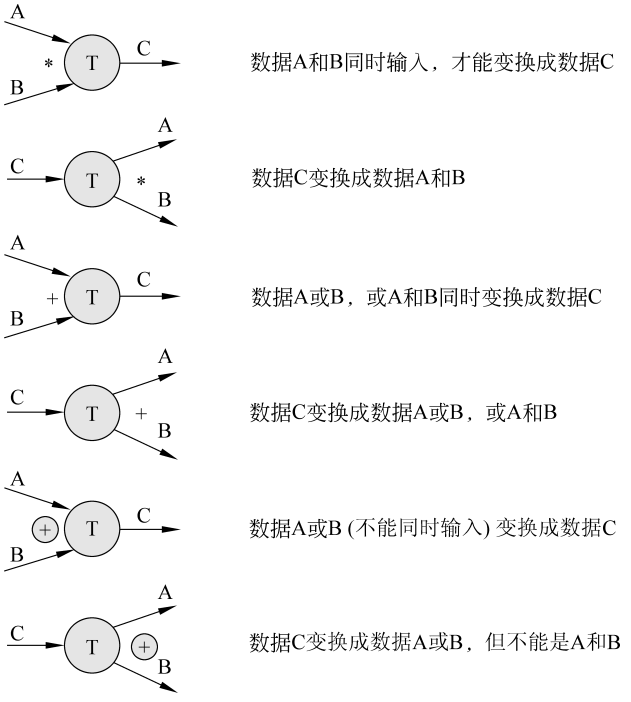

图 3.2 数据流图附加符号的含义

2. 数据流图的绘制方法

数据流图依据“自顶向下、从左到右、由粗到细、逐步求精”的基本原则进行绘制。分层数据流图如图 3.3 所示。

图 3.3 中，0 层数据流图(也叫顶层数据流图)中的加工 S 有 1 个输入和 2 个输出，1 层数据流图中将加工 S 分解成 3 个加工 1、2、3。加工 S 的输入作为分解后加工 1 的输

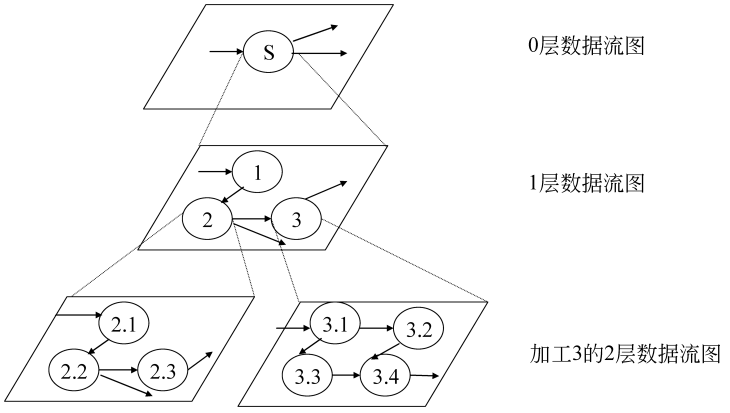

图 3.3　分层数据流图

入，S 的两个输出分别由分解后的加工 2 和加工 3 承担。1 层数据流图中的加工 3 有 1 个输入和 1 个输出，可再分解为 4 个加工：加工 3.1、加工 3.2、加工 3.3 和加工 3.4，加工 3 的输入作为分解后加工 3.1 的输入，加工 3 的输出由分解后的加工 3.4 承担。由此可以得出，在分层的数据流图中，下层图（子图）要和高层图（父图）在数据流输入和输出上保持平衡。

（1）顶层（0 层）数据流图的绘制。

0 层数据流图只有 1 张图，1 个加工，即系统自身作为一个加工。图中只需要指明数据源、输入流、输出流及数据终点。

（2）1 层数据流图的绘制。

1 层数据流图中也只有 1 张图，把 0 层数据流图中的一个加工分解成若干个加工，这些加工之间可以有 0 条或多条数据流。1 层数据流图中包括 0 层数据流图中的输入数据流和输出数据流、各个加工之间的数据流以及 1 个以上加工需要读或写的文件。

（3）加工的内部绘制。

把每个加工看作一个小系统，把加工的输入输出数据流看成小系统的输入输出流。于是，可以像画 0 层图一样画出每个小系统的加工的 DFD。

一个子图对应上层数据流图（父图）的一个加工的分解图。子图中包括父图中对应加工的输入输出数据流、子图内部各加工之间 0 个或多个数据流，以及读写文件数据流。

（4）画子加工的分解图。

对第（3）步分解出来的 DFD 中的每个加工，重复（3）和（4）分解过程，直到图中尚未分解的加工都足够简单（即不可再分解），至此便可得到一套分层的数据流图。

（5）对数据流图和加工编号。

对于一个软件系统，其数据流图可能有许多层，每层又有许多张图。为了区分不同的加工和不同的 DFD 子图，应该对每张图进行编号，以便于管理。

① 0 层图（顶层图）只有一张，图中的加工也只有一个，所以不必为其编号。

② 1 层图也只有一张，0 层图中的分解加工号分别是 1、2、3 等。

③ 1 层以下子图中的加工号由父图号、圆点和序号组成，如 1 层加工 1 的 3 个子加工的编号分别为 1.1、1.2、1.3，子加工 1.1 的子加工号则分别为 1.1.1、1.1.2、1.1.3 等。

3. 绘制数据流图的注意事项

绘制数据流图时，要注意以下几个事项：

(1) 命名。

不论数据流、数据存储，还是加工，合适的命名使人们易于理解其含义。

(2) 画数据流，而不是控制流。

数据流反映系统“做什么”，不反映“如何做”，因此，箭头上的数据流名称只能是名词或名词短语，整个图中不反映加工的执行顺序。

(3) 一般不画物质流。

数据流反映能用计算机处理的数据，因此，目标系统的数据流图一般不画物质流。

(4) 保证数据守恒。

每个加工至少有一个输入数据流和一个输出数据流，反映出此加工数据的来源与加工的结果，且输入流不同于输出流(否则，加工相当于一个管道，毫无意义)。

(5) 编号。

如果一张数据流图中的某个加工分解成另一张数据流图时，则上层图为父图，直接下层图为子图。子图及其所有的加工都应按层次级别编号。

(6) 父图与子图的平衡。

子图的输入输出数据流必须同父图相应加工的输入输出数据流一致(在数量和名字上相同)，此即父图与子图的平衡。

(7) 局部数据存储。

当某层数据流图中的数据存储不是父图中相应加工的外部接口，而只是本图中某些加工之间的数据接口，则称这些数据存储为局部数据存储。

(8) 提高数据流图的易懂性。

注意合理分解，要把一个加工分解成几个功能相对独立的子加工，这样可以减少加工之间输入、输出数据流的数目，增加数据流图的可理解性。

(9) 分解要掌握适度。

当分解涉及具体的实现一个功能时就不应该再分解了；每次只对一个功能进行分解。

4. 实例分析

下面通过案例具体说明怎样绘制数据流图。

【案例】 为方便旅客，某航空公司拟开发一个“机票预订系统”。旅客把自己的信息(姓名、身份证号码、出发时间、出发地和目的地等)输入到该系统，系统为旅客安排航班，印出取票通知和账单，旅客在起飞的前一天可以凭取票通知和账单交款取票，系统校对无误即印出机票给旅客。

绘制该案例的数据流图，可以分 4 个步骤进行描述。

(1) 从案例的描述中提取数据流图中的 4 种成分。

① 确定数据源点和数据终点。从案例描述中可以得出，数据源点和数据终点都是旅客。

② 确定加工(处理)。根据案例描述，可以得出一个明确的加工，即“安排航班”；航班

确定后，才能通知取票和账单、缴款和取票，因此，这些内容可以汇总为另一个加工“产生机票”。注意，在案例的描述中没有明确的加工时，可以根据需要抽象一种加工，在分层的数据流图中再进行细化。

③ 确定数据流。旅客将自己的信息输入到系统，显然，旅客信息是一个数据流；安排航班后，印出的取票通知、账单、款额和机票也是数据流。

④ 确定数据存储。既然要安排航班，航班信息必不可少，航班信息可视为来自另一个子系统产生的数据文件；系统为旅客安排航班后，产生一个订票信息，订票信息也是一个数据存储，由该信息可以导出取票通知、账单、款额和机票等数据流；此外，旅客信息经过审核后，主要的信息必须进行存储。为了区别数据流，将数据存储定义为“×××表”。

表 3.1 总结了步骤(1)中分析的结果。

表 3.1　案例数据流图中的 4 种成分(加“*”标记的是在问题叙述中隐含的成分)

源点/终点		加工	
旅客		安排航班、产生机票	
数据流		数据存储	
旅客信息	取票通知	旅客表	订票表
姓名	身份证号码	姓名	身份证号码
身份证号码	姓名	手机号码*	姓名
出发时间	手机号码*	身份证号码	手机号码*
出发地	出发时间	出发时间	出发时间
目的地	到达时间*	到达时间*	到达时间*
账单	航班号*	出发地	航班号*
身份证号码	出发地	目的地	航班名称
姓名	目的地	航班表*	出发地
手机号码*	款额	航班号	目的地
出发时间	付款额	航班名称	单价
到达时间*	支付类型*	出发时间	票数*
航班号*	机票	到达时间*	金额*
出发地	姓名	出发地	
目的地	出发时间	目的地	
票数*	到达时间*	票数	
金额*	航班号*	余票	
	出发地	单价	
	目的地		
	单价*		

注意，并不是所有的数据流和数据存储都能从案例问题描述中提取出来，在确定数据流和数据存储时，需要将问题描述中隐式的数据流或数据存储补充到数据流图中。

(2) 绘制 0 层数据流图。

把系统当一个加工，输入流是旅客信息，输出流是取票通知、账单和机票，图中的输入和输出都来源于表 3.1 中的数据流或数据存储，如图 3.4 所示。

(3) 绘制 1 层数据流图。

1 层数据流图中包括步骤(1)中产生的两个加工，即“安排航班”加工和“产生机票”加工。这两个加工需要进一步分解，因此，必须给这两个加工标注序号，便于引用和跟踪。

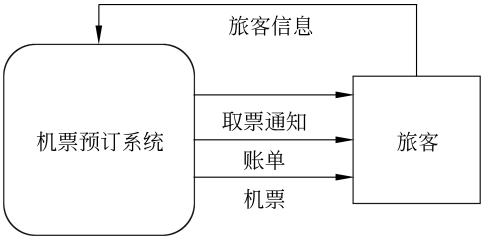

图 3.4 “机票预订系统”的 0 层数据流图

“安排航班”加工的一个输入来源于 0 层数据流图中的旅客信息(与父图保持平衡),另一个输入是航班表;输出是旅客表和订票表。

“产生机票”加工的一个输入是“安排航班”加工的输出,即订票表,另一个输入是款额;输出则是取票通知、账单和机票(与父图保持平衡)。

同样,图 3.4 中的输入和输出都来源于表 3.1 中的数据流或数据存储,数据流和数据存储可以是同样数据的两种不同形式。

“机票预订系统”的 1 层数据流图如图 3.5 所示。

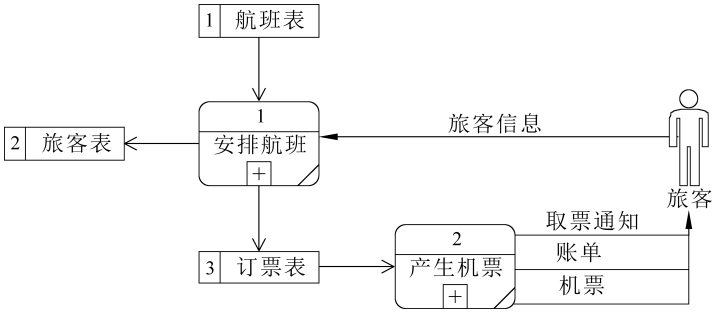

图 3.5 “机票预订系统”的 1 层数据流图

(4) 绘制 2 层数据流图。

接着,对(3) 中的 1 层数据流图进一步细化。

① 细化“安排航班”加工。考虑到旅客信息筛选后,才能产生旅客表,因此,细化出“安排航班”加工的第 1 个子加工“审核信息”,它的输入为旅客信息(来自 1 层数据流图,与父图保持平衡),输出为旅客表(与父图保持平衡),加工编号为 1.1。

“安排航班”加工的第 2 个子加工是“订票”加工,它的输入为旅客表和航班表(与父图的一个输出和一个输入保持平衡);输出为订票表(与父图的一个输出保持平衡),其加工编号为 1.2。

既然有了“订票”加工,在此可以添加第 3 个“退票”加工(虽然案例陈述中没有此功能,根据信息管理系统中数据操作的特点,应该补充),以维持系统的完整性;当然,此加工输入和输出都是订票表,加工编号为 1.3。同理,也可以添加“查询订单”加工(这里不再赘述)。

② 细化“产生机票”加工。根据案例问题描述,“产生几篇”加工可以细化为 3 个子加工,分别是“印出通知”加工、“印出账单”以及“印出机票”加工,它们的输入都是订票表,输出分别是取票通知、账单和机票,加工编号分别是 2.1、2.2 和 2.3。

细化后的加工都能产生表 3.1 中的数据流或数据存储，已经没必要进一步分解 2 层数据流图了。

最终得到的“机票预订系统”的数据流图如图 3.6 所示。

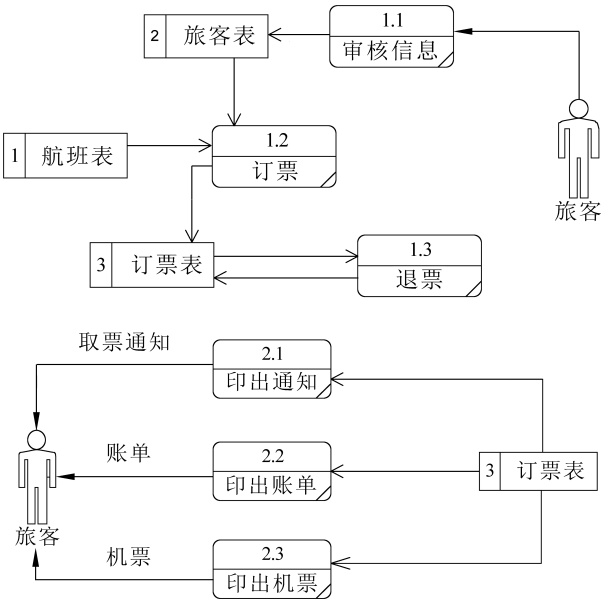

图 3.6 “机票预订系统”的 2 层数据流图

数据流图的绘制

通过实例，可以得出以下几个结论：

(1) 当对数据流图分层细化时，必须保持信息连续性，也就是说，当把一个加工分解为一系列加工时，分解前和分解后的输入输出数据流必须相同。在步骤(3)和(4)中都有详细说明。

(2) 注意加工的编号方法。把 0 分解成 1 和 2。把 1 进一步分解成 1.1、1.2 和 1.3；把 2 进一步分解为 2.1、2.2 和 2.3。

(3) 应该把握分解的尺寸。当分解后的加工能够直接通过模块设计和实现，就无须分解；否则会增加系统的复杂度。

(4) 每次分解一个加工。一些调查表明，一张数据流图中的加工应在 5～9 个，否则会难于理解。

5. 数据流图的用途

数据流图是结构化方法描述系统功能的有效模型。

(1) 利用数据流图作为用户和开发人员交流信息的工具，共同理解现行系统和规划系统的框架；并且，绝大多数用户都可以理解和评价它。

(2) 数据流图的另一个主要作用是它可以作为分析和设计的工具，着重描绘系统需要完成的功能，而不是系统的物理实现方案。

(3) 此外，数据流图对更详细的设计步骤也有帮助。在结构化方法中，结构化设计阶段将从数据流图出发映射出软件结构。因此，结构化方法是一种面向数据流的设计方法。

3.3.2 数据字典

数据字典(Data Dictionary)是关于数据的信息集合,也就是对数据流图中包含的所有元素的定义的集合。

数据流图和数据字典共同构成系统的逻辑模型,没有数据字典,数据流图就不严格;然而,没有数据流图,数据字典也就失去了解释的内容,难以发挥其作用。只有将数据流图以及数据流图中每个元素的精确定义结合在一起,才能共同构成系统的规格说明。

1. 数据字典的用途和内容

数据字典最重要的用途是作为分析阶段的工具。在数据字典中建立的一组严密一致的定义有助于改进分析员和用户之间的通信,因此会消除许多可能的误解,同时也有助于改进在不同的开发人员或不同的开发小组之间的通信;此外,如果要求所有开发人员都根据公共的数据字典描述数据和设计模块,则能避免许多麻烦的接口问题。

数据字典对数据流图中的每个构成要素做出具体的定义和说明,是系统分析阶段的重要文档。数据字典一般由4类元素的定义组成:数据流、数据文件、数据项和加工。

2. 定义数据字典

1) 数据流定义

数据流定义主要说明数据流由哪些数据项组成,包括数据流编号、名称、来源、去向、组成、时间、数量以及峰值等。其中,数据流名、组成(包含的数据项)必不可少。

常用的表示数据流组成的符号有以下几个。

(1) a+b:表示a与b。

(2) [a | b]:表示a或b,即选择括号中的某一项。

(3) m{a}n:表示a可以重复出现,至少出现m次,最多出现n次。

(4) (a):表示a可以出现0次或1次,即括号中的项可选可不选。

数据流定义示例见表3.2。

表3.2 数据流定义示例

项目名称	说明或定义		
系统名称	机票预订系统	数据流名称	机票
别名	无	编号	X0001
说明	旅客订票后,系统核对信息并收到款额后,通知旅客取票		
来源	旅客,航班	去向	旅客
数据流组成	机票=姓名+航班号+出发地+目的地+出发时间+到达时间+单价 时间=年+月+日 年=1{[0..9]}4 月=[1..12] 日=[1..31]		

2）数据文件定义

数据文件定义用于描述数据文件的内容及组织方式，一般包括系统名称、文件编号、别名、说明、组织方式、主关键字、记录数以及记录组成等。数据文件的组成可以使用与数据流组成相同的符号。

数据文件定义示例见表 3.3。

表 3.3　数据文件定义示例

项目名称	说明或定义		
系统名称	机票预订系统		
文件名称	订票表	别名	预定表
说明	存储旅客预订航班的信息，每次下单一条记录		
编号	Y0002		
组织方式	以航班为主升序、预订时间为次升序		
主关键字	身份证	记录数	30000
记录组成	记录＝身份证＋姓名＋出发地＋航班号＋航班名称＋目的地＋出发时间＋到达时间＋单价＋票数＋金额		

3）数据项定义

数据项定义是对数据流、文件和加工中所列的数据项进一步描述，主要说明数据项的类型、长度、组成以及取值范围等。

数据项定义示例见表 3.4。

表 3.4　数据项定义示例

项目名称	说明或定义		
系统名称	机票预订系统		
数据项名称	身份证	别名	无
说明	符合身份证编码规则		
数据类型	字符串	长度	18
数据项组成	身份证＝地址码＋出生日期码＋顺序码＋校验码 地址码＝6{[0..9]}6 出生日期码＝年＋月＋日 年＝1{[0..9]}4　　月＝[1..12]　　日＝[1..31] 顺序码＝3{[0..9]}3 校验码＝[0..9|X]		

4）加工定义

加工定义是针对数据流图中不能再分的加工进行描述，由加工名称、加工编号、处理逻辑、输入输出数据流等组成，其中加工编号与数据流图中的加工编号一致。

加工定义示例见表3.5。

表3.5 加工定义示例

项目名称	说明或定义		
系统名称	机票预订系统		
加工名称	订票	加工编号	1.2
输入数据流	旅客表、航班表	输出数据流	订票表
加工逻辑	旅客选中具体航班，输入预订票数即可完成预订信息的填写		
说明	通过身份验证的旅客，对有余票的航班进行订票操作		

3.4 项目可行性分析报告格式

软件可行性研究之后，需要编制一份详细的分析报告，详细记录可行性分析的内容和过程，为决策者做出决策提供依据。软件项目开发可行性研究报告格式的参考模板如下所示。

1 引言
1.1 编写目的
1.2 背景
1.3 定义
1.4 参考资料
2 可行性研究的前提
2.1 要求
2.2 目标
2.3 条件、假定和限制
2.4 进行可行性研究的方法
2.5 评价尺度
3 对现有系统的分析
3.1 处理流程和数据流程
3.2 工作负荷
3.3 经费开支
3.4 人员
3.5 设备
3.6 局限性
4 所建议的系统
4.1 对所建议系统的说明
4.2 软件系统处理流程和数据流程
4.3 影响
4.4 局限性
4.5 技术条件方面的可行性


可行性研究
报告示例

5 可选择的其他系统方案
6 投资及效益分析
6.1 支出
6.2 收益
6.3 收益/投资比
6.4 投资回收周期
7 社会因素方面的可行性
7.1 法律方面的可行性
7.2 使用方面的可行性
8 结论

3.5 成本/效益分析

开发软件是一种投资，对开发商来说，希望获得更大的经济效益。经济效益通常表现为减少运行费用或增加收入。投资开发新系统往往需要冒一定风险，随着开发的深入，一些无法预测的行为，如开发人员的流动、资金投入的延缓、用户需求的变更等，会使新系统的开发成本比预期的高，效益自然会随之降低。

成本/效益的分析目的是从经济的角度出发，分析和判断新系统的开发是否值得，进而帮助项目投资人正确做出是否投资开发新项目的决定。为了对比成本和效益，必须估算它们的数量。

3.5.1 成本估计技术

软件开发的成本主要表现人力消耗(乘以平均工资，则得到开发费用)。成本的估计不是精确的科学，因此应该使用几种不同的估计技术以便相互校验。

1. 代码行技术

代码行技术是比较简单的定量估算方法，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关联起来。通常，根据经验和历史数据估计实现一个功能需要的源代码行数。当有开发类似工程的历史数据做参考时，这个方法还是比较有效的。

一旦估计出源代码行数后，每行代码的平均成本×行数＝软件的成本，而每行代码的平均成本主要取决于软件的复杂程度和工资水平。

2. 任务分解技术

这种方法首先把软件开发过程分解为几个相对独立的任务；其次，分别估算每个单独任务的开发成本；最后，累加得出软件开发总成本。在估算每个任务的成本时，通常先估计完成任务需要的人力(以人·月为单位)，乘以每人每月的平均工资后便是每个任务的成本。

典型环境下各个开发阶段使用的人力百分比例见表 3.6。

表 3.6 典型环境下各个开发阶段使用的人力百分比例

任　务	人力/%
可行性研究	5
需求分析	10
设计	25
编码和单元测试	20
集成测试	40
总　计	100

3. 自动估计成本技术

采用自动估计成本的软件工具可以减轻人的劳动，并且使得估计的结果更客观。不过，采用该技术需要有长期搜集的历史数据，并且要有良好的数据库系统支撑。

3.5.2 成本/效益分析的方法

成本/效益分析的第一步是估计开发成本、运行费用和新系统将带来的经济效益。运行费用取决于系统的操作费用（如操作人员数、工作时间、消耗的物资等）和维护费用。新系统的经济效益等于因使用新系统而增加的收入加上使用新系统可以节省的运行费用。效益和软件生命周期的长度有关，因此，第二步要合理估算软件的寿命。预期的寿命越长，废弃的可能性越大，保险起见，在进行成本/效益分析时，假设软件生命周期为 5 年。

应该比较新系统的开发成本和经济，从经济角度判断系统是否值得开发。但是，目前的投资和未来的效益，不能简单比较成本和效益，应考虑货币的时间价值。现以实例说明，如何计算货币的时间价值、投资回收期、纯收入以及投资回收率等因素。

【例 3-1】 修改一个已有的库存清单系统，使它每天给采购员一份订货报表。修改已有的库存清单程序并且编写产生报表的程序，估计共需 5000 元。系统修改后能及时订货，这将消除零件短缺问题，估计每年可省 2500 元。

1. 货币的时间价值

通常，用利率的形式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 年后可得到 F 元。

$$F = P \times (1+i)^n \qquad ①$$

F 是 P 在 n 年后的价值。反之，如果 n 年后输入 F 元，那么现在的价值是

$$P = F/(1+i)^n \qquad ②$$

因此，5 年后库存清单系统可节省的钱并不是 12500 元，假定年利率为 12%，利用公式①和②，可以算出修改库存清单系统后每年预计节省的钱的现在的价值，见表 3.7。

表 3.7　库存清单系统将来的收入折算为现在值

年	将来值/元	$(1+i)^n$	现在值/元	累计的现在值/元
1	2500	1.1200	2232.14	2232.14
2	2500	1.2544	1992.98	4225.12
3	2500	1.404928	1779.45	6004.57
4	2500	1.57351936	1588.80	7593.37
5	2500	1.7623416832	1418.57	9011.94

2. 投资回收期

投资回收期是使累计的经济效益等于最初投资所需要的时间。显然，投资回收期越短，就能越快获得效益。

例 3-1 中，修改库存清单系统两年后可省 4225.12 元，比最初投资 5000 元少了 774.88 元，第 3 年再省 1779.45 元，那么，投资回收期是 2＋774.88/1779.45＝2.44(年)。

投资回收期仅是一项经济指标，为了衡量软件项目的价值，应考虑别的经济指标。

3. 纯收入

衡量软件项目价值的另一个经济指标是项目的纯收入，也就是整个软件生命周期内系统的累计经济效益(折合成现在值)与投资值之差。如果纯收入小于零，那么软件项目显然不值得投资。

例 3-1 中，修改库存清单系统项目的纯收入预计是 9011.94－5000＝4011.94(元)。

4. 投资回收率

投资回收率可以衡量投资效益的大小，并且可以把它和年利率比较，在衡量项目的经济效益时，它是重要的参考数据。设想把投资额存入银行，每年从银行中取回的钱就是系统每年预期可获得的效益。在投资时间等于系统寿命时，这个假象的年利率就是投资回收率。因此不难推测出：

$$P = F_1/(1+j) + F_2/(1+j)^2 + \cdots + F_n/(1+j)^n$$

其中，P 是现在的投资额；F 是第 i 年年底的效益($i=1,2,\cdots,n$)；n 是系统的使用寿命；j 是投资回收率。假设系统的使用寿命为 5 年，则例 3-1 中，修改库存清单系统的项目投资回收率为 41%～42%。

3.6　随 堂 笔 记

一、本章摘要

二、练练手

(1) 可行性研究要进行一次(　　)需求分析。

A. 详细的　　B. 全面的

C. 简化的、压缩的　　D. 彻底的

(2) 可行性研究的目的是用最小的代价在尽可能短的时间内确定问题(　　)。

A. 人员配置　　B. 具体实施过程

C. 时间长短　　D. 能否解决

(3) 在软件项目开发过程中评估软件项目风险时,(　　)与风险无关。

A. 高级管理人员是否正式承诺支持该项目

B. 开发人员和用户是否充分理解系统的需求

C. 最终用户是否统一部署已开发的系统

D. 开发需求的资金是否能按时到位

(4) 分层数据流图是一种比较严格又易于理解的描述方式,它的顶层(0层)数据流图描绘了系统的(　　)。

A. 总貌　　B. 细节　　C. 抽象　　D. 软件的作者

(5) 绘制数据流图时,遵循父图与子图平衡的原则,所谓平衡,是指(　　)。

A. 父图和子图都不改变数据流的性质

B. 子图不改变父图数据流的一致性

C. 父图的输入输出数据流与子图的输入输出数据流一致

D. 子图的输出数据流完全由父图的输入数据流确定

(6) 数据字典是软件需求分析阶段的重要工具之一,它的基本功能是(　　)。

A. 数据定义　　B. 数据维护　　C. 数据通信　　D. 数据库设计

(7) 结构化分析(SA)方法最常见的图形工具是(　　)。

A. 数据流图　　B. 实体联系图　　C. 程序流图　　D. 结构图

(8) 数据流图描述的是实际系统的(　　)。

A. 逻辑模型　　B. 物理模型　　C. 程序流程　　D. 数据结构

(9) CVS是一种(　　)工具。

A. 需求分析　　B. 版本控制　　C. 程序编码　　D. 编译

(10) 任务分解技术中最常用的是按(　　)划分任务。

A. 开发阶段　　B. 开发目标　　C. 设计过程　　D. 以上都不正确

(11) 可行性研究中所指的投资是(　　)。

A. 所有投资　　B. 直接投资　　C. 间接投资　　D. 纯金融投资

(12) 研究开发项目需要的成本和资源属于可行性研究中的(　　)。

A. 技术可行性　　B. 经济可行性　　C. 社会可行性　　D. 操作可行性

(13) 在数据流图中,(　　)说法不正确。

A. 加工必须有 1 个或多个输入

B. 加工必须有 1 个或多个输出

C. 加工的输入和输出不能相同

D. 加工之间必须有 1 个或多个数据流

(14) 关于数据流图中加工的命名规则,正确的是(　　)。

A. 加工的名字要说明对数据进行的处理和算法

B. 加工的名字要说明被加工的数据以及产生的结果

C. 加工的名字既要说明被加工的数据,又要说明对数据的处理

D. 加工的名字应该与输出结果一致

(15) 下列阶段中,(　　)的主要目的是判断项目是否有生命力,是否值得投入更多的人力和资金进行可行性研究。

A. 初步可行性研究　　B. 投资机会研究

C. 项目前评估　　D. 项目申请报告

三、动动脑

(1) 根据 3.3.1 节中案例的数据流程图初步绘制出软件结构图。

(2) 完成案例《机票预订系统》的可行性研究报告。

四、读读书

计算机软件文档编制规范 GB/T 8567—2006

实施日期:2006-7-1

颁布部门:中华人民共和国国家质量监督检验检疫总局、中国国家标准化管理委员会

本标准对软件的开发过程和管理过程应编制的主要文档及其编制的内容、格式规定了基本要求。本标准原则上适用于所有类型的软件产品的开发过程和管理过程。

软件工程标准化涉及的方面包括:

(1) 软件设计的标准化:包括设计方法、设计表达方法、程序结构、程序设计语言、程序设计风格、用户接口设计、数据结构设计、算法设计等。

(2) 文件编写的标准化:包括管理文件、项目实施计划、质量保证计划、开发进程月表、分析文件、设计文件说明书、用户文件、系统实现文件等。

(3) 项目管理标准:包括开发流程、开发作业、计划与进度管理、人员组织、质量管理、成本管理、维护管理、配置管理等。

软件工程标准应对软件生命周期中各个阶段的工作(包括技术性和管理性工作)做出合理的、统一的规定,包括对软件的对象、特性、配置、状态、动作、过程、方法、责任、义务、权限等做出具体的规定。

软件工程标准化的意义主要体现在以下几个方面:

(1) 提高软件的可靠性、可维护性和可移植性,即软件工程的标准化可以提高软件产品的质量。

(2) 提高软件的生产率。

(3) 提高软件人员的技术水平。

(4) 改善软件开发人员之间的通信效率,减少差错。

(5) 有利于软件管理。

(6) 有利于减少软件成本,缩短软件开发周期。

使用者可根据实际情况对本标准进行适当剪裁(可剪裁所需的文档类型,也可对规范的内容做适当裁剪)。

软件文档从使用的角度大致可分为软件的用户需要的用户文档和开发方在开发过程中使用的内部文档(开发文档)。供方应提供的文档的类型和规模,由软件的需方和供方在合同中规定。

总之,没有规矩不成方圆,《计算机软件文档编制规范》能规范软件开发过程、展示阶段性开发过程内容、增进开发者之间的沟通,从而为软件维护提供依据。