COSMIC 功能规模度量方法
4.0.2 版

度量手册

(COSMIC 关于 ISO/IEC 19761: 2011 的操作指南)

2017 年 10 月

致 谢

4.0.2版本评审员		
Diana Baklizky TI Metricas Brazil	Jean-Marc Desharnais École de Technologie Supérieure – Université du Québec Canada	Peter Fagg Pentad United Kingdom
Cigdem Gencel Free University of Bozen-Bolzano Italy	Arlan Lesterhuis* The Netherlands	Dylan Ren Measures Technology LLC China
Bruce Reynolds Tecolote Research United States	Hassan Soubra ESTACA France	Charles Symons* United Kingdom
Francisco Valdés Souto Spingere Mexico	Frank Vogelezang Metri The Netherlands	Chris Woodward United Kingdom

* COSMIC 方法 4.0.2 版的编辑

历史版本的 COSMIC 方法评审员在相应的文件内均有记录。

本升级版本中文版贡献者	
翻译组织者	麦哲思科技（北京）有限公司 www.measures.net.cn 电话：400-1780727
初版翻译	郭玲　麦哲思科技高级咨询顾问，COSMIC 讲师
校对	徐丹霞　麦哲思科技高级咨询顾问，CMMI 教员，大规模敏捷教练，COSMIC 讲师

注：对 COSMIC 及本文档的任何疑问或指正之处，请加入 COSMIC 交流 QQ 群——309842452。

COSMIC 度量手册和技术报告的公开版本，包括其它语言的翻译版，可以从门户网站 www.cosmic-sizing.org 上找到。

版本控制

下表汇总了本文档各版本的历史：

日期	评审员	修改/增加
1999-03-31	Serge Oligny	手册首版，用于征求评审员的意见
1999-10-29	COSMIC 核心组	修改版，发布“领域试验”2.0 版前的最终意见
2001-05-01	COSMIC 核心组	为符合 ISO/IEC 14143-1: 1998 进行的修改，以及对 2.1 版度量规则的澄清。
2003-01-31	COSMIC 度量实践委员会	为符合 ISO/IEC FDIS 19761: 2002 进行的修改，以及对 2.2 版本度量规则的进一步澄清。
2007-09-01	COSMIC 度量实践委员会	对度量规则进行修改和进一步的声明之后的 3.0 版，尤其是针对度量策略阶段的修改。方法名称由“COSMIC-FFP”更改为“COSMIC 方法”。2.2 版升级到 3.0 版后，2.2 版度量手册的部分内容分离出来形成其他文档。
2009-05-01	COSMIC 度量实践委员会	3.0 版至 3.0.1 的修订进行了较小的编辑改进和澄清，并使案例更清晰。此版本也吸收了方法更新公报（MUB）3，4 和 5 中提出的更改。
2014-04-01	COSMIC 度量实践委员会等	4.0 版的修订体现了方法更新公报（MUB）6 至 11 的内容，也进行了编辑改进以及增加了术语表。更改详情请参阅附录 E。
2015-04-01	COSMIC 度量实践委会等	4.0.1 版的修订了在 4.0 版中的一些错误和编辑的改进。更改详情请参阅附录 E。
2017-10	COSMIC 度量实践委会	4.0.2 版的修订参考了 MUBs 的第 12、13、14 条并进行了编辑改进。更改详情请参阅附录 E。

前 言

COSMIC 方法为“业务应用”（或 MIS）软件，“实时软件”，“基础设施 ”软件以及一些科学/工程软件提供了一种度量软件功能规模的标准方法。

COSMIC 方法于 2002 年 12 月被 ISO/IEC JTC1 SC7 接受，成为国际标准。最新版本为 ISO/IEC 19761:2011“软件工程—COSMIC—功能规模度量方法”[1]（以下简称 ISO/IEC 19761）。

像 3.0 版一样，ISO/IEC 19761 只包括了基本规范的定义和方法的规则。度量手册的目的不仅是提供这些规则和定义，而且提供进一步的解释和更多案例，来帮助度量者完全理解如何应用这些方法。度量手册为度量者提供了在实践中应用 COSMIC 方法最主要的标准描述。

COSMIC 方法简介

除了“度量手册”以外，“软件度量的 COSMIC 方法介绍”[2]也概括了 4.0/4.01 版的方法。对功能规模度量（简称 FSM）的新手、对其他 FSM 方法熟悉但正考虑改变者，或仅想大致了解 COSMIC 方法的读者，在阅读本手册之前，应该阅读此“介绍”。更多关于 FSM 和 COSMIC 方法的背景知识、相关指南（例如，在不同特殊情景下如何使用该方法等）、案例分析以及研究报告等均能从门户网站 www.cosmic-sizing.com 上找到。

COSMIC 方法 4.0.2 版的主要变更

4.0.2 版对方法的基本原则没有变化。针对定义和规则的所有修改是为了便于理解和提高度量可重复性。最主要的修改主要集中于 3.2 和 3.3 节，并且主要是依据 MUBs(方法更新公告)进行修改的（该公告已经出版）。这些修改如下：

- 修改“触发事件”的定义，明确只有功能用户生成的第一个数据组，是由触发输入移动的。同时，在功能用户的定义中明确一个功能用户（也可能是兴趣对象）可能有“双重角色”（MUB 12）。
- 增加新规则：“识别同一功能处理中的不同数据组（也就是不同兴趣对象）”（MUB 13）[1]。
- 在“兴趣对象”的定义中，“处理和/或存储数据”替换为“向软件输入/输出数据组，或从/向持久存储介质移动数据组” 。这个调整主要为了明确一个兴趣对象只有在它作为被移动的数据组主体时，才可能被识别为兴趣对象。如果一个兴趣对象只是作为数据运算的主体，可能它并不会被识别为兴趣对象（MUB 14）。

附录 E 提供了更详细的关于“度量手册从 4.0 版到 4.0.1 版、从 4.0.1 版到 4.0.2 版”的变更清单。自发布之日起，度量手册 4.0.2 版本将作为该方法的现行标准定义。

4.0.2 版中的主要变化对现有规模度量结果的影响，等等。

自 1999 年度量手册的初稿发布以来，尽管为了成为国际标准以及为了将该方法的所有版本升级到 4.0.2 进行了大量的改进和增加，但 COSMIC 方法最初的基本原则一直未变。

[1] 这些新规则首次在《业务应用软件规模度量指南》1.3 版本（2017 年 5 月）中发布。

根据度量手册 4.0.2 版的原则和规则进行度量得到的功能规模可能不同于较早版本，这只是因为新规则更准确和完整。因此，度量人员对规则的理解较之前版本偏差更小。

为了进一步体现方法的连续性，通过了 3.0/3.0.1/4.0 版基础水平认证考试者将被认为具有 4.0.1 及 4.0.2 版方法基础水平资格。

术语的说明

本手册中使用的术语参见附录 F。本手册使用了标准的 ISO 术语，即：

- “必须（shall）”指某一规则是强制性的；“应该（should）”指某一规则是被推荐的（如果两术语都没有出现，默认为“必须”） 。
- “may” 指“被允许”；“can”指“可以” 。

本手册的主要内容

第一章论述了 COSMIC 方法适用的软件类型。定义了“功能性用户需求”（FUR）术语以及 COSMIC 方法的基本原则。另外，本章节还介绍了 COSMIC 方法的度量过程及度量单位。

第二章描述了度量过程的第一个阶段——度量策略阶段及其主要参数，如度量目的、度量范围及软件块的功能用户。必须在开始度量前定义这些参数，保证度量结果被认可且理解一致。第三章讨论了度量过程的第二个阶段——映射阶段，定义了如何把 FUR 映射到功能处理和数据移动。该阶段的输出物是可以被度量的 FUR 的 COSMIC 模型。第四章描述了度量过程的最终阶段——度量阶段，定义了度量软件块功能性用户需求规模的规则以及累计不同软件块规模的方法。此外，本章节也讨论了如何度量软件变更的规模以及对标准 COSMIC 方法进行“本地化扩展”的可行性。

第五章列出了记录度量过程及结果需要考虑的相关参数。

附录 A 到 E 提供了更详细的关于“原则和规则，度量手册从 4.0 版到 4.0.1 版以及到现行的 4.0.2 版本”的总结。

附录 F 包括了本方法术语的词汇表。

通用软件度量国际联盟（COSMIC）

成立于 1998 年，通用软件度量国际联盟（COSMIC）是一个由全球软件度量专家组成的非盈利自愿性组织。除了受版权许可限制，该组织的所有出版物都是公开发行、免费流通的。更多关于 COSMIC 以及该组织的信息，请登录 COSMIC 的网站 www.cosmic-sizing.org

COSMIC 度量实践委员会将继续对文档进行改进，以确保其更加易于理解。我们强烈建议如果读者及译者在阅读过程中发现任何错误或内容表述不清楚的地方，请与我们联系，反馈流程请参考附录 G。

COSMIC 度量实践委员会

目录

第一章 简介

1.0 章节概要

本章具有四个目的：

- 解释可以应用 COSMIC 方法的软件类型（适用领域）及使用时的限制。
- 定义功能性用户需求(FUR)，即 COSMIC 方法旨在度量的软件功能需求。我们概括性地解释了度量者如何从可获得的软件制品中提取或导出 FUR 以进行功能规模度量。另外，本章节也定义了“非功能需求”（NFR），因为随着项目的推进，最初呈现为非功能的需求会部分或完全转变为可以进行度量的功能性用户需求(FUR)。
- 通过两个模型定义了 COSMIC 方法的基本原则。“软件环境模型”用来描述被度量软件块的特征。“通用软件模型”定义了 COSMIC 中对被度量的功能需求进行建模的主要原则。
- 定义 COSMIC 方法的度量过程及度量原则（与方法的度量单位相关）。

1.1 COSMIC 方法的适用性

COSMIC方法被设计适用于以下领域的软件功能度量：

- 业务应用软件。这类软件通常用于支持业务管理，如银行、保险、会计、人事、采购、配送及制造等。这类软件的特点常常被归结为“数据密集”，因为其需要管理与现实世界中的事件和对象相关的大量数据。
- 实时软件。其任务是监视或控制现实世界中事件的发生。例如电话交换和报文交换软件；嵌入在设备中用于控制机器的软件，如家用电器、电梯、车辆和飞机；用于过程控制和自动数据获取的软件，任何用于物联网的软件，以及计算机操作系统内部的软件。
- 支撑上述软件的平台软件，如可复用的构件及设备驱动程序等等。
- 一些科学/工程软件。

1.2 功能性用户需求

COSMIC 方法包含了一组用来度量给定软件块的功能性用户需求（或简称 FUR）的模型、原则、规则和过程。其结果是一个数字化的“量化数值”（按照 ISO 的定义），代表了根据 COSMIC 方法得到的软件块的功能规模。

ISO [17]定义的功能性用户需求如下：

定义——功能性用户需求（FUR）
用户需求的子集。这些需求以任务和服务的形式描述软件做什么。 注 1：功能性用户需求包含但不限于： • 数据传输（比如输入客户数据；发送控制信号） • 数据变换（比如计算银行利息；导出平均温度） • 数据存储（比如存储客户订单；记录随时间变化的环境温度）

- 数据检索（比如列出当前雇员；检索飞机最新的位置）

属于用户需求但不是功能性用户需求的例子包括但不限于：

- 质量约束（比如可用性、可靠性、效率和可移植性）
- 组织约束（比如操作场所、目标硬件和遵从的标准）
- 环境约束（比如互操作性、保密性、隐私和安全性）
- 实现约束（比如开发语言、交付日期）

注 2：在 COSMIC 文档中，术语“FUR”特指如下的功能性用户需求：

- 从已有的软件制品（如需求文档、设计文档及物理制品等）中提取的；
- 必要时，通过合理的假设来克服现有制品中含糊不清的地方；
- 包含所有 COSMIC 功能规模度量所需要的信息。

除此之外，根据上下文的需要，我们会使用“功能性需求”“实际需求”或“物理制品”等词汇。

需注意的是按此 ISO 定义，COSMIC 方法承认某些类型的需求（如质量和环境约束）在软件项目早期呈现为“非功能需求”，但随着项目的进展这些同样的需求会转变为功能性用户需求。详情请查阅本手册的章节 1.2.3。

COSMIC 方法度量的功能规模被设计成只与被度量软件的 FUR 有关（包括从 NFR 提取初的 FUR），而不依赖于实现 FUR 的需求和限制。“功能”可以不严谨地定义为“软件必须为其用户执行的信息处理”。

1.2.1 如何从软件制品中提取 FUR

在实际的软件开发中，无论是详细描述软件的各种文档，还是能将软件直观表现出来的显示屏幕，很少能够发现有哪个软件制品已经将 FUR 与其他需求清楚地区分开，并且表示成一种适合于直接度量无需任何解释的形式。这就意味着，在把 FUR 映射为 COSMIC“软件模型”前，度量者通常需要从软件的已有制品中提取 FUR。

如图 1.1 所示，在软件开发出来之前，可以从软件工程制品中导出 FUR。这样，软件的功能规模可在计算机系统实现前被度量出来。

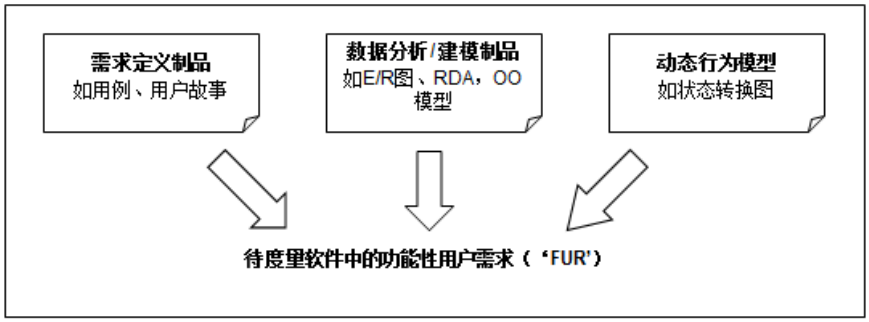


图 1.1 功能性用户需求在软件实现前的来源

注：功能性用户需求在分配到硬件或软件之前就可能产生了。因为 COSMIC 方法旨在度量软件块的 FUR 规模，所以只有分配给软件的 FUR 才被度量。然而，在 FUR 被分配给硬件或软件之前，理论上就可以对需求应用 COSMIC 方法了，无论最终的分配结果是什么。例如，可以

直接用 COSMIC 方法对一个便携式计算器的功能规模进行度量，不需要知道该计算器包含的硬件或软件（如果有的话）。然而，关于 COSMIC 方法可用于度量分配给硬件的需求规模的设想，还需要在实践中进行更多的检验，才能被认为不需要额外规则就完全有效。

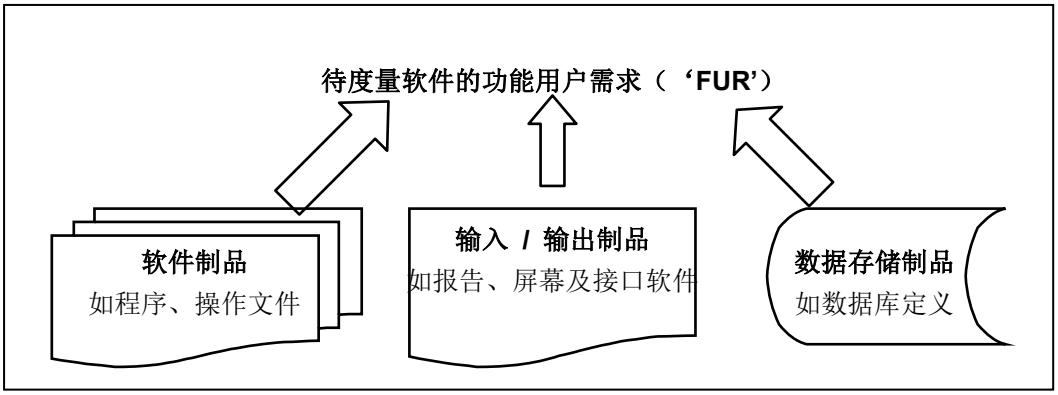


图 1.2 功能性用户需求在软件实现后的来源

在其他情况下，可能需要对已存在的软件进行度量，而这些软件没有或只有很少的架构或设计制品，并且 FUR 也可能没有文档化（例如：对于遗产软件[2]）。这种情况下，仍然可能从计算机系统上的制品中导出 FUR，即使该软件已被实现，见图 1.2。

1.2.2 从软件制品中提取或导出 FUR 的过程

从不同类型软件工程文档中提取 FUR 或者从已安装软件中导出 FUR 的过程及花费的工作量显著不同。这些过程不可能在本手册中进行讨论。根据度量目的，方法假定被度量软件的 FUR 要么存在，要么可从制品中提取或推导出来。

因此，度量手册的内容限定为定义和描述 COSMIC 软件模型的概念（“软件环境模型”及“通用软件模型”——请查阅 1.3 章节），以及在度量给定软件 FUR 的过程中如何应用这些模型。[3]

如果度量者真正理解这两个模型，即使在度量的过程中由于缺少信息或信息模糊不得不做出假设，度量者也是可以从被度量软件中提取 FUR 的。

1.2.3 非功能需求

ISO 给出的功能性用户需求（简称 FUR，参看上文）的标准定义列出了几种在注 1 中的非 FUR 的用户需求，意味着这些需求就是非功能需求（NFR）。

对于一个软件项目而言，NFR 可能非常重要。在极端的情况下，软件密集型系统的需求定义中对 NFR 的描述可能与 FUR 的一样多。但 NFR 与 FUR 的区别并不像 ISO 定义的 FUR 中提到的那么简单。COSMIC 方法可以用来度量某些最初呈现为非功能的需求。首先我们需要定义 NFR[18]：

---

[2] 译者注：随着计算机技术的广泛使用,出现了一些早期开发的、难以维护和进化的、大规模的复杂软件系统,称之为遗产软件系统。

[3] 很多 COSMIC 指南，如：业务应用软件指南[7]，以及实时软件指南[4]，均对数据分析的方法、需求确定的方法、软件制品与 COSMIC 概念的映射提供指导。

定义——（软件的）非功能需求
任何关于一个硬件/软件系统或软件产品的，除软件的功能性用户需求以外的软件部分的需求，包括如何开发、维护以及在操作中如何执行它。非功能需求关注于： ● 软件质量； ● 软件实现的环境或其所服务的环境； ● 开发或维护此软件所用的流程和技术； ● 软件运行所用的技术。 注：随着项目的演化，最初呈现为非功能的系统或软件需求会部分或完全转变为软件的 FUR。

研究[3]表明一些最初呈现为系统 NFR 的需求会随着项目的进展演化为在软件功能中可以实现的混合需求，而另外一部分则是真正的非功能需求或约束。参照图 1.3。对于许多系统质量和约束而言确实如此，如响应时间、易用性和可维护性等。这些在项目初期“藏”在 NFR 中的软件功能，一旦被识别，就可以像对待别的软件功能一样，用 COSMIC 方法度量其规模。软件规模会随着项目的进展不断增长的原因之一就是没有识别出这些“隐藏”的功能规模。

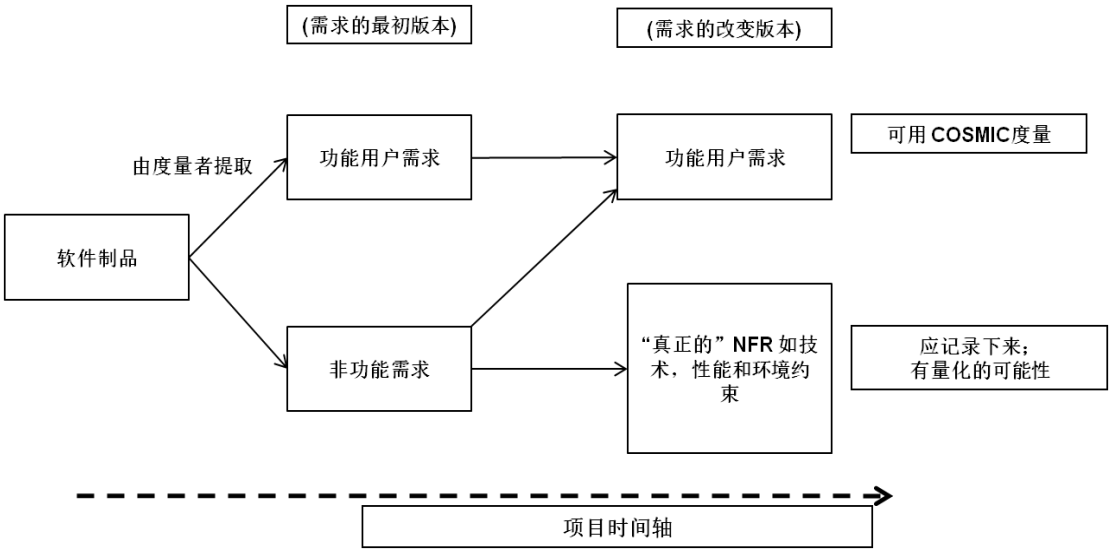


图 1.3 许多最初呈现为 NFR 的需求随着项目的推进会发展成 FUR

业务软件案例：一个新软件系统的需求描述如下“用户必须拥有通过加密来保护文件的选项”。开发此系统的项目正处于估算工作量和成本的阶段，此时有两个选择可以考虑：

- *开发专门的加密软件。为了进行项目估算，度量加密软件 FUR 的规模可能是必要的。*
- *购买已有的商品现货（COTS）程序包。为了进行项目估算，可能只需要度量集成 COTS 程序包所需的软件功能的规模。在估算项目成本时，也应该考虑到程序包的成本以及集成、测试文件加密程序包的工作量。*

实时软件案例：航天系统需求的可靠性或容错性是通过系统的冗余设计及物理备份来实现的。如一个引擎监控的功能被实现在 2 台或更多台独立的嵌入式计算机里。这个功能作为一个有严格时间约束的 NFR 被描述为：“每一台独立的计算机都必须在特定的时间里做出响应。如

果其中任何一台计算机一再地晚于规定时间做出响应，或其显示的结果与其他计算机不一致，该计算机必须被淘汰”（通过该机制描述为一个功能需求）。因此，一个最初呈现为非功能需求的关于容错的需求演变成了可度量的功能性用户需求。这种时间控制机制也可以部分地用软件实现，而且这一功能也可被度量（请查阅 3.2 节中的案例“度量实时软件规模的指南”[4]）。

想阅读更多案例，请查看附录 B。全部这些案例都证明了以下观点：在项目初期，当出现了要度量某个软件规模的需求时，重要的是考虑某些 NFR 能否演变成软件的 FUR，以及这些软件 FUR 的规模是否也应该被度量。

1.3 COSMIC 方法的基本原则

COSMIC 方法基于基本的软件工程原则，这些原则总结在两个模型中。

在丈量房子的时候，房子的大小可能有很多种尺寸，这取决于你想度量什么，与此类似，在度量软件块的规模时，即使使用相同的度量单位，软件的规模也能通过多种方法来度量。“软件环境模型”里的原则帮助度量者定义被度量的软件及规模度量元。这保证了未来的使用者能正确理解并一致解读度量的结果。

“通用软件模型”里的原则定义了待度量软件的 FUR 如何被建模，以供度量。

在手册此部分介绍这两个模型主要是为了展示 COSMIC 方法本质上的简单性。在手册后面的内容中，我们也会提及这两个模型。但是，对于度量新手而言，不要期望读了这两个模型就能准确度量。要想把模型应用到特定的度量场景中，度量者需结合手册中提到的各种概念定义、原则、规则、解释以及案例。

注：以下 1.3.1 和 1.3.2 节中对首次用到的术语采用粗体表示该术语是 COSMIC 方法特有的术语。正式的定义参见度量手册附录 F 中的术语表。想进一步了解原则的详细内容，可参见各原则所对应的章节内容。

1.3.1 COSMIC 软件环境模型

原则——COSMIC 软件环境模型
a) 软件被硬件所界定。
b) 软件通常结构化为多**层**。（2.2.2）
c) 一层可包含一个或多个单独的**“对等”**软件块。（2.2.2）
d) 任何待度量的软件应由其**度量范围**定义，并完全限定在一个单一的层中。（2.2）
e) 待度量软件块的范围依赖于度量**目的。**（2.1）
f) 可以从待度量软件的 FUR 中识别该软件的**功能用户**，这些功能用户分别作为数据的发送者和/或接受者。（2.3）
g) 软件的功能性需求可以在不同的**颗粒度级别**上表达。（2.4）
h) 精确的 COSMIC 软件规模度量需要该软件块的 FUR 达到能够识别出**功能处理**和子处理的颗粒度级别。（2.4.3）
i) 如果一个软件块的功能性需求只有在较高的颗粒度级别上的描述，则可以采用近似的 COSMIC 方法度量软件块的规模，并将其缩放至功能处理及子处理的颗

粒度级别。（2.4.3）

1.3.2 通用软件模型

根据软件环境模型识别和定义了待度量软件的 FUR 后，我们可应用通用软件模型来从 FUR 中识别出需要度量的功能构件。对于所有可用本方法度量的软件，通用软件模型假定下面的通用原则均成立。

原则——COSMIC 通用软件模型

a) 软件块跨越**边界**与功能用户交互、并与边界内的**持久存储介质**进行交互。（2.3）

b) 被度量软件块的 FUR 能够被映射到唯一的一组功能处理。（3.2）

c) 每个功能处理由一系列子处理组成。（3.2）

d) 一个子处理可以是一个**数据移动**或者是一个**数据运算。**（3.2）

e) 一个数据移动仅移动单个**数据组。**（3.3）

f) 有四类数据移动：**输入，输出，写和读**。（3.5）

- 输入从功能用户移动一个数据组到功能处理内。
- 输出从功能处理中移出一个数据组到功能用户。
- 写从功能处理移动一个数据组到持久存储介质。
- 读从持久存储介质移动一个数据组到功能处理。

g) 一个数据组由唯一的一组**数据属性**构成，描述了一个单一的**兴趣对象**。（3.3）

h) 功能处理被输入数据移动所触发。功能用户为响应**触发事件**而产生了触发输入，触发输入移动的数据组由一个响应**触发事件**的功能用户生成。（3.2）

i) 一个功能处理的规模等于其数据移动的总数。

j) 一个功能处理包括至少一个触发输入数据移动，以及一个写或输出数据移动，即一个功能处理应该包含至少两个数据移动。一个功能处理中数据移动的数量没有上限。（3.3）

k) 作为对度量目的的一种近似处理，数据运算子处理不单独度量。任何数据运算的功能被假定已经计算在相关的数据移动内了。（3.5.6）

注：COSMIC 通用软件模型，正如其名字所示，是一个逻辑“模型”。该模型将软件处理数据的各个单元模型化，以便进行功能规模度量。模型并非按照物理顺序描述软件执行步骤或软件的技术实现过程。

1.3.3 类型与实例[4]

为了更好的理解本度量手册，读者需要区分“类型”和“实例”这两个概念。

- 一般来说，事物的类型是一个抽象的类别，该类别的所有事物都有一些共同特点。（“类型”的同义词是“分类”或“种类”）；在功能规模度量的概念中特指，“事物”如果有同样的 FUR 则称其为同一类型。

[4] 译者注：经过再三斟酌，occurrence 在本手册中被翻译成了“实例”，而没有翻译为“出现、出场、发生”，目的是为了更加符合软件开发的术语习惯。尽管在手册中也提到了：occurrence 是 instance 的近义词。

- 事物的一个“实例”是该事物出现在实际场景中，例如现实世界的一个场景，或当一个事件发生时，或当人或计算机执行一个处理时。(“occurrence”的同义词是“instance”。一个“实例”产生于当事物的某一“类型”的特征被赋予一个实际值时，在面向对象的开发中被称为“实例化”——某个实例的产生。)

所有的功能规模度量方法都会定义：对于一个给定的功能需求，度量者为了度量其功能规模而需要识别的“事物的类型”。

在 COSMIC 方法中，“事物的类型”包括软件环境模型中的功能用户类型和功能处理类型，以及通用软件模型中的所有加粗的概念，例如数据移动类型，兴趣对象类型等。

为了便于阅读，我们一般会省略这些概念中的“类型”两个字。

软件环境模型的案例

案例 1：一个支持 100 名员工的呼叫中心系统，负责接听处理客户问题。此系统的环境模型会显示有一个功能用户类型：“呼叫中心的员工”，此类型有 100 个实例。

案例 2：一个数字无线电的嵌入式软件会将其输出发送给一对立体声扬声器。该软件向这两个扬声器分别发送相同类型的信号，他们都用同样的方式把接受的电信号转化为声音。此软件的环境模型会显示一个名为“扬声器”的功能用户类型，此类型有 2 个实例。

通用软件模型的案例

案例 3：假设一个功能处理（类型）用于新用户数据的输入和确认。当一个人类功能用户注册新用户数据时，这个功能处理会被执行，即它会发生一次。而在它执行过程中，通过搜索数据库以检查客户是否已存在来确认输入数据的读(类型)数据移动可能发生一次或多次(取决于数据库的设计）。

案例 4：假设一个功能处理（类型）用于每十秒控制烤箱温度一次。每十秒这个功能处理会被执行一次，即它会发生一次。在它执行期间，任意一个循环内，打开或关闭加热器的输出（类型）数据移动可能会被执行或不执行，即，在每一次循环中可能发生也可能不发生，取决于是否需要开启或关闭加热器，还是保持它当前的状态。

注意：任何功能用户，输入（类型）、输出（类型）、读（类型）、或写（类型）数据移动的实例个数与度量其 COSMIC 功能规模不相关。但是，在某些情况下，确定数据移动实例的频率可以帮助区分不同的数据移动类型。（见 3.3.2 章节的规则案例）。

1.4 COSMIC 度量过程及度量单位

COSMIC 度量过程由三个阶段组成：

- 度量策略阶段，此阶段定义了度量目的与范围。在该阶段应用软件环境模型，以便明确地定义待度量的软件及需要的度量方法。（第二章）
- 映射阶段，在该阶段对于待度量软件的 FUR 应用通用软件模型，以生成可度量软件的 COSMIC 模型。（第三章）
- 度量阶段，在该阶段度量实际的规模。（第四章）

第五章讲述了度量结果的记录规则。

COSMIC 方法三个阶段的关系，如图 1.4 所示。

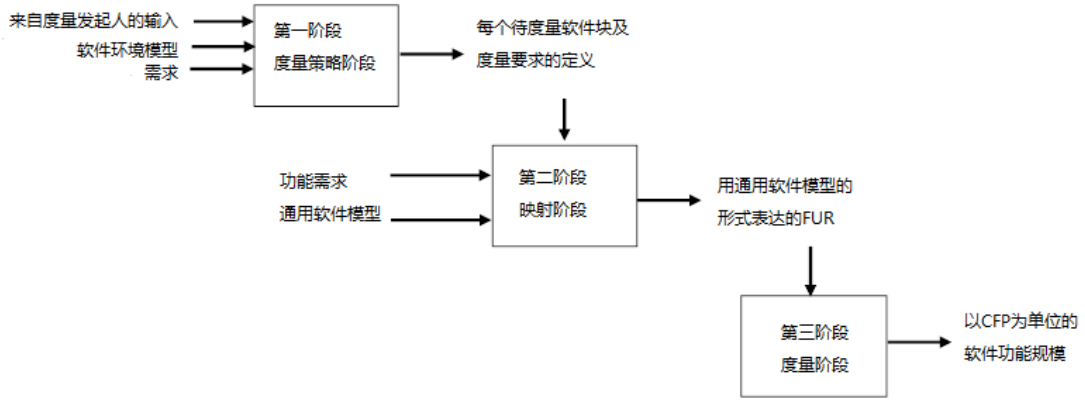


图 1.4 COSMIC 方法度量过程

COSMIC 度量单位（CFP）及度量原则的定义如下：

定义——COSMIC 度量单位
1CFP(COSMIC 功能点)，表示一个数据移动的规模。

原则——COSMIC 度量原则
a) 功能处理的规模等于其数据移动的个数。 b) 给定范围的软件块的功能规模等于其功能处理的规模总和。

软件需求变更的规模按如下方法度量：

- 需求变更影响的任何数据移动（即必须增加、修改或删除的数据移动）的规模约定为按照 1 个 CFP 计算。
- 软件块需求变更的规模等于被需求变更影响的数据移动的个数。
- 软件块变更的最小规模为 1CFP。

更多关于度量和度量汇总的规则和指南，请查阅本手册的第 4.1 到 4.4 节。

1.5 COSMIC 方法适用性的限制

请查阅 4.5 节了解方法可能存在的限制及如何通过本地化扩展方法来克服这些限制。

第二章 度量策略阶段

2.0 章节概要

本章讨论在实际开始度量之前，在第一阶段“度量策略”阶段必须考虑的关键参数。包括（斜体部分）：

- 度量*目的*，即度量结果的用途。目的决定了度量的其他参数。
- 待度量软件的*整体范围*及软件各个部分的*度量范围*（当软件包含多个软件块，且这些软件块的规模应该分别度量时，例如分布式软件系统的构件）。我们也需要确定每个待度量软件块所处的层，并可能需要确定被度量软件块的分解层级。
- 每个待度量软件块所在的软件结构的*层*。每个待度量软件块的*分解等级*，如：是整个应用还是主要组件层级，还是复用组件层级（来自某一面向服务架构）等。
- 每个待度量软件块的*功能用户*。功能用户向待度量软件发送数据或希望从软件接收数据，他们可能是人、硬件设备或其它软件块。
- 可获取到的描述待度量软件的工作产品的*颗粒度级别*。比如，是否所有的需求描述都是同一详细程度？根据现有的材料，我们是否可以进行精确的 COSMIC 度量或是只能近似度量？确定这些参数有助于解答以下问题：“应该度量哪个模块的规模？”或者“我们想要多么准确的度量结果？”等等。

记录这些参数有助于度量结果的未来使用者决定如何对其进行解读。

值得注意的是，这些参数及相关概念并非 COSMIC 功能规模度量（FSM）方法所特有，它们是所有 FSM 方法所共通的。其他 FSM 方法可能不区分功能用户的类型，可能不讨论颗粒度级别等等。只是因为 COSMIC 方法具有更广泛的适用性和灵活性，才要求比其他 FSM 方法更仔细地考虑这些参数。

记录每次度量结果时，记录来自该度量策略阶段的数据（在 5.2 节列出）是非常重要的。如果这些参数缺乏一致地定义和记录，将导致度量结果无法被可信地解读和比较，也无法可信地作为估算项目工作量的过程输入。

如下图 2.0 所示，本章的各小节给出了关于每个关键参数的正式定义、原则、规则和一些实例，以帮助度量者完成确定度量策略过程。

每一小节给出了关键参数为何如此重要的背景说明，采用类比的方法展现了为何在其他度量领域必然要考虑的参数，在软件功能规模度量领域也必须考虑类似的参数。

从图 2.0 可以看到，确定度量策略的参数可能需要几次迭代。

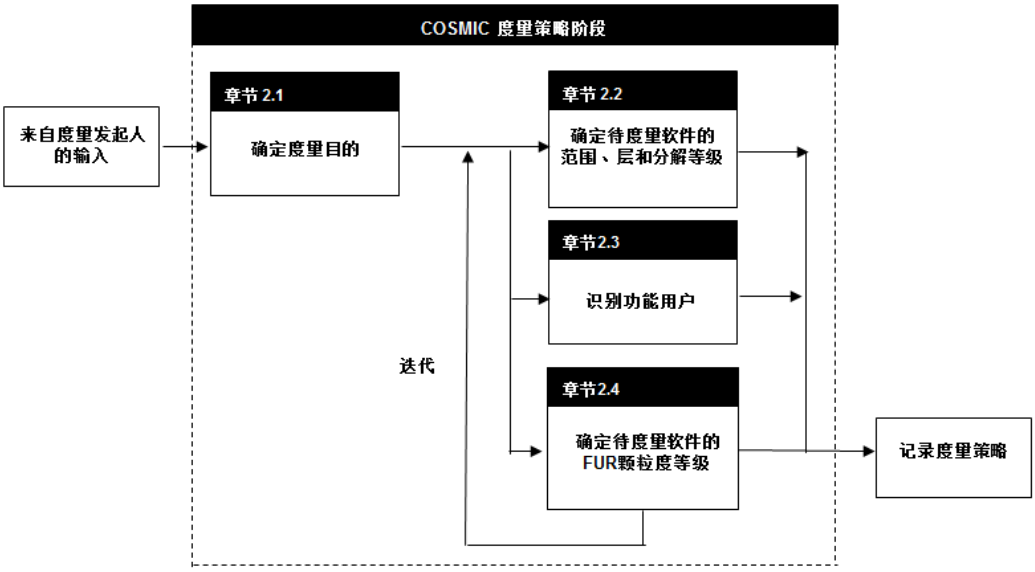


图 2.0 确定度量策略的过程

度量策略模式

作为确定度量策略的辅助，“度量策略模式”[5]指南为不同类型的软件描述了一组度量软件规模的标准参数集，将其称之为“度量策略模式”（简称“度量模式”）。

定义——度量（策略）模式
在度量特定软件功能领域的软件块的规模时，可以使用的一个标准模板。此模板定义了可能与软件交互的功能用户类型、软件的分解层级及软件可能会处理到的数据移动类型。

一致地使用相同的度量模式可以帮助度量者确保相同目的的度量能一致地执行，可以有把握地与其他使用同样模式的度量进行对比，也有助于所有日后的使用者对度量的正确解读。使用标准模式的另一个好处是大大减少了确定度量策略参数的工作量。但是，在使用标准模式之前，我们强烈建议度量者学习并掌握 COSMIC 方法，特别是度量策略参数。

2.1 定义度量目的

“目的”这一术语采用其通常的英语含义。

定义——度量目的
为什么需要度量和度量结果用途的描述。

2.1.1 度量目的的类比

就像度量房屋的面积有很多的理由一样，度量软件功能规模也有很多的理由。在以下的两个场景中，理由不同，得到的规模大小也不同。从这个类比看，房屋的度量面积会根据以下因素而截然不同：

- 度量的原因及时机（例如，依赖于是要度量客户的概略说明以制定预算、还是度量设计方案以做出准确的成本估算、或者是度量完工后的实际面积以计划需要的地板覆盖物）。

- 被度量的制品（如设计方案或房屋本身）。

然而，需要注意的是，在所有度量活动中使用的都是相同的度量原则及度量单位。与度量房屋同理，软件块的度量规模会根据以下因素而截然不同：

- 度量的原因及时机（例如，依赖于是在软件开发前度量以进行估算、或是在开发过程中度量以跟踪范围的蔓延、或是在安装后度量开发者的绩效）。
- 被度量的产物（如需求描述或软件本身）。

但是，像上文提到的类比一样，在所有度量中使用的都是相同的度量原则及度量单位。

显然，何时度量（开发前、开发中或开发后）、度量什么（项目交付的所有软件或去除复用的软件）以及从哪个制品提取待度量的 FUR（需求描述或已安装的软件），所有这些都是由度量者根据度量目的决定的。

案例：下面是典型的度量目的。

- *随着 FUR 的演化度量其规模，作为开发工作量估算过程的输入。（见[6]，关于对项目早期近似规模度量的指导）*
- *度量 FUR 在其最初被认可之后的变更规模，以便管理项目的“范围蔓延”。*
- *度量所交付软件的 FUR 规模，作为度量开发组织绩效的输入数据。*
- *度量整个交付软件的 FUR 规模，以及新开发软件的 FUR 规模，来获得功能复用的度量。*
- *度量现有软件的 FUR 规模，作为度量负责软件维护和支持人员的绩效的输入。*
- *度量现有软件系统（FUR）变更部分的规模，以度量一个维护型项目团队的规模产出。*
- *度量提供给人类功能用户的、必须开发的完整软件功能子集的规模。*

2.1.2 度量目的重要性

度量目的帮助度量者确定：

- 度量的范围和度量所需要的制品。
- 功能用户（如将在 2.3 节中介绍的，功能规模的变化依赖于谁或什么被定义为功能用户）。
- 在项目生命周期中进行度量的时间点。
- 度量的精确性，以及是否应该使用 COSMIC 方法，或者应该使用 COSMIC 方法的近似版本（例如，在项目生命周期早期 FUR 被完整定义之前）。

后两者将决定被度量 FUR 的颗粒度级别。

2.2 定义度量范围

定义——度量范围
在一次具体的功能规模度量活动中所包括的功能性用户需求的集合。 注：（仅针对 COSMIC 方法）应该区分“总体范围”与总体范围内的各软件块的“范围”，“总体范围”是指根据目的应度量的所有软件，而各软件块的规模应该分别度量。本手册中术语“范围”（或“度量范围”）将关联到规模必须单独度量的一个软件块。

规则——度量范围
a) 任何待度量软件块的范围必须从度量目的中导出。 b) 任何一次度量范围不能延伸超过被度量软件所在的层。

关于总体范围和度量范围的案例，请参阅下一节。

2.2.1 从度量目的导出度量范围

定义了总体范围的软件可根据其度量目的不同，使用不同的办法划分为具有不同度量范围的多个软件块。假定总体范围被定义为“X 组织的应用程序产品包”或“Y 项目将要交付的所有软件块”，可根据以下因素进行划分：

- 不同层的软件（根据上文提到的规则 b）。
- 不同的组织职责，如按照客户群或子项目团队。
- 根据性能度量、工作量估算或者软件合同等不同的目的需要区分不同的交付物。

正是由于如下的原因，需要划分软件块的度量范围：

- 采用不同的技术路线进行开发，如硬件平台、编程语言等。
- 采用不同的运行模式，如联机模式与批处理模式[5]。
- 开发用而非交付用的软件（包括应用封装软件或其他复用软件）。
- 在不同分解层次上的软件，例如，一个完整的应用程序或一个主要/次要构件，如可复用的对象。
- 是主要的交付物，而非使用一次、用后即弃的软件，例如用于数据转换，后者可能不值得花费工作量去度量。
- 敏捷过程中单次迭代交付的软件。
- 是新开发的，而非对软件的增强。

以及以上因素的任意组合。

总之，度量目的必须总是用来确定：(a)总体范围中应该包含和排除在外的软件；（b）将包含的软件分解成各个独立块的方法，每个块有自己的范围，并且被分别度量。

实践中，范围的描述需要详细描述而不是概括性的，如项目团队 A 开发的产出，或者应用程序 B，或者企业 C 的产品包。为了清晰起见，范围描述也需要指明哪些是排除在外的。

商业案例：图 2.1 显示了一个项目团队交付的所有单独的软件块（即总体范围）：

- *一个已实现的软件包的客户端和服务器端构件。*
- *服务器构件的新开发程序包和已有应用程序间的接口程序。*
- *软件包所需的一次性数据转换程序，用于按照程序包的需要将现有的数据转换为新的格式。此程序使用了一些由项目组开发的可复用对象。*
- *为新硬件而开发的设备驱动软件，客户端构件的程序包将在其上运行。*

（每个定义了度量范围的单独的软件块用一个矩形框表示。）

[5] 译者注：批处理模式是指在计算机上无须人工干预而执行系列程序的作业。批处理任务无须人工交互，所有的输入数据预先设置于程序或命令行参数中。

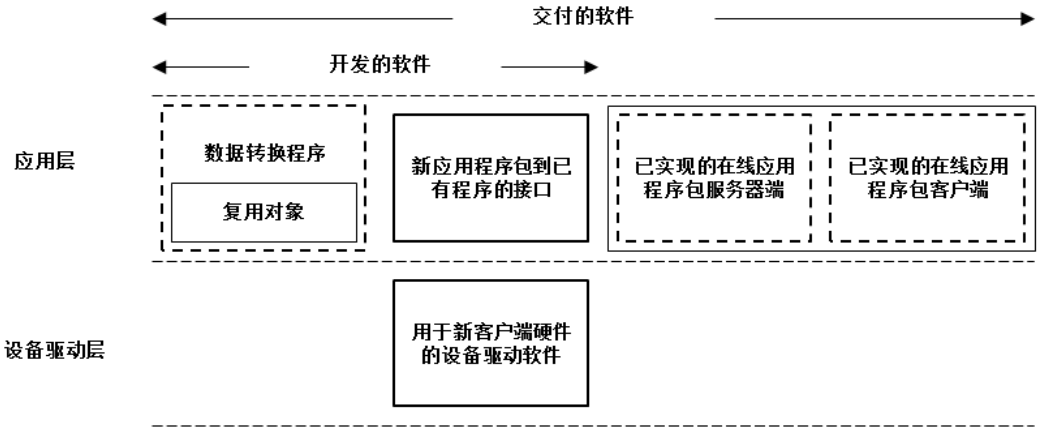


图 2.1 一个软件项目交付物的总体范围以及单独的度量范围

图中显示了交付软件是由两部分组成的：新开发的软件以及项目团队已实现的软件。目的是要度量将要加入组织级软件产品包的各个软件块的FUR，需要把软件包看作一个整体，即忽略客户端-服务器的构件结构。

已实现的程序包规模加上接口程序的规模，用于更新组织的应用程序产品包的总规模。由于数据转换程序只被使用一次就废弃，其规模无需度量。但每一个复用对象的规模以及新设备驱动的规模均记录在组织基础设施软件清单中。它们也是单独归类的。

由于交付物的多样性，使得度量项目团队的绩效时，把所有交付软件的规模加起来是没有意义的。交付不同软件块的团队的绩效应被分别度量。

需要注意的是，通常我们只需要度量软件 FUR 中需要实现的那部分，而不是度量软件包的整个FUR。只有对于软件包供应商，度量后者才有意义。

2.2.2 层

由于被度量软件块的范围必须被限制在一个软件层里，因此定义范围时可能需要度量者首先确定软件体系结构有哪些层次。本节我们将定义和讨论 COSMIC 方法中使用的软件“层”的概念。我们需要这些定义和规则的理由如下：

- 度量者可能会面对一些在“遗产”环境里的软件，这些软件在没有遵循任何基础体系结构设计的情况下演化了很多年（这种软件有所谓的“意大利面条结构”[6]之称）。因此，需要指导度量者如何根据 COSMIC 术语区分层。
- “层”和“分层体系结构”的表述在软件业的使用并不一致。如果度量者必须度量某个“分层体系结构”中的软件，核对该系统结构中“层”的定义与 COSMIC 方法是否兼容是明智的。为做到这一点，度量者应该建立此“分层体系结构”中具体的体系结构对象与本手册的“层”概念之间的等价关系。

可以根据以下的定义和原则来识别层：

[6] 译者摘自维基百科：面条式代码（Spaghetti code）是软件工程中反面模式的一种，是指一个代码的控制结构复杂、混乱而难以理解，尤其是用了很多 GOTO、例外、线程、或其他无组织的分歧架构。其命名的原因是因为程序的流向就像一盘面一样的扭曲纠结。

定义——层
一个软件系统体系结构的功能划分。

在一个已定义的软件体系结构里，每一层必须服从下面的原则：

原则——层
a) 一层中的软件会根据已定义的准则提供一组内聚的服务，而且其他层里的软件无需了解这些服务是如何实现的也能利用它们。 b) 任意两层中软件之间的关系可用“通信规则”来定义，有两种情况： • “分层的”，如 A 层的软件可以使用 B 层软件提供的服务，但反之则不成立（这里层次结构的关系有可能是自上而下或自下而上），或： • “双向的”，如 A 层的软件可以使用 B 层的软件，反之亦然。 c) 一层的软件通过相应的功能处理与另一层的软件交换数据组。 d) 一层的软件没有必要使用其他层的软件提供的所有功能服务。 e) 在一个已定义的软件体系结构中属于同一层的软件，根据另外一个不同的体系架构可能被划分为其他的层。

一次度量可能会涉及两个或两个以上的对等软件块，定义如下：

定义——对等软件块
若两个软件块处于同一层中，则它们是相互对等的。

案例：图 2.1 中处于应用层的软件块都是相互对等的。

如果待度量软件的整体范围涉及到多个层，则度量人员应按照如下步骤进行：

- 如果待度量软件存在于一个已建立的层次结构中，此结构又可以映射到如上定义的 COSMIC 分层原则中，那么该结构就可以用来识别层以满足度量目的。
- 但是，如果度量目的要求度量某些并非按照 COSMIC 分层原则结构化的软件，度量者应该尝试使用上文定义的原则将软件分解为层。按照惯例，提供服务以被其他层软件使用的基础设施软件程序包，如数据库管理系统、操作系统或设备驱动等都各自处于单独的层内。

一般在软件体系结构中，“顶”层，即在整个分层结构中不从属于任何其他层的层，通常被称作“应用程序”层。这个应用层里的软件依赖所有其他层的软件提供的服务来正常运行。处于此“顶”层的软件可以再进行分层，例如由用户界面、业务规则以及数据服务构件所组成的“三层结构”（参见下文的例 5）。

一旦被识别出来，每个层都可以使用相应的标签登记在通用软件模型矩阵（附录 A）中。

业务软件案例 1：图 2.2 给出了支持业务应用软件的一个典型分层软件体系结构（使用此处定义的“层”术语）的物理结构：

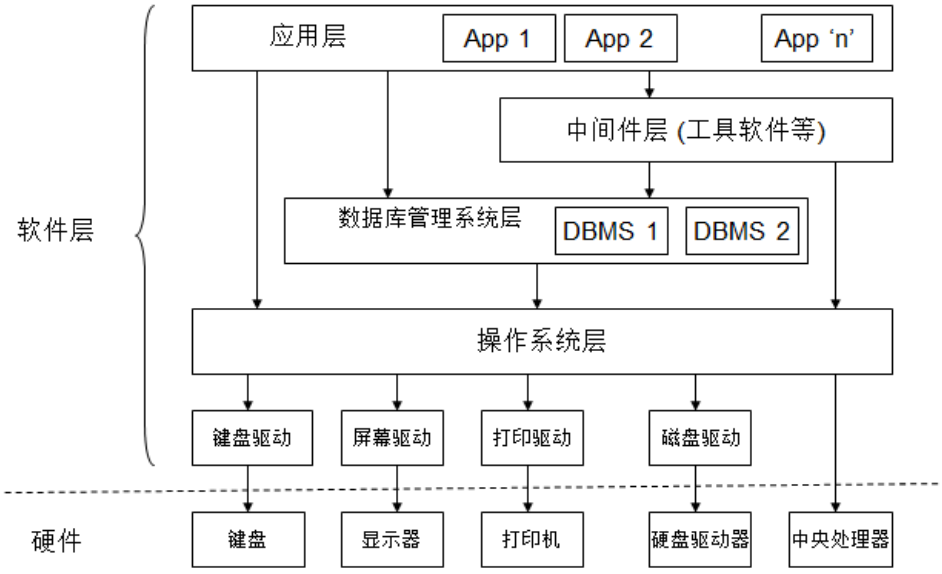


图 2.2 商业/MIS 计算机系统的典型分层软件体系结构

实时软件案例 2：图 2.3 给出了支持嵌入式实时软件的一个典型分层软件体系结构（使用此处定义的“层”术语）的物理结构。（注：简单的单任务实时嵌入式软件可能不需要实时操作系统。）

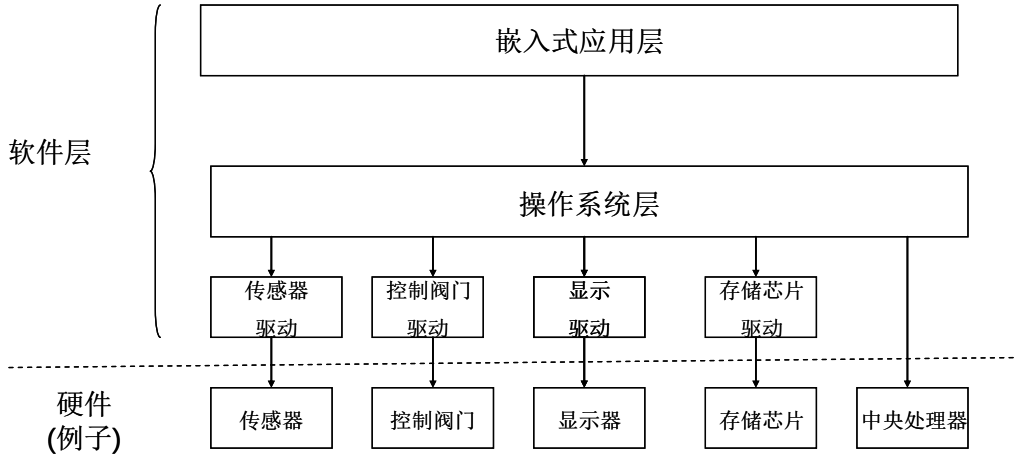


图 2.3 实时嵌入式软件计算机系统的典型分层体系结构

实时软件案例 3：通信系统的 ISO 7 层（OSI）模型[7]。此模型定义了一种分层体系结构：信息接收层软件的层次通信规则与信息发送层的规则是相反的。

实时软件案例 4：在汽车工业中，“汽车开放系统架构”（AUTOSAR）[8][19]展现了层与层之间所有不同类型的通信规则，也是使用层的原理描述的。

[7] 译者注：开放式系统互联参考模型（Open System Interconnection Reference Model，ISO/IEC 7498-1），简称为 OSI 模型（OSI model），由国际标准化组织（ISO）提出的一种概念模型，试图使各种计算机在世界范围内互连为网络的标准框架。OSI 将计算机网络体系结构由上至下分为以下七层：应用层(Application Layer)、表示层(Presentation Layer)、会话层(Session Layer)、传输层(Transport Layer)、网络层(Network Layer)、数据链路层(Data link Layer)和物理层(Physical Layer)。

从不同“视角”看待软件的体系结构，会呈现为不同的层次结构。

商业案例 5：如图 2.4，应用程序 A 处于一个分层软件体系结构中，从不同“视角”看待此体系结构，会得到三种不同的层次结构 a)， b) 和 c)。

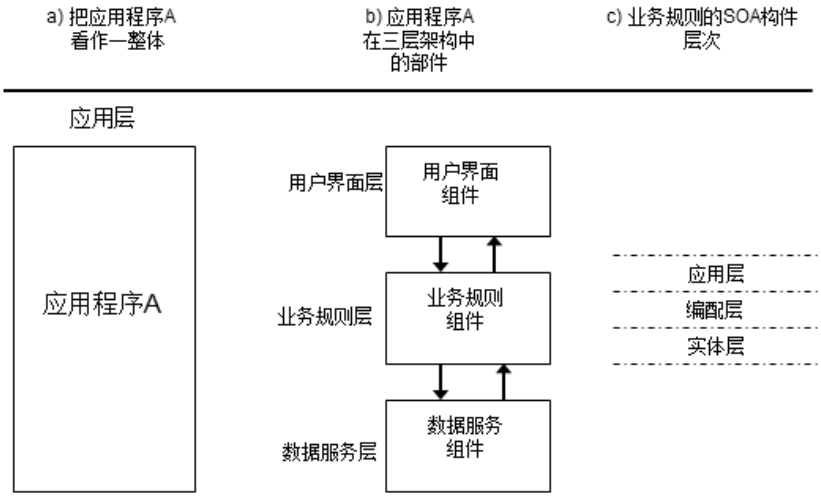


图 2.4 一个应用程序分层的三种视角[9]

目的 1：如视角 a 所示，把应用程序 A 看作一个整体，度量其功能规模。度量范围是整个程序 A，其完全处于一个“应用”层内。

目的 2：应用程序 A 根据“三层”架构构建而成，包括用户界面、业务规则以及数据服务构件。目的 2 就是为了分别度量视角 b) 中的 3 个构件。每个构件都处于三层结构中自己的层内，各构件的度量范围必须分开定义。

目的 3：应用程序的业务规则构件使用了面向服务的体系结构（SOA）的可复用构件进行构建，它有自己的层次结构。如视角 C 所示，目的 3 是要度量业务规则构件的 SOA 构件。每个 SOA 构件都处于 SOA 结构中的一层内，各构件的度量范围必须分开定义。（注意：SOA 术语在自己的体系结构中也使用“应用层”一词。）

2.2.3 分解层级

软件块的“分解层级”定义如下：

定义——分解层级

[8]译者摘自百度百科：AUTOSAR 是 AUTomotive Open System Architecture（汽车开放系统架构）的首字母缩写，是一家致力于制定汽车电子软件标准的联盟。AUTOSAR 是由全球汽车制造商、部件供应商及其他电子、半导体和软件系统公司联合建立，各成员保持开发合作伙伴关系。自 2003 年起，各伙伴公司携手合作，致力于为汽车工业开发一个开放的、标准化的软件架构。AUTOSAR 这个架构有利于车辆电子系统软件的交换与更新，并为高效管理愈来愈复杂的车辆电子、软件系统提供了一个基础。此外，AUTOSAR 在确保产品及服务质量的同时，提高了成本效率。

[9]译者摘自维基百科：编配（英语：orchestration）描述复杂计算机系统、中间件(middleware)和业务的自动化的安排、协调和管理。

将软件块分解为构件而形成的任意级别（例如，称为“1 级”），然后将构件分解为子构件（“2 级”），再将子构件分解为子子构件（“3 级”），等等。 注 1：不要和描述需求详细程度等级的“颗粒度级别”混淆。 注 2：软件块中构件的规模度量结果只可以和处于同一分解层级的构件直接比较。 注 3：一个软件块不同的分解等级可能对应不同的软件层次。例如图 2.4。并且，软件可以分解为不同“层级”，与软件是否按照分层架构模式进行设计无关。

上述定义中的注释 2 很重要，因为在没有考虑 4.3.1 章节的累加规则时，不同分解层级的软件块的规模不能被简单地累加。进一步讲，作为本注释的应用结果，如果所有的软件块都处在相同的分解层级上，开发不同软件块的项目性能（如生产率=规模/工作量）是可以安全地进行比较的。

案例：“度量策略模式”指南[5]中归纳了三个标准的分解层级：“应用整体”“主要构件”和“次要构件”。请查阅本度量手册 2.2.2 节的业务软件案例 5，该例中的图 2.4 展示了这三种级别。

2.2.4 定义度量范围的小结

确定一个度量范围不仅仅是简单的决定哪些功能需要被包括在度量中，这个决定也可能包括考虑软件所在的层以及软件将在哪个分解层级上进行度量，所有这些都依赖于度量目的。

2.3 识别功能用户及持久存储介质

实际上，“用户”[10]被定义为“任何与被度量软件交互的事物”。对于 COSMIC 方法的需求而言，这个定义太宽泛了。对于 COSMIC 方法，用户的选择取决于必须度量的功能“用户”需求以及度量目的。这个（类）用户称为“功能用户”，定义如下：

定义——功能用户
一个（类）用户是软件块的功能性用户需求中软件所处理数据的发送者或预期的接收者。

功能用户的类型通常取决于软件所属的领域。

- 业务应用类软件，功能用户通常为人类，以及其他与软件交互的对等应用程序。
- 实时类软件，功能用户通常是直接与软件交互的硬件设备，以及其他与实时软件交互的对等软件。

注：强烈建议**不要**既通过人类视角又通过硬件设备视角来度量软件的功能规模。度量的结果会非常难以理解、区分。

注意所有“用户”的集合，即 “任何与软件交互的事物”，则必须包括操作系统。但是任何应用软件的 FUR 不会把操作系统作为用户。操作系统的需求可能适用于所有软件，并且通常

[10] 借鉴自 ISO/IEC 141431/1:2007 的术语定义

由编译器或解析器来处理，该部分对于实际的功能用户来说是不可见的，因此不包含在 FUR 中。在实际度量功能规模时，通常不会把操作系统作为功能用户来考虑。

（然而，如果度量目的是度量操作系统这部分组件，如设备驱动，那么调用驱动的操作系统将会作为功能用户。）

规则——功能用户
a) 待度量软件块的功能用户应取决于度量目的。 b) 当软件块的度量目的与开发或者修改软件块的工作量相关时，根据其 FUR 的要求，功能用户应该是与所有新的或者修改的功能进行交互的数据不同类型的发送者和/或数据预期接受者。 注：FUR 中可能指定功能用户存在多个实例需要单独识别。然而，如果每个实例都属于相同的 FUR，它们就是同一类功能用户。

说明规则 b)的业务软件案例 1：一个订单系统有很多员工（人类功能用户）负责维护订单数据。在所有输入的数据组中增加员工 ID 一项。应识别为一个员工功能用户类型，因为订单系统 FUR 对于所有员工是相同的。

说明规则 b)的实时软件案例 2：汽车的每个轮子都有一个传感器用于监测其轮胎压力。每隔一段时间，一个功能处理必须获得四个轮胎的压力值。如果压力过小或过大-安全压力的范围存储在软件中-软件将在仪表盘的轮胎状态显示屏上显示有压力问题的轮胎。。功能用户是四个传感器和显示屏由于四个传感器属于相同类型，只要识别一个“传感器”功能用户类型以及一个显示器功能用户类型。

注意：四个轮胎的压力传感器的工作方式是一致的；它们发送的数据组类型是相同的，且这些数据的处理过程是相同的。显然，软件必须能够区分四个轮胎，（如，根据它们反馈轮胎压力的顺序，或者根据它们发送轮胎压力时携带的 ID（如 1-4）以便在显示屏上显示对应的轮胎压力。但是处理每个轮胎数据的 FUR 是相同的，所以只存在一个功能用户类型：轮胎。

2.3.1 功能规模可能会随功能用户的不同而变化

功能用户并不总是很容易识别出来的。“事物”的不同使用者会看到“事物”的不同功能，因此会度量其不同的规模。对于软件而言，不同类型的功能用户需要不同的功能（通过他们的 FUR），因此功能规模会随着选择的功能用户的不同而变化。

业务软件案例 1：一个软件系统的功能是维护基本的人力信息，人力资源部门的所有员工均可查阅该信息，而对于敏感的薪资信息只有其中部分员工可查阅。因此，该软件有两个功能用户类型。该软件的度量目的应该定义度量范围是所有人员都可以访问的全部功能，还是仅限于其中一类员工可以访问的功能。

实时软件案例 2：一个复印机嵌入式软件。不考虑操作系统为功能用户。软件的功能用户可以以两种方式确定。可以定义为（a）想要进行复印的人类用户，或者（b）复印机的硬件设备，如控制按钮，向人类用户显示信息的屏幕，送纸系统，卡纸传感器，油墨控制器，指示灯等与软件直接交互的硬件。这两种功能用户类型，人类或者硬件设备能“看到”不同的功能。比如人类用户，只接触到复印机的部分功能。

*开发复印机驱动程序的嵌入式软件开发人员需要把硬件设备作为其功能用户。而市场人员为了比较产品的价格/性能[11]，可能会发现度量人类用户能看到的那部分复印机功能更有用，以便与竞争对手的复印机功能进行比较。**不要**将二者混淆：同时以人类用户及硬件设备的角度度量规模将会非常难以理解。*

确定了功能用户之后，可以直接识别边界。边界处于待度量软件块和其功能用户之间。我们忽略其他介于两者之间的硬件或软件[12]。

定义——边界
被度量软件和它的功能用户之间的一个概念性接口。 注意：从以上定义可以引申出，在同一层或不同层间的任何两个有数据交换的软件块之间，存在一个边界，此时，一个软件块是另一软件块的功能用户，反之亦然。

注意：该“边界”的定义来自于 ISO/IEC 14143/1:2007[13]，增加了“功能性”来修饰“用户”。为避免歧义，注意不应该把边界混淆为用来圈定被度量软件的度量范围所画的“线”。边界不是用来定义度量范围的。

2.3.2 持久存储介质

定义——持久存储介质
使得功能处理在其生命周期结束后仍然能够存储数据组的存储介质，并且/或者，通过该存储介质，功能处理也可以检索数据组，此数据组由另一个功能处理存储，或由同一功能处理之前的事件存储、也可能由某些其他过程存储。 注 1：在 COSMIC 方法中，持久存储介质是一个只存在于被度量软件边界内的概念，因此，它不能被视为被度量软件的功能用户。 注 2：“某些其他过程”的一个例子是只读存储器的生产过程。

（关于“功能处理”的定义，请看 3.2 节）

多数情况下，当分析软件的 FUR 时，如果度量人员发现需求是从特定文件或数据库获取数据或存入数据，FUR 与文件/数据库的物理存放位置和存放方式无关；后者属于技术需求。所有的 FUR 只需要明确：需要存储数据。在通用软件模型中，存储数据的地方称为“持久存储介质”。不需要识别持久存储介质的具体位置；在软件的任何层都可对其进行访问。

该规则的例外是，如果待度量软件的 FUR 指明了数据需要直接存入/来源于一个特定的物理硬件存储设备。（“直接”表示“不需要经过其他软件”。）如果待度量软件直接与硬件设备交互，必须把该设备识别为软件的功能用户；不能把它识别为持久存储介质。更多内容请参考 3.5.8 节。

[11]例如，2002 年 10 月，在德国马格德堡举行的国际软件测量研讨会上，Toivonen 在“移动终端软件内存效率的度量定义”中比较了仅供人类用户使用的手机功能的大小。

[12]事实上，如果度量人员事先已根据 FUR 识别了数据发送者和预期接收者，那么边界也自然确定了。

[13] 译者注：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》

2.3.3 环境图

定义软件的度量范围及功能用户时，画“环境图”是很有用的。包括此手册在内的所有 COSMIC 指南中，环境图用于展示待度量软件块在其功能用户（人，硬件设备或其他软件）环境中的范围以及它们之间的数据移动（环境图通常也展示相关的持久存储介质）。

实际上，环境图是将度量模式（请看 2.0 节）应用到待度量软件的一个实例。图 2.5 列出了环境图所用到的主要符号：

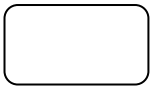
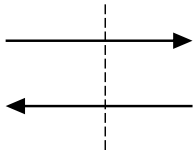
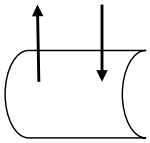
符号	解释
	待度量软件块（加粗框），即度量范围的定义。
	任何待度量软件的功能用户。
	箭头代表跨越功能用户和待度量软件间的边界（虚线）的所有数据移动。
	箭头表示待度量软件与“持久存储介质”间的所有数据移动。 （代表“数据存储”的标准流程图标志强调持久存储介质是一个抽象的概念。使用此标志表明软件并不直接与物理硬件存储器交互。）

图 2.5——环境图所用到的主要符号

业务软件案例：图 2.6 显示了客户/服务器软件的一个已实现的应用程序包（参照 2.2.1 节的图 2.1）的环境图，它被当作一个“整体”度量。即：度量整个包的规模时，会忽略其包含 2 个构件（客户端和服务器端）的事实。

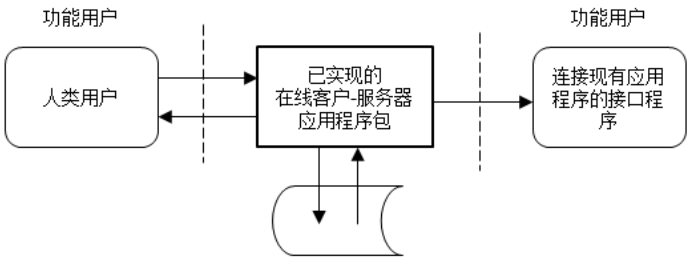


图 2.6 2.2.1 节中客户-服务器应用程序的环境图

实时软件案例：图 2.7 显示了一个简单的嵌入式防盗报警软件系统的环境图（摘自“COSMIC 实时软件规模度量指南”[4]，1.1 版，4.3 节）

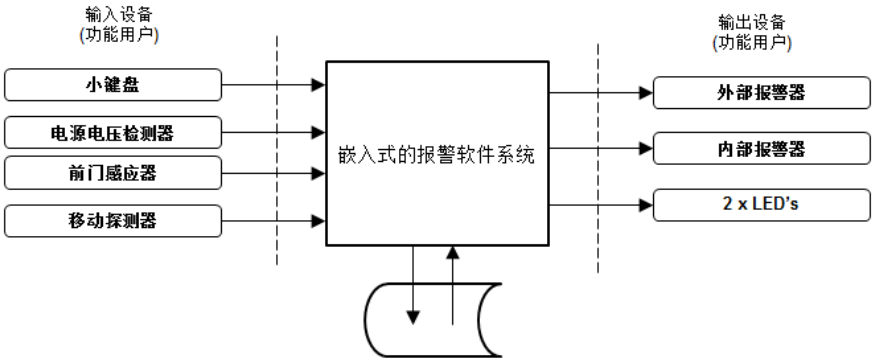


图 2.7 嵌入式防盗报警软件系统的环境图

2.4 识别颗粒度级别

2.4.1 标准颗粒度级别的要求

在软件开发项目的初始阶段，实际需求描述是“高级别”的，即概要的、粗略的。随着项目的推进，实际需求不断完善（如：经历了版本 1、2 和 3 等），“在较低级别”上会给出越来越多的细节。需求（以及从中导出的 FUR）的不同详细程度称作不同的“颗粒度级别”。（可能与此处定义的颗粒度级别相混淆的其他术语，见 2.4.2 章节）

定义——颗粒度级别
对于一个软件块任意部分的描述（例如:对需求的陈述或者对软件结构的描述）的任意扩展级别，每一次进一步的扩展，对软件块功能的描述也更加细化并具有一致的详细程度。 注意：度量人员应该意识到当需求在软件项目的早期演化过程中，在任何时候，需要的软件功能的不同部分通常以不同的颗粒度级别被文档化。

大部分产品开发活动都采用标准的缩放比例绘制平面图，很容易将一个图中度量的尺寸转换成另一个采用不同缩放比例的图的尺寸。相比之下，软件各种级别的颗粒度并没有一个标准的缩放比例，所以可能很难确定两个功能性需求的描述是否在同一个颗粒度级别上。对于在什么颗粒度级别上进行度量（或者度量缩放到什么比例）不能达成共识的话，就不可能知道两个功能规模度量结果是否可以比较。

为了进一步阐述这个问题，请思考如下的类比。一组地图从三个不同的颗粒度级别显示了一个国家公路网的详细信息：

- 地图 A 只显示了高速公路和主干道。
- 地图 B 显示了所有的高速公路，主干道和次干道（就像驾车者的地图册那样）。
- 地图 C 显示了所有的道路并带有名称（就像本地行政区道路图册那样）。

如果我们不了解不同颗粒度级别的现象，可能会认为这三个地图显示了国家公路网的不同规模。当然，对于地图，每个人都理解其展示的不同详细程度，并且有标准的缩放比例来解释各个级别展示的公路网的规模。在这些不同地图的缩放比例的背后隐藏着“颗粒度级别”这个抽象概念。

对于软件度量，仅有一个标准的颗粒度级别可能被明确定义。在此颗粒度级别下，可以识别和定义各个功能处理及其数据移动。度量应该尽可能在这个级别上进行，或者缩放到这个级别上[14]。

2.4.2 对颗粒度级别的澄清

在继续下面的内容之前，确保对 COSMIC 方法中的“颗粒度级别”没有误解是很重要的。把功能性需求进行放大，即把某软件的描述从“较高”扩展到“较低”的颗粒度级别，展示更多的细节，而范围并没有改变。该过程不能和以下活动混淆：

- 对软件进行放大，以展示构件和子构件等（在不同的“分解层级”，见上面 2.2.3 节）。当度量目的要求将总体度量范围按软件的物理结构进行分解时，可能需要这种放大。
- 随着开发周期的进展，如从需求到逻辑设计、物理设计等，软件的描述也在不断演化。不管处于软件开发的什么阶段，我们只对与度量目的相关的 FUR 感兴趣。

因此，“颗粒度级别”概念倾向于仅应用于软件的功能性用户需求。

2.4.3 标准的功能处理颗粒度级别

“功能处理颗粒度级别”是唯一可以将功能需求的颗粒度级别进行标准定义的颗粒度级别。在该颗粒度级别上，我们可以定义一组概念，这些概念使得规模度量是可重复的，进而可比较的。[15]

定义——功能处理颗粒度级别

对软件块描述的一个颗粒度级别，在该级别上：

- 功能用户（类型）是单独的人、工程设备或软件块（而不是它们的任何组合），并且
- 软件块应响应的是单个的事件（类型）（而不是定义为事件组的任何颗粒度级别）

注 1：在实践中，软件文档对功能性需求的描述在不同部分通常具有不同的颗粒度级别，尤其当文档还在演化时。功能处理可以在其中任意颗粒度级别显现出来。

注 2：人、工程设备或软件块的组合（功能用户），举例来说，可能是一个“部门”，它的成员处理多种类型的功能处理，或者是一个有多种仪器的“控制面板”，或者是“中央系统”。

注 3：“事件组”，举例来说，如在一个高颗粒度级别的功能性需求描述中提到的会计软件系统中“销售交易”的输入事件流，或者航空电子软件系统中“飞行命令”的输入事件流。

有了这个定义，我们现在能够定义下面的规则和建议。

规则——度量功能处理时的颗粒度级别

[14] 关于“把度量从一个颗粒度级别缩放到另一级别”这一话题，现已放在 “ 早期或快速 COSMIC 功能规模近似度量指南”[6]中。

[15]之所以命名为“功能处理颗粒度级别”是因为功能处理是在这一级别被识别的——关于功能处理更详尽的讨论，请看 3.2 节。

a) 精确的软件块功能规模度量要求其 FUR 的颗粒度达到能够识别功能处理及数据移动子处理的级别。

b) 如果必须要对一些还没有达到足够详细程度的需求进行度量，可采用近似方法度量需求。这些方法定义了如何在高颗粒度级别度量需求。在高颗粒度级别的度量中运用缩放系数得出功能处理及其数据移动子处理颗粒度级别的近似规模。见“早期或快速 COSMIC 功能规模近似度量指南” [6]。

除了以上规则，COSMIC 建议[16]将能够识别出功能处理及其数据移动子处理的颗粒度级别作为功能规模度量的标准，基准服务的提供者、用于支持或使用功能规模度量（如用来估算项目工作量）的软件工具的设计者都应使用该标准。

业务软件案例：这个实例来自于业务应用软件领域，是一个知名的网上购物系统的一部分，我们称之“Everest 订货应用”。本案例的目的是展示不同的颗粒度级别，从而看到不同级别下显露出的功能处理。下面的描述是高度简化的，用以展示其颗粒度级别。

如果我们想度量此应用程序，我们可以假设度量目的是为了确定应用软件的人类顾客用户（功能用户）使用的功能的规模。然后，我们会定义度量范围为“顾客订货时通过网络可访问的那部分 Everest 应用程序”。但是，注意由于这个案例的目的是为了阐述不同的颗粒度级别，因此，我们只研究此系统整体功能的某些部分，这些部分足以令我们理解颗粒度级别的概念。该案例仅仅是关于 FUR 的颗粒度级别，与底层软件分解层级无关。

在这部分应用的最高“级别 1”（主要功能），Everest 订货应用的需求是如下的一个简单的概括性陈述：

Everest 订货应用必须能够使顾客可以查询、选择、订货、付费和配送 Everest 的产品范围内的所有产品，包括从第三方供货商获得的产品。

放大这个最高级别的需求陈述，我们发现在级别 2，Everest 订货应用由 4 个子功能组成，如图 2.8（a）所示。

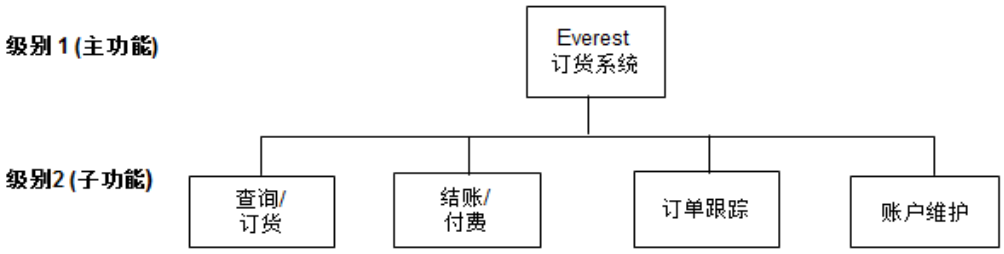


图 2.8（a） Everest 订货系统的分析：前两个颗粒度级别

四个子功能的需求如下：

- *查询/订货子功能，让客户在 Everest 数据库中查询商品价格和是否有货，以及将选择的商品添加到购物篮里。这个子功能同时提供特价商品推荐来提升销售量、提供所选商品的评价、配送周期等常见问题的查询。这是一个非常复杂的子功能。因此，在此例中我们没有在级别 2 之下对其进行更详细的分析。*

[16] 对于功能处理颗粒度级别的使用是“建议性的”而不作为一项规则是因为这一建议不仅适用于 COSMIC 方法的个体使用者，而且适用于那些采用功能点度量方法的服务和工具的提供商。对于后者这一更大的群体，COSMIC 只能给出建议。

- *结账/付费子功能，它使顾客能够确认订单并为购物篮中的商品付费。*
- *订单跟踪子功能，它使顾客可以查询一个订单的配送进展，维护他们的订单（比如修改送货地址）和退回不满意的产品。*
- *账户维护子功能，它使顾客能维护账户的各种详细资料，比如家庭地址、付费方式等。*

图 2.8 的(b)和(c)也显示了放大需求后出现的一些细节，即结账/付费子功能、订单跟踪子功能和账户维护子功能的下一级。放大过程中注意到以下方面是很重要的：

- *我们没有改变被度量功能的范围，并且*
- *Everest 应用的所有级别描述都是顾客（作为功能用户）可用的功能。顾客可以“看到”所有这些颗粒度级别的功能。*

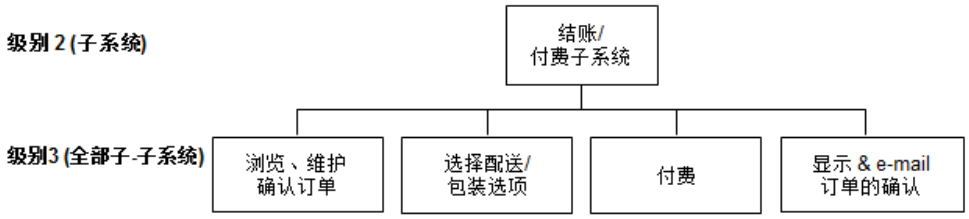


图 2.8（b） 结账/付费子系统的分解

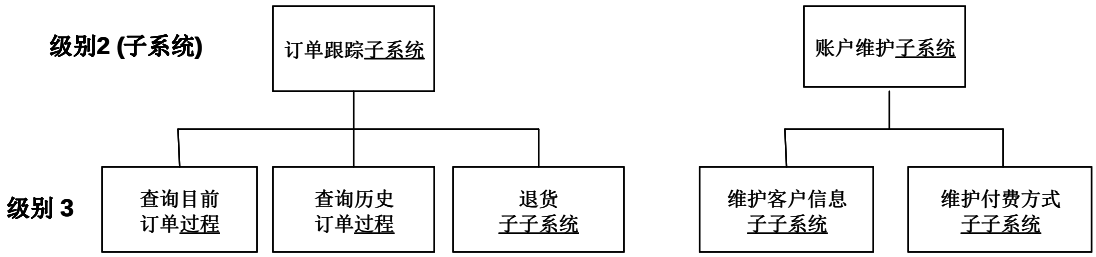


图 2.8（c） 订单跟踪子系统及账户维护子系统的分解

图 2.8(c)展示了将订单跟踪子功能放大到级别 3 时，我们在级别 3 发现了 2 个独立的功能处理（订单跟踪子功能的两个查询）。如果我们继续放大级别 3 的子子功能，会发现更多的功能处理。因此，这个实例说明，当采用“自顶向下”的方法分析某些功能时，不能假设图中某个特定“级别”的功能总是符合 COSMIC 方法定义的在相同的“颗粒度级别”（这个定义要求，在任何同一个颗粒度级别上的功能 “其详细程度是相当的”）。

此外，其他分析师可能也会画出不同的分析图，在图的每个级别展示其他的功能分组。缩放这样一个复杂系统的功能不存在一种唯一“正确的”方法[17]。

鉴于在实践中这些变数不可避免会出现，度量者必须仔细检查分析图的各个级别来找到必须度量的功能处理。当实践中不可能做到这一点的时候，比如，因为分析还没有达到展示出所有功能处理的级别，则必须应用上面的规则(b)。为了说明这一点，我们来检查一下“维护顾客详细信息子子功能”（上图 2.8（c））， 它在账户维护子功能分支中。

对经验丰富的度量者来说，“维护”这个词几乎毫无例外的代表一组事件也就是一组功能处理。因此，我们可以假设“维护”子功能必定包含三个功能处理，称为“查询顾客信息”、“更新顾客信息”和“删除顾客信息”。（“创建顾客信息”处理显然也必须存在，但出现

[17] 图 2.8 可能不是一个最佳实践案例，但对于展示如何画出此类图表，是个典型的案例。

在系统的其他分支里，当顾客在第一次订货时调用该处理。它不属于这个简单实例的度量范围。）

经验丰富的度量者应该能够用 COSMIC 功能点为单位来“推测”这个子子功能的规模，先假设功能处理的数量（本例为三个），然后用这个数乘以功能处理的平均规模。平均规模可以通过对这个系统的其他部分或者其他相似系统进行基准度量而得到。基准度量过程的实例在“使用近似方法进行 COSMIC 早期或快速功能规模近似度量的指南”[6]中给出，此文档还包含了近似规模度量的其他方法的实例。

很明显，这样的近似方法有其局限性。如果我们将这样的方法应用到上面需求的级别 1 的陈述上（“Everest 应用必须使顾客可以查询、选择、订货、付费和配送 Everest 的产品范围内的所有产品……”），我们只能识别出少量的功能处理。但是更详细的分析显示出，这个复杂应用的功能处理一定多得多。这就是为什么当更详细的需求建立后，功能规模通常会随之增加，即使度量范围没有变化。当可获得的细节非常少时，在高颗粒度级别上使用这些近似方法必须非常小心。

关于在各种颗粒度级别和分解层级上度量规模的实例，请查阅使用近似方法进行 COSMIC 早期或快速功能规模近似度量指南中的通信系统实例[6]。

2.5 度量策略阶段的总结

为确保度量规模的结果在日后能被正确理解和运用，确定度量策略的各项参数并将其文档化是很重要的。功能规模度量目的大多与开发工作量相关，如度量项目的开发效率，或者用于项目估算。在大部分情况下，可能会采用标准度量模式，详情请参见度量策略模式指南[5]。

第三章 映射阶段

3.0 章节概要

本章节讨论了度量过程的第二阶段——“映射”阶段，定义了通用软件模型的主要概念，以及为度量功能性用户需求而把软件映射为此模型的步骤。通用软件模型的主要概念为：（斜体部分）

- 引发*功能用户*向被度量软件块发出服务请求的事件称为“*触发事件*”，所请求的服务称为“*功能处理*”。
- 功能处理由两种子处理构成：移动数据的子处理（“*数据移动*”）和运算数据的子处理（“*数据运算*”）。数据运算子处理不单独识别，而被认为由其关联的数据移动所承担。
- 一个数据移动会移动一个“*数据组*”。一个数据组由数据属性组成，这些数据属性都描述了同一个“*兴趣对象*”，即功能用户所关心的、感兴趣的对象。
- 有四种数据移动：*输入*将一个数据组从功能用户一方跨越边界移入到功能处理；*输出*将一个数据组从功能处理一侧跨越*边界*移出到功能用户；*读*和*写*在功能处理和*持久存储介质*之间移动一个数据组。

上述概念都定义在本章节中，并提供了完整的原则和规则，以帮助正确地识别这些概念。同时，也提供了上述概念在不同类型软件中的大量实例。

3.1 把功能性用户需求映射为通用软件模型

图 3.0 展示了把已有的软件制品中的功能性用户需求（FUR）映射为 COSMIC 通用软件模型所需格式的过程。此过程的每一个步骤是一个特定小节的主题，如图 3.0 中的标题条所示。

该映射过程可以适用的软件制品范围非常广泛。我们鼓励度量者使用此通用过程派生出更多特定规则在他们本地化环境中使用，以便于将其按照本地化软件工程方法生成的软件制品映射到 COSMIC 通用软件模型。制定特定的本地化过程的目的在于：减少映射阶段的不确定性，并提高度量的准确性及可重复性，这些过程通过本地化具体示例来说明。

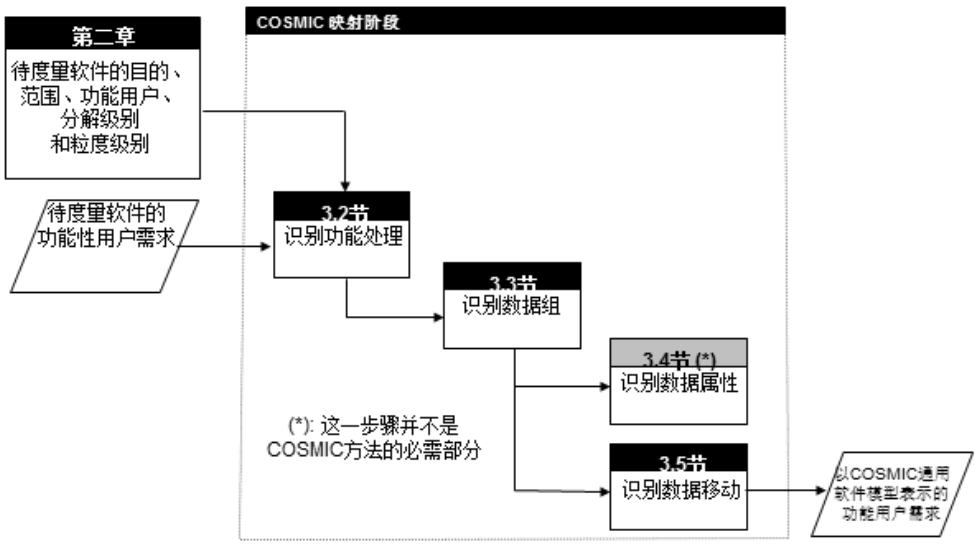


图 3.0 COSMIC 映射过程的通用方法

目前已有一些指南描述了如何将不同领域中各种数据分析和需求定义方法映射到 COSMIC 方法的概念。如“业务应用软件规模度量指南” [7]、“数据仓库应用软件规模度量指南” [8]、“面向服务架构软件规模度量指南”[9]以及“实时软件规模度量指南” [4]。关于商业[10]和实时领域[11]，也有些“快速参考指南”用较少的篇幅概述了整个映射过程。

图 3.1 和 3.2 的目的在于帮助我们从度量策略阶段使用的软件环境模型转换到通用软件模型。一个适用于业务应用软件，一个适用于典型的实时嵌入式软件，并分别对应于图 2.6 和 2.7 的软件环境模型。

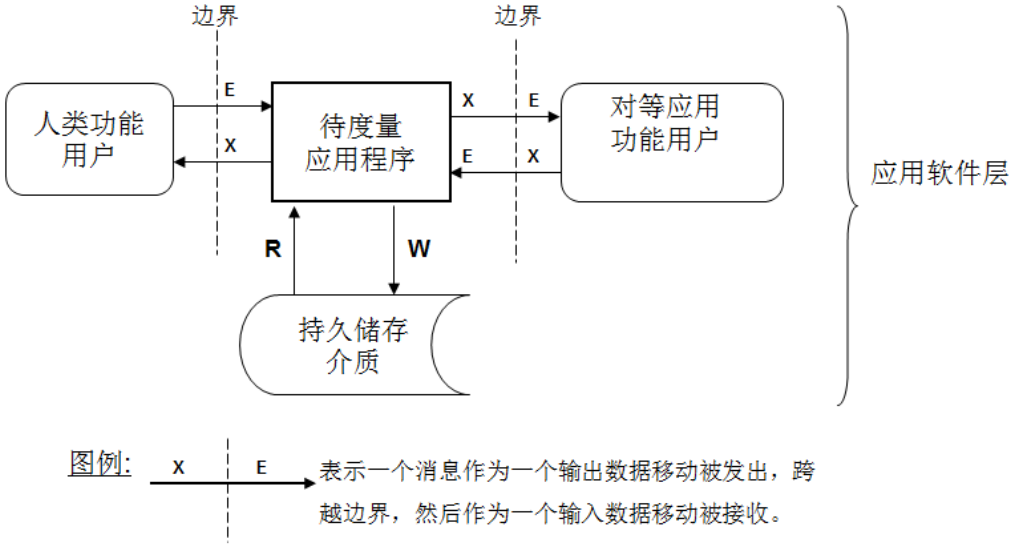


图 3.1 人和对等应用程序作为功能用户的业务应用软件

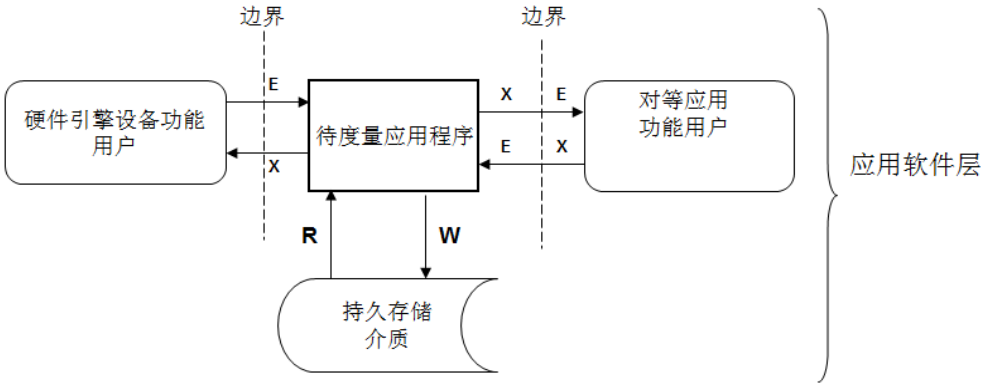


图 3.2 各种硬件工程设备和对等软件作为功能用户的实时嵌入式应用软件

根据软件环境模型的原则 f)（参见 1.3.1 节），需要注意，待度量软件的功能用户是“向软件发送数据的发送者和/或从软件接收数据的预期接收者”。通用软件模型的原则 a)（参见 1.3.2 节）告诉我们，软件块“跨越边界与其功能用户进行交互，并与边界内的持久存储介质进行交互”。

图 3.1 显示了待度量应用软件有两个功能用户，人和另一个对等应用。图 3.2 中的嵌入式软件有硬件设备和对等应用作为其功能用户。显示数据移动的箭头用不同的缩写字母标记，分别代表不同的类型（E=输入，X=输出，R=读，W=写）。

其中需注意，“持久存储介质”是指待度量应用程序通过读或写数据移动需要访问的任何逻辑存储介质，而不是指任何类型的物理存储介质。

3.2 识别功能处理

映射阶段的第一步是从待度量软件块的功能性用户需求中识别出功能处理集。

3.2.1 定义

定义——事件
发生的某事。

定义——触发事件
待度量软件的功能性用户需求中可识别的一个事件，此事件使得一个或多个软件功能用户产生一个或多个数据组，第一个产生的数据组（可能是由任何一个功能用户所产生的）随后被一个触发输入所移动。一个触发事件不可再拆分，并且要么已经发生，要么没有发生。 注：时钟和定时事件可以作为触发事件。

定义——功能处理
a) 体现了待度量软件的功能性用户需求基本部件的一组数据移动，该功能处理在这些 FUR 中是独一无二的，并能独立于这些 FUR 的其他功能处理被定义。 b) 一个功能处理必须只有一个触发输入。每个功能处理在接受到由其触发输入数据移动所移动的一个数据组后，开始进行处理。 c) 一个功能处理的数据移动的集合是响应触发输入的所有可能的功能性需求所需要的集合。[18] 注 1：实现时，一个功能处理实例，在收到一个触发输入实例移动的数据组实例时，才开始执行处理。 注 2：除了触发输入外，一个功能处理的 FUR 可能需要一个或多个其他的输入。 注 3：如果功能用户发送了包含错误的数据组，例如，由于传感器失灵，或者人输入的命令存在错误，通常是由功能处理来判断事件是否确实发生、和/或输入的数据是否有效、以及如何响应。

[18] 译者注：英文原文为：The set of all data movements of a functional process is the set that is needed to meet its FUR for all the possible responses to its triggering Entry.

注 4：如果一个功能处理是由一个来自于 FUR 的、独一无二的触发事件产生的触发输入而引起的，那么该功能处理是独一无二的（如 a）所述），且其规模应计入该 FUR 规模中。同一段 FUR 描述的两个或多个功能处理可能是互不相同的，即使他们共享某些功能。见 3.2.7 节的例子，共享相同功能的功能处理。

定义——触发输入

触发输入是一个功能处理的输入数据移动，它移动了功能用户产生的一个数据组，该数据组是功能处理开始执行处理所必需的[19]。

注：该定义来自通用软件模型，该模型是一个逻辑模型。实际上，一个功能处理可能在数据还没有输入前就开始运行了。如：当一个人类用户点击菜单，显示了空白界面等待数据输入。

图 3.3 描述了触发事件、功能用户和触发待度量功能处理的输入数据移动之间的关系。对该图的解释为：*一个触发事件引起功能用户生成一个数据组，该数据组由功能处理的触发输入移动，以启动功能处理。*

注意：为了方便阅读，我们通常省略对数据组的提及，直接陈述为功能用户发起了一个触发输入，从而启动了一个功能处理，或者更简洁地说，功能用户触发了一个功能处理。

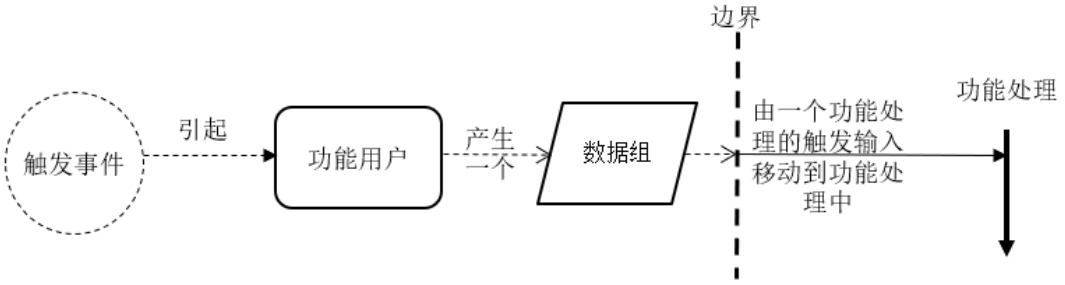


图 3.3 触发事件、功能用户及功能处理间的关系

图 3.3 提到的所有概念（触发事件/功能用户/触发输入/功能处理）间的关系可能是一对多、多对一、或者多对多，也有一个例外。这个例外即由任何一个触发输入移动的数据组可能只引发一个功能处理——请看功能处理的规则 b)。下面列出了一些可能的基数[20]的例子：

- 一个触发事件可能被多个功能用户所感知，如多台传感器侦测到同一场地震。
- 一个业务应用程序的人类功能用户能感知到很多类型的事件。当一个人类用户决定此事件必须通知该应用程序，例如，这是个触发事件，用户会启动一个触发输入。（当一个人类用户想要进行一个查询时，他甚至能够有效地“产生”一个事件。）同样地，一个软件应用程序

[19] 译者注：英文原文为：The Entry data movement of a functional process that moves a data group generated by a functional user that the functional process needs to start processing.

[20] 译者注：基数译自 cardinalities 这个单词，意为两个实体之间的关系是 1：1，1：多，多：多等。

A 是另一个应用程序 B 的功能用户，A 可能会因为各种不同的目的向 B 发出“请求”。每一个请求（类型）都对应着 A 向 B 发出的一个独立的触发事件。

- 在某些安全攸关的软件中，一个硬件功能用户在检测到一个触发事件时，可能会发起多个触发输入。
- 一些功能处理可能会被不同的功能用户所触发，例如，一个可复用的软件构件可被其所有的软件功能用户调用。

在实践中，被度量软件所有处理链的基数（即描述了特定事件引起特定功能用户触发特定功能处理）将在软件的 FUR 中进行约束。关于图 3.3 处理链的基数更详细的讨论以及更多实例，请查阅本度量手册的附录 C。

由于一个触发事件引起的、由功能用户产生的数据组，依赖于处理该数据的功能处理的 FUR。实例如下：

业务软件案例 1：一个人事软件系统的功能处理可能开始于如下的触发输入：移动一个描述某新员工信息的数据组。该数据组是由使用人事软件输入数据的人类功能用户所生成。

实时软件案例 2：一个实时软件系统的功能处理可能开始于如下的触发输入：告知功能处理产生了一个时钟节拍（功能用户）。被移动的数据组传递的数据只是告知一个事件发生了（一个节拍，可能只用一个比特表示）。

实时软件案例 3：一个工业用实时火灾探测软件系统的功能处理可能开始于如下的触发输入：被某个烟雾探测器（功能用户）触发。探测器生成的数据组传递的数据是“检测到烟雾”（已发生的事件）这一信息及探测器的 ID（能用于定位事件发生地点的数据）。

实时软件案例 4: 当一条形码出现在其感应窗口时（触发事件），在超市收款台的条形码阅读器（功能用户）便开始扫描。阅读器会生成一个含有条形码图像的数据组，并输入给结账软件。数据组图像被触发输入移动到功能处理中。如果条形码有效的话，后者将在顾客的账单上增加产品的价格，发出“哔”的响声告知顾客产品已被接受，并登记此销售记录等等。

3.2.2 识别功能处理的方法

识别功能处理的方法取决于度量者可用的软件制品，而软件制品相应地又取决于在软件生命周期中的哪个时间点提出了度量要求，同时还取决于当前使用的软件分析、设计、开发方法。由于后者的情况千差万别，本度量手册只提供识别功能处理的一般过程。

识别了功能用户、给出了被度量软件的 FUR 以及参考了附录 C 中的例子后，识别功能处理的过程按照如图 3.3 的流程执行，即：

- 从被度量软件的功能用户的领域，识别出必须响应的单个事件——“触发事件”（可以在状态图和实体生命周期图中识别触发事件，因为一些状态迁移和生命周期迁移都对应着软件必须响应的触发事件）；
- 识别出软件中每个触发事件必须响应的功能用户。
- 识别出每个功能用户为响应该事件而发起的触发输入(或输入)。
- 识别出被每个触发输入启动的功能处理。

使用以下规则来检查被识别出的候选功能处理是否恰当。

规则——功能处理

a) 一个功能处理应该完全属于某层且仅属于某一层的一个软件块的度量范围。

b) 一个功能处理至少包含两个数据移动，即一个触发输入加上一个输出或写，最小规模为 2CFP。一个功能处理中数据移动的数量没有上限，因此其规模也没有上限。

c) 一个执行中的功能处理，当其响应了触发输入并满足 FUR 时，则功能处理结束。由技术原因导致处理出现暂停时，不能认为功能处理结束了。

3.2.3 业务应用软件领域的触发事件和功能处理

a) 一个在线业务应用软件的触发事件通常出现在人类功能用户的现实世界中。人类用户通过输入关于事件的数据，将事件的发生传递给功能处理。

业务软件案例 1：在一家公司里，收到一个订单（触发事件），引发一位员工（功能用户）录入订单数据（触发输入传递了关于兴趣对象“订单”的数据），这就是“订单录入”功能处理的第一个数据移动。

b) 不同的触发事件（类型）及因此产生的不同功能处理（类型）应按以下场景进行区分:

当人类功能用户需要做出关于“下一步做什么”的决策时，而这些决策是在软件之外做出的、且在时间上是各自独立的，这就要求软件做出不同的响应，那么每个独立决策就是一个单独的触发事件，软件必须分别为其提供一个单独的功能处理。

业务软件案例2：一个功能用户输入顾客购买一件复杂工业设备的订单，并随后确认顾客订单的接收情况。在输入及验收订单的过程中，用户可能会查询新订单能否在要求的配送日送达以及顾客的信用度等等。尽管输入了订单后才可以验收订单，在本场景中用户必须做出独立的决策来验收订单。这意味着输入订单与验收订单是两个独立的功能处理（并且查询新订单与已配送的订单也要区分为不同的功能处理）。

当活动的责任各自独立的情况：

业务软件案例3：在人事管理系统中，维护人员基本数据与维护薪资数据这两种责任是分开的，这意味着各自的功能用户有自己独立的功能处理。

c）应用 A 有一个对等应用 B，A 需要向 B 发送数据，或者 B 从 A 接收数据。当 B 需要向 A 发送数据或者从 A 接收数据时，B 对 A 触发一个功能处理。那么，B 就是 A 的一个功能用户。

商业软件案例4：假设业务软件案例1 接收到一个订单，订单处理应用程序需要将所有新客户的详细信息发送给中心客户注册程序，后者是被度量的软件。那么，订单处理应用程序就成了中心程序的功能用户。接收到新客户数据的订单程序生成客户数据组，并将其发送到中心客户注册程序，触发一个功能处理来存储这些数据。

d) 分析一个功能处理时，其采取的是在线处理还是批处理模式，在原则上没有差异。根据定义，批处理的所有输入数据在处理开始前都必须暂存在某处。请看图 3.4（注意：我们把输入数据（所有输入）与可能被批处理读或写的持久数据区分对待）。当度量批处理软件时，应该使用与直接输入到应用程序的数据相同的方法分析暂存的输入数据。这些输入数据不是“持久数据”。

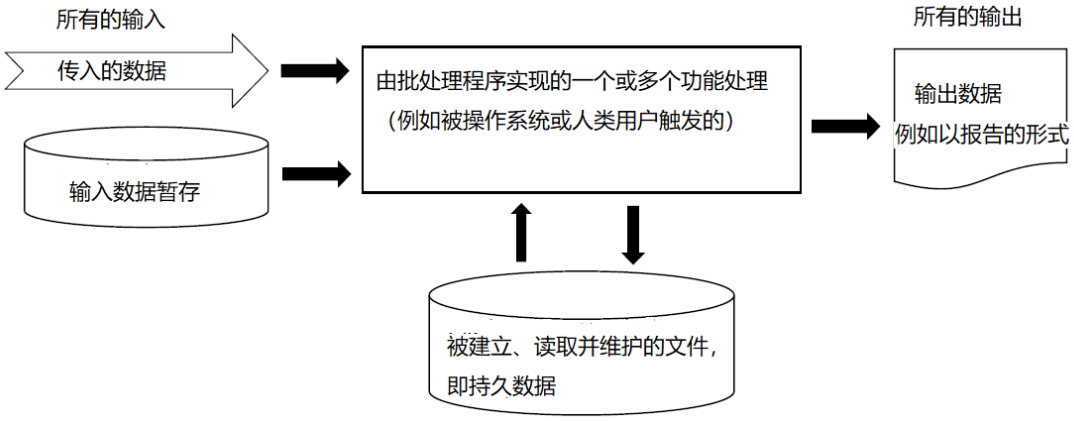


图 3.4 一个批处理过程的“作业”，实现一系列的功能处理

注意：“某些输入数据应该进行批处理”这是一个非功能需求（NFR）。这个 NFR 的作用是在把数据输入到批处理应用程序中时，这些数据必须是可用的（以一“批”的形式）。在实践中，其如何实现与批处理软件 FUR 的分析无关。

还要注意的是，分析批处理“作业”中的每个功能处理（类型）时，应将其视为一个整体来分析，独立于同一作业中的其他功能处理。作业中的每个功能处理都应有自己的触发输入。

业务软件案例 5：假设业务软件案例 1 中订单采用“线下”的方式输入，如通过纸质文件扫描，并且为进行自动的批处理而临时存储起来。如果订单输入功能处理是成批处理的，如何分析呢？功能用户是在线下输入订单数据并准备好进行批处理的人；进行订单批处理的触发输入是功能处理的一个数据移动，此数据移动把订单数据组从临时存储器移动到处理中。（如果一定要度量线下处理，它涉及另一个独立的功能处理。实际上，功能用户发起了两个触发输入，一个启动了线下处理过程把订单载入到临时存储器内，另一个启动订单的批处理。）

业务软件案例 6：假设一个年终批处理应用程序的 FUR 要求报告当年的经营成果，并为新一年的开始重置各项指标。物理上，操作系统产生的年终时钟节拍使得该应用程序启动处理。然而，逻辑上，应用程序的每个功能处理都从待进行批处理的数据流中获取输入数据。应采用常规的方式对其进行分析（如，功能处理的输入数据包括一个或多个输入，其中第一个输入就是该功能处理的触发输入）。

然而，假设批处理应用程序中有一个特殊的功能处理，它不需要任何输入数据来生成它的报告集。物理上，（人）功能用户已经把触发这个功能处理的任务交给了操作系统。由于每一个功能处理必须有一个触发输入，我们可以认为启动批处理流的年终时钟节拍担当了该处理的这一角色。该功能处理可能需要数个读和很多个输出来产生它的报告。逻辑上，在这个案例中，如果功能用户通过鼠标点击在线菜单的一个选项来生成一个或多个报告，而不是将工作交给操作系统来触发以批处理形式生成报告，对该案例的分析也没有差别。

关于辨别批处理流中的触发事件和功能处理的一些案例，请参见“使用 COSMIC 度量业务应用软件规模的指南” [7]4.6.3 节。

3.2.4 实时应用程序领域的触发事件和功能处理

a) 触发事件通常由传感器检测到。

实时软件案例 1：当传感器（功能用户）检测到温度达到一个特定值（触发事件）时，传感器会发出一个信号启动一个功能处理的触发输入数据移动来关闭加热器(另一个功能用户)。

实时软件案例 2：军用飞机上装有探测“导弹接近”的传感器。该传感器是对威胁做出响应的软件的功能用户。对该软件来说，当传感器检测到什么时才会发生一个事件，是传感器（功能用户）生成一个数据组启动一个触发输入，比如说“2 号传感器检测到了一枚导弹”，也许还包括导弹的接近速度和坐标的数据流。[21]

b) 来自时钟的周期性信号（“时钟节拍”）可以触发一个功能处理。

参见 3.2.1 节中的实时软件案例 2。没有伴随时钟节拍的其他数据。功能处理从各种各样的传感器中获取数据并采取必要的行动。

3.2.5 关于区分功能处理的更多信息

软件仅依赖 FUR 来区分事件，并提供相应的功能处理。度量软件时，有时很难确定哪些独立事件是软件要求识别的。这种情况尤其会发生在当原始需求已经无法获得时，例如，开发人员可能发现在一次实际处理中同时实现几个需求更为经济。在这种情况下，检查输入数据的组织形式（请参照下面的案例），或者检查已安装软件的菜单，可以帮助识别那些软件需要响应的独立事件和相应的功能处理。

业务软件案例 1：假设一个功能性用户需求针对两种社会福利，一个是面向多添一个孩子的税收抵免，另一个面向低收入人群的工作税收抵免。对人类功能用户来说，需要软件对两个独立的事件做出响应。因此，尽管使用一张税务报表就可以获得两种情况的数据，也应该有两个功能处理。

根据功能处理定义中的条款 c)，一个功能处理必须“满足 FUR，以响应其所有可能的触发输入”。这意味着，同一个功能处理类型必须能够处理由触发输入移动数据组的所有数据属性可能出现的值，包括有效和无效的值，甚至在某些案例中包括数据值缺失的情况。所有这些由触发输入移动的数据值的变化，通常会导致功能处理在执行时遵循不同的处理路径。但尽管存在这么多变化，我们仍需识别为由一个触发输入类型启动的一个功能处理类型。（度量者只需识别此功能处理的所有数据移动；他们可能处在各种处理路径中的哪一条上与度量无关。）

业务软件案例 2:一个对数据库提供通用搜索功能的功能处理可能需要接受最多 4 个搜索参数（其触发输入的属性）。但是如果只输入 1 个、2 个或 3 个参数的值，该功能处理也会运行。

业务软件案例 3：对于一个汽车租赁公司登记新客户信息的功能处理来说，大部分数据属性的数据是必填的，但有些信息（如一些合同细节）是可选的，可以空着不填。无论输入了所有数据属性还是只输入了一部分，也只有一个功能处理用于登记新客户的信息。

业务软件案例 4：继续分析案例 3，对于该公司中办理汽车租赁预约服务的功能处理来说，有很多可选的服务，它们可能被选中或者没选中，如额外的保险、第二驾驶员和儿童座椅等。在汽车租赁预约的功能处理中，这些不同的选项导致不同的处理路径，但也只有一个功能处理用于汽车租赁预约。

实时软件案例：一个发送到航空电子系统的功能处理的触发输入（由地理定位系统发送的飞机高度信息）将根据输入的值，在功能处理中从两条完全不同的处理路径中选择其一来触发，

[21] 原文中此案例有编号为 b)，与 Charles 确认后认为是编辑错误，此案例和上一个案例都是在说明传感器感知触发事件的例子，故修订。

即飞机的高度是高于还是低于某一设定的高度，不同的路径将在飞行员的地图上显示不同的数据组，如果高度过低，将会发送追加警告。这个案例中也只有一个功能处理。

一旦识别出来后，每个功能处理就可以被登记到通用软件模型矩阵（附录 A）中单独的一行，且归属在合适的层和指定的软件块之下。

3.2.6 度量分布式软件系统的构件

当度量目的是要分别度量分布式软件系统中各构件的规模时，则要为每个构件定义单独的度量范围。在这种情况下，度量每个构件的功能处理遵循上文所述的一般规则。

每个度量过程（…定义范围、功能用户和边界等…）遵循以下规则：如果一个软件块由两个或多个构件构成，每个构件的度量范围之间不能有重叠。每个构件的度量范围必须定义一套完整的功能处理。例如，一个功能处理不能一部分在一个范围内，一部分在另一个范围内。同样地，即使两个构件有信息交互，在度量范围内的一个构件的功能处理不需要有任何关于度量范围内的另一个构件的信息。

每个构件的功能用户是通过检查触发功能处理的事件在被检查构件中发生的位置来确定的（触发事件只在功能用户的一方发生）。

图 3.7 展示了客户—服务器分布式系统两个构件的功能处理以及他们交换的数据移动。

3.2.7 共享相同或相似功能的功能处理的独立性：复用

在同一被度量软件中，任意的两个或多个功能处理中可能会有一些相同或非常相似的功能。这种现象被称作“功能共性”，或功能的“相似”。

但是，在 COSMIC 方法中（与其他所有 FSM 方法一样）每个功能处理都是独立定义、建模和度量的，即不参考同一被度量软件中的其他功能处理（请看功能处理定义中的条款 a））。在同一被度量软件中，如果在两个或多个功能处理间有相同或相似的功能，则在度量规模时，此功能必须在每个功能处理中分别纳入。以下是在实践中可能遇到的功能共性与相似情况的实例：

业务软件案例：在同一被度量软件中的几个功能处理可能需要相同的确认功能，如用于确认“订单日期”，或需要访问相同的持久数据，或需要执行相同的利息计算。

实时软件案例：在同一被度量软件中的几个功能处理可能需要从同一个传感器（对同一数据组的相同移动）中获取数据，或需要执行相同的缩放换算，如从华氏温度转换到摄氏温度（相同的数据运算）。

当一个 FUR 在软件里被实现时，任何“功能共性”可能开发为可复用软件，也可能不会。当度量软件规模时，必须忽略所有包括实际的或潜在的软件复用的实现决策。但是，当使用功能规模度量法进行项目工作量分析或估算时，也许需要考虑复用。

3.2.8 触发软件系统开始执行的事件

当度量软件块的规模时，只识别事件及相应的触发输入，这些触发输入所触发的功能处理按其 FUR 的定义是软件必须响应的。启动（或“打开”）软件本身的功能并不是这些功能处理的一部分，应该被忽略（或者有需要的话，单独度量）。下面的实例描述了在三个领域中如何启动软件。

业务软件案例：对于业务应用软件来说，启动应用程序的功能用户可能是操作系统的任务调度构件、电脑操作者或其他人类用户（如个人电脑的使用者打开一个浏览器或者文字处理软件）。

实时软件案例：对于实时应用程序来说，启动应用程序的功能用户可能是操作系统、发出时钟信号的网管系统或者人类操作者（如从操作员工作站启动一个过程控制系统）。

基础设施软件案例：对于电脑操作系统来说，启动操作系统的功能用户是一个引导程序，当电脑的电源被打开时，此程序便被启动。

下面是来自两个不同领域中的案例：在 FUR 的描述中如果存在关系的话，启动被度量软件的事件/处理与软件必须执行的事件/处理之间的关系。

业务软件案例 1：以批处理模式为各种各样的功能处理输入数据的应用程序可能是由操作系统的一个调度程序启动。如果目的是度量批处理应用程序的 FUR，应该忽略“启动系统”这一功能。批处理应用程序的功能处理的触发输入和其他所需输入将形成该批处理应用程序的输入数据。

业务软件案例 2：例外，在某一时间周期的结束时生成总结报告的批处理应用程序可能不需要任何功能用户直接提供输入数据就能启动。详情请看 3.2.3 节的业务软件例 6。

实时软件案例：现代化的汽车装有一个叫做电子控制单元(ECU's)的分布式系统来控制多种功能，如发动机管理、刹车和空调等等。在汽车开放系统架构（AUTOSAR）中，在一个分布式系统中，有一个一直在运行的“网络管理”(NM)的模块，它负责激活联接到网络（“总线”）的电子控制单元。这个模块同时也负责处理 ECU 不同工作状态间的协调转换：正常运作、电力不足和睡眠状态。因此，NM 负责唤醒或使 ECU 休眠。当度量任何 ECU 应用软件时，应该忽略此 NM 的功能。

3.3 识别兴趣对象和数据组

3.3.1 定义和原则

识别了功能处理以后，接下来的主要目的就是识别它们的数据移动。为此我们必须回忆一下通用软件模型（参见 1.3.2 章节）和本章的 3.0 节中对此概念的介绍：数据移动移动了一个数据组，它的数据属性描述了一个单一的兴趣对象。因此，为了理解数据移动，我们必须首先定义和理解这些概念。

定义——兴趣对象
从功能性用户需求中识别出来的、存在于功能用户世界中的任何“事物”，软件要为之输入/输出数据组，和/或向/从持久存储介质移动数据组。可能是具体的事物，也可能是概念性对象或其一部分。 注 1：在 COSMIC 方法中，采用“兴趣对象”术语，以避免与特定的软件工程方法相关。该术语并不意味着是面向对象方法中的“对象”。 注 2：当一个功能用户发送了一个描述其本身的数据组，比如它的状态或者它的身份，或当一个功能用户接收了一个描述其本身的数据组，那么该功能用户也同样满足“事物”的定义，因此它也是该数据组所描述的兴趣对象。

注 3：没有绝对的兴趣对象。即使在同一个待度量的软件中，一个“事物”在一个或多个功能处理中，对于某个功能用户来说是感兴趣的对象；但可能在其他功能处理中就不是另一个功能用户感兴趣的对象。

更多关于此定义注 2 的信息，请参考 3.3.4 节。

定义——数据组
一个唯一的、非空的、无序的数据属性的集合，包含的每个数据属性描述了同一个兴趣对象的一个互补的侧面。 注：术语“数据组”并不一定指的是“描述某个兴趣对象的所有数据属性的集合”。软件的 FUR 可以根据不同的功能处理的需要，把描述同一兴趣对象的数据属性任意组合成为数据组。

一旦被识别出来，每个候选的数据组必须符合以下原则：

原则——数据组
通过它的唯一的数据属性集合，每个被识别的数据组必须是唯一的和可区别的。

定义——数据属性
一个数据属性是一个已识别的数据组中最小的信息单元[22]， 包含了来自于软件 FUR 角度的一种含义。 注：同义词“数据元素”

（更多关于数据属性的内容，请见 3.4 节。）

实践中，数据组的具体化有多种形式，例如：

a) 作为硬件存储设备上的一个物理记录结构（文件、数据库表和 ROM 存储器等等）。

b) 作为计算机易失性存储器中的一个物理结构（动态分配的数据结构，或者是内存空间中预先分配的一个内存块）。

c) 作为与功能有关的数据属性在 I/O 设备上(显示屏幕，打印的报告，控制面板显示器)的集中展现。

d) 作为在设备与计算机之间、或在网络中传输的一条消息。

一经识别出来，数据组就可以被登记到通用软件模型矩阵（附录 A）单独的一列中，置于标题为“数据组名称”的列下。

3.3.2 关于兴趣对象和数据组的识别

[22] 译者注：The smallest parcel of information 此处翻译为了：最小的信息单元，而没有翻译为：最小的信息包。单元中包含了基本的含义，不可继续细分的含义。

为适用于尽可能广泛的软件，兴趣对象和数据组的定义、原则刻意地被放宽。这一点有时导致在度量一个特定软件块时，很难应用这些定义和原则。因此，下面提供一些案例为具体情况下应用这些原则提供帮助。

当一组数据属性被移入或移出一个功能处理，或者被一个功能处理移入或移出持久存储介质，分析它们时关键是要确定这些属性传递的数据是否全部是关于一个单一的“兴趣对象”的。按 COSMIC 方法的定义，数据移动所移动的是数据组，因此“兴趣对象”决定了“数据组”的数量。例如，如果输入到一个功能处理中的数据属性是属于三个独立的兴趣对象的，那么我们就需要识别出三个独立的“输入”数据移动。

在分析业务应用软件的功能处理的输出时，判断数据组的数量尤其困难，可能包含如下情况：

- 多个数据组，每个描述的是不同的兴趣对象，如，一张报表显示了多个级别的汇总数据；
- 根据输入条件变化而变化的查询结果；
- 数据组之间各自不相关，如，一张发票包含了其他无关服务的广告。

以下规则可以帮助区分功能处理输出的数据组，也就是兴趣对象。同时，这些规则也适用于功能处理的输入，从/向持久性存储介质读/写数据，并且适用于任何功能领域。

规则——识别同一个功能处理中不同的数据组（即不同的兴趣对象）

对于作为功能处理输入的所有数据属性：

a) 具有不同实例频率的数据属性集描述的是不同的兴趣对象；

b) 具有相同实例频率但不同关键属性的数据属性集，描述的是不同的兴趣对象。

c) 按照规则 a)和 b)判断后归为一个集合的数据属性属于同一数据组类型，除非在 FUR 中明确说明了输入此功能处理的、描述同一个兴趣对象的是多个数据组类型。（见注 3）

这些规则也同样适用于功能处理输出的数据属性，或从功能处理移动至持久存储介质的数据属性，或从持久存储介质移动至功能处理中的数据属性。

注 1：上述规则有助于分析复杂的输出，例如：某报告描述了多个兴趣对象，可以把每个数据组当作是由单独的功能处理输出的。当度量复杂的报告时，按照此方法识别的每一个数据组类型也需要分别计数。例如，“业务应用软件度量指南”[7] 2.6.1 节的例子以及 2.6.2 节的分析。也可参考 4.2.4 节中例 4、5 的分析。

注 2：检查输入/输出中的数据组实际是如何组成的或拆分的，也可以帮助识别出不同的数据组类型，但不要依赖于此方法。比如，某一输入或输出有两个或多个数据属性集，为了美观或易于理解把它们从物理角度进行拆分，在这种情况下，如果满足上述规则，则应属于同一数据组。

注 3：见度量手册 3.5 节关于*数据移动*的定义、原则和规则。关于这些规则的例外情况，见 3.5.7 节（例 2、3、4、5）以及 3.5.11 节，如上述规则 c 所述。

关于该规则的简单示例，见 3.3.2 节的业务应用软件例 3。更多复杂的案例，请见“业务应用软件规模度量指南”[7]。

业务应用软件领域的兴趣对象和数据组

在业务应用软件中，如果数据在功能处理结束后仍然存在，则数据是“持久的”，如果数据仅仅存在于该功能处理的输入或输出中，则数据是“瞬态的”。

注：持久与瞬态的区分并不适用于兴趣对象；后者是“事物”，实体上的或概念上的，是从待度量软件的 FUR 中识别的。

业务软件案例 1：在业务应用软件领域，假设软件用来存储关于雇员或订单的数据，那么一个兴趣对象可以是“雇员”（实体的）或“订单”（概念上的）。如果是订单，一般来说对于多行订单的 FUR 可以识别出两个兴趣对象：“订单”和“订单行”[23]。对应的数据组可以叫做：“订单数据”和“订单行数据”。

当有一个查询请求或报告请求想要查询关于某个或多个“事物”（即 “兴趣对象”）的数据时，瞬态数据组就形成了，这些数据没有保存在持久存储介质中，但是可以从持久存储介质保存的数据中检索或单纯经过数据运算得到。此类查询的输入数据组（为检索所需数据而选择的参数）和输出数据组（包含所需要的属性）都是瞬态数据组，因为他们只存在于输入和输出中，在功能处理结束后不存在。

业务软件案例 2：假设有一个针对人事数据库的临时查询，获取所有年龄超过某岁的员工总数，年龄为必填项。输入参数（年龄下限）是一个瞬态数据组，定义了兴趣对象“超过年龄下限的员工集合”。输出数据组包括了超出该年龄下限的员工总数，也是瞬态数据，与输入描述的兴趣对象相同。而为了得到员工总数而查询的员工文件表是持久的数据。

业务软件案例 3：假设与案例 2 相同的临时查询，但是除了需要输出符合年龄条件的员工总数，还需要在查询结果中列出所有符合年龄条件的员工姓名。分析的过程与案例 2 一致，但是该功能处理需要输出两个数据组：员工总数（瞬态数据）以及符合年龄条件的员工姓名（来源于持久数据）。因为他们的实例频率不同，因此要区分为两个数据组（一个员工总数，以及 0 个、1 个或多个员工姓名，见如上规则）。

从案例 3 中得到的一个重要启示是，“一个集合”以及“一个集合中的成员”经常表示的是不同“事物”，因此需要识别为不同的兴趣对象。在这个例子中，功能处理的输出数据组描述了两个兴趣对象，分别为“符合年龄条件的员工集合”以及“员工”。他们都是只有一个属性的数据组，分别是“年龄超过给定条件的员工总数”以及“员工姓名”。

关于分析数据以确定兴趣对象和区分数据组的方法的详细讨论，参见“COSMIC 业务应用软件规模度量指南”[7]。

实时软件领域的兴趣对象和数据组

在实时软件中，通常不需要考虑数据是否是瞬态的。一般来说，对于所有实时功能处理的输入/输出都是瞬态的数据，除非有意将其持久化（比如为了记录日志），或从持久存储介质中获取。如下为实时软件关于数据组及兴趣对象的例子。

实时软件案例 4：来自一个物理设备的输入数据组可能表示：

[23] 译者注：比如一个订单上有多个商品，则有多个订单行。订单和订单上的商品是一对多的关系。

- *兴趣对象的当前状态，如阀门是打开还是关闭。（见 3.3.4 节，当一个功能用户发送关于其本身的数据时，该功能用户也是兴趣对象）。*
- *一个触发事件已发生，启动了一个功能处理（该功能处理可能中断了一个正在执行的功能处理）。该事件即为兴趣对象。*

类似地，输出到物理设备的数据组（例如打开或关闭警告灯），传递的是兴趣对象灯的数据。

实时软件案例 5：一个报文交换软件系统，根据其具体的软件 FUR，可能会将收到的一个报文数据作为一个输入，然后不作修改地作为一个输出推送出去。报文数据组的属性可以如："报文 ID、发送者 ID、接收者 ID、路由码和报文内容"，兴趣对象是"报文"。

实时软件案例 6：一个代表了在 FUR 中提到的兴趣对象的公共数据结构，可以由功能处理来维护，并且对于待度量软件中的大多数功能处理来说是可访问的。

实时软件案例 7：一个存放在永久存储器（例如 ROM）中的、代表了 FUR 中的图或表的引用数据结构，并且对于待度量软件中的大多数功能处理来说是可访问的。

实时软件案例 8：文件，通常称为"平面文件"[24]，体现 FUR 中提到的兴趣对象，存放在持久存储介质中。

3.3.3 不适合作为数据移动候选对象的数据或数据组

任何出现在输入输出屏幕或报告中、与功能用户兴趣对象无关的数据，都不应识别为数据移动的对象，因此不应该被度量。

业务软件案例 1：出现在屏幕上的应用程序常规信息，如标题和脚注（公司名称、应用程序名称、系统日期等等）。

业务软件案例 2：功能用户用来控制软件使用，而不是用来移动数据的"控制命令"（仅在业务应用软件领域定义的一个概念），如向上/下翻页命令，点击"OK"响应一条错误信息等等——详细信息参见 3.5.10 节。

COSMIC 通用软件模型假设功能处理内部的所有数据运算都对应着四种数据移动中的一种（参见 3.5.6 节）。因此，一个功能处理中除了输入、输出、读和写之外，不能识别出其它与数据组有关的移动或运算。（关于数据的运算和移动可能被错误解释为数据移动的例子，参见 3.5.4 节"读"的原则 c，3.5.5 节"写"的原则 d）。

3.3.4 功能用户作为兴趣对象

正如兴趣对象定义中的注 2 所指出的，待度量软件的功能用户也可以作为向功能用户发送数据或接收来自功能用户数据的兴趣对象。

在许多实时软件系统中，例如 3.3.2 节提到的实时软件案例 1，在策略阶段一个物理设备可以是功能用户，如传感器，然后，在映射阶段它也可以被识别为一个兴趣对象。这只是因为物理设备在"与软件交互"的同时也是"软件要处理和/或存储数据的一个事物"。实际上，物理设备以数据组的形式提供或接收自身的某方面信息，跨越边界与软件交互。

[24] 译者摘自百度百科：平面文件(flat file)是去除了所有特定应用（程序）格式的电子记录，从而使数据元素可以迁移到其他的应用上进行处理。这种去除电子数据格式的模式可以避免因为硬件和专有软件的过时而导致数据丢失。 平面文件是一种计算机文件，所有信息都在一个信号字符串中。

在业务应用软件中也是如此，人类功能用户可以作为向人类用户发送或接收来自人类用户数据的兴趣对象。

业务软件案例：一个人类功能用户在登录功能处理中向系统输入了 ID 和密码。所输入数据组的兴趣对象是人类用户。

实时软件案例：假设温度传感器 A 发送某物质的当前温度给某功能处理。从这个角度看，传感器可被认为在提供关于自身状态的信息。因此，传感器符合被识别为兴趣对象的条件，可从 FUR 中被映射为兴趣对象。但是，物理传感器属于用户一方，而且“跨越边界与软件交互”，因此它也符合被识别为功能用户的条件，应该出现在软件环境图中。

3.4 识别数据属性（可选）

本节讨论待度量软件块使用到的数据属性的识别。在 COSMIC 方法中，并不强制要求识别数据属性。但是，弄清楚“数据属性”的概念对理解“度量变更”章节的内容是必要的（当 FUR 变更了一个数据属性时，会产生待度量的数据移动）。而且，在区分数据组和兴趣对象的过程中，分析并识别数据属性是有好处的，如果需要规模度量元的子单位[25]，那么也需要识别数据属性，参见 4.5 节“扩展 COSMIC 度量方法”。

3.4.1 数据属性举例

（关于数据属性的定义，见 3.3.1 章节）。

业务软件案例 1：一个“员工”兴趣对象，描述它的数据组可能是“员工主要信息”，所包含的数据属性有“员工 ID”，“姓名”，“地址”，“出生日期”，“性别”，“婚姻情况”，“社会保险号码”，“级别”，“职位”等。

实时软件案例 1：一个热传感器可以响应一个请求而报告其属性“温度”。安全系统的传感器可以检测入侵者并发送属性“检测到移动”。传输的某个消息可能由属性“来源，去向，内容”组成。

3.4.2 关于数据属性和数据组的聚合[26]

理论上讲，从功能性用户需求的角度来看，如果一个属性足以描述兴趣对象的话，那么，数据组就可能只包含一个数据属性。实践中，这样的情况通常出现在实时应用软件中（如“输入”传送了一个实时时钟的节拍或读一个传感器），而在业务应用软件中不多见。

3.5 识别数据移动

本步骤的内容包括识别每个功能处理的数据移动（输入、输出、读和写）。

---

[25] 译者注：比如米的子单位是分米。

[26] 译者注：association 此处翻译为了聚合，意为数据组与属性的整体部分关系。

3.5.1 数据移动类型的定义

定义——数据移动
移动单个数据组类型的基本功能构件。 注 1：数据移动类型有四种子类型，即：输入、输出、读和写（类型）。 注 2：基于度量目的，每种数据移动被认为负责了某些与之相关的数据运算。详见 3.5.6 节。 注 3：更准确地说，是一次数据移动（而不是数据移动类型）实际移动了一个数据组的实例（不是类型）。这种解释也适用于输入、输出、读和写的定义。

定义——输入(E)
输入 (E)是一种数据移动，将一个数据组从功能用户一侧跨越边界移动给需要它的功能处理。 注意：输入被认为负责了某些相关的数据运算（见第 3.5.6 节）。

定义——输出 (X)
输出 (X)是一种数据移动，将一个数据组从功能处理侧跨越边界移动给需要它的功能用户。 注意：输出被认为负责了与之相关的某些数据运算（见第 3.5.6 节）。

定义——读 (R)
读 (R)是一种数据移动，将一个数据组从持久存储介质移动到需要它的功能处理。 注意：读被认为负责了与之相关的某些数据运算（见第 3.5.6 节）。

定义——写 (W)
写(W)是一种数据移动，将一个数据组从功能处理内部移动到持久存储介质中。 注意：写被认为负责了与之相关的某些数据运算（见第 3.5.6 节）。

下图 3.5 描述了四类数据移动、其所属的功能处理和被度量软件的边界之间的整体关系。

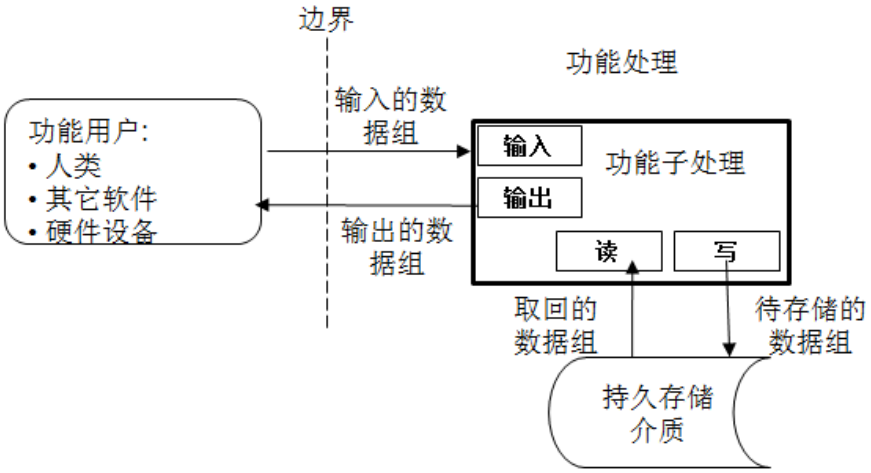


图 3.5-四类数据移动（各移动一个数据组）及与功能处理的关系（当然，一个功能处理可以有许多 E， X， R 和 W 数据移动）

3.5.2 识别输入(E)

候选的输入数据移动必须遵循以下原则：

原则——输入 (E)
a) 输入(E)应移动描述单个兴趣对象的一个数据组，从功能用户一侧跨越边界移动到功能处理内，输入(E)是该功能处理的一个组成部分。如果功能处理的输入包含多个数据组，每个数据组描述一个不同的兴趣对象，则为输入的每个数据组识别为一个输入(E)。（见 3.5.7 节“数据移动的唯一性”）
b) 输入(E)不跨越边界输出数据，也不从持久存储介质读取数据或向其写数据。

下列规则有助于确定一个候选的输入(E)数据移动的状态：

规则——输入 (E)
a) 触发输入的数据组可能仅由一个数据属性组成，这个触发输入只是为了通知软件“事件 Y 发生了”。大多数情况下，尤其是业务应用软件，触发输入的数据组有若干个数据属性，通知软件“事件 Y 发生了，这是与该特定事件相关的数据”。
b) 时钟节拍作为触发事件，总是位于待度量软件之外。因此，举例来说，每 3 秒发生一次的时钟事件应该与输入仅有一个数据属性的数据组相关。注意，此周期性触发事件无论是由硬件产生的，还是被度量软件边界外的软件产生的，这两者没有区别。
c) 除非需要一个特定的功能处理，否则从系统时钟获得日期和/或时间不可以被认为是一个输入或任何其他数据移动。
d) 如果一个特定事件的发生引发了一个输入，该输入的数据组包含了特定兴趣对象的“n”个数据属性，而 FUR 允许相同事件的另一次发生引发另一个输入，而该输入的数据组只包含兴趣对象“n”个属性的一个子集。那么，此处只能识别为一个输入，包含了所有“n”个数据属性。
e) 人类功能用户通过屏幕向功能处理输入数据，在识别这样的屏幕输入时，只分析填充了

数据的屏幕。忽略那些除了缺省值以外均是格式化内容的“空白”屏幕，以及帮助人类用户理解所需输入数据的所有字段和其他标题。

注意：当度量 FUR 中输入的规模变更时，有可能需要考虑此类字段及其他标题的变化——请看 4.4.1 节。

说明规则 c)的一个业务软件案例：当一个功能处理在记录中增加一个时间戳以使其永久保存或输出时，获取系统时钟值不被识别为一个输入。

一经识别出来，每个输入数据移动就可以登记到通用软件模型矩阵（附录 A）中，在对应的单元格中标注上“E”。

3.5.3 识别输出 (X)

候选的输出数据移动必须遵从下列原则：

原则——输出 (X)

a) 输出(X)将描述单个兴趣对象的一个数据组从功能处理一侧跨越边界移动到一个功能用户处。如果功能处理的输出不止包含一个数据组，那么，输出的每一个数据组都识别为一个输出(X)（参见 3.5.7 节“数据移动唯一性”） 。

b) 输出(X)不应跨越边界输入数据，也不应从持久存储介质读取数据或写入数据。

下列规则有助于确定一个候选的输出(X)数据移动的状态：

规则——输出 (X)

a) 对输出固定文本的查询应识别一个输出(X)（“固定”指消息不含可变数据值，比如在一个购物网站上点击“条款声明”按钮所得到的结果）。

注意：关于“帮助”功能的输出，参看“业务应用软件规模度量指南”。关于错误状态或成功确认的相关信息的输出，参看本度量手册的 3.5.11。

b) 如果功能处理的一个输出移动的数据组包含一个特定兴趣对象的“n”个数据属性，而 FUR 允许功能处理有这样一个输出，该输出移动的数据组仅含有兴趣对象“n”个属性的一个子集。那么，只能识别为一个输出，该输出移动一个数据组包含所有的“n”个数据属性。

c) 在识别输出时，忽略所有帮助人类用户理解输出数据的字段和其他标题。

注意：当度量 FUR 中输出的规模变更时，有可能需要考虑此类字段及其他标题。——请看 4.4.1 节。

一经识别出来，每个输出数据移动就可以登记到通用软件模型矩阵（附录 A）中，在对应的单元格中标注上“X”。

关于错误信息如何识别输出数据移动，请看 3.5.11 节。

3.5.4 识别读 (R)

候选的读数据移动必须遵从下列原则：

原则——读 (R)

a) 读(R)将描述单个兴趣对象的一个数据组从持久存储介质移动到功能处理，读(R)是该功能处理的一部分。如果功能处理必须从持久存储介质中检索一个以上的数据组，那么，为每个检索的数据组识别一个读(R)。（参见 3.5.7“数据移动的唯一性”。）

b) 读(R)不应跨越边界接收或输出数据，也不向持久存储介质中写数据。

c) 在功能处理执行中，计算或移动常量、功能处理内部的并且只能由程序员更改的变量、计算过程的中间结果、由执行功能处理而产生并存储的而非来自于用户功能需求的数据，这些都不应该被看作读数据移动。

d) 读(R)数据移动总是包含某些“读请求”的功能（所以任何“读请求”都不会被计算为单独的数据移动），见 3.5.9 节。

下列规则有助于确定候选的读(R)数据移动的状态：

规则——读 (R)

a) 根据 FUR，当被度量软件必须从持久存储介质中检索一个数据组时，识别为一个读。

b) 当被度量软件的 FUR 指定任何软件或硬件功能用户作为数据组的来源或者作为对已存储的数据组的检索手段时，不应识别为读[27]。（关于此案例，请看输入和输出的原则和规则。）

一经识别出来，每个读数据移动就可以登记到通用软件模型矩阵（附录 A）中，在对应的单元格中标注上“R”。

3.5.5 识别写 (W)

候选的写数据移动必须遵从下列原则：

原则——写 (W)

a) 一个写(W)将描述单个兴趣对象的一个数据组从功能处理移动到持久存储介质，写(W)构成该功能处理的一部分。如果功能处理必须移动不止一个数据组到持久存储介质，移动到持久存储介质的每个数据组都单独识别为一个写(W)。（参见 3.5.7“数据移动的唯一性”。）

b) 一个写(W)不跨越边界接收或输出数据，不从持久存储介质中读数据。

c) 从持久存储介质删除一个数据组的需求应被度量为一个写(W)数据移动。

d) 以下情形不认为是写(W)数据移动：
   - 在功能处理开始时不存在并且在功能处理完成后也没有持久化的数据的移动或运算。
   - 功能处理内部变量的生成、更新或中间结果。
   - 功能处理的数据存储是实现导致的而不是 FUR 所要求的（如批处理作业中执行一个大的排序处理时做了数据的缓存。）

[27] 译者注：此规则意味着不要把输入和输出识别为读。

下列规则有助于确定候选的写(X)数据移动的状态：

规则——写 (W)
a) 根据 FUR，当被度量软件移动一个数据组到持久存储介质中时，须识别为一个写。 b) 当被度量软件的 FUR 指定任何软件或硬件功能用户为数据组移动的目的地或作为存储数据组的手段时，不应识别为写。（关于此案例，请看输入和输出的原则和规则。）

一经识别出来，每个写数据移动就可以登记到通用软件模型矩阵（附录 A）中，在对应的单元格中标注上“W”。

3.5.6 与数据移动关联的数据运算

正如通用软件模型的原则 d)（见 1.3 节）所定义的，子处理要么是数据移动，要么是数据运算。然而，按 COSMIC 的约定（见通用软件模型的原则 j）），不识别数据运算子过程的独立存在。

所有数据运算都被认为是由相关的数据移动负责的。因此数据运算可以被忽略，除非需要度量 FUR 中某个数据运算的变更。对于这种情况，我们需要一个规则来决定哪种类型的数据运算应该与哪种类型的数据移动相关联。

一个典型的变更请求会同时影响某个特定数据移动所移动的数据属性和数据运算。对 FUR 变更的度量规则（参见 4.4.1 节“修改功能”的定义）是：如果一个数据移动所移动的数据组中的任意属性或关联的数据运算发生变更，那么该数据移动应该被识别并度量为一个变更。所以，如果一个变更请求只影响到数据运算，那么我们需要遵循规则来识别相关被变更的数据移动进行度量。

定义——数据运算
一个功能处理在处理数据时，除了进/出功能处理的数据移动，以及在功能处理和持久存储介质之间的数据移动之外，发生在数据上的其他任何行为。

以下原则决定了 COSMIC 方法如何处理数据运算。

原则——与数据移动关联的数据运算
功能处理中所有数据运算都与四类数据移动（E，X，R 和 W）关联。按照惯例，功能处理的数据移动假设也负责了功能处理的数据运算。

规则——与数据移动关联的数据运算
a) 输入数据移动负责使得数据组能被功能用户输入数据和确认数据的所有数据运算（如格式化和显示）。 b) 输出数据移动负责创建数据组属性的所有数据运算，此数据组用于输出和/或使得数据组能够被输出（如格式化和显示），并能被发送到预期的功能用户。

c) 读数据移动负责为从持久存储介质上检索数据组所需的所有计算和/或逻辑处理。

d) 写数据移动负责为创建或更新待写数据组或删除数据组所需的所有计算和/或逻辑处理。

e) 与以上任何数据移动相关联的数据运算不包括成功完成数据移动后所需的数据运算，也不包括任何与其他数据移动相关联的数据运算。

*业务软件案例 1：一个输入包括为使人类用户能够输入数据并确认所输入数据而对屏幕进行格式化的所有运算，但**不包括**确认某些输入的数据或代码，或获取相关代码描述而需要的读。*

*业务软件案例 2：一个输出包括为了格式化输出和为打印（或输出在屏幕上）准备某些数据属性而进行的所有运算，包括人类可读的标题栏[28]，但**不包括**为提供其中一些打印数据属性的值或描述而需要的读或写。*

3.5.7 数据移动的唯一性和不常见的情况

：通常情况下，任何一个功能处理中，描述其所需的兴趣对象的*所有*数据通常是在输入数据移动类型中进行输入，在输出数据移动类型中输出，在读数据移动类型中读，在写数据移动类型中写。模型还进一步假设，被移动数据组的数据属性的所有可能值引发的所有数据运算与一个数据移动相关联。同样的，两个数据移动，如果有 FUR 定义了它们关联了不同的数据运算，也不能被识别为不同的数据移动，因为“功能处理中所有数据运算都与四类数据移动（E，X，R 和 W）关联”（参见 3.5.6 节的原则）。

说明后一个原则的案例：以一个（种）功能处理的两次执行为例。假设第一次执行时，被移动的一些属性的值引发一个数据运算子过程（-类型）“A”，而同一功能处理在另一次执行时，属性的值引发另一个的数据运算子过程（-类型）“B”。在这种情况下，两个数据运算子过程“A”和“B”应该与同一数据移动关联，因此在这个功能处理中只有一个数据移动被识别和计数。

然而，也有例外的情况，比如描述了一个特定兴趣对象的不同数据组类型可能需要被同一个功能处理的同一类数据移动（E，R，W，X）移动。

以下规则包含了常规情况（规则 a）），以及其他可能的有效情况（规则 b）和 c))。规则 d）是关于数据移动的多个实例（而不是多个类型）。

规则——数据移动的唯一性和可能的例外

注意：所有 COSMIC 规则关注的都是功能用户、数据组、数据移动、功能处理和兴趣对象的各种*类型*。为了便于阅读，我们一般省略这些短语中的“类型”。但是在下面的规则 d)中，我们保留了“类型”这个词，以便区分“类型”和“实例”。

a) 除非功能性用户需求定义了规则 b)或 c)的情况，否则描述要输入到一个功能处理的任意一个兴趣对象的所有数据，应被识别为一个输入所移动的数据组。

注：一个功能处理当然可以有多个输入，每一个移动的数据描述了不同的兴趣对象。

相同的等价规则适用于任意一个功能处理的读、写或输出数据移动。

[28] 本案例适用于度量供人类使用的应用软件，不论其所属领域为何。而用来度量可重用对象组件显然不合适，比如支持输入/输出屏幕字段标题显示的对象组件。

b) 如果功能用户需求明确定义了需要输入到一个功能处理中的、来自于必须被该功能处理单独识别的功能用户的、描述了同一个兴趣对象的不同数据组，针对每个不同的数据组应该识别一个输入。

相同的等价规则适用于从一个功能处理输出数据到不同的功能用户。

注意：任何一个功能处理应该只有一个触发输入。

c) 如果功能用户需求明确定义了从持久存储介质移动不同的数据组到一个功能处理中，每个数据组描述的是同一个兴趣对象，此时应该为每个不同的数据组识别不同的读。

相同的等价规则适用于任意给定功能处理的写。

注：这条规则类似于规则 b），如果在 FUR 中指定对同一兴趣对象的读取区分为不同数据组，它们很可能最初来自不同的功能用户。如果在 FUR 中指定写区分为不同的数据组时，它们可能是被不同的功能用户存储进去的。

d) 任何在执行过程中重复出现的数据移动类型，不应被计数。

这一条也适用于如下情况：数据移动类型发生多次不同的执行，是因为所移动数据组的数据属性取值不同而导致在该功能处理类型中采取了不同的处理路径。

以下案例说明上述规则。

针对规则 a)的业务软件案例 1:正常的情况是：将包含一个兴趣对象所有数据属性的数据组进行持久化的数据移动，识别为一个写。

针对规则 c)和 a)对比的业务软件案例 2：假设 FUR 要求一个功能处理 A 从银行的资金账户文件检索两个数据组给另外两个各自独立的程序稍后使用。第一个数据组是“透支账户明细”（其中包括负帐目属性）。第二个数据组是“高价值账户明细”（其中仅有开户人姓名和地址，用于市场推广邮寄广告）。功能处理 A 将有 2 个写，每个数据组一个，都描述了同一个兴趣对象“账户”。

针对规则 b)的实时软件案例 3：一个功能处理要接收来自不同地震检波器（功能用户）的不同数据组。这两个数据组各自描述同一兴趣对象（事件）的数据，如一个爆炸试验，应识别两个输入。

针对规则 b)的业务软件案例 4：假定 FUR 定义了一个功能处理，会产生两个甚至更多的输出来移动同一个兴趣对象的不同数据组，给不同的功能用户。例如，当公司有一个新员工加入时，生成一份报告给批准该员工个人数据的人员，并发送一个消息到安保部门授权该员工进入大楼。应识别两个输出。

针对规则 c)的业务软件案例 5：假定一个程序的 FUR 要求将描述同一兴趣对象的两个永久数据文件进行合并，例如一个文件描述兴趣对象“X”的已有数据，另一个文件描述同一兴趣对象“X”的新定义属性，则识别两个读。

针对规则 c)的业务软件案例 6：假定 FUR 要求有一个数据组的读，但开发人员决定这样实现：在功能处理中的两个不同点，分别用两条命令从持久存储介质中检索同一兴趣对象的不同数据属性子集，只能识别一个读。

针对规则 d)的业务软件案例 7：见 3.5.2 节，规则 d)关于输入，3.5.3 节，规则 b)关于输出。（这两条规则是关于输入或输出只移动所有数据类型属性个数中的一个子集的情况，根据其 FUR。）

针对规则 d)的业务软件案例 8：假定 FUR 要求执行一个读，该读操作需要检索多次，如搜索一个文件里的内容。只识别一个读。

针对规则 d)的实时软件案例 9：假设一实时功能处理的 FUR 要求同一数据必须由给定的功能用户（如一硬件设备）输入，在固定的时间间隔内输入两次，目的是度量运行过程中的变更率。所以这两次数据移动是同一个输入的两次执行。在这个功能处理的这个数据组中只识别为一个输入。（没有数据运算与这两次输入相关联。变更率的计算必须与报告此值的输出相关联。参见 3.5.6 节与读相关的数据运算类型。）

针对规则 b)和 d)的实时软件案例 10：假定有一台生产平面产品（例如纸张或塑料薄膜）机器的过程控制系统。该机器有 100 个相同的传感器阵列分布在产品传送方向上，用于检测产品上的裂纹或孔，也就是说，每个传感器都是独立的、能区分开的。对于此功能处理从所有传感器获取的数据只识别一个输入。

3.5.8 当功能处理要求从持久存储介质中移入或移出数据时

当 FUR 中要求向存储介质存储数据或从存储介质检索数据时，度量人员必须确认是否数据的存储或检索是在其边界内，即向/从“持久性存储介质”，或者数据的存储/检索是否需要待度量软件的功能用户的帮助（即通过其他软件块，或直接向/从硬件设备）。本节通过以下四个实例，描述当软件块的功能处理要求从持久存储介质获取数据时的数据移动，

- 案例 1 是关于一个典型应用软件的实例，该例也可以应用于任何层的软件，但不能应用于软件与物理硬件设备存储直接交互的层。功能处理要从持久存储介质中检索数据，但 FUR 并不关心数据存取是如何被其他软件处理的。
- 案例 2 是关于一个“客户—服务器”架构的软件。在此软件中，客户端的功能处理必须请求服务器提供持久数据。
- 案例 3 展示了不同的软件具有对持久数据的不同访问权限。

在案例 4 中，数据必须直接从一个物理硬件存储器中获得，可能经由某设备驱动软件。

这些案例用消息顺序图的约定来展示。图例说明如下：

- 向下指向的加粗箭头表示一个功能处理。
- 水平箭头表示数据移动，上面标注 E、X、R 或 W 分别表示输入、输出、读和写。输入和读被表示成指向功能处理的箭头，输出和写表示成从功能处理发出的箭头；尽可能按照功能处理的要求顺序，从顶部到底部排列。
- 垂直的虚线表示一个边界。

案例 1：当功能处理要求从持久存储介质中移入或移出一个数据组时。在这个案例中，软件块 A 要求检索一个已保存的数据组，但软件 A 的 FUR 并不关注处于同一层或不同层的其他软件怎样处理这些数据访问。

例如，如果软件 A 位于应用层，对一个已存储的数据组进行查询，那么软件 A 的功能用户可以是人类用户。图 3.6 展示了此查询的 COSMIC 数据移动。被一个输入触发的查询，紧跟着一个来自持久存储介质的数据组的读，然后是带有查询结果的输出。FP A 不关心从哪里检索的数据，只需要知道它是持久数据。

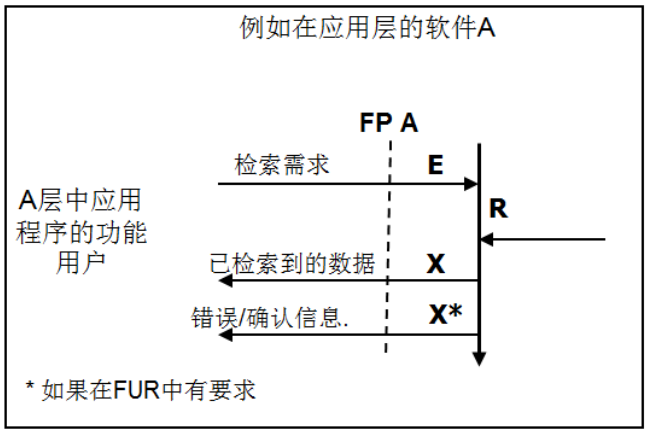


图 3.6-应用层软件“A”发起一个读操作的解决方案

如果功能处理 FP A 要求通过一个写数据移动使一个数据组持久化，那么可以应用一个完全类似的模型。按照针对错误状态的规则 d)(参见 3.5.11 节)，读和写数据移动被认为要包含任何返回代码或错误状态报告。

图 3.6 展示了如果无法找到要求的记录，FP A 可能发出一个应用程序特有的错误信息。但是，如果错误状态并不是应用程序特有的，如“磁盘故障”则不计作 FP A 的输出。请参见 3.5.11 节中关于错误/确认信息的章节内容。

案例 2：当功能处理要求从另一个软件块获取数据时。

在这个实例中，假设待度量的软件有“客户/服务器”关系，也就是说，客户端从位于同一层或不同层的服务器获取服务和/或数据，图 3.7 显示了这种关系的一个案例，该案例中两个软件块是同一应用程序的两个主要构件。在任何一个客户端/服务器关系中，客户端构件 C1 的 FUR 会识别服务器构件 C2 为其的一个功能用户，反之亦然。如果两个软件块是独立的应用程序，或其中一个软件块是另一个独立应用程序的一部分，那么也同样存在这种关系，并且同样适用此图。

物理上，两个构件可以分别运行在不同的处理器上；在此情况下，它们将通过各自的操作系统和任意的其他中间层交换数据，图 2.2 描述了此类软件架构。但在逻辑上，应用 COSMIC 模型时，这两个构件是通过一个输出紧跟着一个输入来交换数据。模型中所有的中间硬件和软件都被忽略（类似的案例请看图 3.1 右边所示）。

图 3.7 显示客户端构件 C1 的一个功能处理 FP C1 被来自其功能用户（如人）的输入所触发，例如该输入包含了查询参数。构件 C1 的 FUR 将识别出该构件必须向服务器构件 C2 请求查询数据，而且必须告知它需要什么数据组。

为了获得需要的数据组，FP C1 发起一个包含查询请求参数的输出给 C2。这个输出数据移动跨越 C1 和 C2 的边界，成为构件 C2 中的功能处理 FP C2 的触发输入。构件 C2 的功能处理 FP C2 被假定通过一个读从其自身的持久存储介质获得要求的数据组，再通过一个输出将数据组返回给 C1。构件 C1 的功能处理 FP C1 接收这个数据组作为一个输入。FP C1 之后用一个输出传递数据组给功能用户以满足查询请求。

考虑到客户可能发送的错误/确认消息，为满足案例 2 的查询请求需要构件 C1 的 5 个数据移动（即 5CFP）和构件 C2 的 3 CFP。相较之下，如果 C1 能够通过一个读从自身边界内

的持久存储介质检索数据组，则只需要4 个数据移动（1 x E， 1 x R 和 2 x X），如图3.7 所示。

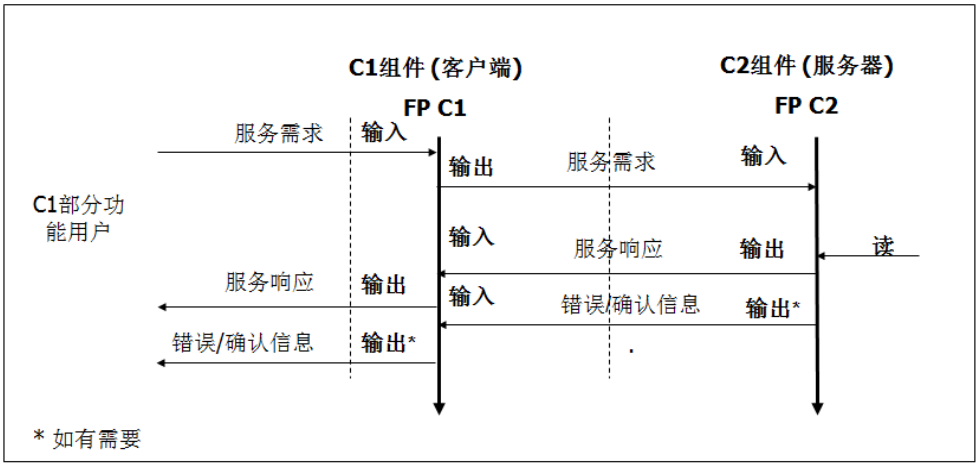


图 3.7-客户端和服务器构件之间的数据交换

当然，构件 C2 可能使用软件架构中其它层的某个存储设备驱动软件的服务来从硬件一侧检索数据，如例4 中图3.9（b）所示。

案例 1 和案例 2 分别描述了以下两种情况的数据移动：第一种是 FUR 清晰地指明待度量软件必须访问其边界内的持久存储介质，以及对应的第二种必须把访问请求传递给其边界外的其他软件块。但是，根据具体的待访问数据属性和/或访问的类型（存储或检索），有时待度量软件块可能不得不使用不同的“路线”去访问持久数据。出于如安全、隐私（参见后面的案例 3）或通过限制创建、更新和删除过程的存取来确保数据完整性的需要，当被度量软件对数据的访问权受制于不同的规则或“权利”时，以上提到的情况便会发生。当被度量软件的 FUR 在此处存在不清晰的地方，度量者必须谨慎定义实际的访问权限。（不要把数据的“访问权限”与数据“所有权”混淆，后者与 COSMIC 模型无关。持久存储介质并不被任何软件所“拥有”。）

注：图 3.7 的案例假设客户端和服务器端是同步通信，即客户等待服务器端的响应。客户端和服务器端也可以不同步，那么这个案例的建模将会稍有调整，但是数据移动的个数与图 3.7 相同。（见“业务应用软件规模度量指南”[7]）

注：实际上，客户端-服务器端的通信时长可能会受到监控，以防止服务器端没有在指定的时间段内响应。对这种情况如何进行建模和度量，请见“实时软件规模度量指南”[4]。

案例3：根据不同目的，软件对已存储数据有不同的访问权限。

参看图3.8。待度量软件块A 被允许检索某个已存数据Z（如案例1 中的图3.6），但它没有权限直接维护（即创建、更新或删除）此数据Z。当软件A 被要求维护数据Z 时，软件A 必须通过一个输出紧跟着一个输入把需求传递到另一个软件块 B（类似于案例 2，图 3.7 中，构件C1 把需求传递到构件C2）。软件块B 被要求通过一致性确认来保证数据Z 的完整性，所以它处理数据Z 的所有数据维护请求。

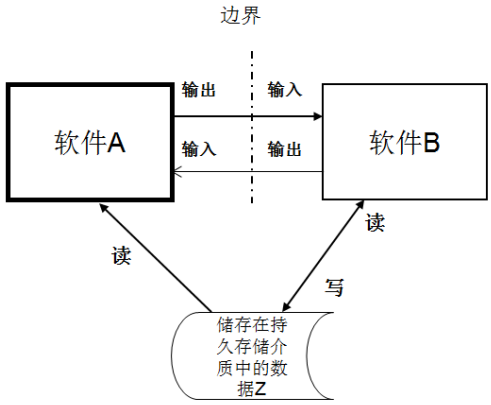


图 3.8-在软件 A 和软件 B 的共同边界内用于读取的持久数据 Z

在案例 3 中，COSMIC 模型显示数据 Z 位于软件 A 边界内的持久存储介质中，但只在出于检索目的它才能被读数据移动访问。对于软件 B 而言，同一数据 Z 位于其边界内的持久存储介质中，但软件 B 能同时读和写数据 Z。关于此例中的错误处理，请看上文的案例 1 和案例 2。

基础设施软件案例 4：与物理存储设备交互的设备驱动软件是如何获取持久数据的。

此实例涉及例 1 中需要检索已存储数据组的软件块 A。此外，也涉及智能硬件存储介质的设备驱动软件块“B”，该软件保存着软件块 A 要访问的数据组。（简单起见，我们忽略可能存在的操作系统，操作系统传递应用软件的请求到设备驱动软件，返回请求的结果。）

如图 2.2 所示，这两个软件块在同一架构中的不同层内。例如软件 A 在应用层，而软件 B 在设备驱动层。物理上，两个软件之间很可能存在层次关系（忽略操作系统），并且两个层上的软件之间存在一个物理接口，如图 2.2 所示。但是，软件 A 和 B 的功能处理模型本质上是独立的不同层之间的关系，可能是层次关系或是双向关系。

驱动层中软件 B 的功能用户是软件 A（忽略操作系统）和存有所需数据的智能硬件存储设备（“智能”指设备必须被告知需要的是什么数据）。

假设软件 A 的一个查询功能处理 FP A 需要检索一个已保存的数据组。图 3.9(a) 显示了这个查询的 COSMIC 模型。图 3.9(b) 显示了位于设备驱动层的软件 B 的功能处理 FP B 如何从硬件存储设备（如磁盘或 USB 记忆棒）上检索所需要的数据。

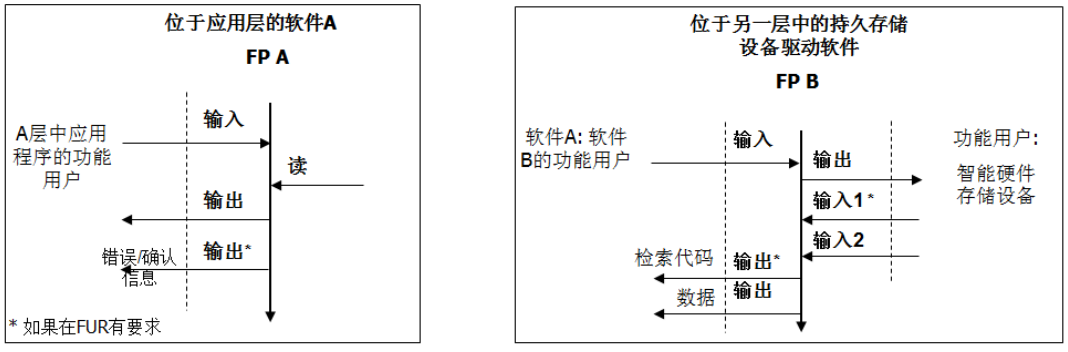


图 3.9(a)和(b)-应用层软件 A 发起一个读操作到设备驱动层软件 B 的解决方案

图 3.9（b）显示软件 A 的读请求被功能处理 FP B 接受并作为触发输入，FP B 然后把请求作为一个输出传送到硬件设备。后者的响应取决于具体的硬件设备。设备可能仅返回被要

求的数据，如图3.9b)中的输入2所示。设备也可能会发布一个单独的错误信息，此信息描述请求成功或者请求失败的原因，如“数据不存在”或者“磁盘错误”，如图3.9b)中的输入1所示。FP B把数据作为一个输出返回给软件A。通常情况下，FP B也会发布一个描述请求成功或失败原因的“返回码”。（尽管物理上，返回码附属于已返数据，但逻辑上它是一个区别于返回数据的不同的数据组——它是关于请求处理结果的数据）。根据输入的规则d)，由于是读数据移动负责返回数据和错误信息，所以对于FP A来说，不用识别这些消息的输入。另外，如果有要求的话，对于FP A来说，错误/确认消息被识别为一个输出。

注意：在实践中，设备驱动软件和智能硬件设备间的数据移动可能比图3.8b)所显示的多。例如，此图没有显示设备驱动程序测量硬件超时无响应的影响。

比较案例 2 和 4，我们看出在案例 4 中，软件块 A 和设备驱动程序 B 的模型不能像案例 2 那样组合在一起。这是因为 A 和 B 在不同层，而且读操作不能跨越边界。图 3.9(b)显示软件 A 是设备驱动软件 B 的功能用户。但反过来却不成立，因为读不能跨越边界。相反，图 3.7 能在一个模型中展示两个部件，因为 C1 是 C2 的功能用户，反之亦然，它们共享一个公共边界。

3.5.9 当功能处理从功能用户处获取数据时

如果功能处理必须从功能用户处获取数据，有两种情况。如果功能处理不需要告知功能用户发送什么数据，只需要一个输入就够了（针对每一个兴趣对象）；如果功能处理需要告知功能用户发送什么数据，就需要一个输出跟着一个输入。应用下列规则：

规则——功能处理从功能用户处获取数据

a) 当功能处理不需要告知功能用户发送什么数据时，功能处理会通过来自功能用户的输入数据移动获取一个数据组，如下面四个情形之中的任意一种：
   - 功能用户通过触发输入发送一个数据组启动功能处理；
   - 接收到触发输入发送的数据组的功能处理处在等待状态，等待来自功能用户输入的下一个数据组 (这种情况在业务应用软件中会发生，由人类用户输入数据给软件)；
   - 当功能处理已经启动，向功能用户请求“如果你有数据，现在就发给我”，功能用户即发送数据；
   - 当功能处理已经启动，随即检查功能用户的状态并检索其需要的数据。

   对于后面两种情形（通常发生在实时“轮询”的软件中），按照惯例，获取所需的数据请求不应识别为来自于功能处理的输出。功能处理仅仅需要发送一个提示信息让功能用户输入数据，提示信息这一功能被认为是输入的一部分。功能处理知道等待什么数据。在这案例中只需要识别一个输入。

b) 当功能处理需要获取功能用户的服务（如获取数据），并且功能用户需要被告知应该发送什么时（典型的情况是，功能用户是被度量软件范围之外的另一个软件），将识别为一个输出跟着一个输入。输出发送对特定数据的请求，输入接收返回数据。

针对规则a)的实时软件案例1，第三或第四公告：假设实时过程控制应用软件系统的一个功能处理被要求轮询一组相同的传统传感器。在应用层，功能处理对数据的需求以及数据的接收算作一个输入（类型）。（因为传感器是相同的，只有一个输入（类型）被识别和计数，尽管发生多次。）

再假设在实践中对数据的需求必须被传递到软件架构下层的设备驱动软件，该软件实际从传感器组获取所要求的数据，如图 2.3 的层次体系结构所示。过程控制应用软件以及传统传感器的设备驱动软件的的功能处理将如下图3.10 (a) 和 (b)所示。

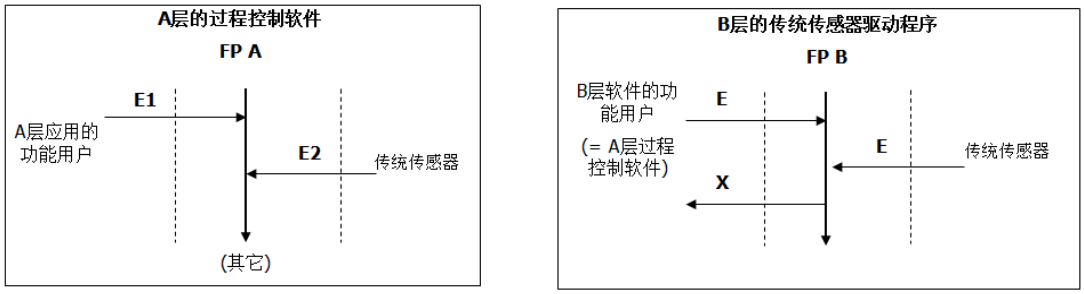


图 3.10 (a) 和 (b)-由过程控制软件层的软件 A 发送一组传统传感器到传统传感器设备驱动层的软件 B 进行处理的解决方案

图3.10（a）显示应用软件功能处理FP A 被来自例如时钟节拍的输入 E1 触发。功能处理随后通过输入 E2 获得来自传统传感器组的数据，接收到传感器的多次读数。传统传感器也是该应用层模型的过程控制软件的功能用户。（设备驱动软件隐藏在该层中。）

图 3.10（b）显示了驱动传统传感器设备的软件模型。它通过输入从应用程序接收数据（实践中可能是通过操作系统）作为功能处理FP B 的触发。该功能处理通过输入 E 从它的功能用户（传统传感器组）获得所需的数据。

数据组通过一个输出被回传给过程控制软件。这个输出作为输入 E2 被应用程序功能处理FP A 接收。FP A 随后继续处理传感器数据。同样地，从每个相同传感器收集数据的循环发生多次的这一事实与模型无关。

来自处理控制应用程序软件的传统传感器的一个输入 E2 和来自设备驱动软件的被一个输出紧跟着的输入，这两者之间明显不匹配， 因为按惯例，来自传统传感器的输入被认为包括任何“输入请求”功能（因为传统传感器功能用户没有处理来自功能处理的任何信息的能力）。

针对规则 b)的实时软件案例 2：假定功能处理发送一些查询参数或计算参数或要压缩的数据到它的一个功能用户，如一个“智能”硬件设备或另一对等软件块。功能用户的回应是通过一个功能处理发出的输出紧跟着一个输入数据移动的接收来获得的，如 3.5.8 节案例 2 所述。

3.5.10 为人类用户导航和显示的控制命令（“控制命令”）

在所有应用中，“控制命令”被认为是这样一种命令：它可能会被人类功能用户所使用，但当度量功能规模时必须被忽略。其定义是：

定义——控制命令
人类功能用户用来控制软件使用的命令，但不包含任何被度量软件 FUR 中关于兴趣对象的数据移动。 注意：控制命令不是数据移动，因为它不移动关于兴趣对象的数据。

规则——带有人机界面的应用领域的控制命令
带有人机界面的应用程序中“控制命令”应该被忽略，因为它们不包含任何关于兴趣对象的数据移动。

控制命令的例子：

- *物理屏幕的上下翻页，*
- *点击 Tab 或者 Enter 键，或点击某个按钮继续，*
- *点击“OK”确认或取消之前的操作，或响应错误消息或者确认已输入的数据等等，*
- *用户可以展示/隐藏一个标题或已计算好的小计，*
- *菜单命令，将用户导航到一个或多个功能处理，但是并没有实际启动某个功能处理，*
- *显示一个空白页面等待数据输入的命令。*

注意：在带有人机界面的应用领域之外，“控制命令”这一概念没有特殊意义，任何来自功能用户的关于兴趣对象的数据移动或信号必须被计数，即必须被度量。

3.5.11 错误/确认消息和其他出错状态的提示

定义——错误/确认消息
由功能处理发送给人类用户的输出，要么是确认输入数据已被接收，要么是提示输入数据存在错误。 注意：任何包含错误标志、但不是发送给人类功能用户的输出，都不是错误/确认消息。

规则——错误/确认消息以及其他错误状态的提示
a) 被度量软件的任何一个功能处理发布的所有各类错误/确认信息，根据其 FUR，不论什么原因都应被识别并归为一个输出，如关于以下事情的成功或失败：对输入数据的确认、检索数据，或使数据持久化的要求，或对其它软件块请求的服务响应。 注意：如果功能处理的 FUR 没有要求发出任何形式的错误/确认消息，无须识别相应的输出。
b) 如果一条发送给人类功能用户的消息，除了确认输入数据已被接受或输入数据有误，还提供了其他的数据，那么除了这个错误/确认输出外，这些额外的数据应被识别为一个由输出移动的数据组。
c) 根据标准的 COSMIC 规则，所有被度量的软件发送到或接收来自其硬件或软件功能用户的其它数据，不管数据值是否指示一个错误状况，都应根据 FUR 分别作为输出或输入来分析。
d) 读和写被看作负责报告与之关联的错误状态。因此，被度量的功能处理中，由于对持久数据读或写而接收到的任何错误提示不被识别为输入。
e) 在软件使用中，可能会发出错误状态信息，但该软件的 FUR 并没有要求对该错误状态

进行任何处理，如操作系统发布的错误消息。对于所有指示这类错误状态的消息，都不应识别输入或输出。

业务软件案例 1 针对规则 a)：人机对话中，在确认输入数据的过程中可能出现的错误消息的例子有“格式错误”、“客户不存在”、“错误：若已阅读我们的条款声明，请在复选框里打勾”、“超出信用上限”等。所有这些错误消息都应被看作该消息发生的每个功能处理的一个输出的多次出现（可命名为“错误消息”）。

业务软件案例 2 针对规则 a)：功能处理“A”可以潜在地向功能用户发布 2 条不同的确认消息和 5 条错误消息。识别一个输出来负责这 5+2=7 条错误/确认消息。功能处理“B”可以潜在地向功能用户发布 8 条错误消息。识别一个输出来负责这 8 条错误消息。

业务软件案例 3 针对规则 b)：银行 ATM 机（即自动提款机）的某功能处理，当提取特定金额的现金时，会有以下五类消息提示：

- *错误：机器内无现金。*
- *错误：输入的金额必须为$10 的倍数。*
- *提取现金失败。账户已锁，请联系银行。*
- *提取现金失败。（将会超出信用额度$139.14）*
- *提取现金成功。你的账户余额$756.25。*

前四类消息都描述的是错误的情况，而第五类消息的前半部分是确认信息。对于这几类输出，根据规则 a)，计数为 1 个输出。最后两类消息也包括了和用户账户有关的数据属性，根据规则 b)，需要计数为 1 个输出，并要考虑 3.5.7 节关于数据移动唯一性规则中的规则 a)。因此该功能处理总共识别 2 个输出，即 2CFP。

业务软件案例 4 针对规则 e)：输出给人类用户但没被度量软件生成或处理的错误消息应在度量应用程序时被完全忽略。此类消息的一个例子是从操作系统发出的 “X 打印机没有响应”。

实时软件案例 1 针对规则 c)：在一个实时系统中，定期检查所有硬件设备是否正常运作的功能处理可能会发布一条这样的消息报告“X 传感器失效”（X 为一个变量）。在该功能处理中，该消息应被识别为一个输出。

实时软件案例 2 针对规则 c)：3.5.9 节中的实时软件案例 1 的 FUR 可能也有如下描述：当设备驱动软件从一个或多个传统传感器组中获取数据失败时，FP A 和 FP B 必须处理这个错误状况。从定义上说，传统传感器不能发布错误消息。最有可能的是，设备驱动 FP B 将从传统传感器组获得一串数值，如状态 1、状态 2、状态 3、无响应、状态 5、无响应、状态 7 等等，然后它将这串数值输出给应用程序的 FP A 作为 FP A 的输入。不能将单独的错误消息识别为设备驱动软件的 FP B 的输出，也不能识别为发送到处理控制应用程序的 FP A 的输入。

第四章 度量阶段

4.0 章节概要

本章讨论度量过程的最后一个步骤。首先，定义了 COSMIC 度量单位（即一个数据移动被度量为一个 COSMIC 功能点，或“CFP”）。然后，列出了对被度量软件 FUR 进行规模赋值的相关规则，也定义了如何汇总各个软件块规模的规则。

此外，还定义了如何度量软件变更规模的规则（如：处理“增强型”项目）。章节的最后讨论当采用标准 COSMIC 方法时，进行“本地化扩展”的可能性。例如，某组织希望对某方面的功能制定一个在本地环境中有意义的本地标准。

4.1 度量阶段的过程

当软件块的功能性用户需求已经表示为 COSMIC 通用软件模型后，度量它的通用方法概括如下图 4.0。

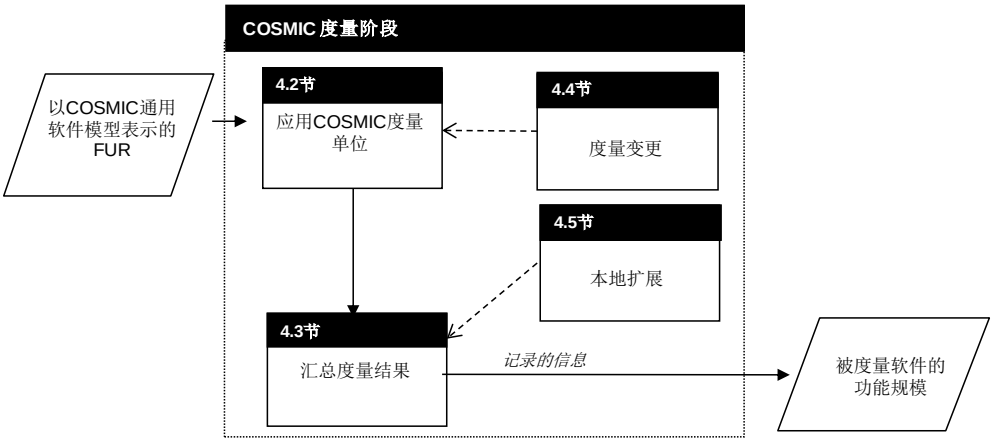


图 4.0-COSMIC 度量阶段的一般步骤

该方法的每个步骤对应本章节的一个特定小节的主题，每节中给出了定义和应用原则，以及一些规则和案例。

4.2 应用 COSMIC 度量单位

定义——COSMIC 度量单位
1CFP（COSMIC 功能点），被定义为一个数据移动规模。 注意：3.0 以前的版本中，度量单位称为“Cfsu”（COSMIC 功能规模单位）

依照此定义，被度量软件中要求增加、修改或删除的每个数据移动(输入， 输出， 读或写)也都度量为 1CFP。

4.3 汇总度量结果

本步骤汇总所有识别出的数据移动规模，形成一个功能规模值。本步骤依照下列规则来完成。

4.3.1 汇总的一般规则

规则——汇总度量结果

a) 对于任何功能处理，各个数据移动的功能规模应该通过累加的方式汇总为以 CFP 为单位的一个功能规模值。

规模 (功能处理 $_i$) = Σ 规模（输入 $_i$）+ Σ 规模（输出 $_i$） +
Σ 规模（读 $_i$）+ Σ 规模（写 $_i$）

b) 对于任何功能处理，其功能性用户需求中功能规模的变更规模如果以 CFP 为单位衡量的话，应该是功能处理中增加、修改、删除的数据移动的规模的汇总，采用以下公式计算。

规模 (变更(功能处理 $_i$)) = Σ 规模 (增加的数据移动 $_i$) +
Σ 规模 (修改的数据移动 $_i$) +
Σ 规模 (删除的数据移动 $_i$)

关于汇总功能规模的更多内容，参阅 4.3.2 节。关于对变更软件的规模度量，参阅 4.4.2 节。

c) 一个指定范围的软件块的规模是通过汇总该软件块功能处理的规模得到的，遵循下面的规则 e)和 f)。

d) 一个指定范围的软件块任意变更的规模是通过汇总该软件块所有功能处理的变更规模得到的，遵循下面的规则 e)和 f)。

e) 只有在 FUR 的同一功能处理颗粒度级别进行度量时，软件块的规模或者软件变更的规模才可以累加。

f) 只有对度量目的有意义时，同一层或不同层的软件规模或软件变更的规模才可以进行累加。

g) 一个软件块的规模是通过累加其构件（不管它是如何被分解的）的规模来获得的，并且构件间的数据移动需要被剔除。

h) 某个功能处理中发送给人类功能用户的所有错误/确认消息仅识别为一个输出。

i) 如果对 COSMIC 方法进行本地化扩展（例如要度量标准方法没有覆盖的某方面规模），那么经本地化扩展得到的度量规模必须如 5.1 节中描述的那样单独报告，不可以累加到通过标准方法获得的以 CFP 为单位的规模中去（详细信息参见 4.5 节）。

针对规则 b)和 c)的案例 1：可能有这样一个对软件的变更要求：“需要增加一个规模为 6CFP 的新功能处理，在另一个功能处理中增加一个数据移动、修改三个数据移动、删除两个数据移动。”此变更的全部规模为 6 + 1 + 3 + 2 = 12 CFP。

针对规则 f)的案例 2：如果软件的各主要部分由不同的项目子团队使用不同的技术开发，那么把它们的规模累加起来可能是没有使用价值的。

针对规则 g)的案例 3：如果一个软件块：

- *首先作为“一个整体”度量，即全部在一个范围内*
- *然后，每个构件单独度量，即每个部分在它自己的范围内*

那么，把各部件规模（情形 2）累加得到的规模将超过作为“一个整体”度量的规模（情形 1），差别在于部件之间的数据移动规模。部件之间的数据移动在软件作为“一个整体”度量时是不可见的。更多案例请参考早期快速 COSMIC 功能规模度量指南[6]中“在软件架构的多种颗粒度级别实施度量”的章节。

需要注意的是，在每一个已识别的层里，汇总函数是完全可计数的。因此，可以为一个功能处理或一层内的所有软件产生一个小计，这取决于每次度量实践的目的和范围，遵守上面的规则 d)、e)和 f)。

4.3.2 关于功能规模汇总的更多信息

在某些情况下，功能规模被用作一个模型的变量，如估算工作量时。待度量的软件有一个以上的层，汇总通常是对每个层进行，因为不同层的软件通常使用不同的实现技术。

案例 1：假如某个软件的应用层是使用第三代语言和已有的库来实现的，驱动层可能使用汇编语言实现。每层软件构造时单位规模的工作量很可能是不同的，因此，工作量估算将在各层分别进行。把两层软件的规模累加起来，不太可能有意义。

案例 2：如果项目组需要开发大量的软件块，并且对其整体生产率感兴趣，那么可以将每块软件的开发工时累加起来。同样的，当且仅当满足上面的规则时，项目组可以将已经开发的软件块的规模累加起来。

一个标准层次结构的不同层中的软件块，如果是在相同功能处理颗粒度级别上度量的，其规模可以进行累加，如果这么做有意义的话（比如所有的软件块都采用相同的开发技术），因为这样的结构具有紧耦合定义的功能用户集合。每个层的软件是它所用到的其他层的软件的功能用户，并且任何软件块可以是其他对等软件块的功能用户。因此，应用以上的规则 d)、e)和 f)累加各软件块的规模是合乎逻辑的。然而，相反地，一个软件块的规模不可以通过累加其可复用对象部件的规模来获得，除非按照规则 g)将对象间的数据移动剔除。

按照数据移动的类型汇总度量结果，对分析每个类型对给定层的软件总规模的贡献度有帮助，也可以帮助刻画被度量软件的给定层的功能特征。

4.4 关于软件变更规模度量的更多信息

现有软件的“功能变更”在 COSMIC 方法中解释为“对现有数据移动进行修改、删除、增加新的数据移动这几种方式的任意组合，包括相关联的数据运算”。术语“增强”和“维护”[29]通常用于这里所说的“功能变更”。

软件变更的需求可能来源于以下任意一种：

- 一个新的 FUR（即只在现有功能基础上进行增加），或
- 对 FUR 的变更（可能包括增加、修改和删除），或
- 为修正一个缺陷所需的“维护”。

[29] 度量的一般惯例是：如果为了使软件与其 FUR 一致而必须变更软件来修正一个缺陷，那么软件的功能规模没有发生变更。如果变更是为了修正 FUR 中的缺陷，那么软件的功能规模发生了变更。

这些变更的度量规则是相同的，但度量员在进行性能度量和估算时要注意区分各种情况。

当一个软件块完全被取代，例如重写它，扩展或不扩展其功能，和/或忽略其功能，那么，这种变更的规模就是替代软件的规模，按照度量新软件的正常规则进行度量。本节不对这种情况作进一步讨论。然而，度量人员应该知道在进行性能度量或估算时，需要区分全新开发软件的项目与“再次开发”或“替换”现有软件的项目。

通常，删除（用“分离”这个词可能更恰当）一个应用软件的淘汰部分，只是删除对它的调用，而代码仍然保留在那里。当淘汰部分的功能总计为 100CFP，而这部分可以通过 2 个数据移动的变更被分离，那么，此功能变更的规模应该识别为 100 而不是 2 个数据移动。我们度量的是需求的规模，而不是实现的规模。

注意，由于分离与“真正的”删除很不同，出于估算的目的，这一部分的功能变更建议使用不同的生产率。或者，出于度量目的，度量将被实现的规模（此例中为 2CFP）比度量需求的规模（此例中为 100CFP）更好。如果 作为“项目规模”度量为 2CFP，应该把它清楚地记下来，并与从 FUR 的度量角度所应减掉的 100CFP 的规模区别开。

需要注意此处讨论的功能变更的规模和软件功能规模的变化是有区别的[30]。通常他们是不同的。后者的规模在 4.4.2 节中讨论。

4.4.1 修改功能

任何给定类型（E、X、R 和 W）的数据移动包括两类功能：移动一个数据组，及其相关的数据运算（后者见 3.5.6 节）。因此，从度量目的看，一个数据移动在功能方面的修改如下所述：

定义——修改（一个数据移动的功能）
a) 如果以下情况至少一种适用，那么数据移动就是在功能上被修改了： ● 移动的数据组被修改了。 ● 关联的数据运算被修改了。 b) 如果以下情况至少一种适用，那么数据组是被修改了： ● 一个或多个新属性添加到数据组中。 ● 一个或多个现有属性从数据组删除。 ● 一个或多个现有属性被改变，例如意义或格式上（而不是它们的值）发生改变。 c) 如果发生任何功能性变化，一个数据运算就被改变了。

案例：一个数据运算通过以下方式被修改：改变计算方法、特定的格式化、显示方式，和/或数据的确认。“显示方式”包括如字体、背景颜色、字段长度、字段标题和小数位数等等。

控制命令和业务应用软件的程序通用数据是不包含数据移动的，因为没有对兴趣对象的数据移动。因此，控制命令和程序通用数据的改变不应该被度量。例如，当所有屏的颜色被改变了，这种改变不应该被度量。（关于控制命令和程序通用数据的解释见 3.5.10 节。）

规则——修改一个数据移动
a) 如果一个数据移动必须被变更（因为与数据移动关联的数据运算有变化，以及/或因为

[30] 译者注：变更的规模按上述的累加规则 b)，d)进行合计，规模的变化=增加的功能点-删除的功能点。

移动的数据组的属性数量或类型有变化），需要计算为一个变更的 CFP，而不管一个数据移动中的实际修改数值。

b) 如果一个数据组必须被修改，此修改并不影响该变更数据组的数据移动功能，则该数据移动就不应该识别为变更的数据移动。

注意：对输入或输出屏上任何数据的改变，如果不涉及功能用户的兴趣对象，则不能被识别为一个变更的 CFP（这类数据的例子见 3.3.4 节）。

针对规则 a)的案例：某个功能处理的变更请求要求对其触发输入的数据运算有 3 处变更，并且与输出有关的数据运算有 2 处变更，同时对这个输出的数据组有 2 个属性的变化。总的变更的规模是 2CFP，即对属性和关联的数据运算发生变化的数据移动的个数进行计数，不对数据运算或数据属性变化的个数进行计数。

针对规则 a)和 b)的案例：假设要求增加或修改数据组 D1 的数据属性，修改后变成 D2。此修改要求在功能处理 A 中进行，所有受修改影响的数据移动都应该被识别并计数为修改点。这样，按照原则 a)，如果功能处理 A 将数据组 D2 持久化和/或输出，就识别为一个写和/或一个输出数据移动的修改。然而，有可能其他功能处理读或输入 D2，但是它们的功能不受修改的影响，因为它们不使用修改或增加的数据属性。这些功能处理延续以前的方式处理该数据组的移动，就好像它仍然是 D1。这样，按照规则 b)，这些不受修改影响的其他功能处理的数据移动不能被识别并计数为修改点。

业务软件案例：如果一条错误/确认消息被要求变更（即文字上的增加、修改或删除），不管改变的文字是不是由于改变其它数据移动的需求所带来的结果，它也应该被识别和度量。

4.4.2 功能发生变更后软件的规模

规则——功能发生变更后软件的规模

软件功能改变后：

新的总规模（变更后的软件块）=原规模+Σ 规模（新增的数据移动）- Σ 规模（删除的数据移动）

修改的数据移动对软件的规模没有影响，因为它们在修改前后同时存在。

案例：回顾一下在 4.3.1 节的案例 1：可能有这样一个对软件的变更要求：“需要增加一个规模为 6CFP 的新功能处理，在另一个功能处理中增加一个数据移动、修改三个数据移动、删除两个数据移动。”此变更的全部规模为 6 + 1 + 3 + 2 = 12 CFP。”

该软件块的总规模因为新增了一个功能处理而增加 6CFP，同时因为在另一个功能处理中新增了 1 CFP 但是删除了 2CFP，进而总规模减少 1CFP。变更后，软件块的规模将会增加（+6-1）=5CFP。

4.5 扩展 COSMIC 度量方法

4.5.1 简介

度量功能规模的 COSMIC 方法并不假定能度量软件所有可能方面的规模。因此，COSMIC 度量方法当前没有设计成用来单独并明确地度量数据运算子过程的 FUR 规模。数据运算子过程的规模影响通过一个简化假设进行了考虑，该假设对大范围的软件领域有效，这在 1.1 节讨论方法的适用性时有定义。此外，该方法也没有捕获数据移动的数据属性数量对软件规模的影响。

其他参数如“复杂性”（不管怎样定义）可能被认为对功能规模有贡献。对此问题进行建设性的争论时，首先需要对应用到软件领域模糊定义的“规模”概念所包含的其他元素有公认的定义。当前这些定义争议很大，仍然是今后需要进一步研究的对象。

不过，就方法针对的目的和应用领域来说，COSMIC 规模度量被认为是一种很好的近似。然而，在采用 COSMIC 度量方法的组织的本地化环境中，它可能被期望作为计算功能规模的本地标准。因为这一原因，COSMIC 度量方法提供了本地化扩展。当使用本地化扩展时，度量结果必须按 5.1 节中的特别约定来报告。以下各节展示如何对方法进行本地化扩展。

4.5.2 数据运算为主的软件

COSMIC 方法被设计用于度量“数据移动为主”的软件。像其它正式的功能规模度量（FSM）方法一样，它不是设计用来精确地度量数据运算功能的。取而代之的是，方法假定数据移动类型负责了数据运算的功能（详见下文）。此假设已被证明是合理的，不论是在实际用途上（如项目性能度量和估计，这是 COSMIC 方法的设计初衷），还是在其通常应用的领域上。

然而，经验证明 COSMIC 方法也能成功地用于度量“数据运算为主”的软件，如一些科学/工程类软件。例如，本方法能成功应用在这样的软件中：软件必须处理大量的数据，导致生成大量数据移动类型。后者能有力代表任何可能出现的计算复杂型数据运算。“成功地应用”指针对度量目的，运用本方法能得出有意义和有用的规模。例子包括专家系统的规模度量，能数字化地处理持续变量的软件，能从科学实验或工程过程中收集并分析数据的软件，等等。

然而，考虑到 COSMIC 方法的设计初衷，当方法使用者要度量数据运算为主的软件的功能规模时，应该自行决定：针对度量目的，方法能否真的能得出既有意义又有用的功能规模。当遇到方法不足以用于度量数据运算的情况时，可以采取本地化扩展来克服此局限——参见 4.5 节。

4.5.3 功能规模贡献因子的局限性

在其适用领域内，COSMIC 方法没有试图度量所有可能对软件“规模”有贡献的功能方面。例如，度量方法没有明确地捕获软件“复杂性”的影响。但复杂性的种类有很多，如结构上的、语义上的、时序上的、过程和数据等等，而在度量功能规模的过程中，方法实际上将过程复杂性对规模的贡献采用了一种简化的处理方式（间接地对数据复杂性也做了同样处理）。

方法也没有考虑数据移动中数据属性数量对功能规模的影响。如果有必要，正如 4.5.6 节所描述的，可以通过本地化扩展来支持功能规模的这些方面。

4.5.4 度量非常小的软件的局限性

所有功能规模度量方法均基于一个简化的软件功能模型假设，即对于它预期的应用领域以及应用于项目性能度量和估算时，在“平均水平”上是趋向于合理的。因此应该注意，在度量、比较或使用非常小的软件规模，特别是软件的变更非常小时，“平均水平”假设是不成立的。对于 COSMIC 方法，“非常小”意味着“极少量的数据移动”。

注：以上的注意事项并不会影响 COSMIC 方法在敏捷软件中的应用。正相反，COSMIC 方法可以成功应用于度量敏捷软件开发的用户故事的规模，即使有时只涉及到很少的数据移动。已经有很多报告验证了该做法的有效性，这些报告给出了汇总单个的用户故事的 CFP 规模得到的整个敏捷迭代（或称为“sprint”）规模，并且此规模与迭代工作量相关性很好（该相关性比工作量与故事点个数的相关性更强）。关于此类对比报告的案例请见[16]。

4.5.5 针对复杂算法的本地化扩展

如果认定有必要解决复杂算法的度量问题，可以为这种例外的功能设置一个本地标准。对于任何包含非常复杂的数据运算功能子处理的功能处理，度量者可以自由地给出他（她）自己的按照本地标准定义的功能点。

案例：一个本地化扩展标准可以是：“在我们的组织中，诸如（在本地有意义的、好理解的例子列表）的数学算法计作 1 个本地 FP。（例子的另一个列表）计作 2 个本地 FP，等等。”

4.5.6 针对度量子单位的本地化扩展

当数据移动度量需要更高精度时，可以定义度量的子单位。例如，1 米可以分为 100 厘米或 1000 毫米。类似地，一个数据属性的移动可以被用作度量的子单位。对少量软件样本用 COSMIC 方法进行度量试验，结果显示对于四类数据移动，每个数据移动的平均数据属性数量变化不大。因为这个原因以及为了度量的方便性，COSMIC 的度量单位“1CFP”就固定为在一个数据移动的水平。然而，当需要对两个不同的用 CFP 度量规模的软件进行比较，而两个软件中数据移动的平均数据属性数量差别很大时，显然应该引起警惕。

任何想通过引入度量子单位来改进 COSMIC 方法的想法，都是允许的，但应该指明度量结果不是用标准 COSMIC 功能点来表达的。

第五章 度量报告

5.0 章节概要

当度量已完成并被认可时，必须报告结果，并且为了确保结果总能清晰地被解读，度量过程数据必须存档。本章列出了记录时需要考虑的参数。

5.1 标识

通用软件模型可以描述为一个矩阵形式，行表示功能处理（可以按层分组），列表示数据组，单元格放置识别出来的子处理（输入， 输出， 读和写）。附录 A 给出了通用软件模型的这种表示方式。

COSMIC 度量结果将根据以下约定进行报告和存档。报告 COSMIC 功能规模时，应该按照下列与 ISO/IEC 14143-1:2007 标准一致的约定进行标识：

规则——COSMIC 度量的标识
COSMIC 度量结果表示为“***x*** CFP (v)”，其中： • “**x**”表示功能规模的数值。 • “v”表示用于获得功能规模数值“**x**”的 COSMIC 方法的标准版本的标识。 注：如果使用的是本地近似方法，而不是标准 COSMIC 版本的约定获得度量结果，那么也使用上述的标识，但应该在其他地方说明使用的近似方法——参阅 5.2 节。

*案例：一个用本度量手册的规则得到的结果表示为 “**x** CFP (v4.0)”。*

使用了本地化扩展时（如 4.5 节定义的那样），度量结果必须按照如下定义进行报告。

规则——COSMIC 本地化扩展的标识
使用本地化扩展的 COSMIC 度量结果表示为： “***x*** CFP (v) + z Local FP”，其中： • “**x**”表示使用 v.y 版本的标准 COSMIC 方法得到的汇总所有单个度量结果后得到的数值。 • “v”表示用于获得功能规模数值“**x**”的 COSMIC 方法的标准版本的标识。 • z 表示使用 COSMIC 方法本地化扩展得到的汇总所有单个度量结果后得到的数值。

5.2 COSMIC 度量结果的存档

存档 COSMIC 度量结果时，为确保度量结果总是可以解释的，应保存下述信息：

规则——COSMIC 度量结果报告
除了按照 5.1 节对实际度量结果进行记录外，根据度量目的以及期望的与其他度量的可比

性（如想要与基准比较），还应该记录每次度量的下述部分或所有属性：

a) 被度量软件构件的标识（名字、版本 ID 或配置 ID）。

b) 用于识别度量所使用的 FUR 的信息来源。

c) 软件所属领域。

d) 度量针对的层次结构的描述（如果层次适用的话）。

e) 度量目的的描述。

f) 度量范围的描述，以及与一组相关度量（如果有的话）的总体范围之间的关系（使用 2.2 节中的总体范围分类）。

g) 所用的度量模式（COSMIC 或本地），以及处理的模式（联机或批处理）。

h) 软件的功能用户。

i) 可用软件制品的颗粒度级别和软件的分解级别。

j) 进行度量时项目所处的生命周期时点（特别是，度量是基于不完整需求的估算，还是以实际已交付功能为基础进行的度量）。

k) 度量的目标或者可信的误差范围。

l) 指明是否使用了标准 COSMIC 度量方法，和/或使用了标准方法的本地近似，和/或使用了本地化扩展（见 4.5 节）。使用 5.1 或 5.2 节的约定进行标识。

m) 指明度量的是开发的功能，还是交付的功能（“开发的”功能是通过创建新软件获得的；而“交付的”功能包括开发的功能，以及通过其他途径获得的功能，即：对现有软件的所有形式的复用，软件包的安装使用，使用现有参数增加或变更的功能，等等）。

n) 指明度量的是新提供的功能，还是“增强”活动的结果（即：增加、修改和删除功能的总和——见 4.4 节）。

o) 主要构件（如果适用的话）的数目，这些构件的规模已经增加到总体规模记录中。

p) 复用软件所占的功能百分比。

q) 对于总体度量范围中的每个范围，按照附录 A 建立一个度量矩阵。

r) 度量者的姓名以及 COSMIC 认证资质、度量日期。

参考文献

下列 COSMIC 文件（包括译本）均能在 www.cosmic-sizing.org 上找到。

COSMIC 文件的题目没有标明与其相关的方法版本号。所有文件将定期根据方法的最新版进行更新。

[1] ISO/IEC 19761:2011 Software Engineering – COSMIC: a functional size measurement method, www.iso.org

[2] Introduction to the COSMIC Method of measuring software

[3] (Example of several papers by the same authors) Al-Sarayreh, K.T. and A. Abran, Specification and Measurement of System Configuration Non Functional Requirements, 20th International Workshop on Software Measurement (IWSM 2010), Stuttgart, Germany, 2010

[4] Guideline for Sizing Real-time Software

[5] Guideline for 'Measurement Strategy Patterns

[6] Guideline for approximate COSMIC functional size measurement (under construction)

[7] Guideline for Sizing Business Application Software

[8] Guideline for sizing Data Warehouse Application Software

[9] Guideline for Sizing Service-Oriented Architecture Software

[10] Quick Reference Guide to the COSMIC method for sizing Business Application Software

[11] Quick Reference Guide to the COSMIC method for sizing Real-Time Application Software

[12] Advanced and Related Topics

[13] Guideline for Convertibility (under construction)

[14] International Vocabulary of Basic and General Terms in Metrology, International Organization for Standardization, Switzerland, 2nd edition, 1993, ISBN 92-67-01075-1

[15] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition

[16] Adapted from Merriam Webster's Collegiate Dictionary, 10th Edition, and La Petit Larousse Illustré, 1996 Edition

[17] ISO/IEC 14143/1:2011 Information technology – software measurement – functional size measurement. Part 1 Definition of concepts

[18] Glossary of terms for 'Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating', COSMIC and IFPUG, September 2015.

[19] See for example: www.wikipedia.org/wiki/AUTOSAR .

附录 A—文档化 COSMIC 规模度量

对于总体范围内的每一个识别出的构件，都已映射到通用软件模型，下面的结构可以作为保存度量结果的资料库。总体度量范围内的每个范围都对应一个矩阵。

在 www.cosmic-sizing.org（或 www.cosmicon.com 的门户）上的出版物知识库中有大量用于记录度量结果的电子表格可供下载。

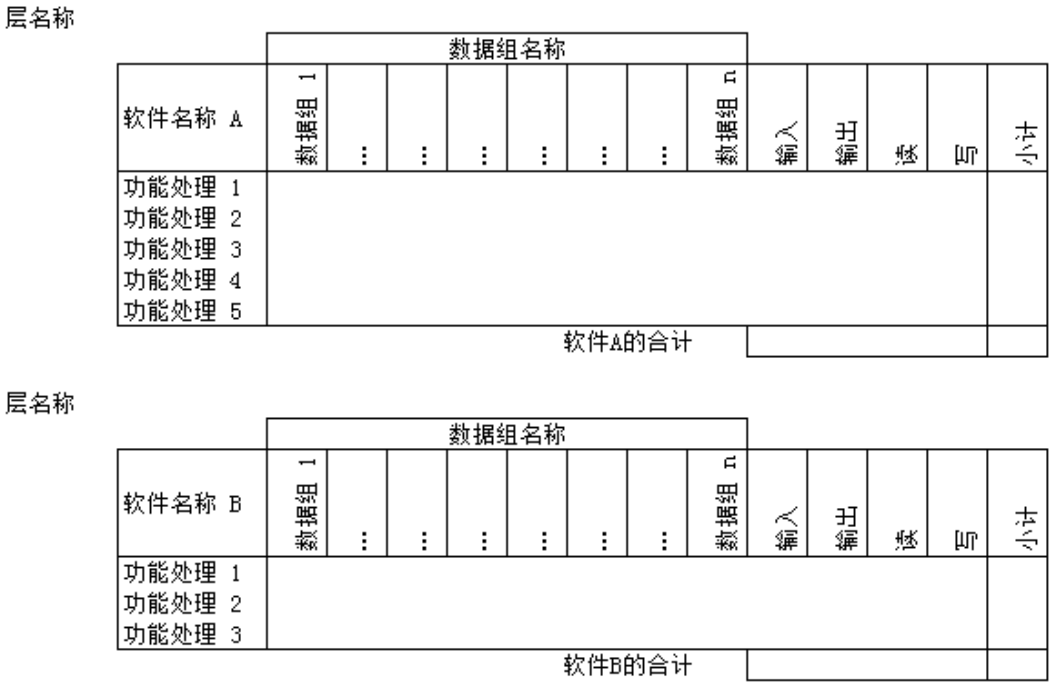


图 A-通用软件模型矩阵

度量策略阶段

- 每层中已定义范围的软件块可以作为一个独立的软件项进行登记。

映射阶段

- 每个识别出的数据组登记在一列中。
- 每个功能处理登记在一个特定行中，按识别的软件项进行分组。
- 对每个识别出的功能处理，按照如下的约定将识别出的数据移动（无论是新增还是修改的）标识在相应的单元中：E 表示输入，X 表示输出，R 表示读，W 表示写。

度量阶段

- 对于每个识别出的功能处理，数据移动按照类型求和，将每个总和登记在矩阵最右边的相应列中。
- 对于每个构件，度量结果可以累加得到，并登记在“合计”行的相应单元格中。

附录 B—非功能需求演变的案例

下表列出了一些需求陈述的实例，这些需求可能最初体现在系统级别（甚至在需求分配到软件或硬件前）或在软件级别呈现为非功能需求，但随着项目的进展，会完全或部分演变为软件的 FUR 与真正的“非功能”需求的混合体。

- 第 1 列：系统或软件 NFR 陈述的实例。
- 第 2 列：随着项目的进展，会从列 1 中的 NFR 演变为软件 FUR 的实例。FUR 可能是待开发软件的 FUR 或者是待采购软件的 FUR，如“COTS”（商品现货软件）。
- 第 3 列：分离出第 2 列中的软件 FUR 后，所留下的系统或项目的需求和约束的实例。因此，这些是真正的“非功能”需求。

最初可能呈现为非功能的系统或软件需求	**由系统最初的 NFR 演变而成的软件（待开发或待采购的软件）FUR 的实例**	**一些系统最初需求演变成软件 FUR 后，留下的真正 NFR 的实例**
高峰时间的系统响应时间平均不应超过 X 秒。	软件可以： • 实时提供系统所需的外部数据 • 监控并报告平均响应时间	• 特定的（快速）硬件 • 一部分用初级语言编写的软件 • 具体的响应时间目标陈述
每年系统的可用性平均应超过 Y%	软件可以在不中断服务的前提下快速切换至备用处理器	• 在“热备份”模式下运作的备用硬件处理器 • 具体的可用性目标陈述
用户可以很容易地维护应用程序的参数	软件可以让用户维护参数表	（无）
未经训练的公众人群也应能使用该系统，而且成功完成率应达 Z%。	软件可以： • 提供全面的辅助工具 • 提供结构优良、便于用户使用的菜单 • 便于具有视力障碍的用户使用	• 对盲文键盘的需求 • 组织公众人群进行广泛测试 • 具体的 Z%目标完成率描述
用户应有通过加密来保护文件的选项	软件可以根据用户需求加密和解密文件	使用硬件“软件狗”或密匙加密设备
系统应能在 X、Y 和 Z 硬件/软件环境之间移植	能够隔离特定接口需求的主要功能与 X、Y 和 Z 环境的软件层	使用具有高度可移植性的语言，例如 Java

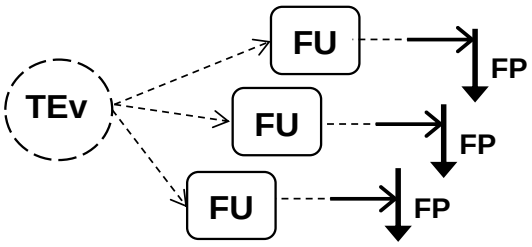
附录 C—触发事件、功能用户以及功能处理的基数

大体上触发事件、功能用户、触发输入以及功能处理之间的关系（如图 3.3 所示）是多对多的，其中只有一个例外。（这个例外即任何一个触发输入可能只会触发一个功能处理——请看 3.2.2 节中的功能处理的规则 b)）。

下表展示了一些可能存在的关系的实例。注意这些案例可能不够全面。表格运用了如下缩写和图例：

TEv	触发事件	FU	功能用户
----→	被一个触发输入（实线）移动的数据组（虚线）	FP	功能处理

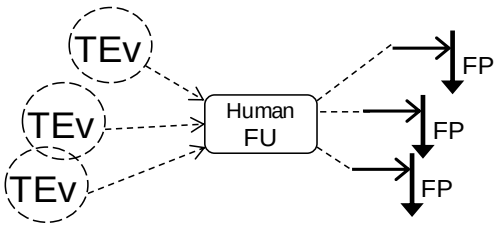
1. 单独一个触发事件可以引起多个功能用户各自在相同或不同的软件系统中触发一个触发输入。每个触发输入启动自己的功能处理。



实时软件案例：地震这一触发事件可以被多个独立的功能用户传感器所感应。每个功能用户触发一个触发输入，这些触发输入在相同或不同的系统中启动功能处理。

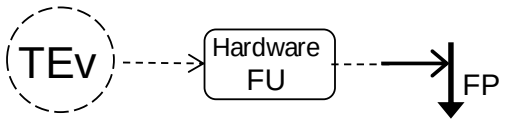
业务软件案例：新员工开始工作这一触发事件使得一个人类功能用户把员工基本信息输入到人事系统中，同时，也使另一个人类功能用户把薪资数据输入到工资管理系统中。

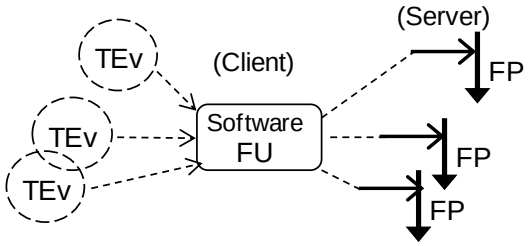
2. 每个触发事件引起一个人类功能用户触发一个不同的触发输入。每个触发输入在相同或不同的软件系统中启动功能处理。



业务软件案例：在警方紧急电话呼叫处理系统中，会收到很多种触发事件，人类功能用户会根据不同的触发事件触发不同的触发输入。而每个触发输入会启动自己的功能处理，来记录该事件。此外，人类用户可能会触发不同的查询触发输入。这些触发输入会在同一呼叫处理系统或其它系统中启动各自的功能处理。

3. 一个硬件或软件功能用户可能被设计用于检测（或“生成”）一个或多个特定类型的事件。每个事件引起功能用户触发一个触发输入。每个触发输入在同一个软件系统中启动功能处理。

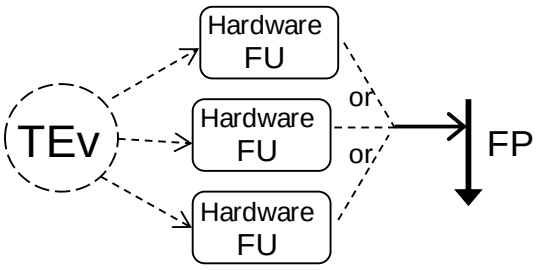




实时软件案例：当液体的温度达到预设水平（触发事件）时，一个热电偶功能用户便触发一个触发输入，使其在一个特定的软件系统中启动功能处理。

业务软件案例：在一个分布式的软件应用程序中，客户端构件是服务器构件的功能用户。客户端构件对信息的不同需求（即触发事件）使其触发不同的触发输入，根据所需的不同类型的服务，每个触发输入会启动服务器构件的不同功能处理。

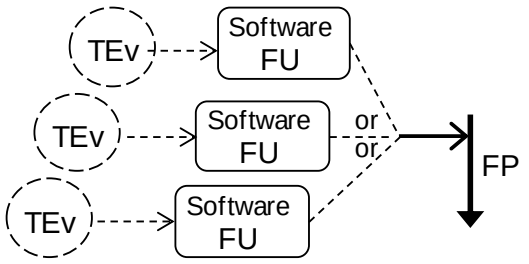
4. 同一软件中两个或多个的硬件功能用户可能会感应到同一触发事件。每个功能用户会触发启动同一功能处理的触发输入。



实时软件案例：在实时过程控制系统中，异常情况（触发事件）可能会被一个或多个硬件功能用户所感应。每个功能用户可能都会启动同一紧急关闭功能处理。

（注意：此功能处理的每一次运行将会被第一个感应到触发事件的功能用户所触发。）

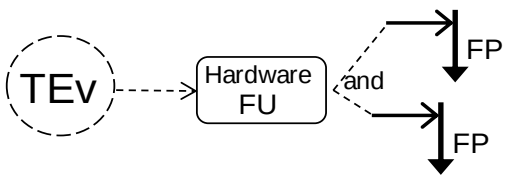
5. 两个或多个软件功能用户会各自触发一个触发输入，它们都启动同一功能处理。



基础设施软件案例：几个软件功能用户会“调用”（即触发）相同复用软件构件中的同一功能处理。（在此案例中，当软件功能用户调用构件时，它便“生成”此事件。）

（注意：此功能处理的每一次运行只能同时被一个可能的软件功能用户所触发。）

6. 在检测到一个触发事件时，功能用户可能会触发两个或更多触发输入。每个触发输入启动各自的功能处理。



实时软件案例：在一个安全攸关的控制系统中，一个触发事件会引起功能用户（通常为硬件）触发两个触发输入，每个触发输入启动各自的功能处理。例如，这两个功能处理可能有相同的功能性用户需求，但出于差异化策略的原因由不同的组开发而成。

附录 D—COSMIC 方法的原则和规则一览表

为了便于更准确的引用，下表给出了 COSMIC 度量方法 v4.0.2 中的所有原则和规则（左边列出了相对应的章节序号）：

章节	原则和规则的描述
1.3.1	**软件环境模型** **原则** a) 软件被硬件所界定。 b) 软件通常结构化为多**层**。（2.2.2） c) 一层可包含一个或多个单独的**“对等”**软件块。（2.2.2） d) 任何待度量的软件应由其度量**范围**定义，并完全限定在一个单一的层中。（2.2） e) 待度量软件块的范围依赖于度量**目的。**（2.1） f) 可以从待度量软件的 FUR 中识别该软件的**功能用户**，这些功能用户分别作为数据的发送者和/或接受者。（2.3） g) 软件的功能性需求可以在不同的**颗粒度级别**上表达。（2.4） h) 精确的 COSMIC 软件规模度量需要该软件块的 FUR 达到能够识别出**功能处理**和子处理的颗粒度级别。（2.4.3） i) 如果一个软件块的功能性需求只有在较高的颗粒度级别上的描述，则可以采用近似的 COSMIC 方法度量软件块的规模，并将其缩放至功能处理及子处理的颗粒度级别。（2.4.3）
1.3.2	**通用软件模型** **原则** a) 软件块跨越**边界**与功能用户交互、并与边界内的**持久存储介质**进行交互。（2.3） b) 被度量软件块的 FUR 能够被映射到唯一的一组功能处理。（3.2） c) 每个功能处理由一系列子处理组成。（3.2） d) 一个子处理可以是一个**数据移动**或者是一个**数据运算。**（3.2） e) 一个数据移动仅移动单个**数据组。**（3.3） f) 有四类数据移动：**输入，输出，写和读**。（3.5） • 输入从功能用户移动一个数据组到功能处理内。 • 输出从功能处理中移出一个数据组到功能用户。 • 写从功能处理移动一个数据组到持久存储介质。 • 读从持久存储介质移动一个数据组到功能处理。 g) 一个数据组由唯一的一组**数据属性**构成，描述了一个单一的**兴趣对象**。（3.3） h) 功能处理被输入数据移动所触发。功能用户为响应触发事件而产生了触发输入，触发输入移动的数据组由一个响应**触发事件**的功能用户生成。（3.2） i) 一个功能处理的规模等于其数据移动的总数。 j) 一个功能处理包括至少一个触发输入数据移动，以及一个写或输出数据移动，即一个功能处理应该包含至少两个数据移动。一个功能处理中数据移动的数量没有

章节	原则和规则的描述
	上限。（3.3） k) 作为对度量目的的一种近似处理，数据运算子处理不单独度量。任何数据运算的功能被假定已经计算在相关的数据移动内了。（3.5.6） 注：COSMIC 通用软件模型，正如其名字所示，是一个逻辑“模型”。该模型将软件处理数据的各个单元模型化，以便进行功能规模度量。模型并非按照物理顺序描述软件执行步骤或软件的技术实现过程。
1.4	**COSMIC 度量原则** **原则** a) 功能处理的规模等于其数据移动的个数。 b) 给定范围的软件块的功能规模等于其功能处理的规模总和。
2.2	**度量范围** **规则** a) 任何待度量软件块的范围必须从度量目的中导出。 b) 任何一次度量的范围不能延伸超过被度量软件所在的层。
2.2.2	**层** **原则** a) 一层中的软件会根据已定义的准则提供一组内聚的服务，而且其他层里的软件无需了解这些服务是如何实现的也能利用它们。 b) 任意两层中软件之间的关系可用“通信规则”来定义，有两种情况： • “分层的”，如 A 层的软件可以使用 B 层软件提供的服务，但反之则不成立（这里层次结构的关系有可能是自上而下或自下而上），或： • “双向的”，如 A 层的软件可以使用 B 层的软件，反之亦然。 c) 一层的软件通过相应的功能处理与另一层的软件交换数据组。 d) 一层的软件没有必要使用其他层的软件提供的所有功能服务。 e) 在一个已定义的软件体系结构中属于同一层的软件，根据另外一个不同的体系架构可能被划分为其他的层。
2.3	**功能用户** **规则** a) 待度量软件块的功能用户应取决于度量目的。 b) 当软件块的度量目的与开发或者修改软件块的工作量相关时，根据其 FUR 的要求，功能用户应该是与所有新的或者修改的功能进行交互的数据发送者和/或数据预期接受者。 注：FUR 中可能指定功能用户存在多个实例需要单独识别。然而，如果每个实例都属于相同的 FUR，它们就是同一类功能用户。
2.4.3	**功能处理颗粒度级别** **规则** a) 软件块功能规模度量要求其 FUR 的颗粒度达到能够识别功能处理及数据移动子处理的级别。

章节	原则和规则的描述
	b) 如果必须要对一些还没有达到足够详细程度的需求进行度量，可采用近似方法度量需求。这些方法定义了如何在高颗粒度级别度量需求。在高颗粒度级别的度量中运用缩放系数得出功能处理及其数据移动子处理颗粒度级别的近似规模。见“早期或快速 COSMIC 功能规模近似度量指南” [6]。
3.2.2	**功能处理** **规则** a) 一个功能处理应该完全属于某层且仅属于某一层的一个软件块的度量范围。 b) 一个功能处理至少包含两个数据移动，即一个触发输入加上一个输出或写，最小规模为 2CFP。一个功能处理中数据移动的数量没有上限，因此其规模也没有上限。 c) 一个执行中的功能处理，当其响应了触发输入并满足 FUR 时，则功能处理结束。由技术原因导致处理出现暂停时，不能认为功能处理结束了。
3.3.1	**数据组** **原则** 通过它的唯一的数据属性集合，每个被识别的数据组必须是唯一的和可区别的。
3.3.2	**识别一个功能处理中不同的数据组（即不同的兴趣对象）** **规则** 对于作为功能处理输入的所有数据属性： d) 具有不同实例频率的数据属性集描述的是不同的兴趣对象； e) 具有相同实例频率但不同关键属性的数据属性集，描述的是不同的兴趣对象。 f) 按照规则 a)和 b)判断后归为一个集合的数据属性属于同一数据组类型，除非在 FUR 中明确说明了输入此功能处理的、描述同一个兴趣对象的是多个数据组类型。（见注 3） 这些规则也同样适用于功能处理输出的数据属性，或从功能处理移动至持久存储介质的数据属性，或从持久存储介质移动至功能处理中的数据属性。 注 1：上述规则有助于分析复杂的输出，例如：某报告描述了多个兴趣对象，可以把每个数据组当作是由单独的功能处理输出的。当度量复杂的报告时，按照此方法识别的每一个数据组类型也需要分别计数。例如，“业务应用软件度量指南”[7] 2.6.1 节的例子以及 2.6.2 节的分析。也可参考 4.2.4 节中例 4、5 的分析。 注 2：检查输入/输出中的数据组实际是如何组成的或拆分的，也可以帮助识别出不同的数据组类型，但不要依赖于此方法。比如，某一输入或输出有两个或多个数据属性集，为了美观或易于理解把它们从物理角度进行拆分，在这种情况下，如果满足上述规则，则应属于同一数据组。 注 3：见度量手册 3.5 节关于*数据移动*的定义、原则和规则。关于这些规则的例外情况，见 3.5.7 节（例 2、3、4、5）以及 3.5.11 节，如上述规则 c 所述。

章节	原则和规则的描述
3.5.2	**输入 (E)** **原则** a) 输入(E)应移动描述单个兴趣对象的一个数据组，从功能用户一侧跨越边界移动到功能处理内，输入(E)是该功能处理的一个组成部分。如果功能处理的输入包含多个数据组，每个数据组描述一个不同的兴趣对象，则为输入的每个数据组识别为一个输入(E)。（见 3.5.7 节“数据移动的唯一性”） b) 输入(E)不跨越边界输出数据，也不从持久存储介质读取数据或向其写数据。 **规则** a) 触发输入的数据组可能仅由一个数据属性组成，这个触发输入只是为了通知软件“事件 Y 发生了”。大多数情况下，尤其是业务应用软件，触发输入的数据组有若干个数据属性，通知软件“事件 Y 发生了，这是与该特定事件相关的数据”。 b) 时钟节拍作为触发事件，总是位于待度量软件之外。因此，举例来说，每 3 秒发生一次的时钟事件应该与输入仅有一个数据属性的数据组相关。注意，此周期性触发事件无论是由硬件产生的，还是被度量软件边界外的软件产生的，这两者没有区别。 c) 除非需要一个特定的功能处理，否则从系统时钟获得日期和/或时间不可以被认为是一个输入或任何其他数据移动。 d) 如果一个特定事件的发生引发了一个输入，该输入的数据组包含了特定兴趣对象的“n”个数据属性，而 FUR 允许相同事件的另一次发生引发另一个输入，而该输入的数据组只包含兴趣对象“n”个属性的一个子集。那么，此处只能识别为一个输入，包含了所有“n”个数据属性。 e) 人类功能用户通过屏幕向功能处理输入数据，在识别这样的屏幕的输入时，只分析填充了数据的屏幕。忽略 任何格式化的“空白”屏幕（缺省值除外），以及帮助人类用户理解所需输入数据的所有字段和其他标题。 注意：当度量 FUR 中输入的规模变更时，有可能需要考虑此类字段及其他标题的变化——请看 4.4.1 节。
3.5.3	**输出 (X)** **原则** a) 输出(X)将描述单个兴趣对象的一个数据组从功能处理一侧跨越边界移动到一个功能用户处。如果功能处理的输出不止包含一个数据组，那么，输出的每一个数据组都识别为一个输出(X)（参见 3.5.7 节“数据移动唯一性”） 。 b) 输出(X)不应跨越边界输入数据，也不应从持久存储介质读取数据或写入数据。 **规则** a) 对输出固定文本的查询应识别一个输出(X)（“固定”指消息不含可变数据值，比如在一个购物网站上点击“条款声明”按钮所得到的结果）。 注意：关于“帮助”功能的输出，参看“业务应用软件规模度量指南”。关于错误状态或成功确认的相关信息的输出，参看本度量手册的 3.5.11。 b) 如果功能处理的一个输出移动的数据组包含一个特定兴趣对象的“n”个数据属性，而 FUR 允许功能处理有这样一个输出，该输出移动的数据组仅含有兴趣对象“n”个属性的一个子集。那么，只能识别为一个输出，该输出移动一个数据

章节	原则和规则的描述
	组包含所有的“n”个数据属性。 c) 在识别输出时，忽略所有帮助人类用户理解输出数据的字段和其他标题。 注意：当度量 FUR 中输出的规模变更时，有可能需要考虑此类字段及其他标题。——请看 4.4.1 节。
3.5.4	**读 (R)** **原则** a) 读(R)将描述单个兴趣对象的一个数据组从持久存储介质移动到功能处理，读(R)是该功能处理的一部分。如果功能处理必须从持久存储介质中检索一个以上的数据组，那么，为每个检索的数据组识别一个读(R)。（参见 3.5.7“数据移动的唯一性”。） b) 读(R)不应跨越边界接收或输出数据，也不向持久存储介质中写数据。 c) 在功能处理执行中，计算或移动常量、功能处理内部的并且只能由程序员更改的变量、计算过程的中间结果、由执行功能处理而产生并存储的而非来自于用户功能需求的数据，这些都不应该被看作读数据移动。 d) 读(R)数据移动总是包含某些“读请求”的功能（所以任何“读请求”都不会被计算为单独的数据移动），见 3.5.9 节。 **规则** e) 根据 FUR，当被度量软件必须从持久存储介质中检索一个数据组时，识别为一个读。 f) 当被度量软件的 FUR 指定任何软件或硬件功能用户作为数据组的来源或者作为对已存储的数据组的检索手段时，不应识别为读。（关于此案例，请看输入和输出的原则和规则。）
3.5.5	**写 (W)** **原则** a) 一个写(W)将描述单个兴趣对象的一个数据组从功能处理移动到持久存储介质，写(W)构成该功能处理的一部分。如果功能处理必须移动不止一个数据组到持久存储介质，移动到持久存储介质的每个数据组都单独识别为一个写(W)。（参见 3.5.7“数据移动的唯一性”。） b) 一个写(W)不跨越边界接收或输出数据，不从持久存储介质中读数据。 c) 从持久存储介质删除一个数据组的需求应被度量为一个写(W)数据移动。 d) 以下情形不认为是写(W)数据移动： • 在功能处理开始时不存在并且功能处理完成后也没有持久化的数据的移动或运算。 • 功能处理内部变量的生成、更新或中间结果。 • 功能处理的数据存储是实现导致的而不是 FUR 所要求的（如批处理作业中一个大的排序处理时进行了数据的缓存）。 **规则** a) 根据 FUR，当被度量软件移动一个数据组到持久存储介质中时，须识别为一个写。 b) 当被度量软件的 FUR 指定任何软件或硬件功能用户为数据组移动的目的地或作为存储数据组的手段时，不应识别为写。（关于此案例，请看输入和输出的原则

章节	原则和规则的描述
	和规则。）
3.5.6	**与数据移动关联的数据运算** **原则** 功能处理中所有数据运算都与四类数据移动（E，X，R 和 W）关联。按照惯例，功能处理的数据移动假设也负责了功能处理的数据运算。 **规则** a) 输入数据移动负责使得数据组能被功能用户输入数据和确认数据的所有数据运算（如格式化和显示）。 b) 输出数据移动负责创建数据组属性的所有数据运算，此数据组用于输出和/或使得数据组能够被输出（如格式化和显示），并能被发送到预期的功能用户。 c) 读数据移动负责为从持久存储介质上检索数据组所需的所有计算和/或逻辑处理。 d) 写数据移动负责为创建或更新待写数据组或删除数据组所需的所有计算和/或逻辑处理。 与以上任何数据移动相关联的数据运算不包括成功完成数据移动后所需的数据运算，也不包括任何与其他数据移动相关联的数据运算。
3.5.7	**数据移动唯一性和不常见的情况** **规则** 注意：所有 COSMIC 规则关注的都是功能用户、数据组、数据移动、功能处理和兴趣对象的各种*类型*。为了便于阅读，我们一般省略这些短语中的“类型”。但是在下面的规则 d)中，我们保留了“类型”这个词，以便区分“类型”和“实例”。 a) 除非功能性用户需求定义了规则 b)或 c)的情况，否则描述要输入到一个功能处理的任意一个兴趣对象的所有数据，应被识别为一个输入所移动的数据组。 注：一个功能处理当然可以有多个输入，每一个移动的数据描述了不同的兴趣对象。 相同的等价规则适用于任意一个功能处理的读、写或输出数据移动。 b) 如果功能用户需求明确定义了需要输入到一个功能处理中的、来自于必须被该功能处理单独识别的功能用户的、刻画了同一个兴趣对象的不同数据组，针对每个不同的数据组应该识别一个输入。 相同的等价规则适用于从一个功能处理输出数据到不同的功能用户。 注意：任何一个功能处理应该只有一个触发输入。 c) 如果功能用户需求明确定义了从持久存储介质移动不同的数据组到一个功能处理中，每个数据组描述的是同一个兴趣对象，此时应该为每个不同的数据组识别不同的读。 相同的等价规则适用于任意给定功能处理的写。 注：这条规则类似于规则 b），如果在 FUR 中指定对同一兴趣对象的读取区分为不同数据组，它们很可能最初来自不同的功能用户。如果在 FUR 中指定写区分为不同的数据组时，它们可能是被不同的功能用户存储进去的。 d) 任何在执行过程中重复出现的数据移动类型，不应被计数。 这一条也适用于如下情况：数据移动类型发生多次不同的执行，是因为所移动数据组的数据属性取值不同而导致在该功能处理类型中采取了不同的处理路径。

章节	原则和规则的描述
3.5.9	**功能处理从功能用户处获取数据** **规则** a) 当功能处理不需要告知功能用户发送什么数据时，功能处理会通过来自功能用户的输入数据移动获取一个数据组，如下面四个情形之中的任意一种： • 功能用户通过触发输入发送一个数据组启动功能处理； • 接收到触发输入发送的数据组的功能处理处在等待状态，等待来自功能用户输入的下一个数据组 (这种情况在业务应用软件中会发生，由人类用户输入数据给软件)； • 当功能处理已经启动，向功能用户请求“如果你有数据，现在就发给我”，功能用户即发送数据； • 当功能处理已经启动，随即检查功能用户的状态并检索其需要的数据。 对于后面两种情形（通常发生在实时“轮询”的软件中），按照惯例，获取所需的数据请求不应识别为来自于功能处理的输出。功能处理仅仅需要发送一个提示信息让功能用户输入数据，提示信息这一功能被认为是输入的一部分。功能处理知道等待什么数据。在这案例中只需要识别一个输入。 b) 当功能处理需要获取功能用户的服务（如获取数据），并且功能用户需要被告知应该发送什么时（典型的情况是，功能用户是被度量软件范围之外的另一个软件），将识别为一个输出跟着一个输入。输出发送对特定数据的请求，输入接收返回数据。
3.5.10	**带有人机界面的应用领域的控制命令** **规则** 带有人机界面的应用程序中“控制命令”应该被忽略，因为它们不包含任何关于兴趣对象的数据移动。
3.5.11	**错误/确认消息以及其他错误状态的提示** **规则** a) 被度量软件的任何一个功能处理发布的所有各类错误/确认信息，根据其 FUR，不论什么原因都应被识别并归为一个输出，如关于以下事情的成功或失败：对输入数据的确认、检索数据，或使数据持久化的要求，或对其它软件块请求的服务响应。 注意：如果功能处理的 FUR 没有要求发出任何形式的错误/确认消息，无须识别相应的输出。 b) 如果一条发送给人类功能用户的消息，除了确认输入数据已被接受或输入数据有误，还提供了其他的数据，那么除了这个错误/确认输出外，这些额外的数据应被识别为一个由输出移动的数据组。 c) 根据标准的 COSMIC 规则，所有被度量的软件发送到或接收来自其硬件或软件功能用户的其它数据，不管数据值是否指示一个错误状态，都应根据 FUR 分别作为输出或输入来分析。 d) 读和写被看作负责报告与之关联的错误状态。因此，被度量的功能处理中，由于

章节	原则和规则的描述
	对持久数据读或写而接收到的任何错误提示不被识别为输入。 e) 在软件使用中，可能会发出错误状态信息，但该软件的 FUR 并没有要求对该错误状态进行任何处理，如操作系统发布的错误消息。对于所有指示这类错误状态的消息，都不应识别输入或输出。
4.3.1	**汇总度量结果** **规则** a) 对于任何功能处理，各个数据移动的功能规模应该通过累加的方式汇总为以 CFP 为单位的一个功能规模值。 规模 (功能处理 $_i$) = Σ 规模（输入 $_i$）+ Σ 规模（输出 $_i$） + Σ 规模（读 $_i$）+ Σ 规模（写 $_i$） b) 对于任何功能处理，其功能性用户需求中功能规模的变更规模如果以 CFP 为单位衡量的话，应该是功能处理中增加、修改、删除的数据移动的规模的汇总，采用以下公式计算。 规模 (变更(功能处理 $_i$)) = Σ 规模 (增加的数据移动 $_i$) + Σ 规模 (修改的数据移动 $_i$) + Σ 规模 (删除的数据移动 $_i$) 关于汇总功能规模的更多内容，参阅 4.3.2 节。关于对变更软件的规模度量，参阅 4.4.2 节。 c) 一个指定范围的软件块的规模是通过汇总该软件块功能处理的规模得到的，遵循下面的规则 e)和 f)。 d) 一个指定范围的软件块任意变更的规模是通过汇总该软件块所有功能处理的变更规模得到的，遵循下面的规则 e)和 f)。 e) 只有在 FUR 的同一功能处理颗粒度级别进行度量时，软件块的规模或者软件变更的规模才可以累加。 f) 只有对度量目的有意义时，同一层或不同层的软件规模或软件变更的规模才可以进行累加。 g) 一个软件块的规模是通过累加其构件（不管它是如何被分解的）的规模来获得的，并且构件间的数据移动需要被剔除。 h) 某个功能处理中发送给人类功能用户的所有错误/确认消息仅识别为一个输出。 i) 如果对 COSMIC 方法进行本地化扩展（例如要度量标准方法没有覆盖的某方面规模），那么经本地化扩展得到的度量规模必须如 5.1 节中描述的那样单独报告，不可以累加到通过标准方法获得的以 CFP 为单位的规模中去（详细信息参见 4.5 节）。

章节	原则和规则的描述
4.4.1	**修改一个数据移动** **规则** a) 如果一个数据移动必须被变更（因为与数据移动关联的数据运算有变化，以及/或因为移动的数据组的属性数量或类型有变化），需要计算为一个变更的 CFP，而不管一个数据移动中的实际修改数值。 b) 如果一个数据组必须被修改，此修改并不影响该变更数据组的数据移动功能，则该数据移动就不应该识别为变更的数据移动。 注意：对输入或输出屏上任何数据的改变，如果不涉及功能用户的兴趣对象，则不能被识别为一个变更的 CFP（这类数据的例子见 3.3.4 节）。
4.4.2	**规则——功能发生变更后软件的规模** 软件功能改变后： 新的总规模（变更后的软件块）=原规模+Σ 规模（新增的数据移动）- Σ 规模（删除的数据移动）
5.1	**COSMIC 度量的标识** **规则** COSMIC 度量结果表示为“***x*** CFP (v)”，其中： • “**x**”表示功能规模的数值， • “v”表示用于获得功能规模数值“**x**”的 COSMIC 方法的标准版本的标识。 注：如果使用的是本地近似方法，而不是标准 COSMIC 版本的约定获得度量结果，那么也使用上述的标识，但应该在其他地方说明使用的近似方法——参阅 5.2 节。 **本地化扩展的标识** **规则** 使用本地化扩展的 COSMIC 度量结果表示为： “***x*** CFP (v) + z Local FP”，其中： • “**x**”表示使用 v.y 版本的标准 COSMIC 方法得到的汇总所有单个度量结果后得到的数值， • “v”表示用于获得功能规模数值“**x**”的 COSMIC 方法的标准版本的标识。 z 表示使用 COSMIC 方法本地化扩展得到的汇总所有单个度量结果后得到的数值。
5.2	**COSMIC 度量结果报告** **规则** 除了按照 5.1 节对实际度量结果进行记录外，根据度量目的以及期望的与其他度量的可比性（如想要与基准比较），还应该记录每次度量的下述部分或所有属性： a) 被度量软件构件的标识（名字、版本 ID 或配置 ID）。 b) 用于识别度量所使用的 FUR 的信息来源。 c) 软件所属领域。 d) 度量针对的层次结构的描述（如果层次适用的话）。 e) 度量目的的描述。

章节	原则和规则的描述
	f) 度量范围的描述，以及与一组相关度量（如果有的话）的总体范围之间的关系。（使用 2.2 节中的总体范围分类）。 g) 所用的度量模式（COSMIC 或本地），以及处理的模式（联机或批处理）。 h) 软件的功能用户。 i) 可用软件制品的颗粒度级别和软件的分解级别。 j) 进行度量时项目所处的生命周期时点（特别是，度量是基于不完整需求的估算，还是以实际已交付功能为基础进行的度量）。 k) 度量的目标或者可信的误差范围。 l) 指明是否使用了标准 COSMIC 度量方法，和/或使用了标准方法的本地近似，和/或使用了本地化扩展（见 4.5 节）。使用 5.1 或 5.2 节的约定进行标识。 m) 指明度量的是开发的功能，还是交付的功能（“开发的”功能是通过创建新软件获得的；而“交付的”功能包括开发的功能，以及通过其他途径获得的功能，即：对现有软件的所有形式的复用，包的实现，使用现有参数增加或变更的功能，等等）。 n) 指明度量的是新提供的功能，还是“增强”活动的结果（即：增加、修改和删除功能的总和——见 4.4 节）。 o) 主要构件（如果适用的话）的数目，这些构件的规模已经增加到总体规模记录中。 p) 复用软件所占的功能百分比。 q) 对于总体度量范围中的每个范围，按照附录 A 建立一个度量矩阵。 度量者的姓名以及 COSMIC 认证资质、度量日期。

附录 E—从 4.0 版到 4.0.1 版和 4.0.2 版的主要变更

本附录包括了 COSMIC 功能规模度量方法从 4.0 版到 4.0.1 版和当前的 4.0.2 版的主要变更。想追溯本方法更早以前的变更，请查阅度量手册的各个早期版本（2.0、3.0、3.0.1 和 4.0）。“MUB”是方法更新公报，它发表于度量手册主要版本之间，用以宣布并解释改进提议。

E1 从 4.0 版到 4.0.1 版的主要变更

V4.0.1 参考	变更
前言	如果是新手或正在转向 COSMIC 方法，应该首先阅读简介文档。通过对已有的段落增加了一个小标题对此进行了强调。
1.2.3	“非功能需求”的定义更清晰，并排除了项目需求和约束。图 1.3 也做了相应修改。
1.3.3	增加新章节 1.3.3“类型和实例”。COSMIC 从未明确定义过“类型”和“实例”，这使得度量者有时难以区分这两个概念。
3.2.1	“功能处理”的定义有一处比较含糊，即“独一无二”是修饰“一组数据移动”的还是“功能性用户需求基本部件”的。解决方式是：在“移动”后面加了一个逗号，并将出现两次的代词“that”改为“these”。
3.3.1	“数据组”的定义中，不必要的、比较繁琐的地方被简化了。原则中，有两条因为没有价值而被移除。
3.3.1	“兴趣对象”的定义被修订了，并增加了一个注释，以使定义更清晰。
3.5.1	“数据移动”，“输入”，“输出”，“读”，“写”的定义改为每个都被认为“负责”（而不是“包含”）任何与之相关的数据运算。
3.5.6	新增加的一段文本强调了这一节的规则仅适用于对已有 FUR 的变更的度量。 定义了哪种类型的数据运算与哪种数据移动相关联的规则更清晰了，现在用“关联于”替换了“包括于”数据移动。
3.5.7	对“数据移动的唯一性和可能的例外”规则的语言进行了简化。更正了一个错误（两个数据移动，如果有 FUR 定义了它们关联了不同的数据运算，也不能被识别为不同的数据移动，因为这将与“功能处理中所有数据运算都与四类数据移动（E，X，R 和 W）关联”的原则相矛盾）。增加了两个案例，案例被重新排序以便与修订后的规则一致。另外，其他地方做了一些编辑简化。
3.5.11	“错误/确认消息”的定义被澄清了。部分定义变成一条规则。
4.3.1	规则 g）是关于如何从一个软件块的所有构件规模得到软件块整体规模，原来的“双重否定”被修改为肯定的陈述方式。并且扩充了如何累加错误/确认消息的规则。
综合变化	参考 ISO/IEC 标准 14143/1 的（FSM 方法）概念定义和 ISO/IEC 19761 对 COSMIC 方法主要概念的定义有些混淆，进行了纠正。
词汇表	之前的“输入数据”和“输出数据”定义过于复杂，进行了简化。

E2: 从 4.0.1 版到 4.0.2 版的主要变更

注意：变更的类型定义为如下三方面：

- “方法”，是指 COSMIC 方法的定义或规则的新增或修改，以便更易于理解。（修改或新增都标注为“文字调整”）
- “文字调整”，为了易于理解而调整了文字描述。除了如下表列举情况，也对很多文字描述进行了细微调整。
- “纠正”，v4.0.1 版本中的错误在本版中得到修正。

V4.0.2参考	变更类型	变更内容
-	文字调整	“近似COSMIC功能规模度量指南”改名为“早期或快速COSMIC功能规模近似度量指南”
前言		增加了4.0.2版本主要变更的概括
1.2	文字调整	在4.0版本的COSMIC方法中，我们对术语“功能性用户需求”（FUR）的使用进行了约束。此约束现已明确公示在该术语定义的注释2中。
1.2, 1.3	纠正	在第1章的多个地方，尤其是在软件环境模型的两个原则中，术语“FUR”由“功能性需求”代替，为了与FUR的使用约束相一致。在该约束引入4.0版本时，本应该对这些术语的使用进行更正。
1.3.2	方法	新增一条原则“一个功能处理的规模等于其数据移动的总数”，该原则原先只在4.3.1节作为规则提到过。关于功能处理的规模范围的原则现在明确地表明，其规模没有上限。
文字调整		在通用软件模型中增加注释，该模型并不是用来描述功能处理执行步骤的物理顺序的。
1.3.3	文字调整	该章节进行了结构调整，并修改了对术语“类型”的解释，以帮助理解，特别是在功能用户为相同类型时，可以便于识别。同时完善了关于实时案例2的解释。
2.0	文字调整	本章的概述更加简洁并与章节内容更贴合。
2.2.1	文字调整	图2.1所示的业务软件例子的描述进行了调整。
2.2.2	文字调整	对AUTOSAR架构的参考WWW地址过于陈旧，被删除。
2.2.3	文字调整	“分解层级”的注释2中，“可能只是”改为“只是”， “可能”在该场景下不正确。 增加了分解层级的注释3，可能与软件架构的层相关，但不是必须相关的。
2.3	文字调整	修改了该章节的标题，因为并不需要识别“持久存储介质”。2.3节的开篇以及2.3.1节都重新按照逻辑顺序做了调整。
2.3.1	方法	“功能用户”的定义更加明确（MUB 12）。
2.3.1	文字调整	为了便于理解，对“功能用户”的规则增加了注释。 在4.0.1版本中的例1实际并不算做一个例子。该内容重新作为文本编写，后面所有的例子也都重新编号。增加了对“功能用户”的进一步解释。 为了与时俱进更新了实时例子2和4，在例2中用复印机代替了现在已经不常见到的有键盘的手机；在例子4中，输出是在仪表盘的显示屏上显示，而不是在4个LED屏上显示。
2.3.2	文字调整	增加了对持久存储介质的进一步说明，以及它们与文件或数据库的关系。
2.4.1	方法	扩展了“颗粒度级别”的定义，明确其可应用于软件块的“任意部分”，并非必须用于整体描述。

V4.0.2参考	变更类型	变更内容
2.4.3	纠正	“功能处理颗粒度级别”的定义增强了易读性，包括纠正了“FUR”术语的使用。“功能处理颗粒度级别”的规则重新命名为“度量功能处理时的颗粒度级别”以表明度量功能处理需要FUR处于两个颗粒度级别，即功能处理颗粒度级别及数据移动颗粒度级别，规则b)也对从更高颗粒度级别的近似度量进行了描述。
2.4.3	文字调整	明确指出，整个“Everest订货应用”案例都是处于FUR颗粒度级别（而不是软件分解层级）。该案例的目的也进行了明确。
3.2.1	方法	修改了“触发事件”的定义，明确只有任意一个功能用户产生的第一个数据组，才会被触发输入所移动（MUB 14）。
3.2.1	方法	在功能处理定义的第二点中，用“应该（shall）”代替了“可能（may）”。从学术上说，是可以用may，但正如前言所述，在ISO术语中“may”是指“被允许”的意思。而在该定义中，使用“必须（shall）”可以更加清楚地表明这是强制的定义。 在定义中增加了注 4，用来解释在定义中的 a)条款里“唯一性”的用法。
3.2.1	文字调整	在“触发输入”的定义中，增加了注释以解释“功能处理开始执行所必需的”是逻辑GSM的结果，并不表示当输入数据时，处理才真正地物理上执行。在术语表中“触发输入”的定义中，删除了注释。这个注释实际上属于功能处理定义的注释。
3.2.1	文字调整	在图3.3中，数据组箭头用平行四边形替代，为了表示与该图中其他符号同样重要。
3.2.2	文字调整	删除了规则b)关于功能处理的描述，多余。 重新编写后的规则b)的内容更易于理解。 重新编写后的规则c)的内容微调，为了与功能处理的定义保持一致。
3.2.3	文字调整	扩展了度量批处理功能处理的描述，以表明输入数据可能为瞬态数据流（或是临时存储的数据集）。图3.4也进行了修改。
3.3.1	方法	在兴趣对象的定义中，“处理和/或移动数据”用“向/从软件中，或向/从持久存储介质移动数据组”代替。该调整为了防止可能误认为兴趣对象仅仅通过“处理”来识别的，如数据运算。 同时，也因此删除了关于“处理”的注释1。 增加了注释2，当功能用户发送关于其本身的数据时，功能用户也是所发送的数据组的兴趣对象。 增加了注释3，一个“事物”是否是“兴趣对象”没有绝对的标准（MUB 12）
3.3.1	文字调整	对“数据组”的定义增加了注释，阐明数据组并不需要包含兴趣对象的所有数据属性（MUB 13）。
3.3.1	文字调整	把3.4.1中的“数据属性”的定义移动至3.3.1节，逻辑更合理。
3.3.2	方法	增加了新规则：“识别同一个功能处理中不同的数据组（即不同的兴趣对象）”该规则尤其有助于在业务应用软件中识别功能处理的复杂输出中包含的数据组。并且，该规则适用于所有业务领域的软件，也同样适用于从/向持久存储介质移动的数据组。
3.3.2	文字调整	本节内容几乎全部重新调整，增加了例子便于文字理解。
3.3.4	纠正	文字进行调整以明确在业务应用领域及实时领域，功能用户可以是兴趣对象。增加了业务软件举例（与MUB 12保持一致）。

V4.0.2参考	变更类型	变更内容
3.4.1	文字调整	该章节重新命名为“数据属性举例”，并改进了例子。
3.5.2	方法	规则 c)扩展了输入的描述，明确无论是从操作系统获取当前日期还是时间都不识别为输入。 规则 d)关于输入。“触发”一词用“引发”替换，为了避免对该规则理解有误，误认为该规则只适用于触发输入。
3.5.3	文字调整	规则 b)关于输出，进行了微调，以便于理解。
3.5.6	纠正	数据运算的定义 “除了进/出的数据移动……其他任何行为”，进行了修改以明确这只适用于功能处理内部处理的数据。
3.5.7	文字调整	在开篇中描述的“通用软件模型假设，通常…”是错的，已删除。因为这可能会暗示模型是允许异常的，但其实是没有的。修改为“通常情况下…” 对“数据移动唯一性”规则的部分文字调整，增加了一条注释，为了提高易读性。尤其是，在条款d)的最后一句，已删除，为了避免理解有误。这句话没有错，但是可能与新增的区分数据组和兴趣对象的规则有冲突（见3.3.2节）。 更新了例子10；目前该例只与规则d)相关。
3.5.10	文字调整	定义中关于“控制命令”的例子被移到外面。例子重新编排， 更加清晰。
3.5.8	文字调整	新增一段，为了解释度量人员必须首先决定是从待度量软件边界内的持久存储介质中检索数据，还是通过其他软件块的持久存储介质检索数据。
3.5.11	文字调整	业务应用案例3重新编写，提高易读性。
4.3.1	方法	规则g)中关于整合度量结果的两部分分为两条规则，因为这两部分是互不相关的。
4.4.2	方法	度量功能变更后的软件规模的公式从文字描述变为一个规则，并增加了一个例子。
4.5.4	文字调整	增加一个注释，以说明COSMIC功能规模度量已经成功应用于度量用户故事和敏捷开发。
附录C	纠正	例6）并不局限于“在同一软件内”，因此删掉此短语。

附录 F—术语表

根据本章节中的定义，以下术语用于 COSMIC 功能规模度量方法（“COSMIC 方法”）。已被 ISO 定义的术语（如“功能规模度量”或“度量单位”）及其 ISO 的定义都已被 COSMIC 方法采纳。

对于术语表中所列的大部分术语，适当时，都标明了后缀“类型”。由于任何功能规模度量方法的目标是识别数据或功能的“类型”而不是“实例”，所以在整个 COSMIC 方法中，我们总是关注“类型”而不是“实例”。因此，为了方便阅读，在正文中我们将省略后缀“类型”，除非当我们特别需要区分类型和实例。这也是国际标准(ISO/IEC 19761:2011)所采纳的关于 COSMIC 方法的惯例。偶尔在起草这些定义时，这种惯例会导致一些困难——请看下文“数据移动类型”定义中的注 3（没有出现在国际标准中）。

关于类型和实例的完整讨论参见 1.3.3 节。

注意：只在特定领域的 COSMIC“指南”中使用的术语被定义在那些指南中，它们不在下文中列出。

在下文列出的定义中：

- 为了便于互相参照，在本术语表中其他地方定义的术语加了下划线。
- 源自 ISO 标准关于 COSMIC 方法的术语(ISO/IEC 19761)或 COSMIC 方法特有的术语均已***加粗斜体***显示。
- 其它已被 ISO 采纳但非 COSMIC 方法特有的术语**加粗**显示。

应用软件　通过电脑收集、保存、处理和展示数据的软件系统。

注意：这是 ISO/IEC 24570:2005 软件工程 -- NESMA 2.1 版功能规模度量方法所给定义的修改版。

（“应用软件”的另一种定义）与控制电脑本身的软件不同，应用软件被设计用于帮助用户执行特定的任务或处理特定类型的问题。

注意：这是 ISO/IEC 24765:2010 系统与软件工程词汇 4.5 所给定义的细微修改版。

应用程序常规信息　与应用程序相关，而不与特定功能处理的兴趣对象相关的所有信息。

基础功能构件（BFC）　出于度量目的，用 FSM 方法定义的功能性用户需求的基本单位[17]。

注意：COSMIC 方法定义一个数据移动类型为一个 BFC。

基础功能构件类型（BFC 类型）BFC[17]定义的类型。COSMIC 方法有四个 BFC 类型，输入、输出、读和写（类型）。

边界　被度量软件及其功能用户之间的一个概念性接口。

注意：从以上定义可以引申出，在同一层或不同层间的任何两个有数据交换的软件块之间，存在一个边界，此时，一个软件块是另一软件块的功能用户，反之亦然。

构件　软件系统中单独存在的部分。独立出来是出于软件结构体系的原因，和/或由于被独立定义、设计或开发的。

控制命令　人类功能用户用来控制软件使用的命令，但不包含任何对被度量软件 FUR 中定义的兴趣对象的数据移动。

注意：控制指令不是数据移动，因为它不移动关于兴趣对象的数据。

COSMIC 度量单位 1CFP(COSMIC 功能点)，被定义为一个数据移动规模。

注意：3.0 以前的版本中，度量单位称为“Cfsu”（COSMIC 功能规模单位）。

数据属性类型（同义词为“数据元素类型”） 在一个已识别的数据组里信息的最小信息单元，传递了软件功能性用户需求的一个含义。

数据组类型 一个唯一的、非空的、无序的数据属性的集合，包含的每个数据属性描述了同一个兴趣对象的一个互补的侧面。

注：术语“数据组”并不一定指的是“描述某个兴趣对象的所有数据属性的集合”。软件的 FUR 可以根据不同的功能处理的需要，把描述同一兴趣对象的数据属性任意组合成为数据组。

数据运算 一个功能处理在处理数据时，除了进/出功能处理的数据移动，以及在功能处理和持久存储介质之间的数据移动之外，发生在数据上的任何事情。

数据移动类型 移动单个数据组类型的基本功能构件。

注 1：数据移动类型有四种子类型，即：输入、输出、读和写（类型）。

注 2：基于度量目的，每种数据移动被认为负责了某些与之相关的数据运算。详见 3.5.6 节。

注 3：更准确地说，是一次数据移动（而不是数据移动类型）实际移动了一个数据组的实例（不是类型）。这种解释也适用于输入、输出、读和写的定义。

E. 是“输入类型”的缩写。

输入类型 一种数据移动，将一个数据组从功能用户跨越边界移动到需要它的功能处理。

注意：输入被认为负责了某些相关的数据运算—详见度量手册。

错误/确认消息 由功能处理发送给人类用户的输出，要么是确认输入数据已被接收，要么是提示输入数据存在错误。

注意：任何包含错误标志、但不是发送给人类功能用户的输出，都不是错误/确认消息。

事件类型 发生的某事。

输出类型 一种数据移动，将一个数据组从功能处理侧跨越边界移动给需要它的功能用户。

注意：输出被认为负责了与之相关的某些数据运算—详见度量手册。

功能处理类型

a) 体现了待度量软件的功能性用户需求基本部件的一组数据移动，该功能处理在这些 FUR 中是独一无二的，并能独立于这些 FUR 的其他功能处理被定义。

b) 一个功能处理必须只有一个触发输入。每个功能处理在接受到由其触发输入数据移动所移动的一个数据组后，开始进行处理。

c) 一个功能处理的数据移动的集合是响应触发输入的所有可能的功能性需求所需要的集合。

注 1：实现时，一个功能处理实例，在收到一个触发输入实例移动的数据组实例时，才开始执行处理。

注 2：除了触发输入外，一个功能处理的 FUR 可能需要一个或多个其他的输入。

注 3：如果功能用户发送了包含错误的数据组，例如，由于传感器失灵，或者人输入的命令存在错误，通常是由功能处理来判断事件是否确实发生、和/或输入的数据是否有效、以及如何响应。

注 4：如果一个功能处理是由一个来自于 FUR 的、独一无二的触发事件产生的触发输入而引起的，那么该功能处理是独一无二的（如 a）所述），且其规模应计入该 FUR 规模中。同一段 FUR 描述的两个或多个功能处理可能是互不相同的，即使他们共享某些功能。见 3.2.7 节的例子，共享相同功能的功能处理。

功能处理颗粒度级别 对软件块描述的一个颗粒度级别，在该级别上：

- 功能用户（类型）是单独的人、工程设备或软件块（而不是它们的任何组合），并且
- 软件块应响应的是单个的事件（类型）（而不是定义为事件组的任何颗粒度级别），参见注 3。

注 1：在实践中，软件文档对功能性需求的描述在不同部分通常具有不同的颗粒度级别，尤其当文档还在演化时。功能处理可以在其中任意颗粒度级别显现出来。

注 2：人、工程设备或软件块的组合（功能用户），举例来说，可能是一个“部门”，它的成员处理多种类型的功能处理，或者是一个有多种仪器的“控制面板”，或者是“中央系统”。

注 3：“事件组”，举例来说，如在一个高颗粒度级别的功能性需求描述中提到的会计软件系统中“销售交易”的输入事件流，或者航空电子软件系统中“飞行命令”的输入事件流。

功能规模 通过量化功能性用户需求得出的软件规模。[17]

功能规模度量（FSM） 度量功能规模的过程。 [17]

功能规模度量方法 对由一系列规则（这些规则符合 ISO/IEC 14143-1:1998 所规定的特征）所定义的 FSM 的实施过程。[17]

功能用户 一个（类）用户，是软件块的功能性用户需求中软件所处理数据的发送者或预期的接收者。

功能性用户需求（FUR） 用户需求的子集。这些需求以任务和服务的形式描述软件做什么。

注 1：功能性用户需求包含但不限于：

- 数据传输（比如输入客户数据；发送控制信号）
- 数据变换（比如计算银行利息；导出平均温度）
- 数据存储（比如存储客户订单；记录随时间变化的环境温度）
- 数据检索（比如列出当前雇员；检索飞机最新的位置）

属于用户需求但不是功能性用户需求的例子包括但不限于（尽管随着解决方案的详细定义，某些需求会转变为真正的 FUR）：

- 质量约束（比如可用性、可靠性、效率和可移植性）
- 组织约束（比如操作场所、目标硬件和遵从的标准）
- 环境约束（比如互操作性、保密性、隐私和安全性）
- 实现约束（比如开发语言、交付日期）

注 2：在 COSMIC 文档中，术语“FUR”特指如下的功能性用户需求：

- 从已有的软件制品（如需求文档、设计文档及物理制品等）中提取的；
- 必要时，通过合理的假设来克服现有制品中含糊不清的地方；
- 包含所有 COSMIC 功能规模度量所需要的信息。

除此之外，根据上下文的需要，我们会使用“功能性需求”“实际需求”或“物理制品”等词汇。

输入数据 被某一给定功能处理的输入类型移动的数据。

层 一个软件系统体系结构的功能划分。

分解层级 将软件块分解为构件而形成的任意级别（例如，称为“1 级”），然后将构件分解为子构件（“2 级”），再将子构件分解为子—子构件（“3 级”），等等。

注 1：不要和“颗粒度级别”混淆。

注 2：软件块中构件的规模度量结果只可以和处于同一分解层级的构件直接比较。

注 3：一个软件块不同的分解等级可能对应不同的软件层次。例如图 2.4。并且，软件可以分解为不同“层级”，与软件是否按照分层架构模式进行设计无关。

颗粒度级别 对于一个软件块任意部分的描述（例如：对需求的陈述或者对软件块结构的描述）的任意扩展级别，每一次进一步的扩展，对软件块功能的描述也更加细化并具有一致的详细程度。

注意：度量人员应该意识到当需求在软件项目的早期演化过程中，在任何时候，需要的软件功能的不同部分通常以不同的颗粒度级别被文档化。

度量方法 [13] 用通用语言描述的一系列用于执行度量的有逻辑的操作顺序。（一般描述）

度量（策略）模式 在度量特定软件功能领域的软件块规模时可以使用的一个标准模板。此模板定义了可能与软件交互的功能用户类型、软件的分解层级及软件可能会处理到的数据移动类型。

模型 [15] 用于使无法直接观察到的概念形象化的描述或类比。

修改（一个数据移动的功能）

a) 如果以下情况至少一种适用，那么数据移动就是在功能上被修改了：
   - 移动的数据组被修改了，
   - 关联的数据运算被修改了。

b) 如果以下情况至少一种适用，那么数据组是被修改了：
   - 一个或多个新属性添加到数据组中，
   - 一个或多个现有属性从数据组删除，
   - 一个或多个现有属性被改变，例如意义或格式上（而不是它们的值）发生改变，

c) 如果发生任何功能性变化，一个数据运算就被改变了。

非功能需求 任何关于一个硬件/软件系统或软件产品的，除软件的功能性用户需求以外的软件部分的需求，包括如何开发、维护以及在操作中如何执行它。非功能需求关注于：

- 软件质量；
- 软件实现的环境或其所服务的环境；
- 开发或维护此软件所用的流程和技术；
- 软件运行所用的技术。

注：随着项目的演化，最初呈现为非功能的系统或软件需求会部分或完全转变为软件的 FUR。

兴趣对象类型 从功能性用户需求中识别出来的、存在于功能用户世界中的任何“事物”，软件要为之输入/输出数据组，和/或向/从持久存储介质移动数据组。可能是具体的事物，也可能是概念性对象或其一部分。

注 1：在 COSMIC 方法中，采用“兴趣对象”术语，以避免与特定的软件工程方法相关。该术语并不意味着是面向对象方法中的“对象”。

注 2：当一个功能用户发送了一个描述其本身的数据组，比如它的状态或者它的身份，或当一个功能用户接收了一个描述其本身的数据组，那么该功能用户也同样满足“事物”的定义，因此它也是该数据组所描述的兴趣对象。

注 3：没有绝对的兴趣对象。即使在同一个待度量的软件中，一个“事物”在一个或多个功能处理中，对于某个功能用户来说是感兴趣的对象；但可能在其他功能处理中就不是另一个功能用户感兴趣的对象。

操作环境（软件） 应用软件所依赖的、在某特定电脑系统上并发执行的软件集合。

输出数据 被某一给定功能处理的输出类型移动的数据。

软件的对等块 若两个软件块处于同一层，则它们是相互对等的。

持久存储介质 使得功能处理在其生命周期结束后仍然能够存储数据组的存储介质，并且/或者，通过该存储介质，功能处理也可以检索数据组，此数据组由另一个功能处理存储，或由同一功能处理之前的事件存储、也可能由某些其他过程存储。

注 1：在 COSMIC 方法中，持久存储介质是一个只存在于被度量软件边界内的概念，因此，它不能被视为被度量软件的功能用户。

注 2：“某些其他过程”的一个例子是只读存储器的生产过程。

软件块 任何分解级别（包括从整个软件系统级别到软件系统最小构件的级别）中的任意一个独立部件。

度量目的 定义了为什么需要度量和度量结果用途的阐述。

R “读类型”的缩写。

读类型 一种数据移动，将一个数据组从持久存储介质移动到需要它的功能处理。

注意：读被认为负责了与之相关的某些数据运算—详见度量手册。

缩放(一个度量） 把某一规模度量在两种度量单位之间换算的过程。

度量范围 在一次具体的功能规模度量活动中所包括的功能性用户需求的集合。[17]

注：（仅针对 COSMIC 方法）应该区分“总体范围”与总体范围内的各软件块的“范围”，“总体范围”是指根据目的应度量的所有软件，而各软件块的规模应该分别度量。本手册中术语“范围”（或“度量范围”）将关联到规模必须单独度量的一个软件块。

软件 [17] 计算机指令、数据、程序以及操作文件（如果有的话）的集合。此集合用于满足特定的目的，这些目的可从功能角度入手，通过由功能性用户需求、技术和质量需求组成的有限集合来描述。

软件系统 只由软件组成的系统。

子处理类型 功能处理的一部分，它可以移动数据（从功能用户移动到软件内，或把数据从软件移动到功能用户，或把数据移至或移出持久存储介质）或者运算数据。

系统 硬件、软件及人工程序的组合，此组合用于达到既定目的。

注意：此定义是 ISO/IEC 15288:2008 定义的改编。在 COSMIC 定义中，“硬件、软件及人工程序”替换了 ISO/IEC 定义中的“交互元素”。

触发输入类型 触发输入是一个功能处理的输入数据移动，它移动了功能用户产生的一个数据组，该数据组是功能处理开始执行处理所必需的。

注：该定义来自通用软件模型，该模型是一个逻辑模型。实际上，一个功能处理可能在数据还没有输入前就开始运行了。如：当一个人类用户点击菜单，显示了空白界面等待数据输入。

触发事件类型 待度量软件的功能性用户需求中可识别的一个事件，此事件使得一个或多个待度量软件的功能用户产生一个或多个数据组，第一个产生的数据组（可能是由任何一个功能用户所产生的）随后被一个触发输入所移动。一个触发事件不可再拆分，并且要么已经发生，要么没有发生。

注：时钟和定时事件可以作为触发事件。

度量单位 [14] 根据惯例定义和采纳的一个特定量，同种类的其它量可与此进行比较，从而表示出它们的相对大小。需要注意的是：度量单位约定俗成地配有名字和符号。

参见“COSMIC 度量单位”

用户 [17] 任何时候与软件进行沟通或交互的任何人或事物。

注意： “事物”的例子包括（但不局限于）软件应用程序、动物、传感器或其它硬件。

值（某一数量的） [14] 某特定量的大小，一般表达为度量单位乘以某数所得的值。

W “写类型”的缩写。

写类型 一种数据移动，将一个数据组从功能处理内部移动到持久存储介质中。

注意：写被认为负责了与之相关的某些数据运算—详见度量手册。

X “输出类型”的缩写。

附录 G—变更请求和建议程序

COSMIC 度量实践委员会（MPC）非常愿意接受反馈、建议，以及对此指南的变更请求（如果必要的话）。本附录展示了如何与 COSMIC MPC 联系。所有与 COSMIC MPC 的联系都应该通过电子邮件的方式发送到下面的地址：
mpc-chair@cosmic-sizing.org

非正式的反馈和建议

关于此指南的非正式建议和/或反馈，比如理解或应用 COSMIC 方法的任何困难，一般性改进的建议等，都可以通过电子邮件发送到上面的地址。消息会被登记下来，并且通常在收到后的两周内给予答复。度量实践委员会不保证对一般性意见给予答复。

正式的变更请求

本指南的读者如发现某个文字缺陷，或不足之处需要澄清，或者一些文字需要加强，那么可以提交一个正式的变更请求（“CR”）。正式的 CR 会被登记下来，并在收到后的两周内给予答复。每个 CR 将分配一个序列号，在 COSMIC MPC 的成员中循环传递，COSMIC MPC 是一个世界范围内 COSMIC 方法的专家组。他们的正常评审周期最少需要一个月，如果变更请求很难解决，可能需要更长的时间。评审的结果可能是 CR 被接受，或者拒绝，或者“未决待进一步讨论”（后面这种情况，例如本 CR 要依赖另一个 CR），结果会尽快地反馈给提交者。

只有包含了下面的所有信息，一个正式的 CR 才会被接受。

- 提交 CR 的人员姓名、职位和单位
- 提交人的详细联系方式
- 提交日期
- 关于 CR 的目的的总体陈述（如“需要改进文本……”）
- 需要变更、替换或删除的实际文字（或澄清引用出处）
- 建议增加或替换的文字
- 关于变更的必要性的充分解释

提交 CR 的表格可以从门户网站 www.cosmic-sizing.org 获得。COSMIC MPC 对 CR 的评审结果，以及在哪个版本应用这个 CR（如果被接受了的话）的决定是最终的决定。

COSMIC 方法应用的问题

COSMIC MPC 抱歉不能回答与 COSMIC 方法应用相关的问题。有商业机构能够提供本方法的培训和顾问工作，或者支持工具。详细情况请咨询 www.cosmic-sizing.org 网站。