Vantage 8.0 Architecture: Next Generation Manufacturing Software
Abstract

Vantage 8.0 is the next generation manufacturing solution developed by Epicor Software Corporation. It builds on the successes of the Vista, Vantage, Avanté and other associated discrete manufacturing product lines by introducing a total manufacturing solution for the midmarket.

It is built to leverage the Microsoft® .NET™ Framework™ and Progress® OpenEdge™ Business Platform in order to provide a leading-edge technological solution that offers Epicor’s manufacturing customers a framework that will support their business needs now and in the future.

Through its utilization of Progress OpenEdge for the back-end business logic and Microsoft .NET for presentation deployment, Vantage 8.0 delivers unrivalled flexibility and performance for both our developers and customers.

The following diagram represents the main components of the architecture.

The following paper explores the reasons why Epicor selected this framework, and how this technology will help ensure Epicor’s customers remain ahead of their competitors.
Table of Contents

Table of Contents .......................... i
Why a New Technology Platform ............. 3
Architectural Models ....................... 3
Why Adopt This New Model? .................. 4
Why Deploy a Service-Oriented Architecture? .... 5
Implementing the Supply Chain within an SOA .... 7
Technology Foundation ..................... 9
The Key Technology ....................... 9
What is Progress OpenEdge? ............... 9
Database Independence .................... 10
Operating System Independence .......... 10
What is the Microsoft .NET Framework? .... 11
Why Progress OpenEdge and Microsoft .NET .... 12
Business Requirements ................... 12
Reliable and Scalable ..................... 12
Rich User Experience ..................... 13
Accessible .................................. 13
Low Cost of Ownership ................... 14
Collaborative ................................ 15
Messaging .................................. 15
Transformation ............................ 16
Application Interfacing ................... 16
Exception Handling ....................... 16
Security .................................... 16
Flexible .................................... 16
Operating System ......................... 16
Database ................................... 17
Personalization/Customization .......... 17
Functional .................................. 17
Future Proof ................................ 18
Technical Architecture .................. 19
Introduction ................................ 19
Base Server Elements ..................... 19
Application Server ......................... 20
Database Server ............................ 20
Business Logic Layer ...................... 20
Why a New Technology Platform

Architectural Models

When starting a discussion regarding enterprise software, one should first understand some common software architectures. There have been many architectural models in the history of computing, but three in particular are important for our discussion of business applications. These models may be known to you by various names. We apologize if the terminology you are most familiar with is not used.

The first model is that used by mainframe computer software. In this model, the entire application resides on a single machine. Input and output is accomplished through the use of terminals. The structure of this architectural model is monolithic. Monolithic means that the application is, for purposes of deployment, a single program that runs on one machine. Applications using this architecture model also run on mini-computers and it is the architecture for stand-alone PC applications. Over time there have been variations on this model for multi-processor machines, but the monolithic nature of the application remains—the application is a self-contained unit with all processing happening in the application itself.

The second model is the client/server model. In this model processing is no longer monolithic. Some of the application resides on the client and other portions on the server. There are variations of the processing mix between client and server, but in all cases both the client and server are necessary for the application to function. While this model added flexibility and ease of use in application design, the client and server remained tightly coupled. However, one advantage of this model is the ability to create more than one client to interact with the server. Many companies did this to support multiple client platforms (i.e., Windows, Mac, UNIX, etc.).

The latest model, which has recently received a great amount of press, is described as a “service-oriented architecture,” or SOA. While the concepts behind SOA have been around for many years, the execution and deployment have been greatly enhanced by the advent of Web Services.

A service within an SOA is an exposed unit of functionality with three properties:

- The interface to the service is platform independent
- The service can be dynamically located and invoked
- The service is self-contained

Services are self-describing, self-contained modular applications that can be mixed and matched with other services in an SOA. Services are applications that fulfill a specific task or a set of tasks, and that work with many other services in an interoperable manner to carry out their part of a complex workflow or a business transaction. Services interact in a language and operating system independent manner—reducing the complexities of integration.

Service-based architectures are considered loosely coupled and modular. As a result, one service can be replaced with a new one without affecting the client or other services that talk to it.

Web Services are a specific type of service that read and write eXtensible Markup Language (XML) based on the business logic embedded in them. The Web Services model encourages breaking server-side functionality into smaller pieces that exchange XML using Simple Object Access Protocol (SOAP).
Why Adopt This New Model?

Epicor has always been a technology innovator, conscious of how technology would work and be deployed in the markets it served. For example, it would be unlikely that a software vendor selling into the emerging enterprise space would experience great success if their solutions were based on an Oracle database. Database administration for Oracle generally is more expensive and time consuming than administration for a database such as Progress RDBMS or Microsoft® SQL Server. Examples of Epicor as a technology leader include:

- Epicor released the first midmarket Microsoft Windows manufacturing solution.
- Epicor released one of the first manufacturing solutions to utilize relational database technology.
- Epicor released one of the first midmarket manufacturing applications on UNIX and Windows® NT.

In fact, there are still few competitors that can boast as many concurrent users running on Windows NT as Epicor.

Because technology is constantly evolving, Epicor is constantly looking at future trends to ensure we protect our customers’ investment and provide them a competitive edge.

Epicor’s next generation architecture continues its past themes of choice and flexibility. On top of this, Epicor utilizes a framework that makes the technology of the application invisible to the end user. By invisible we mean the ability for end users to select their primary tool for customization and deployment without being limited to the technology in which the product was developed. For example, a user may want to write a new module in Visual Basic and hold information in SQL Server, but require data and business logic from the core Epicor application. The framework will enable users to do this even though the core Epicor application may be running on UNIX, sitting on the Oracle database, and written in Visual Studio .NET. We want our users to select the technology that will make them more effective and competitive in their field of business. Creating a service-oriented architecture is the key to providing this flexibility to our customers.
Why Deploy a Service-Oriented Architecture?

In the world of computing, service-oriented architectures may well prove to be as significant a milestone as any other to date. Unfortunately, as important and revolutionary as SOAs are, the basic concepts surrounding the technology still remain one of the most fundamentally misunderstood. Many people tend to think Web Services are limited to the publishing of software services to the Internet. While this is certainly an important role that Web Services will fill, it is only a small part of a much bigger picture. At Epicor, we believe Web Services, when deployed within an SOA, represent a significant and fundamental evolutionary change in the way that all distributed systems will be created. In fact, we are so convinced that service-oriented architectures will be the standard for the distributed systems of tomorrow that we decided to deliver on that future—today. What does this mean? It means Epicor customers are among the first organizations anywhere to fully exploit the many benefits of a service-oriented architecture. It also means that as their businesses grow and expand, they can rest assured their ERP solution can grow and expand right along with them.

In its most basic form, an SOA need only involve two entities:

![Diagram of Service Consumer and Service Provider]

The above example shows the Vantage 8.0 client interacting with the application server to request a parts listing, validate customer account information and update a sales order (each of which are services offered by the application server).

A consumer of services, however, is not limited to a single provider. In the model below we see multiple service providers available to the consumer.

![Diagram of Multiple Service Providers]

This model might allow a client to make service requests to a number of instances of the application running within the enterprise, either as different databases on a single application server or multiple application servers. Examples of this might include requests for inventory levels across companies, placement of a sales order on the appropriate database or consolidation of financial information from divisions.
Our services model is not constrained to a single type of consumer. In the model below, we see that many types of consumers may use the architecture as long as they are able to interact with the service.

Here we see that not only human interface devices such as PDAs and PCs are interacting with the services, but also other servers. Examples of this capability include the ability for one instance of the application to satisfy demand created elsewhere, perhaps by raising back-to-back purchase and sales orders.

Web Services also provide the potential for creating interfaces for integration that were previously very difficult due to proprietary interfaces (APIs) and language specific data structures. Because the SOA incorporates Web Services using XML (which is text-based) to share information and SOAP to communicate in a way that is programming language independent, it is possible to link other applications as service consumers or providers via an integration backbone often referred to as an Enterprise Service Bus (ESB).

One key advantage of Web Services is a new level of interoperability. Because Web Services interact with each other or with clients via SOAP and XML, it doesn’t matter which language your Web Services are written in (C#, Visual Basic, or Java) or what platform you run (Windows, Linux, or UNIX)—they can talk to other Web Services. This level of abstraction from operating systems and programming languages is critical to understanding the possibilities introduced by Web Services.

The fact that Web Services function over the Internet is also important. Web Services use the most ubiquitous networking transport available. If your server has access to the Internet, then the Web Services that reside on it can talk to other Web servers anywhere on the Internet.
This allows increased componentization of applications with minimal concern about where components are located and how to connect to them.

Finally, service-oriented architectures are expected to save companies money. According to Gartner, Web Services will drive a 30-percent increase in the efficiency of IT development projects. According to Forrester Research, Web Services “crush” the cost of business interactions by replacing manual communications to save time and money, cutting the cost of connecting to partners, making internal services available across departments and geographies, and enabling new kinds of business collaboration.

**Implementing the Supply Chain within an SOA**

Supply chain management (SCM) is the practice of coordinating the flow of goods, services, information, and finances as they move from raw materials to parts supplier to manufacturer to wholesaler to retailer to consumer. This process includes order generation, order taking, information feedback, and the efficient and timely delivery of goods and services.

For many years, ERP systems have continued to refine practices such as Just-in-Time (JIT) manufacturing and vendor managed inventory in order to streamline the flow of goods and thus reduce costs. However, until the recent advent of XML document processing and the ability to transport documents over the Internet, the flow of information between systems has been mired in the complex arena of electronic data interchange (EDI) which has changed little since the 1970’s.

Even with the use of XML to simplify the EDI document format, the EDI paradigm is still such that the systems are loosely connected via a document-based interface. When your customer requests a quotation, or wishes to place an order into your manufacturing system, they may send a document in an agreed format containing the structured data for that transaction. That document is delivered to you, either via a Value Added Network (VAN) or over the Internet. When the document is received, an adaptor (a piece of integration software) evaluates the document, extracts the data and interacts with the business system to perform the necessary function. A response, if any, is constructed and delivered back to the VAN, or through the Internet to the customer.

The EDI interface is prone to errors and complexity due to potential problems with delivery of documents and problems when document formats change or are inconsistent between partners.
Systems that support Web Services, on the other hand, can easily be connected into a service-oriented architecture in order to allow direct interaction with the business logic. The SOA allows functional areas within the manufacturing system to be made available to other authorized systems. Using Web Services, the quoting functions of your system can be made available directly to your customers’ purchasing systems, allowing direct access to the data and business rules that form a quote. Essentially, your computer system and your customer’s system function as one system through the SOA. Web Services will truly transform your relationship with your business partners to a real-time relationship, ensuring your business is running 24x7, both sending and receiving transactions. Imagine trying to do business today without a phone or e-mail. You just wouldn’t be competitive, as you wouldn’t be able to react in time. Web Services will be just as pivotal in the future.
Technology Foundation

The Key Technology

Vantage 8.0 uses two key technologies as a foundation that offers users unrivaled flexibility, reliability, scalability, and low cost of ownership. These two technologies are:

- Progress OpenEdge Business Platform
- Microsoft .NET Platform

What is Progress OpenEdge?

The Progress OpenEdge Business Platform is a complete, tightly integrated platform for the rapid development, deployment, integration, and management of high-performance business applications. OpenEdge is designed specifically for productivity and cost effectiveness. Productivity is gained through the highly-purposed business language and the tight integration between all of the OpenEdge components. Business application developers can very quickly design, build, extend, and modify business logic and the underlying data on which it depends. Logic development in OpenEdge is typically many times faster than in other less purposed languages. Cost effectiveness is delivered through low costs of deployment, low administration costs, lower hardware requirements, and great reliability and scalability. All of this is delivered with platform independence and support of industry standards to maintain the greatest number of choices for the end user.

The OpenEdge platform includes a high-performance embedded database, application servers, data servers, a fully integrated application development environment and framework, Internet-ready messaging, and enterprise-class application management. Offering a complete solution and guaranteeing maximum interoperability, OpenEdge frees developers to focus on the application and users to focus on the business. See www.progress.com for more information on Progress OpenEdge.

Epicor and The Progress Company have worked together for many years to provide leading edge ERP applications for midmarket manufacturers, and the Vantage 8.0 architecture leverages that investment.

Progress OpenEdge 10 will provide support for the .NET Framework. The outcome of this new product is an application framework that allows systems to utilize the very best features of .NET for smart client support and Web service capability, while retaining the strength of the OpenEdge platform for rapid application development and open systems deployment.
Database Independence

The Progress Relational Database Management System (RDBMS) is a fully relational database designed to provide outstanding performance, linear scalability, and extremely low administrative efforts and cost. It provides complete access from the OpenEdge language and from SQL tools and environments. But there are situations where customers of OpenEdge might need or prefer to use an alternative database. Recognizing this, OpenEdge provides optional support for a number of databases, including Microsoft SQL Server and Oracle. So, in addition to providing the framework for the interface and Web Services, the OpenEdge platform also provides full database independence capabilities through the Progress DataServer.

Operating System Independence

For many years, OpenEdge has allowed operating system independence for those applications written in its toolset. OpenEdge users are able to deploy their products on Windows, UNIX or Linux environments. The operating system independence will allow Epicor to deliver solutions that deliver unrivalled flexibility and scalability.
What is the Microsoft .NET Framework?

The Microsoft .NET Framework is a complete set of products including development tools, a common language run-time (CLR), and a variety of servers (operating system – Microsoft Windows; database – SQL Server; Web – Internet Information Server, and integration – BizTalk).

Microsoft defines .NET in this way:

*Microsoft .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web Services: small, discrete, building-block applications that connect to each other—as well as to other, larger applications—via the Internet. .NET connected software delivers what developers need to create XML Web Services and stitch them together. The benefit to individuals is seamless, compelling experiences with information sharing.*

For more information on Microsoft .NET visit: [http://www.microsoft.com/net/defined/default.asp](http://www.microsoft.com/net/defined/default.asp). The following diagram is from Microsoft and helps describe the .NET Platform and its pieces. This diagram and further descriptions can be found at [http://www.microsoft.com/net/defined/whatis.asp](http://www.microsoft.com/net/defined/whatis.asp).
Why Progress OpenEdge and Microsoft .NET

Business Requirements

Epicor decided to use OpenEdge and Microsoft .NET to build its next generation ERP applications based on a service-oriented architecture. But why? The technologies for the framework were selected by analyzing the technology needs of small- to medium-sized companies. This section discusses why those technologies were chosen.

The main technology requirements for Vantage 8.0 were as follows:

- Reliable and Scalable
- Rich User Experience
- Accessible
- Low Cost of Ownership
- Protection of Investment
- Collaborative
- Flexible
- Functional
- Future Proof

Reliable and Scalable

Downtime is very expensive. All companies want applications that work and keep working over long periods of time. Reliability and availability are primarily factors of the quality that a software vendor builds into their solution. However, some frameworks lend themselves to increasing reliability and availability by allowing for redundancy and fail over. Many companies now run 24x7x365. Even though a company's employees may not be on the premises 24x7, many companies' systems are up and running 24x7 to service the needs of global operations and their customers, who can access the system via the Web. It is therefore imperative that the architecture is reliable and does not need to be shutdown for maintenance or backup. The Vantage 8.0 architecture is designed to meet the needs of these companies. The framework allows online backup and complete fail over options to ensure that availability meets the specific business requirements of our customers.

As we talk to potential customers, it is rare to encounter small and medium enterprises that have no intention to grow. This is not surprising given that one of the primary goals of any enterprise solution is to help a business grow. But the last thing companies want to do as they expand their reach is to replace a critical system such as their enterprise solution. Also, in the past, many companies would size their system based on the number of employees accessing the product. In the modern collaborative economy, however, it is difficult for a company to measure how much of their system’s capacity they need to set aside for the needs of their business partners who may be constantly accessing the system via portals, Web Services or mobile solutions. This means that scalability is more important than ever.

The OpenEdge platform was built specifically for high performance in online transaction processing (OLTP), the processing on which business applications depend on. The OpenEdge platform is in use by customers ranging from as small as five users to more than 3,000 concurrent users. Real-world scalability tests with OpenEdge show a remarkably linear
scalability that is based on careful database and platform design combined with very efficient use of hardware and operating system resources.

The Vantage 8.0 architecture will ensure that customers are able to deploy their application on the technology framework that suits the size of their business today, with the firm understanding that they will be able to scale up or indeed down without need to change their business-critical enterprise application.

Rich User Experience

One of the important aspects of the architecture is to deliver a product that can be easily deployed and run over low bandwidth networks. Many companies have delivered HTML-based interfaces in order to do this. These interfaces generally do not offer the rich user experience of a traditional client server application. The Microsoft .NET smart client, however, delivers a very rich user experience along with simple deployment and the ability to be used on a low bandwidth network. The smart client approach is the methodology being deployed with Vantage 8.0.

Vantage 8.0 is a fully n-tier application, with a .NET client working against a .NET interface layer to a Progress application server containing the application business logic. Since the client is abstracted from the application by a .NET interface layer, the architecture supports the possibility of using any .NET-compliant smart client against the interface, and therefore the application as a whole.

The primary user interface that is delivered with Vantage 8.0 is built entirely using the latest generation Microsoft tools available within Microsoft Visual Studio .NET, in conjunction with industry standard presentation layer components from Infragistics, Inc. The .NET Framework provides robust capability for data management (through datasets), transport (via open client), exception management (through exception handler), and presentation (through Visual Studio .NET). It is these capabilities that make it more achievable now than ever before to effectively separate the client from the application code and database layers.

The objective of the smart client is to provide a highly functional, flexible, intuitive, and attractive user interface that works through the Microsoft .NET interface layer against business objects on the application server. While the smart client is the primary means of accessing the business logic on the application server, other smart clients will also be able to access that business logic. Examples are Web browsers (through ASP.NET), .NET-enabled handheld devices (such as bar code scanners) and Web Services. In order to facilitate this flexibility, the architecture places great emphasis on ensuring that the business logic and rules are centralized on the application server, rather than being built into the client user interfaces.

For more information on smart clients see:

Accessible

Using a smart client to provide rich functionality does not eliminate the need companies have for easy, flexible access to ERP information. This need is often expressed as needing “Internet-based applications.” What companies are saying is that they need access options to support a variety of user types. These users may be on a local area network (LAN), or they may be part of a remote office with access to the corporate network via a wide area network (WAN) using long distance dial-up lines, frame-relay, or virtual private network (VPN). There are also times when a user may be remote and need to connect to the corporate network from home or a hotel room. In the case of the remote user, the Internet is usually the easiest way to connect. In all these cases, providing Internet access to the application is a solution to the real problem of providing easy access to a variety of users.

One of the benefits of the .NET platform is the ability to build applications based on Web Services that have either a smart client or a browser-based user interface (or both). This
capability has allowed Epicor to develop a smart client application that uses Internet standards as the transport for talking to Web Services that serve up data, take input, perform validations, and update the database. This means that Vantage 8.0 will work in an Intranet, Internet, or extranet environment—or all three at once.

Low Cost of Ownership

Low cost of ownership has always been one of the fundamental principles that Epicor has brought to the marketplace—the ability to deploy sophisticated solutions without complication. The technology will continue this tradition. Most of Epicor’s customers do not have a full-time database administrator. This is a result of the database and framework technology that Epicor has selected in the past.

When evaluating software solutions, both up-front and ongoing investments must be considered. This includes the initial acquisition cost, as well as maintenance, administration, training, and upgrades. Solution flexibility, ability to leverage existing technology resources, ability to adapt to evolving business, and IT demands all have a major impact on the total cost of ownership.

Vantage 8.0 provides the seamless integration and open standards support needed to connect the IT environment to users, customers, suppliers, and partners. It minimizes administration and maintenance while at the same time maximizing a company’s resources and productivity. Risk is reduced with a high-performing, reliable, available, and scalable solution. No matter how you look at it, Vantage 8.0 provides the most cost-effective platform.

Before investing in any system, it is imperative that a company adds up all the costs: the costs of acquisition, implementation, administration and support, and the hardware and operating system requirements. When you do, you will discover that the high efficiencies, low administration, high scalability, and unparalleled reliability of Vantage 8.0’s architecture will provide the greatest overall value for the business.

At the moment, there is much media attention with the concept of a single technology stack and how this will help minimize complexity and management. Vantage 8.0 will allow companies to choose the technology layers that fit with their company’s standards. For example, companies who want a complete Microsoft solution can deploy a single Microsoft stack of operating system (Microsoft Windows 2000), database (Microsoft SQL Server) and deployment (Microsoft VS.NET).
Collaborative

The n-tier, distributed model adopted by Vantage 8.0 provides an ideal platform to offer a service-oriented architecture for a manufacturing application.

All interfaces against the Vantage 8.0 application are handled via the SOA which provides the necessary messaging, transformation, application interfacing, exception handling, and security capabilities. The SOA that surrounds Vantage 8.0 is designed to support both online and batch processing business functions to include:

- Direct customer- and supplier-side integration within an extended manufacturing supply chain via published Web Service interfaces;
- Message-based customer- and supplier-side supply chain integration encompassing XML-based messaging and/or traditional ANSI X12 or EdiFact EDI standards (via external converters);
- Interfacing of business functions to extended enterprise systems (such as other enterprise business systems within a large corporate infrastructure);
- Interfacing to the accounting functions of a corporate parent;
- Website integration including transactional functions and portal queries to a database;
- Integration to smart devices such as PDAs;
- CAD- and PDM-based integrations to the engineering database; and
- Application interfacing to external enterprise systems such as CRM, SRM, WMS, etc.

In order to support the above functions, the SOA module is built around the following key areas:

**Messaging**

The messaging functions facilitate the movement of data into and out of the interface architecture. Direct or online interfaces are handled via HTTP and HTTPS protocols and run through the transformation services against the Web service provider. Non-direct or batch functions support FTP, SMTP, Message Queue and other means to direct XML, text, or other formatted documents into in and out boxes for processing. E-mail notifications alert both senders and receivers to status of processing (such as to confirm documents received, etc.).
Transformation
The transformation service provides the mapping capability to transform inbound data to the requisite dataset definitions required for each interface to a Vantage 8.0 business application, and to map dataset definitions retrieved from the application to required XML, text or other standard formats for output. For example, sales order information might be received in multiple formats from multiple customers and may include coded, missing and extraneous information that must be transferred and filtered via external business rules before being presented in a standard format to the business application.

Application Interfacing
This component refers to the ability to place function calls against the business application via the methods supported by the individual business objects. These are the same function calls that are made by the client in order to interface with the business logic and application database. Thus the component is able to retrieve data such as an order or a set of customer data, validate data such as part numbers or supplier detail, and update data such as sales orders or customer detail through the same business logic (and with the same business rules) as the core application.

Exception Handling
The SOA leverages the Business Activity Monitor and Workflow functions in order to provide exception notifications for failed record updates and other exception conditions. Thus, if an incoming sales order cannot be processed due to incorrect or missing data, then a responsible person or role will be notified and allowed to review the problem and provide corrective action.

Security
The SOA protects the Web Services interface from unauthorized use by providing an abstracted layer through which all transactions and queries must pass. Services are managed by agents that define authorized users, messaging strategies, transformation mappings, processing schedules, and notification requirements for each interfaced Vantage 8.0 business function. Agents are licensed on an as needed basis.

Flexible
The Vantage 8.0 architecture allows complete flexibility and choice. There are three major aspects to the system’s flexibility:

- Operating System
- Database
- Personalization/Customization

Operating System
When making a business application selection, it is important not to get boxed in to a particular operating system, hardware vendor, or hardware platform. While there are many options available today, it is very important that you protect your options for the future. Applications that run on a single, proprietary system limit not only the ability to configure the best current system, but also lock the user into a single vision of the future. Many applications have been replaced not due to any problem with the application itself, but from a lack of operating system or hardware support. The Vantage 8.0 architecture will allow users to deploy the Epicor solution on a number of operating systems including Windows NT, Windows 2000/2003, UNIX, and Linux. Many larger companies prefer the price performance of UNIX to NT; the same goes for smaller companies with simplicity of NT over the complexities of UNIX. Vantage 8.0 will allow companies to choose the operating system that best fits their business.
Database
Vantage 8.0 will support a number of database platforms: Progress, Microsoft SQL Server, DB2 and Oracle 9i. The choice of database, again, is a choice we will leave to our customers. The OpenEdge platform provides transparent access to a wide range of heterogeneous data sources. OpenEdge provides a native interface to the Progress RDBMS, the world’s leading embedded database. Progress RDBMS delivers high availability with online backup, cluster fail over and self-tuning features that simplify database management. It easily scales to handle thousands of users and high traffic loads, providing fast user response time regardless of how many users are connected. OpenEdge also provides native access to Oracle and Microsoft SQL Server databases through Progress DataServers. These dedicated DataServers enable exceptional performance and scalability, and deliver advanced features such as optimistic record locking (reducing locking congestion and increase total throughput), asynchronous default database connections, and stored procedure support. Transparent access to any ODBC data source is provided through the Progress ODBC DataServer.

Personalization/Customization
There are two important types of customization for enterprise applications. The first is the ability to change existing capabilities—adding or subtracting fields, rearranging fields to support your workflow, or changing the business rules to ensure your processes are followed. The second type of customization enables the purchaser of enterprise software to extend the functionality of an application. The base customization model provides a sophisticated built-in capability to adapt the user-interface (UI) and to leverage exposed events in both the UI and the application server code, in order to address customization to the needs of most business scenarios.

For those requirements that cannot be satisfied by the built-in model, Vantage 8.0 also offers an advanced customization package which is available at an extra cost and supplies the client source, framework UI and business logic tools necessary to write your own additions and extensions to the application using the Vantage 8.0 framework. The model of publishing “events” allows companies to create new services that listen in on application events and perform additional processing based on those events. For instance, a company could add a service to perform notification to another application, or start processing in another application when a specific event occurs in the ERP system.

The framework exposes a consistent event model for easy integrations to external systems (see discussions on the SOA). This decoupling allows for isolation of software vendor code from integrations, making it much easier to integrate services into the system by listening in on application events and performing additional processing based on those events. For instance, a company could add a notification to a shipping system when a customer’s address is changed, or start processing an invoice in a financial application when a billable support call is created.

Also embedded into the architecture is the Business Activity Manager (BAM). This module allows users to build their own business rules into the application without the need to touch base source code, and thus avoid complex upgrade issues. The BAM allows users to apply business logic to every aspect of the system. For example, if a sales order’s profitability drops beneath a set level, the system will e-mail the sales manager, put the sales order on hold, put the job on hold, put any subsequent purchase orders on hold, and delete any outstanding purchase suggestions. The BAM can handle complex logic and execute it as though it were part of the standard code itself.

Functional
Vantage 8.0 contains all of the rich functionality of previous versions, plus many added features. Through continual evolution of technology and strict product design, Epicor has been able to utilize the database schema and much complex business logic from previous versions.

Basing the business logic on established logic and proven database design is important for a number of reasons. With the size and sophistication of manufacturing solutions, it is very
difficult to write a product from the ground up. Utilizing tried and tested code and database structure will ensure that the product works and the data flows are logical. Rather than develop a brand new product, the Vantage 8.0 product will be a simple upgrade for many of our customers. For customers running other solutions, a complete data migration tool exists that will enable them to migrate their application data to Vantage 8.0.

We have been able to reuse the code in our existing manufacturing solutions because of the careful selection of technology partners in the past. This has enabled us to truly protect our customers' investment.

Future Proof

The Vantage 8.0 framework was chosen because it enables Epicor to continually adopt the latest technologies available, and pass them on to our customers so that in turn they can be more competitive in their field of business.

One of the main objectives given to the original team selecting the new technology framework was the ability to write out a technology partner within a 12-month timeframe. This was to ensure that neither Epicor nor its customers are held hostage by the whims or lack of ambition of a technology partner. Not only will Vantage 8.0 provide complete choice of technology, it will also allow the rapid introduction or removal of technology in the future.
Technical Architecture

Introduction

The following section describes the key elements of the technology stack. Many customers will use Vantage 8.0 as it is delivered out-of-the-box and many of the following elements will never become visible to them. However, to some customers, the following layers will allow for the flexibility and scalability that are key to optimizing their use of the system.

The following areas will be explained:

**Base Server Elements**
- Application Server
- Database Server

**Business Logic Layer**
- Business Object
- Dataset Design
- Database Design
- Customization

**Presentation Logic Layer**
- Forms Design
- Attachments
- Adaptability/Customization
- Language Translation
- Web Services
- Workplace/Dashboard

**Miscellaneous Elements**
- Reporting
- Workflow Management
- Distributed Model

**Base Server Elements**

The base server elements allow Vantage 8.0 to be run on different operating system hardware and physical database platforms. The data server and application server components allow the application developers to develop solutions without being preoccupied with scalability or multi database issues. In essence, the Vantage 8.0 application that runs on Microsoft SQL Server with Windows 2000 is identical to the solution that runs on Progress RDBMS with UNIX.
Application Server
The application server layer exists to encapsulate all of the business logic rules and provide a single point of entry to all application logic irrespective of the interface layer. The application server is implemented using the Progress AppServer technology and the Progress 4GL language. The application server is platform independent and can be deployed on Windows, Linux or UNIX platforms. It supports multiple interfaces, among them Web Services, HTML Web access, and Microsoft .NET. Application servers can be deployed on multiple physical servers. This would allow users to deploy processor intensive applications (such as finite scheduling) on their own application server so that it does not slow down normal users and allow the process to execute quicker.

Database Server
The database server enables the application to run on a number of databases with a single application code set. The first versions will support the Progress RDBMS and Microsoft SQL Server databases. Vantage 8.0 is also being designed to support the Oracle and IBM DB2 databases as well. The database server supports the full capabilities of each database. The database server offers users advanced scalability, availability, performance, and data integrity features (e.g., record locking, transaction processing, and roll back recovery).

Business Logic Layer
The Business Objects are created in the Object Design Tool along with the Progress AppBuilder tools. There are four major elements to a business object:

- Database Tables
- Datatables
- Datasets
- Business Objects

Database Definition
The database definition contains the base level definition of the data. The definition would contain the definition of each table, each field within those tables and the indexes associated with each table. This is the traditional way in which the database design would be done and is a similar layer that exists in most application development tools. The database definition takes place at the database level and is done using the Progress Data Dictionary tool. This will also define the base database definition in other databases too, such as Microsoft SQL Server or Oracle.
Datatable Definition
The datatable definition is an abstracted view of the database definition and contains many extra criteria that improve the development and maintenance of the application. A datatable will point to the physical database definition table, but hold a lot of extra information that is not normally held at the physical level. The datatable will hold the following extra information:

**Foreign Key Links**
This allows information from other linked tables to be automatically made available as a virtual field when looking at the main table. So, for example, the customer table may hold the shipvia code. Any time the customer datatable is accessed, the system will bring not only the shipvia code, but also the shipvia description.

**Field Like**
This allows a standard global definition to be made against a field. This allows a base field’s definition to be changed which will then be inherited by all the fields that point to it. For example, shipvia code is held on the shipvia, customer and ordered table. On the orderhead (sales order header table) and customer tables the shipvia code is defined as “like” the shipvia code on the shipvia table.

Therefore, if the shipvia code is extended in length, it will be inherited in all tables automatically.

**Virtual Fields**
This allows, as the name suggests, the ability to add calculated or virtual fields to the datatable. This allows calculated fields such as order total, extended price, etc. to appear on the ordered table as though they were real fields.
Dataset Definition

The dataset is a further abstraction of the physical way in which data is held. A dataset gathers together many datatables and other related datasets in order to present a complete transaction to the end user. Inherent in the dataset is the way in which all of the datatables and datasets interact to produce a complete transaction. For example, a sales order dataset may contain orderhed, orderdet (sales order detail table), orderdetmisc (sales order detail miscellaneous charges), and orderrel (sales order release table). The dataset in short produces a multi-tiered record which contains all of the information relating to a sales order (all the header information, lines and releases). The dataset produces a business view of the physical data.
Business Object Design

A Business Object consists of its defined datasets, base framework methods, and its unique methods contained in an object procedure. To be used, the procedure and its datasets have to have interfaces defined for them. The dataset’s are called an XML Schema Definition and stored in an .XSD file. The procedure’s is a C# class that is built into a .DLL file. These interface files are true, standard Microsoft .NET Assemblies and can be used in all of the standard ways by .NET developers. The application Business Objects are designed in the Object Design Tool (ODT) and in the Progress AppBuilder tools. The basic structure is laid out in the ODT and the business-specific logic is written in the Progress AppBuilder tool. The structure of an object defines what datasets are contained within the object and how they are related to each other. A base object framework of common logic is used to reduce/eliminate many globally needed tasks (such as create, read, update, delete, list, and retrieve foreign key information). Using the ODT and with just referencing the base object framework a non-programmer can define and build a working business object that can immediately be used in UI development. A business object allows complete control over any transaction; looking at a transaction, deleting a transaction, updating a transaction, creating a new transaction, etc. It can be exposed in many different ways allowing any type of client access to its functionality. That client could be a Windows smart client, Web Service, Web client, etc.
Business Logic Customization
The Business Activity Management (BAM) module allows a company to build in proactive business rules to all the transactions within Vantage 8.0. One of the key goals of the framework is to ensure that customers are able to customize their applications without the need to alter standard source code. Keeping business logic standard allows customers to keep up to date with the latest software releases and therefore maximize return on maintenance payments. The BAM allows users to alter the business rules within the base system without touching source code. The Business Activity Manager has three major components:

Change Log
Identify key data areas of the system that you want to monitor. Track who, what and when changes were made to the data elements.

Send Alert
Monitor changes to key areas of the system and immediately send an e-mail alerting internal or external users of those changes. E-mails can be automatically constructed by the system or users can construct sophisticated HTML-based e-mail messages using related information. Users can specify e-mail addresses manually, use scripts to identify e-mails based on rules or let the system automatically derive them based on a person’s roles associated with the transaction. For example, if an alert has been set up to identify changes to the customers address, a message can be sent to the associated salesman and buyer from the customer’s site. The system will automatically generate the appropriate e-mail addresses from the customer record.

Carry Out Process
Users can define rules against changes to individual or groups of data changes throughout the system. When that change takes place, the new system can be set up to automatically carry out a process to update other parts of the system. These processes can be very sophisticated and, simply put, allow users to embed their own business rules into Vantage 8.0 without touching standard source code. Thus users will have business flows that are entirely customized to their own requirements and still be able to easily take on a new release.

Presentation Layer
Forms Design
Today many enterprise applications are being rewritten to support browser-based clients. These clients are generally considered “zero-footprint” or “lite” clients. One advantage of this model is ease of deploying new functionality. Updates are made to the server and then delivered to the browser client automatically. Epicor chose to provide the ease of deployment and accessibility of a browser-based model while simultaneously providing the extra capabilities and enhanced usability of a rich Windows client.
The client forms are developed exclusively in the Microsoft .NET Framework using the C# programming language. This technology allows the client application to utilize object-oriented techniques including inheritance, polymorphism, and data encapsulation. The benefits of using this technology include: increased extensibility, improved quality, shorter time-to-market, reduced maintenance burden, and increased reusability.

The sample below shows the use of the rich client within the purchase order process of Vantage 8.0.
The objective of the rich client is to provide a highly functional, flexible, intuitive, and attractive user interface that works through the .NET interface layer against business objects on the application server.

Usability features of the Vantage 8.0 client that facilitate the main data entry and retrieval functions of the application include:

- Menu bar providing access to file options (save, open, etc.), tools (associated functions within the application) and Help;
- Tree view providing an at-a-glance summary view of transaction structure and attachments at any level of the transaction;
- Powerful search forms that provide basic, full and advanced criteria sets. Basic and full search windows provide different levels of predefined selection criteria for system tables, while advanced searches allow the user to select from any of the fields available in the database for complex tables and to select from a range of criteria against each field;
- Navigation controls allow the user to work with selected groups or sets of data for maintenance and to move rapidly forwards and backwards through records in the set;
- Multiple data sheets per form that can be “torn off” the main form and positioned at will on the users desktop in order to work with multiple views simultaneously;
- Colorful, visual controls which highlight critical data attributes on the form; and
- Editable grids which allow rapid data entry for all maintenance and transactional forms in the system.

In addition to the usability features above, all forms support sophisticated mechanisms designed to allow flexibility of use depending on preference and other factors. These include:

- Personalization options allowing an individual user to create a specific layout style for each form. This includes the ability to skip fields, change tab order, add columns to a grid, reorder columns in a grid, change color or other properties of a field, and much more;
- Optional settings to remember last position of all data sheets and main form on closing the form so that preferred layout is preserved;
- Optional settings to define initial data loading to a form including automatic loading based on predefined, named search definitions. For example, this allows a user to specify that they always work with a specific set of Vendors or Parts, and to automatically load these on form entry each time;
- Support for inbuilt customization allowing add/change/delete of all types of data controls on the form; and
- Support for inbuilt language translation capability.

All personalization, customization and language parameters are stored on a central server and are deployed via XML documents to each client on request. This ensures that:

- Personalization follows a user from one physical client to another, thus logging into a different PC provides the same user personalization settings;
- Customization is always deployed automatically to all users once approved; and
- User’s language is available regardless of which PC they are logged into.

Please refer to detail sections on the following pages for more information on personalization, customization and deployment.

As a design consideration, the forms are sized to fit onto a screen resolution of 1024x768, or better. Some forms will not fit on an 800x600 resolution without the need to scroll—although the form can be personalized to work at any resolution.
User Interface Standards
All forms comply with a strict set of user interface standards designed to allow rapid and intuitive entry and retrieval of data throughout the application.

Attachments
The Vantage 8.0 architecture supports the ability to manage and view attachments against any record or application process. The attachment management functionality is supported through the tree view within each application UI. A user is able to drag a Windows document or link and drop the item into the UI tree view as an attachment.

Documents can be attached to a specific process (e.g., Sales Order Entry), or they can be attached to a specific record. Documents attached at the process level would typically contain information associated to process execution, workflow, or business practices for the process itself. Attachments at the record level would typically provide information, documentation, or context specific to that record. All attachments can be secured using standard role-based security.

Adaptability
The Vantage 8.0 application is built from the ground up to be adaptable in both the functionality and the user interface according to specific needs. This allows Epicor to support a full range of variations ranging from gross differences in product functionality to minor differences in personal preference through a single framework and code base. This adaptability is key to being able to provide cost effective, yet powerful solutions to our customers.

In terms of the application functionality, the business logic is built to support functional variations according to:

- Product version licensed. Multiple versions of the product may support restricted or advanced levels of the functionality.
- Modules licensed. Alternate or additional functionality is available according to licensing.
- Customization. Variations supported via extensions through the Business Activity Manager.

In terms of the user interface, variations are managed via a sophisticated, centralized deployment of XML documents that contain overrides to many properties of a form. The contents of these documents are able to change the presentation of the application by:

- Disabling or hiding fields and/or entire areas of the form.
- Enabling additional fields and/or adding new areas to the form.
- Changing terminology or language used on a form.
- Changing the flow of entry through a form.
- Changing the layout or visuals on the form.

This adaptability is manifested in five distinct categories of adaptation as described below:

Productization
Productization refers to the ability for multiple product variants. This layer allows Vantage 8.0 to meet the requirements of each type of Epicor manufacturing customer by providing an architected presentation and workflow layer that suites the size and type of customer using the product. For example, Vista for emerging enterprises with a simple interface and Vantage for larger, more complex enterprises.

Verticalization
Verticalization refers to the ability to modify terminology, data fields and flow to support a specific industry vertical. Such verticalization can be packaged and delivered as a variation to the base product.
Customization

Customization refers to the ability for the end customer to modify aspects of the user interface and application functionality. This includes the ability to model the changes, track changes, pilot changes, and deploy and rollout changes to the user community.

Customization is a key requirement for many Epicor customers. The application supports two distinct layers of customization depending on company need. These are:

- A sophisticated, built-in customization engine that allows authorized users to add, change and delete fields on a form and to modify flow and visual appearance of controls on the form. The built-in engine also allows customization of database tables via updates made through application triggers and through code associated with the Business Activity Manager.

- An advanced customization package which is available at an extra cost and supplies the client source and framework UI and Business Logic tools necessary to write your own additions and extensions to the application using the framework.

The built-in customization engine is composed of two sections. One section is resident within each UI application form and the other section resides in the business application layer. The UI customization component acts like an Integrated Development Environment (IDE) and allows the following:

- The addition of UI Form components (e.g., tabs, panels, text boxes, check boxes, grids, labels, buttons, etc.) and the ability to bind the value property to fields in the active dataset.

- The modification of existing UI Form component properties (e.g., visible, enabled, location, size, text).

- The ability to add/remove data columns to/from grids.

- Modification of UI flow by changing the tab order on components.

- The ability to add VBScript “code points” to run before or after data field modifications, before or after data view changes, and at Form Load or Form Exit. The code points will have access to the active dataset within the UI application and to the extended column and row properties. Additionally, the code points will have access to the data within the database.
The advanced developer module allows customers to enhance or develop new forms with exactly the same toolset that the Epicor developers use. The advanced developer module contains:

- Source Code to all Forms
- Visual Studio .NET
- Infragistics Controls (advanced UI control objects)
- Business Logic Toolset
- Progress AppBuilder
- Object Design Tool

The advanced developer module allows users to build their own custom modules and integrate and embed them into the product as though they were part of the base system.

**Personalization**

Personalization is composed of runtime personalization and personalization that is really a subset of customization. Runtime Personalization provides the ability to save the current layout and location of a UI application. Included in that is the position, size, and order of UI forms, floating sub-forms, and the layout of grids.

Personalization that is a subset of customization refers to the ability of an individual end user (or group of users) to personalize a form to their own requirements by using the built-in Customization engine. This includes the ability to change the flow through the form, add columns to grids or arrange columns in grids, change field properties such as color, font, etc.

Personalization is restricted to those changes that do not affect the actual data or minimum mandatory requirements of the application.

**Language Translation**

Language translation refers to the ability of the Vantage 8.0 application to manage and store multiple language variants and to deploy these on demand to the client. Language translation processing is described in more detail below.

Translating the language can be done both from within the application or off-line (via an export/import mechanism). Online translation can be performed in 2 places:

- In the language maintenance/translation screen where it is done mainly on a word or phrase basis (e.g., search for all instances of the word “job” in all of the text). Translation would also be done here on all business object text.

- In the actual application screen where it is on a screen/program basis (i.e., all of the screen’s/program’s text is made available and can be translated “in context”).

Translations have a hierarchy of screen/business object then system-wide. Translations are mainly done at the system-wide level, but it is occasionally necessary to override the generally used translation in a specific instance.

Languages also have a hierarchy. Each language has a parent language and if the particular string is not translated in the user’s language the parent language is checked for translation as well. This is to allow for easier localization at a company, industry, regional dialect level—not all strings need to be translated, just the differing ones.

Real-time translation of the text takes place when the screen is initialized. A dataset of all the text that needs to be translated (this is built up by the UI framework automatically) is sent to the Translation Business Object. The Translation Business Object then translates the text and returns it to the client for incorporation into the form.

**Web Services**

During the initial design of the Vantage 8.0 project architecture, Web Services were a primary consideration and requirement. As a result, Web Service access to all business layer functions is a base capability within the product. Web Services provide the ability to use alternate UIs against the standard business rules of the application. Additionally, Web Services provide a
seamless method for participation in extended enterprise value chains—both internal and external to the organization running the product.

Web Services are provided by a Microsoft Internet Information Server (IIS) with access to the OpenEdge Server. The IIS server can physically reside as part of the organizations Intranet for internal Web Service access. The IIS server can also reside within the DMZ (demilitarized zone) to provide secure Web Service access to the application server while honoring standard firewall security policies.

Base Web Services are automatically generated from the business logic dynamic-linked library (DLL) using the .NET interface specification and a technique known as Reflection. This method of automatically generating the Web Service provides a high quality Internet interface that exposes the application business logic in a consistent and predictable manor.

As with many other components of the architecture, existing Web Services are customizable and new Web Services can be added. This allows Web Services to be tailored for specific needs and requirements while at the same time the business rules are executed to ensure the integrity of the application data.

**Role Oriented Workplace/Dashboard**

Vantage 8.0 is made up of many forms that allow users to run a complex manufacturing business. The workplace allows users to define their own virtual system by giving them the ability to generate a form set that presents information in a way that they want to see it. This information may be cross disciple, cross technology and cross system.

For example, a user may want to see their outstanding tasks, customers on hold, a view of the customer’s supplier portal, and the customer’s current bookings shown graphically from the business intelligence system. Sometimes this functionality has been described as a Dashboard.

The workplace allows a user to define a single form set that contains all of this information and dynamically puts into context those areas that the user is interested in. For example, highlighting a part would then show only those PO’s, stocks, commitments, etc. that were related to that part.

If multiple forms are open, the workplace allows the forms to publish and subscribe information to each other. If the PO entry form was open and a workplace had a PO highlighted, the PO entry form would immediately open the PO ready for maintenance. Highlighting another PO would change the PO in PO Entry to the highlighted PO.

The workplace technology replaces the tracker, workbench and Dashboard functionality that was available in previous versions of Vantage with a completely flexible environment that is defined to meet the exact unique needs of each user. Vantage 8.0 comes out-of-the-box with predefined views for all areas of the system.
Miscellaneous Elements

Workflow Management

In the current manufacturing economy it is imperative that mid-sized manufacturing companies are reactive and flexible so they can meet the needs of a dynamic marketplace. For companies to achieve this, they must be in total control of their business processes so that key business decisions don’t get forgotten in the mass of transactions and actions that are made every day. It is probably true that for every routine transaction that a company makes there is at least one ad hoc one. Therefore, there is great potential for an organization to greatly improve its efficiency and information flow by using automated systems to manage workflow.

The Vantage 8.0 solution is designed to meet the needs of discrete mix-mode manufacturing companies. The product is delivered out-of-the-box with built in workflow processes that enable an enterprise to manage the entire order cycle: from Marketing and Sales, to Production and Planning, Sourcing and Procurement, Installation and Service through to Financial recognition. Vantage 8.0 enables the whole process to be real-time and paperless. All users of the system, whether internal such as a planner or external such as a supplier, are presented with a workbench view that summarizes those tasks that need to be carried out. The entire solution is designed to streamline the order cycle so that an enterprise can maximize its resources, minimize its costs and improve profitability.

So although the solution has been designed for the discrete mix-mode manufacturing market, many of those companies still have unique processes and business rules. Therefore, a toolset is required that will allow a company to define and manage its unique business processes in a way that streamlines and automates work but also allows individuals the bandwidth to use their own initiative to solve complex problems.

Two major components of workflow have already been discussed:

- Business Activity Manager: Build your own business rules and alerts into the product.
- Workplace: Build an environment that allows a user to easily carry out their day-to-day tasks.

The final element of workflow management is task management. A lot of times a company’s workflow is a lot more than just executing predefined application screen functions. Typical business processes require the ability to link both computer and non-computer events in a logical and conditional sequence; to assign responsibility for each stage of a process execution; to schedule and manage the event queue; and to provide control and feedback. For example, the processing of a sales opportunity may require substantially more than the entry of a quote on the system. It may require the outline quote to be entered, a sales visit to establish the requirements, a sales engineer to then clarify the requirements, detailed engineering of the potential product, supplier involvement for quoting key materials and services, credit approval from a leasing agency, etc.

Each event may be a key milestone with many other subsequent tasks that need to be carried out. And although a clear process may be known at the beginning circumstances may cause the sales opportunity to be routed via a very different path during its lifecycle.

Task management functionality allows users to define workflow processes that key transactions need to go through in order to be completed. Tasks may be a single step process or may be a series of interlinked processes involving many different parties. At all times users have a dynamic task list that real time informs them of what they are needed to do. Task sets can be dynamically assigned based on circumstances. For example, different types of engineering change may require different levels of authorization and work. The task management system will automatically assign and route the work needed to process the change. Task sets are tightly embedded into the system ensuring that transactions can only be updated if the task set is at the right status. For example, a quote cannot be sent out to the prospect unless the sales manager has approved the pricing.
Key functionality within task set management:

- User-definable task types categorize work (e.g., to-do, Urgent Item, Telemarketing Follow Up);
- User-definable tasks identify type of work to be carried out (e.g., pricing approval, revision update, generate quote, etc.);
- Electronic sign offs ensure traceability and adherence to procedures by requiring a password when signing off a key task;
- User-definable task workflow sets define complete workflow processes for different types of jobs, including multi-level tasks, milestones, mandatory tasks, and alternative routes;
- Automatic assignment and routing of work is based on required role codes and the context of the transaction being worked on. For example, if the opportunity requires a sales manager to sign off the pricing, task management will look at the territory the opportunity is in and select the appropriate manager to sign it off;
- Supports companies striving for ISO certification;
- Visual representation of process status;
- Message log of all interaction, e-mails, calls, etc. regarding the task;
- Optional automatic notification of task creation and completion via e-mail;
- Ability to manually create a task for someone to carry out;
- Ability to manually route a task;
- Ability to manually alter task set against a job based on circumstances; and
- Provide to-do lists specific to each employee, automatically driving a user to the area of the system to be worked on.

All transactions can be associated with a task in Vantage 8.0. For example, suggestion to raise a job, change a job, raise a PO, print an invoice, etc. can all be associated with a task. The workflow system will evaluate, based on roles, who should complete a task and automatically assign the task to that person's queue.

The solution takes into account the real-life circumstances that define any manufacturing environment, both during planning and the actual process. The embedded workflow solution provides a comprehensive and flexible workflow solution that gives users the tools needed to respond quickly to critical business requirements.

Security

There are many layers of security built into the product. Such security is critical in ensuring the integrity of the system in terms of what can be updated and by whom, and the privacy of the system in terms of what may be viewed and by whom. The key levels of security are as follows:

Product Security

Product security includes protection to ensure that the application only allows use of modules and product variations purchased and licensed.

Application Security

Application security ensures that the business logic protects the database from corruption by always ensuring that an update is valid, regardless of the source of the transaction. This is necessary in a service-based architecture since the business logic can be called from many environments including:

- The rich (primary) client
- External Web Services
- Browser-based clients
Other smart devices (PDA’s, bar code devices, etc.)

**Access Security**
Access security verifies that whomever (or whatever) is attempting to access the application server is permitted to do so. This includes:
- Login security to the menu system either by entry of user ID and password, or via Windows user authentication.
- Session security (same as login security) for application components that are run directly from the desktop or other non-menu areas.
- Services security through the SOA to ensure that an external system may access the business logic.

**Business Security**
Business security includes ensuring that individual users and groups of users have access to the business functions and data that they might be attempting to view or update.

**Reporting/Forms**
The primary paper reporting mechanism will utilize Crystal Reports®. This will have special methods for each report to assemble and de-normalize the data in the database into a dataset. This dataset can then be sent to the client for “preview” and/or printing or, if it was scheduled, to the Print Server.

All reports will have the option to either run now or schedule them to be run in the future. Scheduled reports will also have the option to automatically repeat periodically (hourly, daily, weekly, and monthly). Date ranges can also be specified in relation to the actual run date so that ranged reports like “This Week’s Shipments” will be possible to schedule.

**Distributed Model**
Although global communications have improved drastically of the last 10 years, there are many instances where enterprises want to distribute their information around multiple servers in different geographic locations. There may be many reasons for this:
- Low bandwidth networks: In many parts of the world communications are still poor.
- Redundancy: Insuring that all of a company’s information is not on one server.
- Reliability: Many companies truly rely on their enterprise system. If the system goes down, they stop shipping products and cannot take orders. These types of enterprises cannot afford to have their entire systems go down because of power outages, communication problems, etc.
- Mobile Computing: Although Web access allows many more employees to access the system with a PC and an Internet connection there are many occasions that an Internet connection is not available (in flight, customer’s sites, developing countries, etc.).

To help these enterprises, the framework will allow the application to be distributed across multiple servers with two main flavors:
- mobileConnect
- Distributed Multi Company

**mobileConnect**
In the mobileConnect flavor, one system is a subset of another system. For example, with a sales person’s PC. The sales person wants to run a subset of CRM. Anything that the sales person does is sent back to the main server, and anything done on the main server is sent to the sales person.

The application is physically installed at two or more sites (i.e., the main system and a sales person’s laptop). The remote site(s) are initialized from the main site and a direct, initial transfer of configuration information and data is done.
When records are added, updated, or deleted they are written to a queue to be processed for transmission. A separate process picks up the record, examines it, and determines where it has to be sent. For example, in mobileConnect sales information is only sent out to other sites that have access to the customer’s territory.

Most information is sent only one way—from the server to the remote site. This information tends to be the “background” information that is either supporting the main functions (i.e., master files) or transactions that occur only at the main site (i.e., shipping). One way information can be used and viewed, but not modified at the remote site. When two-way information (i.e., sales opportunities) is added, changed, or deleted a process similar to what happens in the other direction occurs and the records are sent back up to the main system.

To avoid data conflicts, blocks of keys are given to the remote sites by the main site so that each site will not create conflicting information. Also, all inbound and outbound information is checked (at the remote site) to make sure simultaneous changes haven’t occurred. If simultaneous changes are detected then, if possible, the changes are “merged” together automatically. If it is not possible to merge the changes then the remote user is notified that the changes they entered will be lost, shown what the changes were (and what the main system also changed), and be allowed to decide what steps need to be taken to resolve the conflict. The system currently uses a “Server Change Wins” model for security and practicality reasons, usually meaning that the user either ignores/cancels their change or re-enters it, if possible.

SonicMQ® is used to perform the actual message transmission. Separate “topics” are set up within Sonic for each territory (again using mobileConnect sales as an example). Each site subscribes to the topics that are relevant to it. Other topics are created for commonly needed information (such as globally needed base master files, etc.) and all sites subscribe to these topics.

**Distributed Connect**

In the distributed connect flavor, separate systems share common information. For example, ensuring that all systems around the world are using the same part number definition. This uses a lot of the same techniques and architecture of the mobileConnect model, with a few noted differences.

“Global” parts, customers, etc. are used to eliminate simultaneous updates and ID conflicts by clearly defining the one owner of the master information. Local copies of the master can have varying levels of differences (locally maintained), but still be tied back to the master information.

Information often “transforms” when moving between companies. For example, Purchase Orders on one system become Sales Orders on another.

Special note is made of buying and selling transactions between companies within the same organization so that proper revenue back out and consolidation can occur.

The distributed connect engine can be used as an alternative to the SOA for integrating to other systems a company may own. Distributed connect is currently used by Epicor customers to integrate to third party financial systems. This may be because they have sites in regions of the world that have unique localization requirements that can only be solved by a local financial package. Or it may be because a company’s parent company uses a separate financial system that they want all the companies in the group to use. Whether it be this or the integration to a CAD, PDM system or supply chain collaboration, distributed connect helps companies share information.
Conclusion

Vantage 8.0 is the next generation of Epicor's manufacturing ERP solution. It leverages existing rich functionality moved to the .NET platform. Vantage 8.0 embraces Microsoft .NET in conjunction with Progress OpenEdge to build an object-oriented client and Web Services-based application. It is a complete ERP solution created for small and medium enterprises and small- to medium-sized divisions of larger organizations.

Vantage 8.0 brings the power of a Service-oriented Architecture to enterprise resource planning.

In the introduction, we explained that an architecture needed to be practical. To us, practical means that we meet our target customers’ key requirements. Vantage 8.0, built upon Microsoft .NET and OpenEdge, meets these key architectural requirements.

Vantage 8.0: The Next Generation of Manufacturing Software

- Built for optimization
- Built for collaboration
- Built for customization
- Built for mobility
- Built for flexibility
- Built for accessibility
- Built for reliability
- Built for scalability
- Built for security
- Built for real-time
- Built for growth
- Built for global business
- Built for low cost of ownership
- Built for rich user experience
- Built for functionality
- Built for Web Services
- Built for .NET
- Built for the shop
- Built for the plant
- Built for the warehouse
- Built for the enterprise

Vantage 8.0 is built for manufacturers.
About Epicor

For 20 years, Epicor has been a recognized leader dedicated to providing leading edge enterprise software solutions to midmarket companies around the world. With over 15,000 customers, Epicor delivers end-to-end, industry-specific solutions that enable companies to immediately improve business operations and build competitive advantage in today’s real-time global economy. Epicor’s comprehensive suite of integrated software solutions for Customer Relationship Management, Financials, Manufacturing, Supply Chain Management, and Services Execution and Control provide the scalability and flexibility to support long-term growth. Epicor’s solutions are complemented by a full range of services, providing a single point of accountability to promote rapid return on investment and low total cost of ownership.

Trademark and Copyright Acknowledgement

Copyright © Epicor Software Corporation 2004. Epicor, Epicor Vista, Epicor Vantage, Epicor Avanté and Epicor Manage 2000 are trademarks and/or registered trademarks of Epicor Software Corporation. All other trademarks acknowledged. Epicor reserves the right to make modifications or changes to the functionality, and plans described herein without further notice. This document is intended solely to inform the audience of Epicor’s current intentions. Epicor makes no warranties, express or implied in or by this document. The contents of this document are believed to be current and accurate as of its date of publication. For a complete description of the product features, please refer to the product’s user guides, reference manuals and release notes.
For more information, contact Epicor at (800) 449-6772 or by e-mail at vantage.info@epicor.com.