7 操千曲而后晓声—软件工程基础

仓储管理系统架构

基础资料

- 仓库
- 库区
- 库位
- 货主
- 客户
- 物料
- 车辆司机
- 容器载具

入库管理

- PO订单
- ASN门户
- 收货处理
- 上架策略
- IQC 检验
- 入库查询
- 标签打印
- 组托解托

库位作业

- 盘点
- 移库
- 调拨
- 库存调整
- 库位锁定
- 库位合并
- BOM
- 加工拆解

出库作业

- 出货通知
- 波次配置
- 出货执行
- 补货执行
- 标签打印
- 单据打印
- 出货交接
- 下架策略

发货处理

- 发货配载
- 装车处理
- 发货查询

作业管理

- 作业分配
- 作业监控
- 作业处理

增值应用

- 条码溯源
- 人员绩效
- 移动应用
- 智能自动化

逆向物流

- 销售退货
- 生产退料
- 其他退货

接口

- 资料接口
- 入库接口
- 报检接口
- 出库接口
- 自动化接口
- 数据查询接口





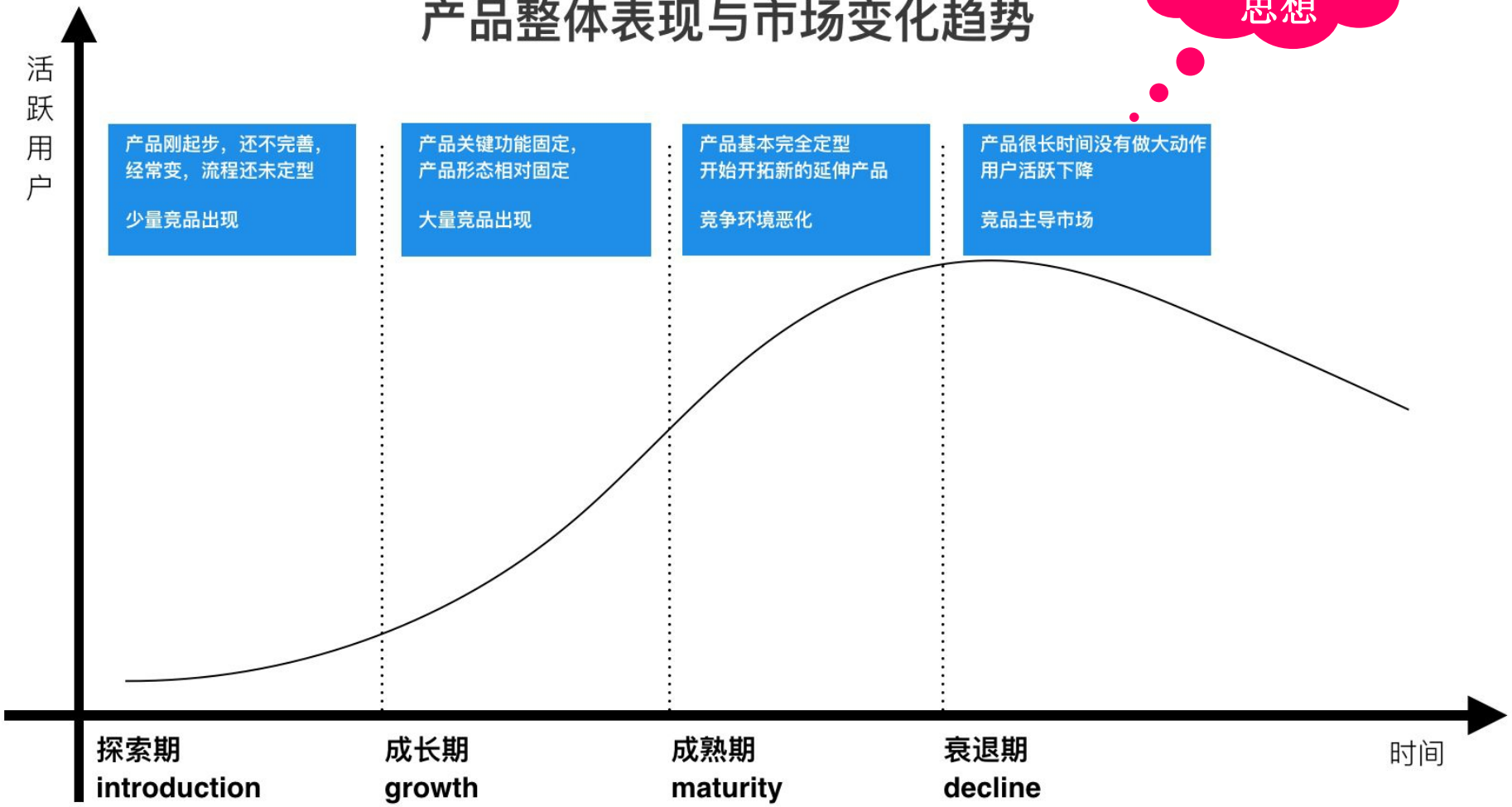


软件危机的教训+软件系统日益庞大且复杂

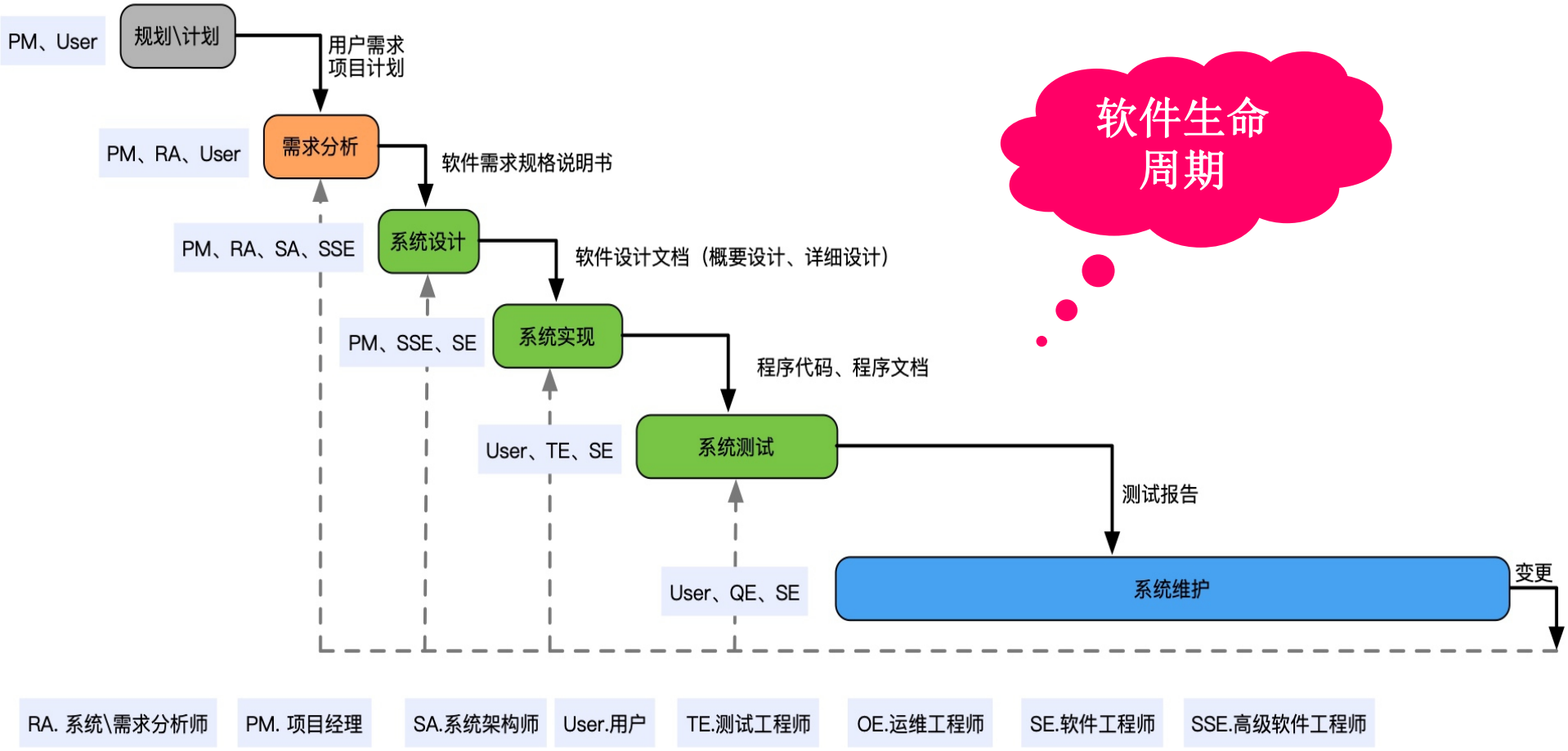
需要有一套完整的理论方法来管理和实施

借鉴
生命周期
思想

产品整体表现与市场变化趋势



7.1 软件生命周期



RA. 系统\需求分析师　PM. 项目经理　SA.系统架构师　User.用户　TE.测试工程师　OE.运维工程师　SE.软件工程师　SSE.高级软件工程师

7.1 软件生命周期

● 信息资源规划阶段

有些企业特别是大型集团企业投以巨资建立起来通信-计算机网络、各种生产自动化控制系统和经营管理信息系统，由于**缺乏高层的统筹规划和统一的信息标准**，致使设计、生产和经营管理信息不能快捷流通，信息不能共享，形成了许多**信息孤岛**，远没有发挥信息化投资的效益。

要解决上述问题，要进行信息资源规划。通过信息资源规划，可以梳理业务流程，搞清信息需求，建立企业信息标准和信息系统模型。用这些标准和模型来衡量现有的信息系统及各种应用，符合的就继承并加以整合，不符合的就进行改造优化或重新开发，从而能积极稳步地推进企业信息化建设。

7.1 软件生命周期

● 信息资源规划阶段

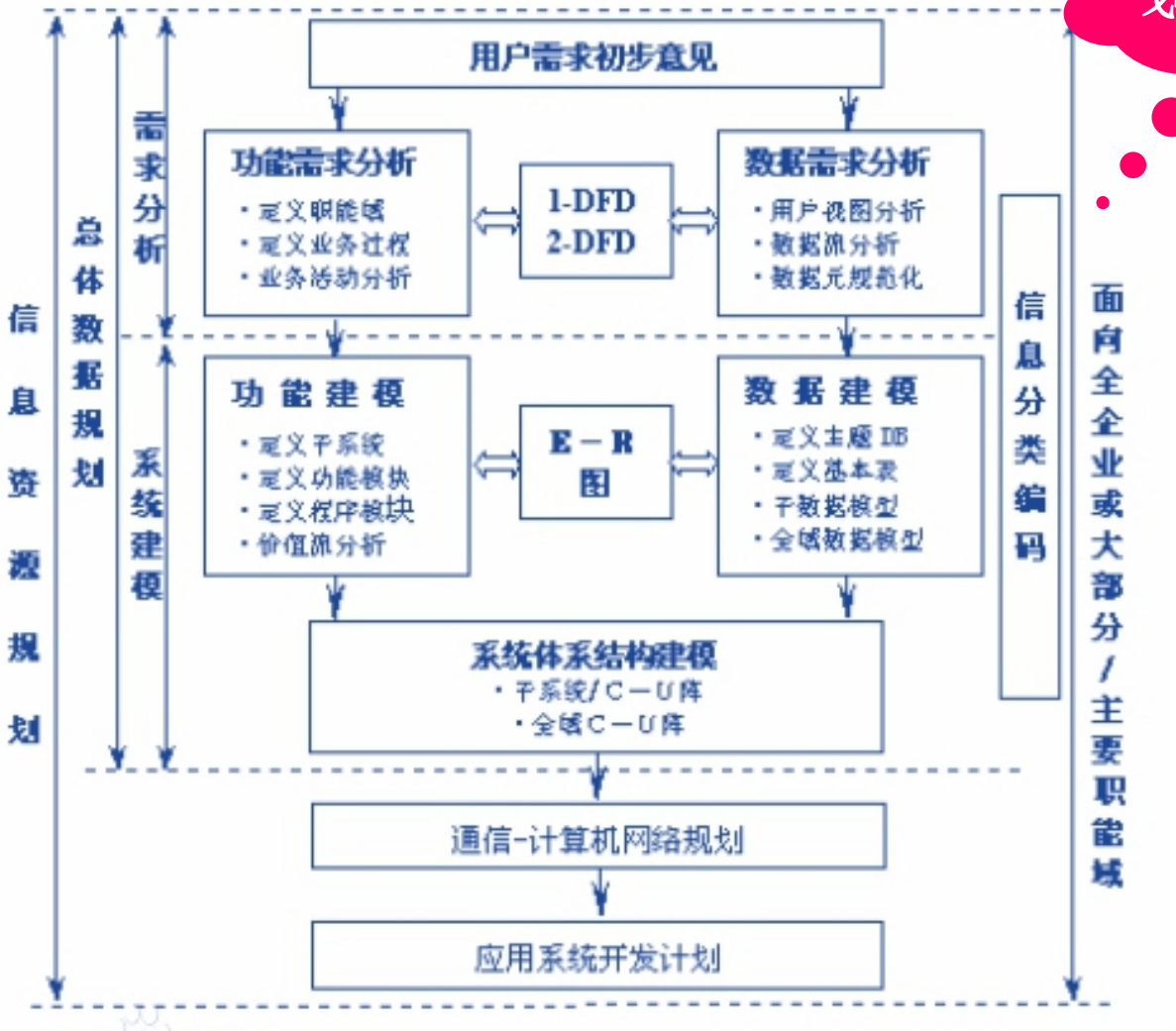
美国信息资源管理学家F.W.霍顿（F.W.Horton）和马钱德（D.A.Marchand）等人在20世纪80年代初就指出：**信息资源（Information Resources）与人力、物力、财力和自然资源一样，都是企业的重要资源**，因此，应该像管理其他资源那样管理信息资源；搞好信息资源管理的目的是通过企业内外信息流的畅通和信息资源的有效利用，来提高企业的效益和竞争力。

信息资源规划（Information Resource Planning, IRP）信息资源规划是指对企业生产经营所需要的信息，从采集、处理、传输到使用的全面规划。在企业的生产经营活动中，无时无刻不充满着信息的产生、流动和使用。要使每个部门内部，部门之间，部门与外部单位的频繁、复杂的信息流畅通，充分发挥信息资源的作用，不进行统一的、全面的规划是不可能的。

7.1 软件生命周期

- 信息资源规划阶段

信息资源规划实施框图



1.摸清有哪些信息资源？

2.信息资源如何充分利用？

3.应用系统该如何开发？

4.与单位发展战略的匹配？

7.1 软件生命周期

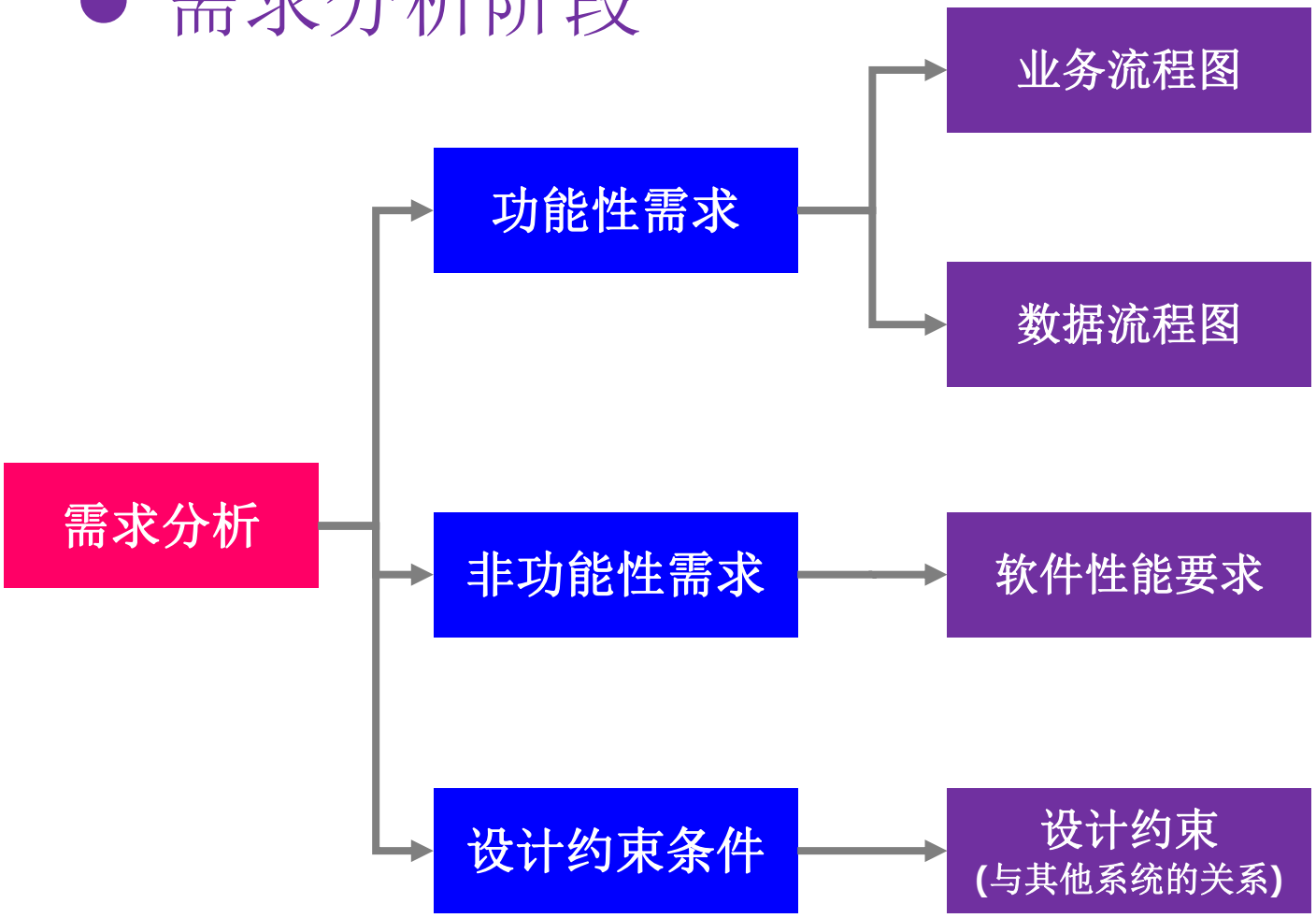
● 需求分析阶段

需求分析是软件生存周期中的一个重要环节，该阶段是分析系统在功能上需要“实现什么”（**功能性需求**），而不是考虑如何去“实现”。需求分析的目标是把用户对待开发软件提出的“要求”或“需要”进行分析与整理，确认后形成描述完整、清晰与规范的文档，确定软件需要实现哪些功能，完成哪些工作。

此外，软件的一些**非功能性需求**(如软件性能、可靠性、响应时间、可扩展性等)，**软件设计的约束条件**，运行时与其他软件的关系等也是软件需求分析的目标。

7.1 软件生命周期

- 需求分析阶段



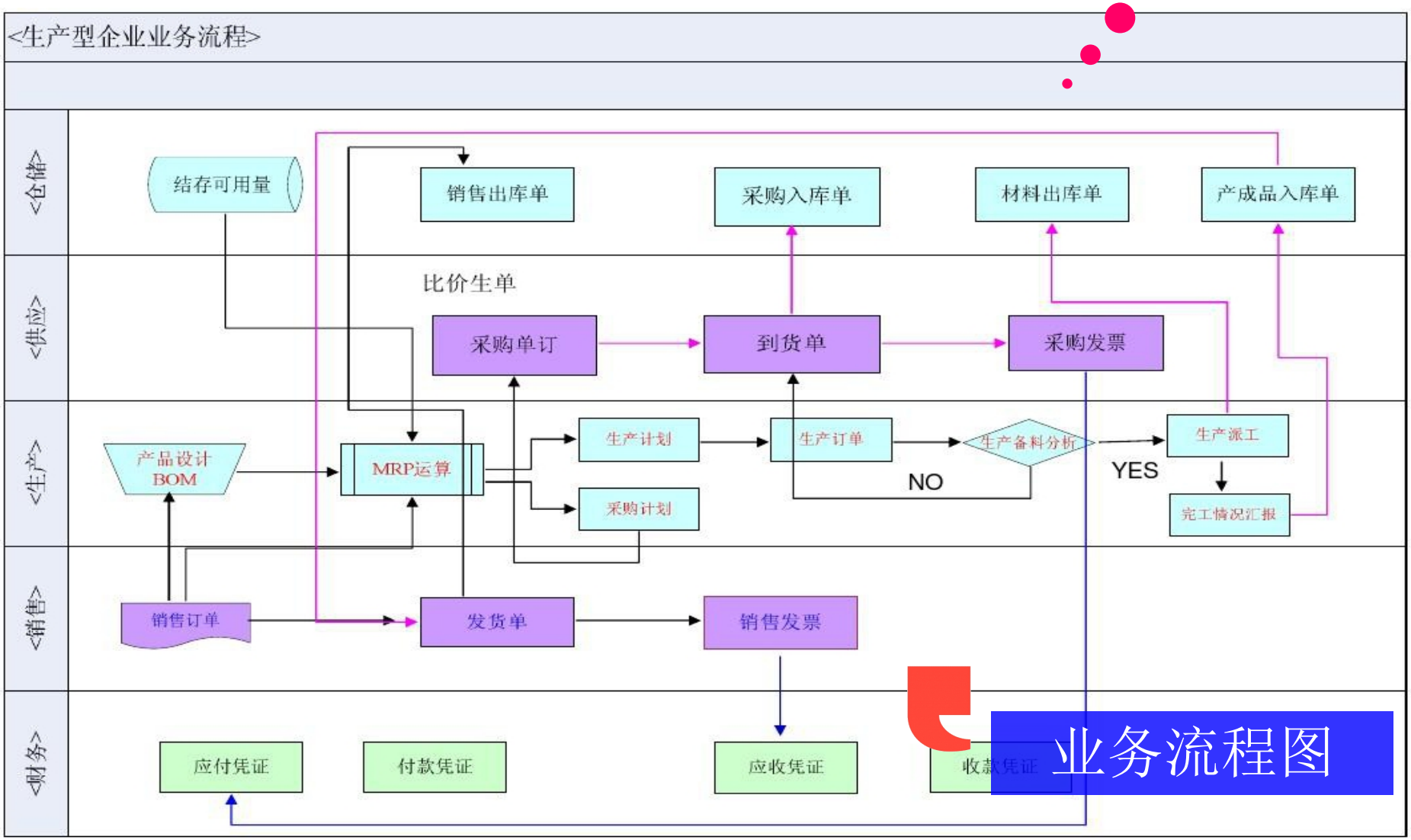
业务流程图(TFD)是一种描述管理系统内各单位、人员之间的业务关系，作业顺序和管理信息流向的图表。

数据流程图(DFD) 用一组符号来描述整个系统中信息的全貌，综合地反映出信息在系统中的流动、处理和存储情况。

7.1 软件生命周期

- 需求分析阶段

业务流程图
(流程优化和再造)

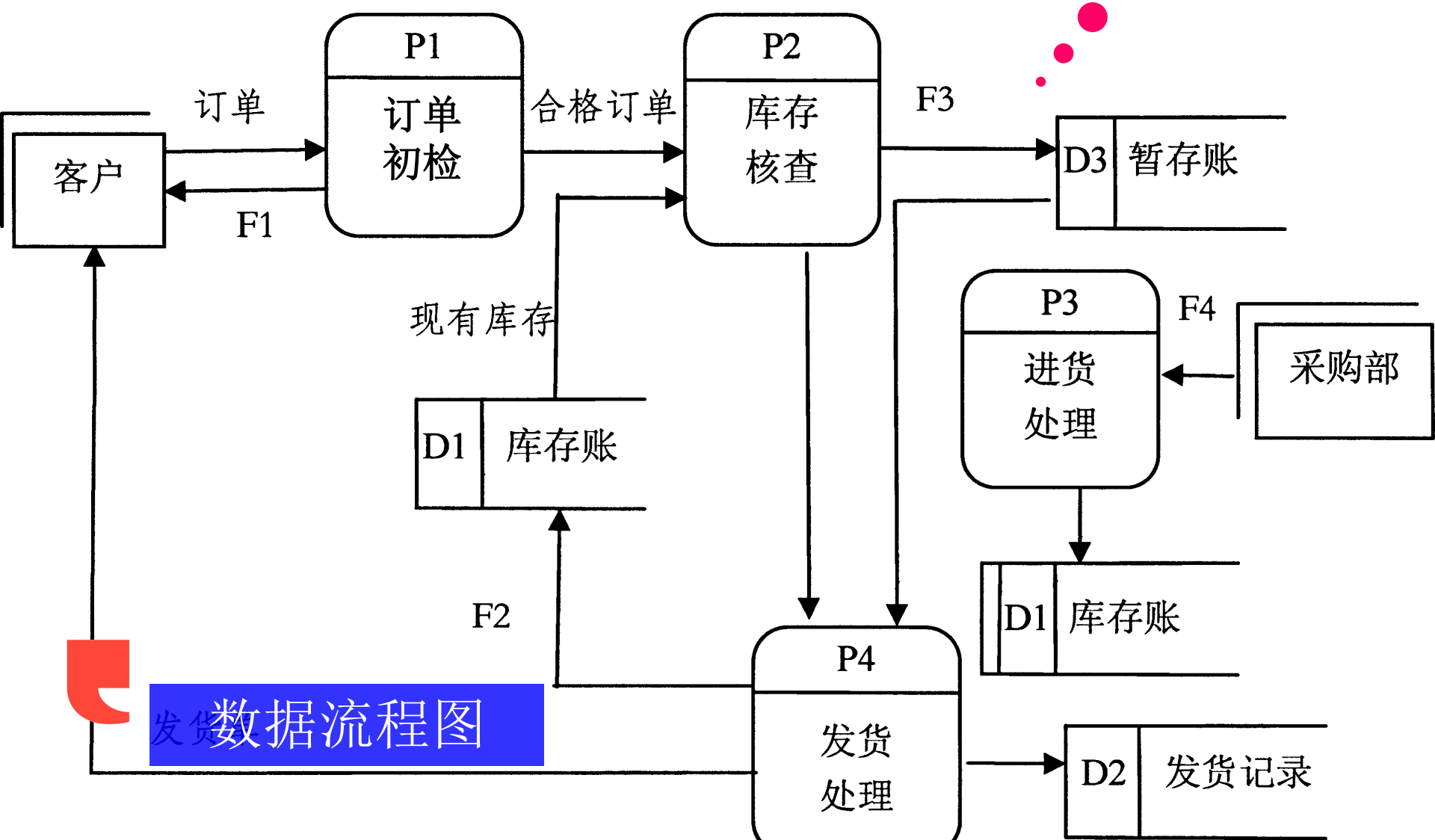


业务流程图

7.1 软件生命周期

- 需求分析阶段

数据流程图
(数据流向及格式)



数据流程图

7.1 软件生命周期

- 需求分析阶段

非功能性需求
(系统响应时间)

一次ATM查询，
你能接受的
响应时间是？

7.1 软件生命周期

- 需求分析阶段

设计约束
(服务器性能)

只有一个i7 CPU，128G内存

7.1 软件生命周期

● 系统设计阶段

系统设计是根据系统分析的结果，运用系统科学的思想和方法，在系统分析的基础上，按照系统思想和优化要求综合运用各有关学科的知识、技术和经验，通过总体研究和详细设计等环节，落实到具体工作或项目上，以创造出满足设计目的软件系统。在系统开发过程或整个系统生命周期中，系统分析着重回答“干什么”的问题，而系统设计则主要解决**“怎么干”**的问题。

系统设计内容，包括确定系统功能、设计方针和方法，产生理想系统并作出草案，通过收集信息对草案作出修正产生可选设计方案，将系统分解为若干子系统，进行子系统和总系统的详细设计并进行评价，对系统方案进行论证并作出性能效果预测。

7.1 软件生命周期

- 系统设计阶段

软件架构设计

软件架构（software architecture）是一个系统的草图，是整体布局，描述的对象是直接构成系统的抽象组件，各个组件之间的连接则明确和相对细致地描述组件之间的关系。

功能模块设计

功能模块设计是将程序划分成若干个功能模块，每个功能模块完成了一个子功能，再把这些功能模块总起来组成一个整体，以满足所要求的整个系统的功能。

数据库设计

数据库设计(Database Design)是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求（信息要求和处理要求）。

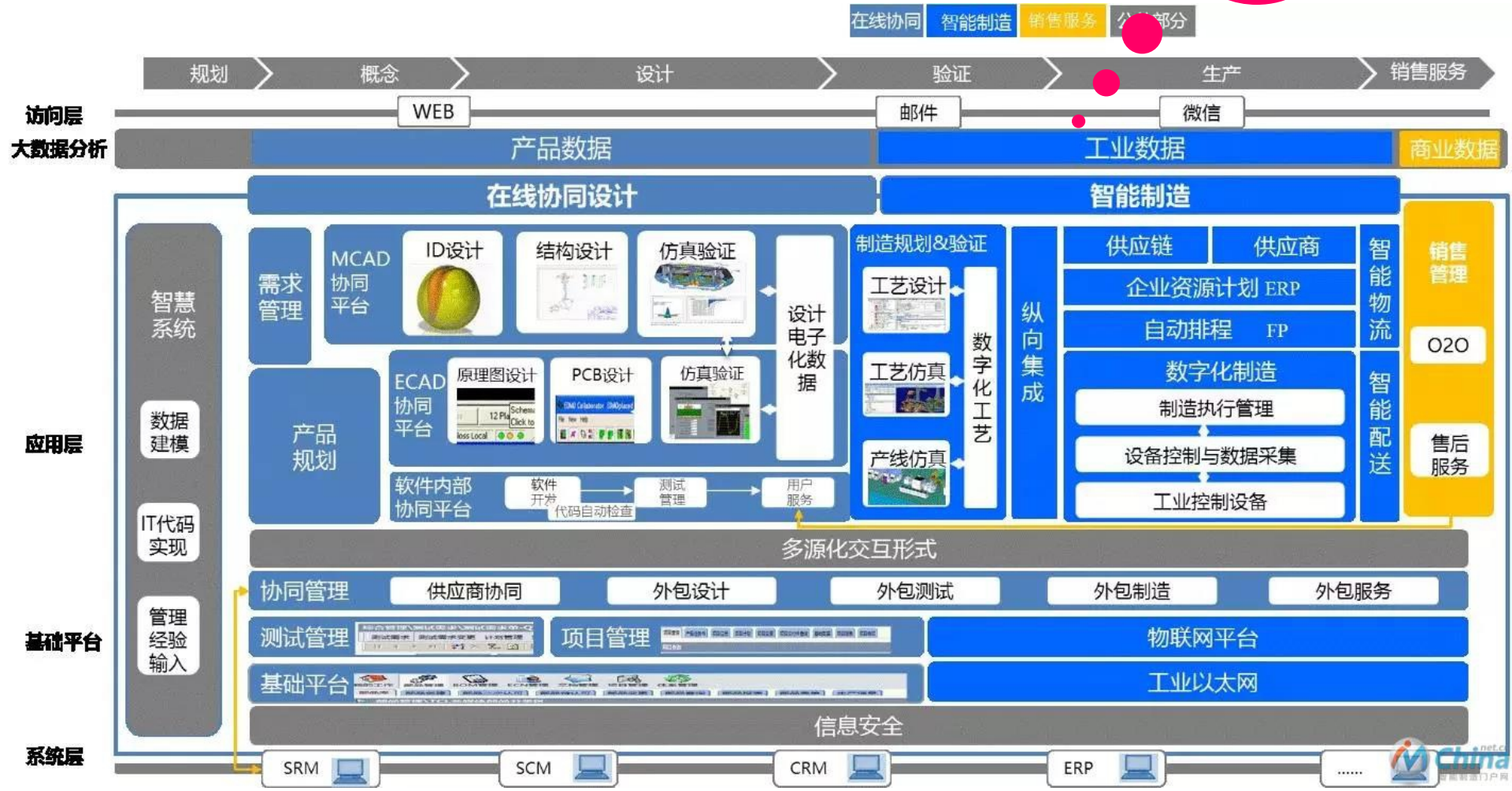
UI界面设计

UI设计（或称界面设计）是指对软件的人机交互、操作逻辑、界面美观的整体设计。

7.1 软件生命周期

系统架构设计
(离散制造系统)

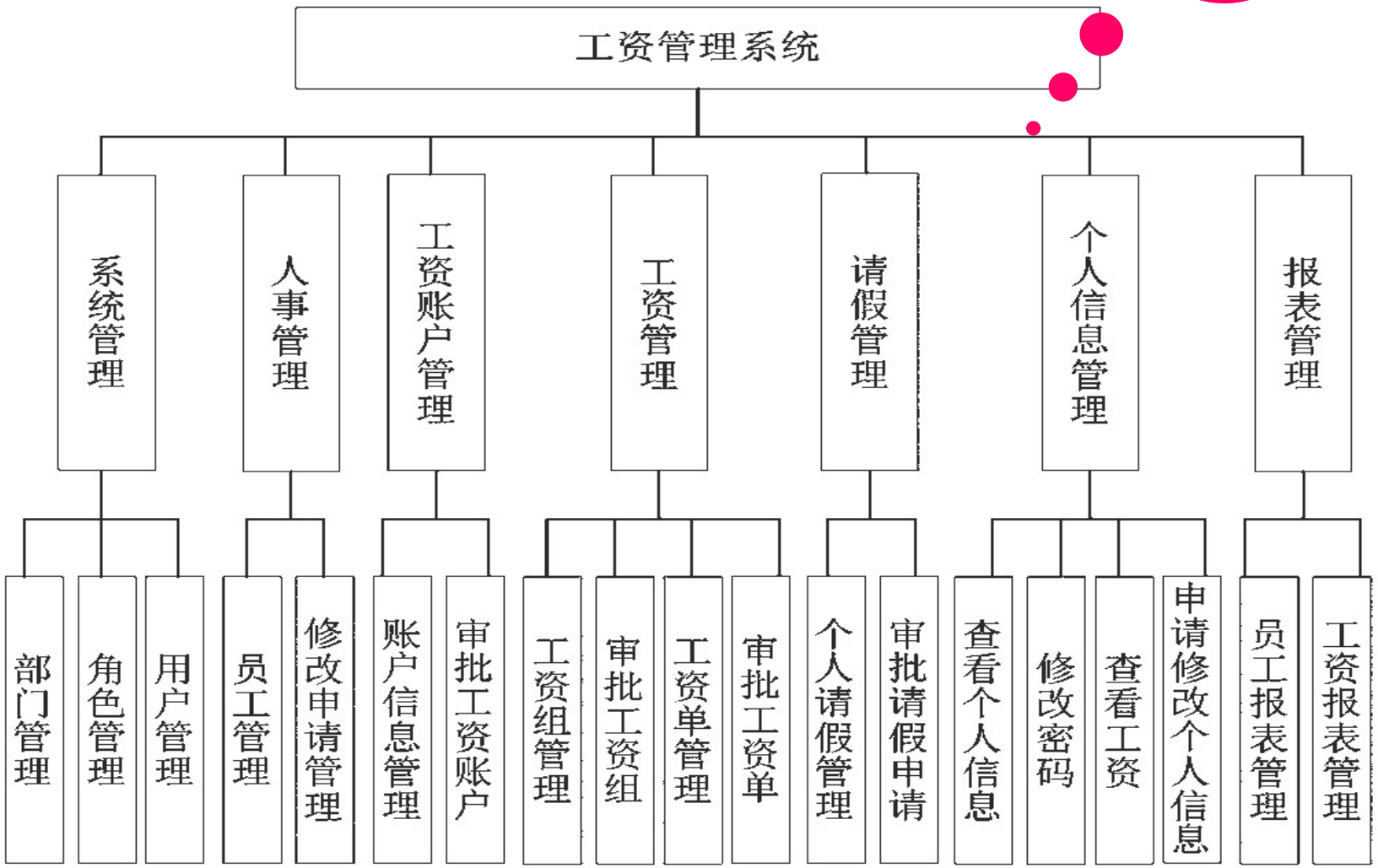
- 系统设计阶段



7.1 软件生命周期

功能模块设计
(工资管理系统)

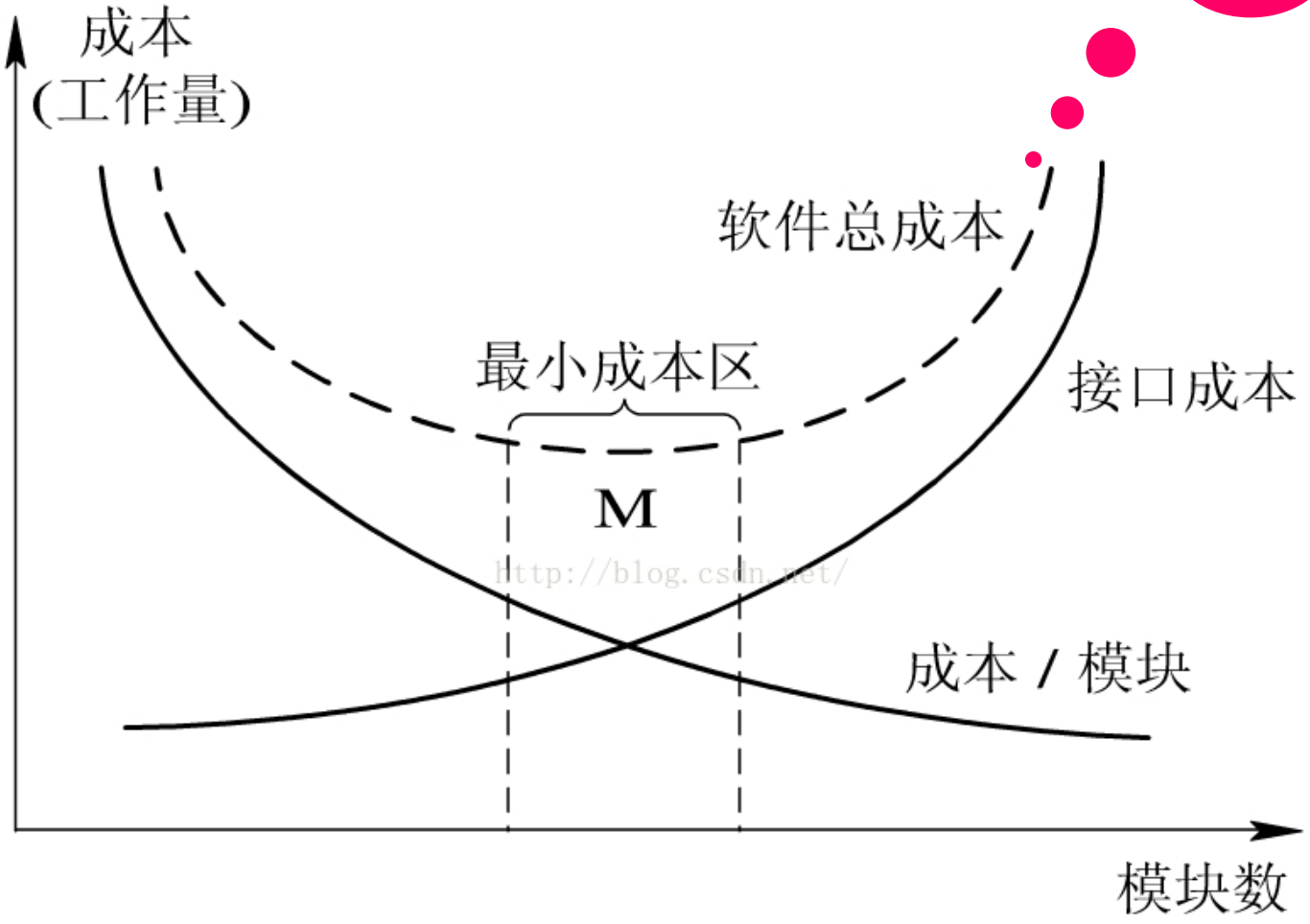
- 系统设计阶段



7.1 软件生命周期

- 系统设计阶段

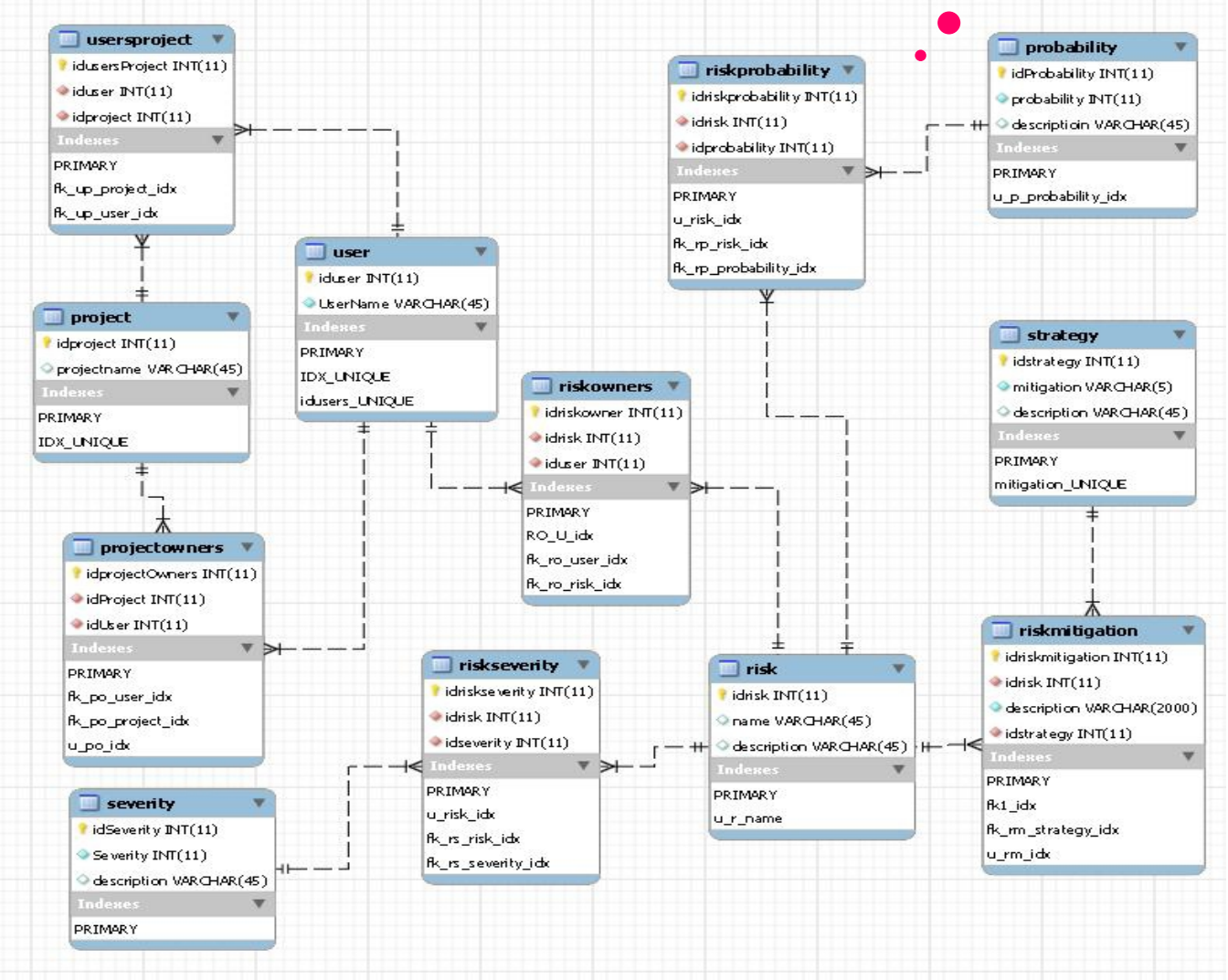
模块数与成本的关系



7.1 软件生命周期

数据库设计

- 系统设计阶段



7.1 软件生命周期

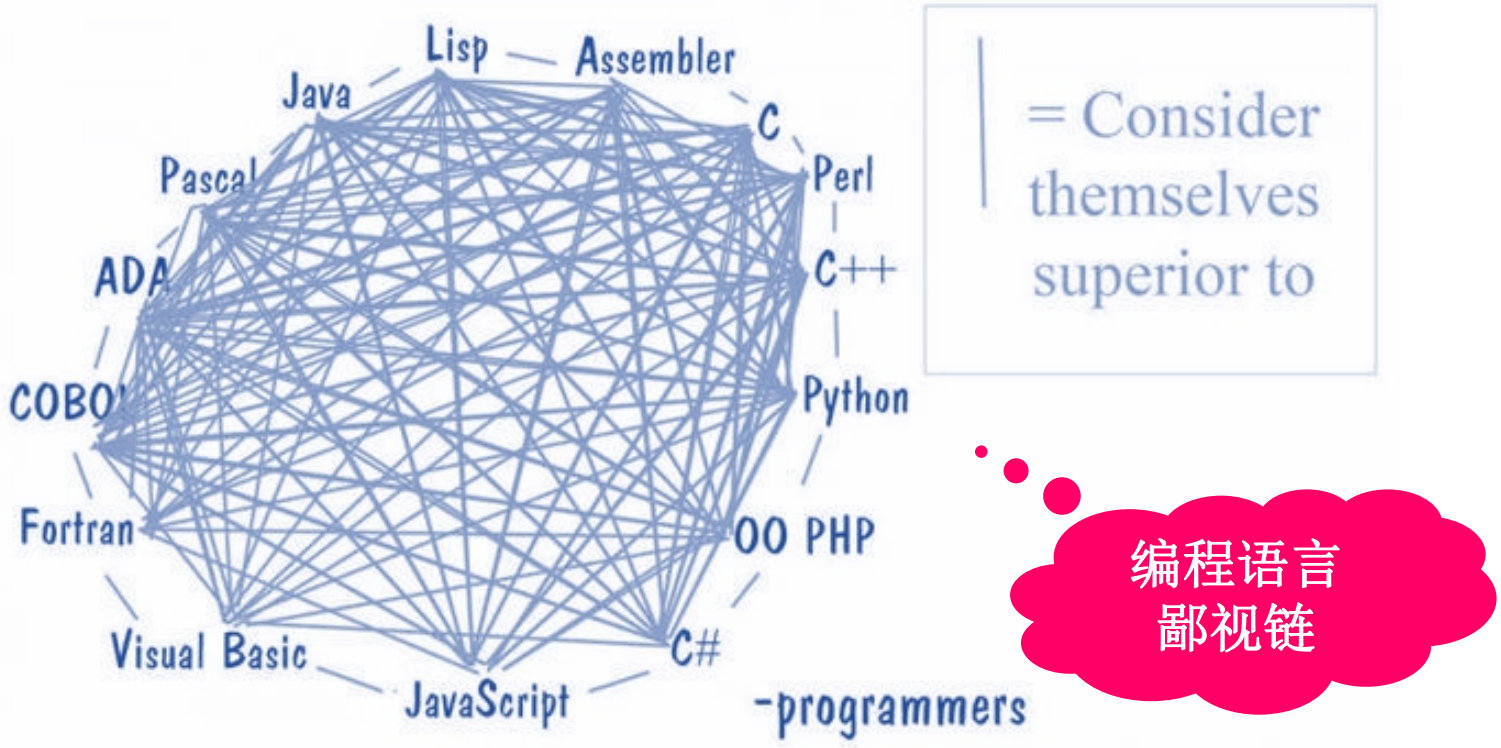
- 系统设计阶段

用户界面设计是指用户与系统之间架起一座桥梁，主要内容包括：定义界面形式、定义基本的交互控制形成、定义图形和符号、定义通用的功能键和组合键的含义及其操作内容、定义帮助策略等。

7.1 软件生命周期

● 系统实现与测试阶段

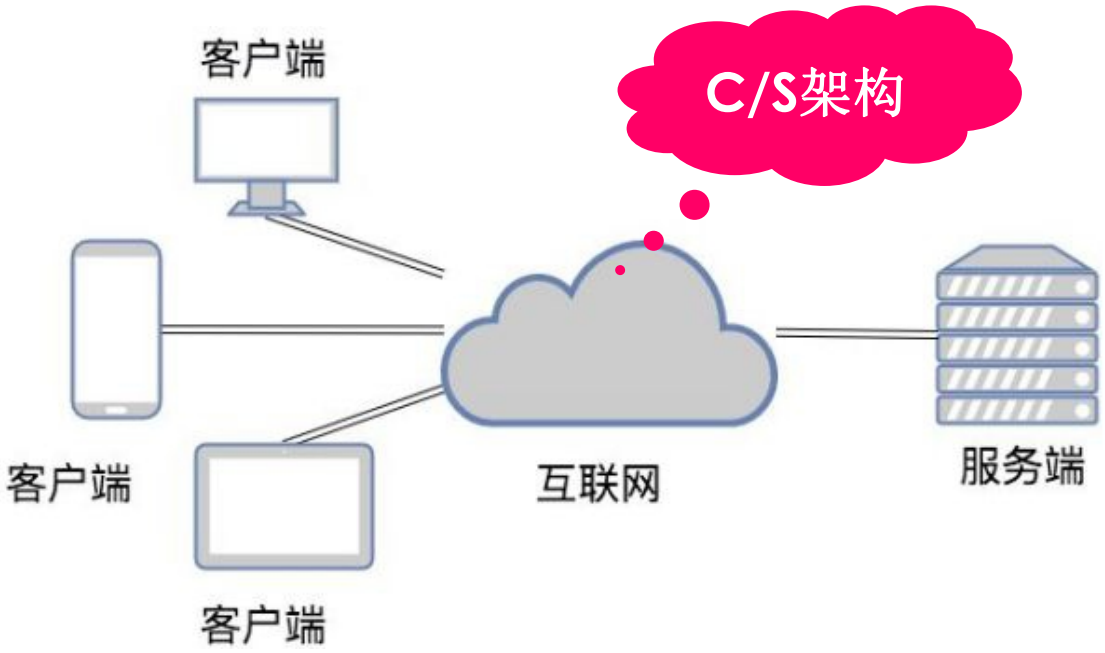
程序设计人员按照系统设计的要求和程序说明书的规定，采用某种程序设计语言来实现各个功能模块的程序编写工作。



7.1 软件生命周期

● 系统实现与测试阶段

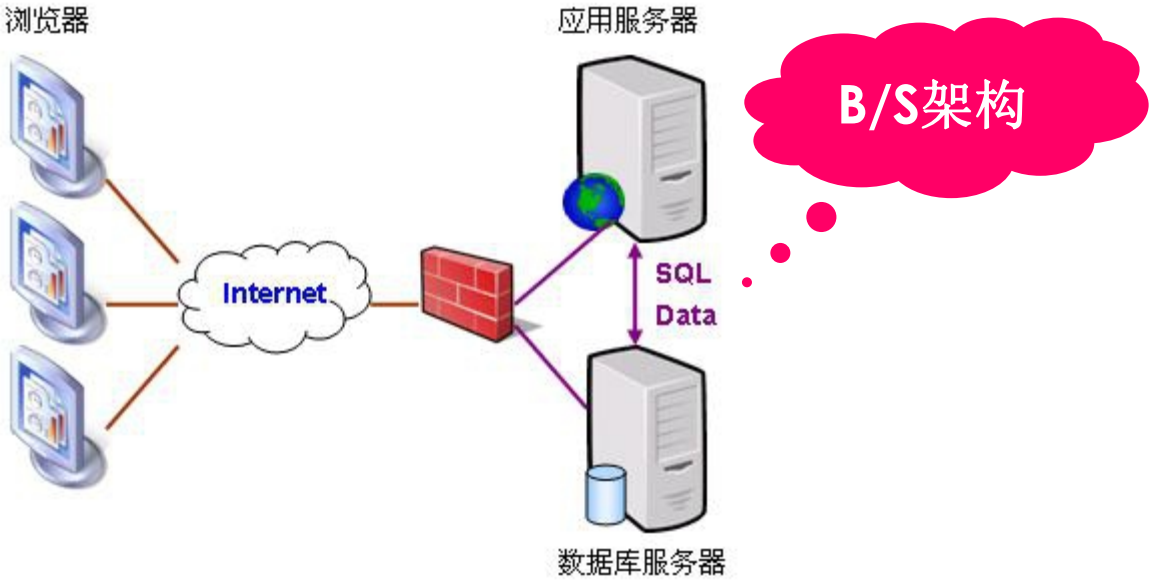
CS即Client/Server（客户机/服务器）结构，每台客户机全部须要安装相对应的客户端程序，分布功能弱并且兼容性差，不可以完成迅速部署安装与配置，C/S结构在技术上非常成熟，它的重要特征就是交互性强、拥有安全的存取形式、网络通信数量低、响应速度快、利于处置大量数据。可是这个结构的程序就是针对性开发，变更不够灵活，维护与管理的难度较大。常常只局限在小型局域网，不利于扩展。



7.1 软件生命周期

● 系统实现与测试阶段

BS即Browser/Server（浏览器/服务器）结构，就是只安装维护一个服务器（Server），而客户端选用浏览器（Browse）运行软件。B/S结构的重要特征就是分布性强、维护方便、开发简单并且共享性强、总体拥有费用低。但数据安全性问题、对服务器需要过高、数据传输速度慢、软件的个性化特征明显减少，这些缺点就是有目共睹的，难以完成传统形式下的特殊功能请求。比如通过浏览器实行大量的数据输入或实行报表的应答、专用性打印输出全部相对比较困难与不便。



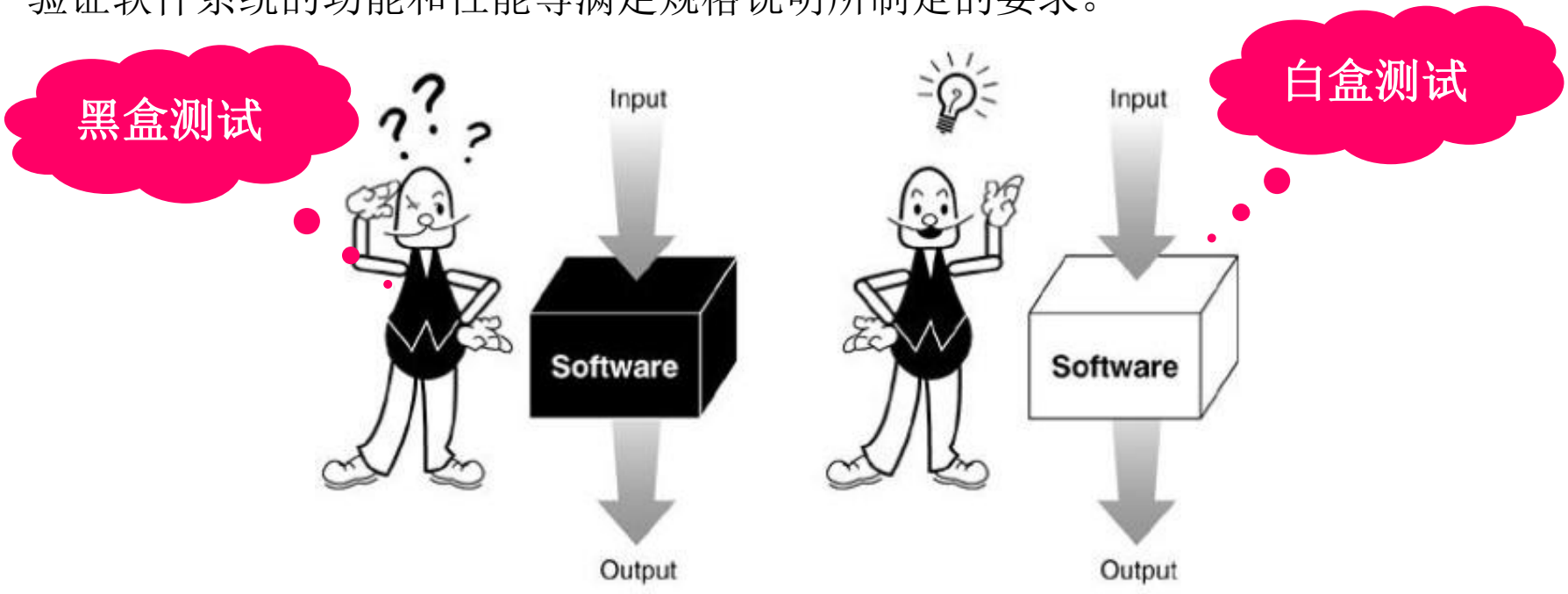
7.1 软件生命周期

- 系统实现与测试阶段

7.1 软件生命周期

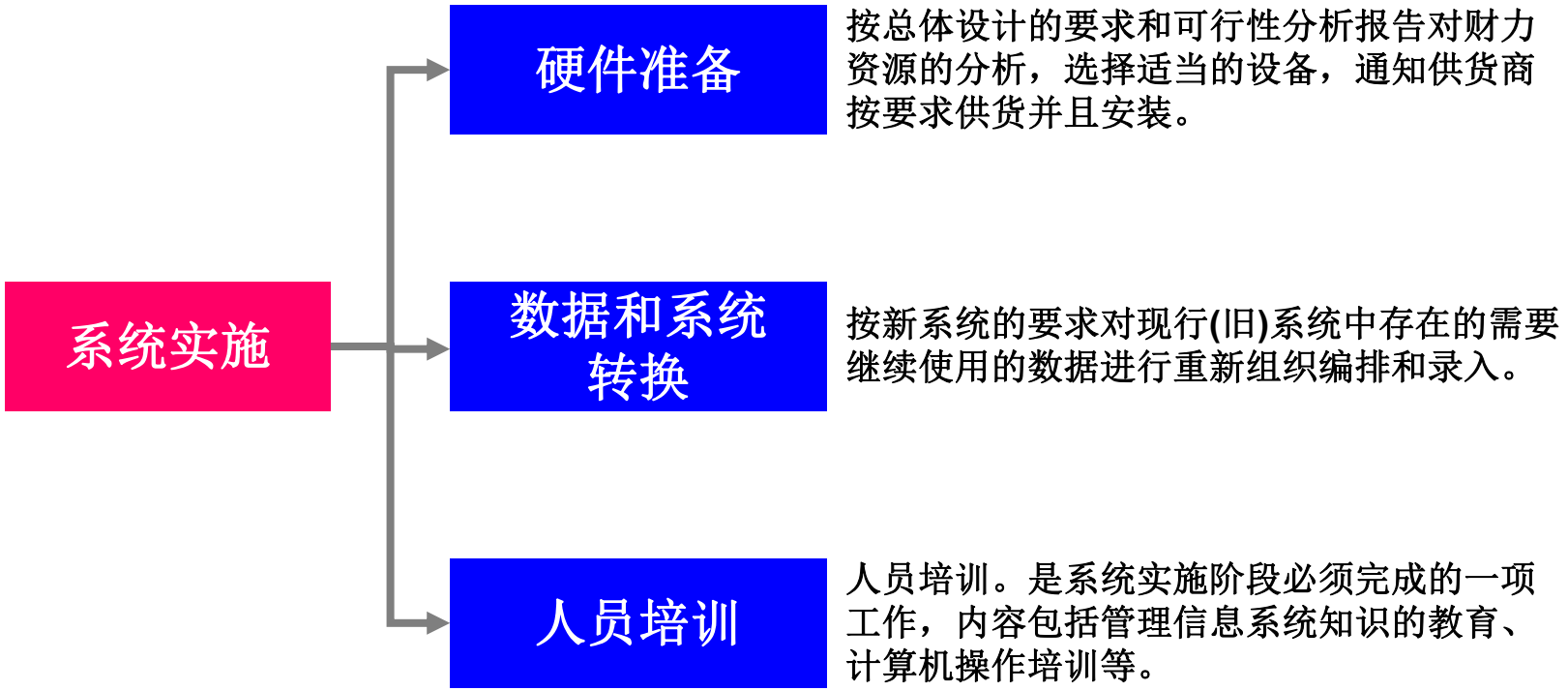
● 系统实现与测试阶段

系统测试是将已经实现好的软件系统，作为计算机系统的一个元素，与计算机硬件、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的集成测试和确认测试，目标是：通过与系统的需求规格说明进行比较，检查软件是否存在与系统规格说明不符合或与之矛盾的地方，从而验证软件系统的功能和性能等满足规格说明所制定的要求。



7.1 软件生命周期

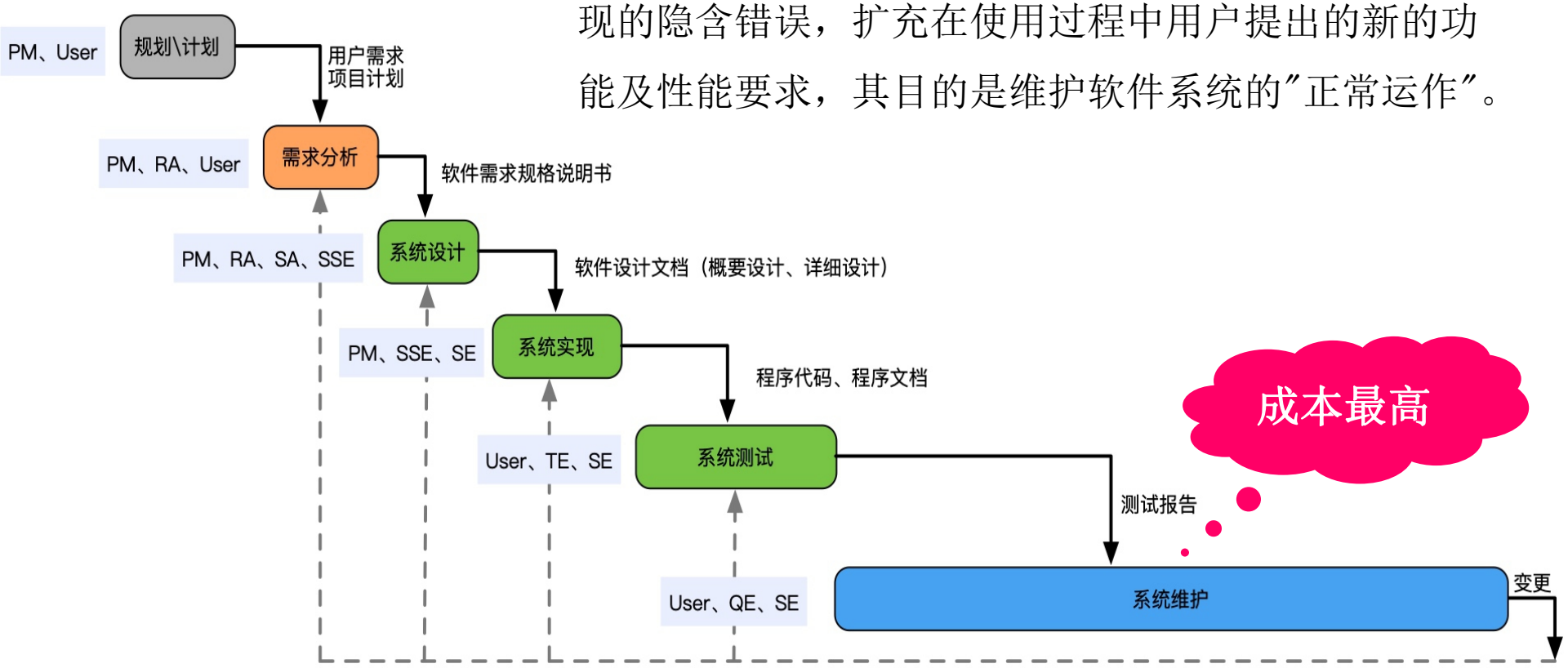
- 系统实施阶段



7.1 软件生命周期

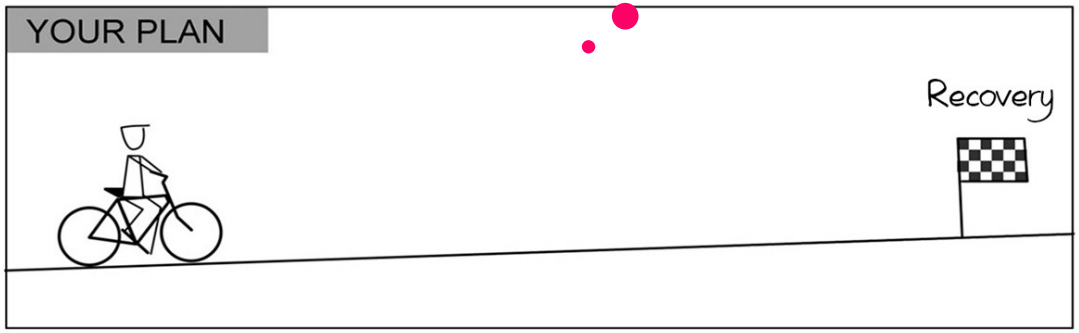
● 系统维护阶段

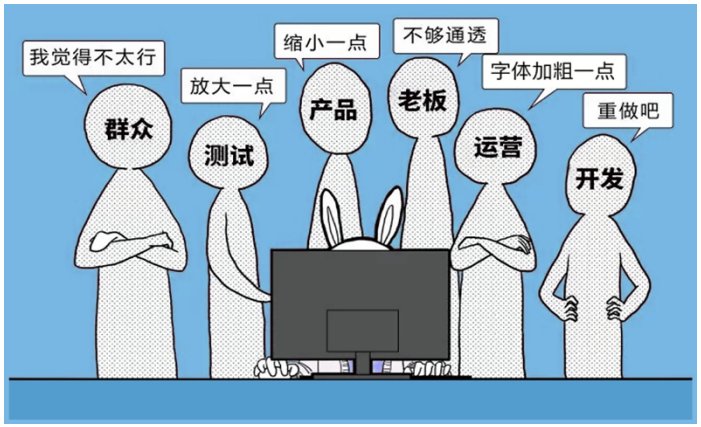
系统维护的任务是改正软件系统在使用过程中发现的隐含错误，扩充在使用过程中用户提出的新的功能及性能要求，其目的是维护软件系统的"正常运作"。

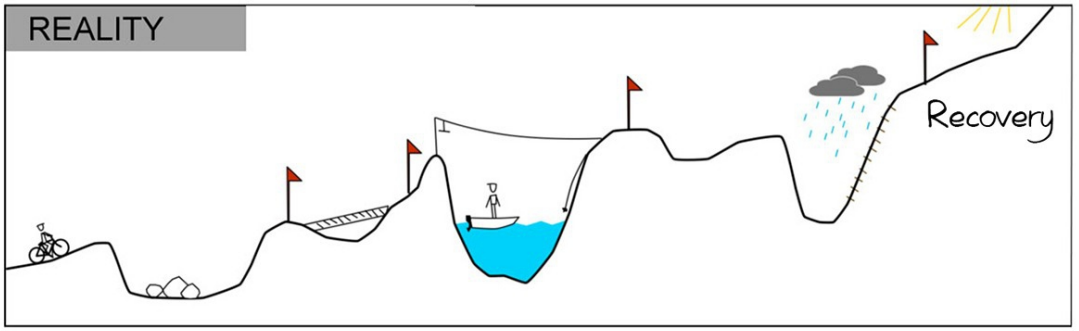


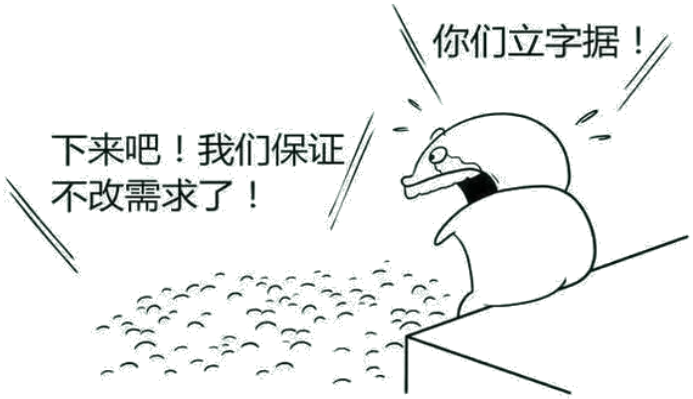
RA. 系统\需求分析师　PM. 项目经理　SA.系统架构师　User.用户　TE.测试工程师　OE.运维工程师　SE.软件工程师　SSE.高级软件工程师

设计与实现之间的差距









软件系统大且复杂，即使经过“生命周期”很难一次成型，牵一发而动全身的情况下，如何修改、完善、优化，使其能最大程度满足需求？也需要一套理论来支撑。

7.2 软件开发模式

软件生存周期一般划分为：制定计划，需求分析，设计，编码实现，测试，运行维护等几阶段，称为软件的生命周期。用不同的方式将软件生命周期中的所有活动组织起来形成一定的结构框架来指导软件开发，从而形成不同的**软件开发模式（模型）**。

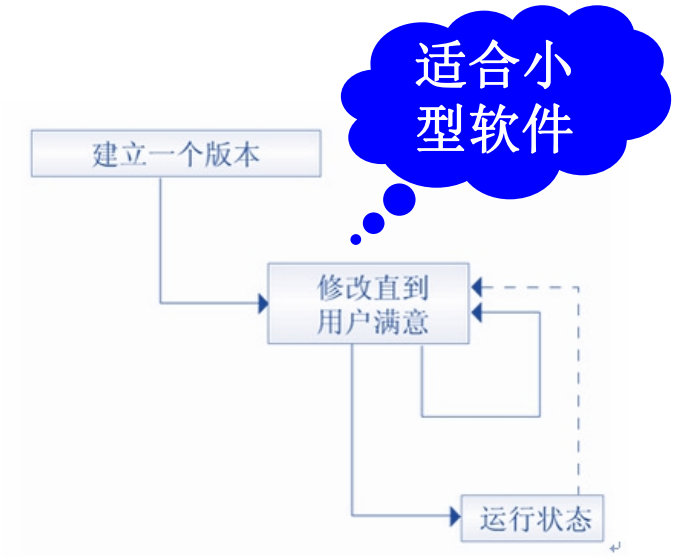
不同的产品需求

总结出来的经验模式

7.2 软件开发模式

● 边做边改模型（Build-and-Fix Model）

在这种模型中，既没有规格说明，也没有经过设计，软件随着客户的需要一次又一次地不断被修改。开发人员**拿到项目立即根据需求编写程序**，调试通过后生成软件的第一个版本。在提供给用户使用后，如果程序出现错误，或者用户提出新的要求，开发人员重新修改代码，直到用户和测试满意为止。

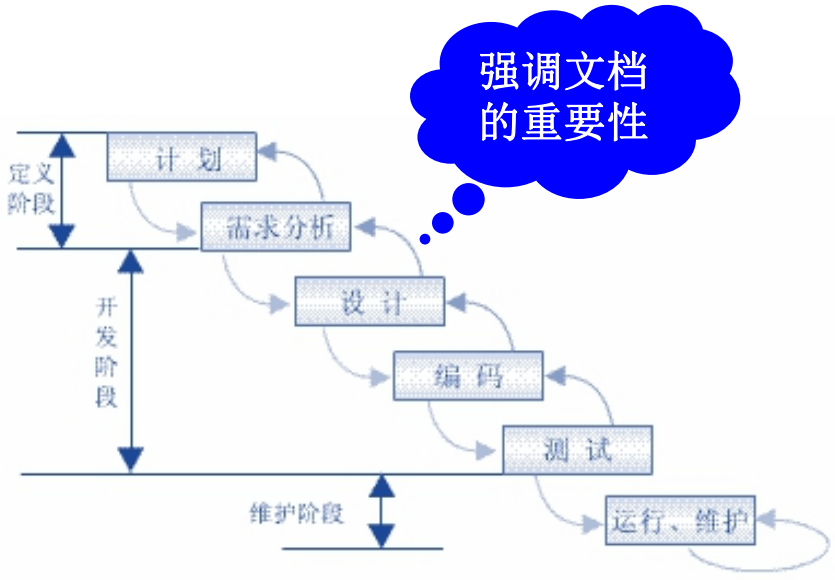


1）缺少规划和设计环节，软件的结构随着不断的修改越来越糟，导致无法继续修改；
2）忽略需求环节，给软件开发带来很大的风险；
3）没有考虑测试和程序的可维护性，也没有任何文档，软件的维护十分困难。

7.2 软件开发模式

● 瀑布模型（Waterfall Model）

1970年温斯顿·罗伊斯提出了著名的“瀑布模型”。瀑布模型将软件生命周期划分：制定计划、需求分析、软件设计、程序编写、软件测试和运行维护等六个基本活动，并且规定了它们**自上而下、相互衔接的固定次序**，如同瀑布流水，逐级下落。在瀑布模型中，软件开发的各项活动严格按照线性方式进行，当前活动接受上一项活动的工作结果，实施完成所需的工作内容。当前活动的工作结果需要进行验证，如验证通过，则该结果作为下一项活动的输入，继续进行下一项活动，否则返回修改。



1）各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量；

2）由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发的风险；

3）早期的错误可能要等到开发后期的测试阶段才能发现，进而带来严重的后果；

4）各个软件生命周期衔接花费时间较长，团队人员交流成本大；

5）瀑布式方法在需求不明并且在项目进行过程中可能变化的情况下基本是不可行的。

7.2 软件开发模式

● 迭代模型（stagewise model）

在迭代模型中，整个开发被组织为**一系列的短小的、固定长度（如3周）的小项目**，每一次迭代都包括了需求分析、设计、实现与测试，开发工作可以在需求被完整地确定之前启动，并在一次迭代中完成系统的一部分功能或业务逻辑的开发工作，再通过客户的反馈来细化需求，并开始新一轮的迭代。

1）降低了在一个增量上的开支风险。如果开发人员重复某个迭代，那么损失只是这一个开发有误的迭代的花费；

2）降低了产品无法按照既定进度进入市场的风险。通过在开发早期就确定风险，可以尽早来解决而不至于在开发后期匆匆忙忙；

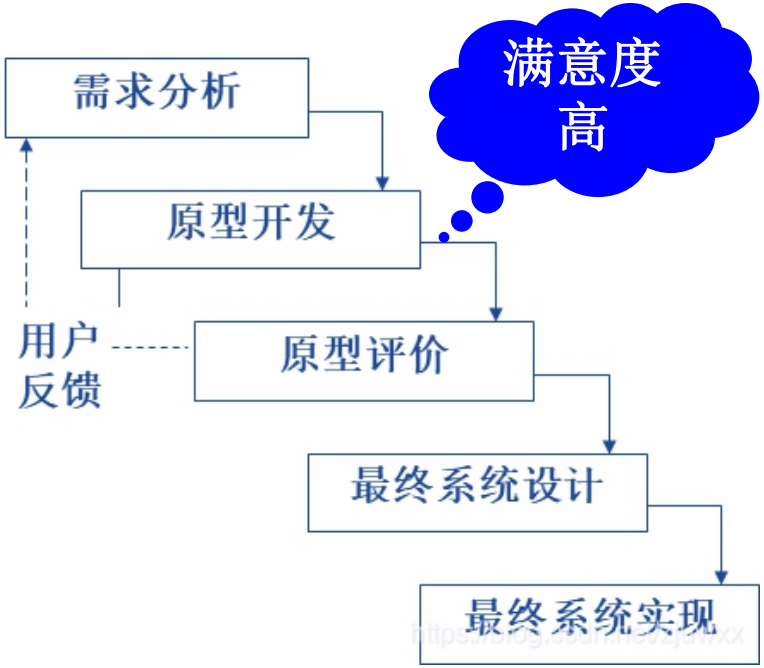
3）加快了整个开发工作的进度。因为开发人员清楚问题的焦点所在，他们的工作会更有效率；

4）由于用户的需求并不能在一开始就作出完全的界定，它们通常是在后续阶段中不断细化的。

7.2 软件开发模式

● 快速原型模型（Rapid Prototype Model）

快速原型模型的第一步是快速建立一个能反映用户主要需求的软件模型。让用户在计算机上使用它，通过实际操作了解目标系统概貌。一旦用户认为这个原型系统确实能满足他们的需求，开发人员便可据此书写软件需求说明，并根据这份文档开发可以满足用户需求的软件产品。

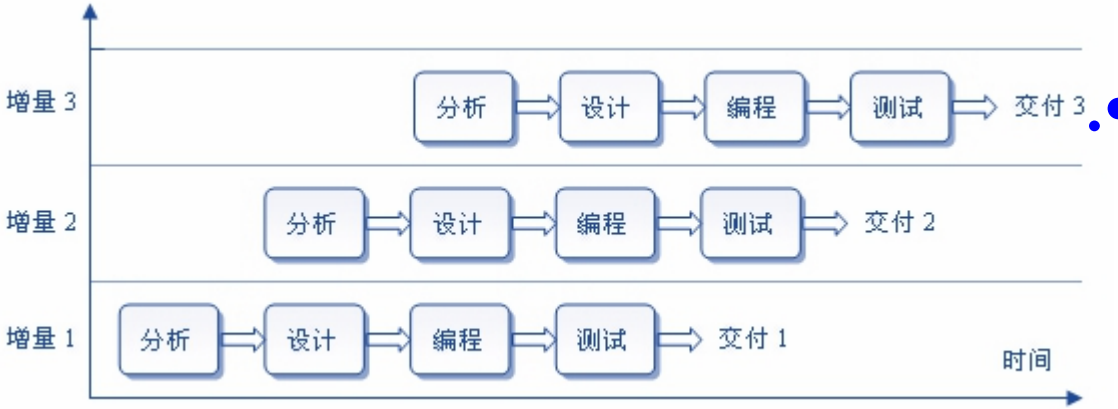


1）减少由于软件需求不明确带来的开发风险；
2）通过逐步调整原型使其满足客户的要求，开发人员可以确定客户的真正需求是什么；
3）在确定原型的基础上，容易开发出客户满意的软件产品。

7.2 软件开发模式

● 增量模型（Incremental Model）

增量模型在各个阶段并不交付一个可运行的完整产品，而是交付满足客户需求的一个子集的可运行产品。整个产品被分解成若干个构件，开发人员逐个构件地交付产品，这样做的好处是软件开发可以较好地适应变化，客户可以不断地看到所开发的软件，从而降低开发风险。



手机APP

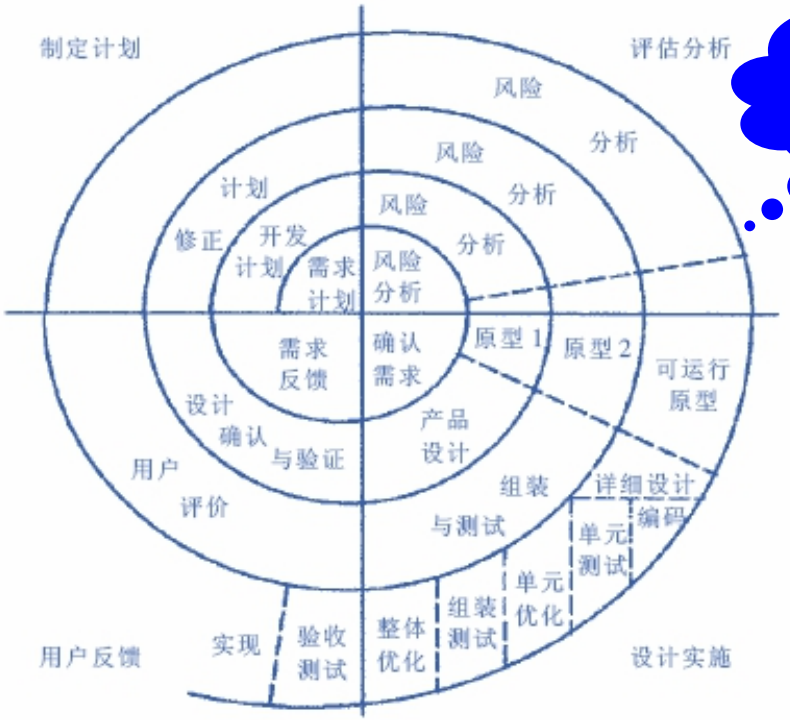
1）由于各个构件是逐渐并入已有的软件体系结构中的，所以加入构件必须不破坏已构造好的系统部分，这需要软件具备开放式的体系结构；

2）在开发过程中，需求的变化是不可避免的。增量模型的灵活性可以使其适应这种变化的能力大大优于瀑布模型和快速原型模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。

7.2 软件开发模式

● 螺旋模型（Spiral Model）

螺旋模型是由瀑布模型与快速模型所结合而来，将风险分析特别拿出来进行强调，在编写程序时，不一定需要将所有事情都了解的明明白白，只需要知道最重要的功能即可，然后我们的目标便是先将它给完成，听取客户意见去进行修改，然后开始进入下一个阶段，不断循环直到最终成品产生。



适合大型复杂系统

1）螺旋模型强调风险分析，但要求许多客户接受和相信这种分析，并做出相关反应是不容易的，因此，这种模型往往适应于内部的大规模软件开发。

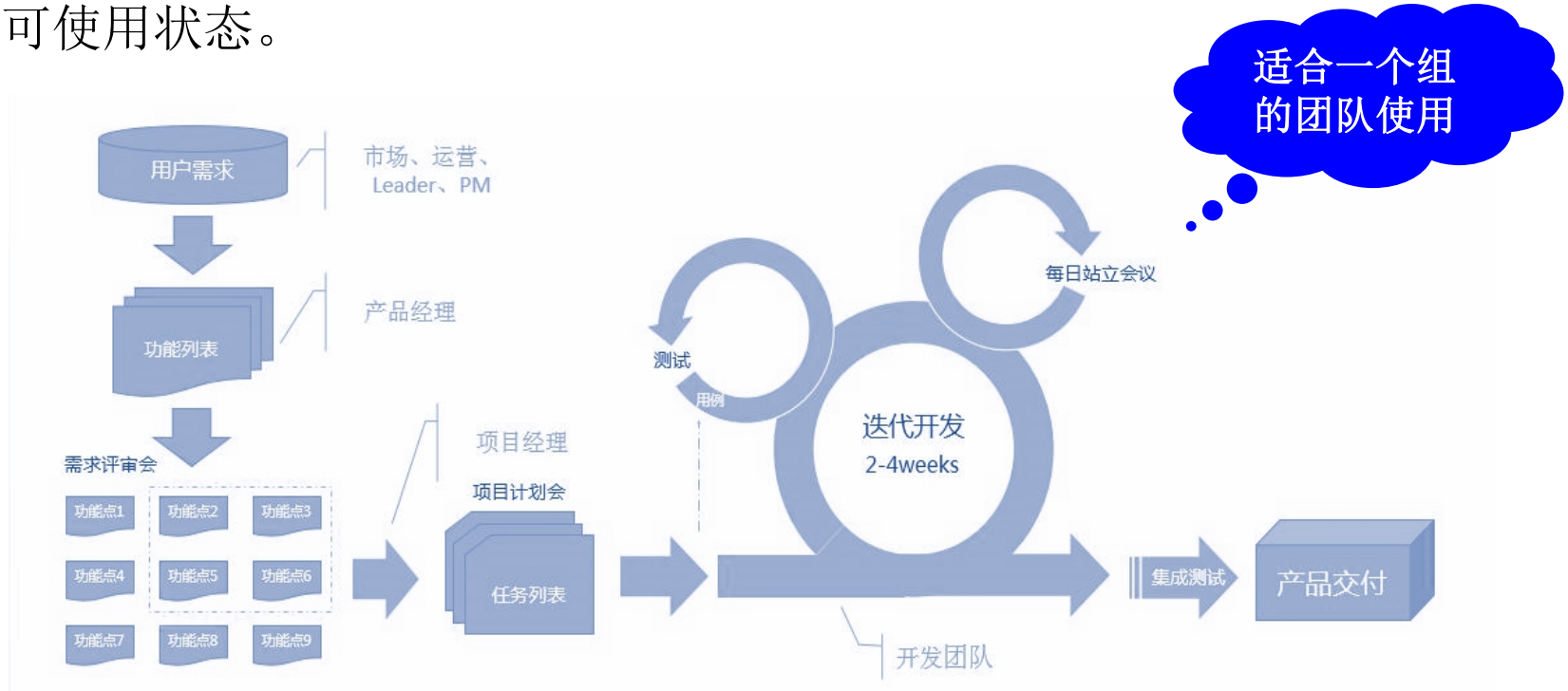
2）如果执行风险分析将大大影响项目的利润，那么进行风险分析毫无意义，因此，螺旋模型只适合于大规模软件项目。

3）软件开发人员应该擅长寻找可能的风险，准确地分析风险，否则将会带来更大的风险

7.2 软件开发模式

● 敏捷开发模型(Agile development)

在敏捷开发中，软件项目的构建被切分成多个子项目，各个子项目的成果都经过测试，具备集成和可运行的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。



7.2 软件开发模式

模型名称	技术特点	适用范围
瀑布模型	简单，分阶段，阶段间存在因果关系，各个阶段完成后都有评审，允许反馈，不支持用户参与，要求预先确定需求	需求易于完善定义且不易变更的软件系统
快速原型模型	不要求需求预先完备定义，支持用户参与，支持需求的渐进式完善和确认，能够适应用户需求的变化	需求复杂、难以确定、动态变化的软件系统
增量模型	软件产品是被增量式地一块块开发的，允许开发活动并行和重叠	技术风险较大、用户需求较为稳定的软件系统
迭代模型	不要求一次性地开发出完整的软件系统，将软件开发视为一个逐步获取用广需求、完善软件产品的过程	需求难以确定、不断变更的软件系统
螺旋模型	结合瀑布模型、快速原型模型和迭代模型的思想，并引进了风险分析活动	需求难以获取和确定、软件开发风险较大的软件系统

7.2 软件开发模式

最新研究进展

序号	刊物名称	刊物全称	地址
1	FSE/ESEC	ACM SIGSOFT Symposium on the Foundation of Software Engineering/ European Software Engineering Conference	http://dblp.uni-trier.de/db/conf/sigsoft/
2	ASE	International Conference on Automated Software Engineering	http://dblp.uni-trier.de/db/conf/kbse/
3	ICSE	International Conference on Software Engineering	http://dblp.uni-trier.de/db/conf/icse/
4	ISSTA	International Symposium on Software Testing and Analysis	http://dblp.uni-trier.de/db/conf/issta/