软件分析与架构设计

课程介绍
《软件分析》+《软件架构设计》

https://plc-cqu.github.io/teaching/

何冬杰

重庆大学

软件开发历史

1970’
- 汇编语言
- 适用于小型程序

1980’
- 高级语言
- 面向过程理论
- 数据流/控制流设计方法
- 应用程序开发库 （类库/函数库）

1995
- 面向对象理论
- 面向对象建模以及设计技术
- 应用程序开发框架: J2EE, .NET
- 组件技术: COM/DCOM, CORBA …

Future
- 对象建模/设计标准化: UML
- 模型驱动开发: MDA
- …

软件演化趋势

- **软件的规模和复杂度越来越高**

  应用领域: 科学计算、工业生产、商业、教育、娱乐

- **软件的抽象程度越来越高**

  机器语言 - >汇编语言 - >高级语言 - > 开发框架面向过程编程 - > 面向对象编程 - > 面向切面编程

为什么要做软件分析?

❑ 软件缺陷可能导致灾难性事故

2019年波音737Max坠机事件：埃塞俄比亚航空一架波音737 MAX 8型飞机在起飞阶段坠毁，机上人员全数遇难。

事故原因：飞机MCAS防失速自动系统软件存在缺陷

2016年特斯拉车祸：自动驾驶模式下的特斯拉 汽车和卡车相撞，导致 驾驶员当场丧生

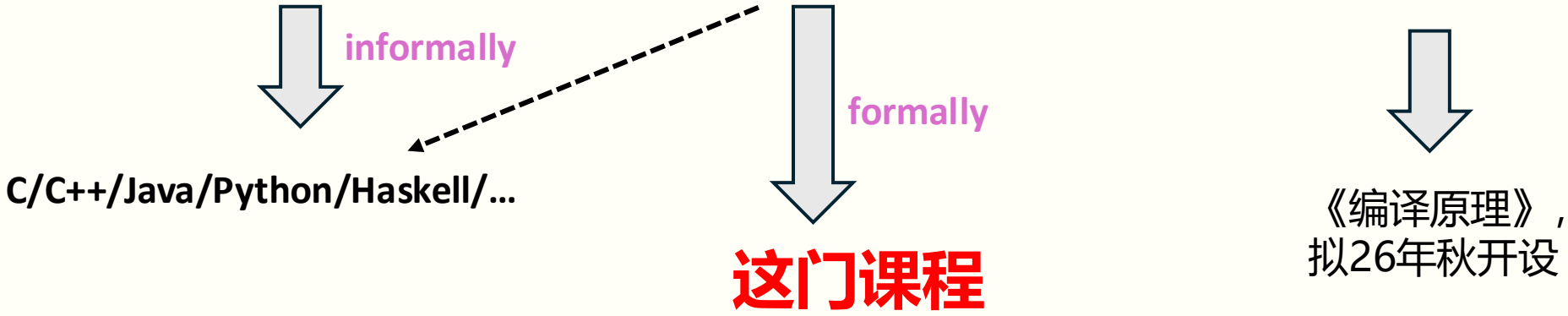
事故原因：在强烈日光条件下，摄像头进入盲区，但软件系统并没有捕获这一情况

2011年亚马逊宕机事故：亚马逊云计算出现了超过2天的宕机事故，造 成的资金和信誉损失难以估算

事故原因：软件配置错误导致部分结点请求激增，不断转发请求压垮网络

《软件分析》分析什么?

PL = Syntax + Semantics + Implementation



- PL (Programming Language)：计算机学科的基础与核心技术
- 学习C等编程语言时已非正式的学过语法和语义
- 编译器是对PL的实现、是关键基础软件
  - 拟26年秋季开始为学院本科生开设
- 本门课将形式化地学习软件语义、并对语义进行分析
  - 分析软件具有什么性质?

许多关于软件性质的问题

- ❑ Does the program terminate on all inputs?
- ❑ How large can the heap become during execution?
- ❑ Can sensitive information leak to non-trusted users?
- ❑ Can non-trusted users affect sensitive information?
- ❑ Can *two program variables* point to the same memory location?
- ❑ Is this piece of code dead (so that it could be eliminated)?
- ❑ Are buffer-overruns possible?
- ❑ Data races?
- ❑ SQL injections?
- ❑ XSS?
- ❑ …

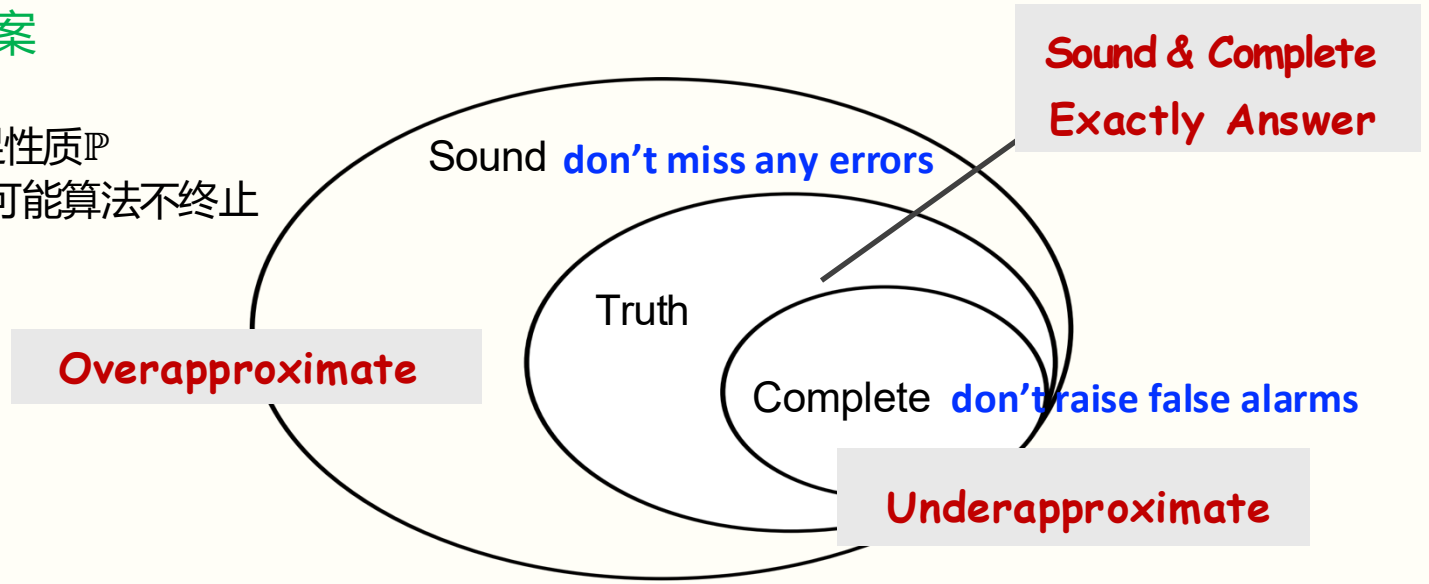
一般性问题：给定程序P，请问P是否满足性质$\mathbb{P}$?

如何确定程序P是否满足性质ℙ?

❑动态语义分析：构造输入，动态运行程序P，根据具体语义进行确定
- ➢具体语义（concrete semantics）
- ➢**问题：程序P可能有无限多的输入和无限多的执行路径**
- ➢ "Testing shows the presence, not the absence of bugs."-- Edsger W. Dijkstra

❑静态语义分析：设计算法，不运行实际程序P，能给出问题的答案
- ➢根据**莱斯定理(Rice's Theorem)**，对于非平凡问题都无法给出精确答案
- ➢可以利用抽象和近似，给出问题的答案
  - **Must**：P不满足性质ℙ；P满足性质ℙ
  - **May**：P可能满足性质ℙ； P可能不满足性质ℙ
  - **Unknown**：不知道P是否满足性质ℙ；可能算法不终止
- ➢抽象语义（abstract semantics）

❑动静结合的语义分析
- ➢例如：concolic execution, …



（optional）莱斯定理(Rice's Theorem)

Any non-trivial property of the behaviour of programs in a Turing-complete language is undecidable!

- **平凡性质**：要么对全体程序都为真，要么对全体程序都为假
- **非平凡（non-trivial）性质**： 不是平凡的所有性质
- **可判定问题（Decidable Problem）**：
  - 存在一个算法，使得对于该问题的每一个实例都能给出是/否的答案
- **停机问题（ the halting problem ）**：判断一个程序在给定输入上是否会终止
  - 图灵1936年证明：不存在一个算法能回答停机问题，即停机问题不可判定
  - 反证法证明：
    - 假设存在停机算法 bool halt(Program p), 构造邪恶程序 void evil() {if (halt(evil) while(1); else return; }
    - 若halt(evil)为true，即evil()能停机，则evil函数进入while(1)无法停机；
    - 若halt(evil)为false, 即evil()不能停机，则evil函数进入else分支通过return终止；出现矛盾
- **莱斯定理证明**：将停机问题归约到该判定问题

（optional）莱斯定理(Rice's Theorem)证明

❑**反证法证明：不存在算法可判定程序P是否满足非平凡性质$\mathbb{P}$。**

根据定义，存在程序f_1满足非平凡性质$\mathbb{P}$，f_2不满足性质$\mathbb{P}$

$\phi_f(x)$: 表示图灵机用输入x运行编号为f的程序

构造编号为$\langle f, x\rangle$的邪恶程序:

$$\phi_{\langle f,x\rangle}(y) = \begin{cases} f_1(y), & \text{如果}\phi_f(x)\text{能停机} \\ \text{未定义}, & \text{否则} \end{cases}$$

```
void evil(f, x, y) {
  f(x);
  f1(y);
}
```

- 存在停机判定算法则存在非平凡性质判定算法
  - 若$\phi_f(x)$能停机，则邪恶程序和f_1一致，满足性质$\mathbb{P}$
  - 若$\phi_f(x)$不能停机，则邪恶程序行为未定义，可假定不满足性质$\mathbb{P}$

- 存在非平凡性质判定算法则存在停机判定算法
  - 若邪恶程序满足性质$\mathbb{P}$，这性质来自f1，因此可断定$\phi_f(x)$能停机
  - 反之，可断定$\phi_f(x)$不能停机

软件分析相关课程内容

- **操作语义（Operational Semantics）：**
  - How would I execute this?
  - 大步语义、小步语义
- **踪迹语义（Trace Semantics)**
  - What is the program state at each step in a path?
  - 符号执行（ symbolic execution）、混合符号执行（concolic execution）
  - 模型检测（LTL、CTL）
- **收集语义（Collecting Semantics)**
  - How can the program possibly run?
  - 经典数据流分析
  - 过程间分析、上下文敏感分析、流敏感分析
  - 控制流分析
  - 约束系统及指针分析
- **指称语义（Denotational Semantics）：**
  - What function is this trying to compute?
  - IFDS/IDE、CFL-可达性问题
- **公理语义（Axiomatic Semantics)**
  - What is true after I execute this?
  - 霍尔逻辑（ Hoare Logic ）、交互式定理证明
- **抽象解释（Abstract Interpretation）理论**
- **可满足性模理论（SMT/SAT)**
- **软件分析应用（Optional)**

该部分课程内容相对晦涩，会根据时间挑选部分讲解

如何学习《软件分析》？

- **预备知识**
  - 熟悉常见的数学符号
- **争取能够看懂课件**
  - 课上认真听讲
  - 课下复习至少一遍课件
- **关于课程难度**
  - 软件分析内容总体偏难，我会尽量用浅显的方式介绍
  - 不会为了降低难度删除困难的内容
    - 困难的内容可能会在某些时刻用到
  - 不要求同学们掌握全部课程内容
    - 会体现在平时评价和期末评价上

必要时可以提前预习，课件尽量提前一周发布出来！

学习《软件分析》的好处

- **可以加深对PL的理解**
  - 有助于深入理解编程语言的语法和语义
  - 有助于写出更可靠、 安全、 高效的程序
- **潜在的大公司核心部门的就业**
  - “谷歌最强的人都在开发内部工具。”
  - “国内白盒工具市场主要用的国外产品。”
  - 国内企业到了开发工具阶段，但人才缺乏
  - **开发工具部门或企业研究院**
- **科学研究**
  - 从事软件相关研究的必要条件
  - 编译器、软件工程、安全分析
  - 。。。
- **论文写作**
  - 毕业论文、会议或期刊论文
  - 有助于提升论文写作中的形式化水平

“完善新型举国体制，采取超常规措施，全链条推动集成电路、工业母机、高端仪器、基础软件、先进材料、生物制造等重点领域关键核心技术攻关取得决定性突破。”——十五五规划建议

“国际形势变化下， IT企业对基础软件的重视达到了空前的高度，正在抢夺软件分析人才”

ORACLE Microsoft IBM Research

Google Uber Meta

SAMSUNG 蚂蚁集团 ANT GROUP

Alibaba Group ByteDance HUAWEI

。。。。。。

为什么要做架构设计?

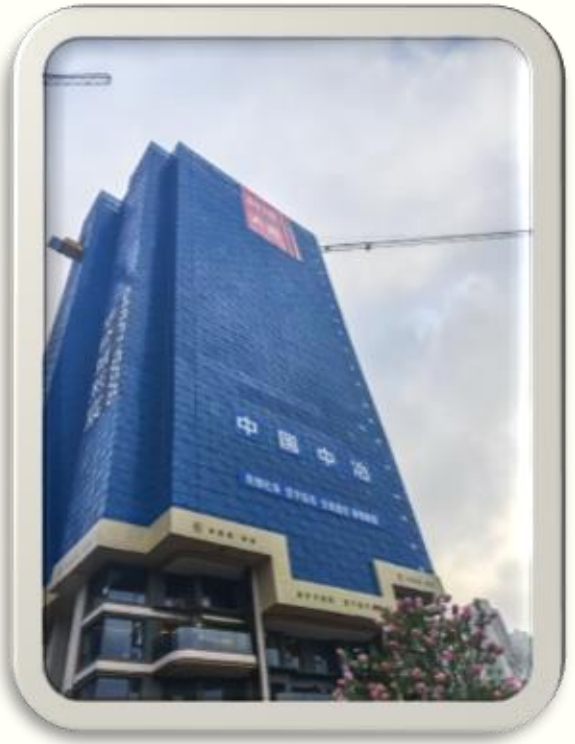
- 规模
- 过程
- 工作计划
- 风险
- 开发团队实力
- 成本
- 所需技术
- 利益相关者

一人就可完成构建

- 无需建模
- 过程简单
- 工具简单

团队设计更为高效

- 需要建模
- 健全的过程
- 有力的工具



更加专业的设计团队

- 安全性验证
- 成本风险评估等

软件架构设计

定义

Software architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution ----- IEEE 1471-2000

- **软件元素**：功能、接口、程序、类模块、层、子系统、客户端/服务器等
- **可见属性**： 提供服务、性能特征、错误处理、公享资源使用等
- **关系**：这些元素之间的组合机制

❑**软件架构是商业和技术决策的结果**

- 好的架构设计对于软件系统成败至关重要
- 架构设计比数据结构和程序算法更为重要

该部分课程内容相对简单，学习起来会轻松愉快许多☺

考核与评分

- **更加重视学习过程**
  - 过程评价 （80%）； 期末评价（20%）； 细则待确定
- **评价方式：**
  - 考勤（30%）： 出勤率低于30%时会抽查
  - 小作业（30%）： 同学们可以随意选择5道布置的作业去写和提交即可
    - 不在意答案和过程是否雷同，主要是为了帮助大家对课程内容的理解
  - 项目（20%）： 会出3个小项目，随意选择1个去完成即可，独立完成
    - 抄袭会很严重
  - 期末考试（20%）：(可商量)
    - 开卷考、不允许联网和交头接耳交流
    - 考前会划重点
    - 会出8道题，学生只需要选答5题
    - 考试首要目的是帮助同学们温故知新
- **主观映像分（10%）：**
  - 根据讲师和助教对学生的主观映像进行打分
  - 只用于给最终打分低于70分但学习态度认真的同学进行鼓励性的适当调分
  - 调分会公示