ICS 35.080
CCS L.77

中 华 人 民 共 和 国 国 家 标 准

GB/T 42588—2023

系统与软件工程 功能规模测量 NESMA方法

Systems and software engineering—Functional size measurement—NESMA method

(ISO/IEC 24570:2018,Software engineering—NESMA functional size measurement method—Definitions and counting guidelines for the application of function point analysis,MOD)

2023-05-23发布 2023-12-01实施

国家市场监督管理总局
国家标准化管理委员会 发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件修改采用ISO/IEC 24570:2018《软件工程　NESMA 功能规模测量方法　功能点分析应用的定义和计数指南》。

本文件与ISO/IEC　24570:2018相比，在结构上有较多调整。两个文件之间的结构编号变化对照一览表见附录A。

本文件与ISO/IEC　24570:2018的技术差异及其原因如下：

——更改了术语概念的相关表述(见3.1,1、3,1.3、3.1.5、3.1.9、3.1.14、3.1.20、3.1.24、3.1.25、3.1.27、3.1.35、3.1.36、3.1.38、3.1,39、3.1.57和3.1.60),确保术语与现行标准一致，增强功能规模测量标准之间的协调一致性；

——增加了图形用户界面、逻辑文件、荷兰软件度量协会缩略语(见3.2),因为这些缩略语在标准中使用；

—删除了ISO/IEC　24570:2018的图1,因为图中关于项目成本、生产率属性的内容不是本文件的技术内容，相关要求在其他文件中提出；

—删除了ISO/IEC　24570:2018附 录B 中 的Automated information system和File 术语，因为不符合术语的编写规则；

———把ISO/IEC　24570:2018附 录B 中的术语High level function point analysis 更改为术语估算功能点分析estimated function point analysis(见3.1.39),因为这两个术语表达的是同样的意思，“估算功能点分析”更易理解；

——删除了ISO/IEC　24570:2018的多个悬置段内容(见1.4、第2章～第9章),因为是介绍或说明性内容，与技术无关又不符合我国标准编写规则。

本文件做了下列编辑性改动：

——为与现有标准协调，将标准名称改为《系统与软件工程功能规模测量 NESMA 方法》；

———对5.5.3的公式(3)中符号进行了修正，因为原文遗漏；

——为悬置段增加了条标题和条编号；

——对图编号进行了修正，并增加了对图的引用；

——为附录 B 和附录C 中无条号的标题增加了条编号；

——增加了附录D(资料性)“本文件应用案例”。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由全国信息技术标准化技术委员会(SAC/TC　28)提出并归口。

本文件起草单位：中国石油天然气股份有限公司规划总院、北京软件造价评估技术创新联盟、中国电子技术标准化研究院、北京高质系统科技有限公司、上海市软件行业协会、道普信息技术有限公司、云南电网有限责任公司信息中心、浙江省电子信息产品检验研究院、国家应用软件产品质量检验检测中心、广西达译科技有限公司、山东省计算中心(国家超级计算济南中心)、山东山科数字经济研究院有限公司、中国航天系统科学与工程研究院、上海计算机软件技术开发中心、广东省科技基础条件平台中心、上海宝信软件股份有限公司、上海同思廷软件技术有限公司、北京软件和信息服务交易所有限公司、中国科学院软件研究所、北京中基数联科技有限公司、北京华宇信息技术有限公司、上海旋思智能科技有限公司。

本文件主要起草人：任昶、李文鹏、杨磊、张旸旸、董丰莲、严亮、杨根兴、许宗敏、王海青、苏伟、楼莉、韩明军、李刚、刘梦云、应志红、李玲璠、车江涛、殷基明、罗鲜、李敏、李旺、崔岩、吴迪龙、沈颖、韦克炜、徐泽进、冯宽、庄园、陈晓佳、潘诗祺、福德鹏、韩德隆、武海军、季永炜、袁谦、张超辉、郭栋、胡芸、郝琳、陈晓武、吕志昆、欧阳树生、杨昕、刘林、赵志宇、赵毅、李伟洪、窦文生、刘芬、刘永超。

引　言

功能点分析(FPA) 方法作为软件功能规模测量方法中的一种，最初由 A.J.Albrecht 在1974年到1979年间开发的，FPA 是对大量项目进行生产率研究得出的结果。FPA 的第一个版本是在1979年推出的，随后根据实践经验在1983年和1984年进行了修订。

FPA 引入了一个单位，即功能点，来帮助度量一个开发或维护的应用程序的规模。在FPA 的框架里，“应用程序”是指“一个自动化的信息系统”。功能点代表了应用程序提供给用户的信息处理的数量。这一测量单位独立于信息处理在技术上的实现方式。功能点是一个抽象术语，和所谓的“租赁点”进行类比。租赁点是根据房屋的房间数量、房间面积、设施数量以及房屋的位置决定的。这些作为住宅的衡量标准，提供给潜在的租户。

FPA 首先被用来测量应用程序构建后系统开发和系统维护的生产率，目前这项技术也用于支持软件项目预算编制和评估工作。

本文件是FPA 方法中的一种，按照项目所处阶段和需求的详实程度将方法FPA 分为预估功能点分析、估算功能点分析和详细功能点分析方法，为功能点分析人员提供帮助和指导。

系统与软件工程 功能规模测量 NESMA 方法

1 范围

1.1 目 的

本文件规定了应用NESMA 功能点分析方法(FPA) 的定义、规则和指南的集合。

1.2 一致性

本文件符合ISO/IEC 14143-1:2007中的所有要求性条款。

1.3 适用性

本文件适用于所有功能领域。

1.4 重 点

本文件重点关注如何确定应用软件的功能规模。本文件在基于功能点规模编制项目预算时，不充当诸如生产率标准和生产率属性等方面的角色。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

ISO/IEC 14143-1:2007 信息技术 软件测量 功能规模测量 第1部分：概念定义(Information technology-Software measurement—Functional size measurement—Part 1:Definition of concepts)

注：GB/T 18491.1-2001 信息技术 软件测量 功能规模测量 第1部分：概念定义(ISO/IEC 14143-1:1998. IDT)

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

应用程序 application

由一个或多个组件、模块或子系统组成，支持业务目标的自动化过程和数据的内聚汇集。

示例：应付账款、应收账款、工资单、采购、车间生产、装配线控制、空中搜索雷达、目标跟踪、武器发射，航班安排和乘客预订。

3.1.2

应用程序边界 application boundary

应用程序与其用户和/或其他应用程序之间的概念接口。

注：当分析的范围确定后，应用程序边界被定义。

3.1.3

应用程序功能规模 application functional size

用功能点表示应用程序的大小，由应用程序功能点计数确定。

注：通过应用程序功能规模，确定支持已实现应用程序所需的工作量。

3.1.4

自主增长 autonomous growth

在功能细化时对初始未识别但隐含在需求中的功能点展示。

3.1.5

基本功能组件 base functional component

功能规模测量方法中为了测量目标而定义和使用的用户功能需求的基本单位。

[来源：ISO/IEC 14143-1:2007,3.1]

注：NESMA FPA方法中的基本功能组件有内部逻辑文件、外部逻辑文件、外部输入、外部输出和外部查询。通过这五种基本功能组件，功能用户需求的功能规模被测量。

3.1.6

功能复杂度 functional complexity

使用本文件定义的规则为一项功能进行特定复杂度评级的权重。

3.1.7

复杂度矩阵 complexity matrix

用来给功能类型分配权重的数据表。

注：复杂度矩阵基于数据元素类型，及其引用的记录类型(对数据功能的复杂度)或者逻辑文件(对事务功能的复杂度)的数量，来分配权重。

3.1.8

概念数据模型 conceptual data model

从用户的角度解释数据组的数据模型。

3.1.9

控制信息 control information

通过指定要处理的数据的内容、时间或方式来影响基本过程的数据。

3.1.10

数据元素类型 data element type;DET

用于控制、记录和传输信息，用户所见的数据的最基本形式。

3.1.11

数据功能 data function

提供给用户以满足内部或外部数据存储要求的逻辑组合。

注：FPA 给每个数据功能分配一个类型，分为以下类型：内部逻辑文件和外部接口文件。

3.1.12

数据功能类型 data function type

FPA 分配给数据功能的一种类别：分为内部逻辑文件和外部接口文件。

3.1.13

数据信息 data information

进入或者退出应用程序的信息，这些信息满足了用户的信息需求。

3.1.14

派生数据 derived data

除了直接从数据函数中检索和验证信息之外，还包括其他步骤过程中所产生的数据。

3.1.15

详细功能点分析 detailed function point analysis

确定应用程序或者项目规模的最精确的计数。

注1:在这样的应用程序或者项目中，FPA 所需规约都已经非常详细：

注 2 :这意味着事务处理已经被定义到引用逻辑文件(也就是所谓的文件类型参考)和数据元素类型的级别，并且逻辑文件已经被定义到记录类型和数据元素类型的级别。至此，每个被认可的功能的复杂度才能够被确立起来。本文件中描述的详细功能点分析在计数中是最准确的类型，另外两种技术是估算功能点分析和预估功能点分析。

3.1.16

开发项目 development project

一个新的应用程序，或基于现有应用程序进行独立增加的规格说明、构造、测试和交付。

3.1.17

开发项目功能点分析 development project functional point analysis

应用本文件来测量开发项目功能规模的活动。

3.1.18

对话 dialog

用户和应用程序之间进行的，执行事务处理必需的信息交互。

3.1.19

基本过程 elementary process

对用户有意义的最小功能单元。

注：基本过程需要同时满足以下两个条件。

a) 该功能对于用户有着自身完备的意义，并且完全执行一个完整的信息处理过程。换句话说，功能是自包含的。

b) 当该功能执行完成后，应用程序处在一个一致状态中。

3.1.20

增强开发 enhancement development

为了改变应用程序规范，对应用程序所开展的活动。

注：这些活动通常也会带来改变功能规模的结果(例如变更请求)。

3.1.21

增强项目 enhancement project

对既有应用程序进行功能完善的项目。

注：可能在既有应用程序中添加、更改和删除功能。

3.1.22

增强项目功能点分析 enhancement project function point analysis

对实现了既有应用程序更改的项目进行测量时进行的计数。

3.1.23

错误信息 error message

当错误的数据被输入，或者当其他处理错误发生时，应用程序给出的消息。

3.1.24

外部输入 external input;EI

处理或控制来自应用程序边界之外的数据或信息的基本过程。

注：外部输入是一种基本功能组件。

3.1.25

外部查询 external inquiry;EQ

向应用程序边界之外发送数据或控制信息的基本过程。

3.1.26

外部逻辑文件 external logical file;ELF

从用户的角度来看的一组永久数据，由一个应用程序使用，但由另一个应用程序维护。

3.1.27

外部输出 external output;EO

向应用程序边界之外发送数据或控制信息的基本过程，包括外部查询之外的额外处理逻辑。

3.1.28

引用文件类型 file type referenced;FTR

被事务处理维护或者读取的内部逻辑文件(ILF) 或者外部逻辑文件(ELF)。

3.1.29

末期功能点分析 final function point analysis

为确定项目结束时的功能点数进行的计数。

注：这个计数记录下被定义或者被安装起来的应用程序的规模，或者能够确定所执行的项目的规模。

3.1.30

FPA表 FPA table

在应用程序中有次要功能的一种实体类型(例如代码表、引用表、带有常量、文本或解码的实体类型),其数据由计数的应用程序或其他应用程序维护。

3.1.31

FPA数据表ELF FPA tables ELF

为在应用系统中识别出的所有FPA 数据表集合所统计的外部接口文件，在被计数的应用程序中会使用这些FPA 数据表，但是由另一个应用程序对它们进行维护。

3.1.32

FPA数据表ILF FPA tables ILF

为应用程序中的可识别、可维护的FPA 数据表集合统计的内部逻辑文件。

3.1.33

功能 function

出于测量的目的而进行定义和使用的需求/规约的一个基本单位。

3.1.34

功能类型 function type

从 FPA 的角度所看到的组成应用程序的五种类型的组件。

注1:五种类型包括外部输入、外部输出、外部查询、内部逻辑文件和外部接口文件。

注2:这些组件确定了应用程序提供给用户的功能数量。

3.1.35

功能点 function point

功能规模的测度单位。

3.1.36

功能点分析 function point analysis

功能规模测量的方法。

3.1.37

功能点数据表 function point table

用于给功能类型分配功能点的数据表。

注：依赖于功能类型和功能复杂度。

3.1.38

功能规模 function size

通过功能用户需求进行量化而导出的软件规模。

注：在NESMA FPA方法中，这是项目或者应用程序中所有增加、更改或删除功能的功能点数量绝对值之和。应用程序功能规模和项目功能规模之间有区别。

[来源：GB/T 18491.1—2001,3.6]

3.1.39

估算功能点分析 estimated function point analysis

一种可能的功能点分析方法，在应用程序生存周期的早期阶段决定应用程序或项目的大小，程序或项目假定有一定的最小规格。

注1:通常功能数量按照类型记录，缺省值用于复杂度，平均值用于事务处理功能，低值用于数据功能(逻辑文件)。

注2:估算功能点分析是本文件所述的中间分析类型。另外两种类型的分析是详细FPA 和预估FPA。

3.1.40

预估功能点分析 indicative function point analysis

仅仅基于概念数据模型或者符合第三范式的数据模型得到一种表示应用程序或者项目的估算规模的说明。

注：预估功能点分析在本文件中是最不精确的计数类型。另外两种是详细功能点分析和估算功能点分析。

3.1.41

初期功能点分析 initial function point analysis

在项目开始时执行的功能点计数。

3.1.42

启动触发器 initiation trigger

负责激活预期事务的控制信息或功能控制数据。

注：启动触发器的例子有：按钮、功能键或菜单项。

3.1.43

中期功能点分析 interim function point analysis

在新的功能开发或者功能完善项目进行期间，为确定完成的临时功能完善规模进行的计数。

示例：为了确定对于功能规约的一个添加、更改或者删除的范围所进行的计数。

注；在应用程序功能点计数和项目功能点计数中的发生的改变都是临时功能点计数的目标。

3.1.44

内部逻辑文件 internal logical file;ILF

一组用户可辨认的在被测应用程序边界内维护的逻辑相关数据或控制信息。

3.1.45

列表功能 list function

一个显示满足，或者不一定满足特定选择标准的实体类型概要信息的在线功能。

3.1.46

逻辑文件 logical file

从用户视角所看到的一个永久数据的逻辑组。

注：逻辑文件分为内部逻辑文件或者外部逻辑文件。参见数据功能。

3.1.47

逻辑布局 logical layout

为输出产品定义的用户需要的一组数据元素类型的集合及其逻辑结构。

注：逻辑布局区别于物理实现，不涉及数据在屏幕、记录或者其他介质上的物理展现方式。

3.1.48

维 护 maintain

通过基事务功能类型添加、更改或删除数据的过程。

3.1.49

菜单 menu

显示在屏幕画面上的，展示可用功能并能选择的列表。

3.1.50

菜单结构 menu structure

通过一系列相关的菜单和屏幕画面实现应用程序与用户之间的对话。

3.1.51

输出产品 output product

由应用程序发布的一种被采用信息的逻辑形式。

注：典型的输出产品包括记录、输出文件或者发送给其他应用程序的消息。

3.1.52

项目功能规模 project functional size

用功能点表示的开发项目或者增强项目的规模。

注1:项目功能点计数能确定实现新软件，或更改既有软件的功能所需要的投入。

注2:在增强项目中， 一个项目功能点计数涉及对功能的添加、更改和删除。

3.1.53

记录类型 record type

逻辑文件中的实体类型。

3.1.54

报告 report

以用户指定的布局下的数据输出。

注：输出介质对于FPA 来说是无关紧要的，记录可能同时涉及外部输出和外部查询。

3.1.55

事务 transaction

参见：事务功能(3.1.57)。

3.1.56

事务文件 transaction file

一个临时数据文件。

注：只被读取一次，并且数据会被消耗掉。

3.1.57

事务功能 transaction function

提供给用户用以处理数据功能的基本过程。

注：事务功能包括外部输人、外部输出和外部查询。

3.1.58

事务功能类型 transactional function type

分配给事务功能的任一个类别。

注：事务功能类别包括三个：外部输入、外部输出和外部查询。

3.1.59

独特的功能 unique function

一个在形式和/或逻辑处理上不同于应用程序提供的其他任何功能的功能。

3.1.60

用户 user

在任何时刻与软件通信或交互的人或事物。

注：“事物”包括但不限于软件应用、动物、传感器或其他硬件。

[来源：GB/T 18491.1—2001,3.11]

3.1.61

用户视角 user perspective

用户描述的功能性用户需求。

注：开发人员将用户视角转换为软件，以便提供解决方案。

3.2 缩略语

下列缩略语适用于本文件。

DET: 数据元素类型(Data Element Type)

EI: 外部输入(External Input)

EO: 外部输出(External Output)

EQ: 外部查询(External inQuiry)

ELF: 外部逻辑文件(External Logical File)

FPA: 功能点分析(Function Point Analysis)

FTR: 文件类型引用(File Type Referenced)

GUI: 图形用户界面(Graphical User Interface)

ILF: 内部逻辑文件(Internal Logical File)

LF: 逻辑文件(Logical File)

NESMA: 荷兰软件度量协会(Netherland Software Measurement Association)

RET: 记录类型(Record Type)

4 FPA总则和总体要求

4.1 FPA定义

FPA 通过对应用程序的分析或对应用程序规格说明的分析，以确定其功能规模。确立应用程序或项目的功能规模的行为通常被称为“功能点分析”。

执行功能点分析应首先确定以下几点：

a) 功能点分析的目的；

b) 分析的范围和待分析应用程序或项目的边界。

4.2 FPA使用范围

FPA 可用于应用程序，也可用于项目。FPA 使用区分以下两个目标：

a) 确定应用程序功能点数：即测量应用程序将要或已提供给用户的全部功能的功能点总数；

b) 确定项目功能点数：即测量新的应用程序的全部功能或已有应用程序更改的全部功能的功能点总数。对已有应用程序的更改包括新增、更改和删除功能。在衡量项目工作量的投入和工作所需工期时，项目功能点为必要的衡量参数。

4.3 功能点分析的类型

基于所提供的功能规格说明的详细程度，功能点分析类型分为：

a) 预估功能点分析；
b) 估算功能点分析；
c) 详细功能点分析。

4.4 项目开发过程中的功能点分析

功能点分析可在项目开发的不同时期进行，根据项目开发的不同阶段功能点分析分为：
a) 初期功能点分析；
b) 中期功能点分析；
c) 末期功能点分析。

4.5 功能点分析的范围及边界

功能点分析范围是指被包括在功能点分析范围内的功能需求/规格说明的集合。 一旦确定范围就可定义边界，边界为应用程序与其用户或其他应用程序之间的概念接口。

功能点分析应首先确定范围和边界，以便能够正确执行功能点计数。

4.6 用户

FPA 有以下三种用户类型：
a) 使用或者将会使用被测量软件的人员和/或组织，包括：最终用户、职能经理和操作人员；
b) 负责确认功能规格说明中所包含的需求和愿望的雇主和/或员工；
c) 使用待分析的应用程序的数据或功能的其他应用程序。

由于功能点分析是从用户的角度出发开展的，且用户是唯一能够确定某项功能是否需要的角色，因此始终需要与用户合作来进行功能点分析，或者至少要让用户对分析结果进行验证。

4.7 功能和功能类型

功能点分析对应用程序的全部或者部分功能进行测量，它随着应用程序功能的逐渐明确而不断完善。用户需要的、认可的，并被认为是非常重要的组件称之为功能或者基本功能组件。每项功能都属于特定的功能类型。

从FPA 的视角看，应用程序中存在五种类型的组件(见图1)。这些组件决定了应用程序提供给用户的功能总量。

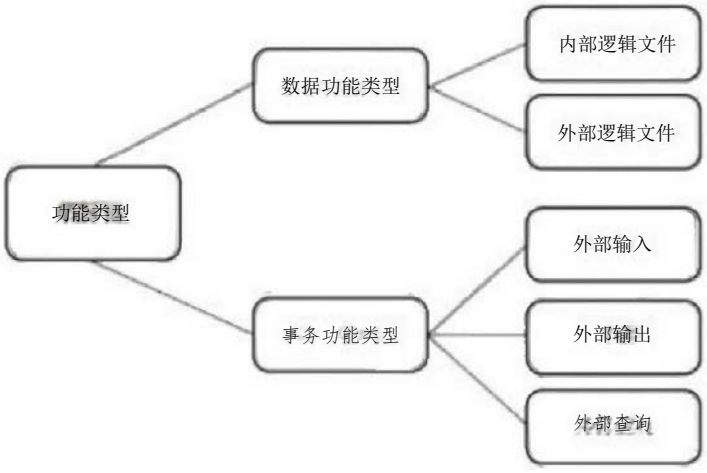

图 1 功能和功能类型

功能类型被划分成两个组：

a) 数据功能类型：包括内外逻辑文件和外部逻辑文件：

b) 事务功能类型：包括外部输入、外部输出和外部查询。

不同功能类型的赋值按照附录B 的要求执行。

4.8 功能复杂度

功能复杂度是对某一特定功能类型的复杂度评级。可使用合适的复杂度矩阵来确定功能的复杂程度。功能复杂度依赖于数据元素类型的数量，以及给定功能关联的逻辑文件数量。存在以下3种级别的复杂度：

a) 低：功能涉及很少的数据元素类型和逻辑文件；

b) 中：功能复杂度介于低和高之间；

c) 高：功能涉及很多数据元素类型和逻辑文件。

4.9 功能的赋值

在确定了功能的复杂度之后，就可为功能分配一定数量的功能点。功能点数的分配应按照表1执行。

表 1 功能点数分配表

复杂度	内部逻辑文件(LF)	外部逻辑文件(ELF)	外部输入(EI)	外部输出(EO)	外部查询(EQ)
低	7	5	3	4	3
中	10	7	4	5	4
高	15	10	6	7	6
注：加粗的数字为预估功能点分析所采用的数值。					

注：在执行估算功能点分析时，规格说明中提供的信息虽然足以用来识别功能及其类型，但是却难以识别这些功能的复杂度。在这种情况下，数据功能被认为是“低”,事务功能被认为是“中”

4.10功能规模

功能点计数是那些位于被统计对象(例如应用程序或项目)的边界之内的所有功能被分配的功能点数的总和。

功能点计数也可作为项目预算编制的基础，可将功能点的总数量乘以基于历史数据的生产率(如每个功能点的耗时率)。

对用户功能需求或软件规格说明书使用NESMA 功能点进行测量时，测量结果应标记如下：

fp(GB/T 42588—2023)

5 FPA操作准则

5.1 FPA的实施步骤

FPA 的实施包括以下步骤。

步骤1:收集可用文档。用于预估功能点分析、估算功能点分析和详细功能点分析宜提供的文件分别在5.2.1、5.2.2和5.2.3中描述。

步骤2:确定应用程序的用户(见4.6)。

步骤3:明确应执行的是应用程序功能点分析还是项目功能点分析。项目功能点分析按5.4的规定进行，应用程序功能点分析按5.5的规定进行。

步骤4:确定要分析的应用程序从其他哪些应用程序中接收和/或使用数据。

步骤5:识别功能并明确其类型和复杂度。根据4.9中的功能点分配表计算得到功能规模。同时记录功能点分析的条目和功能点数，尤其是制约因素和假设条件。

步骤6:如果涉及需要对规格说明做出特别解释的部分，可与用户共同确认。必要时可对功能点分析结果进行修正。

步骤7:需要的话可与FPA 专家一同核对功能点分析结果，并根据核对结果修正功能点分析结果。

5.2 功能点分析的类型

5.2.1 预估功能点分析

5.2.1.1 定义

预估功能点分析是基于概念数据模型或形式化数据模型估算应用程序或项目的规模，应谨慎使用这种预估功能点分析，因为可能出现高达50%的偏差。预估功能点分析的功能规模计算方式主要包括以下几种：

当有概念模型或具有类似详细程度的模型可用时，采用公式(1)计算：

$$FP=CILF\times35+CEIF\times15 \quad \text{.....(1)}$$

式中：

FP ———预估功能点计数；

CILF———概念数据模型中内部逻辑文件数量；

CEIF——概念数据模型中外部逻辑文件数量。

当FPA 表(见6.20)由被计数应用程序维护，将其计为一个内部逻辑文件。当FPA 表由其他应用程序维护，将其计为一个外部逻辑文件。因此，对于所有FPA 表，最多计算两个逻辑文件。

计算因子“35”假设每个内部逻辑文件存在三个外部输入(新增、更改、删除)、两个外部输出、 一个外部查询和一些通用功能。因此，内部逻辑文件的复杂度较低，是7fp, 事务功能的平均复杂度是26fp(3×4+2×5+4), 通用功能是2fp。

计算因子“15”是基于外部逻辑文件的低复杂度5fp、外部输出5fp 和外部查询4fp以及每个外部逻辑文件的一些通用功能1fp。

当有第三范式模型或具有类似详细程度的模型可用时，采用公式(2)计算：

$$FP=NETM\times25+NETE\times10 \quad \text{.....(2)}$$

式中：

FP —预估功能点计数；

NETM——形式化数据模型中维护的实体类型数量；

NETE ——形式化数据模型中引用的实体类型数量。

在这里，当来自FPA 表类型(见6.20)的实体类型由被计数应用程序维护，就作为维护的实体类型计数；来自FPA 表类型的实体类型由其他应用程序维护，就作为引用的实体类型计数。因此，对于所有FPA 表类型，最多计两个实体类型。

5.2.1.2 适用性

当数据模型可用时可进行预估功能点分析，并从中可导出数据功能(见5.7)。数据模型可用多种方式表示，包括Bachman 图、文本形式的描述、实体关系(ERD) 或 UML 类模型。

在应用程序生存周期的随后阶段，数据模型可是概要的(例如用敏捷的方法完成产品待办事项列

表),也可是详细的。这里的概要意味着并非所有细节都是已知的，逻辑文件和维护它们的应用程序是已知，但是没有完整的属性列表，也没有关于它们之间关系和验证的完整细节。

5.2.1.3 规格说明要求

下列内容适用于预估功能点分析：

a) 被计数的应用程序的概念数据模型或形式化数据模型；

b) 需要明确不管在此应用程序或其他应用程序中计数时，被区分的逻辑文件是如何维护的。

5.2.2 估算功能点分析

5.2.2.1 定义

估算功能点分析确定每个功能类型(事务功能类型和数据功能类型)的功能数量，并采用标准值来表示复杂度：事务功能类型的平均值和数据功能类型的低值。

这些复杂度取值优于详细功能点分析的复杂度值规则。

5.2.2.2 适用性

当数据模型(见5.2.1)可用，并且对使用它的事务功能(见4.7)有深入了解时，可进行估算功能点分析。

通常在瀑布工作模式中，估算功能点在分析阶段结束时可用；而在敏捷开发中，当用户场景与事务功能相关且数据功能可识别时，即可应用估算功能点。这时迭代代办事项(有时是产品代办事项)已经可使用了。

5.2.2.3 规格说明要求

使用估算功能点分析应具备以下条件：

a) 明确所包含的逻辑文件及其相关项；

b) 明确逻辑文件的维护情况：由被计数的应用程序或其他应用程序维护；

c) 明确应用程序功能的输入和输出信息流；

d) 明确要计算的应用程序功能与其环境之间的信息流。

5.2.3 详细功能点分析

5.2.3.1 定义

详细功能点分析是最精确的功能点分析，在该分析中，FPA 所需的所有规格应已经详细了解。事务功能已精确到引用逻辑文件和数据元素类型的级别，逻辑文件已经精确到记录类型和数据元素类型的级别。此情况下，已经可确定每个功能的复杂度。

5.2.3.2 适用性

当能够建立详细的数据模型(见5.2.1),并且对其相关事务功能已明确时，可进行详细功能点分析。事务功能对数据模型中的逻辑文件进行操作的所有细节以及关于它们的所有进一步的功能细节都应是明确的。

通常，在瀑布模型中，这些信息在功能设计期间或结束时可用。在敏捷开发中，当相应的用户描述达到“完成”的定义时可用，即在一个迭代内完成。

5.2.3.3 规格说明要求

使用详细功能点分析应具备以下条件：

a) 模型中包含所有逻辑文件及其相关项(例如Bachman 图或实体关系图)；
b) 逻辑文件的记录类型和数据元素类型；
c) 明确逻辑文件的维护情况：由被计数的应用程序或其他应用程序维护；
d) 明确应用程序功能、输入和输出信息流、与函数有关的逻辑文件以及支持功能(帮助功能等)的；
e) 应用程序的输入和输出信息流的详细规格应达到数据元素类型级别。

5.3 规格说明质量的作用

无论使用何种类型的功能点分析方法(见5.2),都需要应用程序具有完备的功能规格说明才能应用功能点分析。根据所使用的方法，所产生功能需求说明的形式可能会有所不同。但是，对同一应用程序的功能点分析宜保持相同结果。

功能点分析的可信度直接取决于所提供规格说明的质量。高质量的规格说明会减少将其转换为功能点的工作量，并且能够得出可靠的计数。规格说明有缺陷或不完整可能导致无法执行 FPA, 或者由于规格说明的歧义而导致分析结果因人而异。

5.4 项目中的FPA

5.4.1 概 述

FPA 在系统开发项目的整个过程中都发挥着作用。例如，在项目的规划阶段，将执行初期功能点分析。在项目执行期间， 一旦提交了与已经记录的规格说明相关的变更请求，便会进行中期功能点分析。项目完全完成后，末期功能点分析将确定所提供产品的规模以及项目的功能规模。

5.4.2初期功能点分析

用于开发应用程序或增强应用程序项目开始时(新增、更改或删除功能)的分析，在这个分析中需要记录应用程序(见5.5)和项目(见5.6)的功能规模。

初期功能点分析宜在项目开始时执行。根据规格的可用性，可使用预估、估算或详细功能点分析。初期功能点分析的目的是根据工作量和进度制定项目预算。

5.4.3 中期功能点分析

用于在开发项目或增强项目中发生变更时(新增、更改或删除功能)的分析。其反应了更改对应用程序功能规模(见5.5)和项目功能规模(见5.6)产生的影响。

当功能规格发生变更时，宜进行中期功能点分析。根据可用的规格说明，可使用预估、估算或详细功能点分析。中期功能点分析的目的是确定变更请求对与客户达成一致的价格和交货日期产生的影响。

5.4.4 末期功能点分析

在开发项目或增强项目结束时(新增、更改或删除功能)进行的分析，其记录了应用程序(见5.5)或项目(见5.6)的最终功能规模。

末期功能点分析是在项目结束时进行。该项目可能涉及应用程序的开发，也可能与应用程序运维阶段的功能完善有关。末期功能点分析的一个目标是确定应用程序功能的功能点规模，另一个目标是

确定项目的规模，以便确定生产率，例如每一个功能点固定价格达成共识后，可进行项目的最终结算。

5.5确定应用程序的功能规模

5.5.1 确定应用程序边界

应用程序边界是指应用程序(正在开发或已开发)与其环境(用户和/或其他应用程序)之间的边界，确定应用程序边界的方法如下。

a) 由应用程序边界划分的应用程序宜组成一个独立的整体，可在很大程度上独立于其他应用程序运行。
b) 识别所有者或主要用户。如果有多个所有者或关键用户，则通常意味着有多个应用程序。
c) 通过用户视角来查看应用程序，因此只使用用户实际可观察到的应用程序部分。从用户视角描述或定义应用程序外部的规格是应用程序环境，这被称为应用程序上下文，它可用上下文图等方式表示，来确定位于应用程序内部和外部的内容。只有用户要求的并且与之有关的事项才适用于功能点分析。
d) 将一个应用程序视为一组整体维护的程序，应用程序边界涵盖了这组程序，需要识别出所有在此边界内的功能。

5.5.2 新应用程序的功能规模

新应用程序的功能规模包括正在构建过程中或根据用户或用户组织的要求已经完成构建的应用程序的功能规模，其功能规模测量遵循以下要求。

a) 在应用程序的开发过程中确定功能规模步骤见5.1。如果开发项目中的应用程序是在单个项目中实现的，则确定应用程序的功能规模与确定项目的功能规模步骤相同(见5.6.2);确定应用程序的功能规模后，转换软件的规模不应再进行计算。
b) 如果以多个子项目合并执行的形式实现应用程序，则应检查所有子项目所提供的全部功能，以确定应用程序的功能规模。在检查这些子项目时，需要确保不对出现在多个子项目(例如逻辑文件)中的功能进行重复计数。
c) 如果对现有应用程序进行了增强(功能的添加、更改或删除),宜按照5.5.3的规定确定(更改的)应用程序的功能规模。

5.5.3 增强型应用程序的功能规模

原则上，增强可发生在应用程序生存周期的每个阶段，但通常在构建过程中或在运维过程中进行。如果进行了重大更改，则可定义一个单独的项目，除可确定应用程序的功能规模外，还可确定项目的功能规模。在所有情况下，增强后都以相同的方式确定应用程序的功能规模。

确定增强后应用程序的功能规模采取的步骤如下：

步骤1:确定更改前应用程序的功能点数(AFPB);

步骤2:确定将哪些事务和/或逻辑文件添加到应用程序，并确定它们的功能点数(ADD);

步骤3:确定从现有应用程序中删除哪些事务和/或逻辑文件，并计算它们的功能点数(DEL);

步骤4:确定更改哪些事务和/或逻辑文件。然后确定更改之前(CHGB) 和更改之后(CHGA) 的 功能点数;

步骤5:确定完善后的应用程序的功能规模(AFPA), 采用公式(3)如下：

$$AFPA=[AFPB+ADD—DEL+(CHGA-CHGB)] \quad \text{.....} (3)$$

式中：

AFPA—— 项目完善后的功能规模;

AFPB ——项目更改前的功能点数；
ADD ____项目添加的功能点数；
DEL—— 项目删除的功能点数；
CHGA——项目更改后的功能点数：
CHGB——项目更改前的功能点数。

5.5.4 重建应用程序的功能规模

如果将一个应用程序替换为具有相同功能的另一个应用程序，则新应用程序的功能规模等于它要替换的旧应用程序的功能规模。

如果增强也是这种替换的结果，则可通过两种方式确定应用程序的功能规模：

a) 替换可被认为是新的应用程序。如果选择此选项，则按照5.5.2所述进行计数；

b) 替换可被视为对要替换的应用程序的完善。在这种情况下，按照5,5.3所述进行计数。

5.6 确定项目的功能规模

5.6.1项目与应用程序功能规模差异

确定一个项目的功能规模(即计算项目的功能点数量)与单纯地计算应用程序的功能规模(即确定已提供或将要提供的功能规模)有许多不同之处，因为投入的工作量并不总是导致功能的增加，例如，考虑删除功能，或者创建一次性功能以转换数据。

下面的内容使用了几种项目的情况，详细说明了如何确定项目的功能规模。表2举例说明了如何确定项目和应用程序的功能规模以及两者之间的差异。

表 2 项目与应用的功能大小对比

规模类型	生存周期阶段					
	初始版本	版本1		版本2		
新增	1.000	200		500		
删除		40		100		
更改前 更改后		80 100		220 200		
项目大小	1.000	新增 更改后 删除 **合计**	200 100 40 **340**	新增 更改后 删除 **合计**	500 200 100 **800**	
应用程序大小	1000	初始版本 新增 更改后 更改前 删除 **合计**	1000 十200 +100 -80 -40 **1180**	版本1 新增 更改后 更改前 删除 **合计**	1180 十500 十200 -220 -100 **1560**	

确定开发应用程序的功能规模与增强应用程序不同。不过，在这两种情况下，都应首先确定应用边界，这一点在下面的子条款中有更深入的介绍。

5.6.2 确定项目功能点分析的范围

5.6.2.1 项目功能点分析的范围

功能点分析包含开发项目或增强项目的功能需求/规格说明集，可能包括一个或多个应用，因此只准许按照5.5.1的规定确定多个应用程序边界。

两种不同项目类型的定义如下。

——开发项目：实现全新应用程序的项目。在其执行过程中，一个开发项目可分为多个子项目，每个子项目负责应用程序中的某个子系统。如果子系统本身就是一个应用程序，那么每个子项目都宜被视为一个单独的开发项目。现有应用程序的重构(重新设计)也是开发项目(见5.5.4)。

增强项目：对现有应用程序进行增强的项目，该类项目可在现有应用程序中新增、更改或删除功能(见5.7)。

5.6.2.2 项目范围确定

以下步骤有助于确定项目的范围。

a) 通过用户的视角来看待要实现的应用程序。确定项目范围宜考虑逻辑结构，而非物理结构：

b) 一个应用程序可分为多个并行执行的子项目进行开发，每个子项目实现一个子系统。因此，此类子项目的范围包括子系统。如果子系统应能独立存在(例如，由于应用程序的分阶段实施或出于功能原因),那么子系统之间的数据交换也包含在每个子项目的功能规模中。该应用程序的范围包含所有子项目，这意味着子系统之间的接口位于整个应用程序边界内。项目功能规模是子项目功能点数量的总和，可高于整个应用程序的功能点数(应用程序功能规模),因为在这种情况下，如特定子项目的内部逻辑文件对于使用同一内部逻辑文件的另一个子项目，也将计为内部逻辑文件或外部逻辑文件。

c) 判断某个功能是否位于用户所需的应用程序边界内的一个实用方法，是询问用户是否真的想为这个功能付费。

d) 存在疑问时与用户进行协商。

5.6.3 开发项目的功能规模

开发项目的功能规模测量采取以下步骤。

步骤1:确定要实现的每个(子)系统的功能点数量。

步骤2:计算转换的功能点数量。这些功能点仅对项目的功能规模有帮助。所需的转换软件不产生任何附加功能，只是一个一次性工具，因此不是要实现的应用程序的一部分。

步骤3:确定项目中正在实现的其他应用程序的变更功能点数量(见5,1中的步骤1至5)。这些功能点将增加项目的功能规模(受项目影响的应用程序的新功能规模也应根据每个应用程序确定)。

步骤4:项目的功能规模是步骤1、2和3所记录的功能点数的总和。

注：项目的功能规模经常被用于预算。当不同的应用程序被更改/增强时，确保对每个应用程序做预算，因为不同的开发环境可能有不同的生产率。

5.6.4 增强项目的功能规模

增强项目确定功能点中项目功能规模的步骤如下：

步骤1:确定哪些事务和/或逻辑文件将被添加到应用程序中，并确定它们的功能点数(ADD);
步骤2:确定哪些事务和/或逻辑文件要从现有应用程序中删除，并确定它们的功能点数(DEL);
步骤3:确定要更改的事务和/或逻辑文件，然后确定它们更改后的功能点数(CHGA);
步骤4:计算增强项目的功能规模，使用公式(4):

$$EFP=ADD+DEL+CHGA \quad \text{.....(4)}$$

式中：
EFP ——项目功能点计数；
ADD ——项目添加的功能点数；
DEL—— 项目删除的功能点数；
CHGA——项目更改后的功能点数。
步骤5:如果由于更改而应制作转换软件，则需确定它的功能点数，并将其添加到步骤4中确定的项目功能规模中。

注：这些功能使用的现有内部逻辑文件和外部逻辑文件将被新增、更改或删除，但它们本身在增强项目期间不会更改，在确定项目的功能规模时不作为内部或外部逻辑文件进行计数。

5.6.5应用程序切换期间的项目功能点分析

通常有必要用效率更高且满足当今信息技术要求的应用程序来替换已长时间运行的应用程序。这需要通过再工程项目来完成。
在这种情况下只准许构建整个应用程序，因此确定项目功能规模的方式与处理开发项目时相同(见5.6.3)。

5.7 功能变更定义

5.7.1 概 述

一个或多个功能的功能规格说明会根据变更申请而调整。如果这些调整产生的活动(设计、编码、测试等等的调整)目标是使应用程序更符合变更后的规格，则这些功能宜视为增强版本内的功能。
使用与评估开发功能规模相同的计数准则来确定一个变更后功能的规模。
所有在变更需求中涉及的功能以及由于提出的变更而变更的所有功能都宜在增强版本的功能点分析中进行计数。

5.7.2事务功能更改

从用户的视角来看，更改后的事务功能宜在功能上有所改变，但用户对此事务功能的主要目的并未改变。
变更申请中所涉及的事务功能若满足下列条件之一则视为功能性更改：

a) 事务功能中新增、更改或删除一个或多个DET;
b) 事务功能中新增或删除一个或多个逻辑文件；
c) 增强版本中一个数据功能发生变更，且其中至少一个更改后的 DET 是该事务功能的一部分；
d) 增强版本中事务功能的逻辑处理方法发生改变(如新增、更改和/或删除了验证或计算的结果）。

5.7.3 数据功能更改

变更申请中所涉及的数据功能若满足下列条件之一则视为功能性更改：

a) 在增强项目中，由于一个或多个DET 被添加到数据功能中，和/或数据功能的一个或多个

DET 被改变(见5.7.4),和/或一个或多个DET 被从数据功能中移除，导致数据功能的结构发生改变；

b) 增强项目中数据功能的性质发生改变。例如，因事务功能的功能变更而使得数据功能从外部逻辑文件变为内部逻辑文件或相反的变更。

5.7.4 DET 更改

若DET 满足下列条件之一，则视为DET 更改：

a) 长度(位置数)变更；
b) 数据类型改变(如从字母数字类型变为纯数字类型)；
c) 小数点位数变更。

5.8 特定情况下的FPA

5.8.1 基于传统设计的分析

传统设计通常使用数据模型和过程模型来描述用户需要的功能。数据模型可采用实体关系图来描述实体类型、属性类型以及它们之间的链接关系。流程模型可采用数据流图的形式来描述功能和数据流之间的链接关系。模型是基于传统设计进行功能点分析的基础，同时也需要相关的屏幕布局和列表布局。屏幕和列表布局虽非必需的，却常常很有用。数据功能从数据模型中识别出来，事务功能则是从过程模型中识别。

基于传统设计进行功能点分析具有以下优势：

a) 从已明确的功能需求视角进行分析，几乎不考虑技术因素；
b) 设计是对所有功能性需求的完整说明。

5.8.2 套装软件分析

5.8.2.1 总则

当套装软件被作为一个可行的解决方案时，首先要确定套装软件是否能在现有的技术基础设施上运行。如果是，则应对选中的套装软件进行如下确认。

a) 套装软件能提供哪些用户需要的功能，这些功能的功能点是多少(套装软件在规格说明书编制阶段所计算的功能点数)。
b) 套装软件不能提供哪些用户需要的功能，这些功能的功能点是多少(套装软件既未提供也不必更改的功能点数。如果随后为了使其符合用户的所有原始需求决定升级套装软件，升级宜被视为增强(见5.5.3和5.6.4)。
c) 套装软件提供了哪些并非用户需要的功能，这些功能的功能点是多少(套装软件提供的额外功能的功能点点数。由于这部分功能是套装软件规模的一部分，项目需要支付费用。但这部分功能不是对用户有效和有用的应用程序功能。根据功能点分析的目的(测量套装软件中对用户有用的功能规模),进行套装软件功能点分析时不应将其计算在内)。

在采购套装软件时使用FPA 的一大优势在于，关注提供的功能和成本因素之间的关系，这在决定是自行开发应用程序，还是购买现成的套装软件时起到关键作用。在FPA 的帮助下，可基于功能、成本以及功能交付使用的时间节点，在自行开发和购买之间做出决定。

套装软件规模分析存在三个可能目标。

a) 确定套装软件的性价比。换句话说，套装软件所提供每个功能点的成本是多少?在进行这项评估时，只有那些用户或客户所需功能点以及套装软件提供的功能点才是相关的。
b) 确定套装软件提供的用户或客户所需功能与用户或客户所需全部功能的比例(包括新增及变

更)。

c) 确定套装软件提供的用户或客户所需功能与套装软件提供全部功能的比例。

5.8.2.2 识别套装软件的数据功能(逻辑文件)

识别套装软件的数据功能遵循以下要求。

a) 在采购套装软件时，所提供的文档中通常不包括概念数据模型。有时候，套装软件会有一个可用数据模型。数据功能就可从这个数据模型中识别出来(见6.21)。

b) 如果没有数据模型，可利用所提供的功能来识别逻辑文件。如果可用，也可从物理数据库结构中识别出来。

5.8.2.3 识别套装软件的事务功能(事务处理)

识别套装软件的事务功能遵循以下要求：

a) 如果有功能规格说明或用户手册，套装软件提供的事务功能可从这些材料中识别；

b) 如果所提供文档不足，但套装软件的测试或演示实施可用的话，则事务功能可从屏幕或窗口中识别(见5.8.3)；

c) 如果只有菜单结构的话，也可尝试从中识别出事务功能。在面对诸如“维护”这样的一个菜单选项时，可识别出三个外部输人(新增、更改和删除)。对于诸如“显示”这样的一个菜单选项，则可识别出一个外部查询或一个外部输出。

5.8.2.4 确定所需功能

确定所需功能时，使用用户或客户的功能需求进行分析，遵循以下要求：

a) 如果功能需求尚未明确，则应与用户合作，确定哪些逻辑文件和套装软件哪些功能是相关的(参考本节前面关于如何确定数据功能和事务功能的描述)；

b) 在确定套装软件总的应用功能规模时可包括套装软件中非用户要求的那些功能，但在确定有效应用功能规模时(对用户有用的应用功能规模)则不必包括。

5.8.3 屏幕或窗口分析

如果功能规格说明缺失或者不足，可通过应用程序的物理组件来识别出(部分)功能，即可通过启动需要分析的应用程序，从屏幕或窗口上开始分析。

GUI 能以多种方式向用户展示应用。可根据内部逻辑文件、外部逻辑文件、外部输入、外部输出以及外部查询来推断应用程序提供的实际功能。

GUI 环境提供的功能，或预期作为GU1 环境中的标准并由GUI 环境中的工具(几乎)自动获得的功能，如GUI 对不同设备的响应，应被视为操作系统的扩展，且不会导致额外功能或数据元素类型的计数。

当通过上述方式确定功能时注意以下几点。

a) 避免重复计数：同样一个事务功能可能出现在菜单的多处地方。

b) 连续的屏幕画面。不可分割的在一起的屏幕或窗口通常只是一个事务功能。

c) 如果出现连续的屏幕画面，但是这些画面并没有不可分割的在一起的话，除非这一新的事务功能已经在其他地方被识别出来了，这样的连续画面通常会产生新的事务功能。

d) 从屏幕画面和菜单结构找到应用程序提供哪些报告，并且将每个单个报告都作为一个外部输出计数，但需避免重复计数(见6.3)。

e) 有时可通过不同的媒介(如显示屏、弹出窗口或打印机等)展示具有相同布局的报表。当逻辑处理相同时，则仅宜对一个外部输出计数(见10.1)。

f) 一个外部查询只有被显式引用(例如在菜单结构里被引用)后才应被计数。在实际情况中，特别是当数据被更改和/或删除时，数据显示经常属于外部输入的一部分。在这种情况下，数据显示不作为外部查询进行计数。
g) 对列表功能进行计数见6.16。
h) 使用文档或者菜单选项来确定存在哪些事务处理。将这些事务处理分别作为外部输入、或者外部输出进行计数。还要确定每个事务处理可识别出多少外部输入和外部输出。
i) 可通过维护功能识别出哪些逻辑文件需要用户进行维护，即存在哪些内部逻辑文件。
j) 从屏幕画面进行分析时，通常很难确定事务功能和数据功能的复杂度。在这种情况下，将事务功能和数据功能的复杂度分别设置为平均值和低值。
k) 对菜单结构进行计数见6.15。
1) 需要深入了解定时执行的(批量处理的)功能。有时会看到屏幕画面上、文档里或者菜单里的一些注释，这些注释说明了事务处理是否会运行或者宜已经运行(如在晚上)。

5.8.4 原型设计的分析

5.8.4.1 适用范围

原型法主要适用于：
a) 作为确定功能规格说明的一种策略；
b) 作为开发已知功能规范的屏幕、窗口和对话框组织形式的一种方法。

5.8.4.2 确定原型开发规格

当使用原型法作为设计策略来确定应用程序的功能规格时，宜谨慎使用功能点分析。通常对要解决的问题，或者用户对于问题的解决方案的期望存在不确定性时，才会使用原型法。这种不确定性源自开发人员和用户之间的沟通不畅，以及当用户使用应用程序后信息需求的转变或变化。只要被开发的应用程序的最终功能存在某种不确定性，功能点分析就无法提供关于软件最终规模的可靠信息。只要有可用的(哪怕是简单的)数据模型，就可进行预估功能点分析。在进行原型开发时编制功能规格说明非常重要，这样一来在原型开发阶段结束时就可进行功能点分析。

5.8.4.3 用户界面的原型设计

如果原型开发是为了进行用户界面设计，并且功能规格说明从一开始就已记录在案的话，则可适用于应用FPA 的通用准则。

5.9 说 明 ：FPA和应用程序生存周期

5.9.1 需求阶段FPA

5.9.1.1 需求阶段活动和成果

这一阶段会对开发或者增强的应用程序进行评估，判断其在技术、经济、社会和组织一级层面上是否可行，且是有价值的。这是开发应用程序的第一步。

在应用程序生存周期的早期，对问题区域进行分析，并确定所有应用程序需求：
a) 现有的应用程序外观如何；
b) 应用程序的边界是什么；
c) 应用程序提供什么样的功能；
d) 用户如何使用应用程序；

e) 对于应用程序的质量有哪些需求；
f) 可复用现有的、计划的或者标准的硬件和软件的哪些部分；
g) 如何才能从现有的状态转换到想要的状态。

产生的成果主要包括：

a) 业务活动模型；
b) 应用程序的基本需求；
c) 将要完成应用程序的环境需求规格说明书；
d) 与技术特性相关的需求；
e) 概要数据模型。

5.9.1.2分析阶段及类型

在系统生存周期的需求阶段结束时要完成的规格说明并不总是足以用来进行估算功能点分析(见5.2.2),但是通常足以用来进行预估功能点分析(见5.2.1)。

在需求阶段估算出的应用规模往往过低。这是因为这些规格具有较高的抽象级别，可能隐藏相关细节。FPA 用户的经验表明，在一个组织内，每次对于规模低估的程度都是相同的。对此，每个组织宜确定一个适合自身的标准来弥补这个被低估的规模，可将这种弥补称为自主增长(见5.9.2和附录C)。

5.9.1.3 分析目标

分析目标是为了获得所开发应用程序的一个初始规模，可用于：

a) 确定开发应用程序所需的资源；
b) 为应用程序设计编制项目预算；
c) 为开发部门修正预算；
d) 为后续系统开发阶段的分包评估报价。

5.9.1.4 文档要求

在所有情况下，文档都应包含5.2.1中所注明的规格说明。

5.9.2分析阶段FPA

5.9.2.1 分析阶段活动和成果

在分析阶段，注意力从分析业务活动转移到对支持这些活动的应用程序进行定义上。定义应用程序存储在逻辑文件中信息的各个方面，以及用户与应用程序进行通信的方式。可创建出一定数量的模型作为所开发应用程序的蓝图。

据此可得到模型成果包括：

——用于说明哪些管理和控制决策在何时以及为何需要信息；
——用来定义哪些输出需要什么样的数据，以及这些数据的重要性是什么；
——用来确立所要开发的应用程序的数据结构、表单和数据集合；
——用来记录实现应用程序需要的技术需求。

系统开发方法的不同会导致这里使用的模型以及由此产生的文档不同。

5.9.2.2 分析阶段和类型

分析阶段输出足够详细规格支撑估算功能点分析。

一旦有估算功能点分析所需的规格，就可开始分析。根据所采用的分析方法的不同，分析可发生在

分析阶段的不同时刻。不过，通常在分析阶段结束时开始估算功能点分析。

在分析阶段得到的应用规模通常还是过低的，这是因为文档中高度抽象的需求隐藏了相关细节。FPA 用户的经验表明，在一个组织内，每次对于规模低估的程度都是相同的。对此，每个组织宜确定一个适合自身的标准来弥补这个被低估的规模。可将这种弥补称为自主增长。分析阶段的自主增长比需求阶段的要低。

自主增长不同于范围蔓延，自主增长是通过确定功能，同时细化和详察功能性用户的需求来实现；涵盖了需求已隐含但最初未被识别的功能。

范围蔓延源于用户增加新功能，生成的功能需求在详细说明需求中也无法体现。

5.9.2.3 分析目标

在分析阶段结束时进行功能点分析的目的是更好地获得待开发应用程序的规模，可用于：

a) 确定开发应用程序所需的资源；
b) 为应用程序开发项目编制预算；
c) 为开发部门修正预算；
d) 为后续应用程序开发阶段的分包评估报价；
e) 当分析阶段通过合同分包时，通过成本核算解决与分析阶段相关的财务事项(在这种情况下，可使用每个指定功能点的固定价格)；
f) 记录和上次标示的功能点分析相比应要多做或者少做的工作。

5.9.2.4 文档要求

在所有情况下，文档都应包含如5.2.2中所注明的规格说明。

5.9.3 功能设计阶段FPA

5.9.3.1 设计阶段活动和成果

设计阶段编写的设计规格可作为实现人工处理程序或计算机程序的基础。在此基础上，逻辑数据结构也被确定下来，这样就能作为创建技术数据结构或者数据库设计的基础。

5.9.3.2 分析阶段和类型

在功能设计阶段，进行详细功能点分析的所有规格说明都已经齐全了。如果在后续阶段对于功能规格没有变更，那么此时得到的详细功能点分析将等同于最终应用程序功能点分析。

同分析阶段一样，可在分析阶段进行中或分析阶段结束时进行功能点计数。

5.9.3.3 分析目标

基于功能设计阶段结束时详细功能点分析结果，可完成：

a) 为实现应用程序而对项目的后续开发进行预算；
b) 估算所需工作量和成本；
c) 评估开发阶段外包的报价；
d) 当设计阶段通过合同分包时，通过成本核算解决与设计阶段相关的财务事项(在这种情况下，可使用每个指定功能点的固定价格)；
e) 和上次标示的功能点分析相比，记录只准许多做或者少做的工作。

5.9.3.4文档要求

在所有情况下，文档都应包含如5.2.3中所注明的规格说明。

5.9.4 开发阶段FPA

5.9.4.1 开发阶段活动和成果

开发阶段包括了应用程序的构建和测试活动。当不是自主开发，而是采购标准套装软件时，该阶段的评估参考套装软件的分析。

5.9.4.2分析阶段和类型

在开发阶段中，如果发生需求变更，就可进行功能点分析。从本质上看，规格说明要回退到分析阶段或者功能设计阶段。在这个阶段一旦发生功能规格说明的变更就要进行中期功能点分析。中期功能点分析反映了变更对项目功能规模和应用程序功能规模的影响。

这里的变更一般是小的功能改进。如果发生了重大的变更通常会启动一个新项目来完成。

在开发阶段结束时，为了确定最终实现的功能规模(应用程序的规模),应对最终开发出来的应用程序再一次进行分析。这称为详细功能点分析。

在开发阶段结束时的功能点分析，也可通过在功能设计阶段之后进行的功能点分析和所有在应用开发阶段发生的中期功能点分析推导出来。

5.9.4.3 分析目标

中期功能点分析的目标主要包括：

a) 记录下需要在原先确定的价格上，根据工作量的增加或减少来相应增加或者减少费用；
b) 确定软件新的规模，以便了解变更对应用程序运行和维护阶段所造成的影响。

在开发阶段结束时进行详细功能点分析的目标如下：

a) 记录应被维护的功能点数量；
b) 根据功能点的增长来测量设计的稳定性；
c) 利用项目文档、末期功能点分析以及项目所花费的工时可测算应用程序开发的生产率。

5.9.4.4 文档要求

本阶段结束时提供的文档对功能点分析不重要。但是分析和功能设计阶段的文档(其中进行了功能更改)非常重要。估算功能点分析(见5.2.2)和详细功能点分析(见5.2.3)所需的的所有文档仍宜明确，因为它与此类变更有关。

5.9.5 实施阶段FPA

5.9.5.1 实施阶段活动和成果

在实现阶段，需要实施的任何变更都已发生，并且已完成数据切换和软件安装。培训和编写用户使用手册也是属于应用程序实施阶段的活动。

5.9.5.2 分析阶段和类型

在实施阶段无功能点分析。

5.9.6运行和维护阶段FPA

5.9.6.1 运行维护阶段活动和成果

在运行维护阶段，应用程序宜以适当的方式进行管理和维护。

5.9.6.2分析时刻和类型

在运行和维护阶段通常会发生小的功能变更。在完成变更后，为了重新确定应管理和维护的功能点数量，应再次进行功能点分析。

在发生大规模的功能改进时，通常会启动新项目，在项目中进行如上所述的功能点分析。

5.9.6.3 分析目标

在此阶段，可用运行中应用程序的功能规模来建立所维护的功能点数量和维护工作量之间的关联。

另一个值得研究的是已安装应用程序的功能点数量和在可用硬件上运行这些应用程序的成本之间的联系。

5.9.6.4 文档要求

到目前为止产生的所有文档均宜齐备。这些文档应包含所有要安装和维护应用程序的功能规模。

6 FPA通用准则

6.1 从逻辑视角进行分析

功能点分析基于组织的业务活动。 一项业务活动可包含一个或多个功能。相反， 一个功能也可支持多项业务活动。在功能点分析时不宜考虑某一特定的应用程序是如何在技术上实现的。业务的逻辑视角才是最重要的。所使用的技术不应影响应用程序的功能点大小。

6.2 应用规则

在应用本准则时应要达成一些共识。如果有任何偏离规则以及应使用经验判断的内容，请务必记录下来和澄清它们。

不要主观地判断功能项的复杂度。

6.3不重复计算

在一个应用程序中， 一个功能只能被计数一次，无论是被一位或是多位用户使用，无论该功能出现在应用程序的一处或是多处位置。这样确保一个特定的逻辑文件在一个应用程序中只会被计数一次：要么作为内部逻辑文件，要么作为外部逻辑文件，但不能同时作为两者计数。例如“更改客户”的功能在一个应用程序中出现多次，只要这个功能是相同的，则只被计数一次。

6.4 构建的功能、非用户需求的功能

在应用程序构建过程中，可能会从现存的应用程序中拷贝代码。因此，额外的功能有时会被构建到应用程序中，而不仅仅是客户最初要求的功能。此外，开发人员会对应用程序做一些技术优化，这些优化并不是用户的需求。

明确应用程序提供的功能与用户需求的功能是否对应。非用户需求的功能可被计入所提供的应用程序规模，但是不能被计入用户需求的应用程序规模。

用户需求的功能是指在用户在初始定义的功能，以及后期用户变更请求的功能，如图2所示。

<table>
<tr><td colspan="2">提供的功能</td></tr>
<tr><td>用户需求的功能</td><td rowspan="2">北用户需求的功能</td></tr>
<tr><td>初始定义的变更请求</td></tr>
</table>

图2用户需求的功能

6.5 复用代码的开发

有时，软件的通用方式开发也可在其他应用程序中复用，此类软件也可作为应用程序之外的功能单元来使用。复用代码的开发不影响功能点数量。

6.6 复用既有代码

如果在构建应用程序时复用了既有的代码，那么就是用既有的软件来实现特定的功能。“复用”可使得某些特定功能的开发变得更加简单。通常这些功能也被算在功能点计数里。在这种情况下，对于应用程序中可能完全或部分复用的功能，可适当调整生产率标准(每个功能点需要更少的工时)。

6.7 屏幕、窗口和报告

屏幕、窗口和报告是应用程序和用户之间进行信息交互的典型代表。正因如此，它们形成了应用程序和外部环境进行信息交互的物理结构。如果缺少了逻辑结构的描述(例如，如果缺少了属于多个不同的屏幕和报告的数据元素类型的描述),则有必要通过(物理的)屏幕、窗口和报告来推断出这个描述。

注 ：一个物理结构由多个逻辑结构组成， 一个逻辑结构由多个物理结构组成。

6.8 输入和输出记录

输入和输出记录是应用程序之间通信的典型代表。正因如此，它们形成了应用程序和外部环境进行信息交互的物理结构。如果缺失了逻辑结构，则有必要要通过(物理的)记录格式来推断出逻辑结构。

注：这种类型的物理记录由多个逻辑结构组成， 一个逻辑结构由多个物理记录组成。

6.9 信息安全和授权

信息安全功能、授权功能和登录功能通常是标准功能，原则上可用于所有应用程序，因此，它们通常不纳入功能点计数；如果它们应在当前计数的应用程序中构建，那么应纳入功能点计数。

6.10 操作系统和工具

操作系统和工具几乎是每台计算机的标准配置，原则上可用于所有应用程序，因此它们通常不纳入功能点计数。

对操作系统进行更改和裁剪来适应特定应用程序的需要会改变生产率，但是这种操作不增加任何用户功能，所以不会影响功能点计数。

6.11 报告生成器和查询工具

报告生成器和查询工具主要包括以下三种类型：

a) 开发环境提供的标准工具，用户可用于定义选择和输出产品；
b) 应用程序中包含的特制工具，用户可用于定义选择和输出产品；
c) 定期的外部输出和外部查询，其中选择和输出产品是固定的。

在确定项目或者应用程序的功能点计数时，如果报表生成器和查询工具是开发环境的一部分，则不

纳入功能点计数。

如果是用户要求开发的报表生成器和查询工具，则应作为应用程序的一部分，纳入功能点计数。基于与用户交互的信息流计数功能项(内部逻辑文件，外部逻辑文件，外部输入，外部输出和外部查询),这些功能项组成报告输出产品和保存用户自定义查询。

定期的外部输出和外部查询宜按照第10章和第11章中的说明进行计数，即使这些功能是在标准报表生成器或查询工具的帮助下构建的。

6．12图形

和报表一样，图形也可视为一种输出。这里的功能点分析并不是制作特定图形并向用户显示所需的技术，而是所使用或显示的图形中的信息。因此，FPA 应使用图形中跨越应用程序边界的数据元素类型来确定输出的复杂度。

6.13帮助功能

如果应用程序提供帮助功能，则应用程序中的每一种类型的帮助功能计数为一个外部查询。

帮助功能的示例包括但不限于以下内容：

——整个应用程序的帮助信息；

——屏幕或窗口的帮助信息；

——字段的帮助信息(包括具有固定值的列表函数)；

——交互式帮助向导；

——上下文帮助索引。

考虑以下情况：如果在每个屏幕或窗口都可调用的帮助信息，那么只能将其整体计数为应用程序的一个外部查询。总的来说，在被计数的应用程序中有多少种帮助功能，就最多有多少个外部查询。如果帮助文本可被维护，则将存储帮助文本的文件视为FPA 表(见6,20)。

在详细功能点分析中，作为外部查询的帮助功能复杂度应被评估为“低”;在估算功能点分析中应被评估为“中”。

6．14消息

消息分为两类：

——计算机系统消息；

——功能消息。

计算机系统消息是由操作系统或其他系统软件生成，这些消息不纳入功能点计数。

功能消息是由应用程序的事务生成，它说明了该事务的使用情况。

如果功能消息与外部输入、外部输出或外部查询相关，那么在特定功能中产生的所有消息作为一个额外的数据元素类型进行计数。

根据第10章中所描述的准则，供多个事务功能调用或同一事务功能反复调用的功能消息(例如日志报告或错误报告)被计数为外部输出(见第10章)。

注：当消息有一个单独的实体并且由用户进行维护时，将该实体视为FPA表(见6.20)。

6.15菜单结构

菜单结构通常不纳入功能点计数。然而，对于每一个被识别的事务功能的启动，不论在菜单结构中需要多少实际的步骤数量，按一个数据元素类型进行计数。如果用户自己可维护菜单结构和文本，则根据计算逻辑文件和外部输入的准则进行计数。

6.16 列表功能

根据5.4描述的准则，显示一份用户可选择的列表(例如，选择界面，窗口、选择功能，选择列表，列表框或弹出窗口),被视为外部输出。显示列表不是外部查询，因为列表的大小事先并不知晓。任何可用的选项不作为独立功能计数。

切记当列表显示存储在FPA 表类型的实体中的数据时(参考6.20),不作为独立功能计数，因为FPA 表 ILF 和 ELF 的功能已经有整体分析确定的标准，参考6.20。

如果列表显示的数据既不在逻辑文件(内部逻辑文件或外部逻辑文件)中，也不在FPA 表中，那么这个列表功能宜被作为字段级别的帮助功能进行计数(参考6.13)。

6.17 浏览和滚动功能

如果应用程序依据非唯一性准则或基于所筛选数据输出结果，则将其计为外部输出。这时屏幕上呈现出选择项的概览(满足条件的每一项对应一行),同时用户可浏览每一项的详细信息(每一项对应一个屏幕或窗口)。对于能够浏览或滚动查看输出，不能算作附加的功能或数据元素类型。

6.18 清除功能

在应用程序中，可在线执行或定期自动执行的删除或归档数据功能，如果是为了满足用户需求，可将每个清除功能计为一个外部输入，并根据通用准则来识别和评估外部输入(参考第9章)。在功能的级别进行计数，不要将每个内部逻辑文件计为一个外部输入。

6.19 功能点分析的完整性检查

对于每个内部逻辑文件，包含至少一个外部输入、 一个外部输出，如果可能的话还包括一个外部查询。对于每个外部逻辑文件，包含至少一个外部输出，或者一个外部查询。另外，外部逻辑文件也可能仅被读取用于验证或编辑目的，因此不需要任何外部输出或者外部查询。如果缺失了这些功能，请询问用户是否有所遗忘，以及文件是否真的和应用程序相关。

6.20 FPA表

应用程序中带有常量、文本、解码等信息的实体类型，被称为FPA 表。用户可在应用程序帮助下进行维护的FPA 表被共同计数为一个内部逻辑文件：FPA 数据表ILF。由其他应用程序维护的FPA 表被共同计数为一个外部逻辑文件：FPA 数据表ELF。如果实体类型不能被维护，它可能是一个系统表，不纳入 FPA 计数。

以下准则用于判断应被算作FPA 表进行计数的实体类型，只要满足其中一个条件，实体类型就是FPA表。

满足以下情况下之一的，实体类型是FPA 表。

a) 无论有多少个数据元素，实体类型应包含一个，且只能包含一个数据项(不能多，也不能少)。

示例1:含有特定组织数据(例如名称和地址)的实体类型。

b) 实体类型只包含常量数据(原则上)。

示例2:实体类型“化学元素”:助记符，原子序号，描述(所有数据元素类型都是常量)。

示例3:在4.9中所示的对功能类型进行赋值的功能点表，其中所有数据元素都是常量。

c) 实体类型由一个键值(可能是复合键)与一个或者多个解释性描述组成，前提是解释类似。

示例4:国家：国家代码、国家英文名、国家法文名(例如中国、CN、CHINA)。

d) 实体类型包括边界值，算法，以及最大或最小值，前提是关键字是独立的。

示例5: 电话号码范围，范围编号、最小的电话号码、最大的电话号码。

以下实体类型不是FPA 表：

a) 实体类型中包括金额、税率、增值税百分比等数据，如果这些数据不是常量；

b) 实体类型中包括几种不同类型的数据(除去上文所列)。

示例6:买方数据，买方编号，买方姓名，地区名(地区名是不同的数据元素类型)。

注意要判断一个实体类型是独立组成一个逻辑文件还是和其他实体类型共同组成逻辑文件。

注：上述对作为FPA 表或逻辑文件(一部分)的实体类型的总结并没有涵盖所有可能的情况。当有疑问时，在本文件的上下文语境中对实体类型进行评估。

执行以下操作以确定FPA 数据表ILF 和FPA 数据表ELF 的复杂度：

a) 统计属于该组的不同FPA 表的总数，作为记录类型数量；

b) 统计所有FPA 表的不同数据元素的总数，作为数据元素类型数量。

此外，对于FPA 表ILF, 总是计数一个外部输入， 一个外部输出和一个外部查询数据表。对于FPA 数据表ELF, 不需要计数外部输入、输出和查询。

执行以下操作以确定FPA 表ILF 的标准外部输入，外部输出和外部查询的复杂度：

a) 统计属于FPA 表ILF 的不同实体类型的总数，作为引用的记录类型数量；

b) 统计属于FPA 表ILF 的所有实体类型中的数据元素总数，作为数据元素类型数量。

6.21 从规范化的数据模型中衍生出逻辑文件(数据功能)

6.21.1 通则

在 FPA 里，逻辑文件(内部逻辑文件或者外部逻辑文件)是概念实体类型。概念实体类型由来自符合第三范式要求的数据模型中的一个或多个实体类型组成，这些实体类型被用户视为一个逻辑单元。

本文件中的术语“实体类型”是指第三范式的数据模型中的实体类型。

如果可使用符合第三范式要求(或等效的其他形式)的数据模型，则可使用以下规则来确定逻辑文件(数据功能)。

在应用准则时注意以下事项：

a) 不在规范化数据模型中的逻辑文件也应被计数(例如，包含汇总数据的历史文件),见5.2;

b) 可能有些实体类型出现在规范化的数据模型中，但并不是逻辑文件(例如，临时文件);

c) 需要仔细审查数据模型和关系的性质，尤其是强制性和可选性。

6.21.2 反规范化的规则

为了从符合第三范式要求的数据模型中衍生出逻辑文件，将执行以下步骤。

a) 确定数据模型中的哪些实体类型是FPA 表，并进一步判断它属于FPA 表ILF 还是FPA 表ELF。FPA 表以特定方式估值(见6.20、7.2和8.2)。

b) 确定哪些实体类型是没有其他属性的“键-键实体”。这些实体类型代表着规范化数据模型里的n:m 关系，并且不会被赋值。对于键-键实体链接起来的两个逻辑文件，引用属性(外键)被计数为一个数据元素类型。

c) 确定哪些实体类型是具有其他属性的“键-键实体”。注意这里可能出现两种情况。

1) 附加属性本质上是技术性的(并非用户需求，例如日期/时间截),无需计数为数据元素类型。如果这些附加属性是唯一的数据元素类型，则依据步骤2进行处理。

2)附加属性本质上是功能性的(用户需求),在这种情况下，附加属性按照步骤4进行处理。

d) 检查其余的实体类型，判断它们是否自身就是一个逻辑文件，或者是否和其他一个或多个相关