COMPUTING INTEGRITY
INCORPORATED
60 Belvedere Avenue
Point Richmond, CA 94801-4023
510.233.5400 Sales
510-233.5444 Support
510.233.5446 Facsimile

PROGRESS SOFTWARE BUSINESS ALLIANCE
ISV Partner

ESB As An Application Architecture

By Thomas Mercer-Hursh, Ph.D.

29 May 2006

In recent years, there has been a dramatic rise in the popularity of the concept of an Enterprise Service Bus (ESB) over a Service-Oriented Architecture (SOA). This popularity has been especially notable in the context of Enterprise Application Integration (EAI) and related domains where the requirement is for multiple applications to interact, often from multiple companies and distributed across multiple computers. Sonic Software has been one of the most notable evangelists of this concept and arguably provides the best overall infrastructure for implementing this approach.

EAI is a domain in which costs can be extremely high, project size is often extensive, and simple plug and play solutions are sparse and limited. Historically, EAI projects have involved very expensive custom programming to integrate each application with little help from tools. As the number of applications increases, the complexity of the project can become almost astronomical. Products providing pre-developed connections for popular packages have been of limited help since a typical EAI project is likely to involve only one or two such popular packages and the remainder of what can be hundreds of applications still need custom development. Moreover, those tools are typically of a hub and spoke design that can have performance issues with high traffic levels.

EAI projects in earlier years often arose when a large corporation with a large number of separate applications found a need to integrate and interconnect those applications, e.g., as might be driven by the desire to create an integrated customer web portal. While this type of EAI project continues, in recent years, it is also common for the need for integration to be between a company's own applications and the applications of its vendors and customers, i.e., supply chain integration. The ESB/SOA strategy is, if anything, even more suited to this type of diverse and rapidly changing integration problem than it is to the classic EAI project, particularly since the integration is often structured by emerging standards such as web services.

The availability of a strong software offering like Sonic ESB, which facilitates implementing this integration strategy, can significantly reduce the expense of implementing an EAI project and can even more dramatically reduce the cost of maintenance and future changes as requirements continue to evolve. It is no surprise, therefore, that EAI products, including Sonic, tend to have moderately substantial entry price points since their perceived and real value is high. It is the purpose of this paper to discuss the desirability of a more localized application of the ESB/SOA approach and thus to suggest the need for an alternate licensing strategy when these tools are used as the architectural backbone of a tightly integrated application.

Historical Background

In the early to mid-90s I made a substantial effort at recruiting officers and locating investment capital. I was looking to make a major enhancement of Computing Integrity's suite of applications called Integrity/Solutions (I/S). I/S is a sophisticated, integrated suite of applications for automating central business functions of distribution companies. It has competed successfully head-to-head with Oracle Financials, PeopleSoft, and SAP. The primary goals for the enhancement were creating a

browser client to replace the ChUI interface and to facilitate distributed deployment of the applications. To distinguish us from "yet another company selling distribution software" (YACSDS), the "sizzle" in our funding proposal was our Specification-Driven Development (SDD) tool that allowed us extremely rapid generation of very high quality P4GL.

At one point, I came very close to bringing in a very attractive VP Sales/CEO, but eventually he decided that the prospect was not sufficiently compelling. During the regrouping that followed, I came to decide that it appeared that it would be many years before the Progress product evolved to provide me with the development platform I wanted for distributed deployment, so I began looking at alternatives. This brought me to look at Forté.

In the early to mid 1990's, prior to the coining of the SOA and ESB terms as such, Forté Software was a strong proponent of a focus on "services" as a primary structuring element in software architecture design. At the time, this did not strike me as a particularly radical approach in the context of a single, integrated application written in TOOL, the Forté OO4GL, but rather just as the fulfillment of a long term move toward modularization and encapsulation. One could, of course, say the same about services in an SOA, but in the end there is more significance to this focus on services and this was largely present in the Forté recommended architecture. One of the really compelling strengths in the Forté development environment was their ability to rapidly configure the deployment of these services across a distributed network for optimum performance and capabilities.

In the mid-1990s, Forté also introduced a product called Fusion aimed at the EAI market. Fusion integrated to applications using "adapters", which utilized XSLT transforms to convert the native API of the application to and from standardized XML. The XML was then transmitted across a reliable messaging bus called the Fusion Bus to other applications. A workflow engine was responsible for directing traffic so that a sender did not need to know which recipient would receive a message or which sender had generated a message. The workflow engine also provided sophisticated logic for recognizing offline services, alternate routing, restarting of services, time-dependent routing, etc.[1] The parallels with a modern ESB system and an orchestration engine are obvious.

In exploring this technology, I developed the idea that I could implement I/S as series of encapsulated applications that interfaced through the Fusion Bus. This would provide me with a number of significant advantages, including:

1. A workflow engine to control the interaction between applications;
2. The ability to run a mixture of P4GL and TOOL applications during the development process, thus allowing us to continue to have a product to deliver while the new product was being developed (a classic SOA benefit);
3. The ability to substitute an external package for one of our applications, e.g., selling to a division of a larger corporation which mandated the use of SAP for the GL;
4. A very direct way to implement distributed deployment, including multiple instances of some components, without any special development;
5. A context in which it would be very straightforward to integrate with web applications; and

[1] A classic case of time-dependent routing is placing a service request in the queue of a credit representative and then moving it to the queue of the credit manager if not handled within a particular time span. One of the more interesting and complex examples of workflow routing was seen in a scheduling system for TransCanada Pipeline where there were many independent computers involved in scheduling traffic through the pipeline. Many of these computers were old and unreliable, making manual access for scheduling tedious and lengthy. A Fusion application automated the scheduling process and included logic that could even restart a non-responsive server and summon tech support if a server could not be reached in an appropriate time.

6. An environment in which it would be easy to add specialized "mini-applications" to supplement the core offering.

While I found this a very compelling architecture, I had one difficulty, which was that the entry price for Fusion was on the order of $75,000. In that my target market went as low as companies with $5M in revenue, this seemed to present an intolerably large base price, one that would effectively block us from a significant part of the market CI wished to target.

Fortunately, Paul Butterworth, VP Technology of Forté, was a very insightful person who quickly understood why this was a compelling architecture for a tightly integrated suite of applications. Through his offices I was able to extend my VAR agreement for the other Forté products, which was based on a percentage of the sale price of the application, to include Fusion, thus allowing me to sell to either large or small customers.

Unfortunately, I was never able to raise the investment capital[2] to convert I/S to this technology and, following Forté's acquisition by Sun, the technology has now faded from the landscape. Fortunately, Progress and Sonic are now reaching the point where a similar architecture can be contemplated in ABL, although some further enhancements such as multi-threaded sessions would make this far easier and more direct.

For the last few years, I have been simply supporting the existing customer base, continuing to evolve the product according to their needs, but making no major architectural upgrades. Recently, the two largest customers ceased to use the software, one because of an ill-conceived move to Oracle Financials and the other because it was purchased by a much larger company with existing software.

Where We Are Today

Therefore, I have refocused my business on the problem of transforming legacy applications into fully modern, OERA[3] compliant, ESB/SOA enabled architectures. This will involve:

1. Analysis tools to extract information from existing ABL applications;
2. Constructing UML models from this information that completely describe the application;
3. Revision of the UML to better conform to modern architecture; and
4. Generation of the complete new application through the techniques of Model-Driven Architecture (MDA).

As a part of this effort, I will have to create an underlying framework for use by the generated applications and I will have to create the template structures to use in generation.

While these tools may be applied to I/S, it is not anticipated at this time to return to active sales of I/S, but rather to focus on the transformation of legacy applications for existing Progress customers and possibly VARs. I feel that there are a large number of Progress sites with either purchased or homegrown applications of significant age who are feeling pressures for web access, supply chain integration, and the like, but whose underlying architecture is very poorly suited to these tasks. I believe that full development of these tools will allow transformation of the entire application, thus putting these companies in an excellent position for the future, but at a cost comparable to patchwork addition of limited integration capabilities.

[2] Ironically, deciding to move to Forté as a technology platform meant that I no longer had the "sizzle" of SDD by which to distinguish CI from other YACSDS. Our second effort at recruiting financing and officers never got past this perception problem.
[3] OpenEdge Reference Architeture.

The packages in use at sites needing modernization will typically be suites of integrated applications, like I/S, possibly with additional non-integrated packages, some of which may not be in ABL. This is a prescription for ESB/SOA, even if all applications are currently integrated because:

1. Externalizing workflow will make the company far more nimble in responding to changed business circumstances;
2. Encapsulation of applications as services in an SOA will allow on-going evolution of software components without disruption;
3. New major and "mini" applications can be readily integrated via the ESB;
4. Integration to web services will be extremely straightforward;
5. Supply chain integration will be greatly facilitated; and
6. If any non-ABL applications are in use, they can be integrated via the ESB and evolved as appropriate over time.

In short, every benefit of ESB/SOA that applies in an EAI context, also applies to an integrated collection of applications, even if they are entirely within the company, tightly integrated at the onset, and all written in the same language. One may not initially use all of the capabilities of an ESB, e.g., a tightly integrated suite will not need XML transform functions, but these capabilities are immediately available should they become necessary, e.g., when initiating supply chain integration.

Thus, it is my position, just as was argued in the context of the Forté tool set, that any time we have multiple interacting applications, it is appropriate and beneficial to use ESB/SOA as the underlying architecture. I believe this is where OERA is pointing us. Interestingly, one of the big advantages of Forté prior independent of Fusion was their highly flexible ability to configure distributed deployment of components. Fusion extended this to other applications and added the workflow engine. A modern ESB/SOA tool such as Sonic achieves a similar kind of distributed deployment flexibility combined with the workflow engine, but does so from "the other end" as it were. Thus, for me, Forté gave me distributed deployment in the base tool set and Fusion added workflow and the potential for interconnection with other applications, but in the present, the distributed deployment, workflow, and potential interconnection all come from the EAI toolset.

What Is Required To Implement This Architecture.

The OO features in 10.1A clearly have been a major leap forward in our ability to implement such architectures, particularly with respect to our ability to utilize proven design patterns from other OO languages. While there are certainly a variety of additional language features which could further facilitate such development, there are two significant problem areas that I see in integrating components into an overall application.

The first of these problem areas is fine-grained integration to create services where the obstacle is the single-threaded session. Here the requirement is for either a multi-threaded session or, at least, for a highly efficient fine-grained interprocess communication mechanism[4] that would allow multiple sessions to behave as if they were part of a single service. See http://www.cintegrity.com/PDF/Multi-threadingUseCase.pdf for a discussion on the need for multi-threaded sessions.

The second of these problem areas is coarse-grained integration as is associated with ESB/SOA. From a technical perspective, this capability is clearly available, and available in exemplary form from Sonic, but the pricing of those products is geared toward the EAI market where per server

[4] As an alternative to multi-threaded sessions, PSC could consider providing a mechanism for extremely rapid interprocess communiction. This would not be as desirable as true multi-threading, but would, at least, provide a better foundation for closely cooperating separate lines of execution such are needed for fine-grained connection of components.

prices of $10,000 or $50,000 can be easily justified in terms of the savings in development. In the context of the use of these technologies for a single integrated suite of applications on a single server, there is certainly still substantial value, but the additional infrastructure cost may well make the total cost excessive, especially if there are no EAI-type goals in the immediate development. Moreover, given the heavy use of tools in the proposed approach, the ESB products will not provide the same kind of development savings that would be experienced in a typical EAI project since the one time investment in MDA will be applied at many sites.

There seems to be two methods by which this problem of entry level price could be solved:

1. Progress Software could elect to publish source code for the Sonic adapter or, at least, a generic JMS version of the adaptor so that developers with a need for ESB/SOA integration could utilize open source alternatives to Sonic. This still puts a lot of burden of development on the individual developer and results in the use of what is almost certainly an inferior tool.
2. Sonic Software could offer a licensing solution which was tailored to this one application, one machine environment, but which also provided a graceful extension as new requirements were developed. This licensing solution needs to cover such added value products as the Orchestration Server in order to provide full value.

Of these options, the second is the far superior option since it provides the robust quality of Sonic software so that one may obtain the highest quality ABL solution. One should observe that the options are not mutually exclusive and there is some argument in favor of implementing the first solution, regardless of whether the second is implemented. The current IT environment is not one in which vendor lock-in is regarded favorably, especially when it is via a mechanism such as failing to provide source code for a key interface.

The obvious difficulty is in arriving at a licensing model that provides an equivalent or greater revenue stream to Progress while enabling small sites to use the products affordably. I think it is important to recognize:

1. The primary difficulty is with respect to an initial licensing solution. Once a company has made a transition to a single application, single computer ESB/SOA implementation, it is highly likely that opportunities will be developed for distributed deployment, supply chain integration, and the like. As these will be new, some additional license cost will be justified and acceptable, especially since the modernized architecture will facilitate the implementation. Thus, there is a high probability of downstream new license revenue. Of course, this does require a structure in which there is no sudden painful jump in license cost in relation to one of these transitions.
2. For customers who are not using CI's whole application transformation service or not doing their own whole product transformations, the attraction of a low cost entry license would be that it would make a pilot project economically justified. If there is a graceful growth path, this is another context in which one can expect additional license revenue as additional projects are undertaken.
3. Part of the licensing problem here is one that is common to a number of "enterprise" oriented products, i.e., if one has clearly established a need, determined that a product is the correct solution, and decided that one is committing substantial resources to that product, then a cost of $10,000 or even $100,000 can be justified, but if one is only intending a pilot project and does not know how extensively the product will be used, then these large expenditures are difficult to justify unless the context is one in which one can get payback on the first project. This contrasts with licensing models in which there is a more step-wise pricing which can scale according to the level of usage.

What Needs To Be Included?

There are a number of ways to limit a Sonic offering for use on a single CPU for a single application suite in order to avoid giving away what one would be justified in charging for in an EAI environment. Let us tentatively refer to the desired product as the OneCA Edition for 1 CPU and 1 Application, although application would need to be defined in terms which could cover a tightly integrated application suite (see below for additional options).

The first and most obvious area of limitation is in which products are included or excluded. Products that could clearly be left out of any entry level offering would be:

1. Any Continuous Availability version, although this should be available for a reasonable increment;
2. Sonic XML Server;
3. Sonic Collaboration Server; and
4. Sonic Database Service[5].

Note that while these products could be omitted from the OneCA Edition, it would be a more attractive licensing model overall if there were no painful steps. See discussion below about entry level versions of other products.

One of the more problematic components is the Orchestration Server. While a small company implementing the OneCA Edition on their own in a piecemeal fashion might not immediately think of implementing the Orchestration Server, a company who used our forthcoming transformation service would be strongly oriented toward externalizing workflow into a business process management tool. Not only I would tell them that they should do this, but I would be wanting to create the MDA model so that this happened. At a $25,000 entry price point, I would probably be forced to offer an open source BPEL engine instead. However, it is also almost certainly the case that the complexity of logic managed by an installation in one of these customers would be very modest compared to the logic managed by the typical existing Sonic Orchestration Server installation. Were it possible to have one or more "junior" or "standard" edition versions as well as the existing "Enterprise" edition like with some of the other products, it would be easier to get companies started on this path.

Other than excluding products and providing more limited editions of products, the other obvious way in which to provide lower entry pricing is to limit products according to some metric such as users, message volume, connections, and the like. This is, of course, far from easy to do in a way that is widely viewed as fair, as is evidenced by the unhappiness about licensing sites that involve web access.

Entry Level Versions of Other Products

To create a licensing context in which there can be a graceful transition from a readily affordable entry point to full use of the product range, one also needs entry level offerings of products not included in the core entry package. For example, a company with no supply chain partners who implemented the OneCA Edition and then added a supply chain partner would find an additional price of $35,000 for Collaboration Server to be rather shocking. Of course, with only one partner, they might also find the Collaboration Server to be unnecessary, but at that price the requirement needs to be substantial before the product becomes attractive. Similarly, having to spend 4 to 10

[5] At this time I am not very familiar with the Sonic Database Service. My expectation that it can be omitted is based on the assumption that it is acceptable for this product to rely specifically on an OpenEdge database or, if another database is required, that this is already covered by a DataServer license. It is possible that, with greater familiarity with the Database Service, I would argue for an OE-only version as an entry level product and for pricing which depended on the number of databases as well as the number of CPUs.

times the price of the entry level system in order to obtain Continuous Availability would be highly unpleasant unless the Sonic ESB Continuous Availability edition also had other features which were seen to be of value.

User based pricing has this attractive character of allowing small incremental increases in licensing cost in relation to small incremental increases in usage. Of course, this varies according to the definition of "user" and its relationship to the business value of additional users, but if these are reasonably in sync, it is a license model which provides for graceful growth.

How Cheap Is Cheap Enough?

This is clearly a question with different answers in different contexts. If CI is doing an application transformation of a large suite for several hundred thousand dollars, then spending $10,000 for a tool is almost certainly not a problem, unless the tool needs to be deployed across 20 sites, in which case the benefits may or may not be sufficient to obviously justify the price. However, if a customer is implementing a pilot project, then $10,000 might well exceed other costs for the project and be perceived as an excessive investment, blocking the implementation from getting started.

It is an old truism that the best pricing is in relation to perceived value. While often difficult to implement because the substance of which value is composed may be difficult to measure and value to different companies can be different qualities, when pricing relates to value the user has difficulty in not seeing the price as appropriate, even if they wish it was less. Conversely, when the price greatly exceeds perceived value, it is difficult to make the sale. For the case of entry level prospective users of Sonic ESB products, the perceived value is likely to be quite low, especially if the immediate project is little more than a replacement of existing functionality with a new technology. But, this same prospect may well grow into more extensive use and then see greater value.

For example, during the period in which Progress was reselling Actuate and for a brief period had a minimum 2 user license available, I sold Actuate to two of my customers. In one case, the customer was replacing Query/Report, which had not been very useful to them, and they were a customer that was disinclined to spend money for enhancements. They only ever implemented a single Actuate report in production. While they value that report, their perceived value of Actuate is limited to that report. Whereas, the other customer had a keen sense of ROI and implemented many reports and forms replacements using Actuate. When the Actuate-Progress relationship ended and Actuate substantially raised the maintenance rates, the first customer naturally discontinued the maintenance, although they have thus far continued to use the one report. The other customer recognized the growth in the user base served by Actuate in their company and anticipated further growth and thus accepted the new higher figure, despite it being a rather dramatic jump.

The difficulty, therefore, is in arriving at a license basis that can be used to provide an entry level pricing option or to create other, more limited versions. I would suggest that it is generally undesirable to create versions that differ only in some fixed limit on a metric like a user count limit since it is certain that some percentage of the customer base will need slightly more than the limit and will thus be highly annoyed if there is a sharp price increase. For creating different versions, it is better to have a difference in features. This difference in features may correlate to ability to accept load, as it does with Enterprise and Workgroup databases, but this is easier to accept if it is a question of performance or capability than if it is just crossing an arbitrary limit. User or volume

count licensing is attractive because it scales so uniformly, but is admittedly more difficult to administer and it is not entirely obvious what metric would be used for scaling in a Sonic context.

I understand that it might be possible for CI to negotiate something akin to the percentage of sale option as was used with Forté, but this is not really appropriate to our current focus on application transformation since this is all services, not product. Moreover, we feel that there is a broad need, not just one limited to CI, and therefore this is an opportunity for PSC that should not be limited to a special deal with an individual company.

Sonic "Lite"

In discussing this topic with a John Green[6], he said:

> "I like "lite" products, as a vendor, because they serve to confuse the market. If a prospect thinks that they just want a low end solution, then they go get a low end solution, and they are done. If you have both low and high end solutions, then when evaluating your low end solution (for comparison to other low end solutions), they invariably also do a comparison of your low end solution to your high end solution, just out of curiosity. More often than not, that curiosity leads them to the decision that what they really want is the high end solution."

This alone, I think, makes a good argument for having entry level versions of all of the Sonic products, but I would add to this that there are many additional customer profiles for which having an entry level product is also desirable. These include:

1. Someone who has a pilot project with modest ROI, but which could serve as the springboard for a whole series of projects. An entry level product allows them to get started within the budget constraints of the pilot, but then to grow into full use of the full product when the technology has proven its value in multiple cases;
2. Someone moving into a new area, e.g., supply chain integration where the known benefits would not justify a large expenditure, but there is possible growth to greater scope;
3. Someone whose initial needs are only for Message Oriented Middleware for reliable communication, but who could find other benefits if they made that implementation on a platform which allowed for easy and modestly priced additions; and, of course,
4. Someone looking at application transformation to an ESB/SOA architecture for tightly coupled applications, but no immediate EAI requirements.

In all of these cases, there are likely to be budgetary concerns that will push the prospective customer in the direction of low end or open source products. If they move in that direction, they are unlikely to return to Sonic, even if their requirements then grow, unless their selected technology fails them in some substantial way. Alternatively, if they select Sonic from the beginning, they will remain "in the family" and, in many cases, will grow to more complete use of full-featured versions and additional products over time.

Don't make the mistake that Actuate made of limiting growth by shutting out all but the high end customers ready to make large commitments.

[6] Private communication 5/27/2006.