（1）标准化服务契约。服务遵守一个给定的通信协议，该协议由一个或多个服务描述文件详细说明。

（2）松散耦合。松散耦合其实软件设计的一个基本原则，从数据抽象到 OO、到 CBSE，都是以松散耦合为设计目标的。因此，以松散耦合来定义 SOA 是不够的。服务被设计成自包含组件，维护最小化依赖于其他服务的关系，仅要求可相互识别。服务契约约束服务之间所需的互操作。这简化了服务的柔性聚合，实现了更灵活的支持企业业务发展的设计策略。抽象。服务是由服务契约和描述文件完整定义的。服务隐藏了自己的逻辑，将其封装在服务实现中。

（3）可重用性。服务被设计为组件的形式，可以更为有效地实现重用，这样就减少了开发时间和相关的成本。可重用性能使得设计、实施和部署成本有效系统变得更加灵活。因此，可以通过支付合理的费用，利用第三方服务来交付所需的功能，而不需要自行开发此功能。

（4）定义与实现相分离。通过抽象类和接口类，OO 也实现了这个特点，设计模式就是该特点的一个典型代表。实际上定义与实现相分离是软件工程 30 多年以来一直努力追求的目标之一。

服务发布、注册和发现。这也不完全是 SOA 特有的思想。任何软件资产都可以发布，包括工作流、协同模型、应用模版、测试用例等。

（5）自治性。服务控制封装逻辑，服务消费者没有必要了解执行情况。

状态缺失。通过提供无状态的互操作模式，提高了服务被重用和集成的机会，尤其是在单一服务被多个属于不同机构和业务领域的消费者使用的情况下。

（6）可发现性。服务由描述文件定义，其中包括有效发现服务的元数据。服务发现为利用第三方资源提供了有效的方法。

（7）可组合性。使用服务作为基础模块可以实现复杂的操作。服务协同和编排为组合服务并实现业务目标提供了坚实的支撑。