Chinese Dutch English French German Portuguese Russian Spanish

带注释的 SOA 宣言

托马斯 • 伊尔关于 SOA 宣言的评论和理解

服务导向是指导信息技术运作的一个模式. 面向服务的体系结构是一种应用服务导向的体系结构.

这个宣言从一开始就是关于两个不同但又紧密相关的话题:面向服务的体系结构模型和服务导向,这种体系结构被定义的模式. 这篇宣言的格式模仿了"敏捷宣言", 它用有限的篇幅来简明陈述它的雄心和价值观,以及实现这种雄心和价值观的指导原则. 这样的一个宣言不是规格说明, 参考模型或者白皮书, 也不能作为提供实际定义的选择. 我们决定增加这个导言, 就是为了澄清这些术语是怎么和为什么在宣言的其它部分被引用.

我们一直在采用服务导向...

服务导向模式最好被看成是实现一个具体目标状态的一种方法或途径,这个具体目标状态将进一步由一整套战略目标和利益定义. 在我们运用服务导向时,我们形成支持实现这个目标状态的软件程序和技术架构. 这使得技术体系结构具有面向服务资格.

...帮助企业组织始终如一地产生持久的商业价值, 提供更多的灵活性和成本效益...

这个导言的续篇强调了面向服务的计算的一些最突出的和通常预期的战略利益. 理解这些好处有助于阐明我们要实现的作为应用服务导向结果的上述目标状态.业务层面的敏捷性与组织的响应能力是可比拟的.如果一个组织能够更轻易地和更有效地回应商务改变,那么这个组织就会更为有效的和更为成功地适应这种改变的影响(并且,进一步利用这种改变可能带来的好处).

服务导向将服务定位作为IT 资产,这个资产被期待着提供的重复价值以后会远超过产出它们所需的初期投资.成本效益主要是关系到对投资的预期收益. 在很多方面,成本效益的增长伴随敏捷性的增长; 如果有更多的机会可以重复使用现有的服务,那么通常就会花费较少的费用来建造新的解决方案.

"可持续的"商务价值是指把为建立软件程序的长期的服务导向目标看作为带有内在灵活性的服务.这种灵活性表现在已经开发出的模块可以不断地被组装到新的解决方案结构当中,同时也会不断地发展, 使其适应日益变化的商务需求.

...符合变化的商务要求

导言中最后的六个词语是理解面向服务的计算这一基本哲理的关键所在. 使商务适应不断变化的这种需求是服务导向的基础,并被认为是一个基本的总体战略目标.

通过我们的工作,我们优先考虑:

即将引入的陈述将建立一套核心的价值观,其中每一个也被表示为优先某个具有一定价值的事情. 为了要贯彻实现面向服务的计算的战略目标和利益,这个价值系统的目的就是解决经常出现的需要选择的难题.

商务价值高于技术策略

如前所述的,适应商务变化的需要是一个总体战略目标. 因此,面向服务的体系结构和任何软件程序,解决方案以及采用服务导向得到的生态系统的基本特征都是商务驱动的. 它不是技术决定商务的方向,而是商务的洞察力规定了技术的利用. 这个优先权可以在一个 IT 企业的范围内产生一个深刻的连锁反应.它将变化引入到了 IT交付生命周期的几乎所有部分,从我们如何计划和如何投资自动化解决方案,到我们如何建造和管理它们. 宣言中的所有其他价值观和原则以某种方式支持这个价值的实现.

战略目标高于具体项目的效益

在历史上,许多 IT 项目集中于单独地建造应用,这些应用专门用于自动化商务处理的需求,这在当时还是流行的. 这实现了直接的(战术上)需要,但是当更多的这些单个用途的应用被提交时,它将导致一个 IT 企业充满了称为应用"信息孤井" 的逻辑和数据的孤立体. 如果新的商务需求出现时,那么或是新的应用"信息孤井"被产生,或是"信息孤井"之间的集成通道被建立. 迄今为止,更多的商务变化出现,集成通道不得不被扩大,这样就不得不产生出更多的"信息孤井",不久 IT 企业场景就会变得错综复杂,越加累赘,增大花费,同时也减缓了企业的发展.

在很多方面,服务导向的出现应对了这些问题. 它是一种模式,这种模式通过坚决优先考虑长期的,商业战略目标的实现,对特定工程,基于"信息孤井"的和集成的应用开发,提供一个可供选择的方案.由服务导向所提出的目标状态没有传统的应用"信息孤井". 甚至在采用服务导向的环境中,存在遗产资源和应用"信息孤井"时,目标状态是将他们协调到任何可行的程度.

内在互操作性高于定制的集成

对于共享数据的软件程序,它们需要是互操作的. 如果软件程序不是设计成可兼容的,他们将可能不是可互操作的. 为了能使不兼容的软件程序之间实现可互操作性,要求他们被集成. 因此,集成是为实现不同的软件程序之间的互操作性所需要做的努力.

尽管通常是必要的,但是定制化的集成可能是费用上昂贵的,而且是费时间的,并且最终导致产生一个阻碍企业发展的脆弱的架构.服务导向的目标之一是通过在一个给定的领域之内,设计软件程序来将定制的集成的需求减至最小,以便这些程序是天生兼容的. 这是称为内在互操作性的特征.服务导向模式包含致力于在几个层面上建立内部互操作性的一整套具体的设计原则.

作为在一个处在给定的领域之内的软件程序的特征,内在互操作性是实现战略利益的关键,这些找略利益是指越来越高的成本效益和敏捷性.

共享服务高于为特殊目的具体实施

正如刚刚解释的,服务导向建立了由一整套设计原理组成的设计方法. 在适用于一个有意义的程度时,这些原则将一个软件程序构造成一个面向服务的逻辑单元,这个单元可以合理地看作为一个服务.

服务装备有直接支持先前描述的目标状态的具体特征(如激活内在互操作性的). 这些特征之一是,能被共享和重复使用以支持不同的商务处理的自动化的多用途的逻辑的封装

作为共享服务建立起来的IT资产,这个 IT 资产在可以提供重复的商业价值的同时可以降低交付新的自动化解决方案花费和努力.虽然传统的,单一用途的应用在一个解决战术商务需求的中存在价值,但在实现面向服务的计算的战略目标(又一次包括成本效益和敏捷性的增长)中,使用共享服务提供更大的价值.

灵活性高于优化

这也许是价值优先排序声明的最广的,并最好地被看成是一个指导性原理.这个原理用来指导如何在交付和发展单个的服务和服务的清单时,如何更好地优先不同的考虑.

优化主要是指,通过调整一个给定的应用设计或加快交付以满足直接需求,来实现战术收益. 关于这个,没有任何不受欢迎的,除了当有关于培育灵活性没有优先考虑时,它能带来上述的基于"窖"的环境.

例如,灵活性的特征不仅仅是服务的有效(和本质地)共享数据能力. 为了真正地对日益变化的商务需求做出及时响应,服务在如何能被组合和聚合成复合解决方案方面也必须是灵活的. 与常常是相对静态的传统分布式应用程序不同,尽管它们是组件化的,服务组合需要设计成具有允许不断增加的具有一定水平内在的灵活性. 这意味着当一个已经存在的商务处理改变时,或者在一个新的商务处理被引进时,我们需要在组合构架之内,能以付出最小的(集成)努力,增加,去除和扩展服务. 这是,为什么服务组合能力是服务导向设计原则的关键之一.

进化改良高于初始阶段的追求完美

当说到与服务定向有关的术语"敏捷性"时, 有一个共同的混淆点.一些设计方法从立即见到收益角度,主张软件程序的快速的交付. 这可以被视为"战术上的敏捷性",因为这种设计方法的中心点是在战术上的并且获得短期的好处. 服务导向带有赋予整个组织应对变化的意图,主张达到组织的或商务一级的敏捷性. 这种形式的组织敏捷性也能称作为"战略灵活性",因为重点是放在我们交付的每一个软件程序的寿命方面,我们想要朝着培育具有长期的战略价值的敏捷性的目标状态方向努力.

对于能使组织的敏捷性起到作用的一个 IT 企业,它必须随着商务一道发展.我们通常不能预测随着时间的推移一个商务将需要怎样发展,因此我们最初不能建 立完美的服务. 同时,在一个组织的商业智能中通常已经存在有大量的知识,这些知识在SOA 项目的分析和建模阶段就能够获得.

与服务导向原则和可行的实施方法一起,这些信息能帮助我们确定和定义一整套服务,这套服务就可以捕获到现今商务是如何存在和运转的,而同时具有足够的灵活性以适应商务未来的变化.

也就是说,虽然我们清楚地认识到右边条项的价值,我们更重视左边的条项

通过研究这些价值被如何优先考虑,我们深入了解到服务导向与其他模式的不同. 这种洞察力将通过几种方法受益于 IT 从业者. 例如,它能帮助建立基本的准则,我们可以用这个准则来决定一个给定的组织或 IT 企业的服务导向的兼容性. 它能进一步地帮助决定服务导向可以或者应当被采用的程度.

对核心价值的判断也能帮助我们理解,在特定的环境中成功地实行 SOA 项目的挑战程

度. 例如,这些优先级中几个可能与已经建立起来的信念和偏好产生冲突. 在这样的情况中,服务导向的好处需要对采用他们可能有的效果和影响加权.(这种效果和影响不仅指技术方面的,而且包括组织方面和 IT 文化方面).

即将给出的指导原则被提供以帮助应对这些类型的许多挑战.

我们遵循这些原则:

迄今为止,宣言已经建立了一个总体构想以及一整套与总体构想相关的核心价值.宣言的其他部分包含一套法则,这套法则是作为坚持价值并且实现构想的指南给出的.

重要的是,要记着这些是对这篇宣言所特有的指导原则.还有一套已建立的设计原则,它包括服务导向设计模式以及有记载的对服务导向和"面向服务的"构架所特有的实践和模板.

尊重企业组织内部的社交和权力结构

最常见的 SOA 隐患之一是正在接近于采纳一个以技术为中心的提案. 这种作法通常总是导致失败,因为我们没有为不可避免的组织影响作好准备.

采纳服务导向是转变我们自动化商务的方式. 然而,不管我们怎么计划这种转变,我们总是必须要从对组织的理解和评估,以及它的结构,它的目标和它的文化开始.

采用服务导向很大程度上是一个人类经验. 它需要来自权力的支持,然后要求一个 IT 文化采用一个战略的以社区为中心的思维方式. 我们必须充分地承认和计划这级组织的变化,以便接受对于达到服务导向的目标状态所要求的必要的长期承诺.

这些类型的考虑不仅帮助我们决定如何最好地开展一个SOA 提案,它们也进一步地帮助我们定义采用的最适当的范围和方法.

认识到SOA最终需要许多层面的变化

有句谚语说:"成功是为机遇准备的". 也许,从迄今为止实施的 SOA 项目吸取的头号教训是,我们必须充分地理解然后计划并准备由于采用服务导向所引起的变化的量和范围. 这里是一些例子.

通过定位软件程序为具有长期的可重复的IT资产, 服务导向改变我们如何建立自动化解决方案. 需要一个前期投资来创造一个由如此的资产组成的环境,需要一个持续的承诺以便维护并充分发挥他们的价值. 因此,在我们如何资助,如何测量和如何维持在 IT 企业内部的系统上,变化从一开始就是需要的.

此外因为服务导向引入了定位为企业资源的服务,在我们如何拥有系统的不同的部分并且调节他们的设计和使用方面,将存在有改变,更何况要求保证连续的可伸缩性和可靠性的基础结构的变化.

采纳SOA的范围可以有所不同,应努力保持工作重点在可控制和有成效的范畴内

一个普通的认识误区是,为了实现面向服务的计算的战略目标,必须在整个企业范围内的基础上采用服务导向.这意味着,建立和实施在整个 IT 企业范围内的设计和工业标准,以便创建一个内部能实现互操作的服务的企业范围内的清单. 尽管这个想法没什么错,它对许多组织来说,并不是一个现实的目标,特别是对于具有较大的 IT 企业的那些组织.

对于任何给定的采纳SOA尝试最合适的范围决定于计划和分析的结果,并结合了各种务实考虑,如上述对组织结构和权力范围的影响,以及由此引起的(企业)文化变迁.

这些类型的因素帮助我们决定可管理的采纳范围.但是对于产生一个促使企业朝向它的战略目标发展的环境的任何采纳工作,这个范围也一定是有意义的.换句话说,它必须是有意义的孤井互联,以便服务的集合可以在一个预定义的边界之内,以彼此的关系被交付. 换句话说,我们想要建立"服务的大陆",而不是非常可怕的"服务的孤岛".

在同样的 IT 企业的领域中,建造独立拥有且能自主控制的服务清单的概念,将会减少通常归咎于"创世大爆炸"的 SOA 项目的许多风险,而且还可以减轻组织和技术进步的影响(因为影响仅局限于一个片段的和可管理的范围内). 它也是顾及阶段性采纳的方法,在这种方法中,一个领域服务清单能够同时被建立.

单靠产品和标准既不会给你带来SOA,也不会为你应用服务导向模式

这个原则应对两个分开的但是非常相关的神话. 第一个是,你可以使用现代的技术产品买到你进入 SOA 的方式,第二个是这个假设,即采纳工业标准(如 XML,WSDL,SCA 等)将自然地产生面向服务的技术构架.

供应商和工业标准社团已经公认为在非专有的框架和平台上,建立现代的服务技术创新. 从服务虚拟化到云计算和和网格计算的每一件事,已经帮助推进建设精深和综合的面向服务的解决方案的潜力. 然而,这些技术中没有一个对 SOA来说是独有的.就像你可以在你私有服务器上,能够轻易地建造云计算的基于孤井的系统.

没有诸如"装在盒子里的 SOA"之类的事情,因为为了实现面向服务的技术结构,服务定向需要被成功地应用； 进而,这要求我们所有的设计和建造由商务独特的方向,构想和需求所驱动.

SOA可以通过各种不同技术和标准来实现

服务导向是一个技术中性的和供应商中性的范例. "面向服务的"体系构架是一技术中性的和供应商中性的构架的模型.面向服务的计算可以被看成是一种特殊形式的分布式计算. 因此,面向服务的解决方案可以使用适合分布式计算的差不多任何技术和工业标准来建造.

当一些技术(特别是基于工业标准的那些)能增加应用一些服务导向设计原理的潜力时,它是满足商务需求的真正潜力,这种潜力最终决定了技术和工业标准的最合适的选择.

应依据工业界的,事实上的,和团体的标准来建立一套统一的企业标准和政策

工业标准代表非专有的技术规范.在其他的事务中,这些规范帮助建立技术构架的一致的基准特征(如传送,界面,消息格式等). 然而,单独使用工业标准不能保证服务将是内部能互操作的.

至于两个软件程序是完全相容的,则要坚持额外的约定(如数据模型和政策).这就是 IT 企业为什么必须建立和强制实施设计标准的原因. 未能在一个特定领域内标准化妥善和监管服务的标准化,就会引起瓦解很多战略利益的实现所要依赖的互操作性的结构.

这个原则不仅主张使用企业的设计标准,它也提醒我们,只要有可能并且是可行的,定制的设计标准一般来说,应当是基于并且纳入工业和社团已经使用着的标准.

追求外部的统一性,同时允许内部的多样性

联盟可以被定义为一整套不同实体的统一. 当允许各个实体能够在内部独立地被管理时,所有都同意坚持一个共同的统一战线.

面向服务的架构的基础部分是引入了联合端点层.在以一个统一的方式和一个给定的领域中公布描述单个服务的一组终点时,这个联合终点层抽象出服务实施的详细信息. 完成这通常涉及实现基于行业和设计标准相结合的统一. 这个跨服务的统一的一致性是实现内在互操作性的关键.

一个联合端点层进一步地帮助增加考察供应商多样性选择的机会. 例如,一个服务可能需要在与另一个服务完全不同的平台上建造.只要这些服务维持相兼容的端点,各自的实施的治理能仍然是独立的. 这不仅强调,可以使用不同的实施媒介(如 EJB,.NET,SOAP,REST 等)建造服务,它也强调,可以根据需要,不同的中间平台和技术能被一起利用.

注意:这种类型的多样性与价格相关. 这个原则不提倡多样化 - 它只是建议我们在证明其是合理的时候允许多样化,以便"最佳的"技术和平台能够被充分利用以最大限度地提高商务需求实施.

商业和技术有关人员必须合作来识别有效的服务

为了技术解决方案是商务驱动的,该技术必须是与商务是同步的.因此,面向服务的计算的另一个目标是使技术和商务一致. 这个一致的过程最初被完成的阶段,是在分析和模型建立的过程期间,这个过程通常先于实际的服务开发和交付.

完成面向服务分析的关键要素,是让商务和技术专家紧密配合来辨别并定义候选服务. 例如,商务专家能准确地帮助定义属于以商务为中心的服务的功能范围,而技术专家可以提供务实的输入,来确保概念的服务的粒度和定义相对于最后的实施环境,仍然是现实的.

通过考虑当前和未来的使用范围来扩大服务的重复应用

一个给定的 SOA 项目的范围可能是整个企业内的,或者它可能局限于企业的某个领域. 无论是什么范围,一个预定义的边界被建立以包含服务清单,这些服务在能够被开发前,需要概念上被模型化. 通过建模彼此有关的多个服务,我们本质上建立了我们最后将要建造的服务设计图. 在尝试确定和定义不同的解决方案能够共享的服务时,这个建模实践是关键的.

有各种各样的方法和途径可以用于实现面向服务的分析阶段. 然而,它们的共同点是,服务的功能的边界被规范化以避免冗余. 甚至到那时,规范化的服务不一定产生高度可再次使用的服务.因为这时其他因素开始起作用,如服务逻辑是足够普通的服务粒度,独立性,状态管理,可伸缩性,组合能力,以及服务逻辑是足够通用以便它能被有效地重复使用的程度.

在商务和技术专专长指导下的这类考虑,提供了定义服务的机会.这些服务能够抓住目前的使用需求,而且具有适应将来变化的灵活性.

检验服务满足业务的要求和目标

如同任何事一样,服务也能够被滥用. 在不断扩展并管理一套服务时,需要验证和估量这些服务满足商务需求的使用和效力. 现代工具提供监控服务使用的各种各样的手段,但

是也要考虑无形的因素,以确保服务不仅仅是因为存在而被使用,而且要验证这些服务正在真正地履行商业需求并且满足预期.

这一点对肩负多个依赖关系的共享服务是特别正确的. 共享服务不仅仅需要适当的基础设施来保证重复使用这些服务的所有的解决方案具有可伸缩性和可靠性,它们也需要仔细地被设计和被扩展以确保不偏离它们的功能范围.

针对实际使用来发展进化服务及其组织

这个指导原则直接回过来联系到"进化改良高于初始阶段的追求完美"价值陈述,以及维持商务和技术的一致性的总体目标.

我们永远不能期望用猜测来决定服务颗粒度,服务需要履行的功能范围,或者服务将被如何组成. 基于我们最初能实施的任何分析程度,一种给定的服务将被分配给一个定义的功能范围,并且将包含一个或多个功能能力,这些能力可能涉及一个或多个服务组成.

如同现实世界商务需求和情况改变,服务可能需要被扩大,扩展,重新分解,或者甚至也许被替换. 服务导向设计原理把天生的灵活性融合进服务构架,以便作为软件程序的服务是有弹性并且适应于改变和针对实际使用被改变.

区分开系统内具有不同变化频率的各种方面

使整体式的和基于孤井的系统缺乏灵活性的是,改变会对它们的现有使用造成很大的影响. 这就是,为什么常常很容易创建新的基于孤井的应用,而不是扩大或扩展现有应用的原因.

在分离关注事物后面的原理(通常称作为软件工程理论)是,一个较大的问题在分解成一组较小的问题或关注事物时,能被更有效地解决. 在将服务定向应用于关注的分离时,我们建立解决个别关注的解决方案逻辑的相应单元,从而除了给我们机会以聚集这些单元使成为不同的配置以便解决其它问题之外,还使得我们能够聚集这些单元以解决更大的问题.

除培育服务可重用性之外,这个方法导入了无数的抽象层.这些抽象层有助于保护组成服务的系统,使其免受改变的影响. 这种形式的抽象能以不同级别的形式存在. 例如,如果由一种服务所封装的遗留资源需要被替换,只要服务能保持其原始的终点和功能行为,这种改变的影响能够被减轻.

另一个例子是不可知论逻辑与非不可知论逻辑的分离. 如果前者类型的逻辑是多用途的并且没有什么变化,那么它就有很大的再使用潜力. 另一方面,非不可知论的逻辑通常代表父业务流程逻辑的单一用途的部分,它们常常是更不稳定的.把这些各自的逻辑类型分离成不同的服务层,进一步地引入抽象,它使服务能够具有可重用性,同时保护这些服务及其使用这些服务的解决方案免受变化的影响.

降低隐式的依赖关系,并公布所有外部的从属关系,以增强稳定性和减少变化造成的影响

最有名的服务导向设计原则之一是服务松散耦合原则. 一个服务构架在内部如何构建,以及服务如何与消费它们的程序(其程序可能包括其他服务)相关联,都可归结为是在服务构架的单独可动部分上形成的依赖.

通过对控制区域所做改动的影响的局部化控制,抽象层有助于进化式改良. 例如,在服务构架之内,服务门面可以用于实施的抽象部分,以便将实施的依赖程度减至最小.

另一方面,公开的技术服务合同需要揭示服务使用者为了与服务交互必须形成的依赖关系.在改动发生的时候,通过减少能影响这些技术性合同的内部依赖,我们能够避免这些改动对服务所依赖的使用者所造成的激增的影响.

在每一个抽象化层面,应围绕内聚的和可管理的功能来组织各个服务

每个服务要求很好定义的功能环境,决定什么逻辑属于或者不属于功能的边界之内. 确定这些功能服务边界的范围和粒度,是在交付服务生命周期期间的最关键的职责之一.

具有粗功能粒度的服务可能太缺乏灵活性,以致不是有效的,特别是如果指望这些服务是可重复使用的. 另一方面,过分细粒度的服务可能增加基础结构的负担,在这个基础结构中,服务组成将需要包括大量的组成成分.

确定功能范围和粒度的恰当平衡需要商务和技术专家的结合,并且进一步地需要理解在一个给定的边界中的多个服务如何彼此相关.

在这篇宣言中描述的许多指导原则将帮助做出这个决定,以支持将每一个服务定位成一个 IT 资产,这个 IT资产能够 推动一个IT 企业朝向实现面向服务的计算的战略利益的这一目标方向的发展.

然而,从根本上讲,它将总是实现所规定的现实世界的商务价值,从概念到交付重复的使用,任何面向服务功能的单元不断进化的道路.

感谢:我在 *2009* 年 *11* 月 *21* 日至 *22* 日的周末写下本页上的内容,是为当时仍然正在构想之中的"下一代 *SOA*" 这本书. 我在这里感谢*Prentice Hall* 出版社,使得这个内容能够作为最初的 *SOA* 宣言的补充公开发表. 我也要感谢工作组成员以及来自 *SOAPatterns.org* 团体的这些人,他们提供了反馈并协助了对所述内容的注释. 许多工作组成员已经发表了自己的文章,博客和关于 *SOA* 宣言的论文,所以我鼓励大家寻求进一步的评论和见解.

- 托马斯 • 伊尔

翻译

Yue Yuan

Original SOA Manifesto PDF About the Translators