

ESB Exploratory Effort Final Report

Version: rc-4

1. Introduction

Universities are now designing information technology infrastructures to support research, teaching, and administration for the 21st century. The core activities of *academe* are being transformed by, and are now intensely dependent on, information technology (IT). Scholarship in the humanities, the social sciences and the sciences rely on stores of digital scholarly materials and the tools and applications that make use of them. IT empowers faculty to create global online collaborative research communities, and to deliver instruction using course management systems. Administration of the academy--be it for student, grant or employee administration--is all done via large scale administrative information systems. And all of these activities and processes, both academic and administrative, are evolving rapidly in the face of new capabilities, disciplines, research methods and government and social pressures. Our challenge is to create IT infrastructure that can meet the increasing complexity of these requirements.

Adaptability will be critical for success: tomorrow's information technology infrastructures must be much more agile than the isolated, often monolithic systems of today. Development over the previous decade produced information systems for research, learning, administration, and collaboration that were either very narrowly focused to address a particular need or very large and complex, designed to achieved integration by creating a single system that would provide many services. Room scheduling software is a good example of the former. Commercial "student information systems" are a good example of the latter. The single-purpose systems have served us poorly because they often cannot integrate with other systems. The large, complex systems have proven the most difficult and expensive to adopt and deploy in higher education. Adaptability will be achieved in the future by combining a modular approach to system architecture with robust, scalable inter-system communications. Put simply, the information systems, whether academic or administrative, must provide focused services that "talk to one

another” easily. Agility and value can be obtained by letting each system do what it does best, while providing the communication architectures that allow every system to easily access every other system’s data and services.

Authentication and identity management systems (IMS) are the best examples of this emerging architecture on campuses today. Instead of developing and supporting identity, account and authentication services for every single campus system, many institutions have deployed centralized, reusable authentication and identity services for other applications to consume. In this architecture, the IMS does what it is designed to do best: manage authentication and authorization. The IMS offers what it does as a “service” to other information systems that “consume” that service when needed. This service-based approach to managing logins provides more effective and consistent access to users and application providers across the university. And by providing IMS as a service that can be developed and implemented once but reused many times across the university, it creates efficiency and flexibility for the entire enterprise.

This same model can apply to other, much richer kinds of services. For example, many of the services offered by libraries can now be consumed not only by people, but also by other information systems. In this model, course management systems can be designed to consume the services from a library’s full-text scholarly databases or institutional repositories. This would enable a student using a course management system to move seamlessly from reading the assignment in an online course environment, to doing a search for scholarly materials, and then having selected resources delivered to a class file space, all without ever leaving the virtual learning environment. A "service oriented architecture", as this approach is sometimes called, could also enable a student information system to provide rich administrative services for faculty and students by subscribing to information services provided by other systems, e.g. retrieving grades from the course management system or applying scholarship funds to tuition balances from the student financials system to complete registration for a term. The picture being sketched here calls for a very different approach to building an enabling information technology infrastructure for academe--rather than huge, monolithic systems with tightly coupled

components that attempt to support huge swathes of campus processes, applications are designed to be modular and open--calling on others and being called by them in an open architecture, a Service Oriented Architecture (SOA).

SOA is one of the great hopes for building a more flexible, interoperable, and connected information system for scholarship as well as teaching applications. SOA is a software design pattern that creates information systems that better support human organizations by representing the functions of those organizations as componentized information services. Consider some services that universities offer: admitting students, advising students, administering research grants, awarding tenure, providing access to library materials, accepting tuition payments, awarding financial aid, granting credit, authorizing access to scholarly archives, etc. A real world SOA analysis would provide a far larger and much finer-grained taxonomy of services than this list, but the examples given here illustrate the general concept behind an SOA: an organization can be described as a multitude of discrete business process and granular, supporting services. The business processes of the organization can then be supported by combining these services together into a network of producers and consumers of services, with a backbone of rich, frequent communication with each other.

This approach has at least three major advantages:

- By modeling institutions as a collection of services with components, or “modules,” an SOA can be an effective and flexible way of supporting the activities of an organization because it creates information systems that are smaller, more discrete, and more manageable
- By componentizing large, complex systems into numerous smaller services with well defined, constant interfaces, the underlying technology of any given module can change while the backbone and form of communication stays constant. Such “loose coupling” means that code changes in one area are less likely to impact changes in the others. Hence, the resulting information system is more agile, and more adaptable to the changing requirements of *academe*.
- By allowing for changes in parts of the system without requiring changes to the whole, higher education can evolve its IT environment without undertaking the expensive, “big-bang” and highly disruptive system replacements and upgrades that are now routine for most monolithic, complex commercial software systems.

If SOA is the way to provide flexible and interoperable software for higher education, then the “enterprise service bus” (ESB) may be a key piece of infrastructure for supporting SOA. In a computer, a “bus” is an electrical subsystem that carries messages from one component to another. For example, the computer bus transports data generated by the central processing unit to and from the disk controller for storage. It also transports data from the disk controller to the primary memory controller. The orchestration of data between components (that share bus resources) is achieved through established communication protocols that isolate internal details of each controller while enabling interoperability among a wide array of hardware components. An “enterprise service bus” (ESB) serves an analogous function for SOA components. An ESB can be characterized in as an integration technology that:

- delivers a Service Oriented Architecture (SOA) through a fabric of **service end points**
- uses rich **messaging** and **transformation** capabilities for reliable, any-to-any delivery of events and data
- supports **service virtualization** to abstract the location and internal workings of providers from consumers
- enables **orchestration** of lower-level sub-services into higher order services
- supports ongoing system changes via **configuration** rather than reprogramming

In less technical terms, the ESB attempts to be a translator and communicator that allows multiple systems supporting an organization to talk to one another. It proposes to be both the “backbone” and the “glue” for a new generation of more flexible, more powerful and more adaptable information systems.

Some of the high level advantages of adopting an ESB are:

- Service virtualization isolates modules from each other; changes in one don't mandate updates to others.
- Service discovery allows new providers and consumers to 'hop on the bus' with relative ease.
- Service orchestration allows complex, higher order services to be composed from simpler, lower level services.
- Changes can be made through configuration rather than (re)programming, thus increasing flexibility and reducing costs.
- Incremental services can be added with relatively little overhead or cost by relying on the underlying capabilities of the bus.

- (Legacy) systems not designed with SOA in mind can use the bus to expose themselves as services in an SOA.
- Building an SOA with an ESB infuses all services in the domain with the rich messaging, routing, transformation, and management capabilities inherent to a bus.
- In contrast to EAI (Enterprise Application Integration), ESB's do not require a central rules engine or message broker.

While the advantages of an ESB may be attractive, adopting one also carries risks and costs:

- ESBs introduce an extra transport layer when compared to regular (one-off) messaging solutions.
- An ESB represents a new layer of IT infrastructure in most organizations, and comes with new demands on money, hardware, staff time and staff skills
- The value of the ESB is achieved only when many disparate systems adopt and leverage its capabilities. This requires agreement on architecture, technology, messaging approaches across a service domain.
- Well-designed ESB's can deliver service interoperability, but data integration and interoperability require agreed data models, which can be complex and difficult to establish.
- Benefits rarely accrue with the first few projects or services; an ongoing commitment to a bus is required to achieve a critical mass of services and a positive ROI (return on investment).
- For effective implementation, ESBs require a mature information technology governance model and a well-defined enterprise IT strategy to be already in place
- Technical (message throughput) and organizational (governance and support) factors can limit an ESB's scalability, keeping it from truly being an 'enterprise' solution.

2. The Project

Given both the advantages and recognized complexities in adopting an ESB, as well as the “buzz” it has generated in the last five years, it behooves anyone designing, building, or investing in information technologies for scholarship, teaching, or administration to understand the potential role of ESB as infrastructure in higher education. This investigation aimed to understand the state of the ESB market (both open source ESBs and commercial ESBs) and their current capabilities to meet the 21st century IT infrastructure needs for academia. It is certainly a snapshot in time, as the SOA and ESB fields are evolving rapidly. But that snapshot is essential for decisions either currently

being made or ones on the immediate horizon for universities, publishers of scholarly materials, and the developing “community source” movement in higher education. Indeed, some of these projects are already facing the question of whether ESBs are the best way to deliver interoperability across campus systems. The Andrew W. Mellon Foundation, recognizing this need and opportunity, funded this investigation into ESBs for higher education.

Led by Carnegie Mellon, and along with partners at Stanford, Cambridge, Cornell, Indiana, the Massachusetts Institute of Technology, the University of Chicago, and the Quali Student Team, this exploration focused specifically on what role an ESB might play in the emerging SOA's of academia, and the current ESB marketplace. The main thrust of the exploration effort contained the following objectives,

- To provide formative evaluation information that can assist any project looking for an OSS and/or commercial ESB.
- To assess if it is possible to re-purpose an ESB from financial/administrative use to academic support, to ascertain at a high level if it would require substantial re-development or re-configuration, and to provide first-steps guidance as to what changes are likely to be required
- To assess the state of the best OSS/commercial ESB product(s) against the dominant commercial alternatives.
- To assess the plans for Quali Rice against the same metrics developed for the ‘pure’ ESBs (to the degree possible given the pre-development status of Rice). Because Rice is different in starting point and goals from a ‘purist’ ESB, this should be a formative assessment, and serve to give the Mellon Foundation and the various Mellon projects a clearer, multidimensional understanding of how Rice will be similar to and different from ESBs, with which each project can reach their own conclusions about the pros/cons of the evolutionary/Rice approach to enterprise integration vs. a purist SOA/ESB approach.

The timeframe for this project was very aggressive given that this was mostly a voluntary effort and was compressed into about four to five months of work.

Defining an ESB

The group's first challenge was to establish a consensus definition of an ESB. Most members had different views of what made up an ESB, reflecting the varied and competing notions of a bus across the industry. The loose boundaries around ESBs are

due in part to the ongoing, rapid evolution of the technology space, and in part because many technologies now laying claim to that space evolved in parallel from different, but overlapping domains (SOA, web services, ESB, EAI). Many ESB products morphed from an earlier product or effort and carried forward features that, while they added value, were not necessarily part of an ESB's core functions.

The group decided to ignore the evolutionary paths and focus on the core features of the current solutions, establishing a lowest common denominator of essential functions. In order to keep the scope manageable, the group did not consider or define the federation or integration of ESBs. By focusing on those aspects that distinguish an ESB from SOA in general or the simple use of web services, the group agreed on a basic definition as outlined in section 1. Essentially, an ESB

- Is a means to realize a Service Oriented Architecture (SOA);
- Supports messaged-based, any-to-any delivery of events and data;
- Uses a service registry to support service virtualization; and
- Provides an agile and reconfigurable platform for the execution of processes.

These characteristics rely on a core set of essential capabilities:

- Messaging
 - One way (push)
 - Two way (query/response)
 - Publish/subscribe
 - Queuing (store & forward)
- Transport, filtering, routing
 - Rules-based routing
 - Content-based routing
 - Support many formats of data (not just XML)
- Highly flexible transformation capabilities of data objects
- Discovery and directory
- Orchestration & workflow
- (Interoperability with) AuthN/AuthZ
- Management
 - Monitoring/logging
 - Diagnostics
 - Configuration
 - Alerting/notification
 - Auditing

These fundamental core capabilities of an ESB provide the capabilities to rapidly build, expose and manage an SOA. An ESB can support large, complex and dynamic SOAs by providing internal sub-services such as discovery, messaging and transformation services, as well as orchestration and process choreography. For the right environments (SOAs with more than 20 services, frequent changes to services and needs for process orchestration)¹, ESBs give an organization an infrastructure to deliver an SOA that is uniform, consistent and manageable, while still providing the flexibility, adaptability and extensibility that SOA promises.

Evaluation Process

To ensure the success of the project it was essential that the requirements for any ESB be driven from specific application use cases that represent real world examples from higher education. In January 2007, the Mellon Foundation solicited twelve organizations within higher education that provide applications supporting the arts and humanities, scholarly research, course management, and administrative computing. Select applications from each institution served as use cases to drive the requirements gathering process for an academic ESB. The overall set of identified needs spanned two categories: technical requirements and business requirements. The technical requirements were written from the perspective of an application developer or project architect. The business requirements were written from the perspective of a project architect or technical manager.

The process for extracting these requirements was based on "a day in the life" scenarios. This process constructs a narrative of a single development problem, and identifies both economic and technical stakeholders of each use case. It is broken down into two phases.

- Phase I, called "a day in the life before," begins with describing a specific instance in the development cycle, and focuses on the difficulties, frustration and "pain" in both developing as well as operating/maintaining the application in question. The author identifies who is affected ("feeling the pain"), as well as the technical and economic outcomes of the end product.

- Phase II, called "a day in the life after," the author of the use case imagines how the application and business environment would change by using an ESB to relieve the pain. This approach specifically challenges the use case author to identify specific features of an ESB that an application developer requires, and would most want to leverage.

To record the information from this "day in the life" process, a detailed matrix was populated for each use case. The next step was for the author of the use case to extract specific requirements for the application in the business and technical domains, based on the questions and considerations as follows.

Business Questions & Considerations:

- What costs does/will the organization incur, both direct and indirect, before and after the proposed solution?
- Can the efficiency gained be measured; if so, how? (Please give examples of the appropriate costs.)
- Can second order impacts to other groups within the organization be identified?
- What are the risks of not committing to this application effort?
- Can this application be outsourced and why/why not?

Technical Questions & Considerations:

- What data will the application handle? (E.g., rich media content, RDF, financial data, binary, etc.)
- What are the application's scalability, performance and QoS requirements?
- What are the application's middleware dependencies: messaging, identity management/authN/authZ, discovery/directory, etc.?
- What ESB/SOA management tools and services would be required?
- What workflow support (business rules and data and process policies) would be required?
- What integration, adaptability and interoperability standards should/must be supported?
- Please identify the development and maintenance costs of this application (multiple components). Address the benefits and costs of home-built, outsourced, and commercially vended components. Also, please comment on if it should be built entirely in house, use various components only, or consider outsourcing if possible?
- Please include an operational management plan, including monitoring requirements, diagnostics and configuration.

Once the business and technical requirements for each application were extracted from the use cases, sixteen ESB candidates were identified through discussions with all the participating organizations in this study, as well as from industry research from Forrester and Gartner.

- Apache Synapse
- BEA
- Cape Clear
- Chain Builder
- Fiorano
- IONA
- JBoss
- Kualo Rice
- Mule
- Open EAI
- Open ESB
- Oracle ESB
- Polar Lake
- ServiceMix
- Sonic
- WebSphere

Volunteers from the group then chose ESB candidates to research, and captured as much high level information as available on each one, including details such as:

- Product cost
- License type
- Installed base / user community
- Development roadmap, community influence and contributions
- Availability of support
- Operational & development platform (messaging, languages, IDE)
- Internal architecture (P2P, hub and spoke, broadcast)
- Management and diagnostic capabilities
- Product architecture graphic
- History of the product

Based on their use cases, each participating institution then created a list of critical business and technical requirements for an ESB, as well as a short list of three to five issues, or "show stoppers," that would prevent them from adopting any specific solution. These decline issues were used to identify candidates that distinguished themselves as likely good fits for academic application environments. The two most common examples of "show stoppers" were "unacceptable license restrictions" and "too complex/too high a threshold to adopt." In the next stage of the process, each ESB candidate was rated as "yes or "no" as a potential solution to support each application use case, along with pros and cons for its suitability to each scenario. The votes were tallied to identify which ESB solutions emerged as clear "best fits" for an academic ESB.

3. Extracting ESB Requirements From Higher Education Applications

Each participating institution identified one or two applications--either existing, in development, or planned--that highlighted the potential need for an ESB in their environment. Each highlighted the needs for agility and interoperability among systems. These real world use cases were generated by group members who were chosen based not only on their technical expertise, but also on their roles as architects or developers with experience supporting applications in higher education. Extracts from each use case are as follows; longer versions can be found on the project wiki.²

- **Preservation within Institutional Repository (D-Space) - Cambridge**

D-Space is receiving submissions in a number of formats from various sources. Some of this information comes from the institutional Virtual Research Environment, currently Sakai, and some comes via batch upload. The quality of the meta-data is controlled but variable as is the media format of the deposit operation. Consequently both Meta-Data and media format have to be checked and recoded as a continuous process. This will be performed by integrating D-Space with an ESB to connect any number of migration and validation services with the core D-Space so that the material preserved within D-Space can be managed according to preservation policies, expressed as business processes within the ESB. To perform this the ESB needs to have messaging with routing, transformation of messages and BPEL⁹ based business processes. The services that take part in this process and service orchestration include D-Space, Sakai and a number of Migration services. Sub Use Cases:

- *Expression of a preservation policy by Library Staff*
 - *Notification of media format migration to an Owner of a VRE deposited item*
- *Change of preservation policy resulting in deprecation of preservation media format*

- **Student Identity Management - Cambridge**

The University maintains several owners of Identity that have a range of authorities depending on the life cycle of the student through the university. Prior to admissions the prospective student acquires an identity known to the University granted by the various admissions organizations. On successful entry to the university the student acquired identity within the central computing service that has given them access to most University Systems. This identity is used and sometime stored by the disparate systems around the University. Two such examples are the Student Information System (PeopleSoft) and the VLE (Sakai). There is currently no mechanism for proactively notifying systems of a change of identity other than by periodic bulk updates or lookups by the target systems. An ESB would facilitate the notification and messaging use cases associated with Identity Management among the departmental core university systems.

- **Student Information System Data Feeds - Carnegie Mellon**

Carnegie Mellon University is currently using a 10 year old student information system. It exchanges information with other applications through data feeds. Until this system is replaced with a newer system there is a desire to improve this exchange of data through an ESB. Focus of this use case would be on effects of an ESB on relationship between SIS and other applications especially in exchanging of data. These applications range from Loan and Payment Processing to Blackboard and LDAP.

- **Fedora "Big Ingest" - Cornell**

One challenge for applications built on Fedora is known as the "Big Ingest" problem. This is the challenge of preparing large batches of content to be stored appropriately as digital objects in a Fedora repository. While the Fedora repository service supports a basic "ingest" operation, this operation expects to receive a single Fedora digital object. Thus, users of Fedora have taken to

creating custom services in front of Fedora to prepare large bodies of digital material to be incrementally ingested into Fedora – one object at a time. These ingest services have been developed in idiosyncratic ways, and they tend to not be easily decomposed and re-used by other institutions with the same problem. Generally, the “Big Ingest” problem begs for standardization and re-use of shared components and services. It also requires a more sophisticated approach to workflow and process orchestration. Our analysis of this process shows that following general activities are required in most ingest scenarios (and, accordingly, one or more services must be called upon to support these activities):

- *content gathering (rally together raw bytestream content)*
- *content format validation (e.g., via JHOVE, GDFR)*
- *automatic metadata extraction/creation*
- *creation of unique identifiers*
- *object integrity enforcement (constraints for genre like book, article, memo, etc.)*
- *object registration*
- *object ingest into repository*
- *object indexing for search*
- *object replication for redundancy*

The Fedora Project currently provides open-source repository software and a set of supporting services to facilitate a full-lifecycle approach to the creation, management, and long-term preservation of digital objects. We are looking to the next generation of Fedora which will be “enterprise-ready” and will use standard approaches to workflow, messaging, transactions, and transformation. We are interested in exploring the ESB architectural pattern in general, and open-source ESB components in particular around solving the “Big Ingest” problem. Our hope is to have a future deployment of Fedora “ESB-ready” and to distribute key open-source components, most likely a JMS broker, workflow engine, and other smaller services. We need to understand which components we must develop and which things we assume are part of the overall “enterprise” environment in which Fedora will be installed.

- **Academic Information Space Project - Cornell**

The primary goal of this project is to create an “Academic Information Space” (AIS) that enables users to work with digital resources, from digital libraries, the web, and their own collections, in an integrated environment that provides powerful tools for analysis, collaboration, and delivery. At the core of the AIS lie two digital object repositories, the Digital Library and the “Scholars’ Workbench”. The entire AIS architecture is reconceived so that it can take advantage of an Enterprise Service Bus. The three main Environments (Content Creation, Authoring, and Analysis) are deconstructed to achieve a clear separation between user interface application, backend services provided by an ESB, and custom services to support the particular needs of the scholarly use cases. The main opportunity area is to use a standard workflow engine and standard approaches to process orchestration engines offered by an ESB platform.

- **HR Hiring - Indiana**

IU like all other universities has a hiring process. The focus of this use case is how the use of ESB could extend and add value to this existing process. The idea is that given an ESB with message queue capabilities, that we could project messages out of our mission critical enterprise transactions that units could then listen for and take appropriate automated programmatic action based on. This will of course have a security element to it, and a transactional element to it as well.

- **Kuali Student - Kuali Student Team (University of British Columbia, Carnegie Mellon)**

Kuali Student is an application being designed from ground up based on SOA principles. Focus of this use case is the use of ESB within the project to enable communication between Kuali Student services. External communications of

Kuali Student with other enterprise applications or external services is not considered in this use case. Since all of Kuali Student services are Web Services, expectations from ESB can possibly be very limited but at minimum it should include:

- *Discovery of service*
- *Support for synchronous and asynchronous messaging*
- *A service monitoring console*

Although other features like BPEL engine, security, or transaction support might be quite helpful to the project. Since Kuali Student System does not exist yet, we would try to demonstrate the contrast between designing the system with ESB and designing without it.

- **Integrating a “Standalone” Performance Review Product with SAP HR using an ESB - MIT**

Currently MIT does not have an application in this area, but there are some pilots in certain departments. We would like to assess the feasibility and cost of integrating these types of systems with SAP HR. There is a possibility that different departments may pilot different tools, and so each would need integration, or we may start with one Performance tool and decide to replace it based on experience. In either of these cases the cost and effectiveness of the integration with our main HR system is a large factor. We would hope that using an ESB solution to provide real-time integration with SAP would be both cheaper and easier, than a custom integration approach, and would reduce future integration cost if we needed to use more than one performance review tool or switch to a new one in the future. Some of the major factors in the evaluation might be:

- *Integration with SAP*
- *Ability to integrate with other commercial tools (Performance management tools)*
- *Security*
- *Operational cost*

If the ESB integration strategy was successful, the HR function would benefit by being free to choose the best product without constraining the choice due to integration cost factors.

- **Real-time Data integration with Departmental Systems - MIT**

Currently MIT makes its central administrative data available to departments through a data Warehouse that is refreshed nightly. We're looking at ways to improve the data distribution to be real-time. An ESB could provide one solution. In this case we would be looking to have a technology that worked with as many types of consuming systems as possible and was easy to use. Most of our departments run small systems with varying technology and don't have a large IT staff. This would mean that the ESB solution must be easy for them to implement and support, and can't work in just one technology environment or language.

- **Integrate Course Services with Library, Student, IdM & Infrastructure - Stanford**

Course Management relies on services within and outside the Library systems. A future, well integrated system will require connections to other Library services, e.g., to reserve course materials or provide access to special collections, and with numerous external systems -- the Course Registry for curriculum information, enrollment and other course details, the Identity Management System for student descriptive data and status, network and account services for Web/File space and access, the Bookstore for textbook orders. Some of this integration will be bi-directional, e.g., sharing information about TA's and auditors with the Student Information System, or the ability to post grades through Course services. An ESB that is truly "enterprise" will facilitate this level of integration if it supports a variety of integration patterns and an ease of plug and play across technologies.

- **Digital collections process framework - Stanford**

The Digital Library Services and Services group (DLSS) supports a variety of efforts to digitize everything from large collections of material to individual items, as well as handling born digital material. Processing requirements consist of

many of the same or similar steps, but can vary broadly depending on funding, data formats, and whether the effort is purely in-house or collaborative with other parts of the University or other institutions. To date the processes have evolved through separate groups and in service of separate projects. The DLSS is building a framework to support this work through common services that can be assembled to support a variety of current and future needs.

- **Interdisciplinary Digital Scholarship - University of Chicago**

Increasingly, scholarship is interdisciplinary both in the academic and data sense. Digital images of medieval manuscripts, marked-up Mayan text, audio from evangelical sermons, and turn-of-the-century census data is singularly used in research, teaching, and learning, but can also be combined to form new lines of scholarship heretofore considered too difficult or complex to undertake. Traditionally, scholarly projects have been largely vertical in nature making reuse of content, methodologies, and results difficult. The goal, then, is to minimize siloing of digital scholarly projects; maximize the potential for content, component and application reuse; maintain the degree of project-level academic freedom enjoyed through custom development; and create opportunities for path-breaking scholarship. Motivation: By moving to a services model of scholarly application development, the vertical siloed nature of digital scholarly projects can be minimized and the potential for interdisciplinary reuse of digital scholarship can be increased provided there is a widely accepted and adopted core of essential services. Such services can include (but not be limited to) service discovery, data discovery, asset retrieval/ storage, data translation/transformation, authorization, and extensibility via custom service registration.

4. Findings

The Current State of the ESB Market

The ESB market is large and dynamic, growing in terms of the number types (commercial and open source), and maturity of available solutions. (During the window of this investigation, two new ESB solutions were announced and/or released their first code: Kuali and Apache Synapse). Overall, the market appears a long way from stabilizing: even the definition of an ESB is still evolving. Some of the tension in the conception of an ESB comes from its two evolutionary strains. A set of the current ESB solutions are evolved EAI tools. In general, these represent a hub-and-spoke architecture (with a single messaging & transformation engine dominating the system). Another set of ESB technologies has emerged from the web services camp; these generally have a more distributed architecture.³

There are three distinct camps in the open source ESB arena. One camp is for fledgling solutions that have yet to achieve critical mass. For those products without widespread adoption, the only support options are for the adopter to support it themselves or get help from the user/developer community. The second camp ranges from more mature small to medium size efforts that may be associated with either a loosely or tightly coupled support organization. These support organizations “may” provide services such as configuration, installation, as well as customization of the source code. The final camp comprises larger, more established players. These have a healthy ecosystem of fee-for-service, value-adding developers, integrators and supporters (e.g., Iona, JBoss). Service companies have developed successful business models by extending, consulting on and supporting open source ESB deployments among their customer base.

For the commercial marketplace, Forrester divides the ESB arena into two distinct camps. Some vendors (e.g., Sonic, IBM) have developed purpose-built ESB solutions from scratch, and these can stand alone in an IT architecture and are designed to provide integration services in heterogeneous environments. Other vendors (e.g., Oracle) have extended their existing portfolio of enterprise systems and added ESB capabilities as one

more element in their suite. These solutions are best positioned to provide bus services to an environment that already has an extensive deployment of that vendor's products.⁴

Similarly, Gartner divides the ESB marketplace into two camps, though with a slightly different distinction.⁵ Vendors with large installed bases of (and revenue streams from) enterprise application suites (e.g., BEA, Oracle, Microsoft) offer complementary ESBs that leverage the unique features of their environment, particularly the application server. These ESBs are well suited to more homogenous IT environments, supported by disciplined teams of developers using uniform interfaces, data and process models. (If half the services in any given SOA domain run on the same application server, Gartner recommends going with the dominant vendor's ESB.)

The other ESB camp Gartner recognizes is one focused on integration of diverse services on heterogeneous platforms. These environments may have a mix of package applications and locally developed custom applications. The ESBs in this category (including Apache ServiceMix, Mule, and Sonic) tend to emphasize platform independence, transformation services, legacy application adapters and protocol bridges.

Because the market has yet to fully mature, no single ESB solution can meet every need for every niche. Even within the same organization, it is common for there to be multiple instances of ESBs, often from different vendors, that are tuned and managed to support different SOA domains with distinct ESB needs.⁶

Three Distinct Profiles of ESB Needs for Higher Education

This marketplace research echoes our findings on higher education's mix of requirements for ESBs. The use cases, business and technical requirements produced during the course of this exploratory effort demonstrate there is no "one size fits all" solution, even among the somewhat cohesive community that higher education represents. This stems from the fact that academic institutions typically have many areas of distributed and/or autonomous control. Enterprise IT manages large scale application domains, e.g., ERP (enterprise resource planning) systems, often guided by established vendor relationships.

They can dictate tools and interfaces that will be used across the institution, sometimes with only partial success. On the other hand, schools, departments, libraries, etc. represent smaller, independent data and development domains, spanning all levels of sophistication and deploying diverse technologies.

Therefore, the scope, scale and intended users of an intended ESB service domain will be major drivers in selecting the right bus for a given purpose. In this study, three distinct sets of requirements emerged from the use cases. To identify which camp fits a particular institution and need best, the following questions must be addressed.

(a) Is an institution ready for a centrally maintained enterprise-wide ESB? Is it intended to be used with major administrative systems, to expose their services across each other and to the community? Or is it intended to offer ESB support for schools, departments, and projects that might want to join in lieu of a local bus implementation? In either case, an institution will be devoting considerable planning and resources to supporting an ESB and will look for highly evolved, feature-rich solutions (including administrative and operational features, such as consoles and clustering).

- Scalability and reliability figure prominently in such planning
- Risk (e.g., lack of product maturity) will be a showstopper
- Dedicated staff will likely be required (and available) to adopt, develop and run the ESB
- Third party contract support will likely be very attractive (or a requirement)
- Cost and licensing are relatively less important.

(b) Is the ESB being considered as a "local" solution -- to build an internal service-oriented infrastructure for an IT sub-community such as within a department or the libraries? For these use cases, services will likely start small and evolve over time. Correspondingly, the focus will be on internal needs rather than addressing the needs of a larger community of ESB participants.

- internal support will be local and likely lean
- the barrier to entry for an ESB needs to be relatively low
- it must be compatible with existing infrastructure and staff skill sets
- a cost effective solution is a must, making free open source attractive

Cost and operational size/complexity are a major concern here, as is avoiding vendor lock-in, which tends to exclude the commercial vendors. In some cases a local solution might be obviated if the institution had support for an enterprise-wide ESB per above, but that is not a situation found today nor could it be considered a prerequisite.

(c) A third distinction, one that might apply to work done under either scenario above, is the repackaging of services for software redistribution. Some collaborative projects such as Sakai, Kuali and Fedora have already adopted or are moving towards service oriented architectures, and can themselves benefit from ESB facilities. Such open source efforts require compatible embedded solutions, so licensing restrictions are deal breakers. And because they may be adopted at a range of institutions with a range of IT resources and skill levels, simplicity and ease of adoption are also critical.

What Makes an "Academic" or "Higher Ed" ESB?

This effort's sampling of use cases reveals a diversity of needs across higher education. Despite the diversity of applications and use cases, some common themes and requirements did emerge. These clear patterns stem not from the technology alone, but rather from what differentiates Higher Ed from the marketplace in general. Taken together, these distinguishing characteristics make an Academic ESB more than the "lowest common denominator" of basic features in the ESB marketplace. Two key characteristics of higher education institutions stand out:

- Academic institutions are far from homogenous internally. They are themselves a collection of tiny to large communities deploying a broad range of technologies -- from operating systems to desktops, from languages to databases to directories. There can be wide variations in IT support and technical competence within units. Collaboration and interoperability present distinct challenges in such an environment.
- Academic institutions are highly collaborative. They seek common solutions for institutional needs not met in the marketplace. They leverage resources in and experiences from one school to benefit others. The ESB investigation itself is a reflection of this spirit. An academic ESB must contribute to the ease of collaboration and sharing in ways specific to Higher Ed.

Because of this, an ideal "Academic" ESB would be characterized by open, community control, standards-based interoperability, and lack of use restrictions. This favors today's open source offerings, unencumbered by potential license restrictions, and open to technical contributions from any interested stakeholder. Collaborative, open source software solutions are generally appealing to academic institutions; they seem better positioned to go beyond what a commercial ESB vendor might do on campus.

The use cases sited in this effort, and in particular the list of showstoppers that would prohibit adopting an ESB, reinforce this finding. The showstopper identified most often by the use-case teams was a strong desire for an open source license, a reflection of a need for an inexpensive solution that is also one whose evolution higher education can help shape. This is coupled with a wariness of all-in-one large vendor solutions and the risk of vendor lock-in. Not far behind licensing was concern over maturity -- as demonstrated by the existence of a community of users, partners, and vendor support. The project wiki contains the full set of showstopper issues.⁷

The technical requirements surfaced by the use cases were more distributed, with the highest ranking being the need for standards, flexibility, and interoperability. Strong support for rich messaging (mainly JMS) came next, along with a assortment of management and diagnostic features. Finally, simplicity was a common technical concern --for both adoption and ongoing operations. The project wiki contains a fuller set of technical requirements derived from each use case.⁸

A consolidated list of essential requirements for an Academic Bus comprises the following:

- Open source license to allow for code changes and facilitate solution sharing across institutions
- Large and active community, including contract-support vendors
- Rich, maturing functionality
- Low barrier to entry; supporting multiple levels of buy-in to suit a range of resource needs
- Small operational footprint
- Based on open standards; does not require a specific application server or similar vendor lock-in

- Compatible with common infrastructure -- Kerberos, AFS, LDAP/directory, Apache/Tomcat, etc.
- Facilitation of inter-institutional sharing of solutions
- Compatibility with continuing or future Higher Ed efforts built on common architecture, e.g., Sakai, Quali, FEDORA

5. Recommendations

The ideal solution would be one that had the maturity, support and broad adoption that would allow it to be considered for all scenarios, even in cases that required large scale (campus-wide) deployment and integration with ERPs or other vendor solutions on campus. We did not find any single ideal product. Three of the open source ESBs stood out as being the most promising candidates for an Academic ESB. These three stand-outs were rated positively by the most use cases, with very few negative votes cast:

- **Service Mix**

Suited for 11 of 12 use cases, ServiceMix is a relatively mature, open source, Apache project. It has wide adoption, a mature feature set (including a solid management console but limited diagnostic abilities), and available commercial support. It is heavily tied (for better or worse) to the Java Business Integration (JBI) specification [JSR 208](#), a rich (but not lightweight) technology that has yet to achieve full traction.

- **Mule**

Also suited for 10 of 12 use cases, it offers open and flexible deployment, built on top of a robust messaging platform, with an active development and support community and widespread adoption. It is particularly strong as an integration platform, with a large base of third party connectors. It too raised concerns about advanced features such as clustering support and monitoring.

- **JBoss**

Suited to 8 of the 12 use cases, JBoss offers relatively mature components from a respected source, with a large community of users, and contractual support

options. It lacks some functions of a full-featured ESB (e.g., clustering, diagnostic console) and may be too tightly bound to the JBoss suite (including the JBoss Application Server) to be fully effective in heterogeneous environments.

Note: specifics about the choices each use case made can be found in the "initial evaluation" matrix in the "pros and cons" section.

We do not offer similar recommendations for the commercial vendors. While generally very strong, their size, complexity, and cost made them a better solution for large scale, internal use. But in such scenarios (e.g., an ESB to support an administrative application service domain), institutions are likely to have a wide variety of distinct, locally driven criteria -- vendor relationships, established platforms, compatibility with ERPs, etc. -- guiding such an ESB selection. This makes selecting a commercial ESB for administrative use (e.g., IBM Websphere over Oracle Fusion ESB, or vice versa) a matter of local rather than multi-institutional concern. With licensing restrictions factored in on top of their complexity, commercial ESBs rated as poor fits for both smaller scale (departmental) and open source academic project use.

This raises a fundamental and pragmatic question: can there be only one preferred ESB for higher education? Can any single ESB meet the diversity of business and technical needs for academia? Would a single ESB provide synergy and benefits across institutions even though it won't be the only ESB or SOA solution present on campus? It is clear from the market trends that the organizational and technical realities of supporting an ESB dictate that one, single ESB for the entire enterprise may be unattainable and unsupported. Rather than spanning an entire enterprise, most ESBs support distinct "SOA domains" with a subset of an enterprise's IT environment. These domains are typically defined by a suite of related applications, supported by a common pool of developers, who have adopted the same methodologies and designs, and who are bounded by organizational lines, all reporting up to the same executive or governance body. given this present reality, no single instance of an Academic ESB could satisfy all

disparate campus needs technically, much less overcome the operational and governance hurdles that would be necessary.

However, even with the diversity of requirements and limits to the scalability of an ESB, we believe there can be a single, preferred Academic ESB: there are significant benefits to standardizing on one project and having multiple installations/implementations of it on each campus. Higher education has seen the successful promotion and evolution of common architecture and infrastructure -- in Kerberos, AFS, Shibboleth, LDAP, in Identity Management and related standards -- efforts that bring true benefits to the community even in cases that still have multiple implementations and a mix of vendors. Many smaller IT groups would welcome guidance and community synergy in creating flexible, shared service architecture at a local level. The community as a whole could help each other evolve towards a service oriented architecture that shares common components and philosophy. Such an environment could only have a positive effect on the challenges of scaling a service architecture to an enterprise, or in coordinating the integration of services across multiple domains.

So what will it take to vault one of the three leading contenders into the status of the "preferred" Academic ESB? While JBoss, Mule and ServiceMix distinguish themselves as likely candidates, a fit with the preliminary business and technical requirements identified in this exploratory effort is not enough. To become a reality, an Academic ESB requires community traction to achieve critical mass. Specifically, there must be:

- **Interest:** Higher education institutions must recognize and rally behind the promise of an ESB to deliver agile, rich, and manageable SOAs. This effort has confirmed that there is initial interest in ESBs, but not yet widespread recognition that they (may) have an integral role to play in the burgeoning SOAs of academia.
- **Interoperability:** The ideal Academic ESB would support the *de facto* technology standards in higher education: Java, JMS, Apache/Tomcat, Kerberos, Internet2 middleware-related standards around authentication, authorization, data models (e.g., eduPerson), diagnostics, etc. It would also avoid any specific vendor- or technology lock-in, through the use of open standards and interfaces to accommodate the range of technologies that have been adopted across higher education.
- **Influence:** Higher education must be able to influence the features and futures of the bus. Our institutions must participate in shaping the evolution of an Academic

ESB by creating and sharing components we need. This requires establishing a direct link to, and active involvement in, the applicable development community. This is a particular requirement for institutions wishing to innovate in the areas of SOA or ESB.

- **Incentives:** There need to be tangible incentives for adopting a common ESB across higher education. Cost savings; software reuse; compatibility across the realm of open source, higher education software projects; access to the expertise of a community who have ventured down the ESB path; all these and more must outweigh local drivers particular to each institution.
- **Implementations:** Individual campus adoptions, along with integration into such efforts as Sakai, Quali Student, and Fedora, are necessary to establish critical mass around a preferred ESB for higher education. Each has its own reason for adopting an ESB, but a "preferred" Academic ESB means little if these major efforts ignore it, or adopt different solutions.
- **Inspiration:** Fostering an Academic Bus is more than selecting the right technology. It will require a vision towards a goal, resources to execute a plan of discovery, communication, integration and growth, and a strategy for long-term sustainability beyond simple community synergies. In short, it will need an institutional home which includes a well-defined leadership role for this to succeed. The institution must be a large stakeholder in the use of the Bus.

Next Steps

To further the identification, evolution and adoption of an Academic ESB, these points must be addressed. Accordingly, we recommend the following next steps:

1. Conducting a symposium on ESBs and their potential as an architectural solution in higher education for interested universities, vendors and open source communities. As envisioned this would be part educational and part working session, with the aims of (first) creating a common baseline understanding of the current state of ESB's among the academic community, and (second) vetting and expanding on the requirements identified in this report, and (third) helping catalyze a community approach and adoption of a common ESB technology.
2. Pursuing an in-depth exploration of the three leading contenders identified in this report (JBoss, Mule and ServiceMix). We recommend that this assessment include:
 - Establishing an objective set of assessment criteria that incorporate the requirements for an 'ideal' academic ESB that have been identified in this report;
 - Trial installations and prototype deployments of each of the three platforms by a common technical & evaluation team to establish a hands-on baseline of experience with each one;
 - Pursuing a deeper investigation into the features, operating platforms, and community strength of these top three contenders, including interviews of

- sites with successful deployments of their software in contexts analogous to higher education;
- Substantial discussions with the lead developers/active development community to gauge their possible evolution and interest in becoming an (the preferred) academic ESB;
 - Final evaluation and rating of these three contenders on a balanced score card to identify a single, preferred provider for moving forward with trial deployments.
3. Identifying (if they already exist) or catalyzing the implementation of two successful, relatively mature implementations of an ESB in a university environment to serve as reference models for the higher education community. This exploration into ESB's revealed widespread interest in their potential, but also much speculation and uncertainty about how a bus might be deployed, configured, operated and leveraged in an SOA within a university. One or two reference models would serve both to illustrate the benefits (and costs) of adopting an ESB, and also to identify best practices among the vanguard of adopters in higher education. One or both of these reference models should specifically illustrate how an ESB can be used to enable existing legacy applications (especially enterprise resource planning (ERP) applications) to become SOA endpoints within a more open, agile environment.
 4. Investigating strategies, methods and architectures for establishing interoperability across ESBs--both across SOA domains within individual institutions as well as among institutions. While exploring the federation of ESBs was out of scope for this initiative, it is quite clear from our findings that this will be a necessary undertaking in the near future, as the uptake of ESBs into higher education IT environments blossoms. This investigation into ESB interoperability should specifically include (but not be limited to) OpenEAI.
 5. Continuing to monitor the progress of Kuali ESB development, and mapping out its relationship to a preferred academic ESB. While Kuali was of great interest to the participants in this study because of its innate focus on higher education and ECL license, it's quite new (it released its first code late in the process of this investigation) and does not yet have wide community uptake. We recommend that the next phase of this ESB work factor in Kuali's relative maturity, development trajectory and relationship to a preferred academic ESB. Without addressing this issue head on, the resources and attention of the higher education community are at risk of being diluted across multiple possible ESB solutions.

Open Questions

This exploratory effort uncovered some key questions that were not resolved during the course of this assessment. These should be addressed in the next phase of the project.

- What other organizations and institutions have a stake in an academic ESB, and what headway have they made in addressing the problem space or associated pieces? How should a broadly-framed community effort take shape that

recognizes these factors, and how should the funding and organization be structured to not only discover it but to sustain the effort and adoption across higher education?

- What are the implications for current and future projects if there were a preferred ESB? Does it affect the funding or technology choices for projects like Sakai or the Kualu/Rice ESB effort?
- "What if" the license restrictions on commercial offerings were removed; would the choices/recommendations change?
- How will institutions integrate and manage the inevitable multiplicity of ESBs that will come from both local development, and embedded in other software projects, e.g. Kualu or Fedora? Exploring such compatibility issues was outside the scope of this study.

6. Postmortem

A collaborative effort involving organizations that are major leaders in higher education has many challenges. Since the conception of this effort in January 2007, the members of this group have been challenged by a complex goal and a short time-line. We felt that it would be beneficial to themselves, the Mellon Foundation, and the community at large to provide some feedback from our experience in completing our goals and objectives of this effort. Here, by consensus, are some of the insights we gleaned:

1. Deciding on how to decide is much more work than deciding. Defining the process for identifying vendors, processes, requirements and findings was a tremendous effort. While the final outcome of defined processes for the analysis was essential to the success of the project it was challenging to bring the group to a consensus.
2. Participants had other day jobs. This was mostly a voluntary effort and while being extremely important to all institutions involved it challenged the human resources that were needed to sustain its success.
3. Given the wide variety of institutions and different representatives such as use case authors, project managers, architects, and software engineers, all with deep and a wide array experience on this subject there was a high degree of confidence in the quality of the analysis and feedback.
4. The Wiki was a key tool but when it was unavailable it was a problem. In collaborative efforts such as this communication tools are essential. While there were times in the early stages of the project that IT resources were unreliable they became more stable as the project matured. The group cannot understate the need for a stable IT environment that enables the ubiquitous sharing of information.
5. Our use-case-driven process was evolutionary and extremely helpful. While the majority of the members used the use case process for defining requirements, defining one that involved so many use cases was challenging but evolved as the group tackled milestones. The ability of the group to develop and change

existing processes to aid in acquiring important requirements was essential to the success of this project.

6. While the ESB topic was of interest to all institutions involved, it unearthed many interesting topics such as addressing federations of ESBs. The group did a good job in scoping the effort so not to lose track of the core goals and objectives.
7. A dedicated project leader/manager was essential to drive the process. Without someone with the responsibility and authority (limited of course by the voluntary nature of much participation) to assign ownership to tasks and coordinate this effort, the goals as well as objectives could not have been met.

7. Acknowledgments

This effort would not be possible without the vision, Herculean effort and steadfast commitment of the following individuals their organizations.

- Cambridge University
 - Ian Boston - use case author
- Carnegie Mellon University
 - Chas DiFatta - project leader, core discussion and analysis team, final report author
 - Parviz Dousti - use case author, core discussion and analysis team
 - Joel Smith - principal investigator, final report author
- Cornell University/Fedora
 - Daniel Davis - core discussion and analysis team
 - Sandy Payette - use case author
- Indiana University
 - Brian McGough - use case author, core discussion and analysis team, vendor analysis
- Ithaca Harbors Inc.
 - Keith Kiser - Wiki and Email services, IT support
- Quali Student Team
 - Parviz Dousti - use case author, core discussion and analysis team
- Massachusetts Institute of Technology
 - Scott Thorne - use case author
- Mellon Foundation
 - Ira Fuchs - funding sponsor
 - Chris Mackie - solicitor of the project
- Stanford University
 - Lois Brooks - Stanford resource coordination, final report review
 - Tom Cramer - vendor analysis, core discussion and analysis team, final report author
 - Rachel Gollub - editing and feedback
 - Lynn McRae - vendor analysis, core discussion and analysis team, final report author
 - Minh Nguyen - vendor analysis

- Mike Olive - vendor analysis
- University of Chicago
 - Kaylea Hascall - use case author, vendor analysis, core discussion and analysis team
 - Chad Kainz - use case author, core discussion and analysis team

References

¹ Schulte, Roy W. *Where to Use an Enterprise Service Bus and Why*. Gartner, 3 May 2007, ID Number G00143292

² <http://tid.ithaka.org/enterprise-service-bus/UseCases>

³ Gilpin, Mike and Ken Vollmer. *The Forrester WaveTM: Enterprise Service Bus, Q4 2005*. Forrester Research, 15 November 2005.

⁴ *ibid.*

⁵ Schulte, Roy W. *Enterprise Service Bus Usage Scenarios and Product Categories*. Gartner, 3 May 2007, ID Number G00143294.

⁶ Schulte, Roy W. *Succeeding With Multiple SOA Service Domains and Disparate ESBs*. Gartner, 3 May 2007, ID Number: G00143293.

⁷ <http://tid.ithaka.org/enterprise-service-bus/UseCaseDeclineReasonsForNOTChoosingASpecificESB>

⁸ <http://tid.ithaka.org/enterprise-service-bus/UseCases>

⁹ BEPL (Business Process Language) is a standard language for defining business processes in an SOA. (<http://en.wikipedia.org/wiki/BPEL>)