**比较课堂上所讲到的软件工作量估计方法**

* 由低向上估计

将项目分成部分任务，再估算每个任务所需要的工作量，分解工作将持续进行到任务不可继续分解。该方法主要运用在项目的后期，在前期使用的话需要进行很多的假设。对于一个全新的没有借鉴的项目，用该方法很有效。

* 自顶向下估计

先估算项目整体的工作量，然后再将其分配到各个模块或子任务，适合在项目前期，当详细设计和具体实现尚未明确时使用。通常参考类似项目的历史数据来进行估算。

* 参数模型

参数模型是一种基于数学公式或经验模型的方法，通过关键参数（如代码行数、功能点等）来估算软件开发的工作量、时间和成本。

* Delphi估算法（专家估算）

多位专家独立对项目的工作量或时间进行估算，收集各专家的估算结果并进行汇总分析，将汇总结果反馈给专家，允许他们基于其他人的意见进行修正，经过多轮迭代后达成共识，最终形成估算值。

* 类比估计法

通过查找和分析与当前项目相似的历史项目（在规模、复杂性、技术栈等方面），来预测当前项目的工作量。在项目初期，详细信息不足时尤为适用，因为它不需要深入的任务分解或模型参数。

* 功能点方法

功能点方法将系统的功能划分为五种类型，并为每种类型分配相应的权重：外部输入（EI）、外部输出（EO）、外部查询（EQ）、内部逻辑文件（ILF）、外部接口文件（EIF），计算功能点加权值之和，最后乘上调整因子，得到最后的功能点总数，以功能点总数估计工作量。这种方法在从需求阶段到开发后期均可使用，且计算结果与技术平台无关、与实现的人员无关，系统之间的规模具有可比性，是客观度量，不依赖于参与估算的人员。

* COSMIC-FFP估算方法（CFP）

COSMIC-FFP通过识别和计数四种基本的数据移动类型来测量软件的功能规模：输入、输出、读、写，每个数据移动被计为一个功能点（Function Point），通过汇总这些功能点，得到软件的总功能规模。

* 对象点方法

对象点是用于量化软件系统规模的单位。一个对象可以是：

用户界面（Screens）：例如窗口、表单、对话框。

报表（Reports）：例如输出报表、统计报表。

第三方组件（3GL Components）：例如外部调用的库、模块、服务。

每个对象根据复杂度（简单、中等、复杂）赋予不同的权重，并通过加权求和得到对象点的总量。这种方法贴合面向对象方法的开发过程，对象点的定义与具体技术实现无关，从用户交互界面和输出结果出发，便于非技术人员理解，但不适用技术驱动型项目比如算法密集型或底层开发。

下表显示了各个方法的纵向对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **方法** | **定义** | **依据** | **适用阶段** | **优点** | **缺点** |
| 自底向上估计法 | 从细节任务入手，逐项估算工作量，然后汇总为整体工作量。 | 详细的任务清单 | 项目中后期，任务明确时 | 精确度高；适合复杂项目的详细规划。 | 时间成本高；需明确所有任务细节，难以适应早期阶段。 |
| 自顶向下方法 | 从项目整体入手，逐步分解为小的子任务进行估算。 | 历史经验和整体需求 | 项目前期，信息不足时 | 简单易用，快速提供整体估算；适合早期阶段。 | 精确性低；缺乏细节支持，难应对项目变更。 |
| Delphi估算法（专家估算） | 通过多轮匿名专家评估和反馈达成一致的工作量估算方法。 | 专家经验和多轮讨论结果。 | 项目前期或缺乏详细数据时。 | 适合信息不足或高度不确定的项目；利用专家知识，避免个体偏差。 | 需要专家参与；多轮讨论耗时；主观性较高，依赖专家的能力和经验。 |
| 类比估计法 | 通过对比与当前项目类似的历史项目来推测工作量和成本。 | 历史项目数据 | 项目前期，信息不足时 | 快速估算，易于理解；适合有丰富历史数据的组织。 | 依赖类似项目的质量和可用性；主观性较强。 |
| 功能点方法 | 基于软件的功能需求，通过功能点数计算工作量。 | 功能点（FP） | 需求分析阶段 | 技术无关性，标准化；适合需求明确、功能驱动型项目。 | 主观性较强，依赖需求的完整性；不适用于技术复杂型项目。 |
| 对象点方法 | 基于软件中用户界面、报表和第三方组件的复杂性进行估算。 | 用户界面、报表、第三方组件数量与复杂性 | 面向对象开发项目 | 适合 GUI 密集型应用和面向对象开发；估算贴近用户视角。 | 主观性较强，复杂度分类偏差影响结果；不适用算法密集型项目。 |
| CFP 方法 | 基于数据移动（Entry、Exit、Read、Write）的分析，量化软件功能规模。 | 数据移动数量（Entry、Exit、Read、Write） | 功能需求明确时 | 标准化方法，独立于技术；适合实时系统、嵌入式系统等复杂系统。 | 需要详细的数据移动分析；对非功能需求考虑有限。 |
| 参数化模型 | 使用数学公式结合关键参数（如 LOC、FP）进行估算，例如 COCOMO 和功能点模型。 | 参数（LOC、FP、调整因子等） | 项目前期或详细阶段 | 科学性强，基于数据和统计；适合有大量历史数据的组织。 | 模型复杂性较高，依赖参数质量；不适合无历史数据的新型项目。 |

功能点估算法的比较

1．IFPUG-FPA

（1）简介

IFPUG的功能点分析（FPA）方法是一种目前被广泛接受的关于软件规模度量的有效方法，从用户的角度出发，将系统分为数据功能和交易功能两大类，分别根据具体的规则来计算功能点，最后结合系统的特征因子来调整功能点数，从而得到最终的系统规模。

（2）主要功能元素

数据功能：

①　ILF – Internal Logical File – 内部逻辑文件

②　EIF – External Interface File – 外部接口文件

交易功能

①　EI – External Input – 外部输入

②　EO – External Output – 外部输出

③　EQ – External Query – 外部查询

（3）适用对象

管理信息系统（开发项目，升级项目，在用项目）

（4）普及程度

高，应用最广泛

（5）优点

从用户角度度量软件规模，使用人群多，易于交流

评估方法成熟，方法比COSMIC细致

（6）缺点

操作复杂，功能分解无方法指导，拆分结果和整体评估结果不一致

2．NESMA

（1）简介

NESMA（Netherland Software Measurement Association，荷兰软件度量协会）功能点估算方法基于IFPUG发展而来。它与IFPUG的概念统一，相对于IFPUG，NESMA显得更加灵活方便，并且更适合项目早期预算、招投标阶段的成本评估，针对IFPUG方法分析过程较复杂、计算工作量大的不足，NESMA方法实现了快速估算。

NESMA提供了三种类型的功能点估算方法，分别是：

①　指示法：

一般用于计划阶段，因为此阶段需求文件多不完善，故而只需关注逻辑文件即可。

②　估算法：

一般用于执行阶段，此时需求文件较为完善，故需要关注逻辑文件和相应的操作；

③　详细法：

主要用于事后评估阶段，此时功能需求非常详细，可关注逻辑文件、相应操作和复杂度。

在IFPUG的基础上，NESMA增加了指示法（Indicative）和估算法（Estimated），使得用户可以在需求不完善的情况下，快速估算软件规模。

从应用角度而言，IFPUG和NESMA标准是国际上最主要的标准，国际基准比对组织中超过90%的数据采用IFPUG/NESMA方法，国内的行业数据百分百采用IFPUG/NESMA方法，由于IFPUG方法和NESMA方法被认为是基本等效的，所以近几年，这两种方法被各行业大量采用。但如想在早期（如预算）阶段进行度量，NESMA是更好的选择。

（2）主要功能元素

数据功能：

　ILF – Internal Logical File – 内部逻辑文件

　EIF – External Interface File – 外部接口文件

交易功能

　EI – External Input – 外部输入

　EO – External Output – 外部输出

　EQ – External Query – 外部查询

（3）适用对象

管理信息系统

（4）普及程度

较为普及

（5）优点

从用户角度度量软件规模，使用人群多，易于交流

评估方法成熟，更适合早期测算

（6）缺点

操作复杂，功能分解无方法指导，拆分结果和整体评估结果不一致

3．COSMIC-FFP

（1）简介

通用软件度量国际协会（Common Software Measurement International Consortium ，COSMIC)提出的全功能点分析方法，COSMIC-FFP不仅适合于信息系统的规模度量，还适合于实时系统和多层系统的规模度量，已经被ISO接受为国际标准。该方法可以在软件开发生命周期的各个阶段使用，从用户功能的视角入手，起源于客户可以理解的术语，不需要调整因子，简单易行。COSMIC-FFP 通过入口(Entry)、出口(Exit)、读取(Read)和写入(Write)4 种数据流类型来确定软件系统功能的规模。先将软件的功能分解为功能处理、数据组、数据属性，然后将功能处理分解为数据移动，根据数据移动来计算软件规模。

（2）主要功能元素

数据移动，入口(Entry)、出口(Exit)、读取(Read)和写入(Write)

（3）适用对象

银行、财务、保险等领域管理信息系统，非复杂计算系统

实时系统，如电话交换系统，嵌入式控制软件等，飞机定票系统等。

（4）普及程度

较普及，第二代功能点方法，部分人认为是趋势

（5）优点

计算规则简单、直接，不需要调整因子，理解后易于操作，可重复性高，支持实时系统

（6）缺点

普遍认为理解比较困难，不考虑数据计算，无法应用于分析性系统

4．UK-MKII

（1）简介

英国软件度量协会提出的Mark II 功能点法（主要在英国、香港等地使用），只测量逻辑和业务需求，不依赖于具体实现方式，从逻辑事务的角度出发，识别输入、处理、输出的过程，计算数据元类型的数量，计算功能规模。

（2）主要功能元素

输入Ni、输出No，处理Ne

（3）适用对象

管理信息系统或者嵌入式系统

（4）普及程度

香港、英国部分的确使用

（5）优点

比IFPUG简单，易于理解，系统整体度量和部分度量总和一致

（6）缺点

使用不广泛，获取资料难；

19项调整因子，增加主观性，调整因子主观性强，不易操作

5．COCOMOⅡ

（1）简介

COCOMO(Constructive Cost Model，构造型成本模型)。Barry Boehm研究大量项目数据后，推导出的一个成本模型，在实际应用中取得了较好的效果，很好的匹配了采用瀑布模型的软件项目。做了调整和改进，提出了一个新的版本-COCOMOII型中使用三个螺旋式的过程模型：应用组装模型，早期设计模型和后体系结构模型。

① 应用组装模型(Application Composition)

应用组装模型是基于对象点的度量模型，它通过计算屏幕、报表、第三代语言(3GL)模块等对象点的数量来确定基本的规模，每个对象点都有权重，由一个三级的复杂性因子表示，将各个对象点的权值累加起来得到一个总体规模，然后再针对复用进行调整。

② 早期设计模型(early design)

在项目开始后的一个阶段或者螺旋周期通常包括探索体系结构的可供选择方案或增量开放测量。为支持这一活动，COCOMOII提出了一个早期设计模型，这一模型使用功能点和等价代码行估算规模。

③ 后体系结构模型(post architecture)

一旦项目进入开放阶段，就必然确定一个具体的生命周期体系结构，此时项目就能够为估算提供更多更准确的信息。

首先采用软件规模估算方法对项目的规模进行估算。再应用五个比例因子，通过相关计算，将规模转化为工作量，并通过十七个成本驱动因子对工作量进行调整。最后，采用进度计算公式，计算出开发该项目所需要的进度以及人数。

（2）主要功能元素

DSI（ 源指令条数 ） ，定义为代码行数，包括除注释行以外的全部代码；MM（ 度量单位为人月 ） 表示开发工作量；TDEV（ 度量单位为月 ） 表示开发进度，由工作量决定。

（3）适用对象

相对较小、较简单的项目和同硬件结合紧密的嵌入性的项目

（4）普及程度

（5）优点

计算精确

（6）缺点

受因子影响巨大，项目前期不容易评估代码行数；缺乏实施指导性文件