《开源软件开发与社区治理》

第二讲 软件工程视角

主讲人：王 伟

viewpoints

DOI:10.1145/1787234.1787249 Thomas J. Misa, Editor

Interview

An Interview with Edsger W. Dijkstra

The computer science luminary, in one of his last interviews before his death in 2002, reflects on a programmer's life.

THE CHARLES BABBAGE INSTITUTE holds one of the world's largest collections of research-grade oral history interviews relating to the history of computers, software, and networking. Most of the 350 interviews have been conducted in the context of specific research projects, which facilitate the interviewer's extensive preparation and often suggest specific lines of questions. Transcripts from these oral histories are a key source in understanding the history of computing, since traditional historical sources are frequently incomplete. This interview with programming pioneer Edsger Dijkstra (1930–2002) was conducted by CBI researcher Phil Frana at Dijkstra's home in Austin, TX, in August 2001 for a NSF-KDI project on "Building a Future for Software History."

Winner of ACM's A.M. Turing Award in 1972, Dijkstra is well known for his contributions to computer science as well as his colorful assessments of the field. His contributions to this magazine continue to enrich new generations of computing scientists and practitioners.

We present this interview posthumously on the eighth anniversary of Dijkstra's death at age 72 in August 2002; this interview has been condensed from the complete transcript, available at http://www.cbi.umn.edu/oh.

—Thomas J. Misa

PHOTOGRAPH COURTESY OF THE UNIVERSITY OF TEXAS AT AUSTIN

Edsger Dijkstra（1972年图灵奖得主）

“我们所使用的工具影响着我们的**思维方式**和**思维习惯**，从而也深刻地影响着我们的**思维能力**。”

软件危机

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly:

- *as long as there were no machines, programming was no problem at all;*
- *when we had a few weak computers, programming became a mild problem,*
- *and now we have gigantic computers, programming has become an equally gigantic problem.*

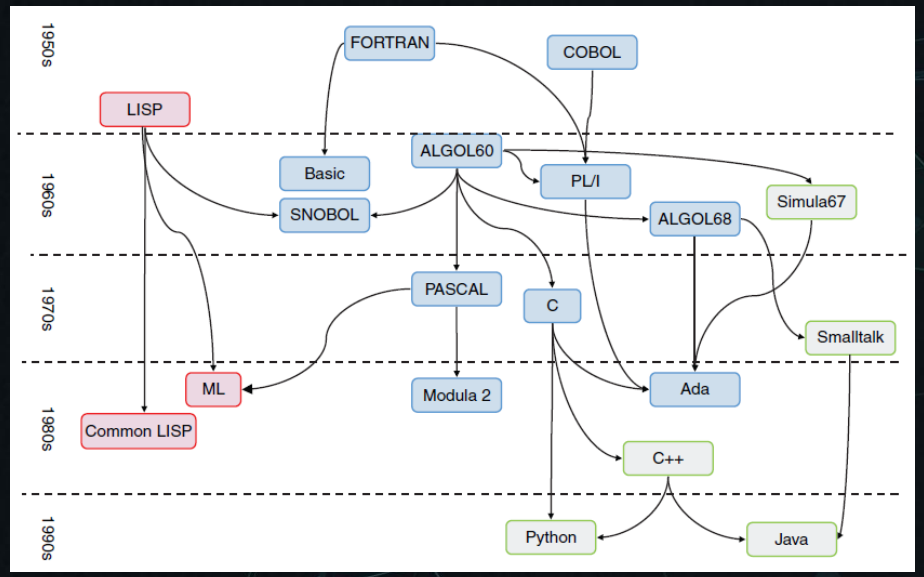
— Edsger Dijkstra

软件危机 1.0（从大型机到个人电脑）

- 软件开发技术难以满大型软件系统的开发需求：
  1. 大多数大型软件开发项目的成本都超过预算，开发进度一拖再拖；
  2. 软件产品质量不高，大型软件系统存在 bug 几乎成为不可避免的问题；
  3. 软件产品难以维护；
  4. 软件产品的开发成本过高；
  5. 软件产品开发的效率跟不上计算机硬件的发展以及用户需求的增长。

解决软件危机的办法

- The evolution of ***programming languages***
  - *Structured programming*
  - *object-oriented programming*
- The emergence of ***software engineering methodologies***



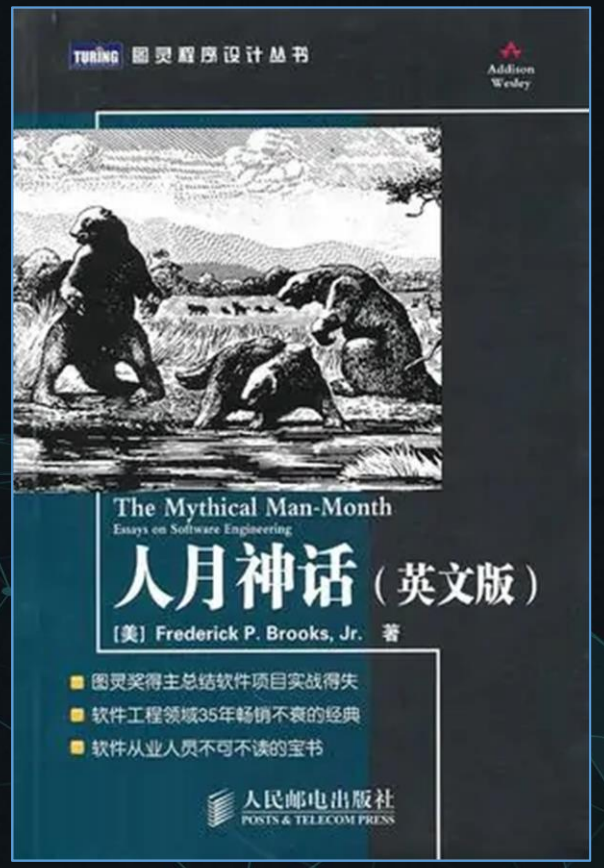
Fred Brooks made a major contribution to the design of IBM/360 computers. His famous book *The Mythical Man-Month* describes his experiences in software development.

开发语言与人月神话

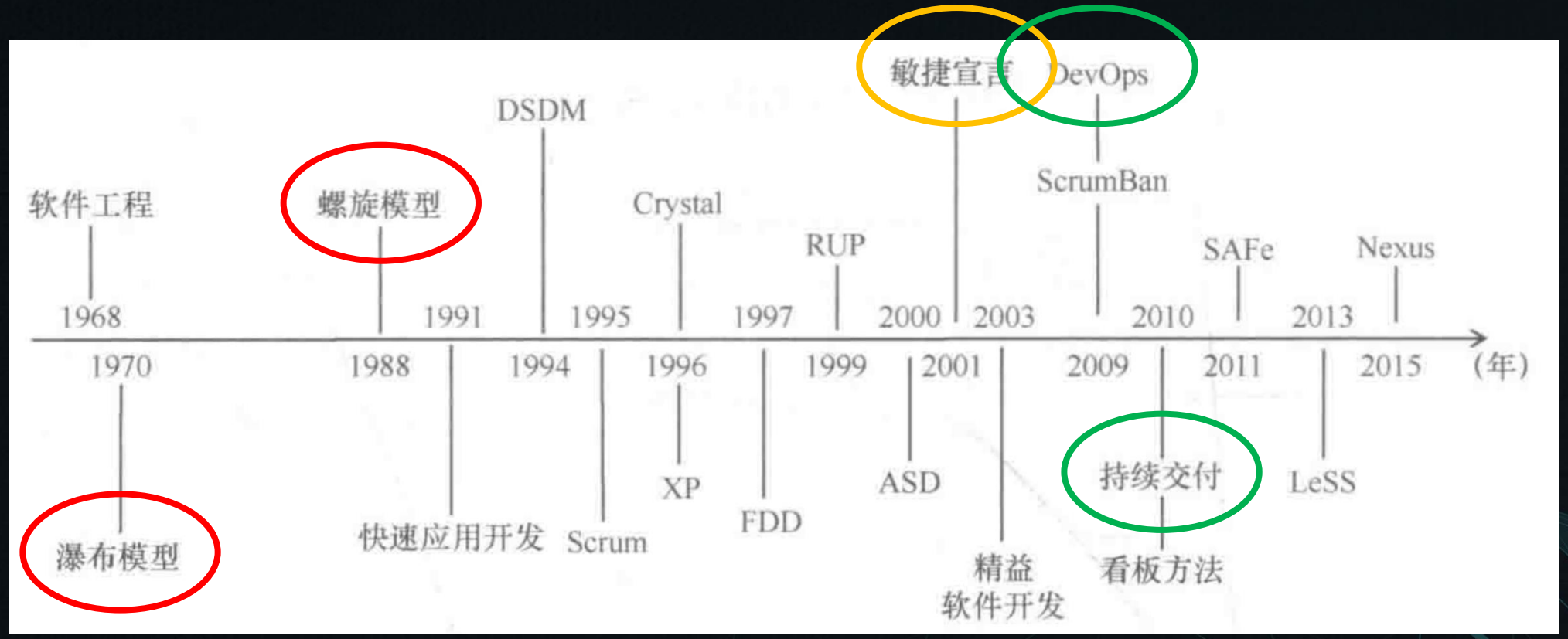
#	全域开发者语言榜	开发者账号数
1	JavaScript	305,814
2	Python	175,610
3	HTML	159,303
4	Java	139,673
5	Ruby	87,780
6	TypeScript	85,116
7	C#	54,343
8	PHP	52,915
9	C++	47,799
10	CSS	46,528

#	Top 10W 万开发者语言榜	开发者账号数
1	JavaScript	15,858
2	Python	10,866
3	TypeScript	7,419
4	Java	6,665
5	Go	5,094
6	C++	4,204
7	Ruby	3,802
8	HTML	3,490
9	PHP	3,000
10	C#	2,892

来源：GitHub 2020 数字洞察报告, 2021-02.



软件工程的发展历史



软件工程 1.0

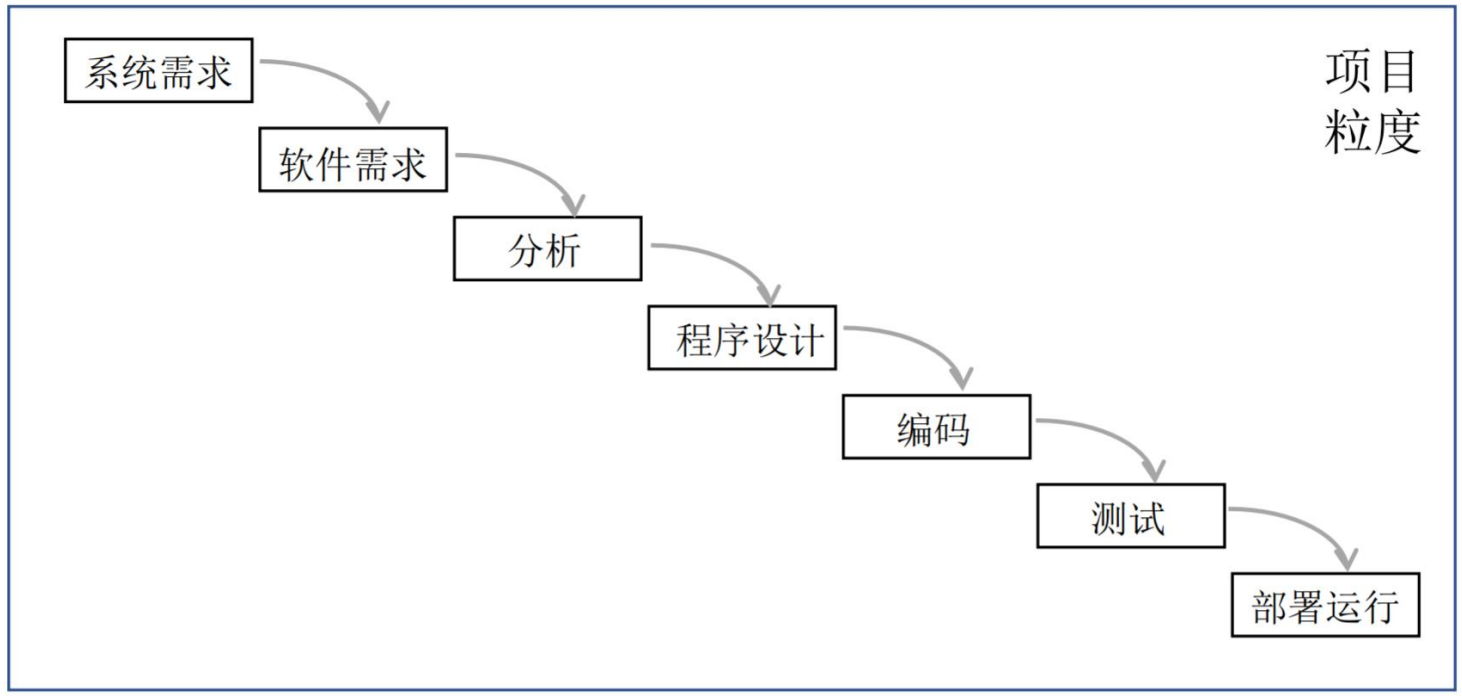
软件工程 2.0

软件工程 3.0

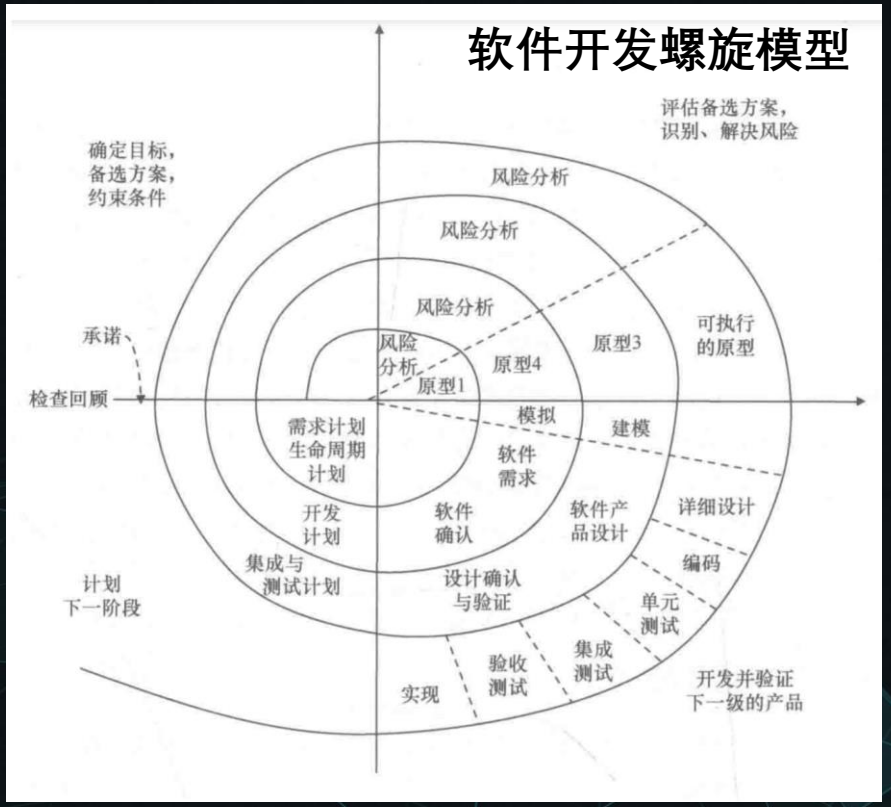
来源：乔梁, 持续交付2.0：业务引领的DevOps精要, 人民邮电出版社, 2019.

软件工程 1.0 —— 20世纪70年代

瀑布模型

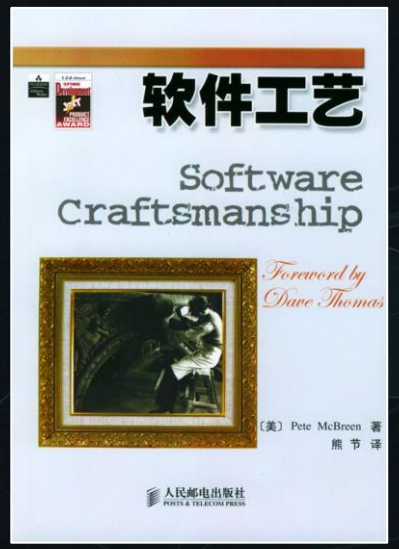


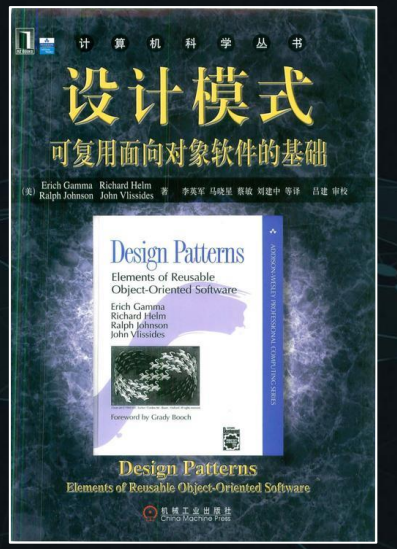
软件开发螺旋模型

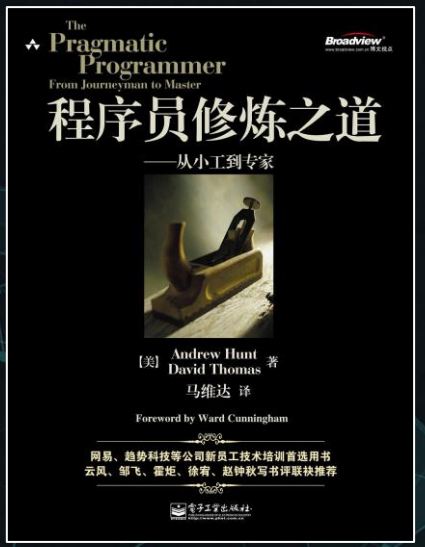


软件工程 1.0 的主要技术成果

- 高级编程语言的出现
  - 面向对象，C++（83），Java（95）
- 构件技术
  - 设计模式（94）、软件架构
- 软件工艺
  - 软件蓝领的出现
- 软件工程最佳实践与标准
  - CMM（98）
  - ITIL / ISO20000等标准

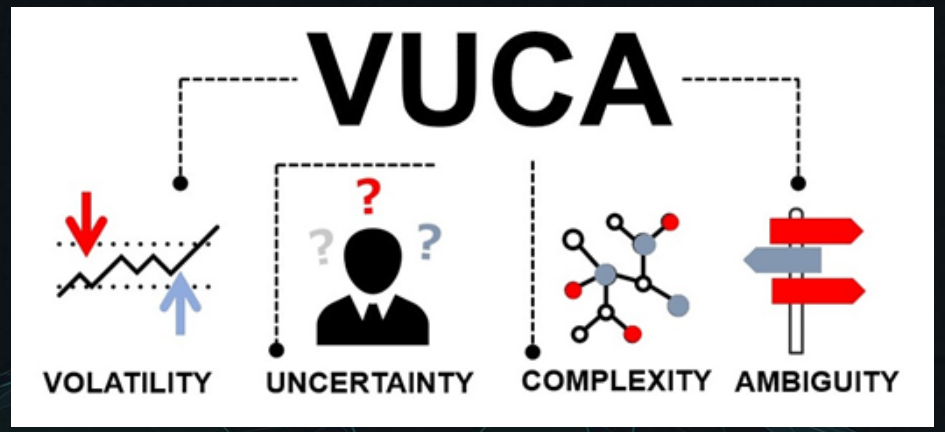






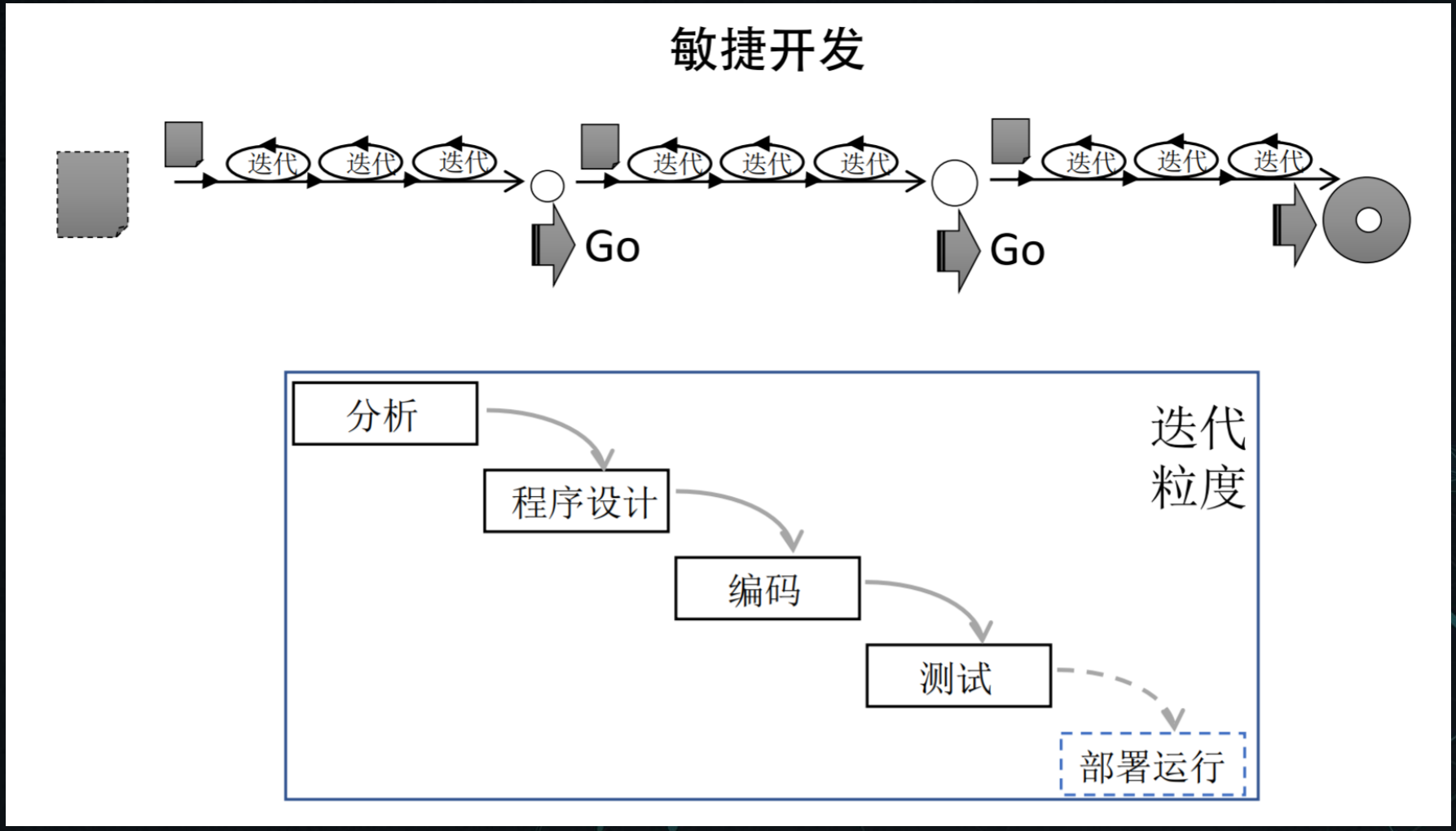
软件危机 2.0（从互联网到移动互联网）

- 世界朝向 VUCA 的方向进行着变化
  1. Volatility：易变性
  2. Uncertainty：不确定性
  3. Complexity：复杂性
  4. Ambiguity：模糊性

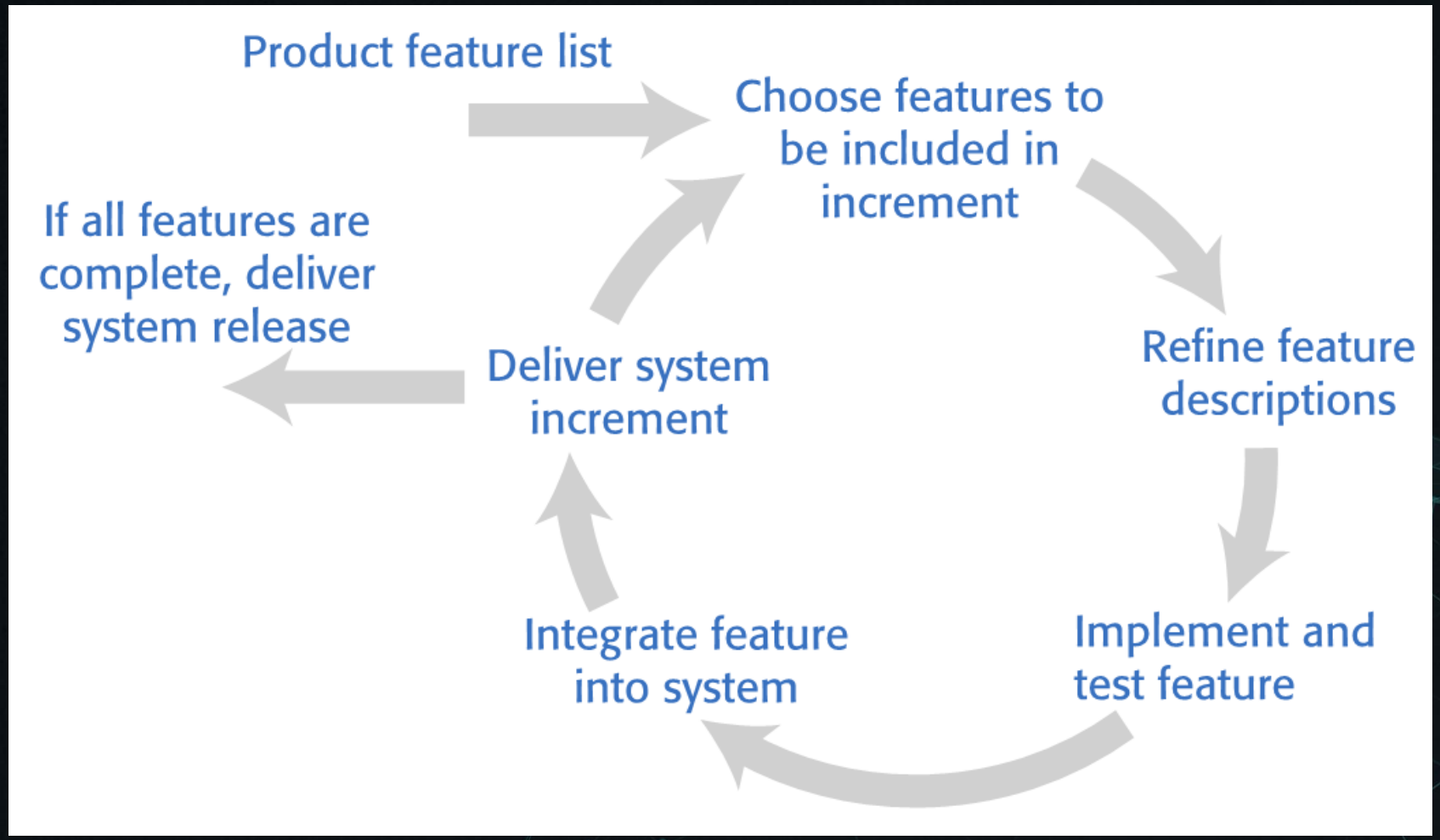


VUCA 这个术语源于军事用语，用来描述冷战结束后的越发不稳定的、不确定的、复杂、模棱两可和多变的世界。

软件工程 2.0 —— 20世纪90年代



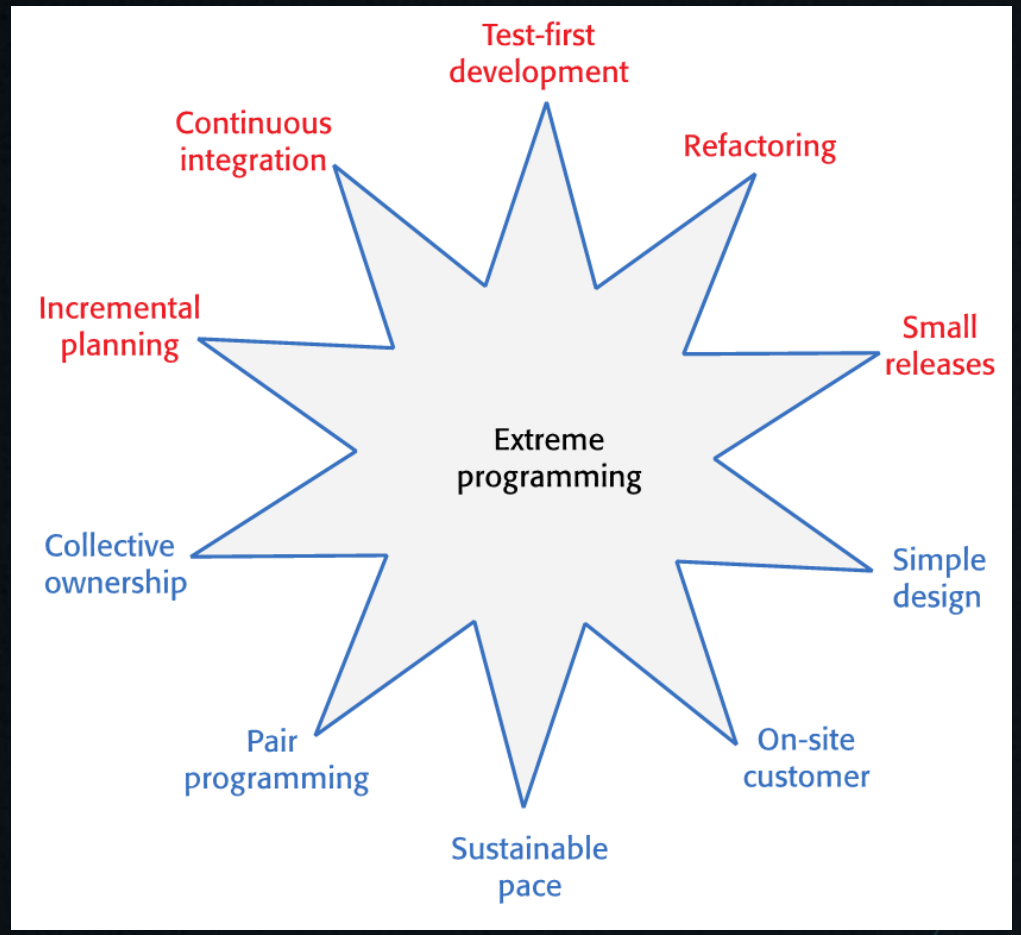
Incremental development

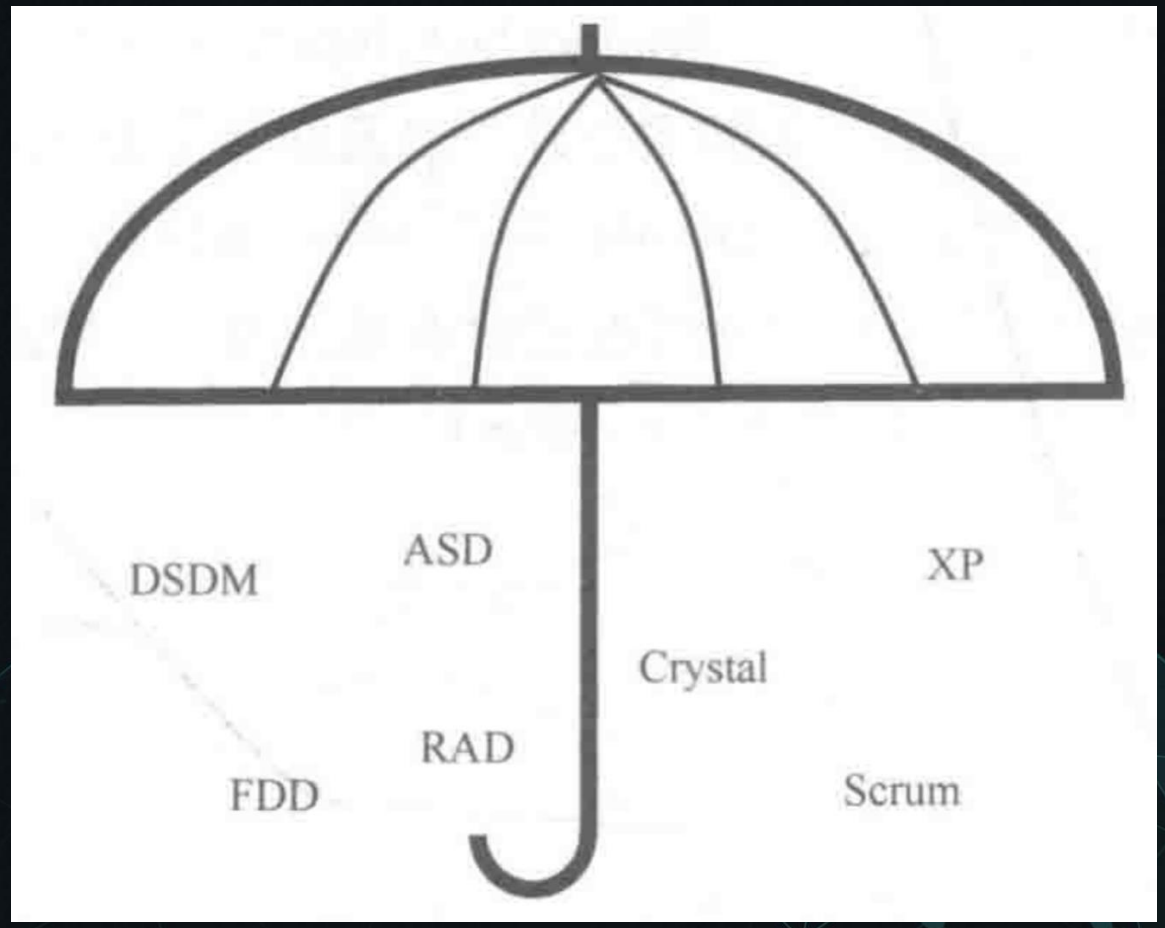


基于快速迭代的软件交付

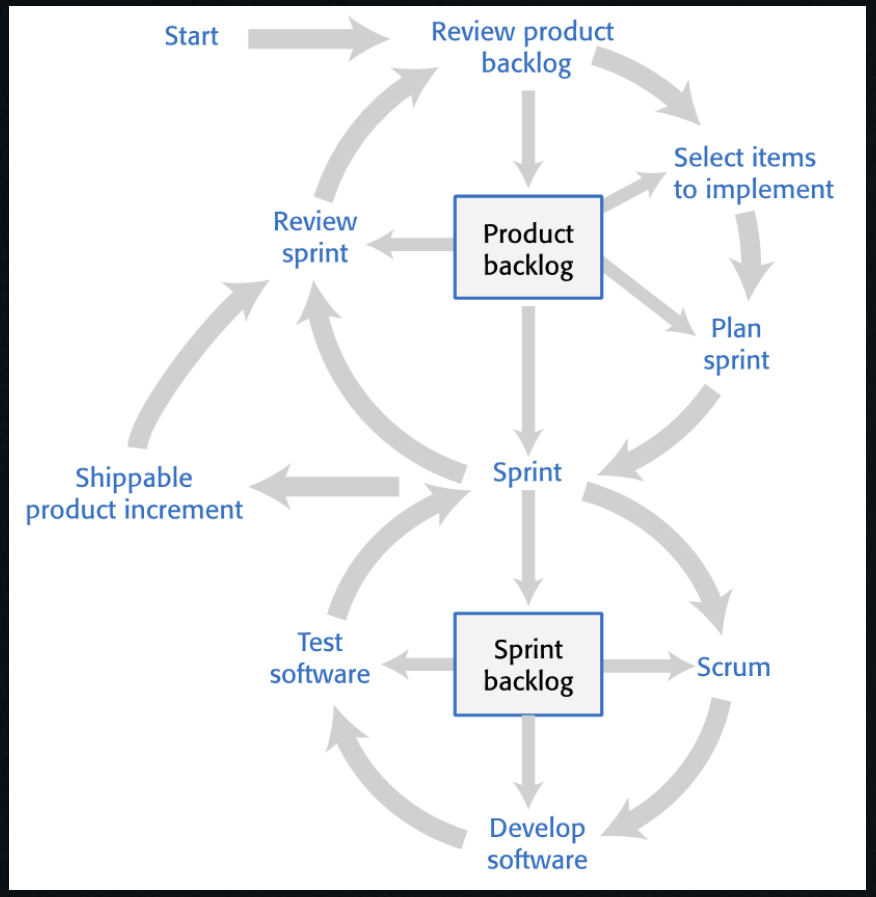
- **需求管理角度**：以用户故事的形式分析和记录需求，使软件需求以一种允许乃至鼓励多次迭代交付的形式出现在软件团队面前；
- **项目管理角度**：围绕用户故事开展的迭代管理方法，包括计划会议、成果展示、每日站会等，加上对进度与质量的度量和可视化呈现，使项目随时处于透明、受控的状态；
- **配置管理角度**：以持续集成为核心，对修改动作的频度和方式做出了严格要求，是软件在迭代演化的过程中始终保持可用状态；
- **质量管理角度**：测试驱动开发使软件获得很高的自动化测试覆盖率，在自动化测试的保护下，通过频繁而小步的重构改善软件内部质量，从而消减软件的频繁改动带来的质量风险。

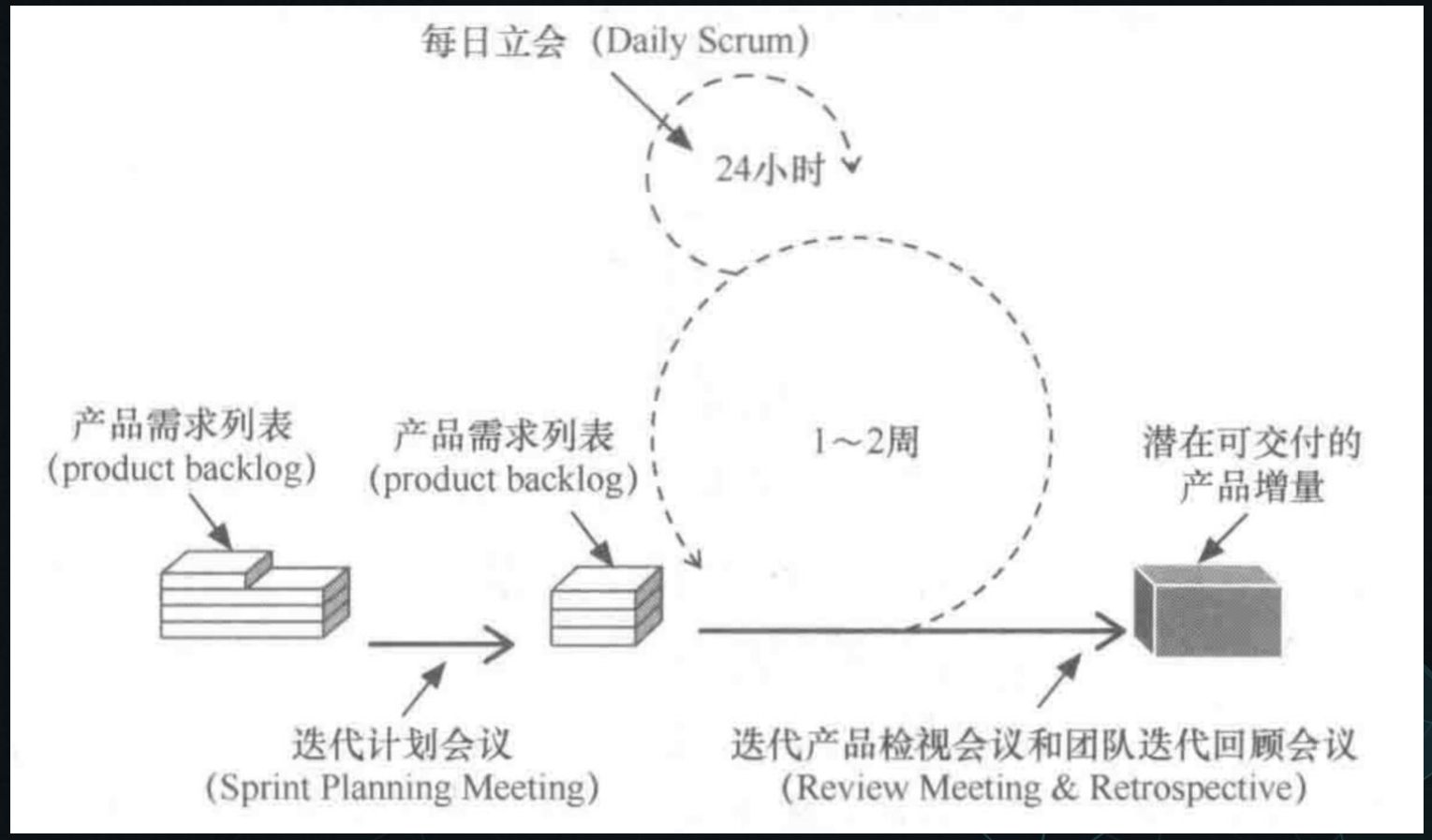
极限编程与敏捷开发


Test-first development
Refactoring
Small releases
Simple design
On-site customer
Sustainable pace
Pair programming
Collective ownership
Incremental planning
Continuous integration
Extreme programming

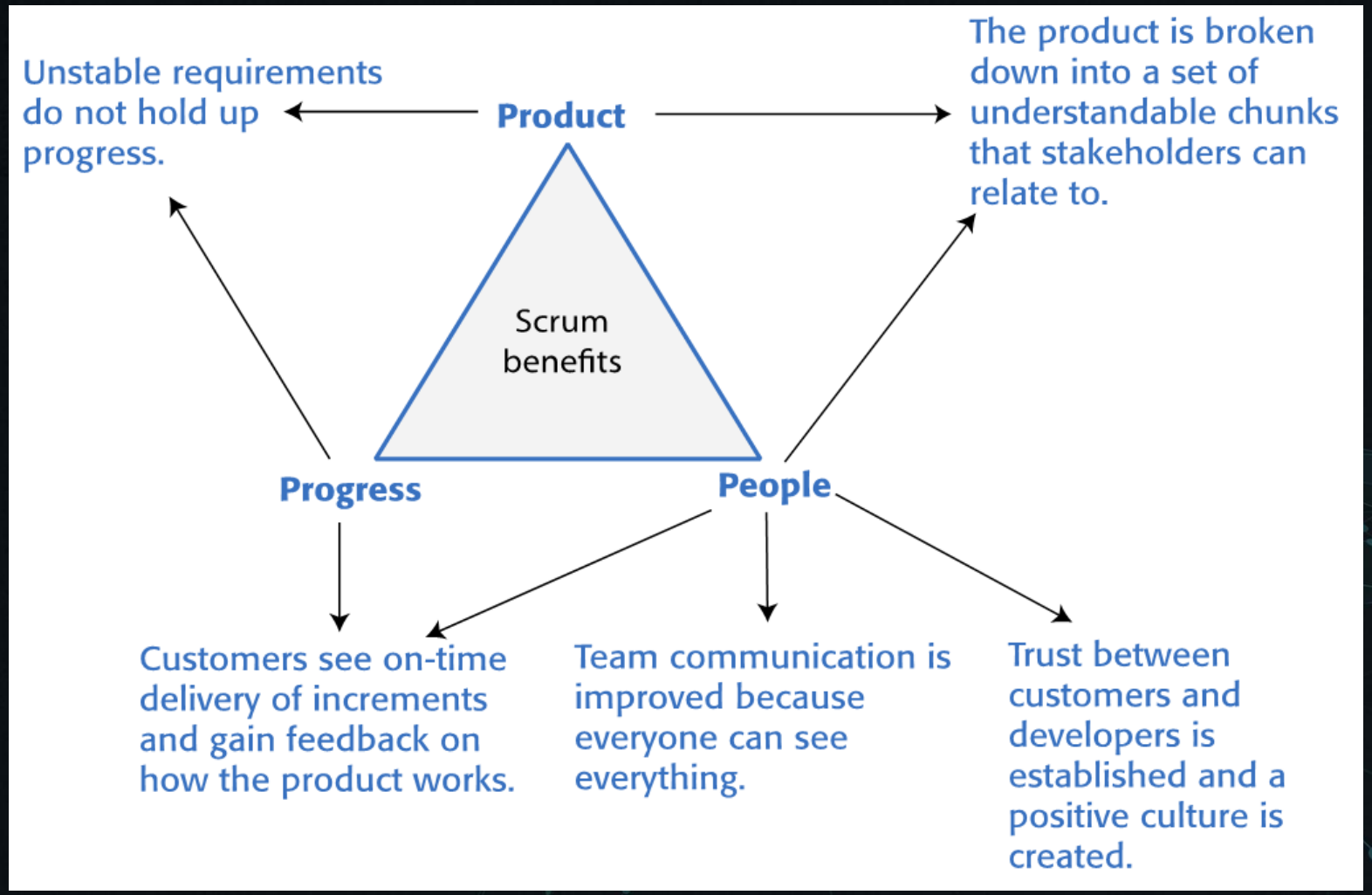

DSDM
ASD
XP
Crystal
RAD
FDD
Scrum

Scrum框架

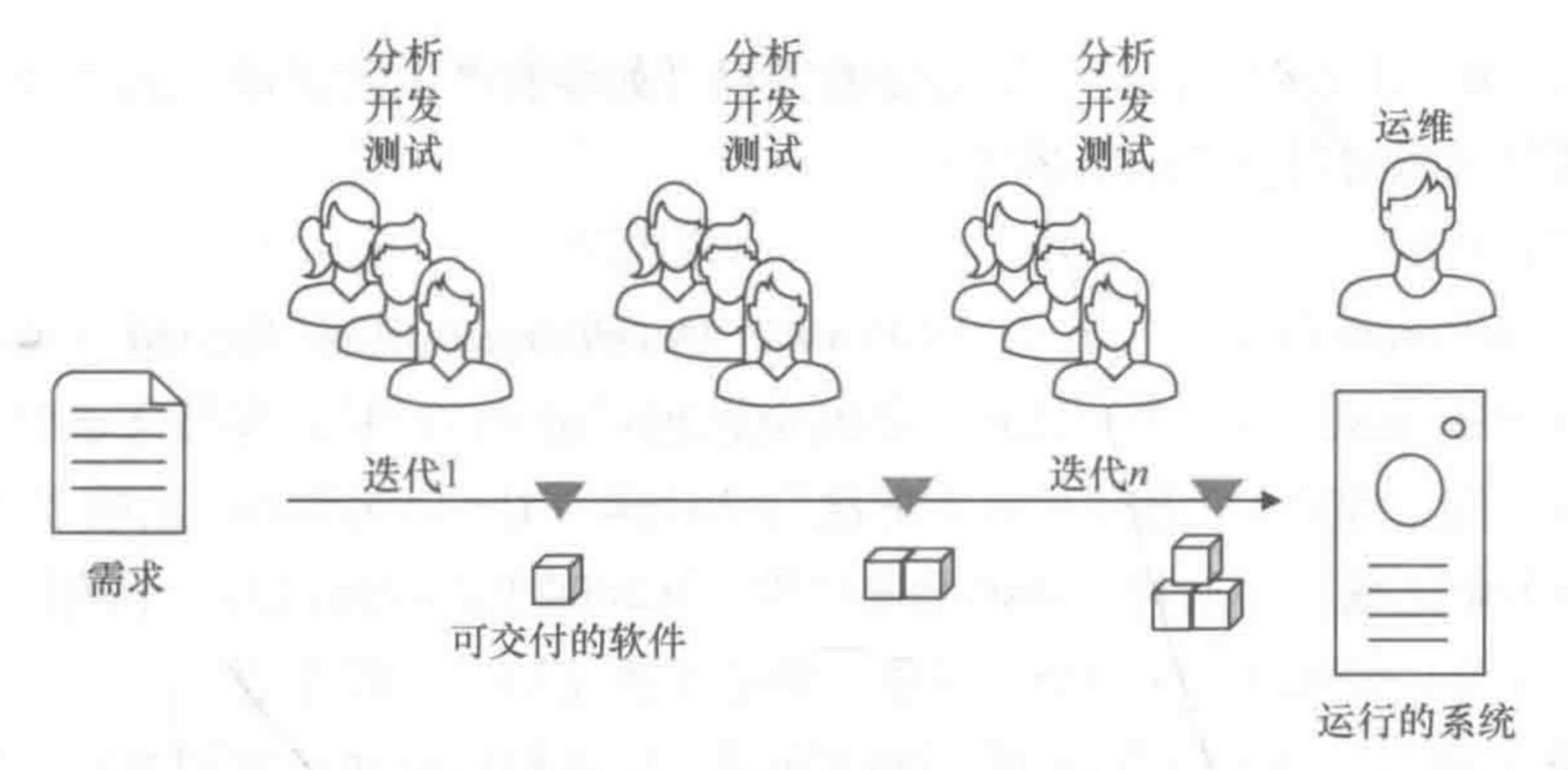

Start
Review product backlog
Select items to implement
Review sprint
Product backlog
Plan sprint
Shippable product increment
Sprint
Test software
Sprint backlog
Scrum
Develop software


每日立会（Daily Scrum）
24小时
产品需求列表
(product backlog)
产品需求列表
(product backlog)
1～2周
潜在可交付的
产品增量
迭代计划会议
(Sprint Planning Meeting)
迭代产品检视会议和团队迭代回顾会议
(Review Meeting & Retrospective)

使用 Scrum 的五大好处



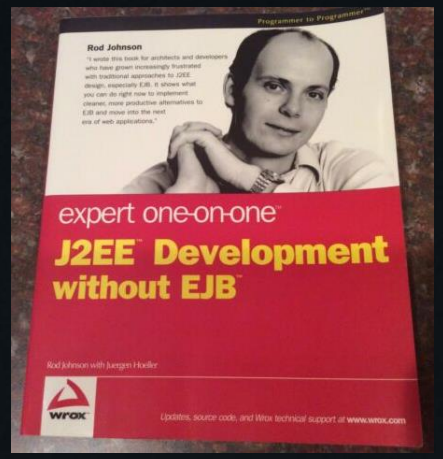
敏捷软件开发小结

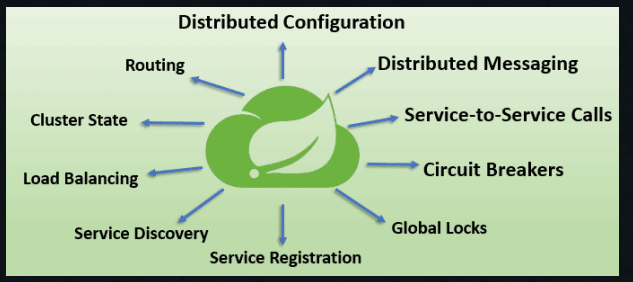


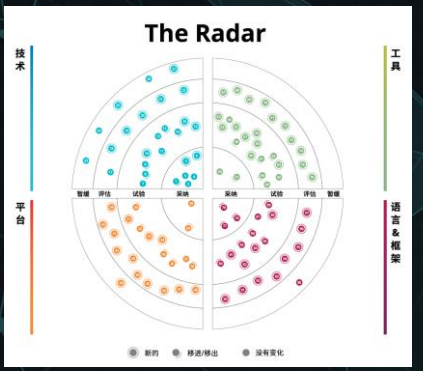
强调短周期迭代与跨部门协作的敏捷

软件工程 2.0 —— 敏捷实践

- Java时代的敏捷实践
  - J2EE → J2EE without EJB → Spring framework
- ThoughtWorks
  - 1993年创办于美国，2005年进入中国
- “敏捷中国”开发者大会（06年）
- AMM（敏捷成熟度模型）
  - 6个评价维度：需求、测试、代码、集体所有、协作、保障与治理、简单性
- 大厂的敏捷转型
  - 华为：2007年开始试点，敏捷IPD流程
  - 腾讯：2006年开始，TAPD敏捷产品
  - 阿里： 2006年开始，钉钉、云效等产品

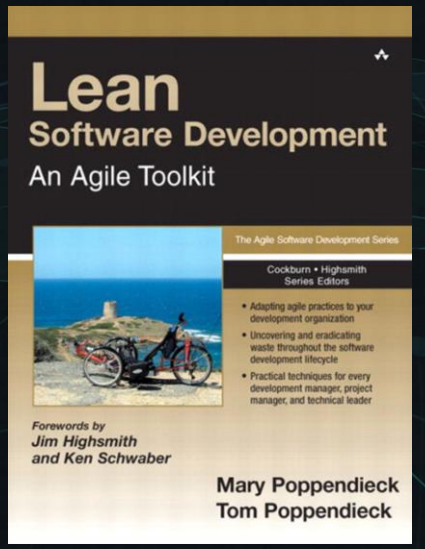


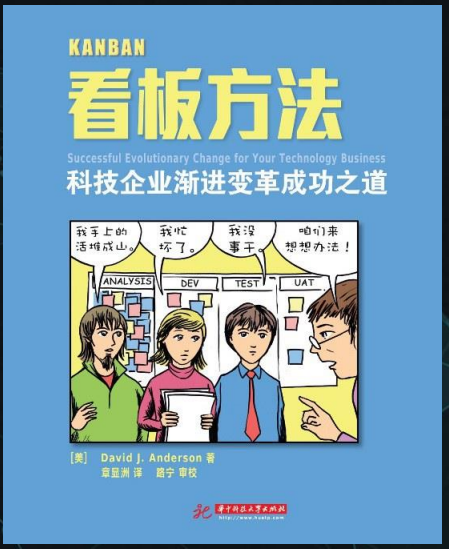




敏捷开发 2.0 —— 敏捷拓展

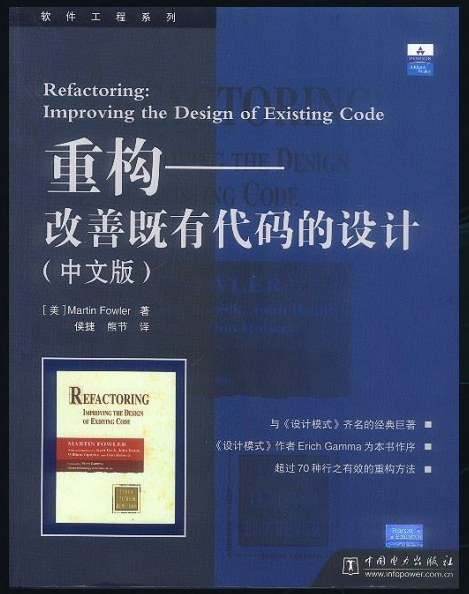
- 敏捷 + 精益
  - 精益思想：强调市场需求拉动（而非由生产方推动）的生产体系
  - “构建-度量-学习”循环
  - 敏捷是在软件研发领域落地精益的实践
- 看板方法
  - 将软件开发过程视为一种价值流
- Scrum与CMM的合流
- 敏捷和开源的核心价值是重叠的

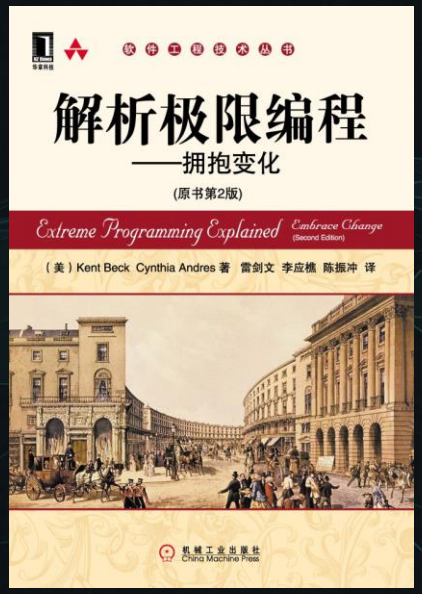


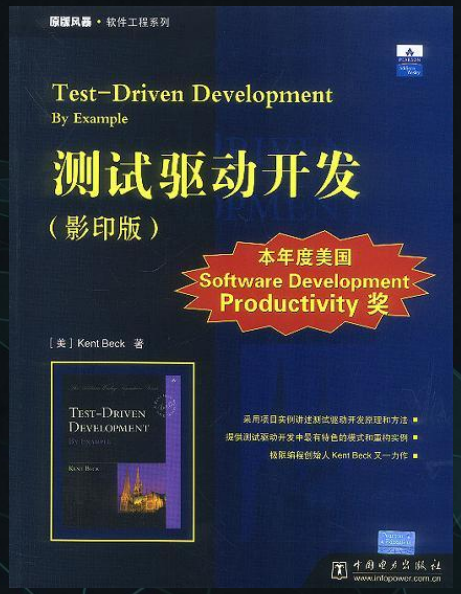


软件工程 2.0 的主要技术成果

- 代码重构技术
- 极限编程技术
  - 敏捷宣言（01年）
- 测试驱动开发技术
  - TDD
- 配置管理
  - Git 源码控制（05年）
- 项目管理技术与工具

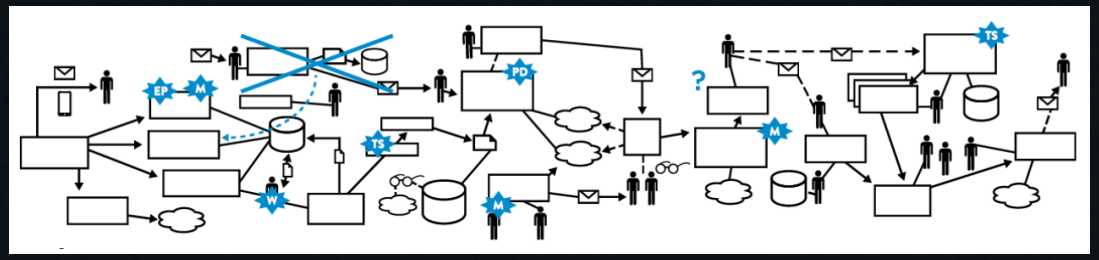


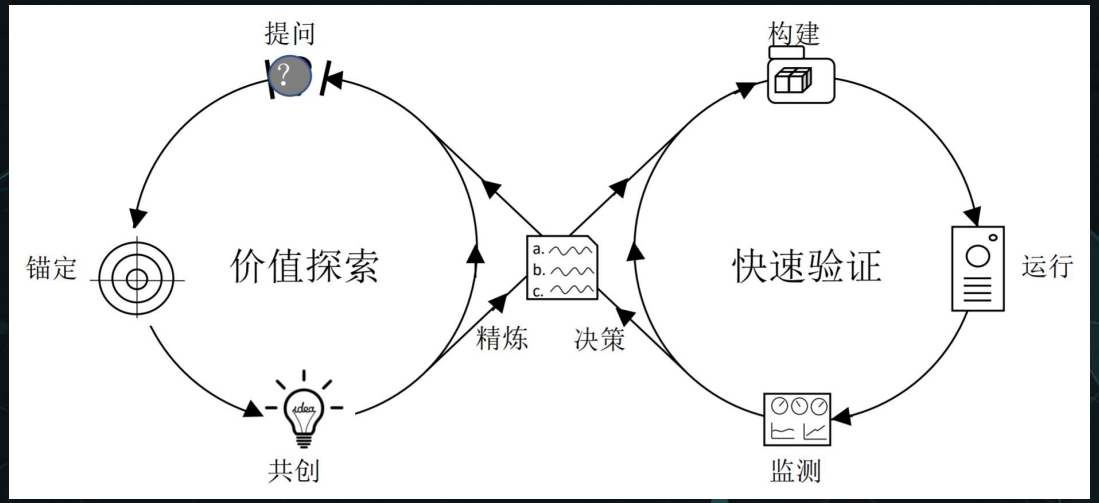




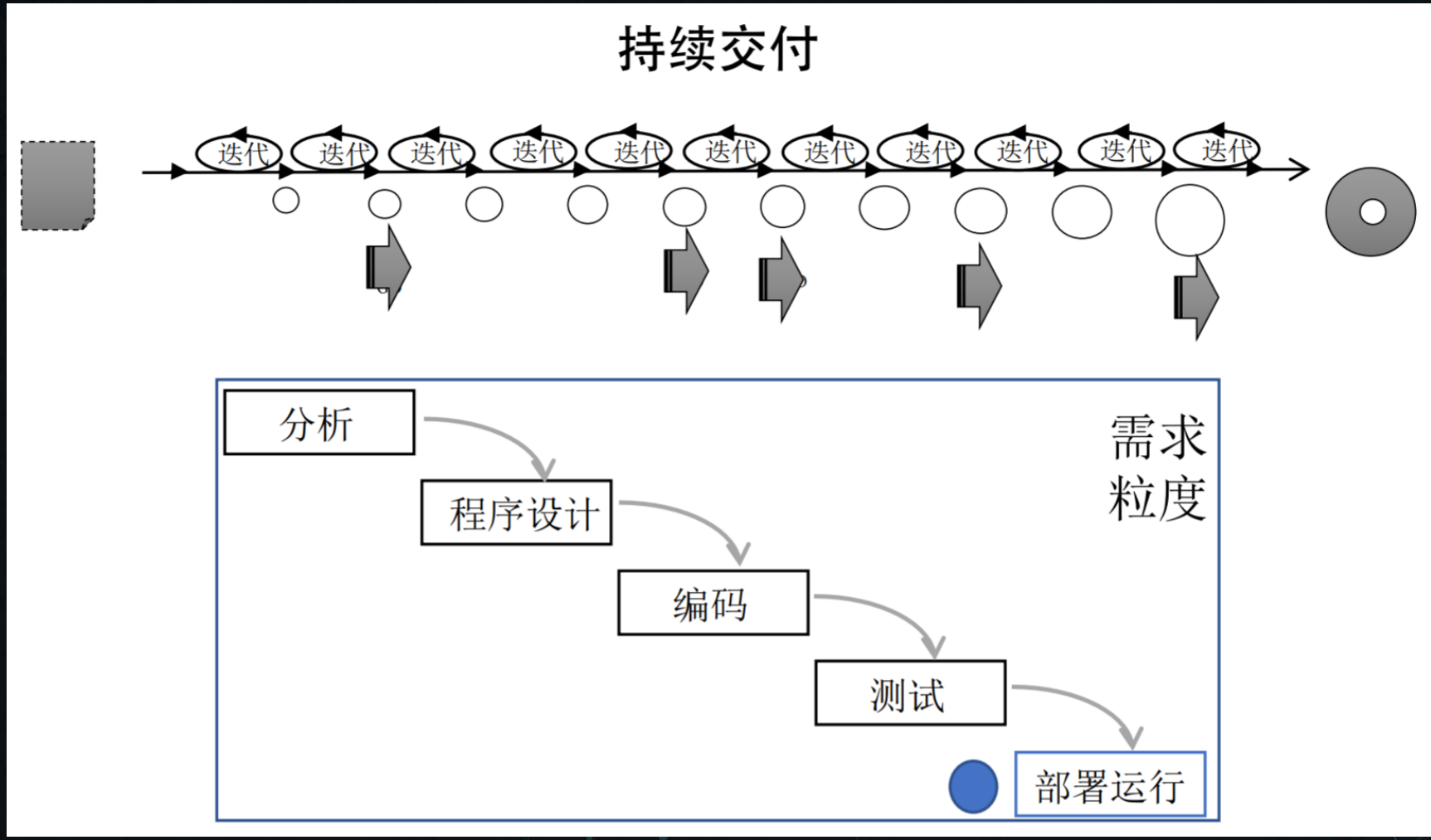
软件危机 3.0（从云计算到大数据）

- **VUCA × 价值探索 × 快速验证 × 持续交付**
  - 云计算对业务场景的冲击与变革
  - SaaS 服务化的潮流
  - 业务流程上云、融合、再造
  - 软件开发过程向运维端的延申
  - 软件开发过程向向业务端延申

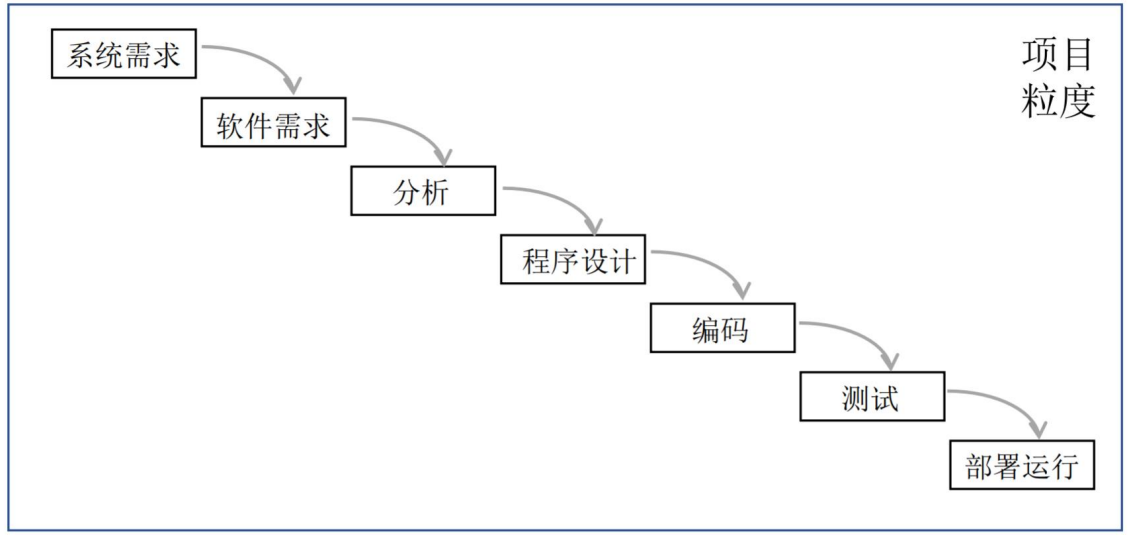




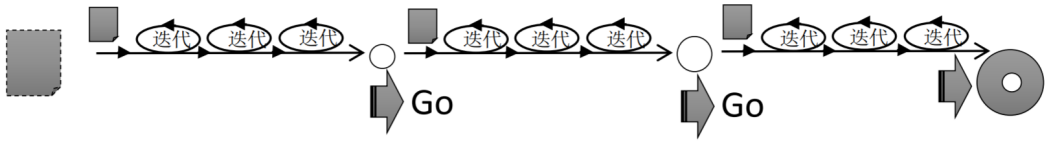
软件工程3.0 ——21世纪10年代

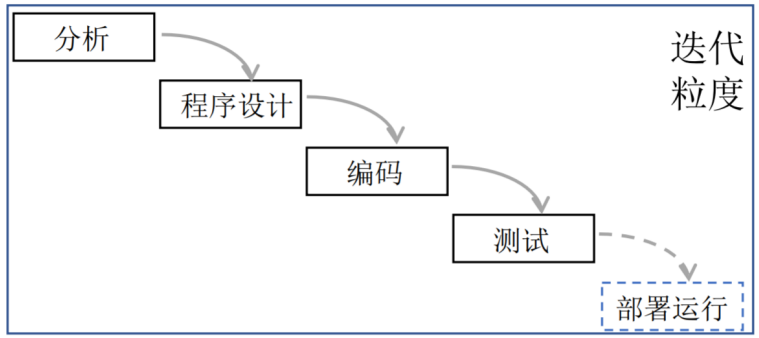


瀑布模型

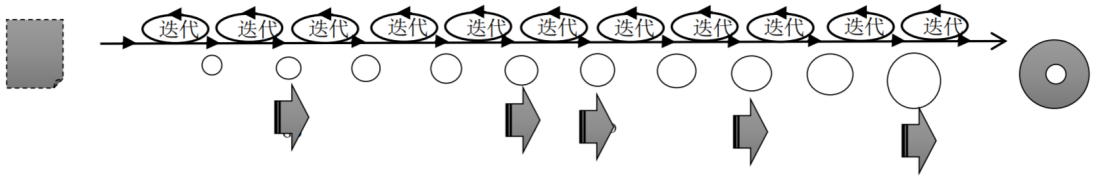


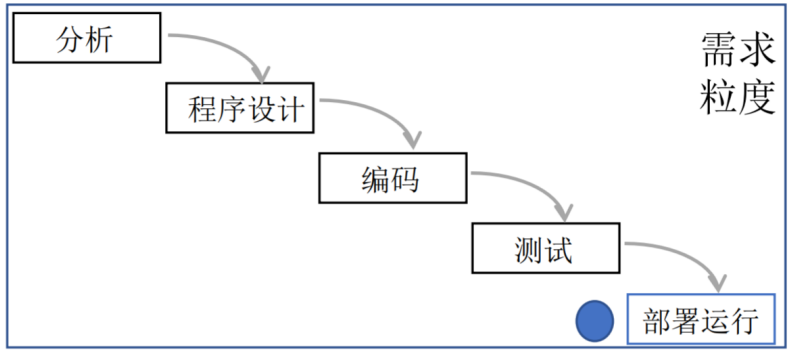
敏捷开发





持续交付





DevOps 技术的兴起

DevOps是一场**运动**，

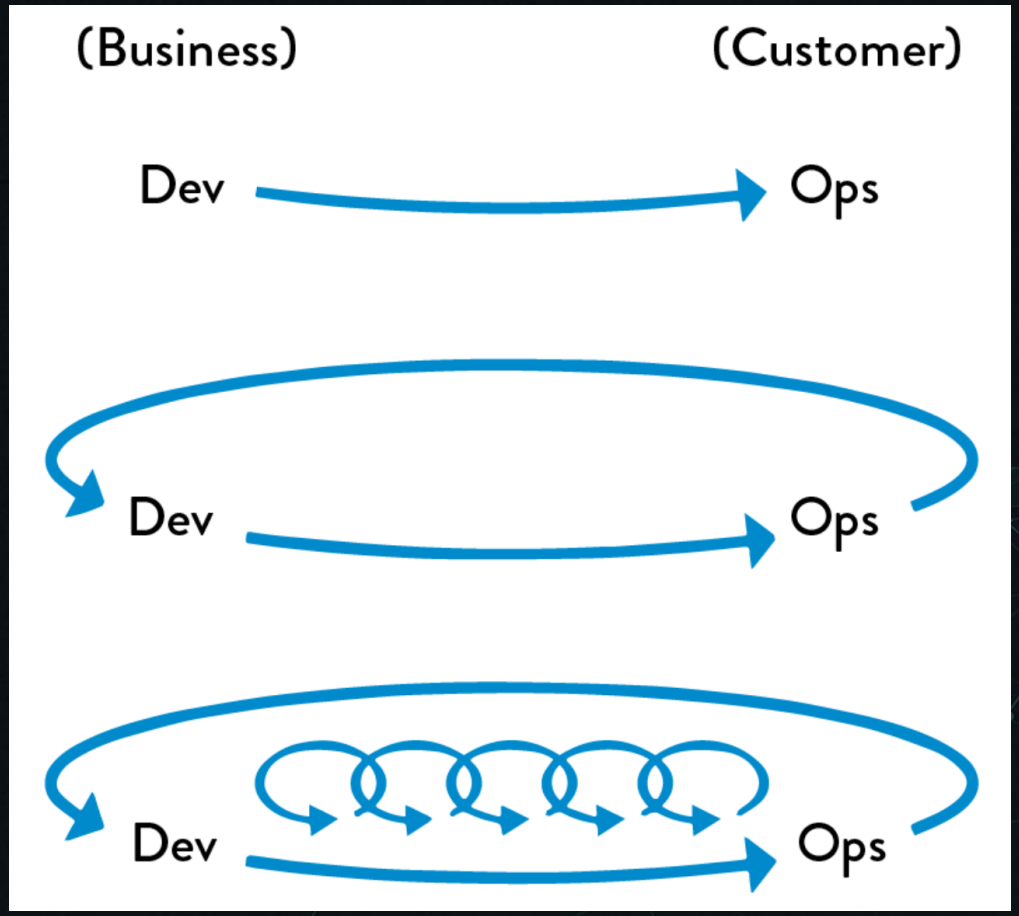
旨在寻找一些列**方法或实践**，

以提升不同角色在软件交付过程中的**协作质量与效率**，

从而提高软件服务的**交付速度**。

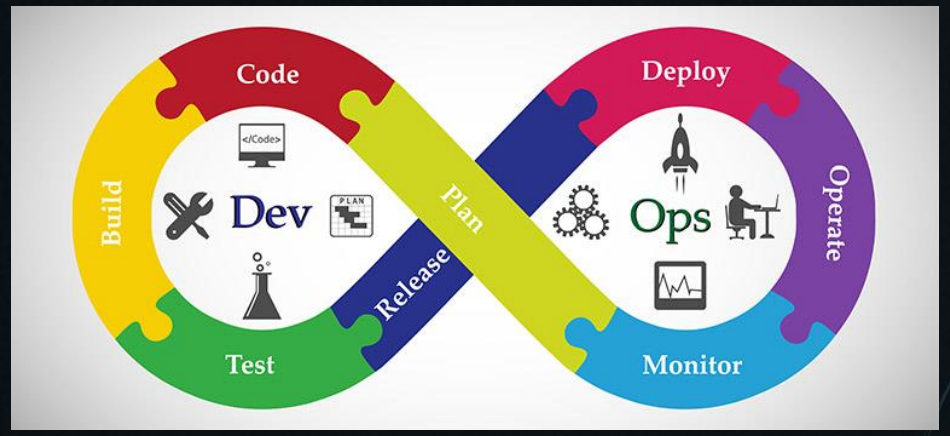
	1970s–1980s	1990s	2000s–Present
Era	Mainframes	Client/Server	Commoditization and Cloud
Representative technology of era	COBOL, DB2 on MVS, etc.	C++, Oracle, Solaris, etc.	Java, MySQL, Red Hat, Ruby on Rails, PHP, etc.
Cycle time	1–5 years	3–12 months	2–12 weeks
Cost	$1M–$100M	$100k–$10M	$10k–$1M
At risk	The whole company	A product line or division	A product feature
Cost of failure	Bankruptcy, sell the company, massive layoffs	Revenue miss, CIO's job	Negligible

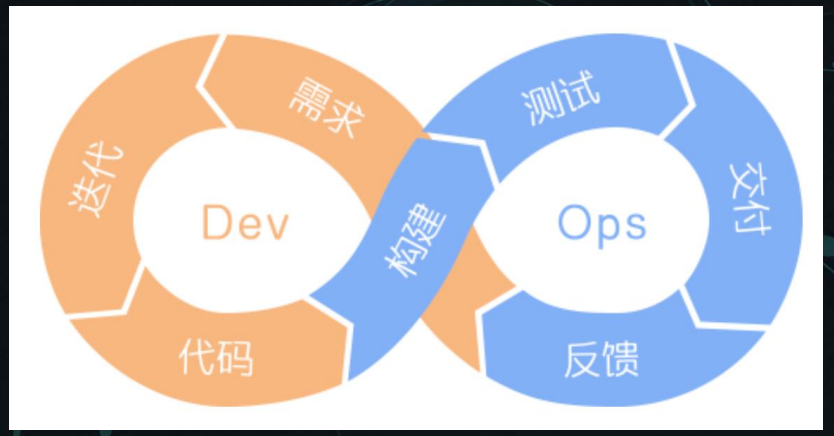
DevOps 三步工作法

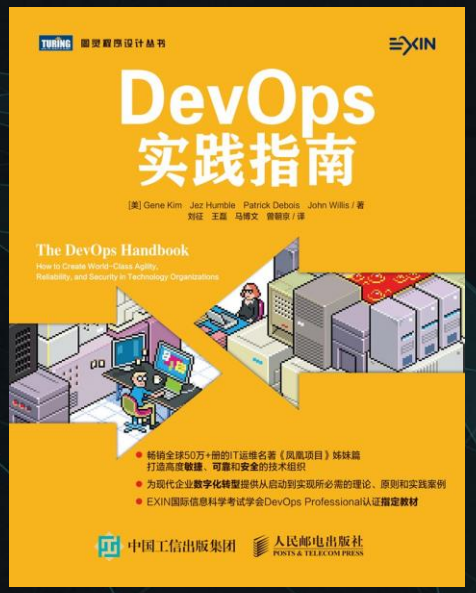


DevOps 最佳实践

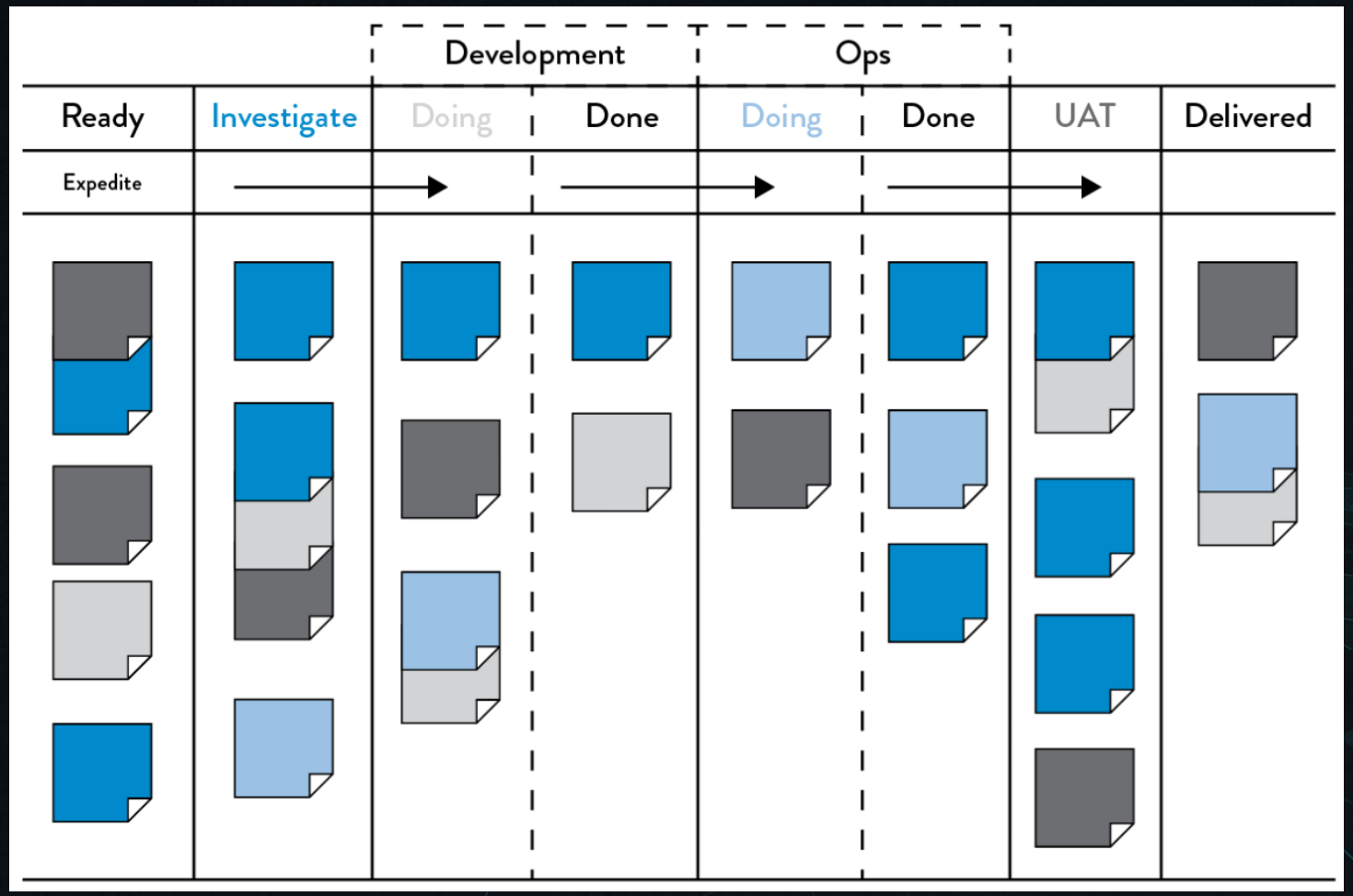
- 流动原则：它加速了从开发、运维到交付给客户的正向流程。
- 反馈原则：它使用组织构建安全、可靠的工作体系，并获得反馈；
- 持续学习与实验原则：它打造出一种高度信任的文化，并将改进和创新融入日常工作中。





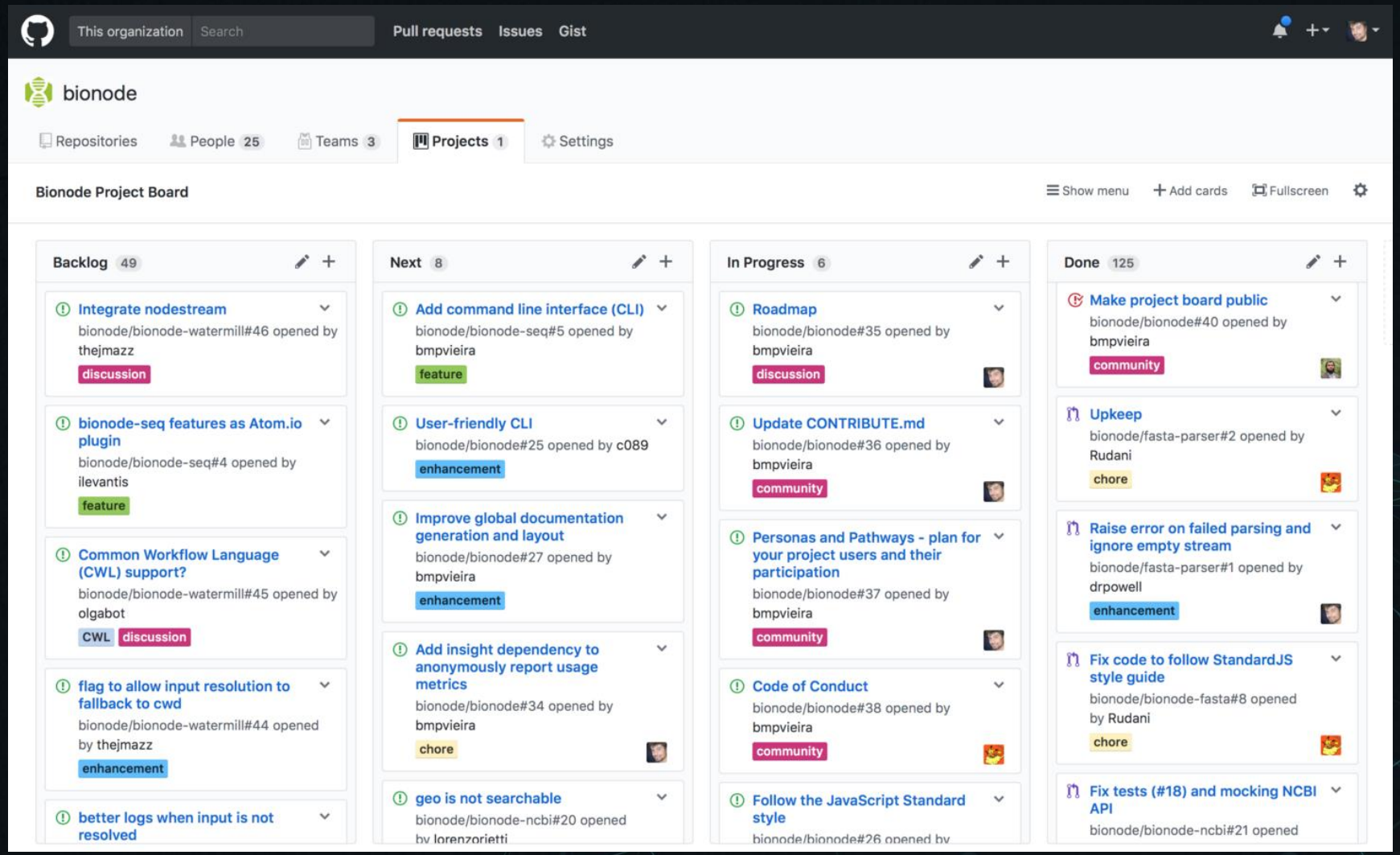


1、基于看板的价值流动可视化

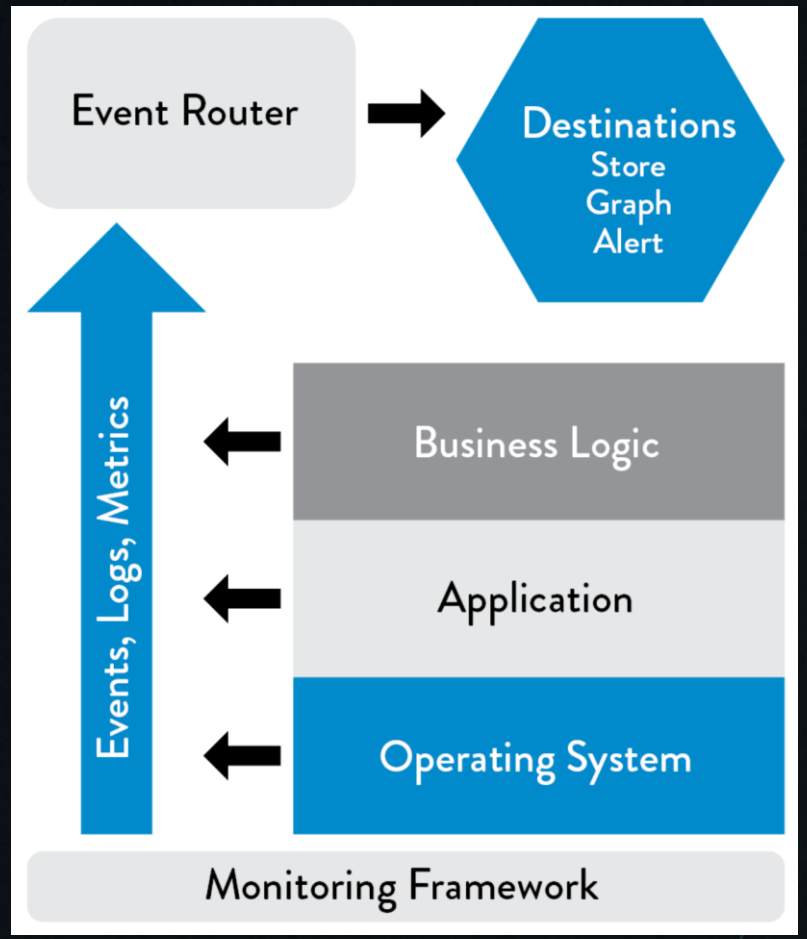


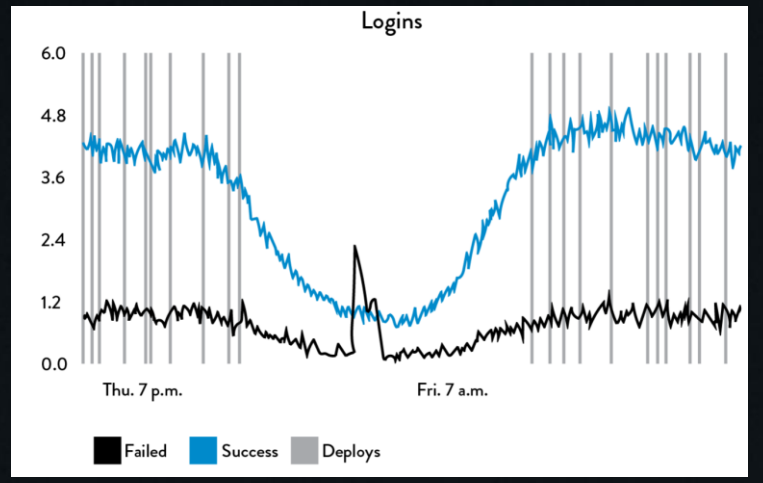
横跨需求、开发、测试、预生产和生产的看板示例

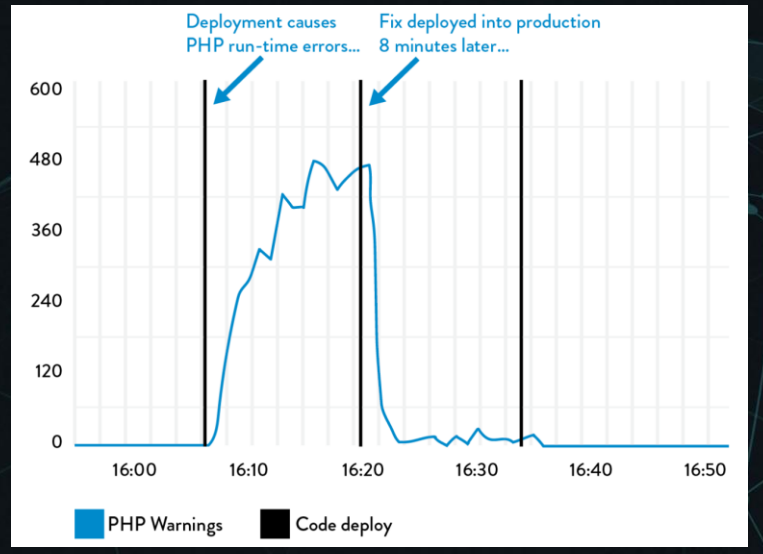
看板工具


This organization
Search
Pull requests
Issues
Gist
bionode
Repositories
People 25
Teams 3
Projects 1
Settings
Bionode Project Board
Show menu
Add cards
Fullscreen
Backlog 49
Integrate nodestream
bionode/bionode-watermill#46 opened by thejmazz
discussion
bionode-seq features as Atom.io plugin
bionode/bionode-seq#4 opened by ilevantis
feature
Common Workflow Language (CWL) support?
bionode/bionode-watermill#45 opened by olgabot
CWL
discussion
flag to allow input resolution to fallback to cwd
bionode/bionode-watermill#44 opened by thejmazz
enhancement
better logs when input is not resolved
Next 8
Add command line interface (CLI)
bionode/bionode-seq#5 opened by bmpvieira
feature
User-friendly CLI
bionode/bionode#25 opened by c089
enhancement
Improve global documentation generation and layout
bionode/bionode#27 opened by bmpvieira
enhancement
Add insight dependency to anonymously report usage metrics
bionode/bionode#34 opened by bmpvieira
chore
geo is not searchable
bionode/bionode-ncbi#20 opened by lorenzorietti
In Progress 6
Roadmap
bionode/bionode#35 opened by bmpvieira
discussion
Update CONTRIBUTE.md
bionode/bionode#36 opened by bmpvieira
community
Personas and Pathways - plan for your project users and their participation
bionode/bionode#37 opened by bmpvieira
community
Code of Conduct
bionode/bionode#38 opened by bmpvieira
community
Follow the JavaScript Standard style
bionode/bionode#26 opened by
Done 125
Make project board public
bionode/bionode#40 opened by bmpvieira
community
Upkeep
bionode/fasta-parser#2 opened by Rudani
chore
Raise error on failed parsing and ignore empty stream
bionode/fasta-parser#1 opened by drpowell
enhancement
Fix code to follow StandardJS style guide
bionode/bionode-fasta#8 opened by Rudani
chore
Fix tests (#18) and mocking NCBI API
bionode/bionode-ncbi#21 opened

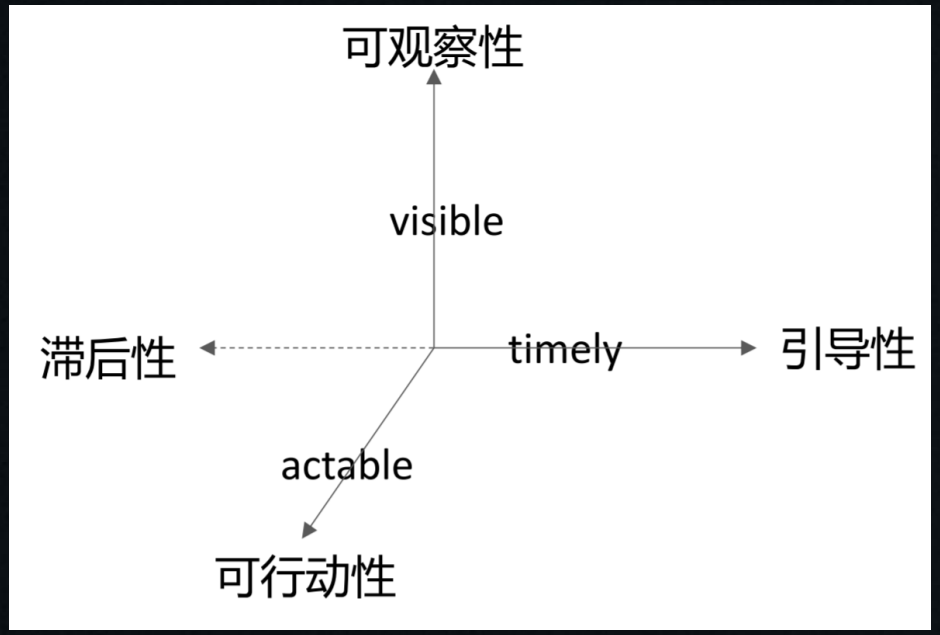
2、建立全面的观测体系与反馈机制

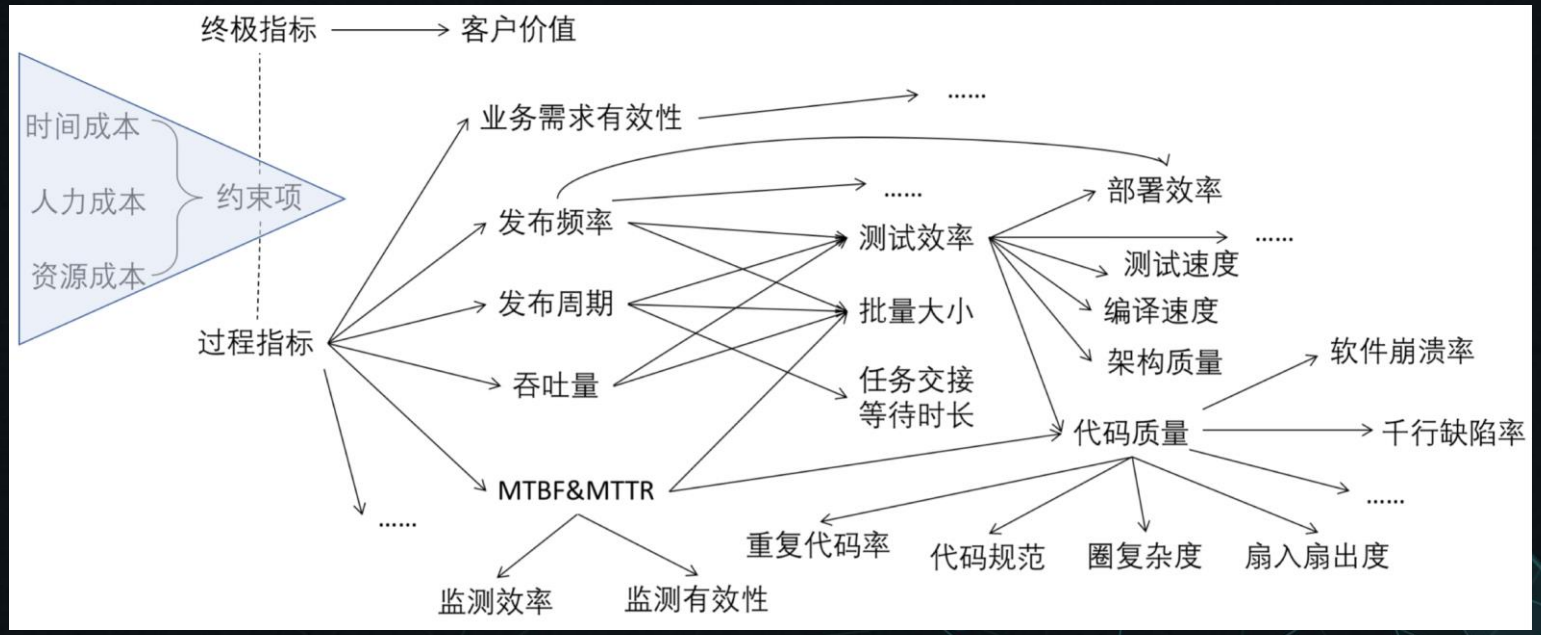




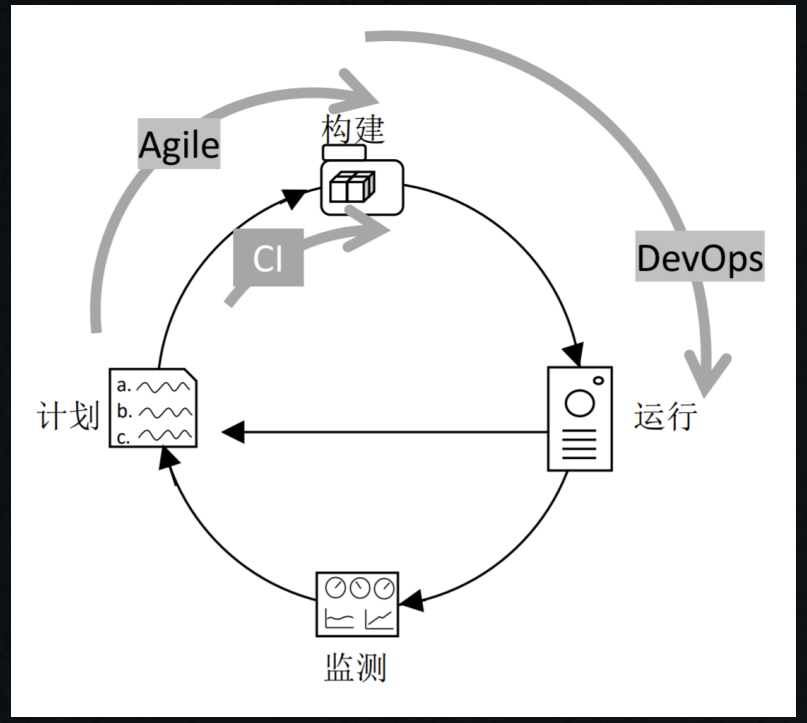


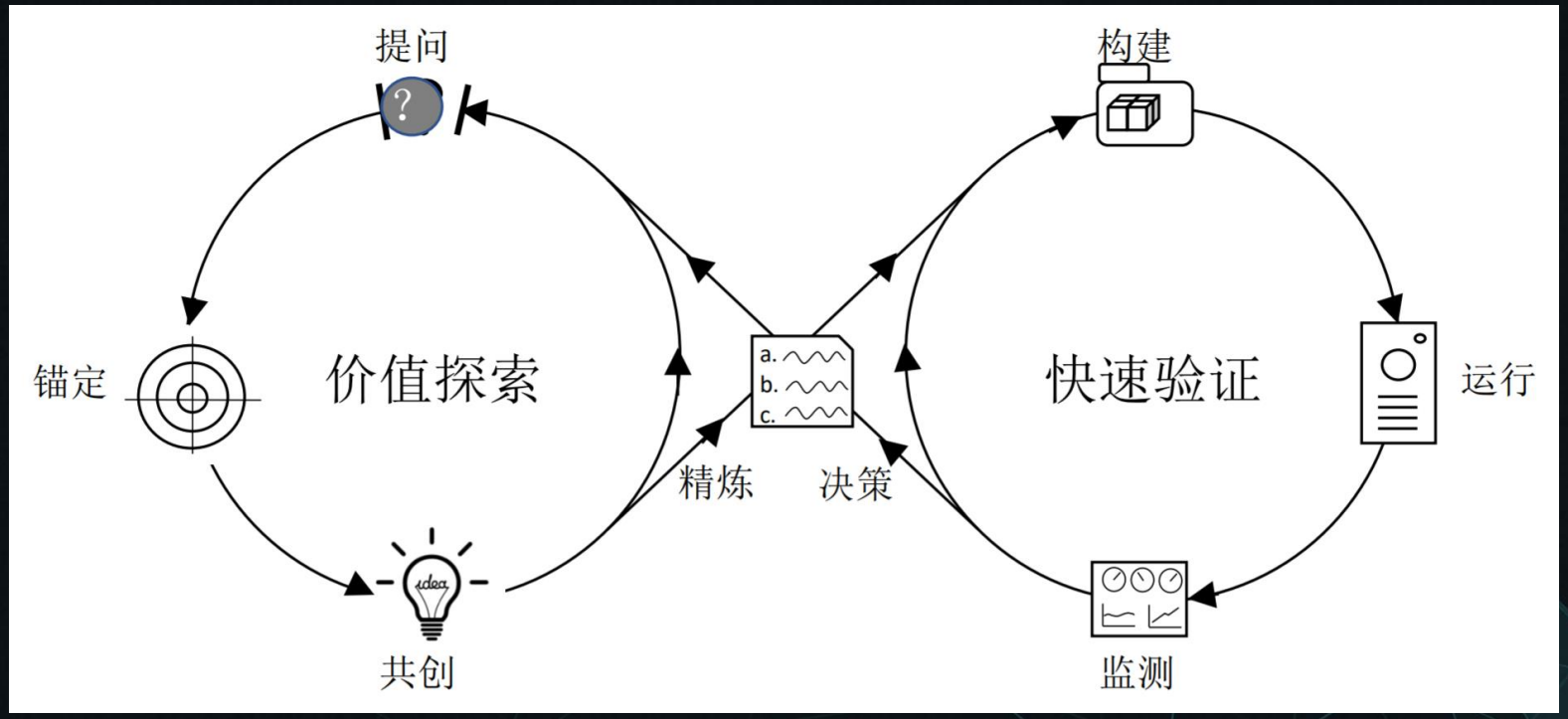
理解"度量"


可观察性
visible
滞后性
timely
引导性
actable
可行动性

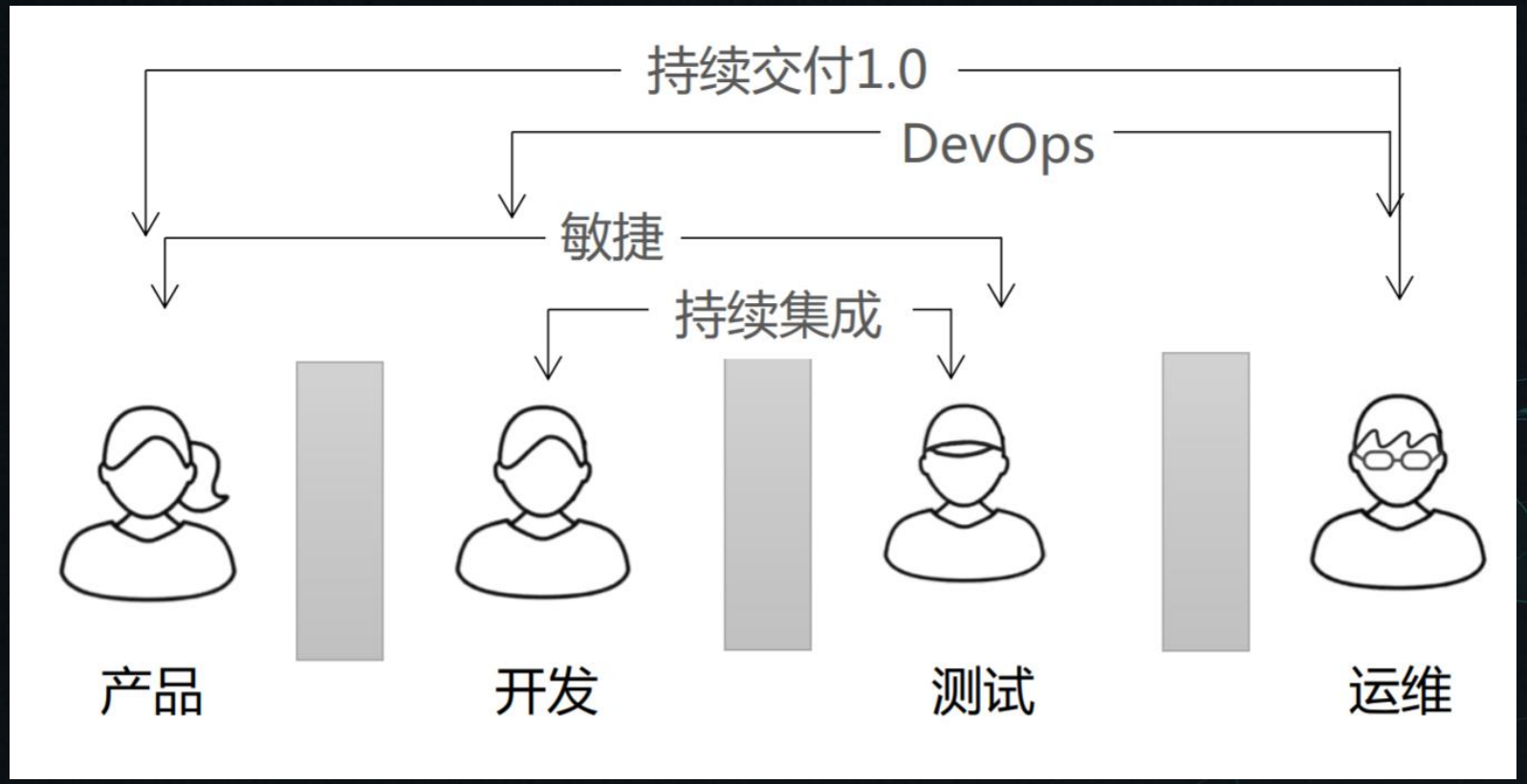

终极指标
客户价值
时间成本
人力成本
资源成本
约束项
过程指标
业务需求有效性
.....
发布频率
.....
部署效率
测试效率
.....
测试速度
发布周期
批量大小
编译速度
吞吐量
任务交接
等待时长
架构质量
软件崩溃率
代码质量
千行缺陷率
MTBF&MTTR
.....
.....
重复代码率
代码规范
圈复杂度
扇入扇出度
监测效率
监测有效性

3、持续学习与改进原则

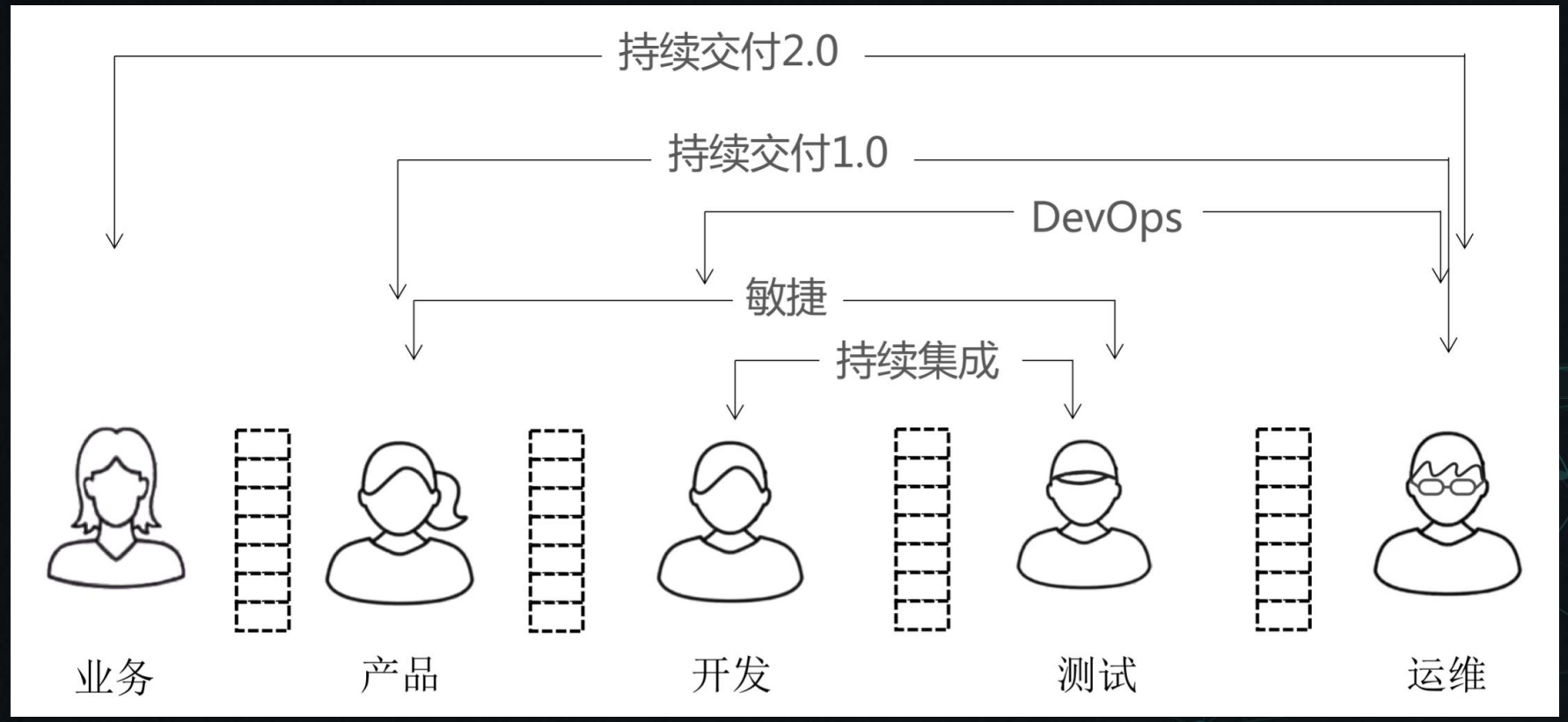

Agile
构建
CI
DevOps
计划
a.
b.
c.
运行
监测


提问
?
构建
锚定
价值探索
a.
b.
c.
精炼
决策
快速验证
运行
idea
共创
监测

Put it all together：持续交付 1.0



进一步演进：持续交付 2.0



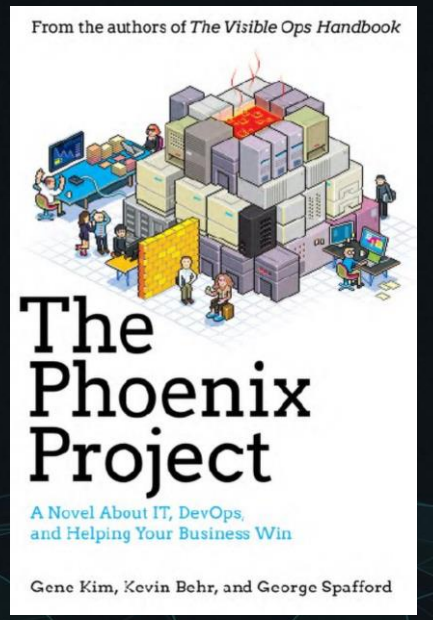
less communication, more alignment

软件工程 3.0 的主要技术成果

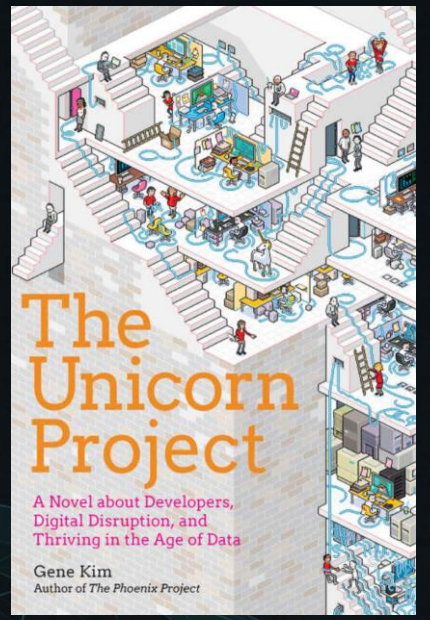
- **云原生与 X-Ops 技术**
  - DevOps、DevSecOps、GitOps、DataOps、AIOps
- **云原生与 CI / CD 技术**
  - Git 大平台
    - GitHub Action
    - GitLab Runner
  - CI / CD 工具
    - Travis CI
    - Jenkins / Jenkins X
- **协作流程机器人技术**

#	开发者账号	活跃度
1	dependabot[bot]	36,082,423.2
2	dependabot-preview[bot]	10,169,281.1
3	pull[bot]	9,591,558.7
4	renovate[bot]	2,668,048.6
5	github-learning-lab[bot]	1,988,666.3
6	github-actions[bot]	984,674.2
7	direwolf-github	894,975.0
8	codecov[bot]	826,249.8
9	snyk-bot	654,743.4
10	sonarcloud[bot]	527,040.0

GitHub 2020 年全球开发者活跃账户

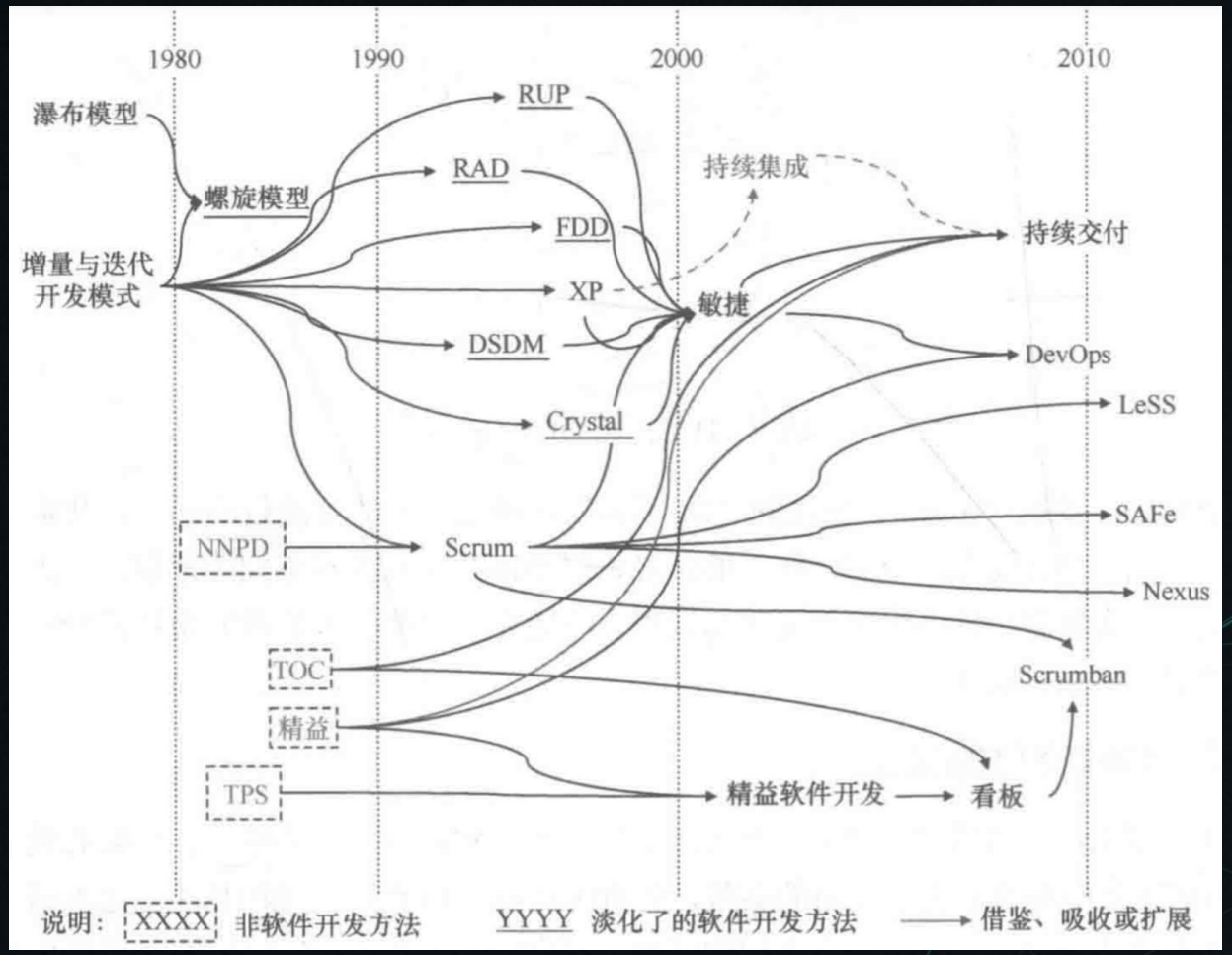


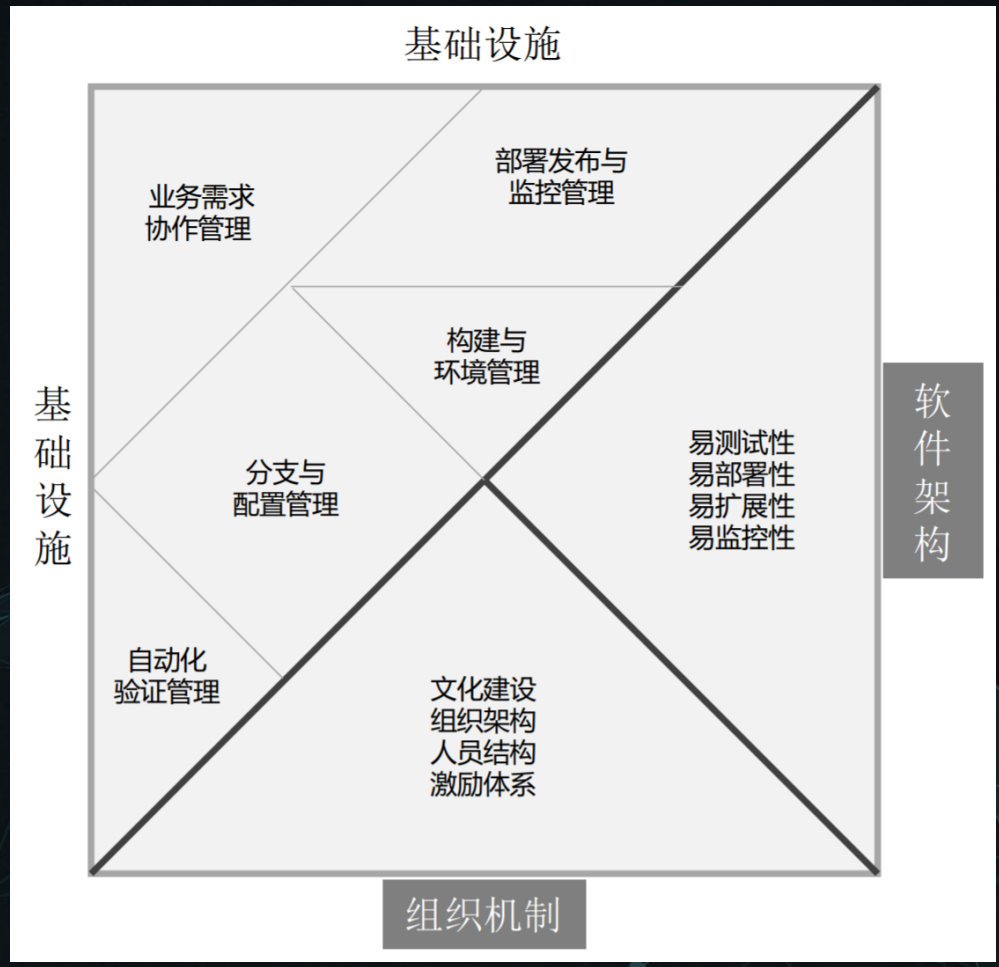
2013



2019

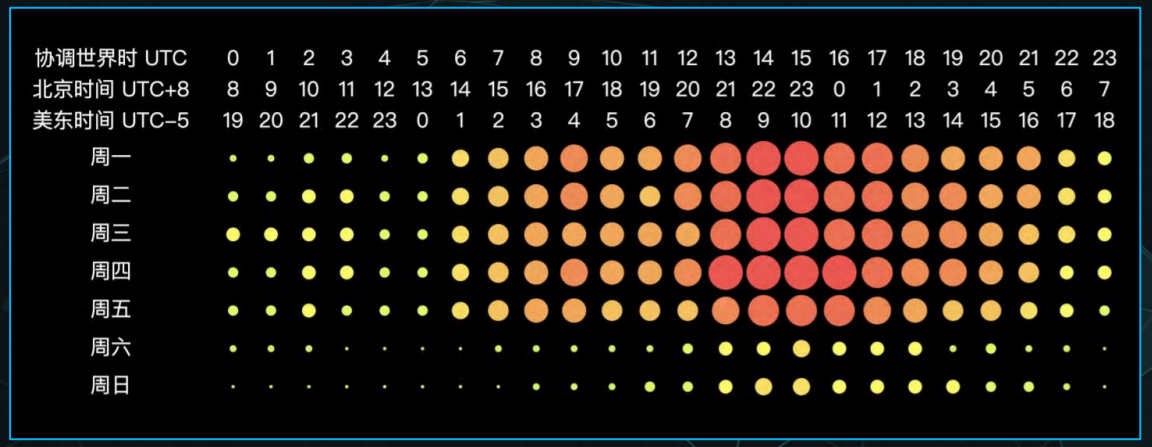
再总结：软件工程方法编年史





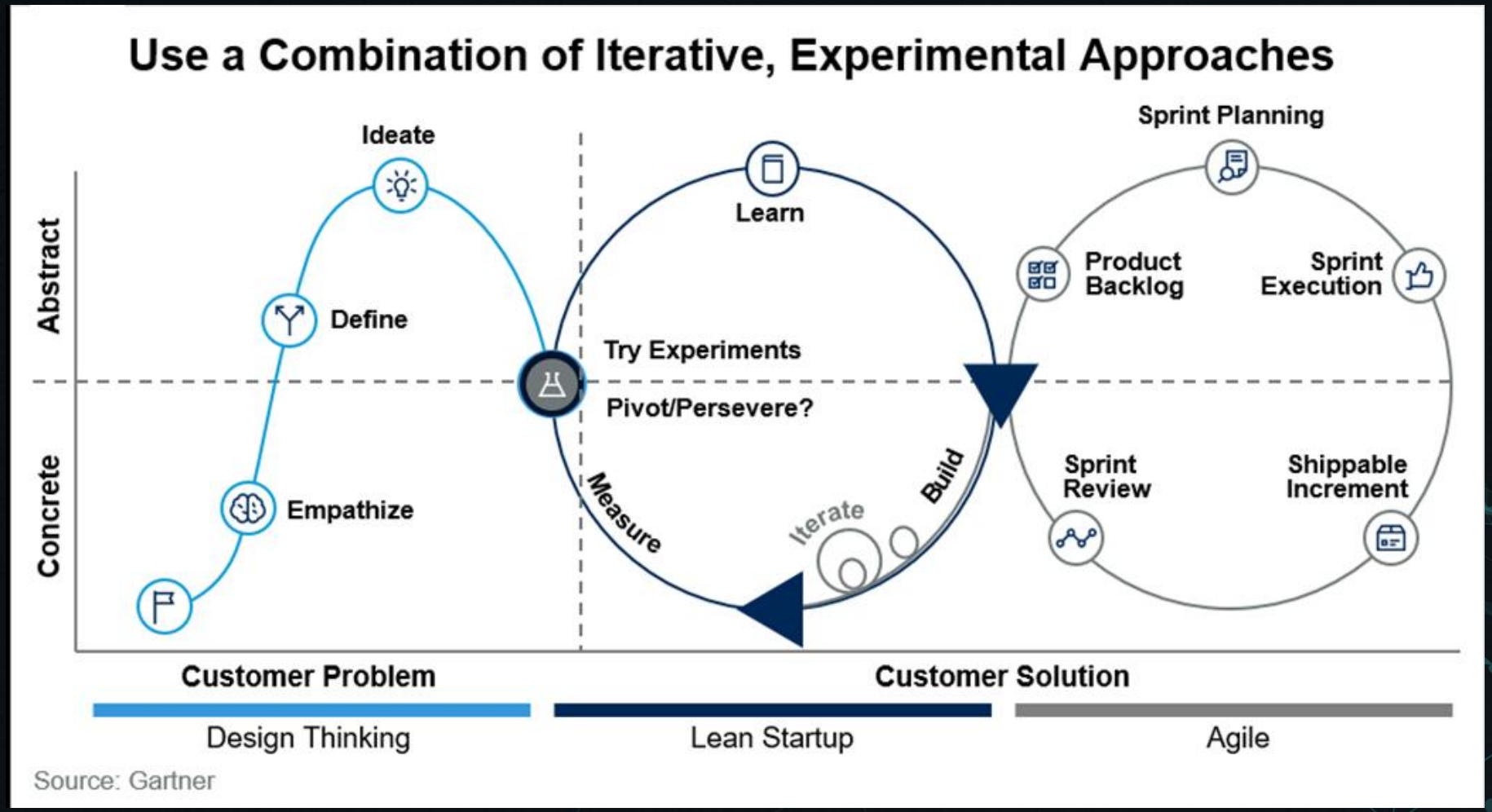
软件危机还会继续吗？（人工智能与未来）

- **人类数字空间的持续扩张与演化**
  - 全行业的数字化转型
  - 全线上协作时代
  - 软件吞噬世界，成为基础设施
  - 软件基础设施需要全球化协作
  - 人人编程，全民开发者时代的到来
  - 全球软件供应链的风险
  - Auto X-Ops



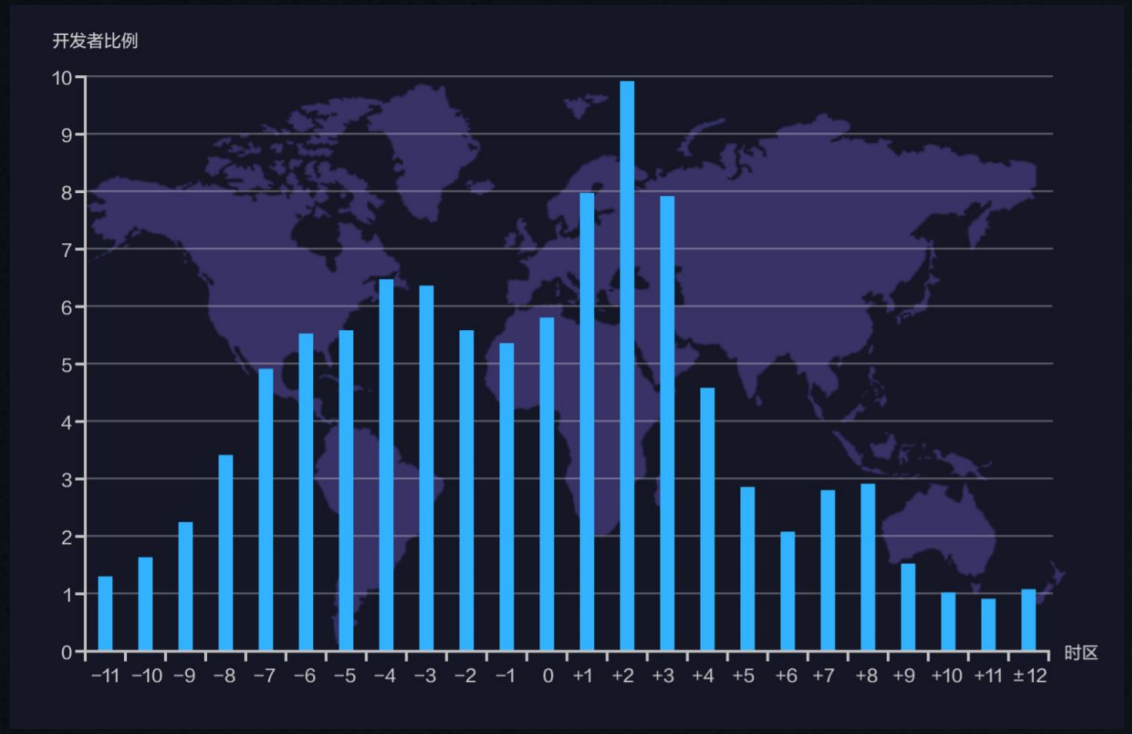
GitHub 2020 年全球日志时间分布情况

设计思维 + 精益 + 敏捷 + ……

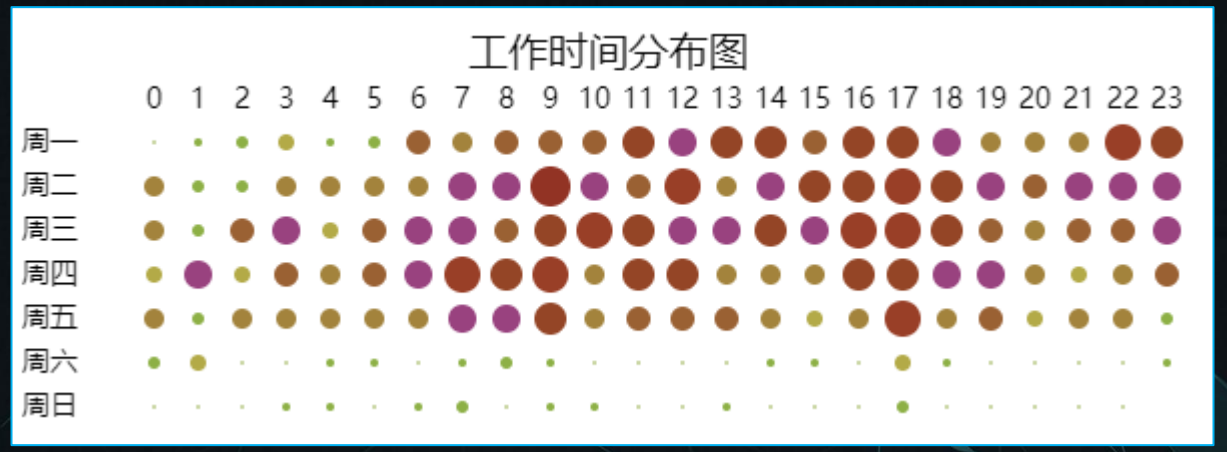


研发流程持续向两端快速延申（纵向一体化）

软件开发的全球化趋势



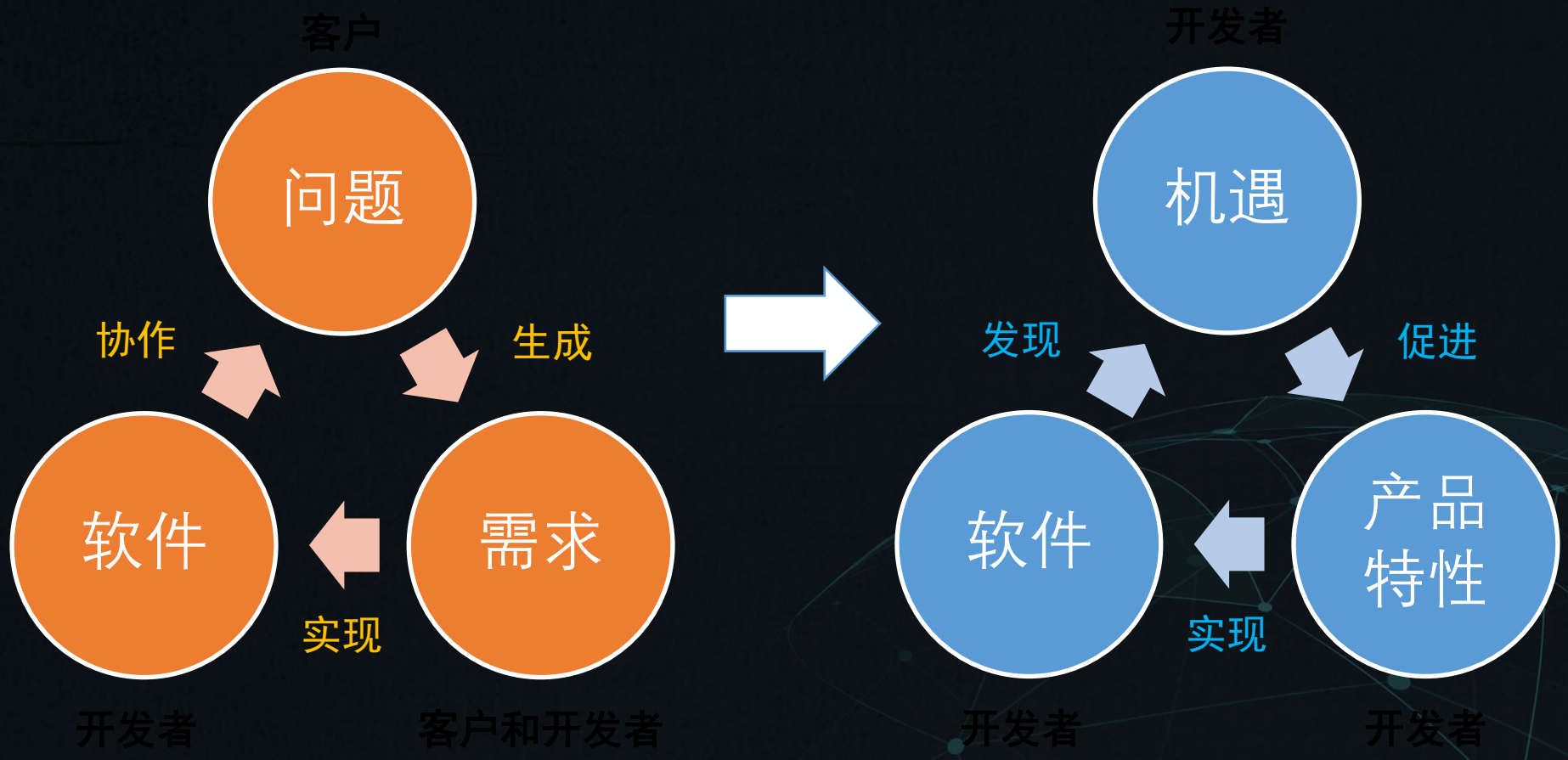
GitHub 2020 年全球开发者时区人数分布



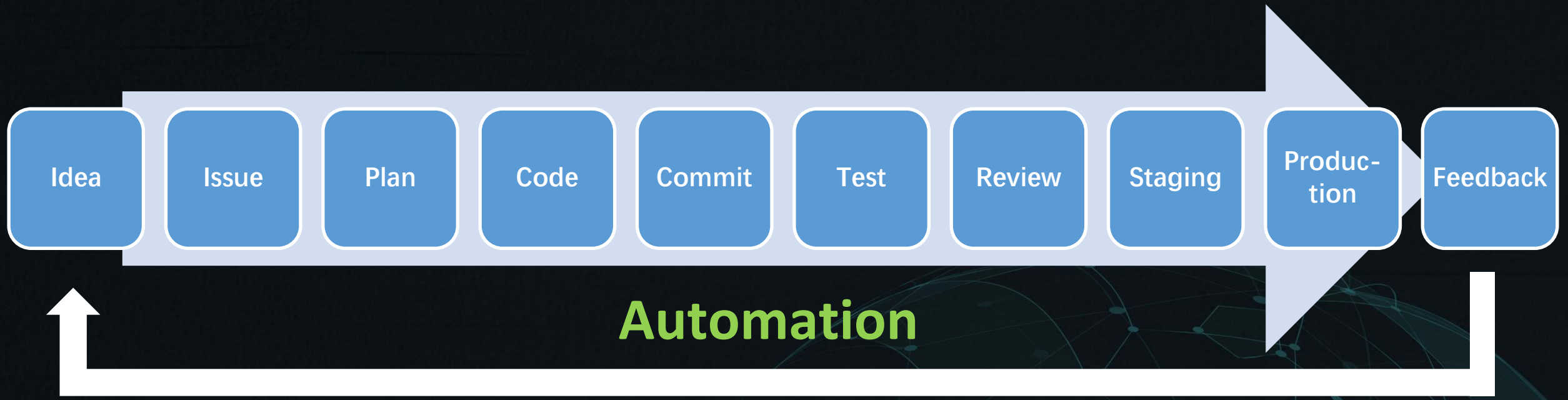
Apache Hadoop-ozone项目的全球化程度

来源：GitHub 2020 数字洞察报告, 2021-02.

从软件项目到软件产品的快速转变



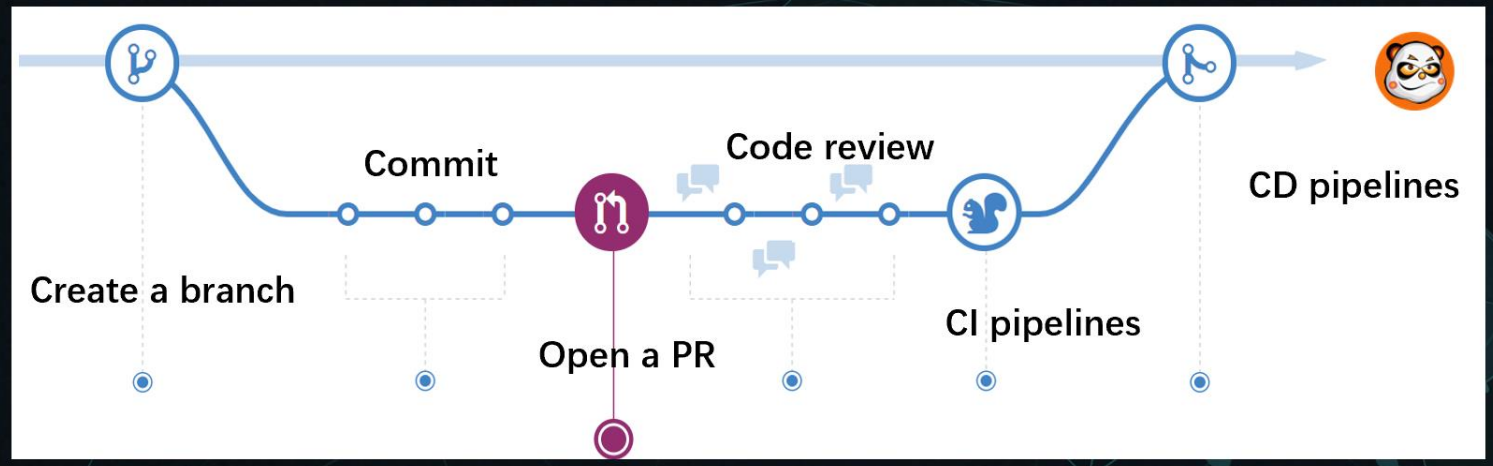
研发工作流的标准化与规范化

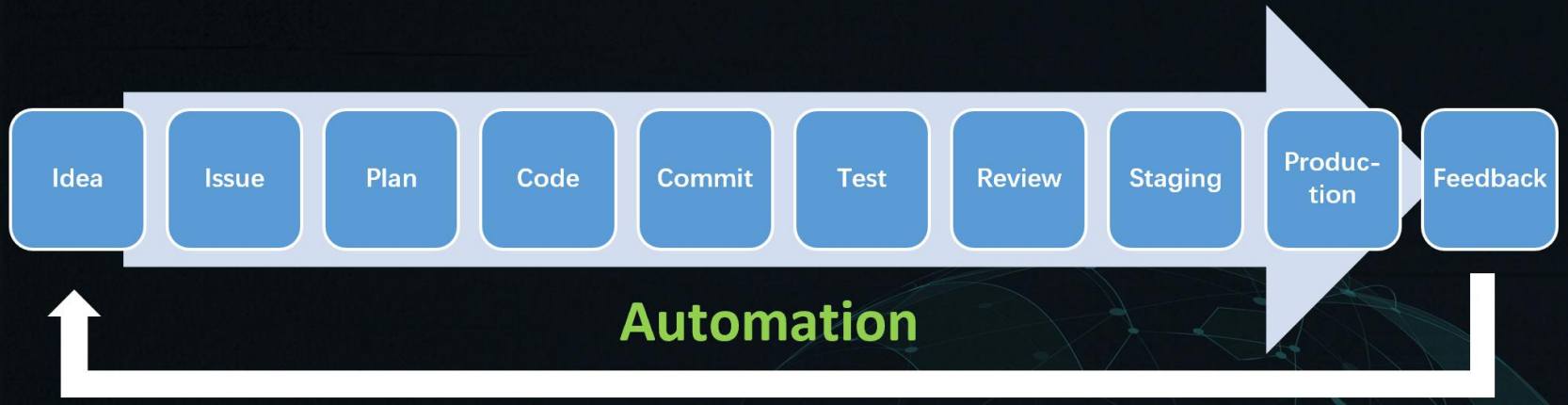


开源软件开发过程是一种最佳工程实践

Pull Request：

- **Branch:** 分布式协作开发
- **Commit:** 代码规范
- **Code Review:** 代码质量控制
- **CI:** 单元测试与集成测试
- **CD:** 持续部署 **(staging & production)**

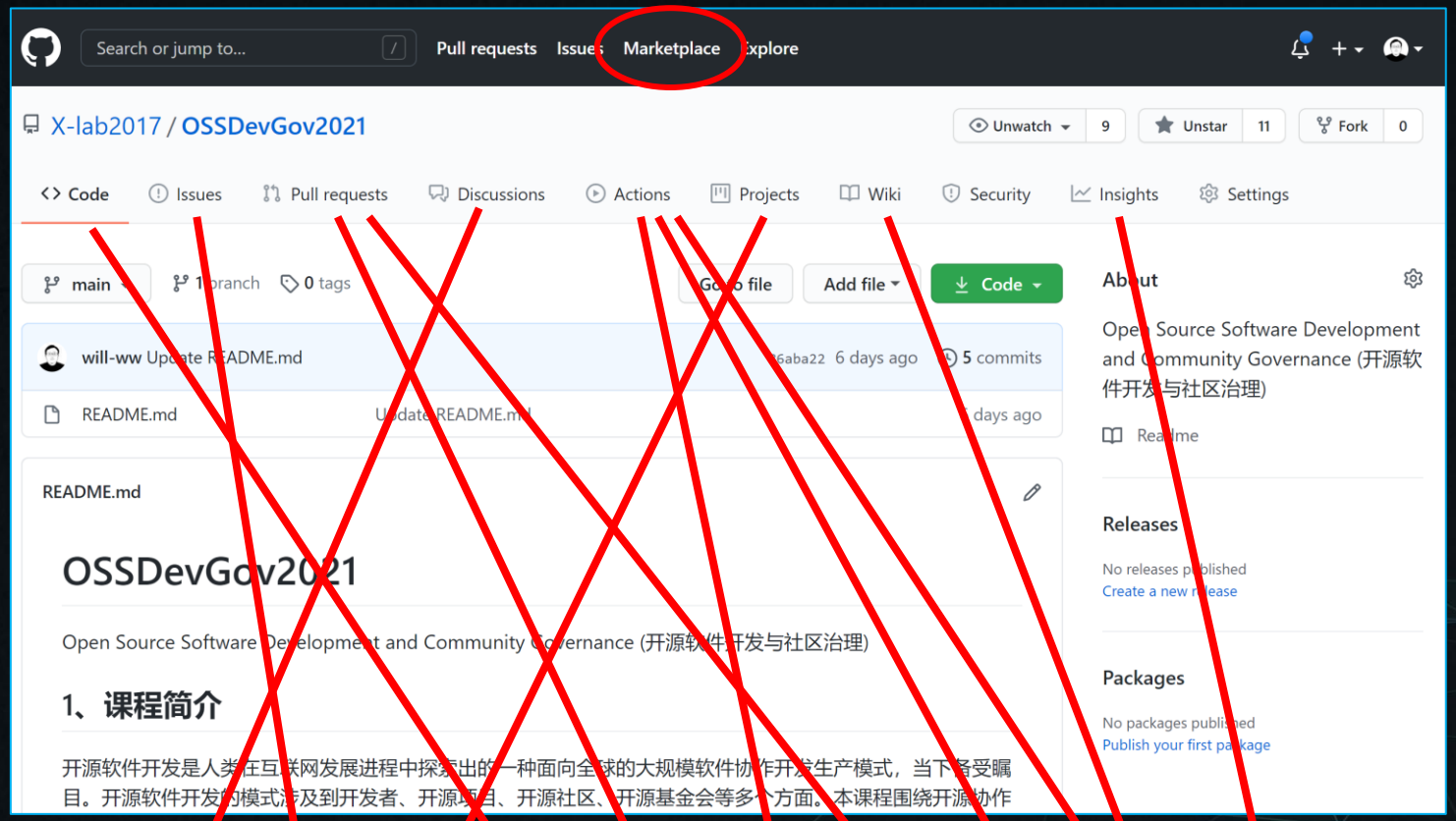




- 分布式协作
  - 大规模
  - 标准化
  - 自动化
  - 高效

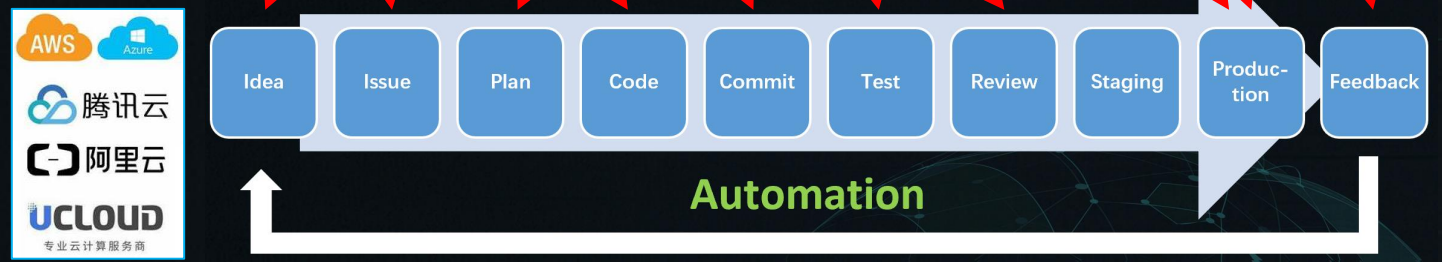
开源软件开发的协作与平台化

创新点子

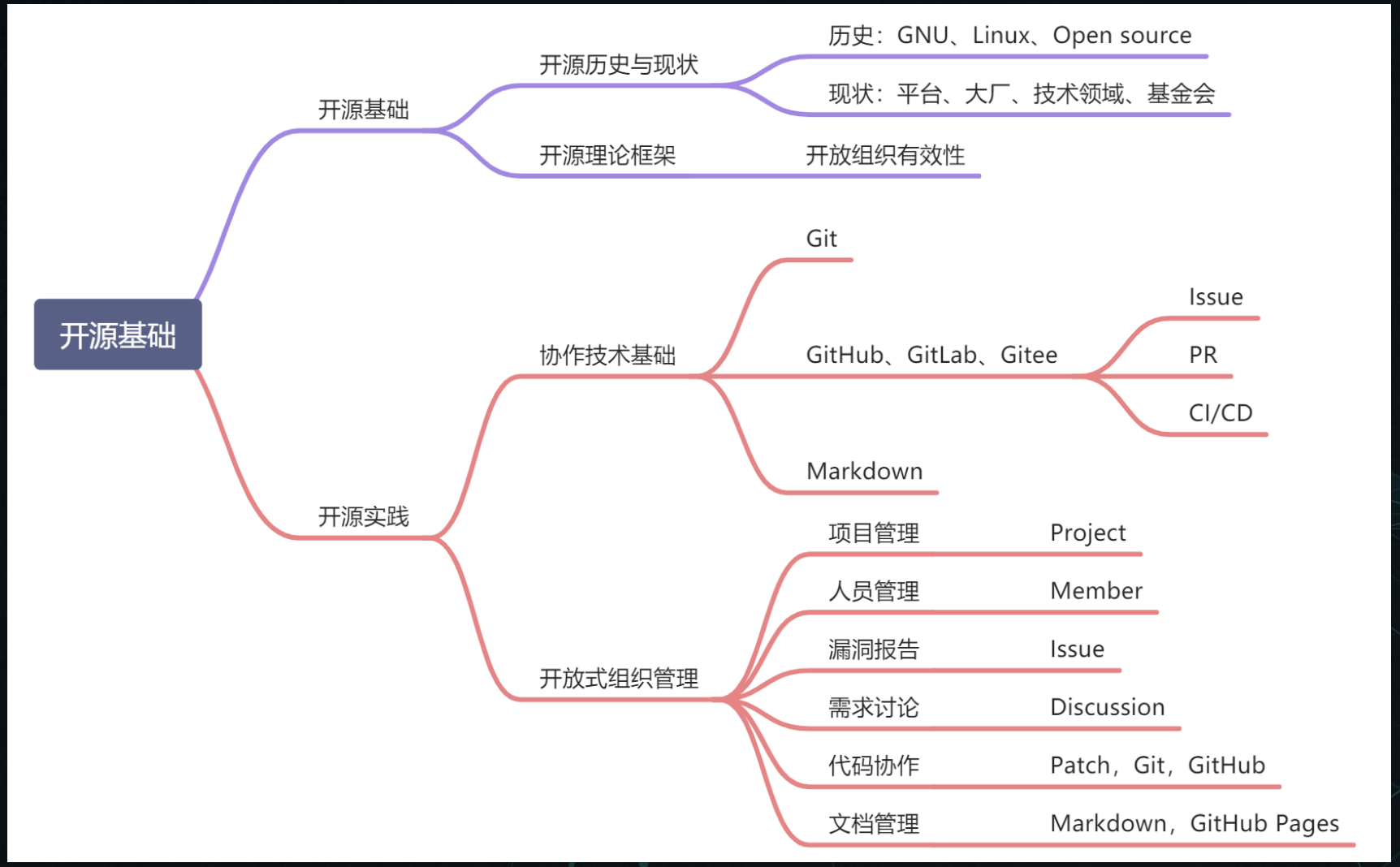


全球市场

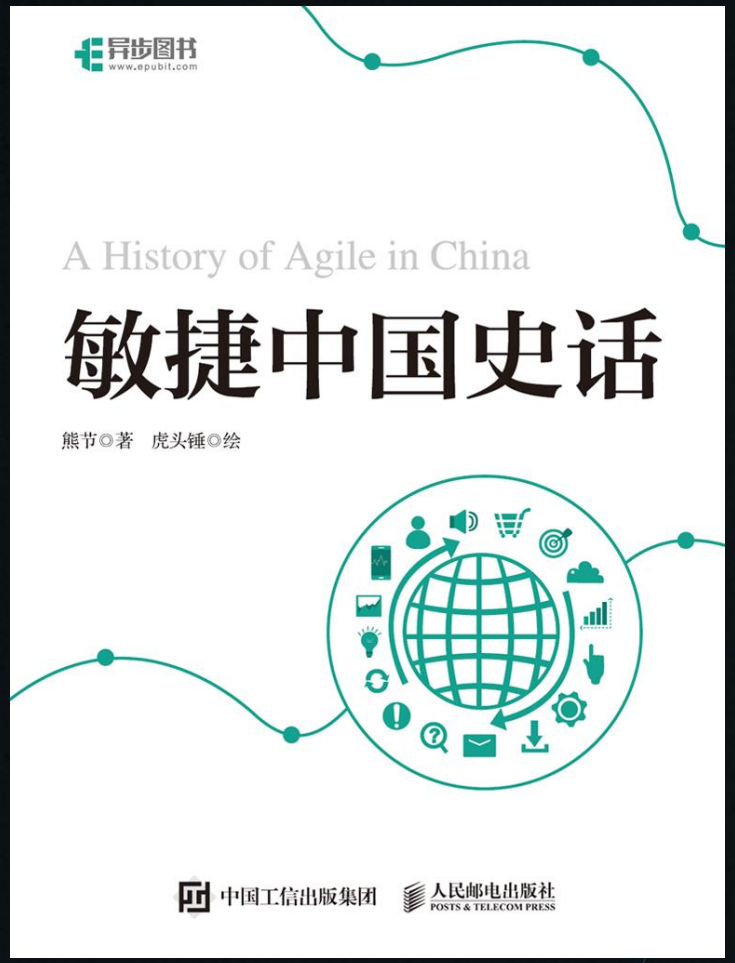
用户创新

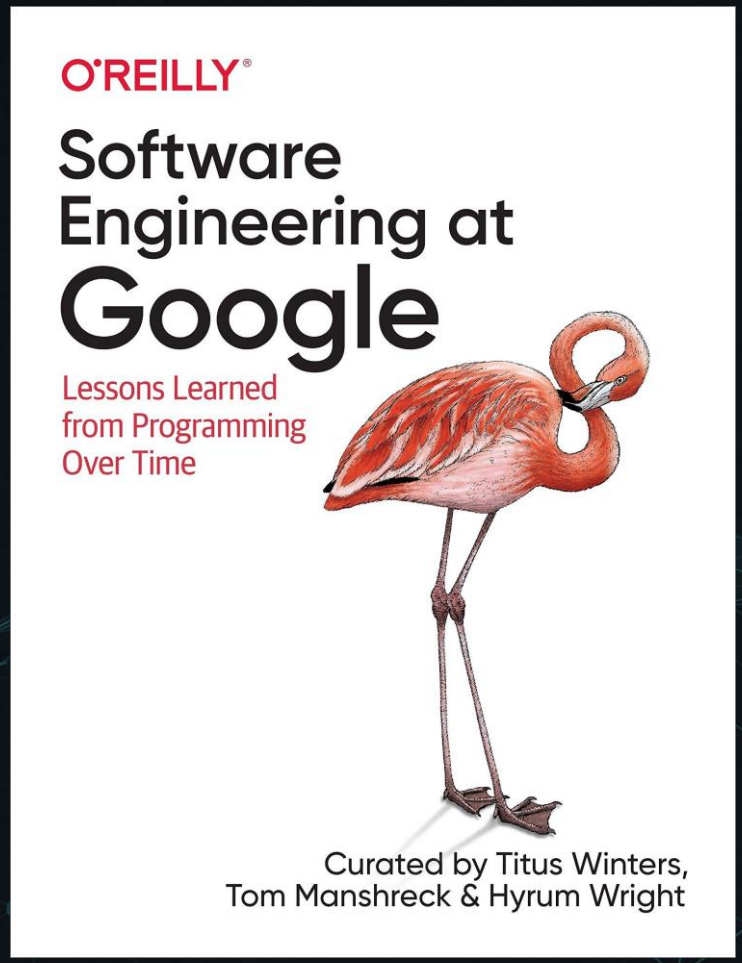


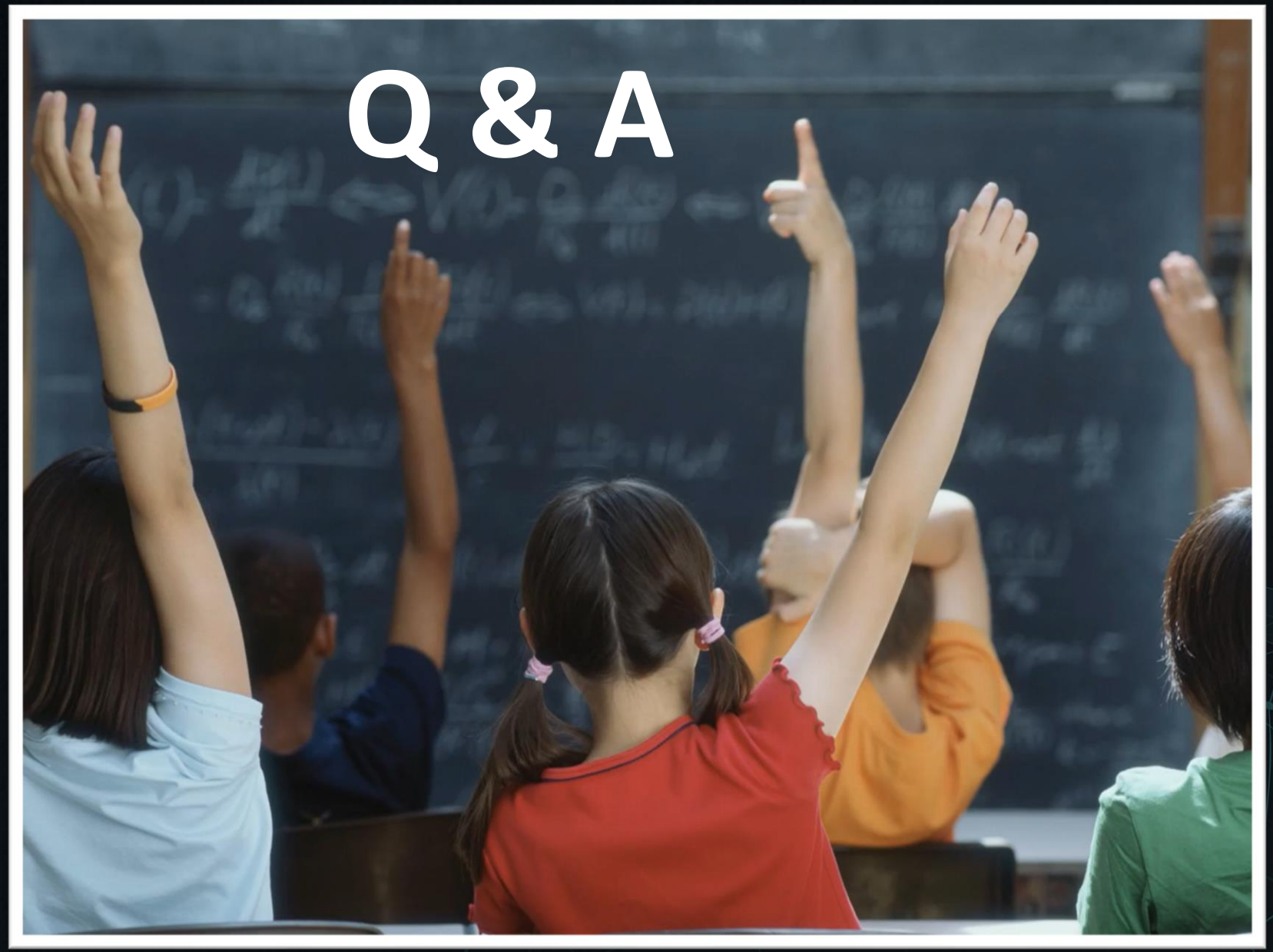
对开源有全面理性的认识，并掌握参与开源项目的基本技术能力。



延申阅读


异步图书
www.epubit.com
A History of Agile in China
敏捷中国史话
熊节◎著 虎头锤◎绘
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS


O'REILLY
Software Engineering at Google
Lessons Learned from Programming Over Time
Curated by Titus Winters, Tom Manshreck & Hyrum Wright


Q & A