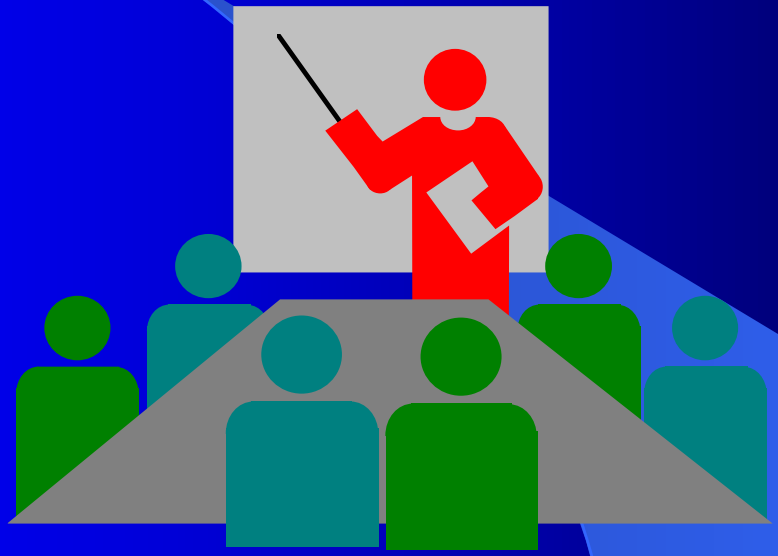
软件体系结构



二0一九年六月

第七章 基于软件体系结构的软件开发

7.1 方法一

7.2 方法二

7.3 方法三

7.4 测试

7.1 方法一

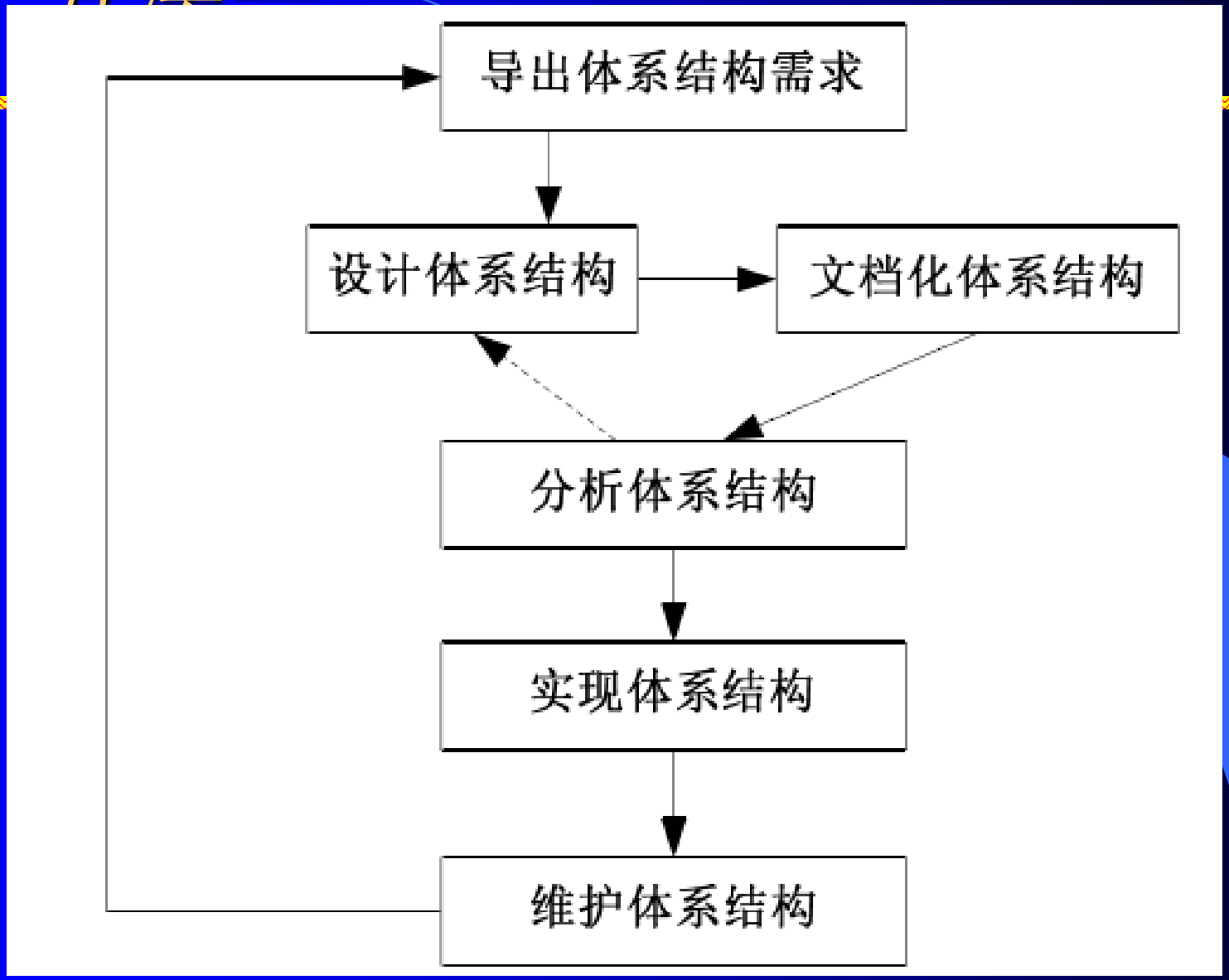
Felix Bachmann等人提出了一种基于体系结构的软件开发方法，该方法将以软件体系结构为核心的方法应用到软件产品的开发中，其研究领域包括：如何定义和表达体系结构；需求的收集、建模及其与体系结构的联系；构件的开发及其与体系结构的联系；体系结构与传统系统的联系；体系结构和产品规划的联系等。

7.1 方法一

该软件设计方法的开发过程主要由以下几部分组成：

（1）导出体系结构需求
（2）设计体系结构
（3）文档化体系结构
（4）分析体系结构
（5）实现体系结构
（6）维护体系结构

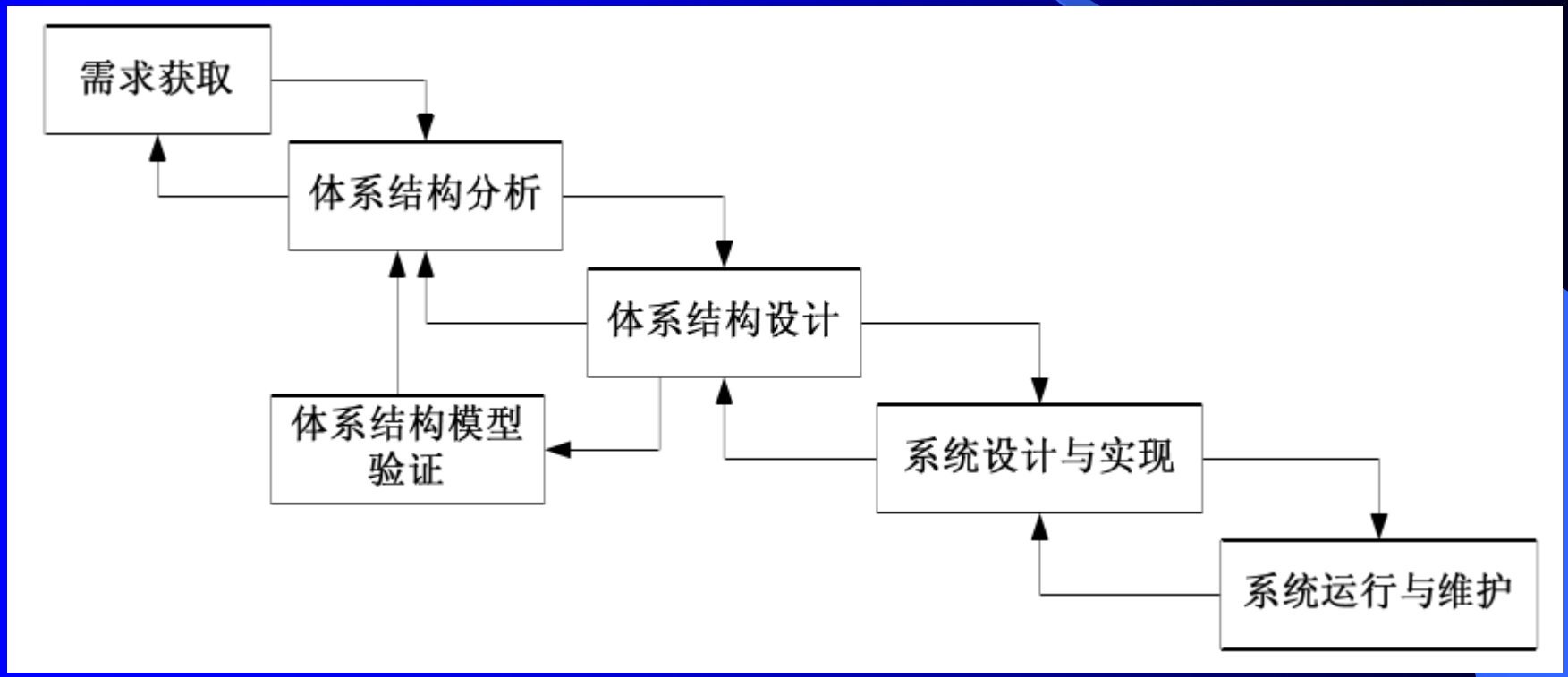
7.1 方法一



7.1 方法一

在以上6个步骤中，其中设计、文档化和分析这3个步骤构成了一个迭代的过程，一旦达成一个可接受的体系结构，它紧接着就被实现，并必须得到维护。该设计方法将软件体系结构作为一个整体贯穿整个软件的生命周期，对于一些大型软件系统得开发十分有利，它可以提高软件的复用粒度。但该方法在软件的运行和维护阶段没有将体系结构进行现行的表示，难以进行软件运行时的行为监控，不利于软件的动态演化。

7.2 方法二：体系结构驱动的过程模型



7.2 方法二：体系结构驱动的过程模型

（1）需求获取

需求获取阶段的目标就是了解用户需要什么，并将用户的这种需求传递给开发人员，让开发人员理解用户的需求，并能将需求得以实现。

在需求获取阶段，体系结构驱动的软件开发方法较传统的软件设计开发不同，它在一开始就从体系结构层面将软件需求的动态变更考虑进去了，有利于软件的后续演化。

7.2 方法二：体系结构驱动的过程模型

（2）体系结构分析

体系结构分析的主要目的是将问题空间中的用户需求映射到问题解空间中的软件体系结构。在该阶段主要涉及到两个任务：领域分析和构件分析。

1）领域分析

领域分析主要是使用已经存在的解决方案来解决与当前领域相似的共同问题，它着重于提取用户需求以及为当前问题设计解决方案。

7.2 方法二：体系结构驱动的过程模型

（2）体系结构分析

2）构件分析

该阶段主要负责构建一个软件体系结构的概念视图，该视图从一个较高的抽象层次对用户的功能需求进行描述。具体可以将该任务细分为两个活动：一个是对整个软件系统进行功能分解，我们称之为系统的分解活动，它主要来自于问题空间，同时也独立于软件系统的实现；另一个任务就是区分问题空间中各个实体之间的交互。在整个设计过程中，这两个活动需要以迭代的方式进行。

7.2 方法二：体系结构驱动的过程模型

（3）体系结构设计

体系结构设计阶段主要以用户需求为基础，并结合上一步的体系结构分析来完成体系结构的设计，它在传统的软件开发过程模型中可以被认为是进行问题解空间的描述。它所包含的主要任务有：选择所需要的构件、识别体系结构中潜在的失配部分、详细描述体系结构配置以及构造用户场景。整个软件体系结构的设计过程是以上四步不断迭代的过程。

7.2 方法二：体系结构驱动的过程模型

（4）体系结构验证

主要任务就是验证体系结构模型的完整性、正确性和一致性。

（5）设计和实现

在体系结构驱动的过程模型中，设计和实现阶段主要完成对构件的详细设计工作，以及利用封装机制来实现构件的重用，解决体系结构的失配问题。

（6）运行和维护

7.3 方法三--基于体系结构的软件开发模型

基于体系结构的软件开发模型（ABSD）把整个基于体系结构的软件过程划分为体系结构需求、设计、文档化、复审、实现、演化等6个子过程。

7.3 方法三--基于体系结构的软件开发模型

（1）体系结构需求

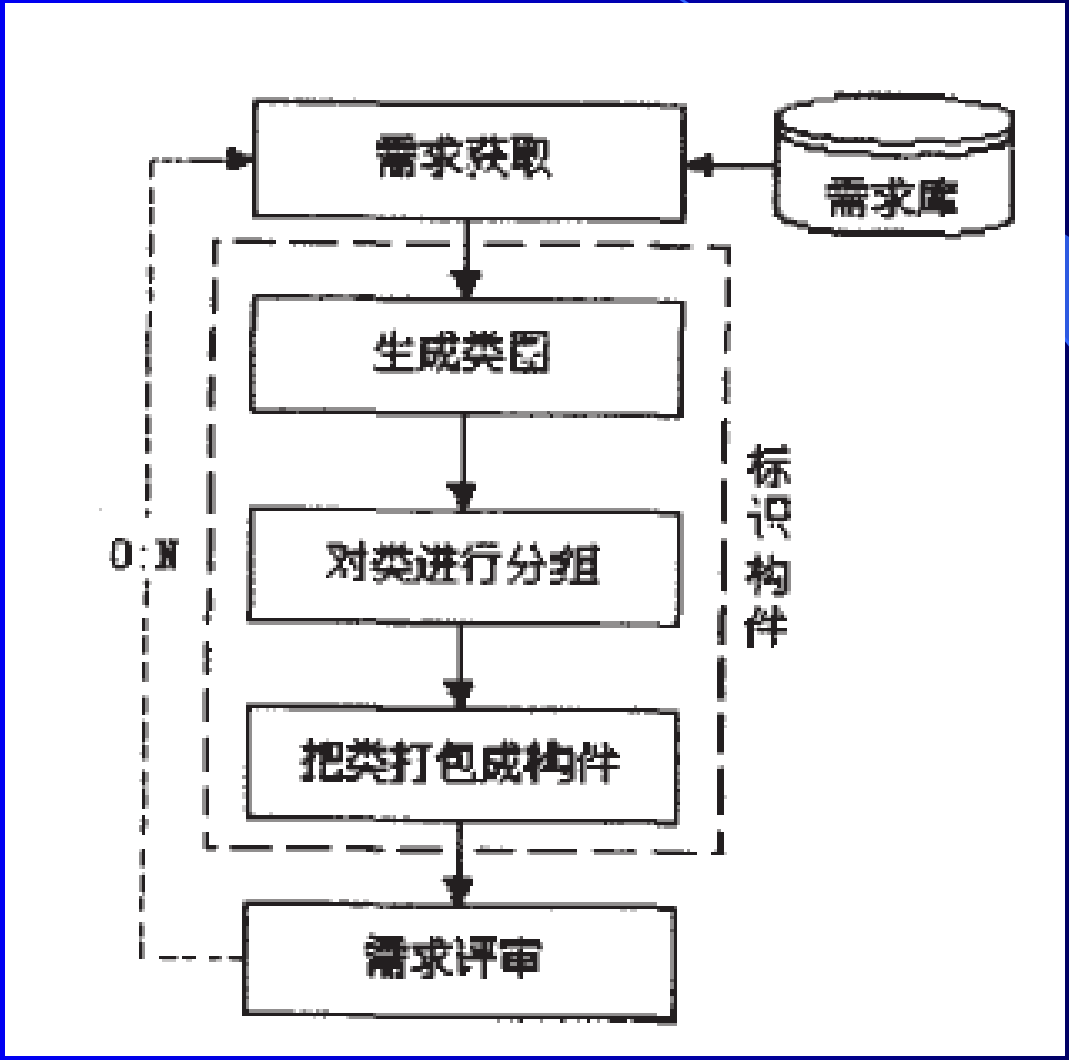
需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。

体系结构需求受技术环境和体系结构设计师的经验影响。

需求过程主要是获取用户需求，标识系统中所要用到的构件。

如果以前有类似的系统体系结构的需求，可以从需求库中取出，加以利用和修改，以节省需求获取的时间，减少重复劳动，提高开发效率。

7.3 方法三--基于体系结构的软件开发模型



7.3 方法三--基于体系结构的软件开发模型

1）需求获取

体系结构需求一般来自三个方面，分别是系统的质量目标、系统的业务目标和系统开发人员的业务目标。

软件体系结构需求获取过程主要是定义开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足业务上的功能需求。与此同时，还要获得软件质量属性，满足一些非功能需求。

7.3 方法三--基于体系结构的软件开发模型

2）标识构件

该过程为系统生成初始逻辑结构，包含大致的构件。这一过程又可分三步来实现。

第一步，生成类图。

第二步，对类进行分组：在生成的类图基础上，使用一些标准对类进行分组可以大大简化类图结构，使之更清晰。一般是，与其他类隔离的类形成一个组，由概括关联的类组成一个附加组，由聚合或合成关联的类也形成一个附加组。

第三步，把类打包成构件：把在第二步得到的类簇打包成构件，这些构件可以分组合并成更大的构件。

7.3 方法三--基于体系结构的软件开发模型

3）需求评审

组织一个由不同代表（如分析人员、客户、设计人员、测试人员）组成的小组，对体系结构需求及相关构件进行仔细审查。

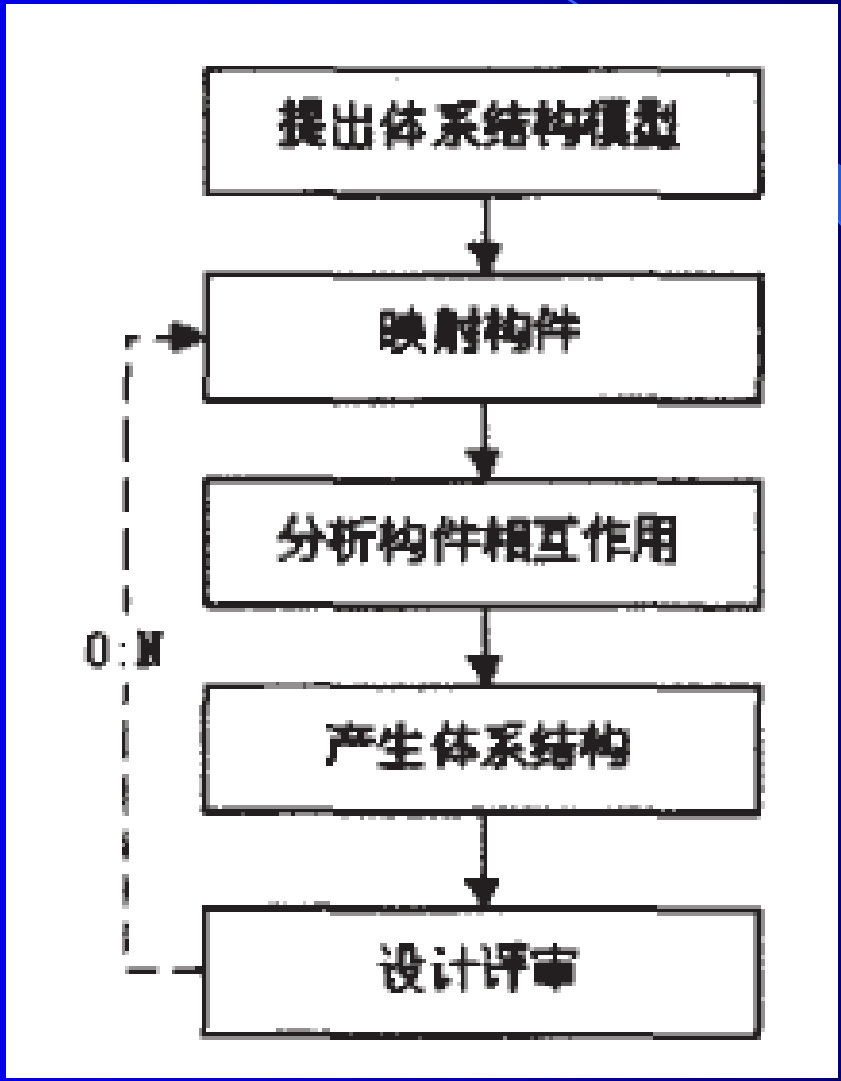
审查的主要内容包括所获取的需求是否真实反映了用户的要求，类的分组是否合理，构件合并是否合理等。

7.3 方法三--基于体系结构的软件开发模型

（2）体系结构设计

体系结构需求用来激发和调整设计决策，不同的视图被用来表达与质量目标有关的信息。体系结构设计是一个迭代过程，如果要开发的系统能够从已有的系统中导出大部分，则可以使用已有系统的设计过程。

7.3 方法三--基于体系结构的软件开发模型



7.3 方法三--基于体系结构的软件开发模型

1）提出软件体系结构模型：在建立体系结构的初期，选择一个合适的体系结构风格是首要的。在这个风格基础上，开发人员通过体系结构模型，可以获得关于体系结构属性的理解。此时，虽然这个模型是理想化的（其中的某些部分可能错误地表示了应用的特征），但是，该模型为将来的实现和演化过程建立了目标。

7.3 方法三--基于体系结构的软件开发模型

2）把已标识的构件映射到软件体系结构中：把在体系结构需求阶段已标识的构件映射到体系结构中，将产生一个中间结构，这个中间结构只包含那些能明确适合体系结构模型的构件。

3）分析构件之间的相互作用：为了把所有已标识的构件集成到体系结构中，必须认真分析这些构件的相互作用和关系。

7.3 方法三--基于体系结构的软件开发模型

4）产生软件体系结构：一旦决定了关键构件之间的关系和相互作用，就可以在第2）阶段得到的中间结构的基础上进行细化。

5）设计评审：一旦设计了软件体系结构，必须邀请独立于系统开发的外部人员对体系结构进行评审。

7.3 方法三--基于体系结构的软件开发模型

（3）体系结构文档化

绝大多数的体系结构都是抽象的，由一些概念上的构件组成。例如，层的概念在任何程序设计语言中都不存在。因此，要让系统分析员和程序员去实现体系结构，还必须把体系结构进行文档化。文档是在系统演化的每一个阶段，系统设计与开发人员的通讯媒介，是为验证体系结构设计和提炼或修改这些设计（必要时）所执行预先分析的基础。

7.3 方法三--基于体系结构的软件开发模型

体系结构文档化过程的主要输出结果是体系结构需求规格说明和测试体系结构需求的质量设计说明书这两个文档。生成需求模型构件的精确形式化的描述，作为用户和开发者之间的一个协约。

软件体系结构的文档要求与软件开发项目中的其他文档是类似的。文档的完整性和质量是软件体系结构成功的关键因素。文档要从使用者的角度进行编写，必须分发给所有与系统有关的开发人员，且必须保证开发者手上的文档是最新的。

7.3 方法三--基于体系结构的软件开发模型

（4）体系结构复审

体系结构设计、文档化和复审是一个迭代过程。从这个方面来说，在一个主版本的软件体系结构分析之后，要安排一次由外部人员（用户代表和领域专家）参加的复审。

7.3 方法三--基于体系结构的软件开发模型

复审的目的是标识潜在的风险，及早发现体系结构设计中的缺陷和错误，包括体系结构能否满足需求、质量需求是否在设计中得到体现、层次是否清晰、构件的划分是否合理、文档表达是否明确、构件的设计是否满足功能与性能的要求等等。

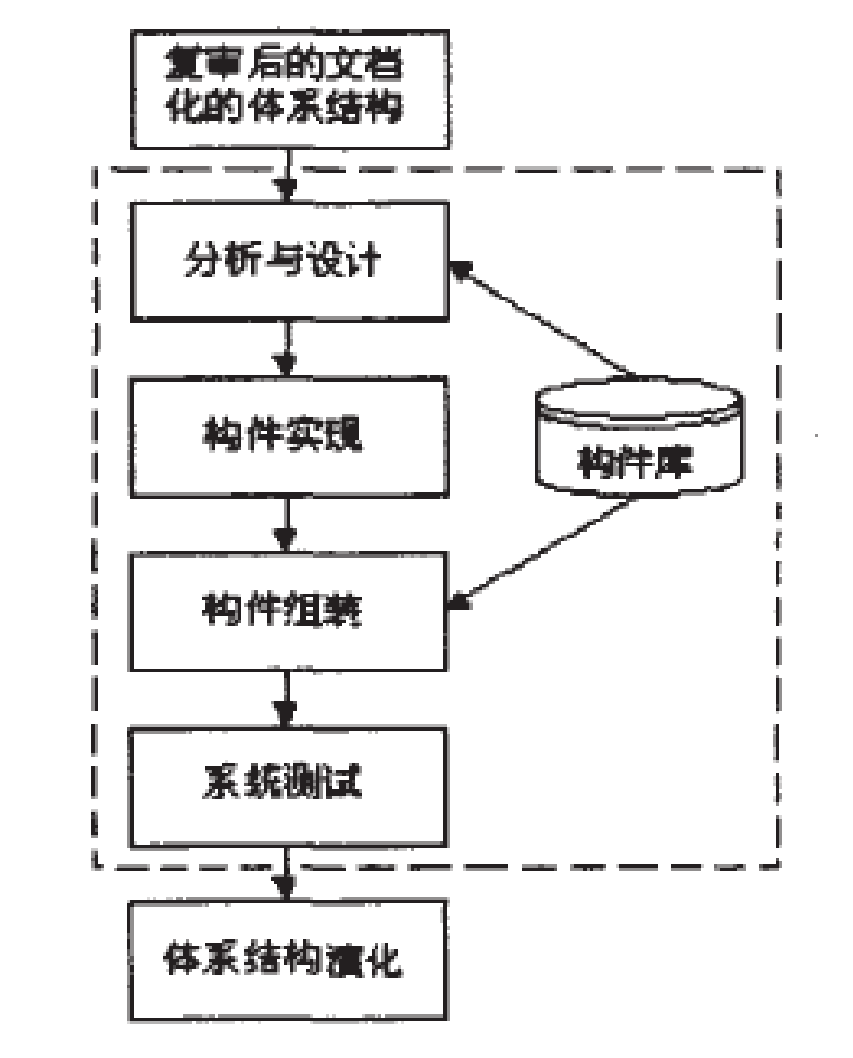
由外部人员进行复审的目的是保证体系结构的设计能够公正地进行检验，使组织的管理者能够决定正式实现体系结构。

7.3 方法三--基于体系结构的软件开发模型

（5）体系结构实现

所谓“实现”就是要用实体来显示出一个软件体系结构，即要符合体系结构所描述的结构性设计决策，分割成规定的构件，按规定方式互相交互。

7.3 方法三--基于体系结构的软件开发模型



7.3 方法三--基于体系结构的软件开发模型

图中的虚框部分是体系结构的实现过程。整个实现过程是以复审后的文档化的体系结构说明书为基础的，每个构件必须满足软件体系结构中说明的对其他构件的责任。这些决定即实现的约束是在系统级或项目范围内作出的，每个构件上工作的实现者是看不见的。

在体系结构说明书中，已经定义了系统中的构件与构件之间的关系。因为在体系结构层次上，构件接口约束对外唯一地代表了构件，所以可以从构件库中查找符合接口约束的构件，必要时开发新的满足要求的构件。

7.3 方法三--基于体系结构的软件开发模型

然后，按照设计提供的结构，通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成。

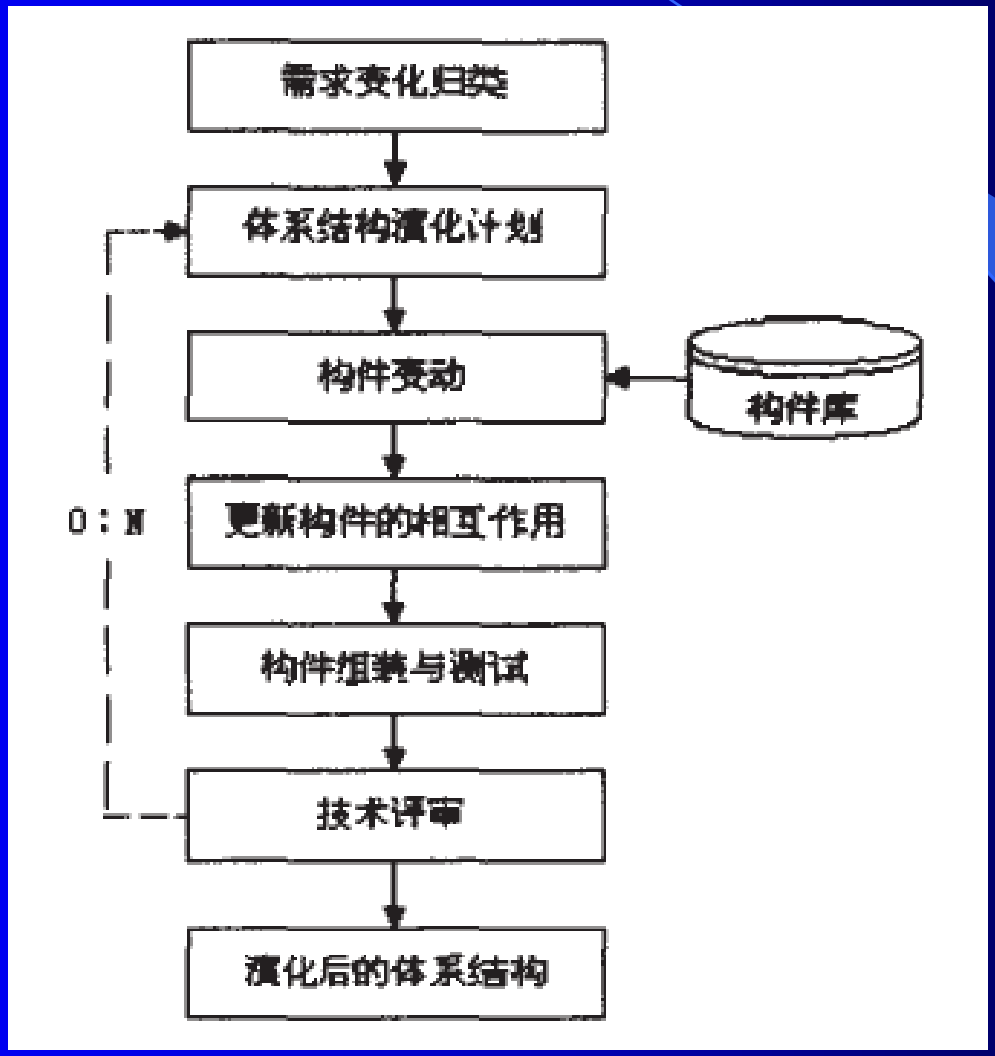
最后一步是测试，包括单个构件的功能性测试和被组装应用的整体功能和性能测试。

7.3 方法三--基于体系结构的软件开发模型

（6）体系结构演化

在构件开发过程中，最终用户的需求可能还有变动。在软件开发完毕，正常运行后，由一个单位移植到另一个单位，需求也会发生变化。在这两种情况下，就必须相应地修改软件体系结构，以适应新的变化了的软件需求。

7.3 方法三--基于体系结构的软件开发模型



7.3 方法三--基于体系结构的软件开发模型

体系结构演化是使用系统演化步骤去修改应用，以满足新的需求。主要包括以下几个步骤：

1）需求变动归类：首先必须对用户需求的变化进行归类，使变化的需求与已有构件对应。对找不到对应构件的变动，也要作好标记，在后续工作中，将创建新的构件，以对应这部分变化的需求。

2）制订体系结构演化计划：在改变原有结构之前，开发组织必须制订一个周密的体系结构演化计划，作为后续演化开发工作的指南。

7.3 方法三--基于体系结构的软件开发模型

3）修改、增加或删除构件：在演化计划的基础上，开发人员可根据在第1）步得到的需求变动的归类情况，决定是否修改或删除存在的构件、增加新构件。最后，对修改和增加的构件进行功能性测试。

4）更新构件的相互作用：随着构件的增加、删除和修改，构件之间的控制流必须得到更新。

5）构件组装与测试：通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成，形成新的体系结构。然后对组装后的系统整体功能和性能进行测试。

7.3 方法三--基于体系结构的软件开发模型

6）技术评审：对以上步骤进行确认，进行技术评审。评审组装后的体系结构是否反映需求变动，符合用户需求。如果不符合，则需要在第2）到第6）步之间进行迭代。

7）产生演化后的体系结构：在原来系统上所作的所有修改必须集成到原来的体系结构中，完成一次演化过程。

7.4 基于体系结构描述的软件测试

（1）测试内容

软件体系结构测试与程序测试有所不同，它是检查软件设计的适用性，这种测试不考虑软件的实现代码，所以基于实现和说明的程序测试方法对软件体系结构测试并不适用。

与传统的软件测试一样，基于体系结构的软件测试也需要研究测试内容、测试准则、测试用例、测试充分性及测试方法等问题。

7.4 基于体系结构描述的软件测试

体系结构的描述必须满足一个基本的要求，即体系结构描述的各个部分必须相互一致，不能彼此冲突，因为体系结构主要关注系统的结构和组装，如果参与组装的各个部分之间彼此冲突，那么由此组装、精化和实现的系统一定不能工作。

因此，体系结构的分析和测试主要考虑：组件端口行为与连接器约束是否一致、兼容，单元间的消息是否一致、可达，相关端口是否可连接，体系结构风格是否可满足。

7.4 基于体系结构描述的软件测试

为了保证测试的充分性，必须对关联组件、连接器的所有端口行为和约束进行测试，即所有接口应该是连接的，数据流、控制流和命令应该是可达的，且在并发时应该保证无死锁发生。

7.4 基于体系结构描述的软件测试

（2）测试准则

在传统测试方法中，测试准则是基于实现和规约得到的，基于实现的测试准则是结构化的，它是利用软件的内部结构来定义测试数据以覆盖系统，主要包括控制流测试覆盖准则和数据流测试覆盖准则。1)控制流测试覆盖准则包括语句覆盖、分支覆盖、多条件覆盖、路径覆盖和决策支持覆盖等。2)数据流覆盖准则包括全定义、全使用和全定义-使用路径覆盖准则等。

7.4 基于体系结构描述的软件测试

近年来，在规约的基础上结构化准则得到了进一步的发展，并根据规约的语法、语义和结构定义了测试准则。软件体系结构描述语言在高层抽象层上的形式化的系统静态、动态特征及系统交互模型，为定义体系结构测试准则奠定了基础。

7.4 基于体系结构描述的软件测试

Zhenyi Jin，J.offutt等人针对软件体系结构描述，对通用的体系结构测试覆盖准则进行了分类：(1)迁移覆盖层：即图中的每一个迁移至少要取到一次；(2)全断言覆盖层：每一个判定和每一个条件至少取一次；(3)迁移对覆盖层：每一个输入迁移和每一个输出迁移必须被顺序取到；(4)全序列层：一个来自系统的完全状态迁移的序列。

7.4 基于体系结构描述的软件测试

随后，J.Offutt等人提出了一组基于软件体系结构的测试用例覆盖准则，即构件内部依赖关系覆盖准则、连接件内部迁移覆盖准则、N2C覆盖准则、C2N覆盖准则、直接N2N覆盖准则、间接N2N覆盖准则和全连接构件覆盖准则。但J.Offutt并未用文字对此准则做进一步说明。

7.4 基于体系结构描述的软件测试

D.J.Richardson，A.L.Woif等人对软件体系结构层的测试覆盖准则进行了研究，在CHAM模型的基础上定义了一组基于体系结构的测试标准：(1)全数据元素准则：要求在体系结构中定义的所有数据是可以通信的，即对每一个数据元素d，至少一个溶液包含一个具有d的分子；(2)全处理元素准则：要求所有处理元素是可执行的，即对每一个处理元素p，至少一个溶液包含一个具有p的分子；(3)全连接元素准则：要求所有通信通道和通信通道上的连接都被用到，即对每一个连接元素c，至少一个溶液包含一个具有c的分子；(4)全变换准则：要求所有变换至少测试一次，即对每一个变换规则T:S1—S2，至少有一条路径包含S1—S2；(5)全变换系统：要求所有不同“路径”是可测试的，或从初始溶液到终止溶液的不重复的变换序列是可测试的，该测试计划包含每一个不重复的反应序列；(6)全数据依赖准则：要求在一个数据元素是一个通信通道的输出且同时直接或间接地作为另一通信通道的输入的情况下，每一个交互序列都要覆盖，即对每一个这样的数据依赖(d，d’)，至少有一个路径包含该交互序列(d，d’)。

7.4 基于体系结构描述的软件测试

测试准则：测试应覆盖所有的组件及各个组件的接口（包括流入、流出该构件的数据类型一致性分析）、各个连接器的接口（包括流入、流出该连接件的数据流(控制流)的匹配性分析）、组件之间的直接连接、组件之间的间接连接。

对于不同的体系结构描述语言，测试准则有不同的演绎，对于特定的ADLs，可以从该准则中定义测试需求并据此生成测试用例。

7.4 基于体系结构描述的软件测试

（3）测试需求和测试用例的生成

实现完整测试的典型方法是利用测试准则定义测试需求，进而生成测试用例。参照文献，可定义以下几种测试路径。

1)组件或连接器内部消息的传递路径。

2)组件或连接器内部端口的执行顺序路径。

3)组件到连接器或连接器到组件的消息传递路径。

4)组件之间的直接连接路径。

5)组件之间的间接连接路径。

6)所有组件的连接路径。

7.4 基于体系结构描述的软件测试

在大多数的软件开发过程中，软件开发的成本和周期制约着软件的质量，三者之间的关系需要平衡。根据不同的体系结构抽象层次，测试准则和覆盖准则的力度也有所不同。

7.4 基于体系结构描述的软件测试

软件体系结构测试过程可以分为单元测试、集成测试和系统测试，单元测试是最底层的测试活动，指组件开发者对组件本身的测试，涉及的消息流是组件内部的消息，一般由组件开发者完成；集成测试的主要任务是测试组件之间的接口以保证组件能够交互，它将组件本身抽象为单元，并关注于组件间的消息传递，组件的交互行为可以通过形式化规约得到，因此这种测试可提前进行；系统测试的主要任务是测试整个系统能否正常运行，以保证系统符合其设计模型。

7.4 基于体系结构描述的软件测试

在不同阶段，测试关注的信息和特征也不相同，因此测试准则的级别也就不同，根据由低到高的逐步抽象过程，定义测试准则的级别。

在测试过程中，根据测试准则级别，可选择不同的测试路径，生成测试用例。

谢谢大家！

再见！