

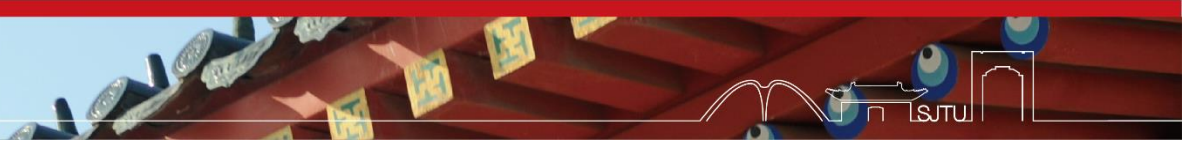


Chap7 航空电子软件工程环境



上海交通大學

SHANGHAI JIAO TONG UNIVERSITY



Civil Avionics Systems

Chap 7 Software Engineering and **Management**

Prof. Xiao Gang



Email: Xiaogang@sjtu.edu.cn

Office: Aerospace Room.A432

Tel/Fax:021-34206192

Advanced Avionics and Intelligent Information Laboratory

<http://www.avionics.icoc.in/>



1

航电系统的开发要求-ARP4754

2

民用航空适航认证标准-DO-178B

3

航空电子软件过程

4

基于模型驱动的航电软件开发方法 – Harmony/ESW



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY



在当代采用高新技术的飞行器研制中，**系统工程**与**软件工程**是密不可分的。这不仅是因为现代飞行器中装备了大量的软件，而且由于在飞行器研制中纯粹的软件开发和开发组织是少见的。软件多是与系统一起开发，并集成到系统中。因此迫切需要将软件工程，乃至软件采办与系统工程集成在一起。

航空电子软件是指应用在航空电子设备中实现**数据采集**、**数据解算**、**自动化控制**以及**数据交互**等功能的计算机软件。



软件工程 VS 航空软件

1 软件工程概述

2 软件的定义及可行性研究

3 需求分析

4 概要设计

5 详细设计

6 面向对象概念和Rose建模技术

7 面向对象的分析与设计

8 编码

9 软件测试

10 软件维护

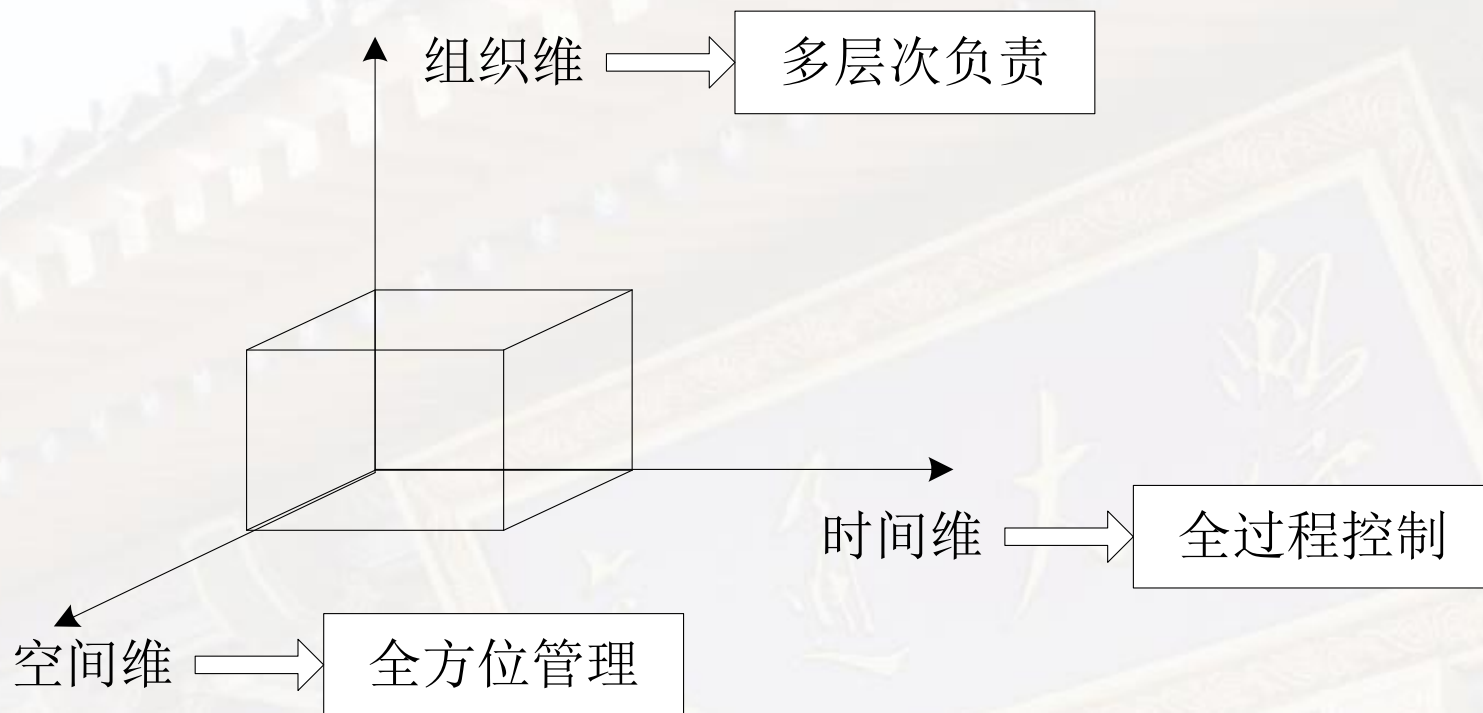
11 软件项目管理



■ 软件质量特性

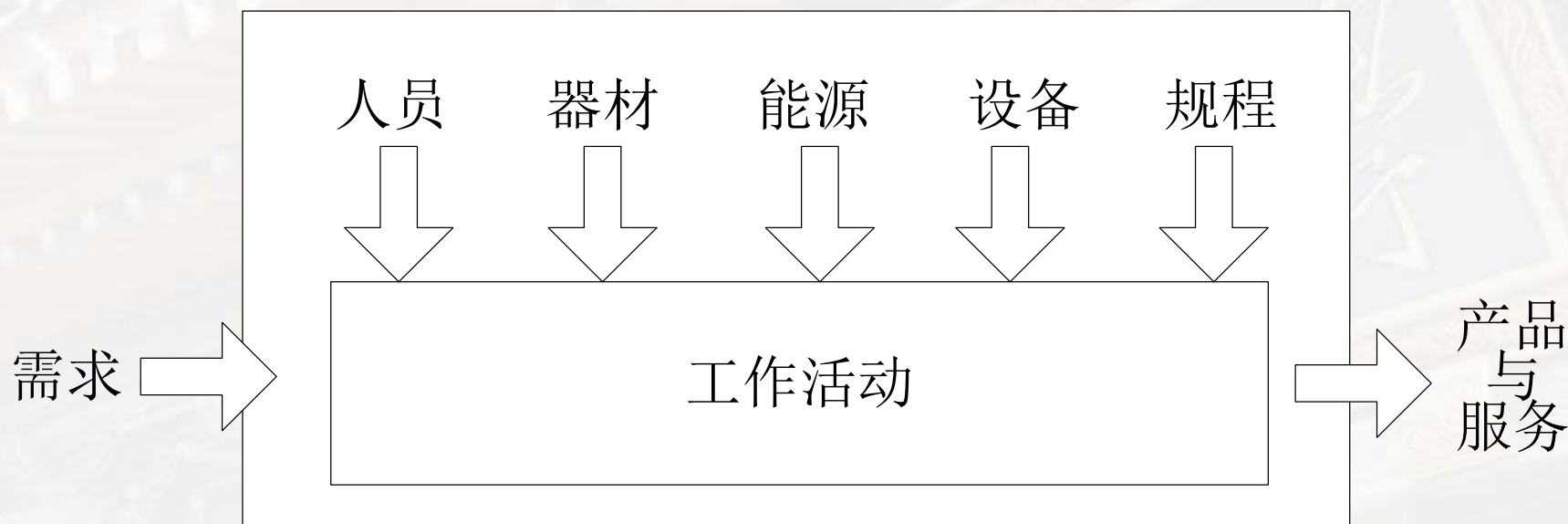


- 软件的质量模型
- 软件质量的Pareto原理
- 软件质量与软件工程化



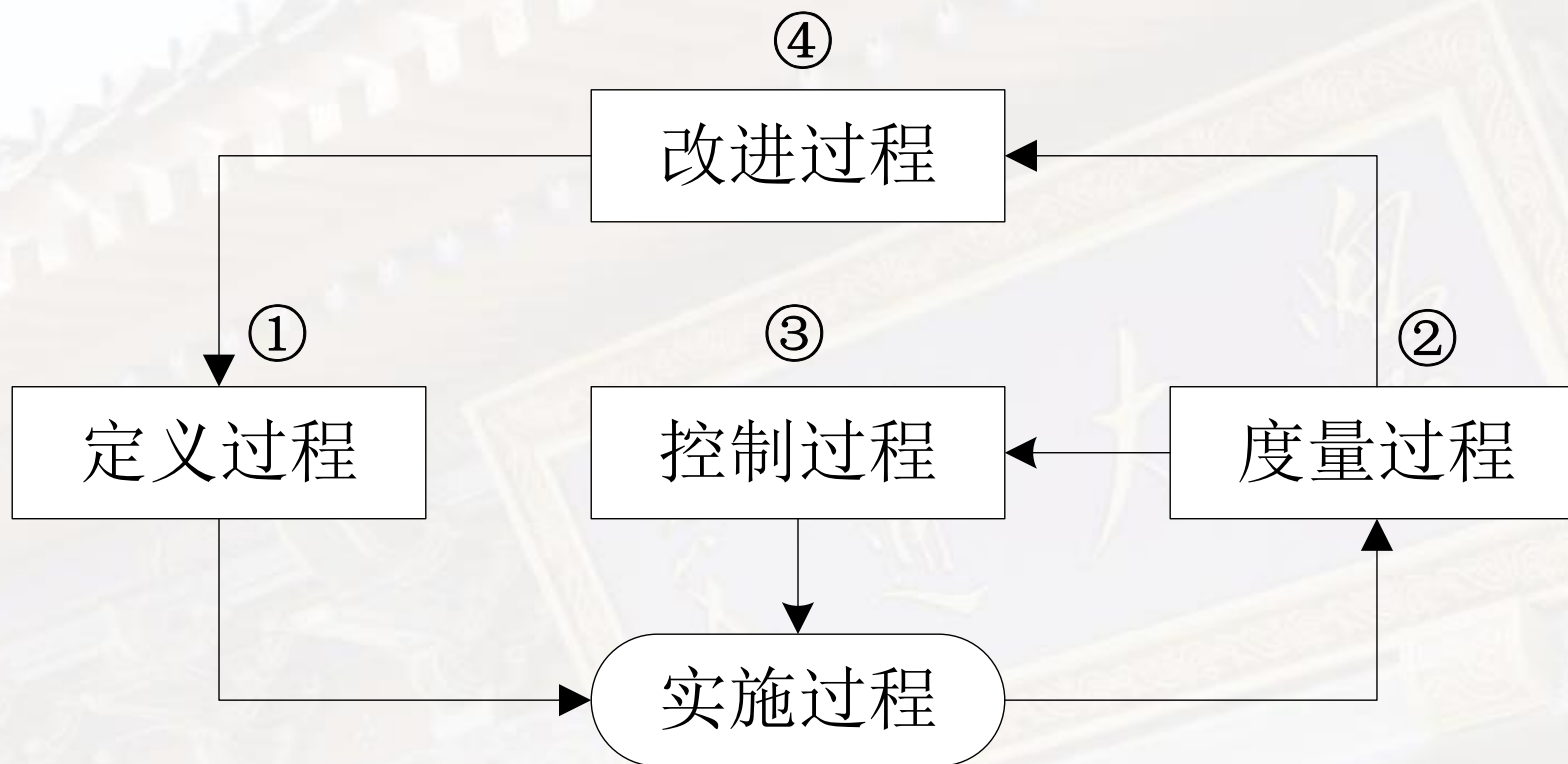
现代软件开发与管理的三维模型

- 过程是指通过人员、设备、器材和规程的交互作用，以期提供一个规定的服务，或生产一个规定的产品。



时间维——对软件生存期的全过程控制

■ 软件过程管理的关键活动



时间维——对软件生存期的全过程控制

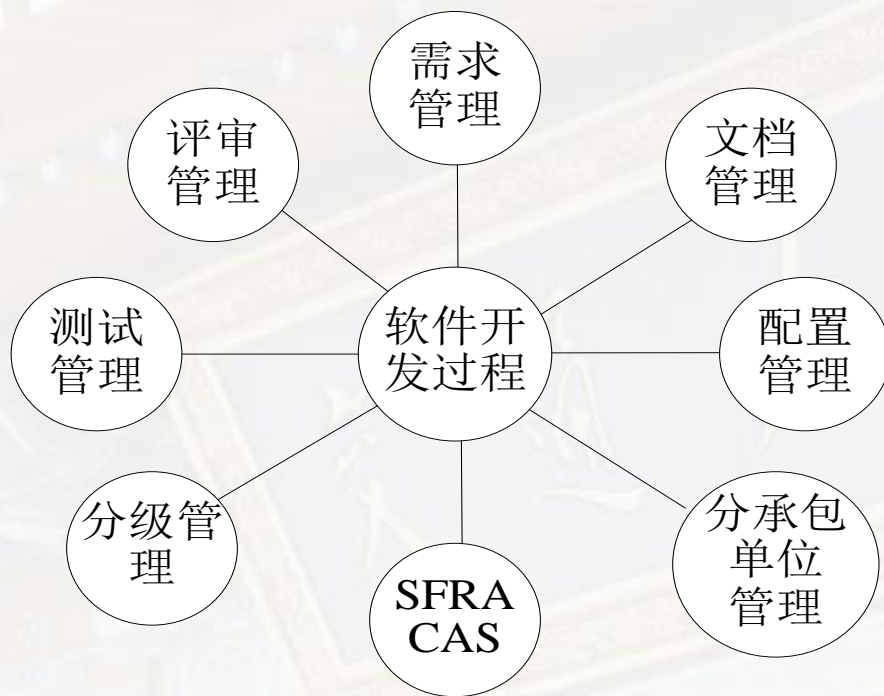
■ 过程控制要点

- 软件开发过程应当是可度量、可控、可改进的
- 软件开发过程应按时间顺序划分各子过程，对各子过程应确定各控制点
- 清楚地描述过程中的各项活动、任务及其结果，特别是应标识出对软件项目成功至关重要的任务
- 使用正确的、解析的质量参数，对软件开发过程提供有效的质量度量
- 在软件开发过程中，应将管理活动与工程活动结合起来
- 在软件开发过程中，必须认真进行软件工程的实践
- 标识软件开发过程中的风险领域，确定描述与跟踪各风险因素的方法和降低风险的途径

■ 软件生存期各阶段的过程控制



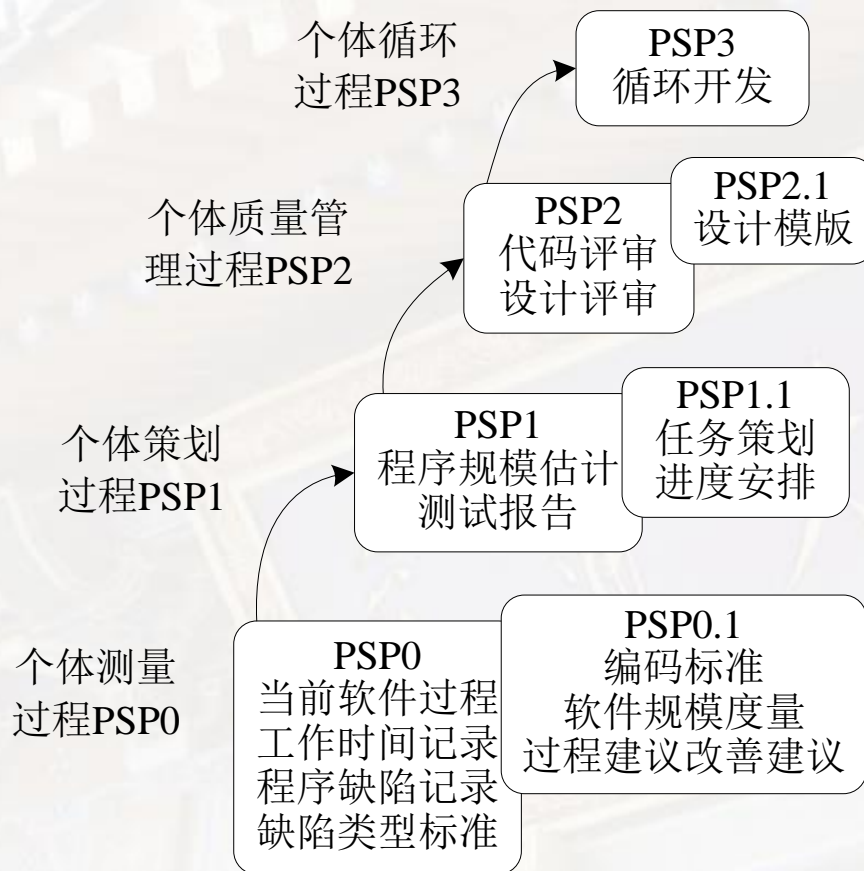
- 软件质量的全方位管理是对影响软件质量的各个关键要素进行严格管理，使软件开发按照软件质量要求规范化地实施



空间维——软件关键质量因素的全方位管理



■ 软件开发者的自我管理——个体软件过程（PSP）



三位一体的软件开发管理模式



- 软件开发者的团队管理（TSP）——小组软件过程
 - TSP的简单框架
 - 小组及其角色的管理目标及其度量评价
 - 软件能力成熟度模型（CMM）
 - 从CMM到集成的能力成熟度模型（CMMI）



下一代飞行器研制中的软件可靠性问题

美国F-22的教训

- F-22是目前美国空军飞机中最复杂的软件密集系统
- F-22软件控制着飞机上80%的功能，其费用占飞机工程与制造研制费用的30%
- IBM公司主要经验：
 - 认真实施软件工程
 - 特别加强软件测试

美国F-22的教训

- 1.认真实施软件工程
 - F-22软件开发强制性要求遵循MIL-STD-2167A标准
 - F-22软件开发强制性要求采用Ada军用标准语言
 - 为降低风险，F-22航空电子软件是分块批次开发的
 - 采用综合产品组（IPT）
 - 软件开发团队的成员采用相同的软件开发环境
 - 重视软件工程师的培训



美国F-22的教训

- F-22航空电子软件开发采用的是多V模型的开发方法
- F-22上的航空电子硬件和软件开发共用了4批次
 - 1.0批次（1999年上半年）
 - 2.0批次（1999年下半年）
 - 3.0批次（2000年4月）
 - 4.0批次（2005年）
- 2.特别加强软件测试
 - 航空电子综合实验（AIL）
 - 飞行实验台（FTB）试验
 - F-22研制试验飞机飞行试验





■ 3.F-22在飞行试验中出现的**软件可靠性问题**

- 美国空军要求F/A-22整个软件包工作1000小时不会发生任何1个航电组件实效的问题。在2000年2月的试验中，3小时飞行试验出现1次软件错误
- F-22飞行试验过程中，出现的众多航空电子软件可靠性问题在于各个组件之间的综合，在软件集成中遇到的最大挑战是实现多传感器的数据融合
- 研制进度延后一年半，费用超支80%。2000年，美国空军称，F-22已解决了航空电子软件可靠性问题，并称，“软件可靠性已不再是问题”

由于软件问题，F-22的座舱系统每运行2小时就要关闭一次。航空电子软件可靠性的问题能造成部分航空电子系统如雷达处于异常状态，甚至完全不能运行。在这种条件下，试飞员必须重新启动F-22航空电子系统





■ 3.F-22在飞行试验中出现的软件可靠性问题

■ 新问题产生：

(1) 飞控有关的软件问题

2004年12月20日，一架美军F/A-22猛禽战斗机坠毁。
此前F/A-22战斗机的原型机YF-22由于软件问题在降落时发生过一次坠毁事故

(2) 全球定位系统问题

2007年2月，美国空军12架F-22A战斗机从夏威夷飞往日本，途经国际日期变更线时，飞机上的全球定位系统纷纷失灵，多个电脑系统发生崩溃，多次重启均告失败

对87架F-22A进行全面检查，对有问题的软件系统实施修改升级





1

航电系统的开发要求-ARP4754



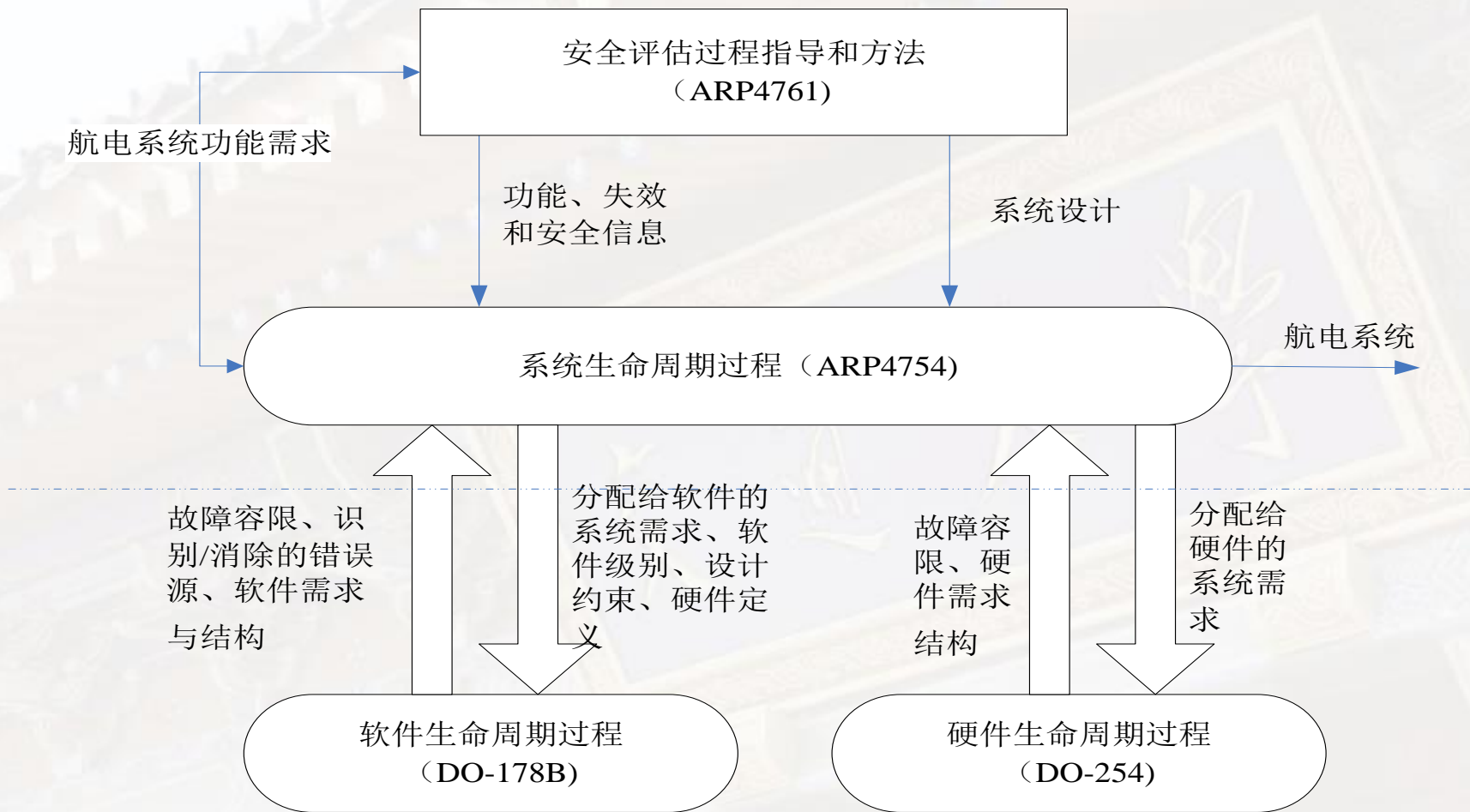


- 1) ARP4754与其他标准之间关系
- 2) 航电系统开发过程模型
- 3) 航空器级功能实现过程模型

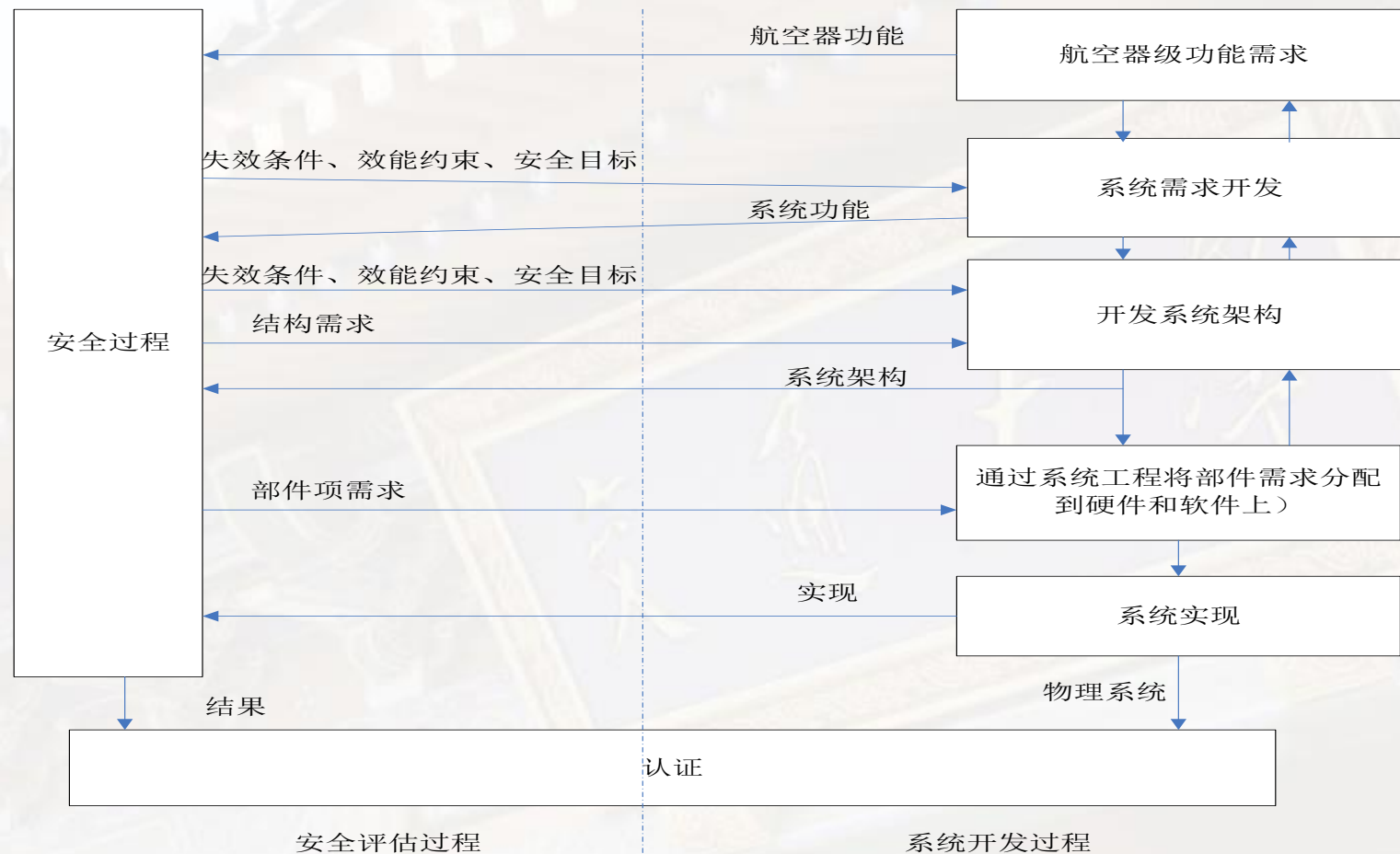




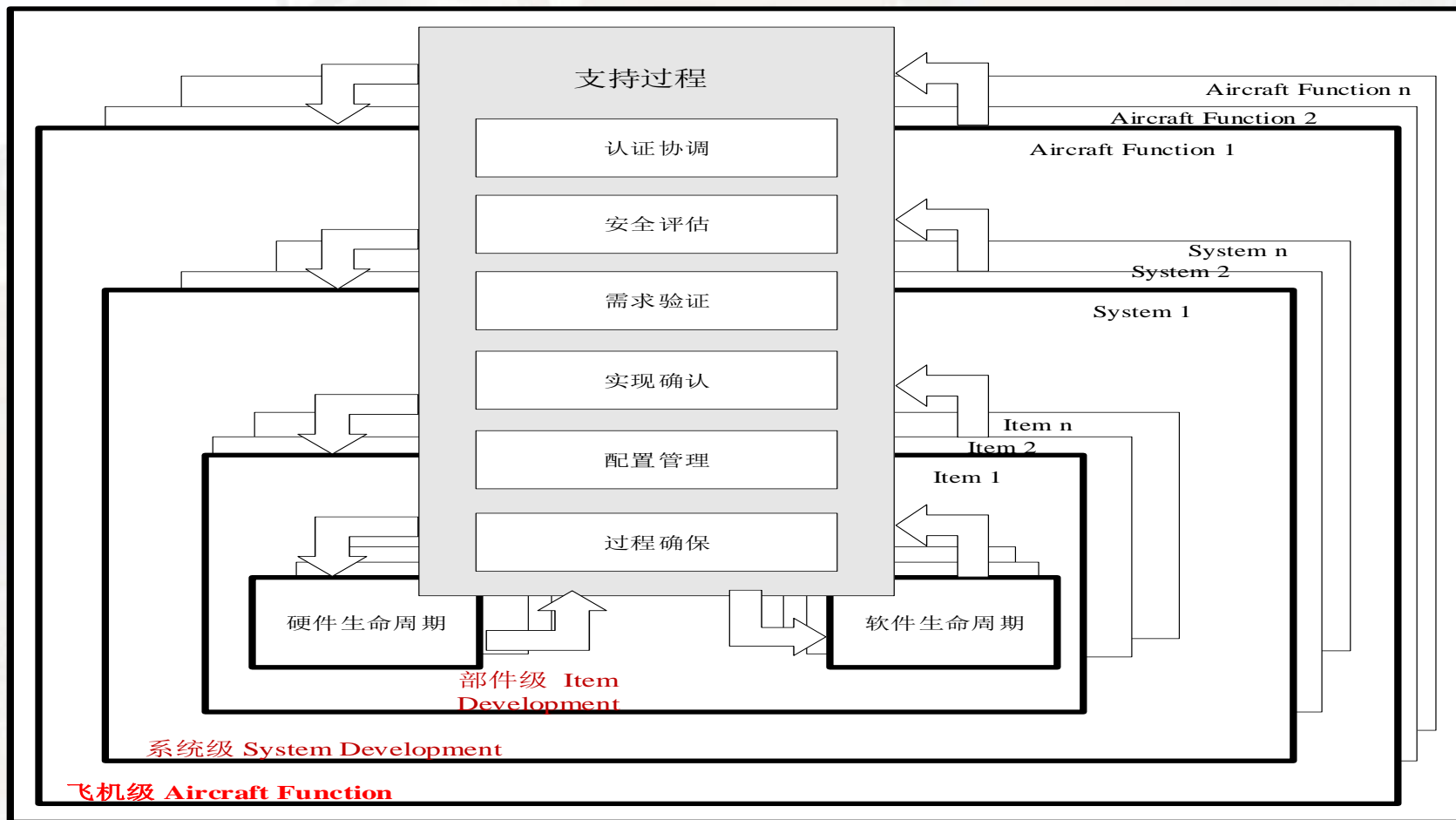
➤ 1) ARP4754与其他标准之间关系



➤ 2) 航电系统开发过程模型



➤ 3) 航空器级功能实现过程模型





2

民用航空适航认证标准-DO-178B





- 1) DO-178B软件过程及目标定义
- 2) 软件计划过程
- 3) 软件开发过程
- 4) 软件综合过程
- 5) 航空电子软件安全性评估





DO-178B 是由 RTCA (Requirements and Technical Concepts for Aviation) 和 EUROCAE (European Organization for Civil Aviation Electronics) 共同发布。





➤ DO-178B软件过程及目标定义

◆ 目标导向

- DO-178B针对不同级别的软件，定义了一系列的目标、这些目标的独立性要求、实现这些目标应生成的生命周期数据（Lifecycle Data），并定义了这些数据的控制类别（Control Category）；
- DO-178B要求给出验证这些目标的方式；
- DO-178B要求给出达成目标的指标及证明。





◆ 软件生存周期:

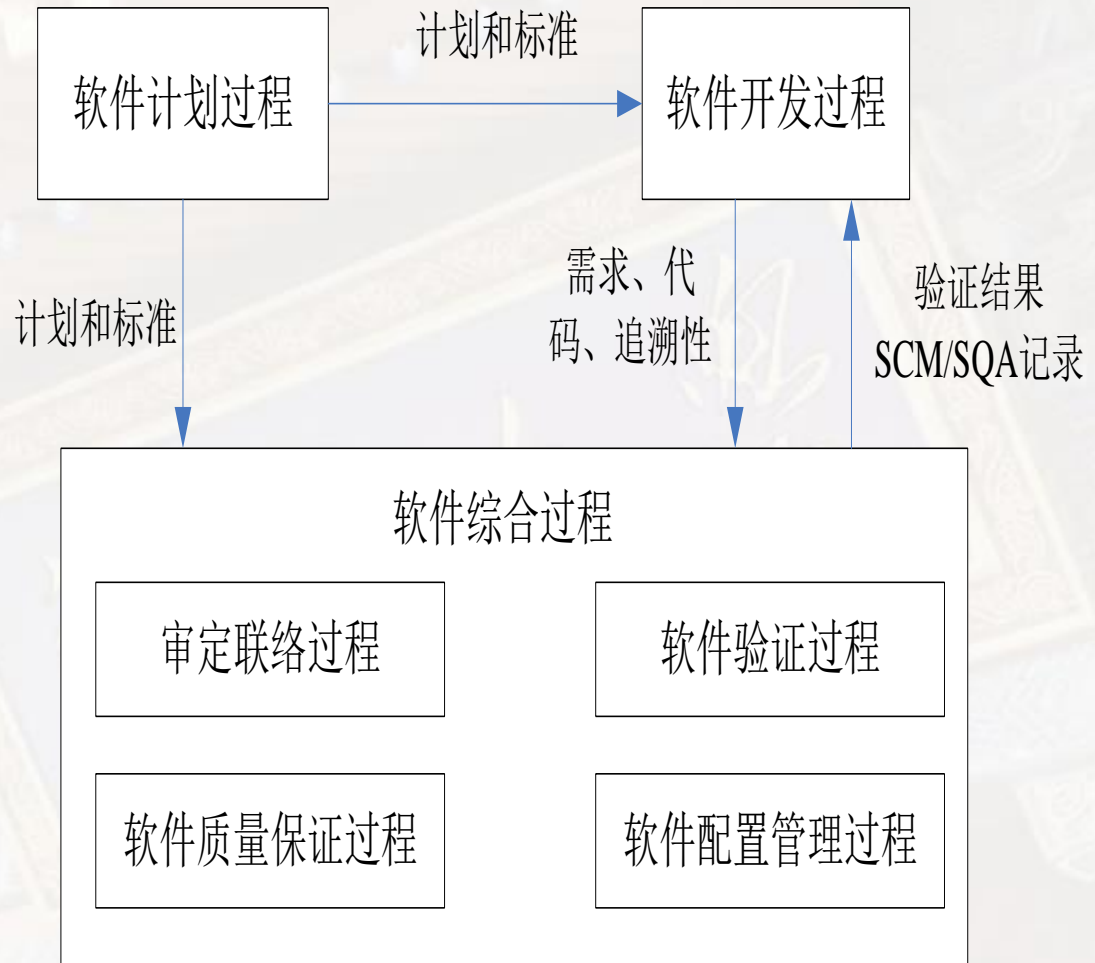
□ 软件计划过程

□ 软件开发过程

- 需求
- 设计
- 编码
- 集成

□ 软件综合过程

- 软件验证
- 软件配置管理
- 软件质量保证
- 合格审定联络



- ❑ 软件计划过程的目标是定义产生满足系统需求并提供与适航要求相一致的置信度水平的软件方法（包括活动及活动之间关系定义、开发标准）；
- ❑ 软件开发过程
 - 软件**需求**过程开发软件高级需求，包括功能、性能、接口和与安全性有关的需求；
 - 软件**设计**过程中，通过一次或多次迭代，逐步完善出软件高级需求，以开发软件结构和能用于实现源代码的低级需求；
 - 软件**编码**过程将软件结构和低级需求转换为源代码；
 - 软件**集成**过程实现把将执行目标代码加载到软件/硬件综合的目标硬件中；

□ 软件综合过程

- 软件**验证**过程的目的是检测和报告在软件开发过程中可能已形成的错误；
- 软件**质量保证**过程用于审核软件的生命周期过程及其输出，确保其目标被实现，错误被检测、评估、跟踪和解决，其他软件生命周期数据满足了软件需求；
- 软件**配置管理**过程用于在软件生命周期中提供确定的、可控的软件配置；
- **证明联络**过程（合格审定过程）用于在整个软件生命周期中建立应用程序与证明授权之间的通讯和理解，辅助软件的证明过程。

➤ 软件计划过程

◆ 过程目标定义

- 1) 定义软件验证的手段，以确保研制的软件满足系统需求，并且有足够的信心保证软件达到适航要求；
- 2) 制定软件计划；
- 3) 定义软件标准；
- 4) 定义软件开发过程和软件综合过程的活动；
- 5) 定义软件生命周期，包括：过程间的相互关系、过程的先后顺序、反馈机制、迁移准则；
- 6) 选择软件生命周期，包括：软件开发环境、语言和编译器、软件测试环境（基本上，包含了每个过程活动中使用的方法和工具）；
- 7) 有必要时提及其他考虑；





◆过程输入

- 1) 系统需求
- 2) 软件安全级别

◆过程输出

- 1) 软件开发计划——SDP：定义时间节点、组织结构、独立性要求、交付物以及里程碑；定义软件生命周期和开发环境；
- 2) 软件验证计划——SVP：定义验证方法、验证环境；定义组织结构；
- 3) 软件配置管理计划——SCMP：说明管理软件和硬件配置的活动；定义用于走查和发布的基线；确保所有的目标代码可以重复地、一致地生成；提供问题报告、跟踪、授权、走查以及防止未授权的变更；保障信息安全，确保所有电子数据的备份和恢复。





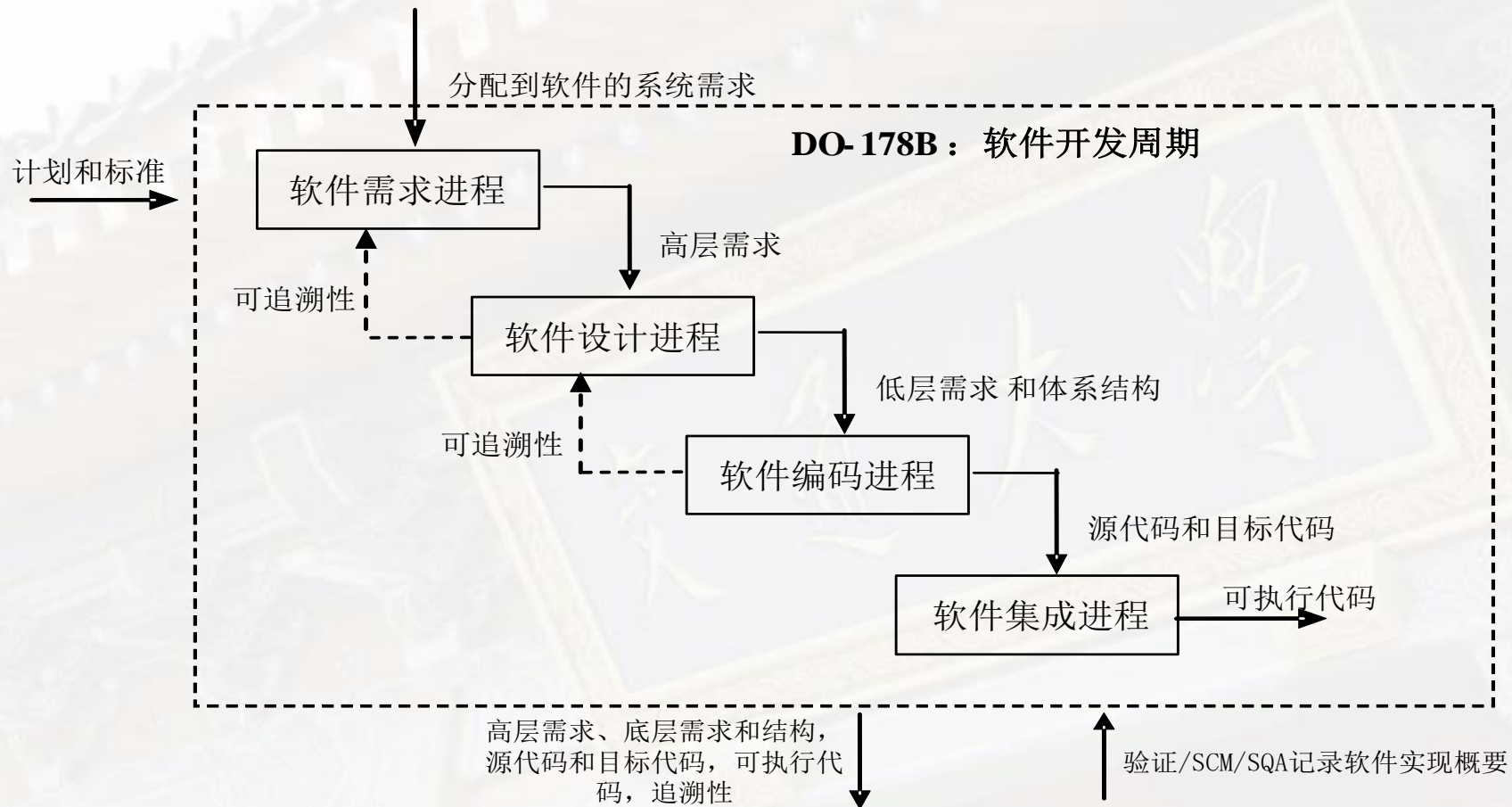
- 4) 软件质量保证计划——SQAP: 定义质量保证的活动, 定义软件生命周期数据和过程的走查方法; 描述软件一致性走查活动。
- 5) 软件需求标准——SRS
- 6) 软件设计标准——SDS
- 7) 软件编码标准——SCS
- 8) 软件合格审定计划——PSAC

◆ 迁移条件

- 1) 被内部质量部门接受;
- 2) 被认证机构或工程委任代表 (DER) 批准;



➤ 软件开发过程



◆ 软件需求过程

□ 活动定义

- 1) 根据分配到软件的系统需求来开发软件高层需求 (HLR)
- 2) 高层需求包括功能、性能、接口和安全相关的需求

□ 输入

- 1) (分配到软件的) 系统需求
- 2) 软件需求标准 (SRS)
- 3) 软件开发计划 (SDP)

□ 输出

- 1) 软件高层需求 (HLR)
- 2) 系统需求和高层需求之间的追踪关系

□ 迁移条件

- 1) 通过了软件需求过程的验证
- 2) 高层需求通过了系统安全评估过程

◆ 软件设计过程

□ 活动定义

- 软件高层需求通过一次或多次的迭代细化得到底层需求和软件架构

□ 输入

- 1) 软件需求数据 (HLR)
- 2) 软件设计标准 (SRS)
- 3) 软件开发计划 (SDP)

□ 输出

- 1) 设计描述软件低层需求
- 2) 低层需求和高层需求之间的追踪关系

□ 迁移条件

- 1) 通过软件设计过程输出的验证



◆ 软件编码过程

□ 活动定义

- 1) 由软件低级需求得到源代码
- 2) 源代码是可追踪的, 可验证的, 内在一致的, 并且正确地实现了底层需求

□ 输入

- 1) 低层需求和软件结构
- 2) 软件编码标准 (SCS)
- 3) 软件开发计划 (SDP)



□ 输出

- 1) 源代码
- 2) 目标代码
- 3) 源代码和低层需求之间的追踪关系

□ 迁移条件

- 1) 源代码符合低级需求与软件结构
- 2) 源代码可验证
- 3) 源代码符合标准，准确和一致
- 4) 源代码可追踪到低级需求





◆ 软件集成过程

□ 活动定义

- 1) 软件的集成：由源代码和目标代码生成可执行代码
- 2) 硬件/软件的集成：将可执行代码下载到目标机以形成集成机载系统和设备

□ 输入

- 1) 软件结构
- 2) 源代码/目标代码
- 3) 目标机



□ 输出

- 1) 可执行目标码, 链接和装载数据
- 2) 集成机载系统或设备

□ 迁移条件

- 1) 软件综合过程的输出完整和正确



➤ 软件综合过程

◆ 软件验证过程

□ 活动定义

- 1) 走查 (review) : 通过检查表或其他类似方法, 进行人工和定性检查;
- 2) 分析: 检查一个软件部件的功能、性能、可追踪性和安全性, 以及与其他部件的关系 (可重复的、定量的检查, 可以自动化);
- 3) 测试: 测试用来必须来自于需求, 对需求内在一致性及完整性进行进一步的评估;



□ 输入

- 1) 软件验证计划
- 2) 所有需要验证的过程的输出（软件开发过程、软件验证过程等）
- 3) 所有需要验证的过程的输入

□ 输出

- 1) 走查与分析结果
- 2) 测试用例、测试过程及测试结果
- 3) 软件验证用例和规程
- 4) 软件验证结果

□ 迁移条件

- 1) 符合软件验证计划



◆ 软件配置管理过程

□ 活动定义

- 1) 配置标识
- 2) 基线的建立和追踪
- 3) 问题报告、跟踪和更正
- 4) 更改控制
- 5) 更改走查
- 6) 配置状态统计
- 7) 存档、恢复和发布
- 8) 软件加载控制
- 9) 软件生命周期环境控制



□ 输入

- 1) 软件开发计划
- 2) 软件配置计划

□ 输出

- 1) 配置项

□ 迁移条件

- 1) 达到了软件配置计划中定义的目标



◆软件质量保证过程

- 该过程由软件质量保证计划定义，用于审核软件的生命周期过程及其输出，确保其目标被实现，错误被检测、评估、跟踪和解决，其他软件生命周期数据满足了软件需求。
- 软件质量保证过程提供软件生命周期生产的软件产品与需求是一致的证明，保证这些过程的执行与软件计划和标准一致；





◆ 审定联络过程

- 合格审定是对产品、服务、组织或人员进行技术检查，并对符合要求的产品颁发国家法律和法规要求的合格证、许可证、批准书或其他证明文件的活动。
- 机载软件的合格审定工作是由适航性合格审定机构（适航部门）完成的，其审查的依据是RTCA DO-178B 《机载系统和设备合格审定中的软件考虑》；
- 审查的对象是申请人开发的机载软件；
- 审查的目的是确定软件是否满足RTCA DO-178B中规定的目标。



➤ 航空电子软件安全性评估

◆ 失效状态和软件等级

□ 失效状态类别

- **灾难性的** 阻止继续安全飞行和着陆的失效状态。
- **危险的 / 严重的** 降低航空器的性能和机组人员克服不利操纵状态的能力的失效状态。这些不利操纵状态达到的程度是：大大降低了安全性余量或功能能力；身体疲劳或高负荷使飞行机组不能精确或完整地完成任务；或对乘客的不利影响，包括对少数乘客严重的或潜在的致命伤害。





- **较重的** 可能降低航空器的性能和机组人员克服不利操纵状态的能力的失效状态。这些不利操纵状态达到的程度如：较大地降低安全余量或功能能力、较大地增加了机组人员的工作量或削弱机组人员工作效率的状态，或造成乘客不舒服，可能包括伤害。
- **较轻的** 不会严重降低航空器安全性及有关机组的活动在他们的能力内能很好完成的失效状态。较轻的失效状态可能包括：如稍微减少安全余量或功能能力；稍微增加机组人员的工作量，如航线飞行计划更改或乘客的某些不方便。
- **无影响的** 不影响航空器的工作性能或不增加机组工作量的失效状态。





□ 软件等级定义

- **A 级** 可能引起或导致系统功能失效进而引起航空器灾难性失效状态的异常状态的软件，这种异常状态可通过系统安全性评估过程来表明。
- **B 级** 可能引起或导致系统功能失效进而引起航空器危险的/严重的失效状态的异常状态的软件，这种异常状态可通过系统安全性评估过程来表明。
- **C 级** 可能引起或导致系统功能失效进而引起航空器较重失效状态的异常状态的软件，这种异常状态可通过系统安全性评估过程来表明。





- **D 级** 可能引起或导致系统功能失效进而引起航空器较轻失效状态的异常状态的软件，这种异常状态可通过系统安全性评估过程来表明。
- **E 级** 可能引起或导致系统功能失效的异常状态的软件。这种异常状态可通过系统安全性评估过程来表明。它不会影响航空器的工作性能或驾驶员工作量。一旦软件由合格审定机构定位E 级，本文件就不再提供进一步的指南。



◆航空电子软件安全性评估

- 功能危害度评估
- 最初系统安全性评估
- 系统安全性评估

目前安全评估主要通过测试、仿真和故障注入等手段进行。





3

航空电子软件过程





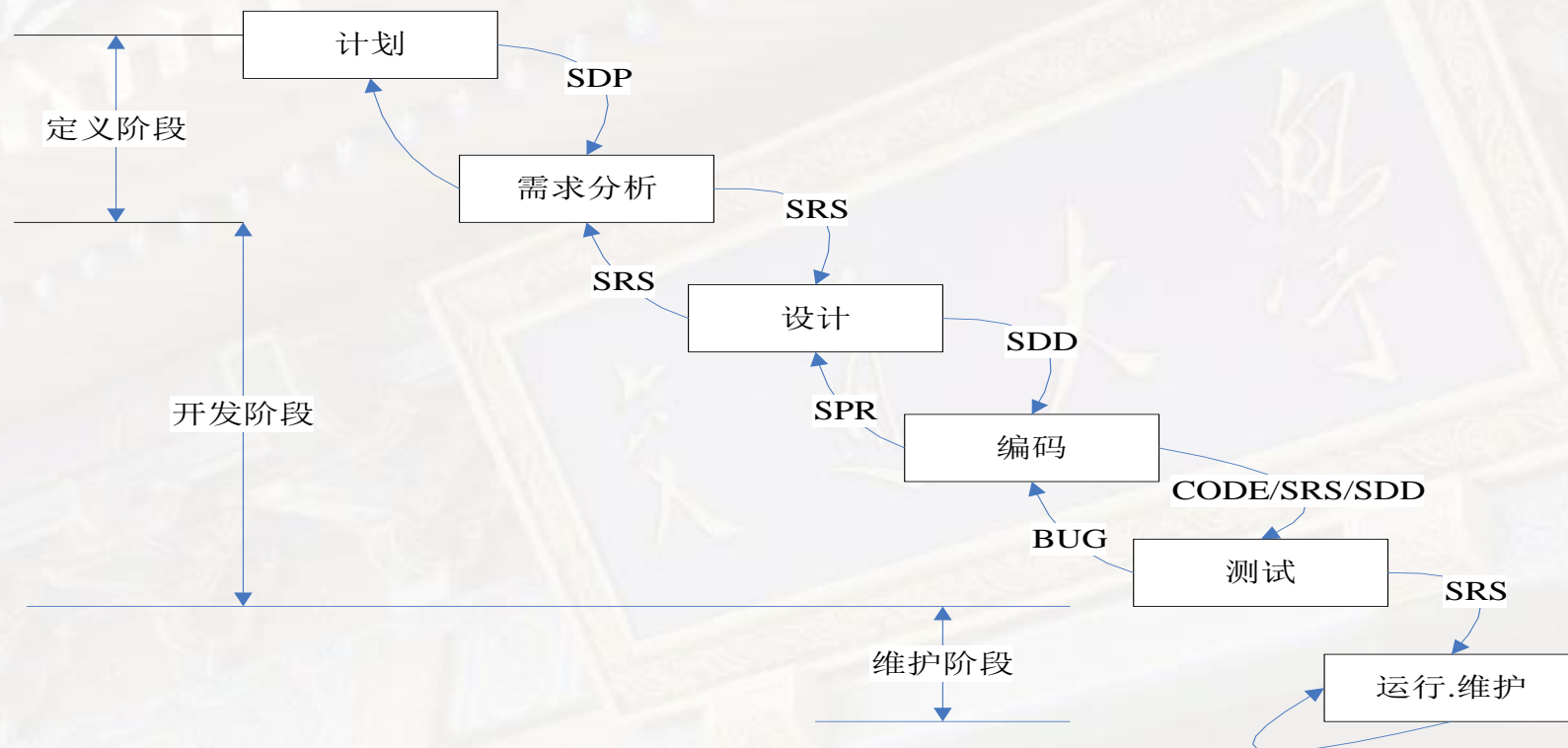
- 1) 软件过程模型
- 2) 软件过程支持工具



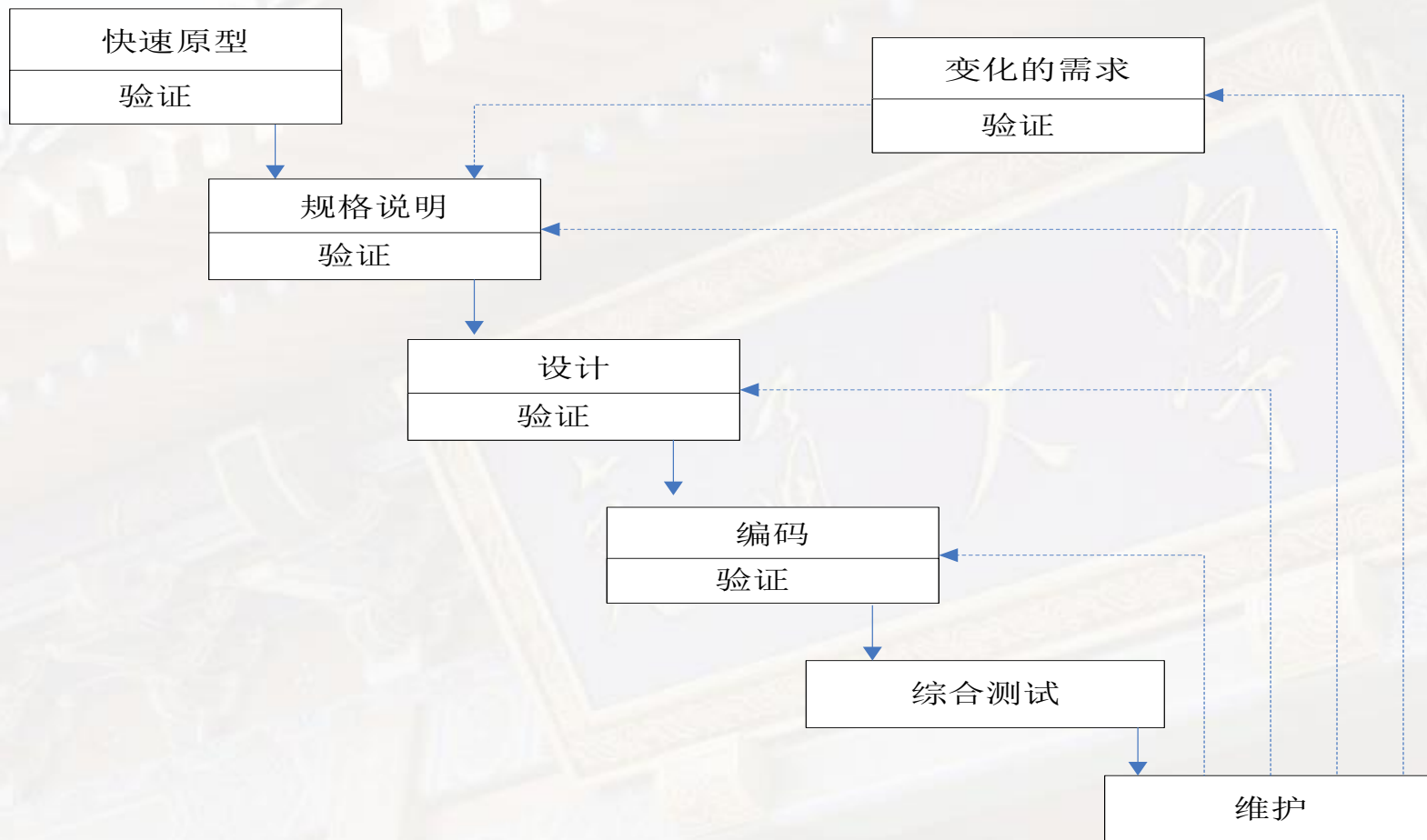


➤ 1) 软件过程模型

◆ 瀑布开发模型

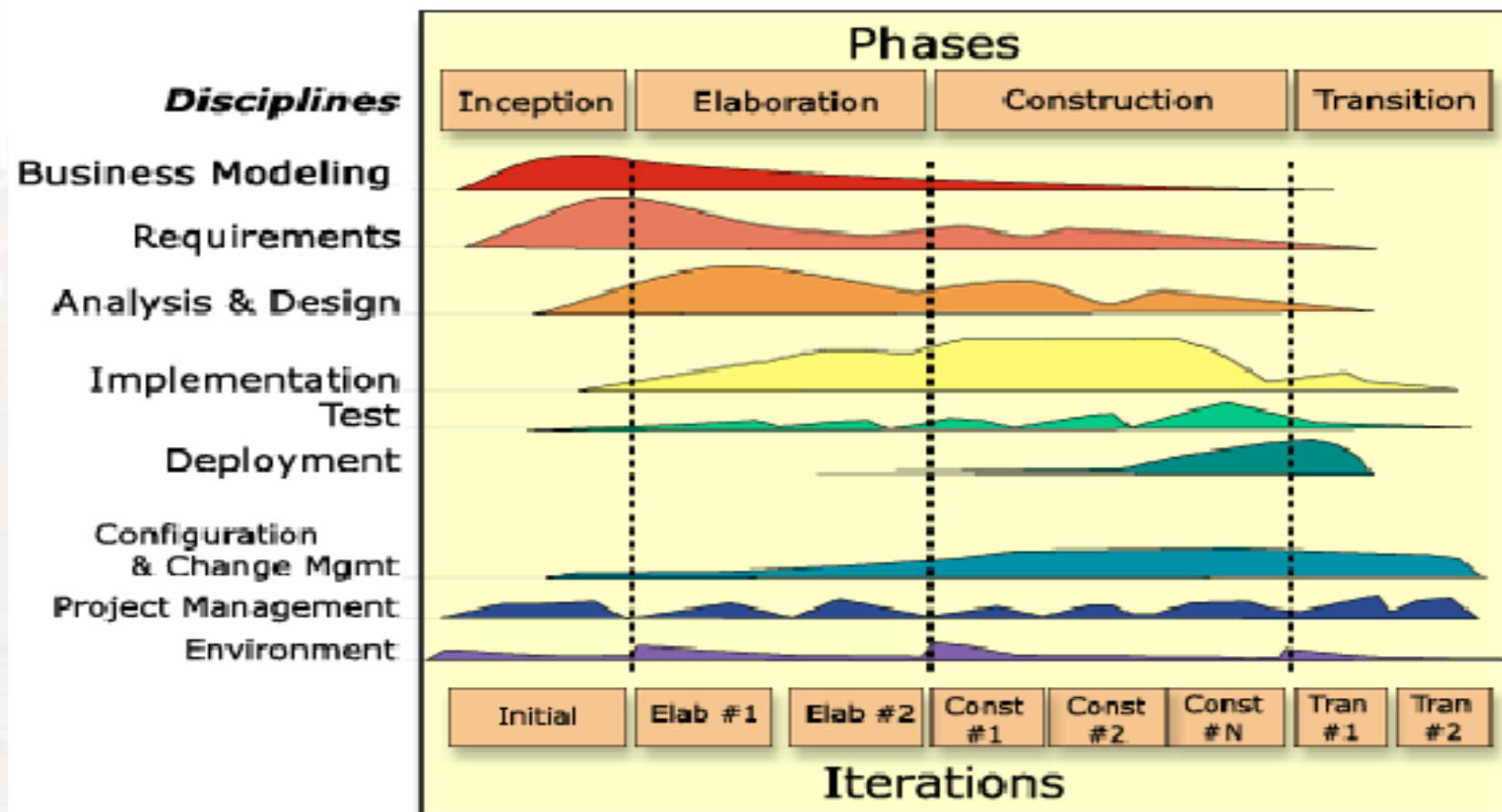


◆快速原型法



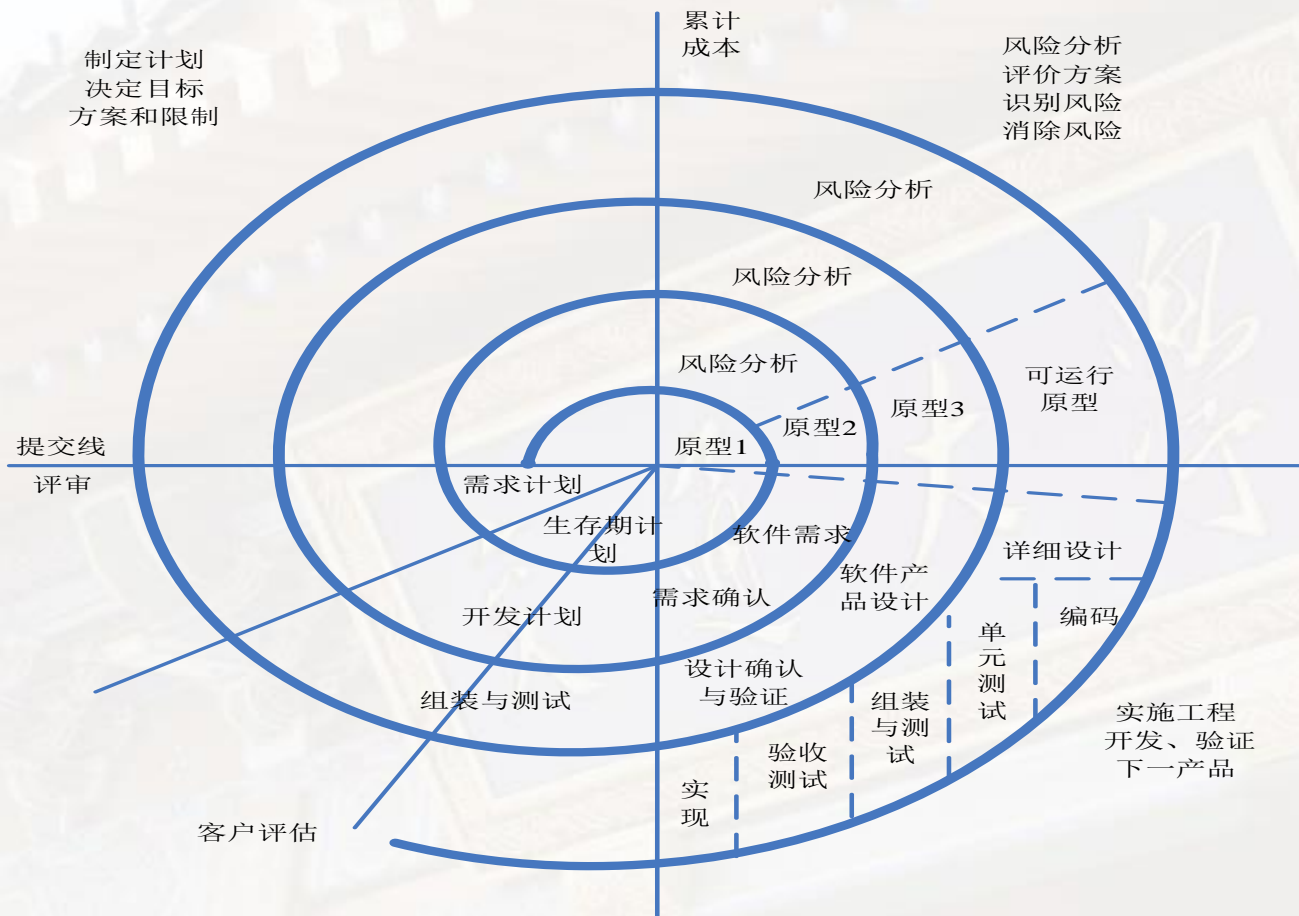


◆RUP迭代模型





◆螺旋式模型





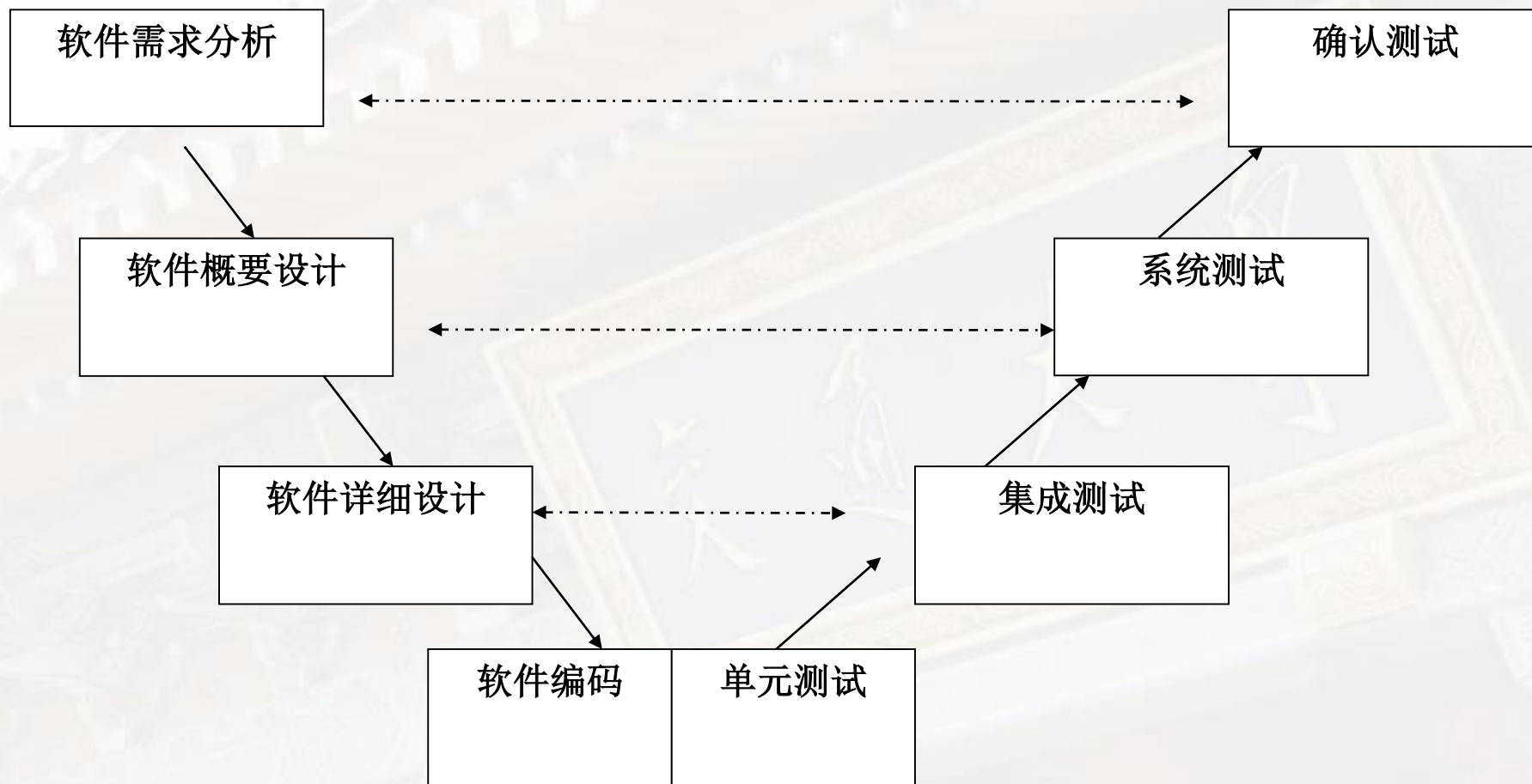
◆敏捷开发模型

□ 敏捷方法的核心思想主要有下面三点：

- （1）敏捷方法是适应性，而非可预测性。与传统方法不同，敏捷方法适应变化的需求，利用变化来发展；
- （2）敏捷方法是以人为本，而非以过程为本。传统的方法以过程为本，强调充分发挥人的特性，不去限制它。并且软件开发在无过程控制和过于严格繁琐的过程控制中取得一种平衡，以保证软件的质量。
- （3）迭代增量式的开发过程。敏捷方法以原型开发思想为基础，采用迭代增量式开发，发行版本小型化。它根据客户需求的优先级和开发风险，制定版本发行计划，每一发行版都是在前一成功发行版的基础上进行功能需求扩充，最后满足客户的所有功能需求。



◆V开发模型





➤ 2) 软件过程支持工具

◆ 软件计划

- Microsoft Office
- Visio
- Microsoft Project

◆ 软件需求分析

- IBM Rational DOORS
- VeroTrace





◆ 软件设计

- IBM Rational Rhapsody
- SCADE

◆ 软件编码

- 常见的代码编辑环境有VC++6.0、Vi、UltraEdit等；
- 编译、调试环境有CL、GCC、GDB、TurboC、Wind River Tornado、TigerSharc等，并最终在Tornado/TigerSharc等编译环境下生成目标代码，源代码通过配置库管理。





◆软件验证

- 非定量手段（如评审、分析、仿真等），专门的测试软件工具如 LDRA Testbed、Logiscope、Macabe等
- 定量手段（如测试）

◆软件配置管理

- Sourcesafe:
- IBM Rational ClearCase
- PVCS





4

基于模型驱动的航电软件开发方法 – Harmony/ESW



- 1) Harmony/ESW方法
- 2) Harmony/ESW过程
- 3) 基于Harmony/ESW的航电软件开发环境实例



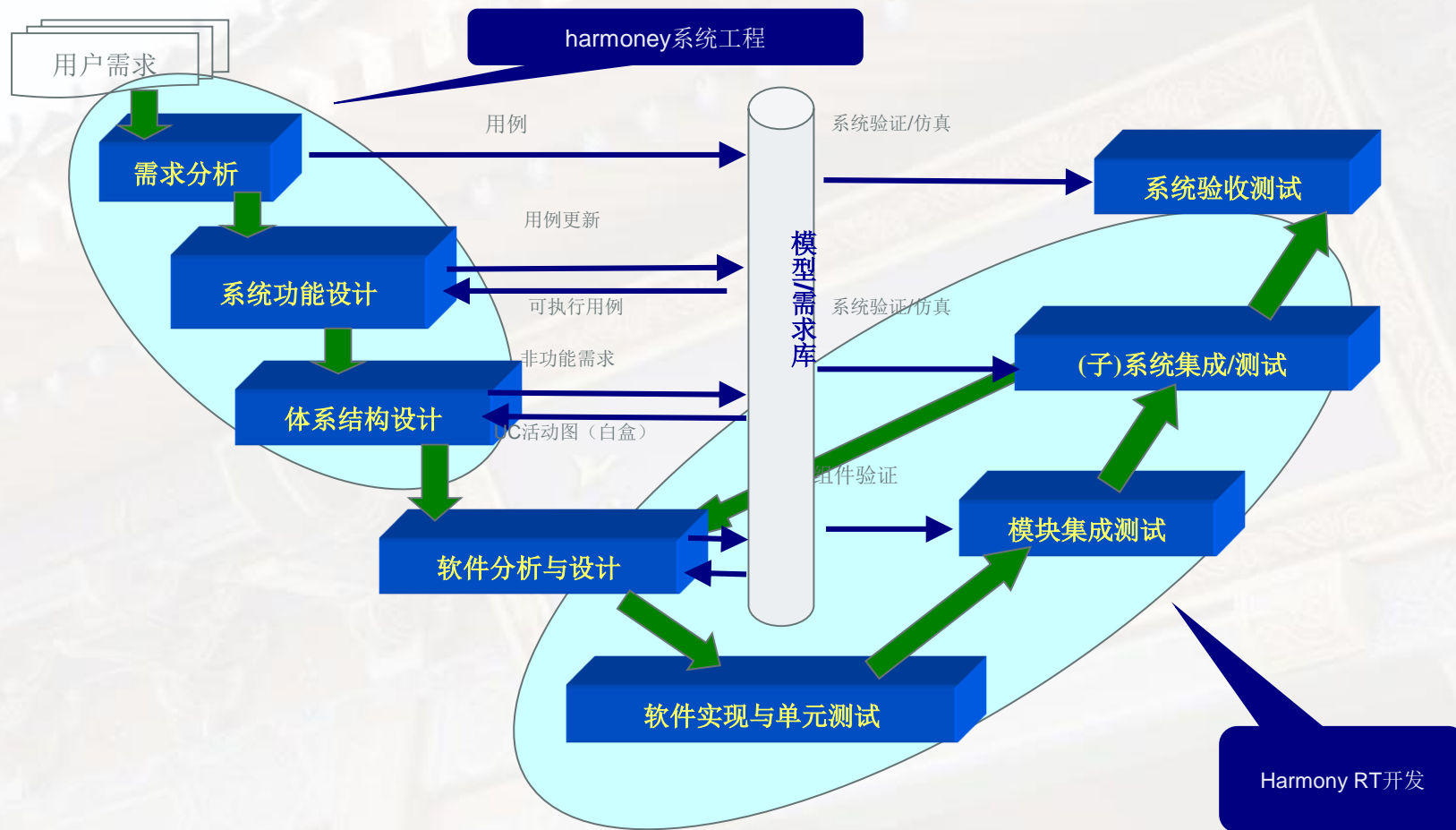


➤ 1) Harmony/ESW方法

- Harmony/ESW是Bruce Powel Douglass与Hans-Peter Hoffmann共同提出的应用于实时嵌入式系统开发的Harmony过程，是结合了模型驱动开发（Model Driver Development——MDD）技术的敏捷开发方法。
- 模型驱动开发（MDD）是模型驱动架构MDA中的一部分，MDA是模型驱动开发方法的概念框架。
- MDA的核心思想是抽象出与实现技术无关、完整描述系统的平台独立模型（Platform Independent Model, PIM），针对不同实现技术制定变换定义；通过变换工具将PIM转换成与具体实现技术相关的平台相关模型（Platform Specific Model, PSM）；最后，再通过变换工具将PSM自动转换成代码。在MDA中，软件开发过程是由对系统的建模行为进行驱动的。



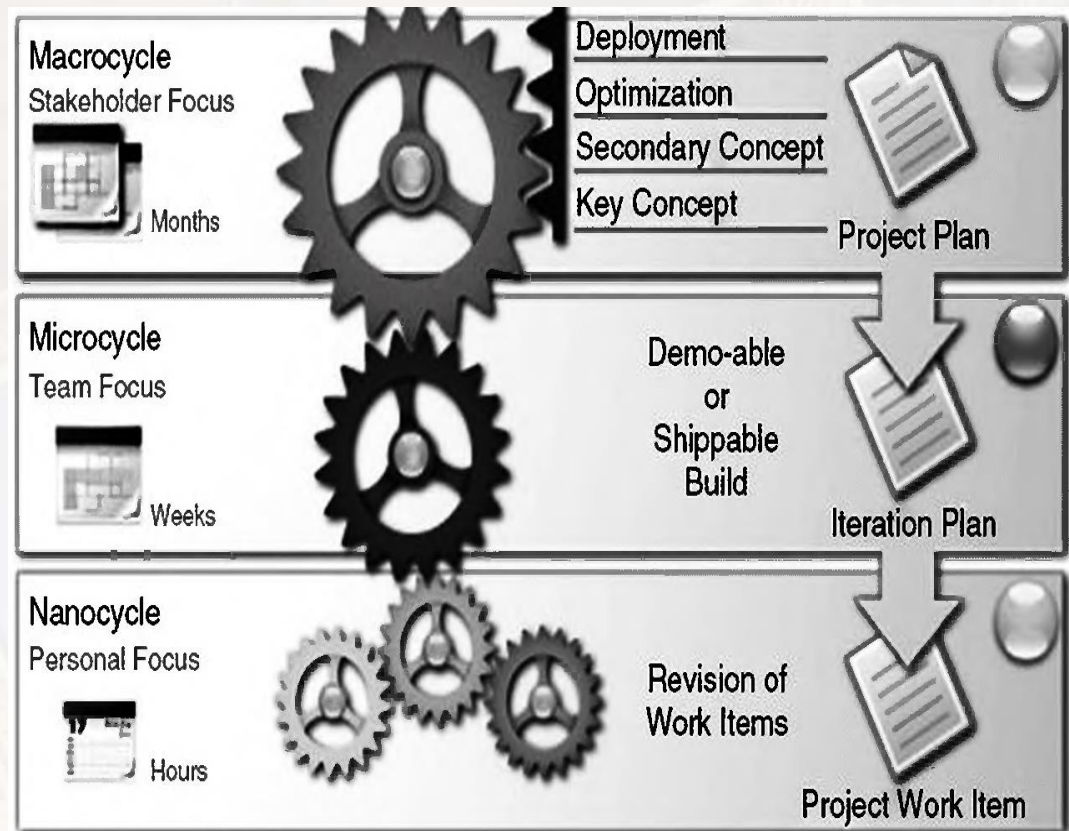
➤ 2) Harmony/ESW过程





◆Harmony过程的生命周期特征

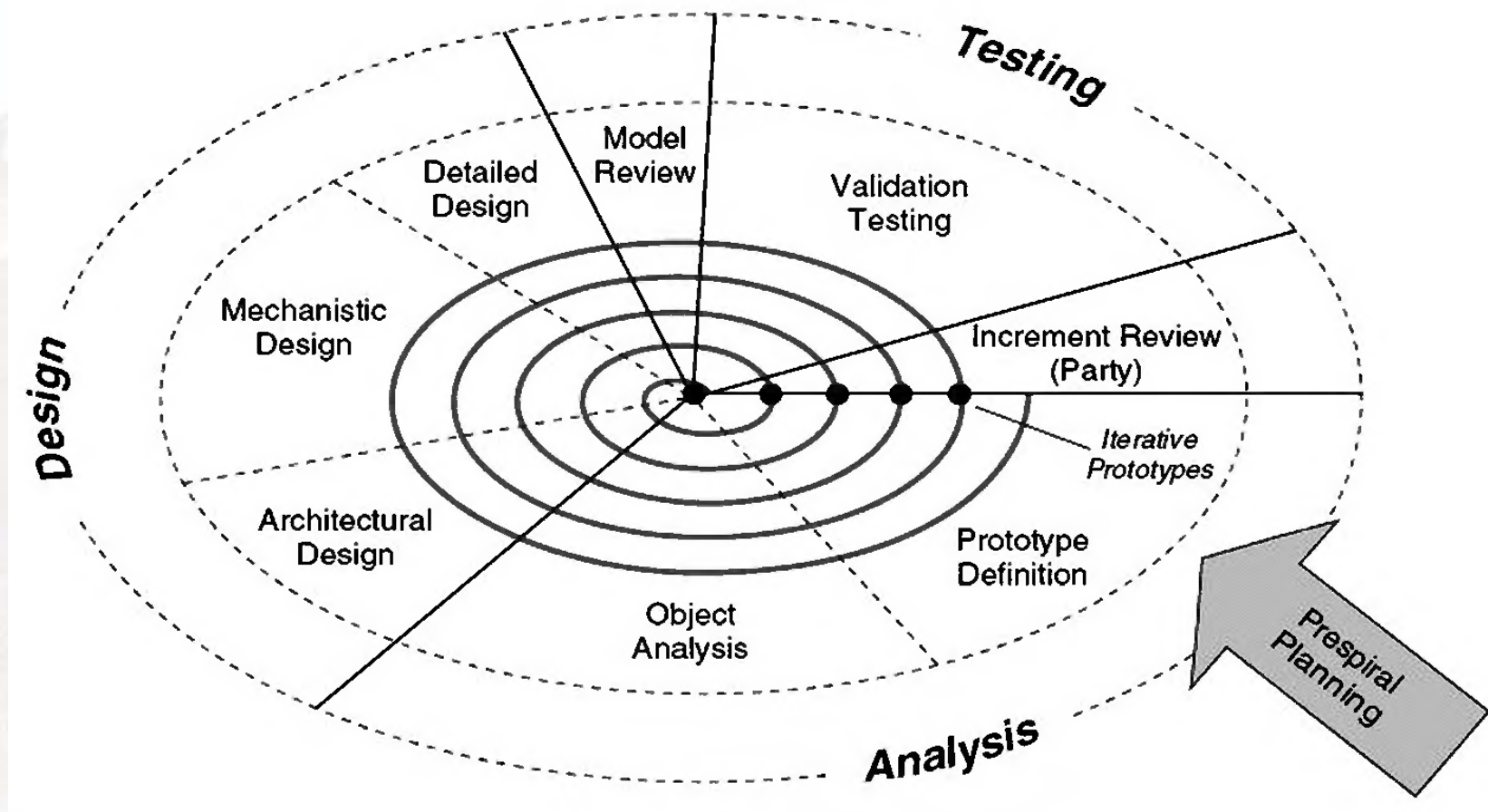
- 宏循环 (Macrocycle)
- 项目级微循环 (Microcycle)
- 个人微循环 (Nanocycle)



◆Harmony 宏循环--Macrocycle

- (1) 第一阶段：定义关键模型元素
- (2) 第二个阶段：关注次要模型元素
- (3) 第三个阶段：主要解决系统优化问题
- (4) 第四个阶段：重点考虑系统部署方面的问题

◆Harmony 微循环 -- Microcycle



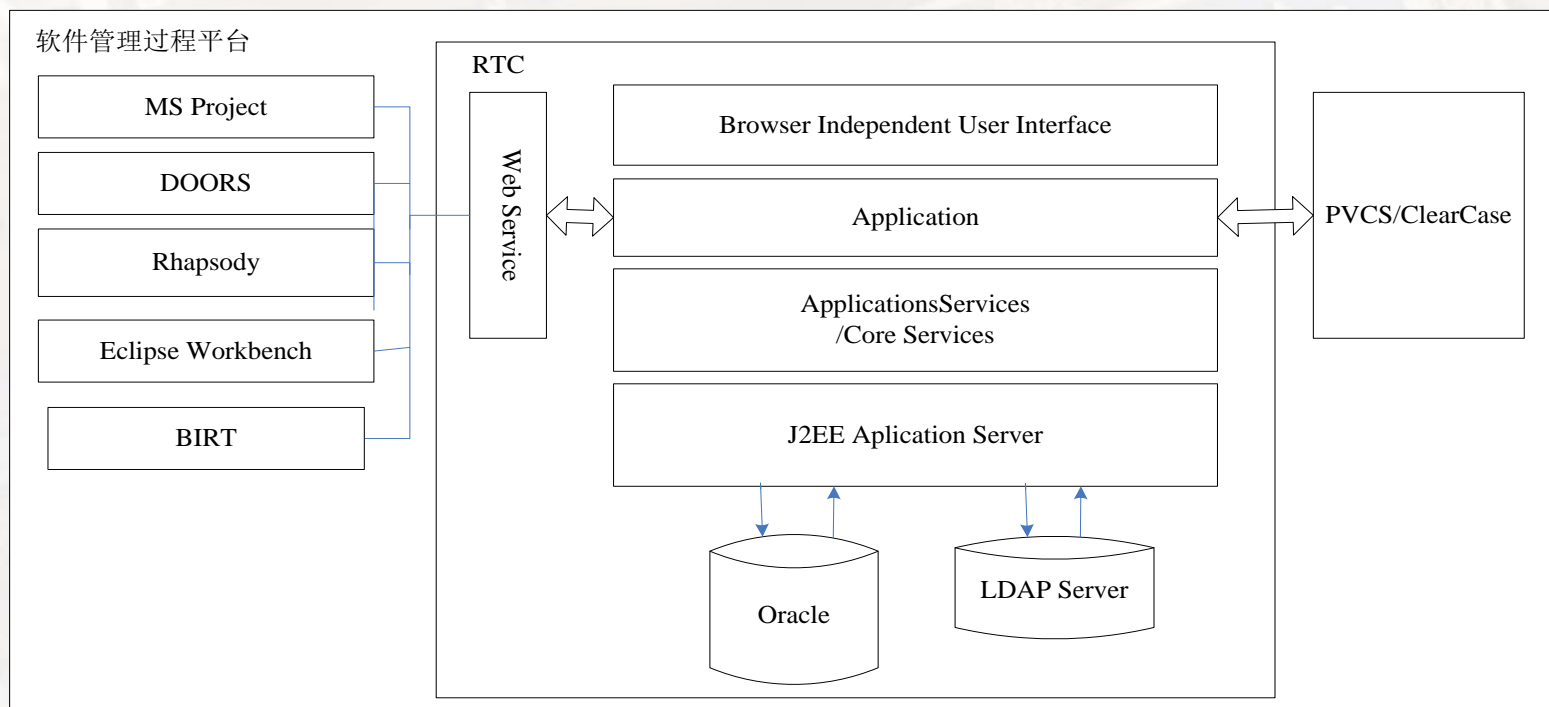


- (1) 分析：包括两个阶段：原型定义和对象分析阶段，其中原型定义阶段主要是需求分析，建立用例，确立系统完成的目标；对象分析阶段主要建立可运行的原型系统模型。
- (2) 设计：对对象分析模型进行优化，这包括三个层次：架构设计层次、机制设计层次、详细设计层次。
- (3) 模型审查：对所有产生的工件进行审查，包括模型、源代码、测试结果。
- (4) 测试与验证：验证模型和源代码是否满足在这个迭代周期内的系统需求目标。
- (5) 增量审查（party）：分析项目进度和软件质量，并据此对过程和环境进行改进。

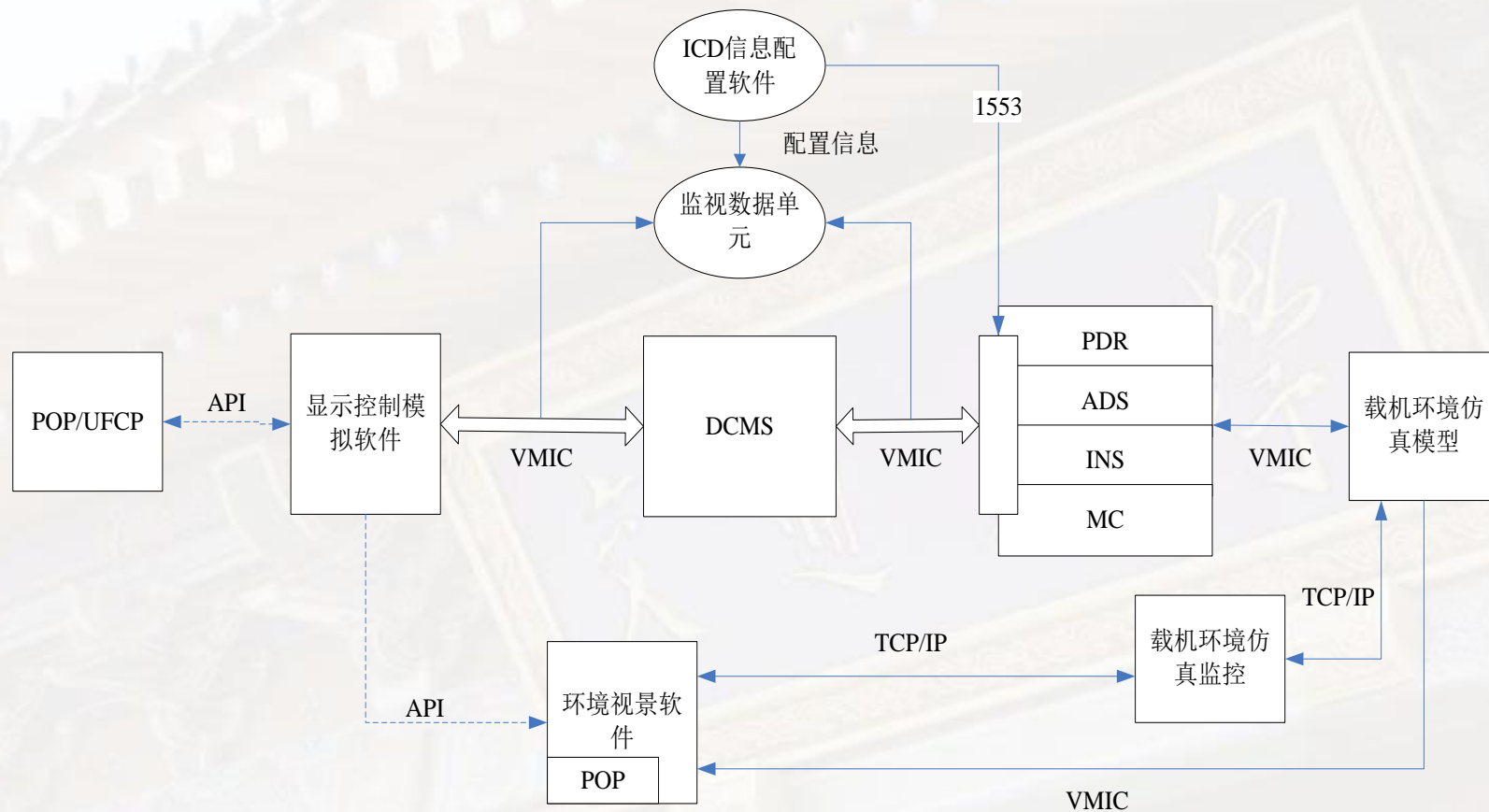


➤ 3) 基于Harmony/ESW的航电软件开发环境实例

◆ 软件工程管理平台

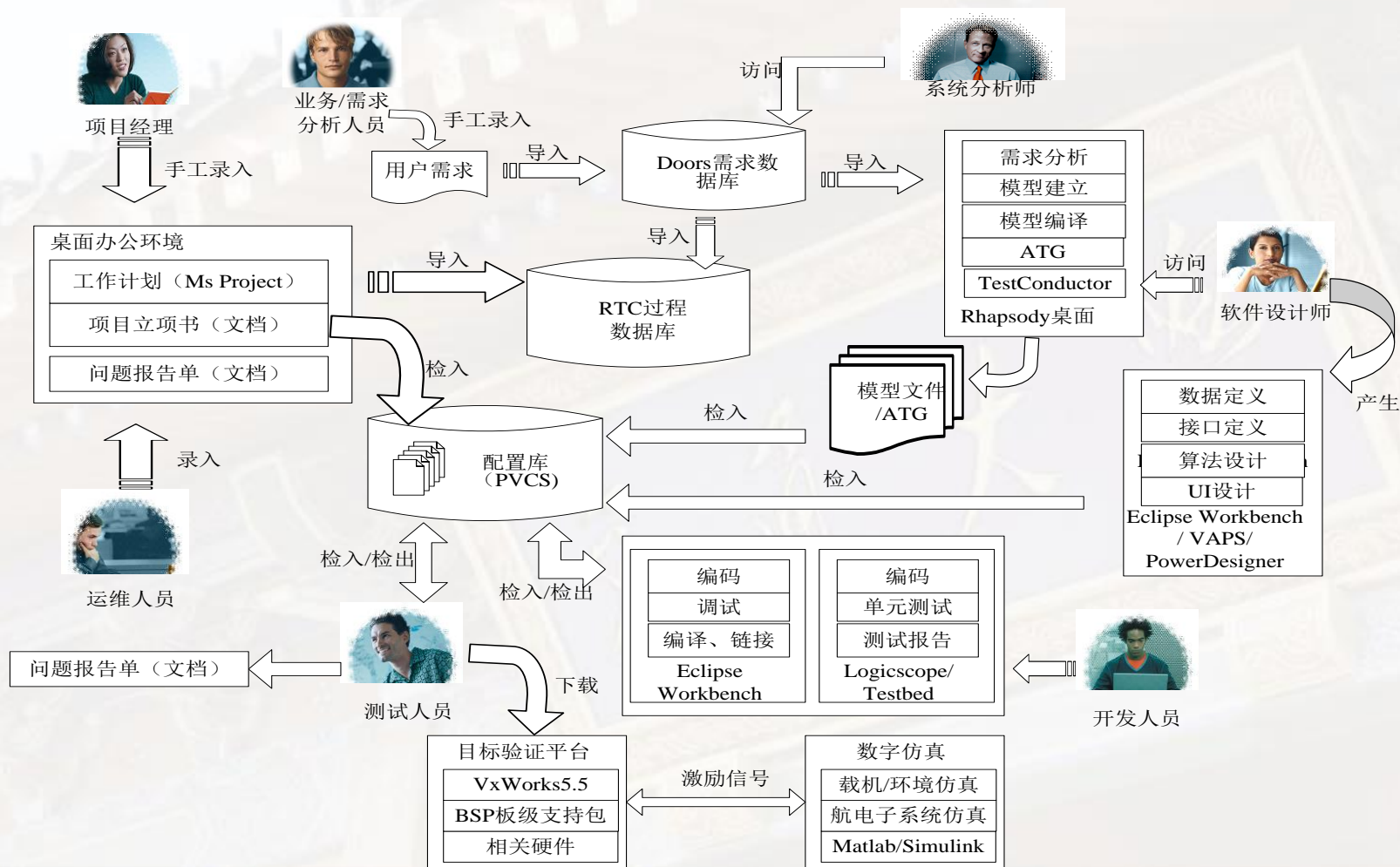


◆ 数字仿真验证平台



4 基于模型驱动的航电软件开发方法 - Harmony/ESW

◆ 软件工程管理与数字仿真平台的集成



本章小结

概述

ARP4754

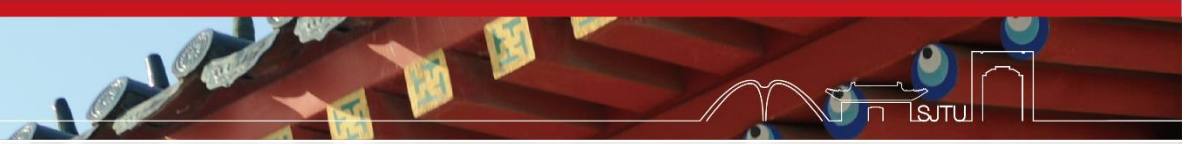
DO178B

开发
过程

开发
方法



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY



1. 阅读5~10篇论文，关于以下方向：

- ❑ DO178B
- ❑ 可靠性
- ❑ 数字化验证
- ❑ Harmony/ESW

并选择其中一个方向做个5mins PPT介绍与交流。



Thanks!
Questions?



上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

上海交通大學