为什么说 .NET 技术对 ERP 系统很重要

用托管代码构建的 ERP 系统的优点

白皮书

引言

微软里程碑式技术 .NET的发布给所有ERP公司带来了前所未有的挑战。在公众还对微软.NET 技术一知半解的时候，有关如何应用.NET技术就已经在2000年成为ERP软件供应商们纷纷考虑需要做出的重要决策战略。每个ERP软件供应商在利用.NET技术方面所选择的方向决定了它们在未来数年的命运以及对其每个客户在战略层面所能达到的影响。制造业在普遍连接和内部系统整合方面正在迅速达到新的高度。虽然不乏反对之声，但一些ERP公司及其同盟客户却有掉队的危险。

许多ERP公司选择了坏的时机

上世纪 90 年代后期，许多ERP公司投身Web浏览器浪潮，实施各种有关互联网和浏览器技术的项目，甚至将其软件改为“精简客户端”（lite-client）或 Web“门户”架构。对于某些公司而言，不幸的是，在这一重大变革后，.NET过于快速的登场。在.NET 技术出现后，一些公司由于技术上的山穷水尽、管理僵化或者依然念念不忘其新的“基于互联网的架构”而没有去积极了解和拥抱 .NET 技术。其他ERP软件公司则继续在具有毁灭性的千年问题（Y2K）、2000年经济不景气及 911 事件期间疲于奔命，且直至现在情况依然如故。它们没有资源来考虑对其产品进行.NET 技术所要求的完全改造。

虽然微软公司竭尽全力来推广.NET 技术，但是其核心内容的改变却代表着自Microsoft Windows出现后软件技术方面最重大的转变。.NET 是互联网革命中的又一枚重磅炸弹，它预示了一种全新的计算模型，该模型不只强调表面的网页交换，还强调系统合作及协作。***对 ERP 软件供应商及其客户来说， .NET 技术意味着企业软件应用的未来。***而且，正如我们在本白皮书中将要探讨的， .NET技术所指向的未来要求ERP软件公司重新思考和重新编写其基础架构。

恰当的地点，恰当的时机：迎接挑战

1998年，成立才4年的 Intuitive Manufacturing Systems 紧跟行业领先公司，也研究如何使用“精简客户端”架构和工具将其ERP软件包带入互联网时代。但Intuitive的研究发现了几件令人担心的事情。

首先，基于Web 览器的用户界面不能提供本机代码（native code）所能提供的丰富功能和初始速度（raw speed）。本机代码（在 PC 机上运行的代码）通过本机 CPU 来执行大多数计算操作，给予用户界面以出色的性能，并对用户的鼠标点击和键盘输入产生丰富的反应。在精简客户端模式下，大多数工作是在服务器上执行的。Web 浏览器界面的运行类似于放映幻灯片，真正干活的是投影机（服务器），荧幕（浏览器）只是挂在那里而已。

显然，精简客户端处理模式反映了大型机附带简易终端设备的处理模式，而不是利用灵活和更强大的PC。

同样，“精简客户端”架构模型给IT资源和预算有限的小型制造企业带来了更多压力。它意味着ERP客户必须使用更大型的服务器硬件和Web服务器软件，并投入IT预算使之保持运行。小型制造企业不会放弃其PC，Web 服务器会成为一个新增管理要素。

最后可能也是最大的担心，转向精简客户端架构并不会对提供丰富连接和分布式处理的功能有任何促进作用，但是这一需要却是显而易见的。硬件、网络及通信公司不断推出更智能和更强大的PC、手持无线设备、蜂窝电话及Dick Tracy（电影《至尊神探》主角）腕表。精简客户端架构在利用这些进展方面几乎无动于衷。

在 .NET框架于2000年6月亮相时，Intuitive简直可以说就在现场。Intuitive公司距微软主园区不过 10 分钟的车程，向来只使用微软技术，所以使用 .NET 是一件自然和合乎情理之举。当在微软的“.NET 早期采用者”（.NET Early Adopter）计划实验室了解了 .NET 技术后，Intuitive 开始不只是简单地使用 .NET 技术，而且还设计了一个全新的 ERP 架构来充分利用 .NET 技术。由于接下我们将会讨论的原因，Intuitive 认识到 .NET 技术要求对每行代码进行完全而彻底的改写。

为什么说 .NET软件是更好的软件

.NET不是Microsoft Windows的另一个版本，而是微软提供的下一代计算平台。 .NET的核心是一个新的软件层，位于 Windows操作系统之上。对于开发和运行软件来说，与我们目前所知的Windows 操作系统相比，新的.NET平台是一个更好的平台。.NET使软件开发人员和计算用户不再受操作系统的缺陷和不兼容问题的困扰。而且.NET 框架提供了一套具备前所未有的强大功能的新工具和预建组件，可用于编写被称为“托管代码（Managed Code）”的新型软件。随着时间的推移，托管代码将被公认具有明显的优势，这是因为：

托管代码更稳健。.NET平台不只是运行这一新型代码，而且还监督其执行过程，使软件错误能够被发现并在导致严重问题出现前得以被阻止。.NET 能够事先防止或减轻使旧 Windows 系统崩溃的“内存泄露”、“内存破坏”及“死机蓝屏”等问题。

版本并存。.NET 软件专注于自己的事情。它们集中待在一个地方而不是把触角伸向系统各处。而且在这样做时，它回避了老式的基于组件对象模型（COM）的机制，该机制会导致软件包的相互冲突和伤害。实际上，.NET允许一个托管代码应用的两个版本在同一计算机上“并肩”运行而无任何交互或冲突。因此，.NET 可以避免为了安装和试用一种软件产品的新版本而必须卸除老版本的问题。

更高的安全性。我们都知道，我们的系统是黑客窥伺的对象。在老式Windows安全方案下，软件是按照软件使用者授予它的权限在计算机上执行操作。黑客们不断寻找歪门邪道来把恶意软件偷偷安装到我们的系统上，并欺骗我们的系统使用特权用户身份运行该软件。相反，托管代码允许一种完全不同的安全模式；软件是根据其自身而非用户的特征获得授权。软件本身的属性（如软件的编写者、软件来源以及软件位置）被用来规定和管辖允许软件所做的事情。.NET平台实行了这一新方案。托管代码只能在.NET平台上运行，因此必须受这一新的严格监督和限制体制的约束。当然，黑客会继续其捣乱行径，但新的 .NET 安全模式在阻止流氓软件和安全破坏方面是一种充满前途的新武器。我们常说，要解决问题而不要制造问题。托管代码就是在解决问题，而购买老式基于COM的软件就是在加重计算机行业面临的安全问题。

更好的连接性。新的.NET 软件开发工具和托管代码基于标准的性质使得开发采用先进连接技术（如 XML Web 服务）的系统更加容易。我们的未来显然会是一个充满了相互协作的系统的万花筒，这些系统或大或小，通过网络相互连接，彼此交换信息和提供服务，同时也为人类提供服务。创造这一高水平的连接功能和达到必不可少的稳健性需要新的组件、工具和以标准为导向的方案。使用老式组件和工具且不遵循标准去实现这一目标是一件不可能的事情。.NET 提供了用于开发下一代软件以实现新的“互联世界”的蓝图和工具集。

更快的软件、更快的开发、更轻松的部署。.NET是一种全新的软件技术，其创始之初就志在利用突破性的技术（如 XML）和消除 Windows 软件中在过去 20 年间形成的效率不高的老旧内层。托管代码精简、快速而轻便。这些特点有助于应用开发和部署方面的创新。而且，.NET包括众多供开发人员使用的工业级强度（industrial-strength）的现成组件，开发人员可以节约开发时间，同时开发出更稳健和强大的应用。托管代码的特点与行业标准的

结合带来了新部署模式（包括“无接触”部署）和能够自我更新的软件。

更低的投资成本。IT 部门每天都疲于修复问题和面对植根于老式基于COM的Windows软件技术的限制。在运用系统升级内容时，公司在防止新问题的工作中投入了大量公司资源。在利用新技术方面的创新和进步被实施新变更时的过分谨慎和悲观主义而减慢。旧Windows系统固有的问题转化为更高的IT成本和机会丧失成本。上文描述的托管代码的每个优点都有助于通过更轻松的软件开发、更轻松和更少麻烦的部署、安装、维护以及更高的安全性来降低成本。

托管代码的优势显而易见，在公众意识和公司董事会会议室中，对这一事实的承认正在逐渐增加。就像基于 MS-DOS 的软件市场在一天被宣布死亡一样，总有一天软件购买者也会只购买基于托管代码的软件。

.NET是一种新的和更好的软件，也是一套用于开发新的和更好软件的工具。而且，.NET使得开发和部署所有人都在谈论的XML Web 服务更加容易。不要低估Web服务的重要性，现在Web服务备受关注的事实是许多ERP公司面临同一个问题的征兆。

鳄梨比喻

从开发人员的眼光来看，ERP软件包的结构就像一个成熟的鳄梨，在一个硬质核心外面有一层软质包裹。鳄梨的软质果肉就像围绕着ERP的界面。交互式图形用户界面包括本机代码和提供网页的Web服务，以及用于通过电子数据交换（EDI）和其他企业应用集成（EAI）机制连接同类系统的非交互式用户界面。想连接另一个系统或者拥有新的视觉外观？只需修改界面就可以办到——这些界面是相当具有延展性的。相比之下，ERP系统的硬质核心是实现业务逻辑的软件，定义每个业务流程与流动规则以及用于确保数据准确性、完整性和适当性的规则与惯例的大量集合。业务规则对公司数据库“堡垒”进行保护，阻止来自外界的不合逻辑的数据破坏公司的信息本身及其价值。业务规则是ERP系统的核心，比数据库设计更宝贵，因为它们更难创建和维护。业务规则及其在其中得以实现的框架不是随便可以改动的。

为更好地理解作为ERP系统的皇冠宝石的业务规则，请考虑一个看似简单的库存提货事务的例子。 我们假设一定数量的某一特定库存项目被提走。这时会发生什么？当然，

ERP系统会减少现有库存量。另一条业务规则会指示在库存事务历史记录表中增加一条该事务的记录。还有其他业务规则执行总账的借方和贷方过账。如果刚发生的提货事务是一个预先计划的事件，则可能还会写下一条记录以减少或取消对提取库存的等待。另外还有业务规则处理任何批次跟踪或序号分支。如果提货是作为针对客户订单的发货而发生的，则有许多其他业务规则发挥作用。这些规则会写下一条发货日志，减少销售单上的“等待发货”（remaining to ship）数量，触发发票系统和应收账款，发送一条信息以提醒客户，更新对该客户的发货的销售历史记录，计算售出货物的价款，写下针对销售人员的佣金等的记录。向哪个总账科目过账？价款如何计算？什么销售人员获得佣金？是否缴了什么税？涉及的批号是什么？业务规则涉及所有这些问题以及更多问题，因此，在本例中，当提货事件发生时，构成 ERP 系统的所有相关子系统和表都得到正确的通知，数据准确性和完整性也得以保持。

这些业务规则是现代ERP 软件系统的核心，是客户期待ERP系统提供的价值的源泉。ERP系统的大多数开发和测试工作都花在确保这些业务规则的正确性、完整性、被正确调用以及所有事务的顺利而高效的执行。除了最表面的软件功能以外，ERP系统中的所有功能都是以业务规则的形式实现的。如果说数据库是ERP系统的心脏，那么业务规则就是ERP系统的灵魂。

一些需要隐藏起来的东西

在今天的许多ERP软件包中，真正的工作仍然是由一个老旧和过时的软件主体——很久以前用旧软件语言编写的业务规则——而完成的。这些旧“宝石”一直被珍藏，夹在较新的软件层之间，与几十年的技术变化相隔绝。通过在其下面插入一层新代码，诞生于已废弃操作系统之上的旧软件还是可以运行于现代 Windows 操作系统之上。而且，通过以现代图形用户界面和EDI代码层进行包裹，它可以与外界相隔绝。

因此，在上世纪70年代和80年代期间开发的软件还居于许多公司系统的中心。这些软件的设计者当然不知道我们目前拥有的微软 .NET、XML、2.3 GHz 处理器和其他所有技术与标准。他们熟知的是过程式语言、8 MB 内存、1200 Kbps 调制解调器、简易终端设备以及计算机机房的活动架空地板。但无论是当时还是现在，在业务规则方面的投资都是巨大的，而且这些老代码看起来依然有用。所以，在谈到ERP系统的中央核心时，“如果它没有破裂，就不要修补它”一直是大家奉行的原则。或者也可以这样说：“不要更新它，把它包裹起来就行了。”

具有讽刺意味的是，对拥有旧业务逻辑代码的一些公司来说，转而对用户界面使用互联网浏览器实乃天之所赐。从营销角度说您有一个 Web 浏览器界面听起来相当“先进”。而承认是用“屏幕抓取器”（screen scraper）包裹了旧代码则没有这种效果。但从技术上讲，效果是相同的，都不过是给旧软件的核心改头换面而已。

后来，在 .NET技术出现后，以新的 .NET等同物来更换其包裹软件和打出 .NET旗号对ERP公司来说相对容易一些。XML Web 服务的情况也一样。实际上，在网页背后和用托管代码构建的新面具底下活动的仍然是大量老旧的 .NET前代码，且经常是上世纪90年代之前的软件。这些旧软件的核心就好像“家丑”一样，而托管代码的优点不多久就会使之暴露无遗。

现在，我们把鳄梨的比喻暂且放下，先来看一看旧软件架构问题的另一个方面。

两个并不比一个好

大多数ERP系统的核心架构是在很久之前设计和编写的，是在我们认识到新互联世界的要求之前。连接和企业应用集成（EAI）支持是事后的想法，是对被小心地应用在旧核心软件边缘的功能的考虑。

按照定义，业务规则必须全面而准确。ERP公司最不想做的事情就是必须维持多套业务规则，但一些ERP公司确实在这样做。 ERP产品的基础是用于处理来自最终用户的每种事务的主要业务规则集。可能存在这些规则的分开实现以处理同一事务——当事务通过EDI或 Web 或者从车间的手持无线设备抵达系统时。

今天EAI策略的典型做法是使集成机制（例如 Microsoft BizTalk）执行其自己的数据验证，并在把数据传送至目的系统前应用部分或所有业务规则，因此，存在于目的系统中的业务规则基本上在 EAI 机制中被重复。这不是没有代价的。此种逻辑重复增加了集成成本，减慢了软件系统的“进化”速度，并使其与其他系统的连接更脆弱且维护成本更高。

业务规则重复的成本隐藏在 IT 和 EAI 预算以及 ERP 软件价格标签中。维护及增强成本由于这一重复而膨胀，更要紧的是，公司的灵活性和敏捷性因此受到严重限制。对特定业务规则的任何新变更必须在该规则存在的所有地方进行。

只是寻找某一给定规则发挥作用的所有地方通常都要花费一定的时间。随着业务的发展，保持这些规则实例的同步是一项困难的任务。

我们生活在一个快速变化的企业环境当中。公司并购与清算经常会搅乱业务关系和公司标准。技术、行业标准及最佳实践也在不断地发展演变。实施业务规则的软件必须随业务的变化而变化。而这一变化的速度有增无减。老式 ERP 架构抗拒改变，是企业实现敏捷性的重大障碍。它们阻挡了通往灵活集成和普遍连接的新世界的道路。

面向 21 世纪的 ERP 架构

今天，过多的讨论集中在“连接”（getting connected）以及从一个系统的包裹或外表面到另一个系统的外表面的通信上。诸如 Web服务、消息队列、电子数据交换（EDI）和 BizTalk 等传输和通信技术吸引了所有注意力。旧核心代码和重复业务逻辑的问题——亦即大多数系统都存在的一个隐密问题——被撇到了一边。大家只是想当然地认为ERP系统都使用了N层架构（包括其对共享中间层业务规则的强调）而且这就是需要讨论的一切。但许多老旧的N层架构存在连接性方面的挑战且需要业务逻辑重复才能蒙混过关。从长远来看，整个ERP架构的设计都应时刻考虑到EAI。面向21世纪的ERP系统应当是“为连接功能而生”（Born to Connect）而非“被迫提供连接功能”(Coerced to Connect）。

现代ERP软件架构处处都是为了有效地实现业务规则，处处都是关于这些规则在系统内是如何封装和调用的以及关于如何使它们方便地显露给合作伙伴系统。这就是为什么 .NET 和基于标准的新世界要求ERP系统进行彻底的重新设计。现代ERP系统的特征核对清单必须包括：

100% 托管代码。真正现代的 ERP 软件包应完全由托管代码构成。只有如此它们方能享受版本并存功能、新的安全性和其他所有优点，以及在 .NET 技术下更低 IT 成本才成为可能。

单一业务规则集。ERP 系统业务规则的封装方式应避免业务规则的重复。理想情况下，每条规则应当有一个拷贝，来自所有来源的所有事务均使用这个实例。一个业务规则对象集应当能够处理所有来源的事务，包括来自交易伙伴以及 ERP 系统自己的用户界面——无论是强大的 PC、互联网客户机还是无线手持设备——的事务。当在用户交互情形下操作以支持丰富用户界面体验，并且还在隔离情形下操作以处理批量数据，而没有需要对之显示出错信息的用户时，业务规则必须能够正确地识别这种情况。单一业务规则集的明显优点就是低

得多的开发和维护成本，以及在连接其他系统时的更高速度和敏捷性。

捆绑验证规则与处理规则。建立强大的单一业务规则集的关键在于保证为了验证目的而测试引入数据（incoming data）的逻辑能够被执行，针对每个事务的处理和数据库更新的规则方便地调用。允许调用服务于用户界面的同一验证逻辑来保护非用户交互式情况（如 EDI 和 BizTalk 情景）下的系统。

业务规则的动态编排。ERP 软件架构的理想特征是它允许业务规则像一个专家网络那样行事，每个专家在其力不能及之时把工作交由另一个专家来做。每个专家对其同伴所做的工作一无所知。系统架构仔细地编排事务处理过程，使每个专家发挥作用，完成其特定工作，然后转交给下一个专家。一个强大的软件架构会让这一切自主发生，然而能够保证整个事务的进行井然序条，如果发生问题则进行“回退”（roll back）。

支持丰富客户端界面和精简客户端界面。丰富客户端界面还是精简客户端界面各有千秋。软件架构应当对基于本机代码和基于 Web 浏览器的用户界面同时提供强大的支持而没有任何开发时间障碍，从而允许软件开发人员自由选择用于开发工作的合适工具。

分布式处理。软件架构应当允许事务处理完全在一个 CPU 上完成，或者分布到许多系统以分散处理负荷。换言之，同时提供“向上扩展”（scale-up）和“向外扩展”（scale-out）功能。

轻松定制。现代 ERP 软件架构应当提供一个框架，在其中可以轻松地对业务规则和用户界面进行定制以满足单个公司的需要，而且这些定制与标准业务规则和用户界面代码保持隔离，以支持软件升级和定制代码的轻松维护。如果存在单一、通用的业务规则集，则可以通过修订公司定制规则的唯一实例来实施变更。

对基于标准连接的本地支持。XML 现在是无可争议的数据交换标准。面向 21 世纪的 ERP 软件架构在与外界“对话”时应当使用 XML 作为沟通语言——通过发布 XSD 格式的内向（inbound）数据和把内向数据接收为 XML 格式，以及通过以 XML 格式发送附带 schema 的外向（outbound）数据。数据在系统内部流动时可显示为 XML 格式为创新打开了大门。不过，系统还应当支持与老旧系统进行以分隔符分隔的文本文件的双向交换。

分散联合。与外部系统的交互应当在业务单据粒度水平上发生，这些业务单据包括发票、销售单、预测、发货通知等。今天的一些 ERP 系统仅向外界显露一个细粒度（fine-grained）的界面，这要求外界具备有关 ERP 系统的细节知识。这些知识虽然允许“紧密”连接，但是会导致界面整体比较脆弱且开发成本高昂。

把基于标准的丰富数据集显露给外界。虽然分散的业务实体应当在更正式的业务单据层面进行数据交换，但软件架构还应当允许外部合作伙伴通过开放、基于标准的机制获取详细数据，以解答自己的问题。简言之，在安全控制之下，任何 ERP 数据或流程应当可以轻松地通过 XML Web 服务显露给外界。软件架构应当显露针对系统中主要实体和事务的常见业务逻辑对象，这些实体和实务包括客户、客户结单、供应商、供应商绩效、物料项目、库存、工作单、采购订单、项目等，以允许交易伙伴轻松查找和检索其所需信息。

针对内向数据的单一焦点。对于内向（inbound）单据，软件架构应显露一个支持 EDI 风格的批量事务和“整批”（batch of one）事务的单一“接收点”（catchers mitt）。这样一个单一输入点应当接收来自有效及获准来源的正确（发布）格式的所有内向 XML 单据。集成器（Integrator）只是简单地传送正确格式（XSD Schema）的 XML 单据或等效文本文件，并使业务规则被调用，以成功地将其处理入系统，或者暂停处理以等待错误复查和纠正。单据可以来自 BizTalk 或其他任何企业应用集成（EAI）机制。

提供基于事件的通信。除了支持与交易伙伴谨慎的双向数据交换之外，软件架构应当激发可被交易伙伴听取的事件。当出现情况或某些条件得到满足时，协作系统可响应彼此的需求。换言之，软件架构应当允许交易伙伴要求或“pull”（拉）其所需的数据，同时在情况要求时将事务和基于事件的消息“push”（推）给它们。

总结

微软 .NET 技术实实在在地证明，21 世纪的商业将处处是丰富和经常的“临时”（ad-hoc）连接，被运行在我们已经开始期待的体积更小和功能更强大的硬件设备上的协作系统所利用。

微软 .NET 技术是唤醒行动的号角，而一些公司错失了行动机会。要在接下来的几年取得进展和成功，任务关键型企业软件系统（如 ERP 系统）的设计必须彻底面向连接和集成，以最低成本提供最高的企业灵活性。此种灵活性的支持者就是这样一种软件架构，其业务逻辑表现方式可以避免为了支持多个事务源而建立多套业务规则的需要。

.NET 技术还与更好的软件有关。托管代码运行速度更快、更小更轻便、可以更好地抵抗黑客攻击，并且易于部署、安装、维护、升级，必要时可从您的计算机予以卸除。托管代码绕开了我们曾经遇到过的许多软件 bug 和安全问题的根源。托管代码的投资成本更低，因而在将来具有战略价值。

谨防伪装者。商业软件供应商绝不会宣扬他们的短处。ERP 系统的彻底改造并非易事，今天的系统在架构设计上并不能很好地满足下一阶段计算和通信的要求。包括 Web 服务在内的许多技术可以让旧软件伪装成现代软件系统。ERP 软件购买者在选择产品时必须慎重，以购买能够充分利用技术来提供最高敏捷性和最低投资成本的系统。