基于消息的SOA企业解决方案

第五章 SOA中的ESB场景及解决方案模式

本章内容

1. 驱动ESB体系结构和设计决策的问题
2. ESB解决方案模式
3. SOA中的ESB场景及解决方案

本章内容

1. 驱动ESB体系结构和设计决策的问题
2. ESB解决方案模式
3. SOA中的ESB场景及解决方案

驱动ESB体系结构和设计决策的问题

- 为了确定用于ESB的合适解决方案模式和实现技术，需要对特定的ESB功能需求进行详细的分析。下面的14个问题旨在帮助进行这一过程。
  - 1. 现有数据接口
  - 2. 业务数据模型
  - 3. 开放标准的使用
  - 4. 对基本或高级通信协议的支持
  - 5. 通过现有系统对数据传递格式的修改
  - 6. 通过新技术公开现有服务
  - 7. 对遗留系统的访问
  - 8. 现有 EAI 技术
  - 9. 需要的保护措施
  - 10. 需要提供多少服务和需要的一致性程度
  - 11. 公司内部及与其他公司间的互操作
  - 12. 对服务编排的需求
  - 13. 服务级需求的基础架构级支持
  - 14. 点到点或端到端安全模型的使用

驱动ESB体系结构和设计决策的问题

- 问题1：现有功能及其数据接口是否与您想要提供的服务相匹配？您是否能够修改或聚合应用程序？
  - 如果不可以，则转换或聚合功能就需要由适配器或 ESB 体系结构来提供，或者不得不由服务客户端来完成。
- 问题2：服务是否可以以一些通用业务数据模型的形式公开？如果可以，则实现这些服务的系统是否已经支持该模型？或者说可以使它们这样做？
  - 如果服务不可以，则转换或聚合功能就需要由适配器或 ESB 体系结构来提供。

驱动ESB体系结构和设计决策的问题

- 问题3：是否需要开放标准？或者是否可以通过EAI中间件来实现适当的互操作性？如果需要开放标准的话，则哪些开放标准是适合的？
  - 虽然使用开放标准是实现互操作性的一种途径，但专有的 EAI 中间件也具有高度的互操作性，并且往往要成熟得多。另外，许多组织还拥有广泛的现有基础架构，在一些场景中，它们可能会使得开放标准的作用几近于无。
  - 在需要开放标准的场景中，Web服务也许是这些情况下最明显的选择。不过，您也可以应用Java Messaging Service(JMS)、JDBC、基本XML或者一些其他的技术（比如EDI或业界通用的XML格式。
  - 在实践中，不能总是假定相同标准的不同实现之间具有互操作性。

驱动ESB体系结构和设计决策的问题

- 问题4：是否需要支持基本通信协议及标准（例如 WMQ、SOAP、WSDL）？或者需要更高级的功能（例如 Web服务安全性（WS-Security）、Web 服务事务（WS-Transaction）等等）？
  - 对支持更复杂标准的需求将对实现技术的选择加以更严格的约束，并且可能意味着使用还不成熟的技术。

驱动ESB体系结构和设计决策的问题

- 问题5：当果考虑更改现有的基础架构使用的消息格式和协议（包括可能采用开放标准）时，需要在整个现有的基础架构中进行这些更改吗？或者很快就要应用新的消息格式和协议吗？如果正在使用或考虑使用 EAI 技术，该技术是否有自己的内部格式？或者它能够将开放标准处理为内部格式吗？
  - 开放标准的任何应用都是受扩展访问的需求驱动的，因此它们对现有基础架构的接口的可用性比在内部使用的这样的标准更重要。
  - 如果需要在内部使用特定的格式、技术或标准，这会给实现技术的选择带来限制。

驱动ESB体系结构和设计决策的问题

- 问题6：将作为服务公开的系统实现功能支持所需的技术或开放标准（比如 SOAP、JMS或 XML）吗？
  - 如果不支持，ESB基础架构或适配器将需要在所需的开放标准和服务提供者支持的格式之间进行转换的功能。
- 问题7：在需要访问遗留系统的情况下，通过使用更新的基于 XML 的技术，可以直接支持（例如 CICS SOAP 支持）遗留系统的可用性吗？是否需要单独的适配器？遗留平台是否支持 XML 处理？如果支持，这种处理是否可以灵活地使用平台功能？
  - 如果因为这其中的任何原因而导致所需的 SOAP 或 XML 功能对遗留平台不可用，则需要在适配器（比如s J2C Connector Architecture（JCA）或 WebSphere Business Integration Adaptors）、集成层或 ESB 基础架构中使用适当的转换功能。

驱动ESB体系结构和设计决策的问题

- 问题8：如果 EAI 技术已经可用，它是否使用适当的功能或接口粒度将服务作为消息流实现？它支持哪些连接性协议（例如 JCA、SOAP、WebSphere MQ 以及 Java 远程方法调用（Java Remote Method Invocation））？
  - 如果现有消息流不提供所需要的服务，则需要另外的流程来执行转换。如果 EAI 技术不直接支持所需的标准，就需要添加一个网关组件。
- 问题9：应该从服务客户端通道以工作负荷缓冲、安全、登录等形式提供给服务提供者系统什么保护措施？
  - 这种缓冲通常是 ESB 基础架构的一个角色，并且定义它所需要一些功能。如果特定的服务提供者系统（例如遗留事务系统（legacy transactional systems））需要额外的保护，则可以使用专用集成层。

驱动ESB体系结构和设计决策的问题

- 问题10：应该实现多少服务？实现的什么方面应该在这些服务中保持一致？如何实施一致性（可能在多个平台上和多个应用程序中）？
  - 如果只需要非常少的服务，简单的点到点（point-to-point）集成模型可能比较适合。然而，如果需要更多的服务或者过一段时间以后可能还是如此，则添加控制点（比如由 ESB 提供的）就变得愈加有益。
- 问题11：服务交互包含在组织内部，还是有一些交互在组织外部？
  - 这常常是不同于在单个组织中实现的 ESB 基础架构的一种情况，因为对安全和服务路由的需求可能与外部可用的服务不同。

驱动ESB体系结构和设计决策的问题

- 问题12：是否需要服务编排？服务编排是否涉及短期（short-lived）或长期（long-lived）（换句话说就是有状态的）流程，还是两者都涉及？它们是否包含人工活动？
  - 在这些需求构成业务功能的情况下，应该在与 ESB 分离的 Service Choreographer 组件中实现编排。关于是支持长期有状态流程还是支持人工活动的需求将对实现技术的选择产生限制。

驱动ESB体系结构和设计决策的问题

- 问题13：基础架构应该支持什么样的服务级需求（例如，服务响应时间、吞吐量、可用性等等）？随着时间的推移，需要如何对其进行扩展？
  - 一些候选的ESB实现技术相对较新，并且可能仅仅在有限的服务级进行过测试。同样，由于相关的开放标准不是最近制订就是正在兴起的，所以在更多的既定产品和技术中对它们的支持也是新出现的。
  - 在可以预见的未来，关键的体系结构决策将专注于特定开放标准优点的平衡，针对服务级需求的新兴或成熟的产品技术支持这些开放标准。制订这些即时决策需要考虑到有些标准和支持它们的产品是相对成熟的（例如 XML、SOAP等等），有些（例如 Web 服务安全（WS-Security））比较新，还有一些（例如 Web 服务事务）是正在兴起的。
  - 标准的优点之间的权衡和经过验证的服务级特征往往驱动一个结合了ESB与SOA体系结构中适应标准的、专有的或自定义技术的混合方法。

驱动ESB体系结构和设计决策的问题

- 问题14：是否需要点到点（point-to-point）或端到端（end-to-end）安全模型（例如，ESB 是否可以简单的对服务请求授权，还是需要将请求者的身份或其他凭证传递给服务提供者）？是否需要使用应用程序或遗留安全系统来集成服务安全模型？
  - 如果点到点安全性是可接受的，则许多现有解决方案（例如 SSL 、对数据库访问的 J2EE 安全性、适配器安全模型等等）就能够得到应用。如果需要端到端安全性，则 Web 服务安全标准就成为可能，提供所有相关的系统来支持它。换句话说，您可以使用带有客户端消息头的客户端模型，或者传送像应用程序数据这样的安全信息。

本章内容

1. 驱动ESB体系结构和设计决策的问题
2. ESB解决方案模式
3. SOA中的ESB场景及解决方案

ESB解决方案模式

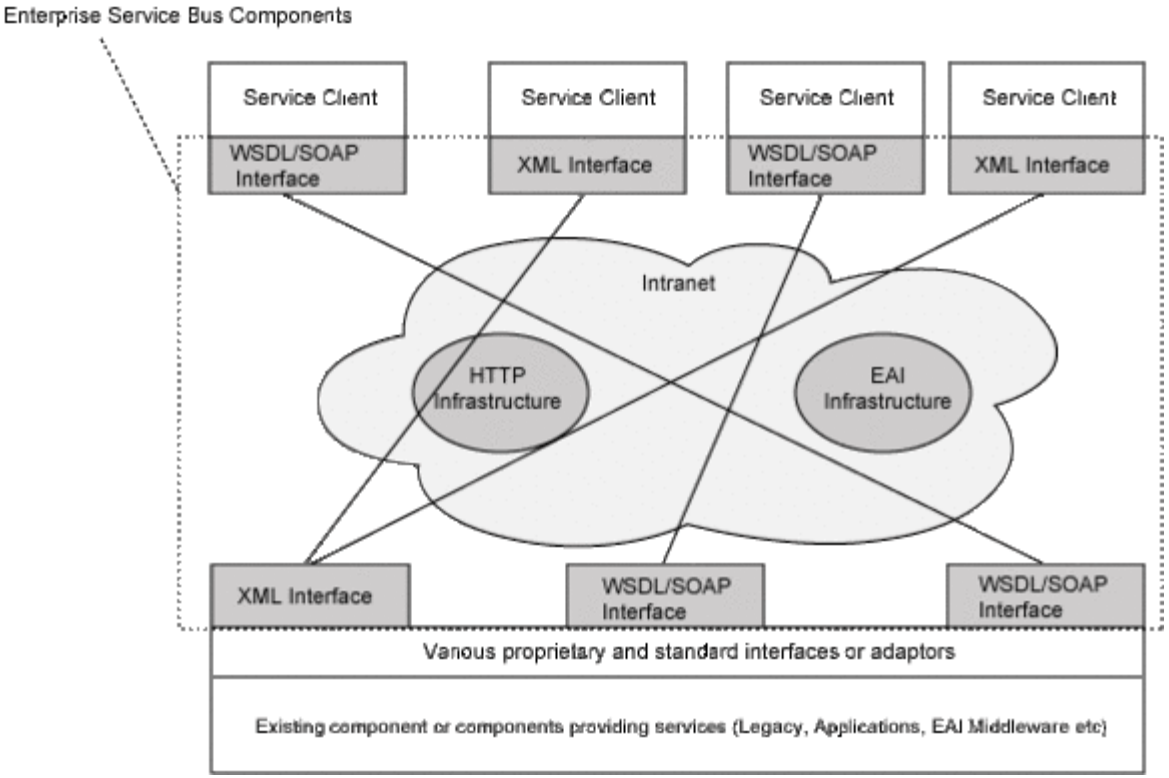
- 本节描述了ESB实现方式的解决方案模式，除了基本适配器（Basic Adaptors）模式以外，其他的都是简单的点到点（P2P）解决方案。每个模式都提出了不同的使用现行技术的实现选择，同时也做出了正反两方面以及移植方面的考虑。
  - 基本适配器（Basic Adaptors）
  - 服务网关
  - Web服务兼容的代理（Web Service-compliant Broker）
  - 面向服务架构的企业应用集成基础架构（EAI Infrastructure for SOA）
  - 服务编排（Service Choreographer）
  - 完整的面向服务架构的基础架构（Full SOA Infrastructure）

基本适配器解决方案模式

- 这种解决方案通过封装器或适配器技术来实现简单的点到点（P2P）服务集成，而不是真正的ESB。这种技术通过WSDL定义的SOAP访问或者其他可互操作的产品技术（比如WMQ）来实现集成。如果这些技术没有为服务接口定义（比如WSDL）提供本地模型，那么将需要使用自定义模型来实现SOA 规范。
- 虽然设计比较简单，但是从该模式中可以获得的好处却不可低估。
- 现在已经有各种各样的适配器可用，而且也可以通过开发工具或运行时技术来创建新的适配器。并能使其提供对Web服务规范和企业应用集成中间件的支持。也可以提供给多种不同类型的系统，包含分布式应用服务器、企业遗留系统以及 Commercial Off-the-Shelf 软件包。

基本适配器解决方案模式

- 一般的基本适配器解决方案包含了使用现有的 HTTP和EAI中间件基础架构来支持新的集成。如果用HTTP来作为通信协议，或者使用某些Internet 兼容的EAI技术（比如 MQ internet pass-thru），那么该解决方案同样可以应用于外部场景。

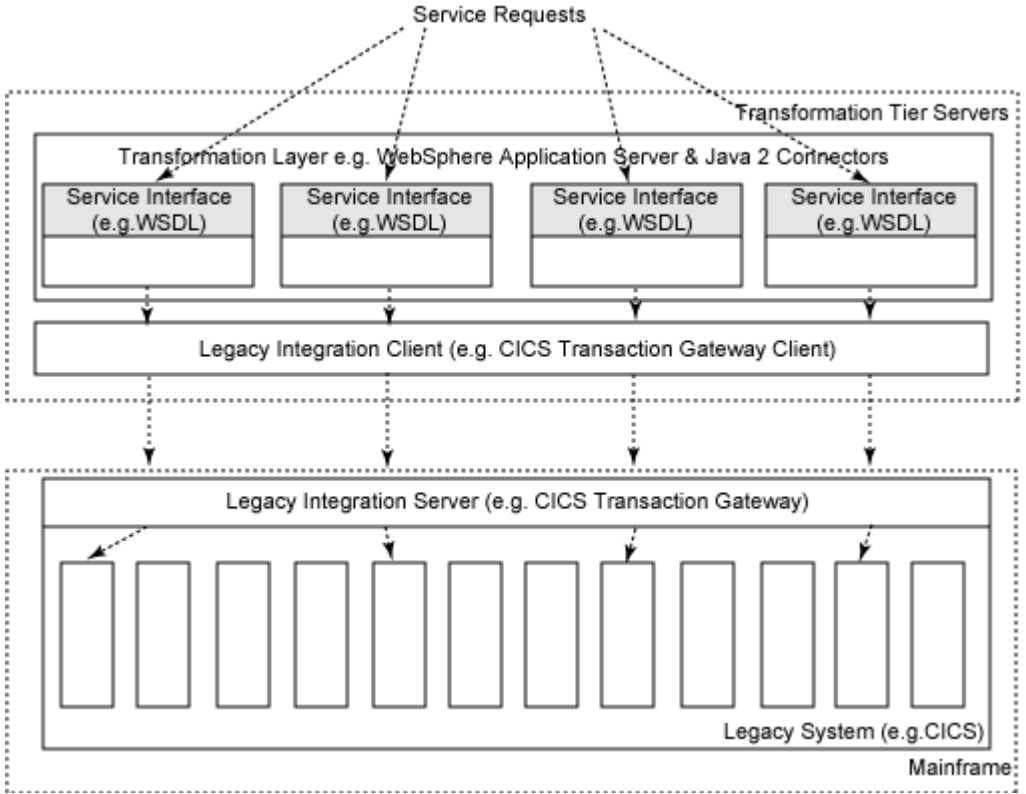


基本适配器的实现技术

- 1.使用遗留系统或应用程序直接提供的 SOAP 或 EAI 功能。例如，IBM CICS 目前直接提供对 SOAP 的支持，以及许多系统和应用程序包能够支持 MQ 或 SOAP 接口。
- 2.如果用于提供访问的应用程序是用户自己开发的应用程序且运行于应用服务器环境，或者只要应用服务器运行时环境和应用开发环境能够用来给应用程序添加封装器。例如，WebSphere Studio Application Developer 能够用来给部署于 WebSphere Application Server（Application Server）的 J2EE 应用程序添加 XML、SOAP 或 MQ 支持。

基本适配器的实现技术

- 3.如果这种支持不可用或不合适（例如， 如果XML转换不适合用来处理现有平台上的资源），那么可能需要其他的体系结构层。这可能是托管了与应用程序或遗留系统集成的适配器的应用服务器层。例如，Application Developer Integration Edition提供了JCA连接器技术来访问遗留系统，并通过 WebSphere运行时环境为其提供了J2EE和Web服务接口。



基本适配器的实现技术

- 4. 如果使用开发工具来创建自己的封装器，那么可以增强工具提供的功能：通过创建一个框架或一组实用工具类来执行通用任务，比如安全性、日志纪录等等。然而，这种方法可能引起范围蠕变（scope creep），并最终导致该框架实际上变成了用户开发的服务网关或Web 服务兼容代理。当定义框架提议的功能时，需要注意验证开发和维护的成本是否合适，以及转换为更复杂的模式是更不合适的。

基本适配器模式分析

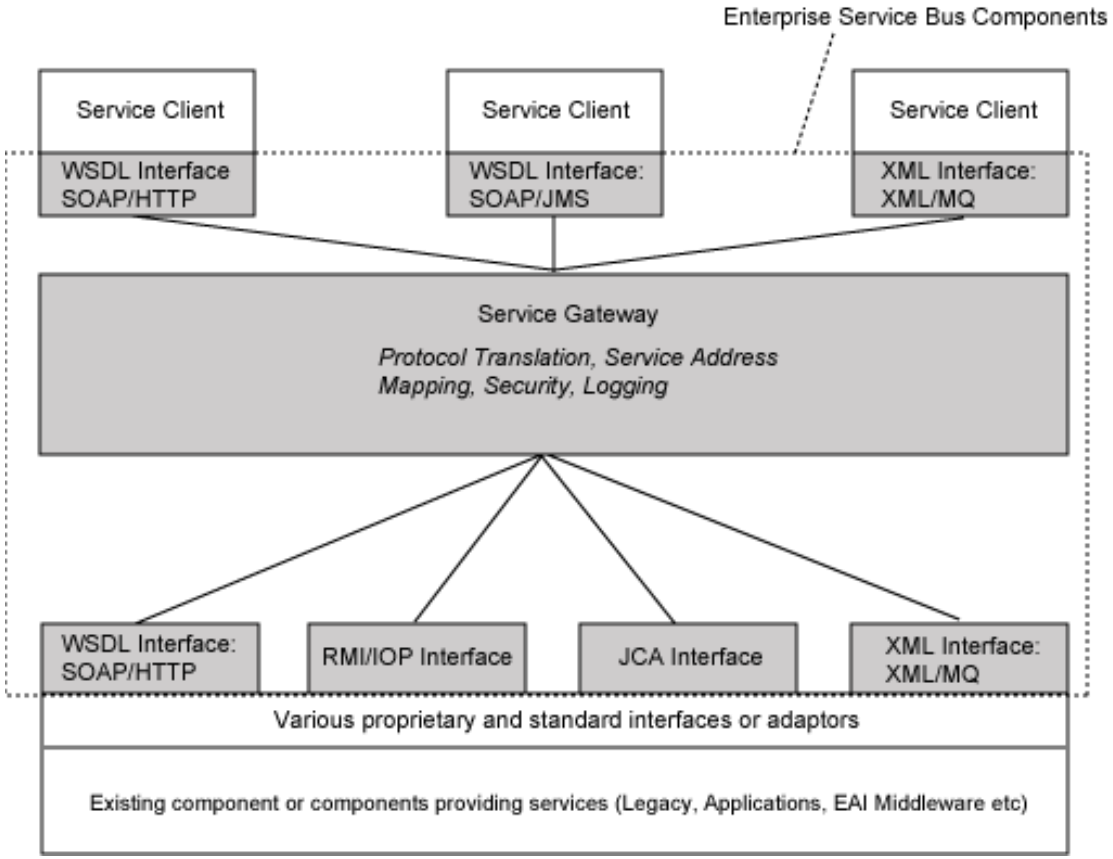
- 从正面来说，这种解决方案模式对新的基本架构的需求最低或是根本不需要，并且使用的都是广泛支持的各种规范和技术。
- 从反面来说，像安全、管理等功能都交给了应用程序或单个封装器的实现来处理。
- 由于该模式基于使用协同操作技术和开放式标准，因此将该模式移植到更复杂的体系结构也就相对比较简单。
- 替代模式
  - 如果以上均不能满足集成的需求，或者存在一些附加功能或服务质量需求，那么封装器方式就可能满足不了需求。如果是这样，从逻辑上说下一步应该是服务网关。如果需要更高级ESB功能，则Web服务兼容代理或EAI Infrastructure for SOA模式会比较适合。

服务网关解决方案模式

- 这种模式代表了一种基本的ESB实现，接近于最简功能ESB。服务网关一般通过SOAP/HTTP、MQ、JMS等来支持客户端连接，但是也可以通过诸如JCA 或WebSphere业务集成适配器（WebSphere Business Integration Adaptors，WBIA）来对服务提供者支持更广泛的集成。网关组件为服务路由、协议转换以及安全担当着中央控制点的角色。
- 网关能够用来向客户端提供一致的服务命名空间，并可以向服务提供授权模型，实际上这些服务是由完全不同的系统通过多种协议来提供的。当需要向外部合作伙伴公开服务时，网关所提供的这些功能便成为一个明显的需求。然而当需要对从应用程序到用多种系统和技术实现的功能的访问进行简化时，这些功能在单个企业内部也很有用。

服务网关解决方案模式

- 网关的一个关键功能是将客户端支持的服务协议转换为提供方支持的服务协议。这些协议可以包括 SOAP/HTTP、MQ 或 SOAP/JMS、JCA、RMI/IIOP 等。



服务网关的实现技术

- 1.使用打包好的网关技术，比如 WebSphere Application Server Network Deployment 或 WebSphere Business Integration Connection 提供的 Web Services Gateway。许多网关技术支持某些形式的中间件过滤器或处理器程序设计模型，以实现自定义增强功能。Web Services Gateway 提供了一些可配置的中间件功能。它也支持基于 XML 的远程过程调用 Java APIs（Java APIs for XML-based Remote Procedure Call ，JAX-RPC）规范中定义的 Web 服务请求/响应处理程序。
- 2.使用应用程序开发和最新应用服务器技术的运行时功能来实现自定义网关。这可能包含与前面基本适配器解决方案模式中所描述的相同类型的适配器。

服务网关的实现技术

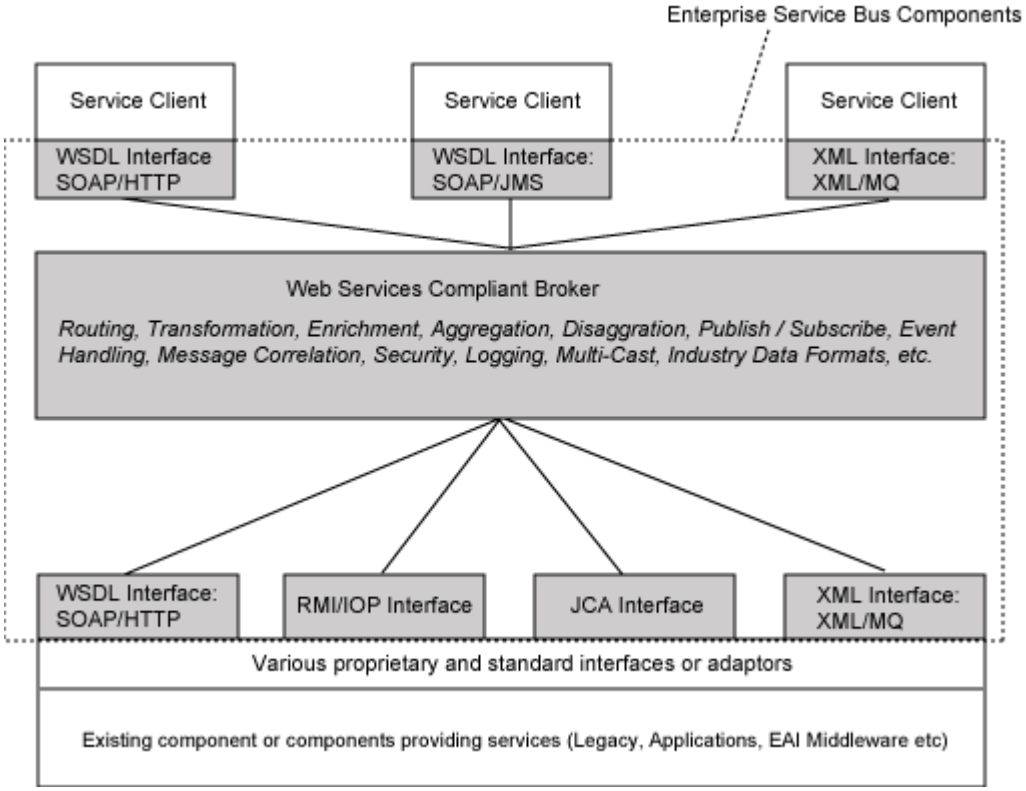
- 3.如果需要更高级的功能，就应该考虑更复杂高级的 EAI 中间件，比如 WebSphere Business Integration Message Broker。
- 4.这种模式的许多实现存在于遗留技术中，这些遗留技术通常没有使用 Web 服务技术。例如，许多组织构建了路由器事务，它对多种遗留事务提供了使用类似文本的（text-like）数据模型的简单接口。这种系统使用具有一些 XML 的可移植优点的自定义数据格式 ，从而有效地实现了网关模式。

服务网关模式分析

- 从正面来说，尽管一些网关技术必须在有适当弹性的方式下部署，但这种解决方案仍然能够包含最低功能基础架构。对互操作协议和开放标准的重视也使基础架构所涉及的方面得以简化。大多数网关技术与许多其他接口类型（比如 RMI/IIOP 和 JCA）进行协同操作的能力，也应该能够减少其他连接性技术的部署。
- 然而，网关技术往往限制了对请求/响应和发布/订阅服务的简单一对一映射的服务处理。更复杂高级的功能，比如消息转换、消息相关性、消息聚集等都可能超出了网关技术所能提供的功能之外，或需要在自定义场景中进行网关技术之外的开发工作。
- 大多数 ESB 技术认可网关模式及其相关功能。有了这一点，互操作协议和开放标准的使用、网关功能的简化等任何移植到更高级的 ESB 基础架构的问题都不会太困难。
- 替换模式
  - 最显而易见的替换模式是Web 服务兼容代理或EAI Infrastructure for SOA。当需求超出了网关所能提供的功能之外，或者超出了已打包的网关技术范围时，这些模式会比较适合。另一方面，如果实际涉及的服务非常少，那么简单的基本适配器解决方案可能比较合适。

Web服务兼容代理解决方案模式

- 这种解决方案代表了高级复杂的ESB实现，它提供了一个功能完整的EAI解决方案的所有功能，并且使用开放标准模型。通过特定场合的明确需求来定义需要什么级别的EAI功能，从而确定适合使用哪种EAI技术。



Web服务兼容代理的实现技术

- 1.最可能用于这种解决方案的实现技术是能提供合适的 Web 服务支持的 EAI 中间件，比如 WebSphere Business Integration Server。
- 2.如果 Web 服务支持主要是为了外部集成需要，则 EAI 中间件的专有特性可以在内部使用，并结合服务网关组件的使用来添加 Web 服务支持。

Web服务兼容代理模式分析

- 这种实现方式的优点是在开放标准模型内提供了丰富的功能。然而，虽然 EAI 中间件技术是成熟的，但是该解决方案支持的开放标准，尤其是更高级的 Web 服务标准，比如 Web 服务策略（WS-Policy）和 Web 服务事务（WS-Transaction）还不够成熟。因此该场景最主要的缺点仅仅是不能简单地对所有情况都适用。
- 替换模式
  - 如果不能提供适当的 Web 服务支持，则服务总线（Service Bus）的需求可以通过EAI Infrastructure for SOA模式用更加专有或定制的方式来实现，也许要与服务网关组件相结合以添加 Web 服务接口。另外，如果开放标准是最关键的需求，而且一些 EAI 功能（比如转换和聚集）能够在别处完成（也许是在应用程序或适配器内），则服务网关模式可能更合适。

面向服务架构的企业应用集成基础架构

- 使用EAI技术（但不排除其他技术）并结合XML来构建自定义的SOA基础架构。只要服务接口已经明确定义并有合适的粒度，EAI中间件就能确保满足SOA的互操作性和位置无关性原则。
- 该模式的优点在于成熟的EAI技术全部的功能和性能都应用于SOA的灵活性上。这个优点既可以应用于为SOA实现新的且坚固稳定的基础架构，也可以应用于在现有基础架构上的实现符合 SOA原则的应用程序。

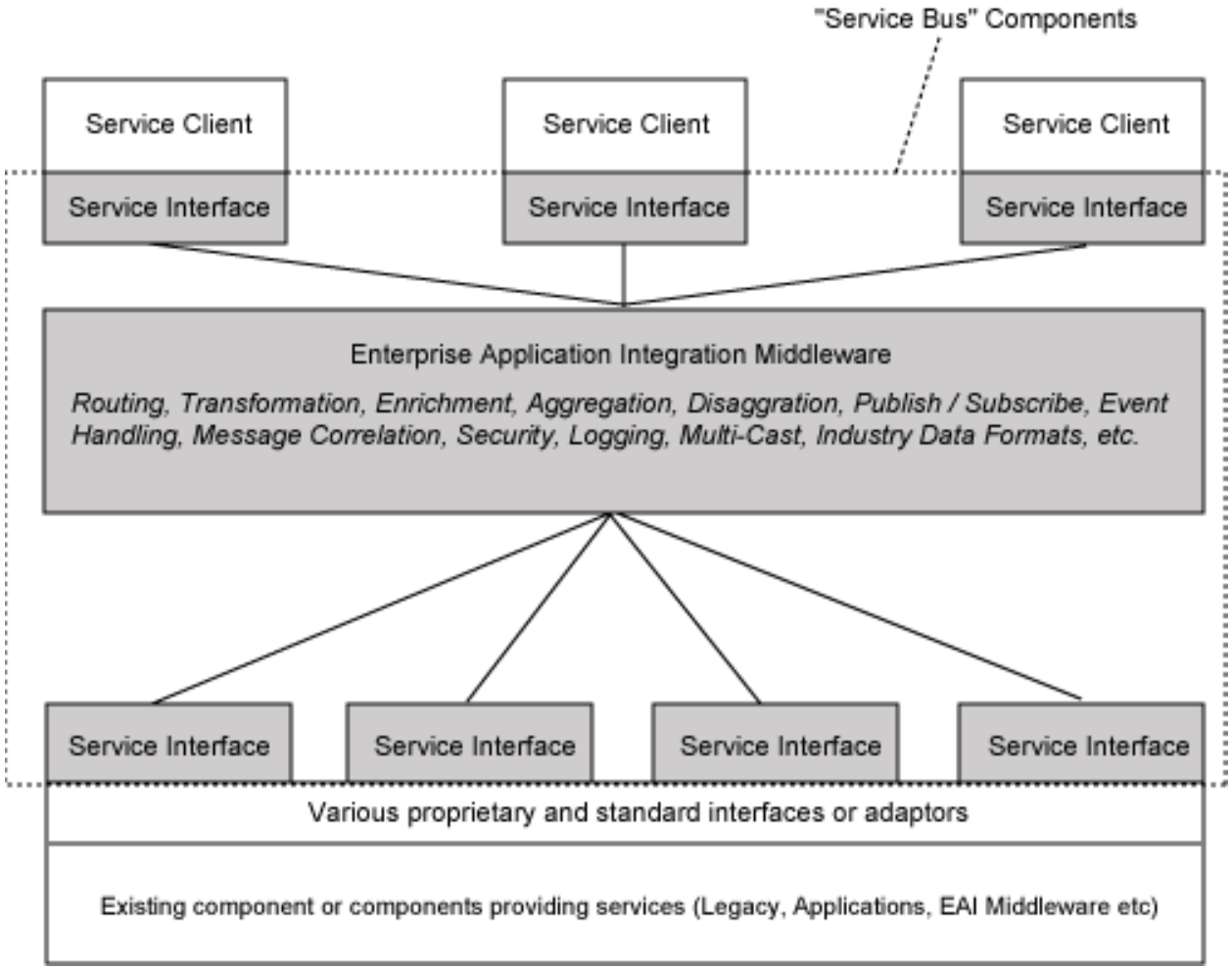
面向服务架构的企业应用集成基础架构

- 用这种方法实现的ESB可以使用这些重要的开放的而且也是事实上的标准，并且从它们中获得好处。这确实是一种可以把这些标准广泛应用到现有IT基础架构中的方法：
  - 许多EAI技术的应用非常广泛，甚至与开放标准一样具有互操作性。
  - 可以使用 XML 数据和消息格式以帮助实现互操作性和平台无关性，就像 XML 在 Web 服务规范中帮助实现了这些优点一样。
  - EAI 将很有可能支持一些形式的 Web 服务，因此可以在适合的场合提供开放标准接口，尤其是对于使用 document/literal SOAP 模型来公开所使用的 XML 格式的场合。另外，可以通过添加服务网关到该解决方案来提供这种访问。
  - 有些时候，可以利用 Java 的平台无关性来提供客户端 API 包，这不但对于 J2EE 环境来说很有用，而且对于单独 Java 环境、支持 Java 的数据库环境以及其他很多场合也是如此。
  - EAI中间件可以支持其他的开放标准（比如JMS），这些标准也许不像 Web 服务那样广泛适用，但仍然有很多技术支持它们。

面向服务架构的企业应用集成基础架构

- 该方法是向完全开放标准的 SOA 基础架构发展的一个重要步骤。虽然从某种意义上说，至少应该考虑将其移植到 Web 服务标准，但这种方法是向完全开放标准的 SOA 基础架构发展的一个重要步骤。EAI 和 XML 技术的过渡使用至少提供了处理问题（比如接口粒度、公共数据模型与格式等）的方法，所有这些都是向前发展的重要步骤。
- 成熟的 EAI 技术通过已被证实的性能、可用性以及可伸缩性等特点提供了极其丰富的 ESB 功能（流程和数据模型、转换、基于内容的路由、服务聚集和编排等等）。由于这些功能是最重要的需求，因此不借助 Web 服务技术而只使用 EAI 技术来实现 ESB，这从解决方案的核心上看是完全符合要求的，尤其是因为如果需要使用 Web 服务，可以在许多方面添加 Web 服务支持。

使用 EAI 中间件实现功能丰富的服务总线



EAI中间件模式的实现技术

- EAI中间件的选择取决于特定情况下所需的ESB功能与各种不同EAI产品（比如 WebSphere Business Integration）的特性。
  - 服务接口定义模型：为了遵循 SOA 原则，应使用直接的接口来定义服务。虽然有些 EAI 技术可能提供了这样的模型，但在其他的情况下，需要自定义解决方案。在实践中，这经常通过使用 XML 模式并结合服务身份确认、寻址以及业务数据来实现。然而，也有不使用 XML 的解决方案，比如某些服务网关模式的遗留系统的实现所使用的文本解决方案。
  - 与数据模型无关的接口模型：用于声明性地定义应该如何使用 EAI 基础架构的特性来调度服务请求和响应。应用程序需要一些机制以解释接口定义及适当的调用 EAI 基础架构。还有，这些机制可以通过 EAI 技术提供，可供选择的技术包括设计与开发规范的实施，或者框架 API 的使用。

EAI中间件模式的实现技术

- 框架API的开发和维护代价较高，但是它比实施跨越多个应用程序规范要更有效。如果至少大多数连接到服务总线的应用程序都支持同一种编程语言（比如 Java）的情况下，这样的方法是最有效的。
- 业务数据模型：是采用基于XML的、专有的还是自定义的。由于存在很多普通的和具体于特定行业的XML数据模型，采用这些模型中的一种可能会有好处。然而，许多这种模型正处于向Web服务规范移植的过程中，如果考虑使用这种解决方案模式的原因是由于可用的 Web 服务技术出于某些原因而不适合使用，那么就不可以选择那些规范。最后，如果 Web 服务或其他基于规范的某些形式的访问需要使用这种自定义模式实现的服务，那么就既可以选择使用 EAI 技术提供的 Web 服务支持，也可以添加一个显式服务网关组件（如果它能够更好的匹配需求的话）。

EAI中间件模式分析

- 由于这种解决方案模式代表了重要的开发、实现和维护工作，所以需要慎重考虑。该模式的优点是它与 SOA 原则完全一致，这被反复证明对交付业务有益，并能够用成熟的技术实现，使之具备企业级的功能、弹性和性能。
- 该解决方案的成本主要有两方面。首先在于解决方案的最初实现和随后进行的维护，其次，在于移植工作，随着 Web 服务技术的成熟并日益引人注目，最终很可能需要采用开放标准的解决方案。
- 这种模式的采用是一个即时的决策，这个决策依赖于近期或中期的利益来证明是否值得进行必要的投入。投入的多少依赖于所使用 EAI 的现有级别，也依赖于附加定制开发的工作量多少。近期或中期的定义依赖于单个组织认为新兴的 Web 服务规范何时会足够的成熟，以满足他们功能性或非功能性的需求。

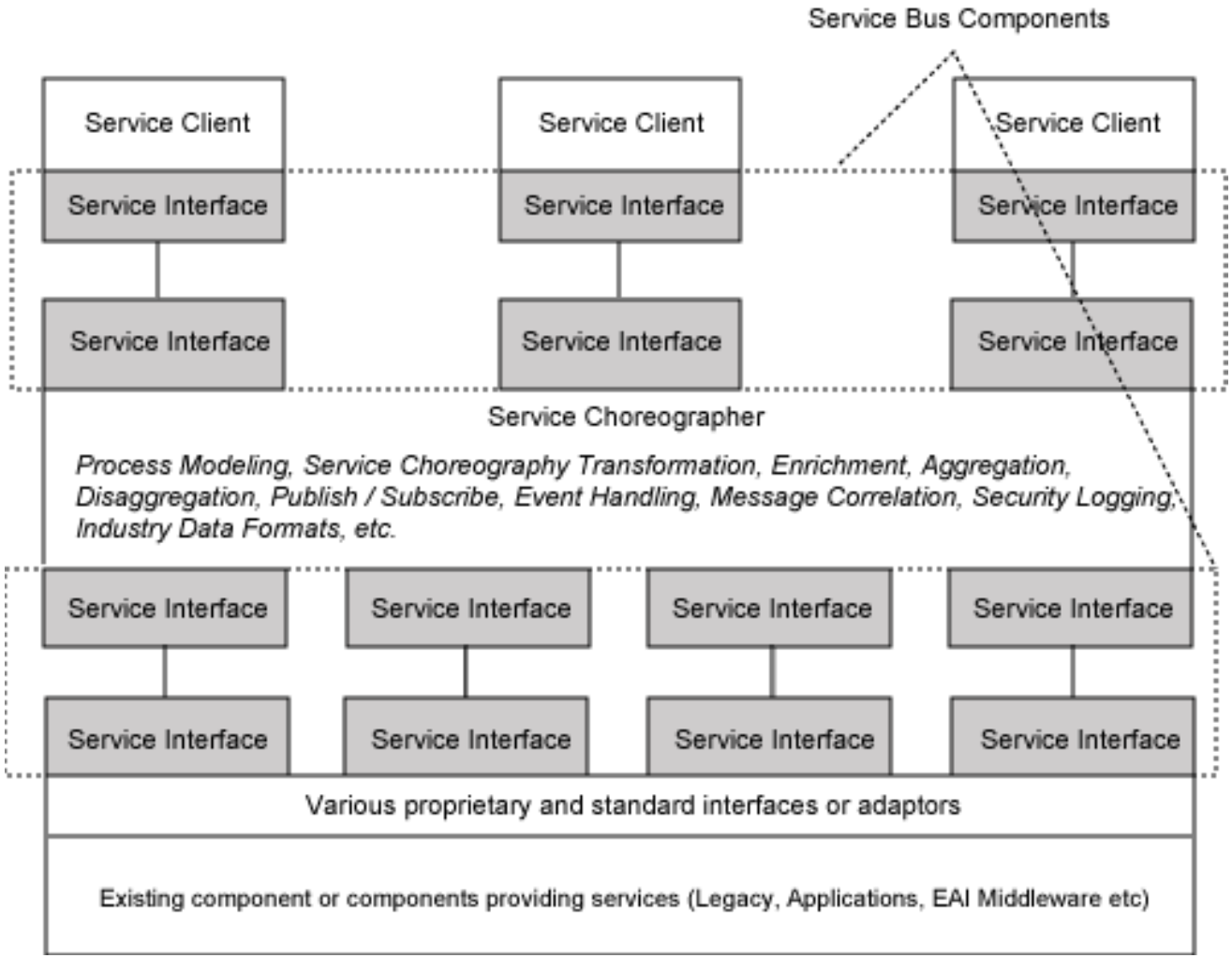
EAI中间件模式分析

- 替代模式
  - Web服务兼容代理模式是与EAI中间件模式相似的使用开放标准技术的实现方式。

服务编排解决方案模式

- 这种模式由专用服务编排组件的实现组成。这种组件不是真正的ESB，但是它通过多种协议（比如SOAP/HTTP或MQ）支持对服务的连接性，这需要或隐含着ESB的存在。在有些场景中，这种支持足以对服务提供方和服务器请求方进行直接连接。但如果情况并非如此，ESB可以通过本文描述的任何其他解决方案模式来提供，这就构成了完整SOA基础架构解决方案模式。

服务编排解决方案模式


Service Bus Components
Service Client
Service Client
Service Client
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Service Choreographer
Process Modeling, Service Choreography Transformation, Enrichment, Aggregation, Disaggregation, Publish / Subscribe, Event Handling, Message Correlation, Security Logging, Industry Data Formats, etc.
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Service Interface
Various proprietary and standard interfaces or adaptors
Existing component or components providing services (Legacy, Applications, EAI Middleware etc)

服务编排的实现技术

- 这种解决方案模式中要做的最重要的选择是它所需开放标准的级别。有以下三个场景：
  - 对服务接口和流程建模大规模地采用Web服务规范。
    - 支持BPEL4WS的产品不够成熟
  - 对服务接口采用Web服务规范，并结合使用专有的流程建模技术。
    - WebSphere Enterprise Process Choreographer 技术提供了对Web服务接口和流程定义的支持。
    - MQ Workflow提供了对更成熟但更专有的服务编排技术的支持，该技术既有Web服务接口也有专有接口。
  - 使用专有的接口和流程建模技术。
    - 为了解决可伸缩性或弹性需求，可以添加服务网关组件以提供Web服务连接性。

服务编排模式分析

- 这种解决方案模式很大程度上依赖于基于规范的或专有的解决方案是否实现。基于规范的解决方案目前还不太成熟，但它最终将能够提供更好的互操作性。专有解决方案将可能在较为熟悉的模型中提供可伸缩性和弹性，还可以充分利用高互操作性的通信技术（比如 MQ）。但是，随着开放标准技术的成熟和蔓延，这种方案可能最终还是需要一些移植工作。

完整SOA基础架构解决方案模式

- 这种模式代表了服务编排组件与服务总线实现的结合。一端采用完全专有解决方案，使用ESB和专有服务编排技术的EAI Infrastructure for SOA模式，另一端采用完全开放标准的解决方案，使用ESB以及适应开放标准的服务编排技术的Web服务兼容代理模式。

本章内容

1. 驱动ESB体系结构和设计决策的问题
2. ESB解决方案模式
3. SOA中的ESB场景及解决方案

SOA中的ESB场景

- 两个系统的基本集成
- 支持一个或多个应用程序实现更广泛的连接性
- 支持遗留系统实现更广泛的连接性
- 支持企业应用程序集成（EAI）体系结构实现更广泛的连接性
- 实现组织之间服务或系统的受控集成
- 通过编排服务使流程自动化
- 实现具有高服务质量和Web服务标准支持的SOA基础架构

两个系统的基本集成

- 场景：企业需要提供用不同的技术（如J2EE、.NET、CICS等等）实现的两个系统之间的集成。Web服务SOAP标准或消息传递中间件可能是候选的集成技术。这个场景的一个重要的问题是，将来是否会出现需要集成其他系统的情况。一开始就使用可扩展解决方案可能会对未来的需要提供支持；但是必须在为构建这样的解决方案而付出的额外工作与解决简单的问题的最初需要之间保持平衡。
- 问题：1,3,4,6,10,13
- 相关的解决方案模式：
  - 使用适配器来实现基本集成
  - 如果想要在将来进行扩展，有以下两种方案：
    - 添加控制服务网关
    - 实现一个复杂的基础架构—比如Web服务兼容代理或EAI中间件模式

支持一个或多个应用程序实现更广泛的连接性

- 场景：现有的已封装或自定义开发的应用程序（例如CRM、ERP等等）可能是用J2EE平台或其他应用程序服务器环境实现的，它们提供的可用功能超出了应用程序本身。以服务的形式公开这些功能的价值在于，既支持应用程序彼此之间的互操作，也提供对新的通道或客户端的访问。使用可互操作或开放的标准通信和服务协议是今后发展的最佳途径。
- 问题：1、2、3、4、6、8、9、10、11、12、13、14
- 相关的解决方案模式：
  - 使用包装器或适配器来实现基本集成—请参见基本适配器。
  - 添加控制服务网关
  - 实现一个复杂的基础架构—如Web服务兼容代理或EAI中间件模式
  - 如果还需要流程编排，就实现服务编排模式或者完整SOA架构模式

支持遗留系统实现更广泛的连接性

- 场景：组织对遗留技术（比如 CICS、IMS 等等）进行了大量的投资，以支持为他们提供核心业务事务和数据访问的应用程序。其重要价值在于提供互操作性或开放标准、以及对这些事务进行基于服务的访问（例如，查询帐户余额的事务、创建订单、日程安排或交付跟踪、查询库存级别等等）。
- 问题：1, 2, 3, 4, 7, 8, 9, 10, 11, 13, 14
- 相关的解决方案模式：
  - 使用适配器来实现基本集成
  - 如果想要在将来进行扩展，有以下两种方案：
    - 添加控制服务网关
    - 实现一个复杂的基础架构—比如Web服务兼容代理或EAI中间件模式

支持EAI体系结构实现更广泛的连接性

- 场景：需要对现有的EAI基础架构（如 IBM WebSphere Business Integration）进行扩展，以对其进行基于可互操作协议或开放标准的访问。虽然根据XML业务数据并通过高度可互操作协议（如HTTP或WMQ）公开服务接口可以在某些场景中提供适当的互操作性级别，但是如果对现有的集成范围的自定义开发或专有扩展都不是可接受的，则可能需要支持WSDL和SOAP Web标准。
- 问题：3、4、5、8、9、11、13、14
- 相关的解决方案模式：
  - 使用开放数据格式及EAI中间件模式来扩展EAI基础架构。
  - 添加控制服务网关。
  - 对带有Web服务兼容代理的基础架构增加开放标准支持。

实现组织之间服务或系统的受控集成

- 场景：组织希望使其客户、供应商或其他合作伙伴能够直接集成由一个或多个应用程序、遗留系统等等提供的功能。控制的重点是需要提供从外部各方到这些应用程序的安全且易管理的访问。因为组织不能直接控制其合作伙伴所使用的技术，因此最好使用开放标准。此场景既可以应用于分散的组织之间，也可以应用于大型分布式组织的各个单位之间。
- 问题：1、2、3、4、6、8、9、10、11、13、14
- 相关的解决方案模式：
  - 添加服务网关
  - 如果需要更多的复杂功能，就实现Web服务兼容代理

通过编排服务使流程自动化

- 场景：
  - 现有的已封装（例如CRM、ERP等等）或自定义开发的应用程序可能是在J2EE平台或其他应用程序服务环境中实现的，它们提供的可用功能超出了应用程序本身。可以使用可互操作或开放通信和服务协议将这些功能作为服务公开，这样应用程序就可以交互。可以在某些层次上组合这些交互以构成业务流程。应该使用适当的建模和流程执行技术（可能遵守适当的开放标准）来对这些流程进行显式建模。
  - 此场景可以看作是支持一个或多个应用程序实现更广泛的连接性场景的发展。它不被当作一个ESB场景，因为服务编排通常是与ESB分开实现的。
- 问题：1、2、3、4、6、10、11、12、13、14
- 相关的解决方案模式：
  - 如果服务的直接连接是可能的，则实现服务编排
  - 如果需要更复杂的集成或控制，则实现完整SOA模式

实现具有高服务质量和 Web服务标准支持的SOA基础架构

- 场景：此场景是由前面的场景组成的。它代表了对由多个应用程序、遗留系统等等提供的服务进行普遍的内部或外部访问的需要。需要各种安全、聚合、转换、路由以及服务编排功能。IT组织以响应所支持的业务不断增加的需求，从而使得能够在业务系统之间进行更普遍且更灵活的集成。
- 问题：1–14
- 相关的解决方案模式：
  - 实现完整SOA模式