没有银弹

思考软件工程中的根本和次要问题

Fredrick P. Brooks

1986

南京大学 软件学院

NANJING UNIVERSITY SOFTWARE INSTITUTE

No Silver Bullet （1986）

- 没有任何一种单纯的技术或管理上的进展，能够独立地承诺十年内使生产率、可靠性或简洁性获得数量级上的进步。
- 所有大家看到的技术、管理方法都不会给软件开发带来意想不到的效果。
- 软件开发在根本上就是困难的。

摘要

- 根本任务(essential)——打造由抽象软件实体构成的复杂概念结构。
- 次要任务(accidental)——使用编程语言表达这些抽象实体，在空间和时间限制内将它们映射成机器语言。
- 除非次要任务占了所有工作的9/10，否则即使全部次要任务的时间缩减到零，也不会给生产率带来数量级上的提高。

The Werewolf of Eschenbach(1685)

Silver bullet – 银弹

- 人狼 - - 可以完全出乎意料地从熟悉的面孔变成可怕的怪物。
- 消灭人狼 - - 银弹。
- 软件项目 - - 常常看似简单明了的东西，却有可能变成一个落后进度、超出预算、存在大量缺陷的怪物。

软件开发与硬件相比

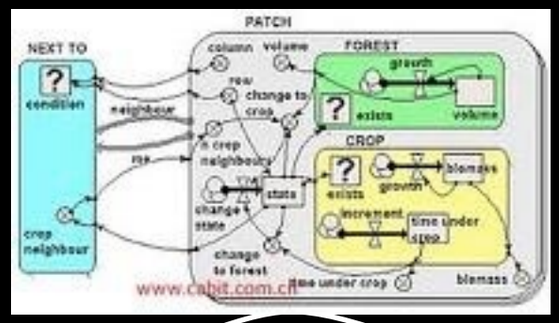
- 为什么软件的发展比硬件慢?
- 不是软件发展慢，而是硬件发展太快。
- “从人类文明开始，没有任何其他产业技术的性价比，能在30年之内取得6个数量级的提高，也没有任何一个产业可以在性能提高或者降低成本方面取得如此的进步。这些 进步来自计算机制造产业的转变，从装配工业转变成流水线工业。”

探寻软件产业发展的问题

- 按照亚里士多德的观点，将软件开发的问题分成根本的(essence)——软件特性中固有的困难，次要的(accident)——出现在目前生产上的，但并非那些与生俱来的困难。

- 一个相互牵制关联的概念结构，是软件实体必不可少的部分，它包括：数据集合、数据条目之间的关系、算法、功能调用等等。这些要素本身是抽象的，体现在相同的概念构架中，可以存在不同的表现形式。尽管如此，它仍然是内容丰富和高度精确的。

概念结构



高级语言

中级语言

低级语言

编译器

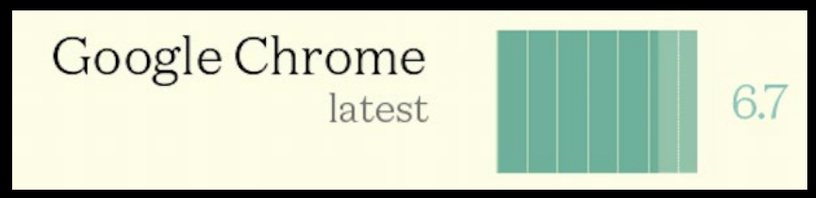
人

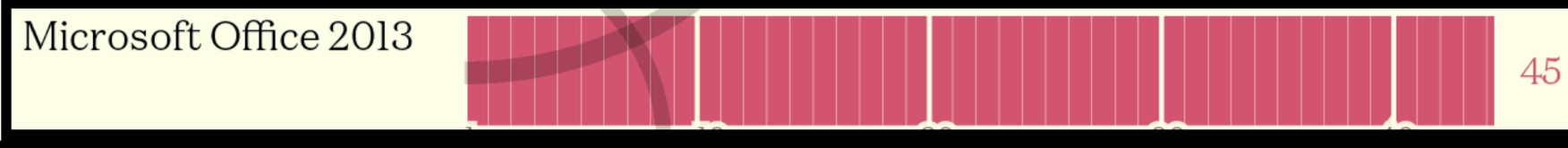
根本困难 - 软件特性中固有的困难

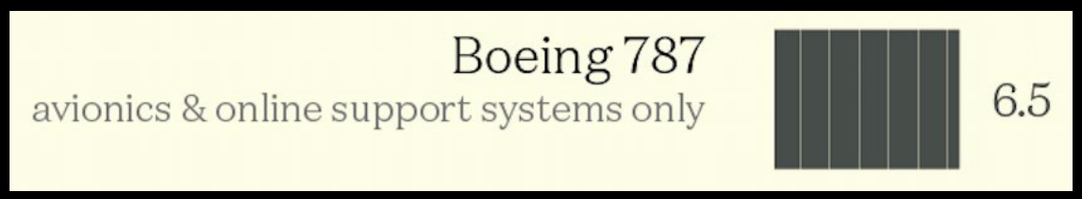
- **我认为软件开发中困难的部分是规格化、设计和测试这些概念上的结构，而不是对概念进行表达和对实现逼真程度进行验证。**

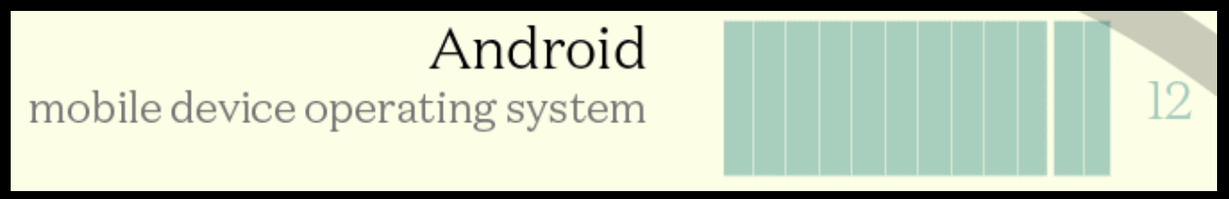
- 如果这是事实，那么软件开发总是非常困难的。天生就没有银弹。
- 现代软件系统中无法规避的内在特性：复杂度、一致性、可变性和不可见性。

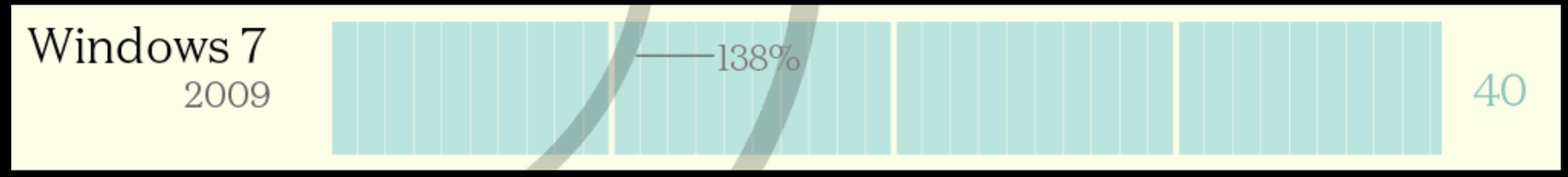
代码量（百万代码行）

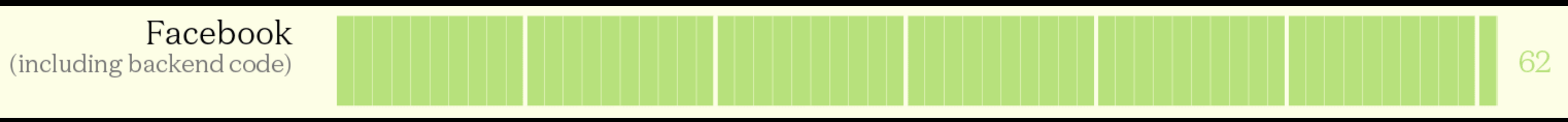


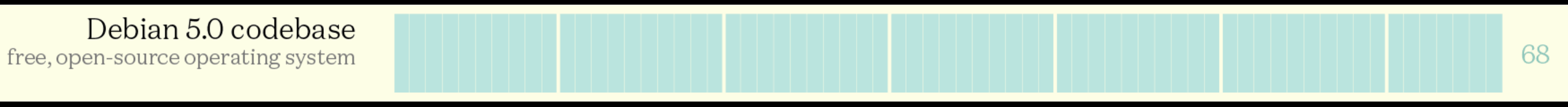


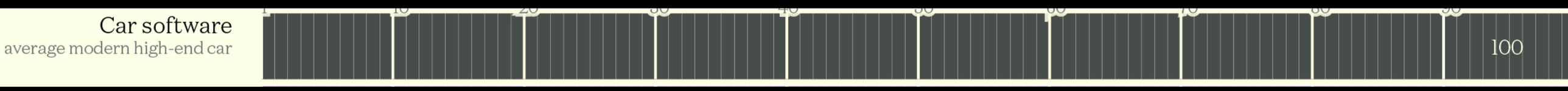












2Billion

复杂度1

- （1）软件实体可能比任何由人类创造的其他实体都要复杂，因为没有任何两个软件部分是相同的，如果有我们会将它们合并。
- （2）数字计算机本身就比人类建造的大多数东西复杂。计算机拥有大量的状态，这使得构思、描述和测试都非常困难。软件系统的状态又比计算机系统状态多若干个数量级。

复杂度2

- （3）软件实体的扩展也不仅仅是相同元素重复添加，而必须是不同元素实体的添加。整个软件的复杂度以很大的非线性级数增长。

复杂度3

- （4）软件的复杂度是必要属性，不是次要因素。抽掉复杂度的软件实体描述常常也去掉了一些本质属性。
- 数学家和物理学家们建立模型以简化复杂的现象，从模型中抽取出各种特性，并通过试验来验证这些特性。这是因为模型中忽略的复杂度不是被研究现象的必要属性。

复杂度4

- 复杂度问题造成软件产品开发问题
- 团队成员之间的沟通非常困难，导致了产品瑕疵、成本超支和进度延迟
- 由于复杂度，列举和理解所有可能的状态十分困难，影响了产品的可靠性

复杂度5

- 由于函数的复杂度，函数调用变得困难，导致程序难以使用
- 由于结构性复杂度，程序难以在不产生副作用的情况下用新函数扩充
- 由于结构性复杂度，造成很多安全机制状态上的不可见性

复杂度6

- 复杂度引发管理上的问题
  - 全面理解问题变得困难，从而妨碍了概念上的完整性
  - 它使所有离散出口难以寻找和控制
  - 它引起了大量学习和理解上的负担，使开发慢慢演变成了一场灾难
  - **信息隐藏原则**

Windows 7 team

- 微软开发windows 7的工程量是大于阿波罗登月计划的。
- Windows 7操作系统，据微软官方给出的数据，就有超过5000万行代码，并且这套操作系统由微软的平均约40人的23个研发小组历时三年研发。要知道在windows之前已经有了windows XP和windows vista的技术积累，而且windows很多核心代码其实并没有推到windows vista等技术重建。
- 几千万行代码的工作量是个什么概念？举个参照系吧，很多人在用的金山的WPS代码行数约有150万行，而金山软件公司重写这个级别的办公软件WPS，用了3年时间。

一致性

- 物理学家面对异常复杂的事物，他们坚信必定存在着某种通用原理。物理学是研究上帝创造的东西。
- 软件开发面对的复杂度往往是随心所欲、毫无规则可言的，来自若干必须遵循的人为惯例和系统。软件开发面对的是人，不是上帝。
- 很多复杂性来自保持与其他接口的一致。

可变性

- 软件实体经常会遭受到持续的变更压力。
- 汽车、建筑可以修改，但很少有人修改，大家都知道成本很高。
- 软件包含了很多功能。
- 软件可以很容易地进行修改——它是纯粹思维活动的产物，可以无限扩展。

- 软件的变更
  - 人们要求扩展，更改功能
  - 硬件的变化
- 软件与整个社会联成一体，后者在不断变动，它强迫软件也跟着变动。

不可见性

- 软件是不可见的和无法可视化的
  - 抽象的功能：几何抽象、机械制图、化学分子模型
  - 帮助我们捕获物理存在的几何特性。
- 软件的客观存在不具有空间的形体特征
  - 现在没有任何一种2维、3维的图形可以描述软件。

- 这限制了个人的设计过程，也严重的阻碍了相互之间的交流。
- UML

没有银弹

- 相对必要任务而言，软件工程师在次要任务上花费了多少时间和精力？除非它占了所有工作的9/10，否则即使全部次要任务的时间缩减到零，也不会给生产率带来数量级上的提高。

当年的银弹(1986)

- Ada和其他高级编程语
- 面向对象编程
- 人工智能
- 专家系统
- “自动”编程
- 图形化编程
- 程序验证
- 环境和工具
- 工作站
- 购买和自行开发
- 需求精炼和快速原型
- 增量开发——增长，而非搭建系统
- 卓越的设计人员

Ada和其他高级编程语言

- 这类语言出现最大的回报来自出现时的冲击，它通过使用更加抽象的语句来开发，降低了机器的次要复杂度。

面向对象编程

- 它们的出现都消除了开发过程中的非本质困难，允许设计人员表达自己设计的内在特性，而不需要表达大量句法上的内容。
- 对于抽象数据类型和层次化类型，它们都是解决了高级别的次要困难和允许采用较高层次的表现形式来表达设计。
- 使得修改局部化，提高了可维护性

人工智能

- 现在有两种差异非常大的AI定义被广泛使用。AI-1：使用计算机来解决以前只能通过人类智慧解决的问题。AI-2：使用启发式和基于规则的特定编程技术。在这种方法中，对人类专家进行研究，判断他们解决方法的启发性思维或者经验法则……。程序被设计成以人类解决问题的方式来运作。

专家系统

- 专家系统是包含归纳推论引擎和规则基础的程序，它接收输入数据和假设条件，通过从基础规则推导逻辑结果，提出结论和建议，向用户展示前因后果，并解释最终的结果。推论引擎除了处理推理逻辑以外，通常还包括复杂逻辑或者概率数据和规则。
- 专家系统最强有力的贡献是给缺乏经验的开发人员提供服务，用最优秀开发者的经验和知识积累为他们提供了指导。
- 现在专家系统的研究已经很少。

“自动”编程

- 近四十年中，人们一直在预言和编写有关“自动编程”的文字，从问题的一段陈述说明自动产生解决问题的程序。
- 大多数情况下所给出的技术说明本质上是问题的解决方法，而不是问题自身。

图形化编程

- 流程图是一种非常差劲软件结构表达方法。
- 现在的屏幕非常小，像素级别，无法同时表现软件图形的所有正式、详细的范围和细节。
- 软件非常难以可视化。

程序验证

- 是否有可能出现银弹，能够在系统设计级别、源代码级别消除bug呢？是否可以在大量工作被投入到实现和测试之前，通过采用证实设计正确性的“深奥”策略，彻底提高软件的生产率和产品的可靠性?

- 不能保证节约劳动力
- 程序验证不意味着零缺陷的程序
- 完美的程序验证只能建立满足技术说明的程序，而这时，软件工作中最困难的部分已经接近完成，形成了完整和一致的说明。

环境和工具

- IDE
- 这样的工作是非常有价值的，它能带来软件生产率和可靠性上的一些提高。但是，由于它自身的特性，目前它的回报很有限。

工作站

- 硬件速度的加快。
- 编译速度，开发速度。
- 1986!

购买和自行开发

- 构建软件最可能的彻底解决方案是不开发任何软件。
- 通用软件。
- 1986!

需求精练和快速原型

- 概念性工作中，没有其他任何一个部分比确定详细的技术需求更加困难。
- 事实上，客户不知道自己需要什么。
- 快速原型明确实际的概念结构，让用户知道他们需要什么。

增量开发

- Grow not building
- 客户
- 士气
- 迭代式开发

卓越的设计人员

- 软件开发是一个创造性的过程。

没有银弹的影响

- 软件开发本质的认识
- 软件过程