Atitit sft arch dev history and pric原则软件架构发展历史

# 快速开发方法 接口 架构等 api

与os 对其（文件操作 数据库 task redis内存

全局变量 函数即可

Oo少量

## 并没有独立的“架构”可言，软件架构与硬件架构是直接物理对

**历史上的软件**危机和契机

计算机刚诞生的年代，硬件规模还很小，甚至程序员仅凭大脑就足够记住数据在几 KB 内存中的布局情况，理解每条指令在电路中的运行逻辑。此时的计算机尽管运算速度比人类快，但其内部却没有什么人所不知道的事情；此时的软件开发并没有独立的“架构”可言，软件架构与硬件架构是直接物理对齐的。

第一次软件危机，同时也是[结构化编程](https://en.wikipedia.org/wiki/Structured_programming" \o "" \t "https://www.infoq.cn/article/_blank)发展的契机，结构化编程扭转了当时直接面向全局数据、满屏 Goto 语句书写[面条式代码](https://en.wikipedia.org/wiki/Spaghetti_code" \o "" \t "https://www.infoq.cn/article/_blank)（Spaghetti Code）的编程风气，强调可独立编写、可重复利用的子过程/局部块的重要性，让软件的每个局部都能够设计专门的算法和数据结构，允许每一位程序员只关注自己所负责的部分，从而在整体上控制住了软件开发的复杂度。此时，软件的架构才开始独立于硬件物理架构而存在，软件业开始出现把控全局设计的架构师与聚焦局部实现的程序员的角色分工。

渡过第二次软件危机的过程中，面向对象编程逐步取代了面向过程的结构化编程，成为主流的程序设计思想。这次思想转变宣告“追求最符合人类思维的视角来抽象问题”取代了“追求最符合机器运行特征的算法与数据结构”成为软件架构的最高优先级，并一直持续沿用至今。这次危机还直接导致软件工程的诞生，如何以系统性的、规范化的、可定量的方法去高质量地开发和维护软件成为一门独立的科学。