THOMSON REUTERS
ELITE™

PRODUCT BRIEF

3E BUSINESS OPTIMIZATION SUITE
3E PLATFORM OVERVIEW

THOMSON REUTERS™

THE FOUNDATION

The 3E Business Optimization Suite™ is built on a powerful technology platform. This platform is designed to meet the following objectives:

- The system should be configurable and extensible to meet customers' unique requirements.
- Every transaction should be workflow-enabled.
- Users should be able to collaborate within the system.
- It must be scalable: capable of supporting large numbers of simultaneous users across diverse network topologies.
- It must be consistent: all parts of the application must behave identically.
- It must be secure.
- It must integrate into a firm's overall IT environment.

To lay the foundation for how 3E® achieves these objectives, we need to understand how 3E differs from earlier technology approaches. Let's begin by looking at the inherent complexity in a modern Web-based application. Consider some of the layers in a typical Java® or Microsoft® .NET component-based Web application[1]:

- Components must be built that expose interfaces for remote and local methods, identity, and error handling.
- Classes must be created and maintained to implement interfaces, handle exceptions, manage network connections, and manipulate metadata.
- System services must be utilized to register component instances, manage their state, and handle security.
- Web server pages and client pages must be generated using forms, scripts, and servlets.
- Data objects must be created. They must handle queues, database connectivity, and object-relational mapping.
- Web services need descriptions, resource locators, protocols, and schemas.
- Connections to legacy and third party systems need special connectors and API's.
- Code is held in multiple artifacts: HTML, development language source code, XML, SQL, configuration, and resource files.
- To deploy applications, executables must be produced and packaged; compilers and interface generators must be coordinated; applications need to be replicated, installed, and registered on target systems that include database servers, application servers, Web servers, and messaging systems.

While this architecture provides significant flexibility, the complexity means that software developers spend much of their time managing the infrastructure. To at least gain productivity on the business logic portions of applications, development teams tried to develop frameworks – collections of code objects, but significant progress proved elusive. To this point, there has been very little object re-use among most development teams; it is simply more expedient to copy and modify snippets of existing application code.

Both software providers and their users suffer from this approach to software engineering. Software applications take an inordinate amount of time, money, and risk when developed initially. But the problem doesn't end there. Software is like a live organism; it must grow and change to survive. Applications are a reflection of the underlying business environment and that environment is constantly evolving. Yet the current generation of applications is ill-suited for change. Some of the leading reasons for this include:

- Stagnation – The cost and risk of big changes are so large that only small incremental improvements are attempted.
- Fatigue – Expediency causes changes to be attempted that were never anticipated in the original design (known as the "adding stories to a dog house" problem).
- Brittleness –Where the software didn't initially anticipate change, it breaks easily. More change causes even greater brittleness.
- Redundancy – Objects and components are copied and then changed with each development iteration. It therefore becomes necessary to keep multiple sets of code and database elements synchronized.

The result is applications that can't adapt to changing business and technology requirements. The interval between software releases becomes longer and the level of significant change becomes smaller.

Over the last few years, a new methodology has been proposed to address many of these frustrations: it is known as the Software Factory.

[1] This section is based on *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools* by Jack Greenfield, Keith Short, Steve Cook, Stuart Kent, and John Crupi; Wiley 2004.

Example:

A firm wants to add unique business logic to the application function that calculates time entry rates (called GetRate). The code change to the customer.object would take the following form:

```
Public Overridable Function GetRate As
Public Overrides Function GetRate
    .... (customer code)
    MyBase.GetRate
    ... (more customer code)
End Function
```

The resulting business object extends the functionality of the GetRate base class.

Even business logic changes are dealt with using the inheritance capability. Depending on the nature of the application, the customer version of the object is allowed to override or append to the base class.

Web pages are also objects that are inherited; they differ from business objects only in that changes to Web pages are handled with a graphic forms editor. On the surface it looks like the customer is creating and modifying a new version of the form, but that would lead to all the headaches of customization. Within 3E, the platform is managing Web page inheritance, just as it manages business logic and archetype inheritance.

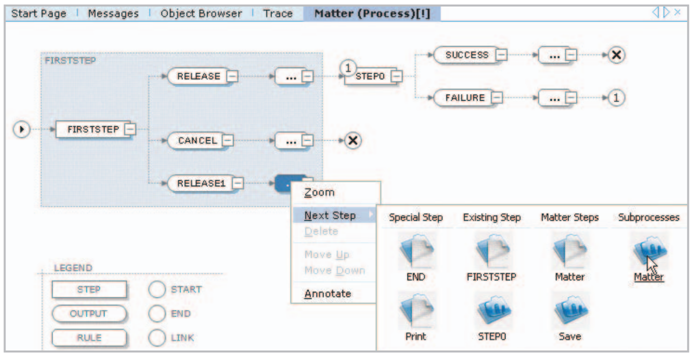
BUILT-IN WORKFLOW AND COLLABORATION

Workflow

A Web page or form in 3E is always part of a process. The 3E platform has a process manager that graphically allows the administrator to control the process flow. For example, the simplest process would display a page, and upon completion would save the page's contents to the database. A more complex process would subject the page to a set of business rules (e.g., is the invoice amount over $10,000?) and route the page to an approval step. Only after the page completes the approval step would its contents be saved to the database.

Every single transaction, every data-capture page, is workflow-enabled without making any changes to the pages or underlying business objects. Workflow is a service provided by the Software Factory. It is inherent to the 3E platform itself. Here, conceptually, is how it works.

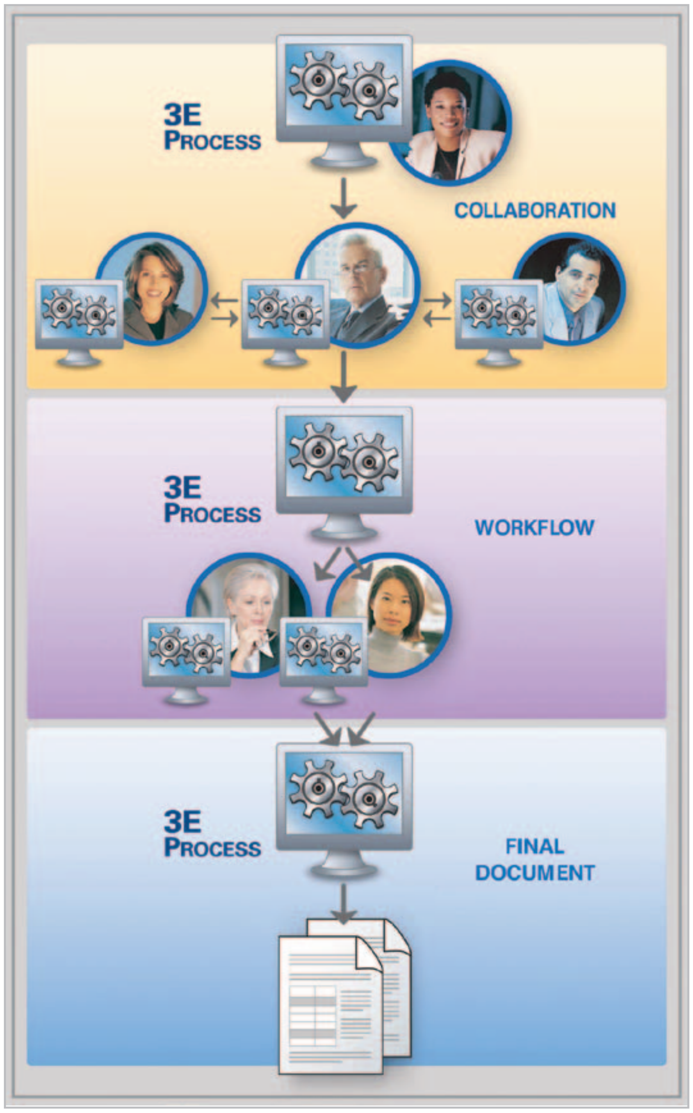
When the user is entering information into a Web page, he is not "talking" to the database; rather, he is building a new XML document. This document is know as a "draft" in 3E terminology and is stored as a separate object in the database. When the user completes the page, the Process Manager checks the XML document against any previously defined business rules. Let's suppose the next step is an approval. The Process Manager would notify the next person (or role) in the process that they need to take an action. The Approver would then view the page (or perhaps a separate approval page) and send it to the next step in its process. Only when the Process Manager determines that the page is ready to be saved will it execute the Save action. The Save will take the information stored in XML format and update the appropriate tables in the database. The original developer of that Web page had no idea what the customer's workflow process would look like. 3E is designed to be configured. It is designed to allow customers to create unique implementations of the application.



Collaboration

Think of collaboration as ad hoc workflow. While workflow is built upon business rules and pre-defined processes, collaboration is a tool that allows users to send pages among each other for comment and update. Consider the following example: a billing lawyer is examining a proforma as part of the bill editing process. She notices time entries from two senior associates that she wasn't aware of. The billing lawyer can send the proforma to these associates for comment and correction by using the collaboration service. When the associates respond, the billing lawyer can accept, reject, or modify their changes and then send the proforma through the remainder of its workflow process.

The collaboration service takes advantage of the XML structure of each page. The billing lawyer is working on a draft. That draft XML document contains an element corresponding to each field on the page. The collaboration capability allows multiple inputs to exist for each field. 3E tracks who entered each version of the field, and displays the different versions to the collaboration originator. An analogy would be the way an author can track changes to Microsoft Office® documents during a collaboration process; 3E has applied that paradigm to transactional applications. Once again, the Software Factory provides the collaboration service. The application developer gets these services "for free."



Collaboration and workflow are both built on the patent-pending draft technology of 3E. By separating the user experience from the event that updates the database, 3E breaks the nexus that forced firms in the past to customize their system or build workflows outside of their applications. The database Save event is controlled by 3E. It understands the customer.object versions of archetypes, business objects, and Web pages. It can be executed whenever the Process Manager determines. Using these fundamental capabilities, 3E allows firms to embed their business processes within the application.

SCALABILITY

3E is designed to allow users to grow their system to support large numbers of simultaneous users without experiencing bottlenecks or performance degradation. Scaling is not just a technology or platform problem; it must be a design goal of the application as well. So, we will first examine aspects of the application design that allow for scalability, followed by design decisions in the 3E technology platform that relate to performance.

Concurrency

Financial applications generally maintain database tables with summary information that is used for online querying and reporting. The more extensive these tables, the faster the inquiries. The trade-off is in concurrency. If multiple users attempt to simultaneously update summary statistics, the result is deadlocks and slow transactions. 3E solves this problem by maintaining a transaction queue—a table that keeps a list of pointers to all new transactions in the database. For example, if the user enters a cash receipt, when the workflow process is completed, 3E will update the transaction tables in the database and write a row to the transaction queue. The 3E Scheduler manages a separate process thread that reads the queue and updates summary tables in "near real time." The online inquiries and reports get the benefit of aggregated data (e.g., client and timekeeper level totals), while the transaction processing engine is unencumbered by locking overhead.

Built-in Business Intelligence

One of the most frequent causes of performance degradation is large numbers of users generating reports (whether to paper or screen). In the past, system designers and administrators have tried to deal with this by building data warehouses, copying information to separate reporting servers, or even restricting the ability to generate reports. None of these solutions is ideal; they all require significant administrative overhead and create questions of "which version of the data were you using?" The 3E Metrics Engine was designed to alleviate this performance problem. The Administrator can schedule reports to be run at pre-defined intervals (usually at off-peak times or end of period); the results of these calculations are stored in special database tables. Report viewers are designed to offer the user the results from these pre-scheduled runs. Only if the user needs absolutely real-time information do they execute a new calculation process (and then typically on a small subset of the data). Offering current (last night or last period-end) information—but not necessarily real-time—information is a powerful tool to eliminate this pervasive cause of periodic performance problems.

3E Architecture

The three-tier design of 3E's technology platform is built for scalability. The first tier is the database server. 3E uses the 64-bit version of database engines, giving the system access to greater memory space for caching and overall performance. In addition, all access to the database is controlled by 3E's Object Query Language (OQL). Any 3E application that interacts with the database must submit its requests through the OQL engine. Among the benefits of OQL is its ability to optimize the performance of database queries and take advantage of platform-specific features. This 3E data access layer also manages connection pooling. Opening and closing connections to the database engine generate considerable overhead. 3E allows the administrator to maintain a pool of connections that are shared by all the users. Connections are released and returned to the pool after every SQL statement is executed.

The second tier in the 3E architecture is the application server. This is where application code is executed and Web pages are generated. Here, too, 3E utilizes 64-bit servers to gain maximum performance. As systems grow, customers can scale by simply adding more low-cost application servers. Unlike prior technologies that scaled only with expensive changes in hardware, 3E allows for incremental scaling—add a simple dual-processor application server to the rack. By adding load-balancers and other administrative tools, it is possible to tune the system to optimize the hardware utilization.

Performance across the network is the final tier. 3E utilizes two fundamental techniques to optimize speed and scalability:

- When a user is in a transaction-oriented page, he communicates with the Web server only by sending XML across the network. The client-side Web browser then renders that XML into HTML and displays the page. The XML that is sent is only a small fraction of the size of the HTML that is ultimately rendered. Client-side rendering can make a tremendous performance difference when operating across wide-area networks that are constrained by bandwidth and latency.
- Whenever possible, 3E takes advantage of caching to improve response time. On the client-side, versioning ensures that XSL files, Javascript, runtime form definitions, and message files are automatically cached until a change forces a refresh. On the application server side, special algorithms load data objects efficiently and cache them on the server wherever possible.

Finally, Thomson Reuters Elite™ has built a library of testing scripts that operate with performance testing tools. By tweaking these scripts it is possible for system designers and administrators to validate system performance in their own environment.

SECURITY

The 3E security model can be conceptually divided into three aspects:

1. Control over what processes a user can access
2. Control over which elements of an object a user can access
3. Control over which portions of the database a user can access

The first two aspects are controlled by 3E's internal security model. A user is assigned to one or more roles. A role is assigned a set of access controls. These controls determine whether a user has access to a process at all. If they do, it then controls whether the user can view or update those attributes for each attribute of the underlying objects. For example, a person might be assigned to a role called "Associate." The Associate role could have access to the Client process and its underlying Client object. They may, however, only be granted permission to view the information, but not update it. Even then, they might be restricted from viewing special rate or origination data.

It is possible for a user to have multiple roles. For example, a lawyer might have the role "Partner" and the role "Practice Group Leader." The 3E security model will always grant the most permissive access right between the two roles. For example, the base Partner role might only allow lawyers to see their own summary information. By adding on the additional Group Leader role, the lawyer might now have permission to view summary data at the practice group level.

3E allows security to be managed at the database row level. A user's access rights can be restricted so that they can only view or add certain data. Typical users might be restricted to viewing only statistics for their own office or from viewing information that is behind ethical walls. 3E handles this by utilizing new features in SQL Server 2005: the ability to build special schemas and views for each user. Whenever a user executes a database transaction, he is connected to the database with his user identification; and his view of the database is enforced. The database responds to his queries strictly with the data he is allowed to see. Although these views are administered within 3E, they are enforced by the database engine itself.

One advantage of this architecture is that security cannot be bypassed with third party tools. For example, if a user would connect to the database with Microsoft Excel® and use its data query tools to pull information into a spreadsheet, the SQL Server engine would still enforce the user's restricted view of the database.

CONSISTENCY

Today you can go to almost any commercial Web site on the Internet and use it without having been specifically trained. Users of internal applications have the same expectation. From the firm's standpoint as well, extensive training is costly and often impossible to conduct. Applications that operate in idiosyncratic ways result in pools of expertise that prevent flexible cross-training of staff. Billing operators only know how the billing application work; accounts payable operators only know their application. 3E's Software Factory technology eliminates this problem. The developer is focused on business logic and functionality. The 3E platform generates the user interface. The basic screen forms, control over collaboration, workflow, queries, and lookups are all determined by 3E. If a user is familiar with how to inquire on a client, they will be able to inquire on a vendor. If they can enter a general ledger journal entry, they will understand how to use any other transactional screen in the system.

Our vision is that users will fall into two classes: first, casual users who access the system primarily to view information or make simple edits; second, power users who need to understand the full functional capabilities of their domain. By implementing consistency and simplicity, the first group should be able to operate with just a brief introduction to the 3E conventions. The second group should now be able to focus their energy on domain expertise rather than on the operation of the application. For example, training for an accounts payable entry operator might focus on correct handling of 1099 or VAT rather than on which button on the screen does what.

INTEGRATION

No application is an island. 3E, like every good software citizen, needs to operate with its colleagues and share information. With prior technologies this was a painful process. Each supplier offered a set of API's (unique programmatic interfaces), and it was the customer's problem to figure out how to connect all the pieces. Those interfaces may not stay consistent across releases and offered no standardized methods for handling authentication, error handling, and administration.

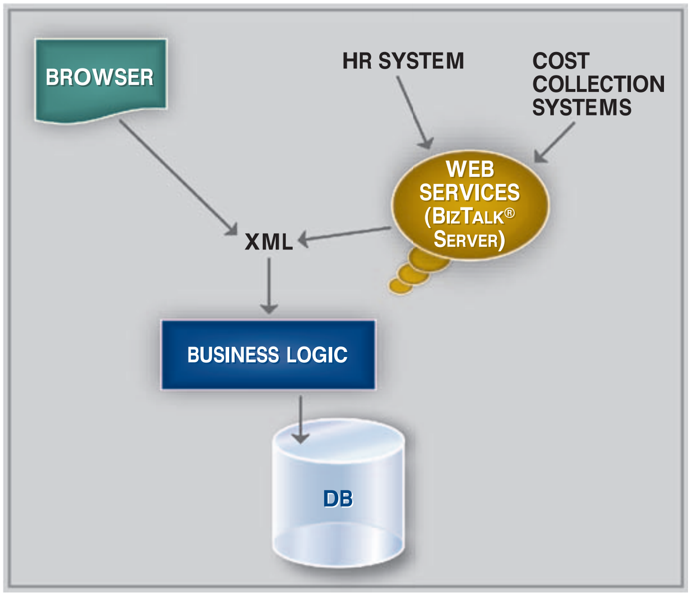
The whole software industry is moving to address these problems through the use of Web services. Web services are being standardized so that applications can readily interact and production environments can administer the flow and execution of these transactions. 3E's services-oriented architecture (SOA) is specifically designed to support this integration environment.

The simplest way to understand this is to go back to our explanation of workflow and collaboration. Each user is operating on a Web page that creates an underlying XML document (the draft). But 3E is equally happy to accept that XML stream if it is generated by a Web service instead of a direct user interaction. Once the XML is submitted to 3E, it is subject to the same security and process management rules that were set up for individual users. For example, one firm might enter new timekeepers using the Timekeeper entry screen. Another firm might choose to enter timekeepers in their PeopleSoft HR application and use a Web service to update the information in 3E. The timekeeper application—the underlying objects—is agnostic; it doesn't care how the XML information is delivered.

3E uses Microsoft BizTalk® Server as its tool for orchestrating the flow of Web services. BizTalk Server is a powerful industryaccepted tool for managing a services environment. 3E is a team player that should enable integration and reduce its cost and complexity.

CONCLUSION

Thomson Reuters Elite has invested in the 3E platform and its underlying .NET technology. This paper has only scratched the surface of explaining the full power and flexibility of this environment. We believe that our customers will join us in reaping the benefits of a platform that truly lives up to its name: 3E will be embedded in your business and it will enable change and improvement in your firm's operations everywhere you have an office.



YOUR PARTNER FOR SUCCESS

Thomson Reuters Elite offers an end-to-end enterprise business management solution that allows law firms and professional services organizations to run all operational aspects of their firms, including business development, risk management, client and matter management, and financial management. As an industry leader for organizations across the globe, we understand the business and financial aspects of firm operations, and we have the tools to streamline processes, improve efficiencies, and provide the flexibility you need to change and grow your business.

To learn more about the Business Optimization Suite or for a global list of office locations, visit **elite.com.**

The trademarks used herein are trademarks of their respective owners.
© 2012 Thomson Reuters L322947/11-12

THOMSON REUTERS™