ESB Exploratory Effort

The Andrew W. Mellon Foundation
2007

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

Goal

Identify possible open-source and commercial enterprise service bus product offering candidates that meet with requirements of the Mellon supported Service Oriented Architecture (SOA) efforts and projects

Participating Organizations

- Cambridge University
- Carnegie Mellon University
- Cornell University
- Indiana University
- Massachusetts Institute of Technology
- Stanford University
- University of British Columbia
- University of Chicago

Aggressive Timeline

- **Startup Phase (Feb 1 - Feb 21)**
- Establish group and processes
- Define scope/goals/objectives (review with group)
- Establish ESB "straw man" Definition

- **Requirements Phase (Feb 22 - April 30)**
- Developing Use Cases for key applications that "may" use an ESB
- Extracting Business and Technical Requirements Matrix

- **Evaluation Phase (May 1 - June 15)**
- Initial Vendor Classification Matrix
- Use Case Decline Reasons for NOT Choosing a Specific ESB
- Defining a baseline ESB for higher education

- **Recommendation Phase (June 15 - July 31)**
- Final Candidate Matrix
- Final Evaluations Deliverables

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

Fundamental Questions

- Why an SOA?
- Why an ESB?
- Requirements/Nature of Academia
- Profile of the ‘ideal’ academic ESB

Why an SOA?

- SOA is an open architecture that stresses
  - Modularity
  - Interoperability
  - Component reusability
  - Direct support of human business processes
- This can produce IT environments that are
  - More manageable (components are bite size)
  - More efficient (through service reuse)
  - More agile (can evolve with time, changing requirements)

Why an ESB?

- ESBs are just one of several ways to achieve an SOA
  - Others include: Web services via SOAP; POX (plain old XML) over HTTP
- ESBs are best in SOA domains that are:
  - Large: more than 20 services
  - Dynamic: frequent changes to services
  - Complex: leverage message routing & orchestration to support workflow & produce higher order services from simpler components [1]

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

Requirements/Nature of Academia

- Decentralized
- Many IT groups, large and small
- Many (or no) governance bodies
- Heterogeneous IT environments
- Rapidly evolving
- Open to (and sometimes, enthusiastic about) open source software

Profile of the ‘Ideal’ Academic ESB

From the use case analyses:

- Open source, to allow for code changes and facilitate solutions sharing across institutions
- Large and active community, including contract support companies
- Rich, maturing functionality
- Low barriers to entry; supporting multiple levels of buy-in to suit a range of resource needs
- Small operational footprint

The ‘Ideal’ Academic ESB, cont’d.

- Based on open standards; does not require a specific application server or similar vendor lock-in
- Compatible with our common infrastructure – Kerberos, AFS, LDAP/directory, Apache/Tomcat, etc.
- Facilitates inter-institutional sharing of solutions
- Compatible with continuing or future Higher Ed efforts on common architecture. E.g., Sakai, Kuali

Establishing an ESB Definition

- Goal: establish the lowest common denominator; don't overload the definition
- Define an ESB in terms of capabilities or features that enable or add value to a SOA
- Focus on ESBs individually: discussion of integration/federation of multiple ESBs is out of scope

ESB Definition

An ESB is 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

ESB Definition, Continued

The fundamental internal capabilities and facilities of an ESB can consist of but are not limited to the following:

- 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)
- Transformation
- Discovery and directory
- Orchestration & workflow
- Interoperability with AuthN/AuthZ
- Management
  - Monitoring/logging
  - Diagnostics
  - Configuration
  - Alerting/notification
  - Auditing

Reasons to “Hop on the Bus”

- Service virtualization isolates modules from each other: changes in one don’t mandate updates to others
- Service discovery allows new providers and consumers to find services and how to use them
- A bus provides an agile and reconfigurable platform for the execution of processes
  - Orchestration allows complex applications to be composed from reusable, lower level services
  - Changes can be made through configuration rather than reprogramming, thus increasing flexibility and reducing costs

“Hop on the Bus” cont’d.

- Incremental services can be added with relatively little overhead or cost by relying on the underlying capabilities of the bus
- (Legacy) systems not designed for SOA 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 of the bus
- In contrast to EAI (Enterprise Application Integration), ESBs do not require a central rules engine or message broker

ESB Risks & Costs

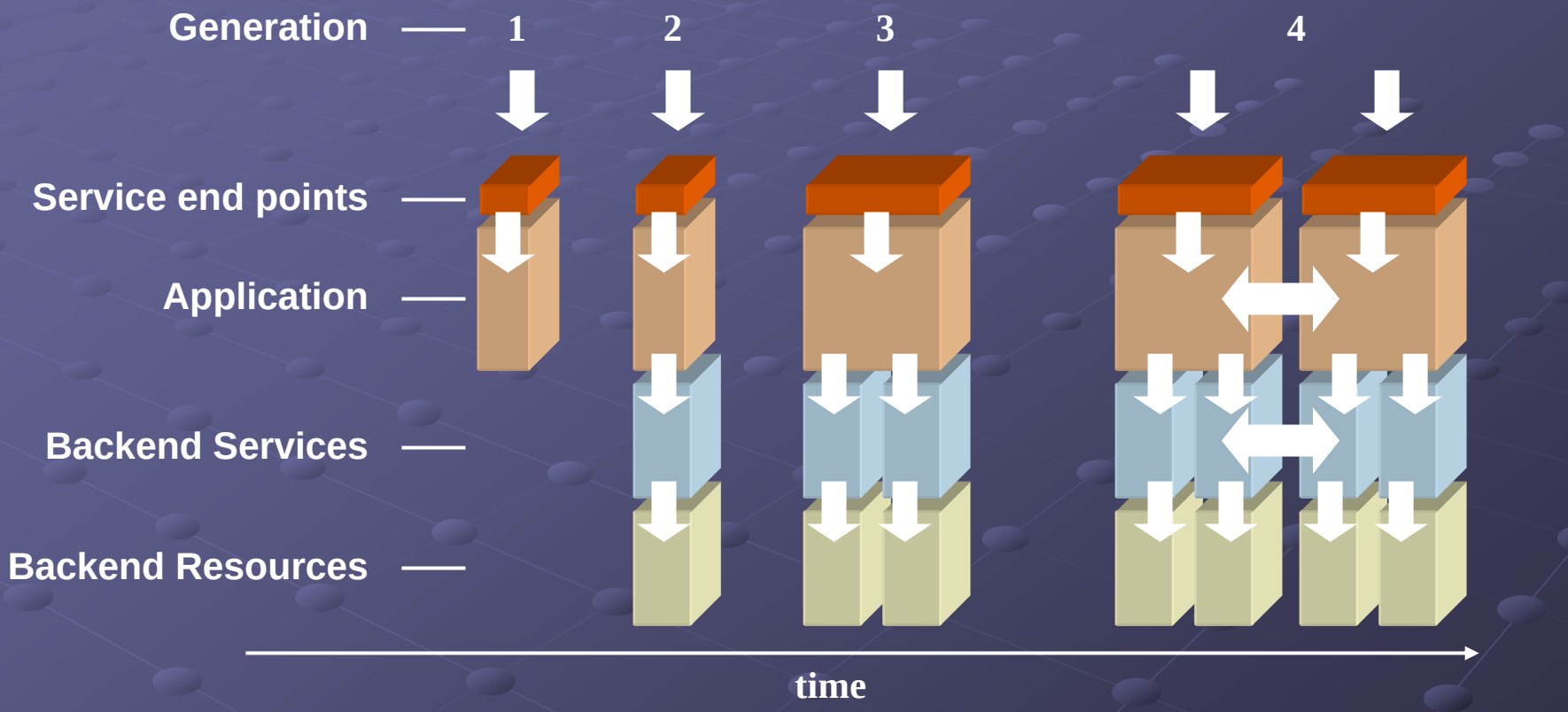
- ESBs introduce an extra transport layer when compared to regular (one-off) messaging solutions
- An ESB is a new layer of IT infrastructure in most organizations, with new demands on money, hardware, staff time, and skills
- The value of the ESB is achieved when multiple systems adopt and leverage its capabilities. This requires agreement on architecture, technology, messaging approaches, and governance across a service domain
- Well-designed ESBs can deliver service interoperability, but data integration and interoperability require agreed data models, which can prove difficult to establish

ESB Risks & Costs (continued)

- 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
- For effective implementation, ESBs require a mature information technology governance model and a well-defined enterprise IT strategy
- Technical and organizational factors can limit an ESBs scalability. Rather than spanning an entire enterprise, most ESB's support distinct 'SOA domains' with
  - A suite of related applications
  - Supported by a common pool of developers
  - Who have adopted the same methodologies and designs
  - And report up to the same executive or governance body[1]

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

Distributed System Evolution



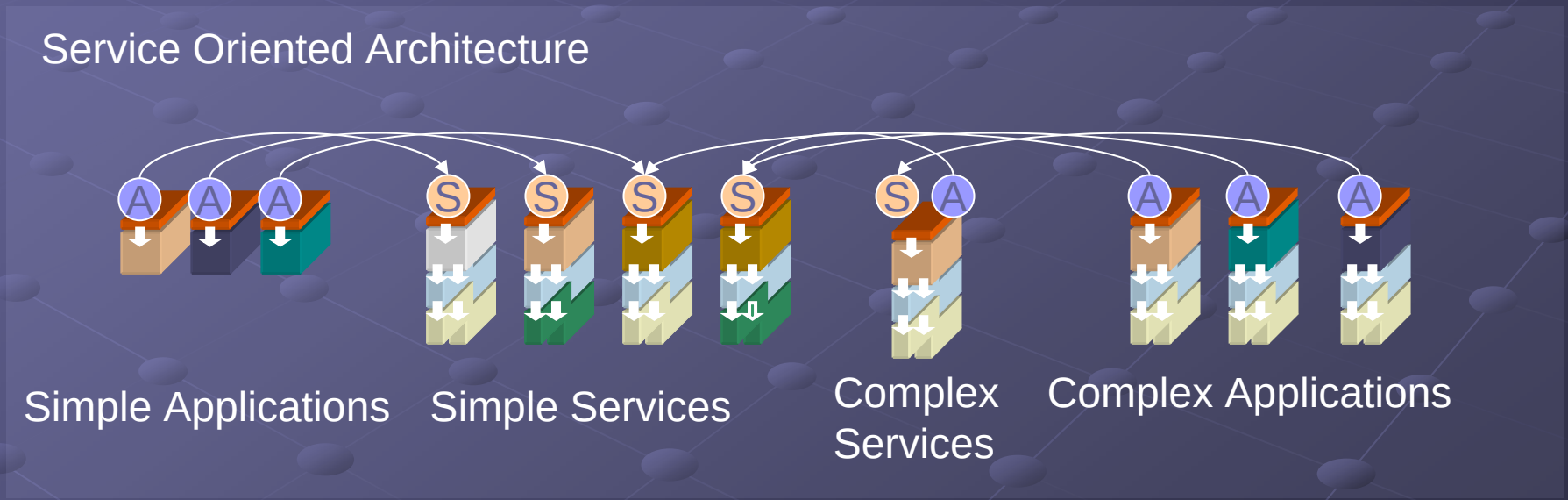
Example applications for each generation:

1: departmental web site, 2: alumni portal, 3:web mail, 4: student registration

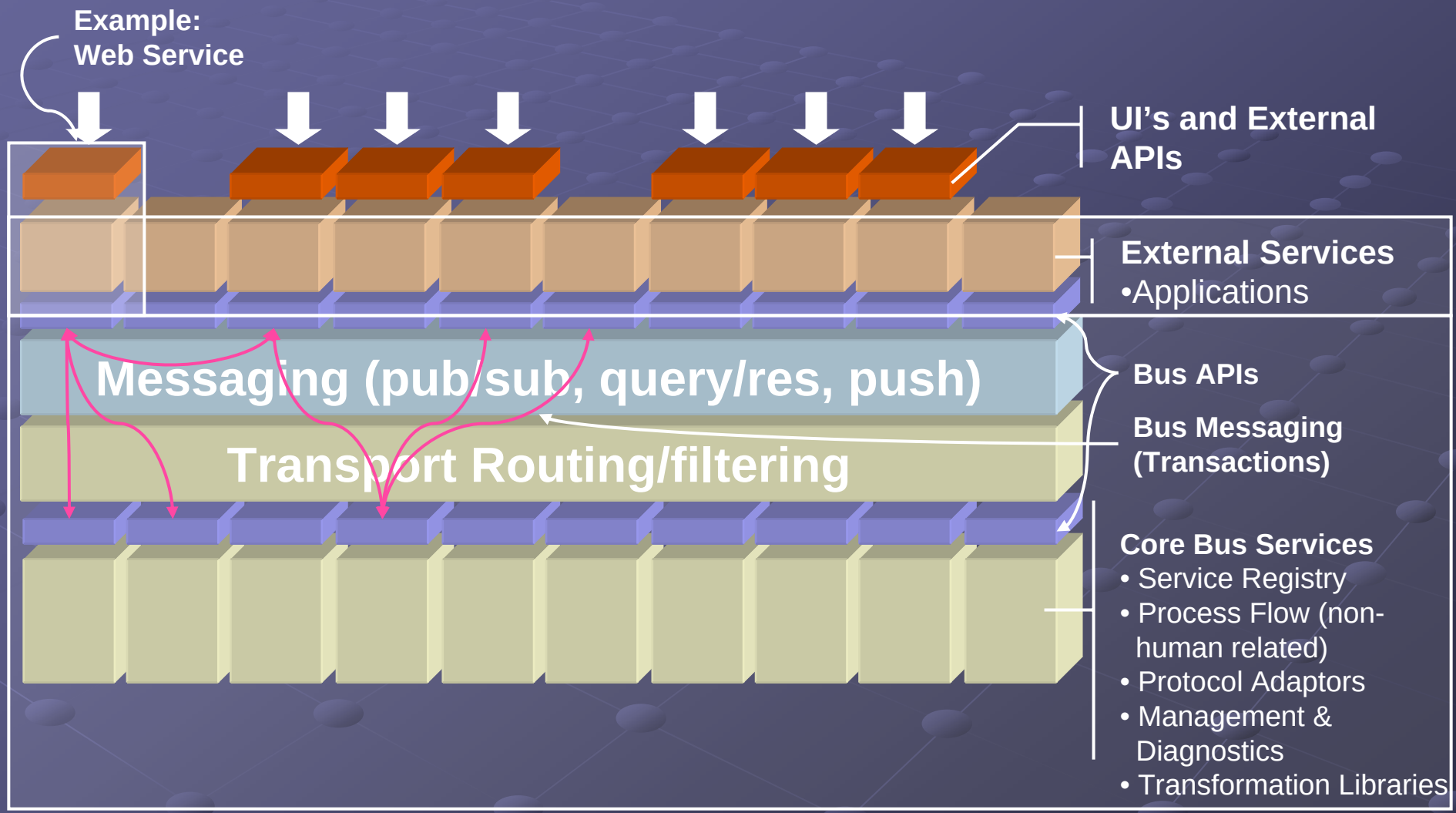
Generation 5 - SOA

Advantages of SOA

- Repurpose/reuse of code and data
- Modularization of service and application domains
- Loose coupling between service areas



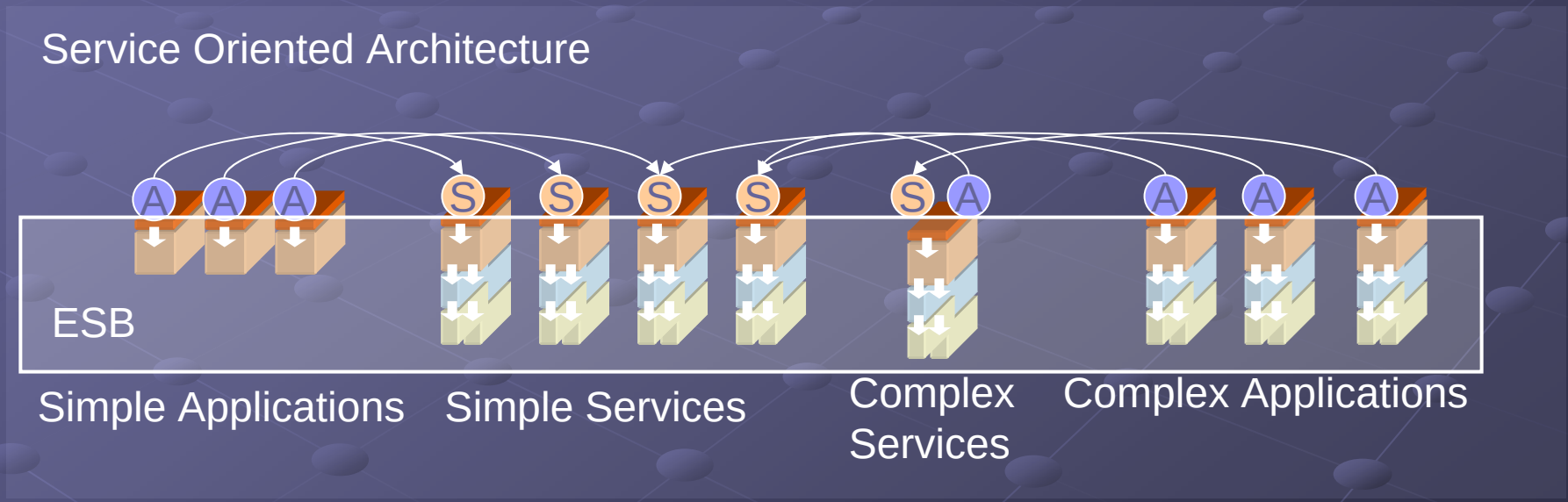
Topology of an ESB



ESB – Augmenting SOA

Achievable ESB Goals

- Improves management
- Provides services virtualization
- Provides any-to-any delivery
- Supports orchestration
- Enables transformation
- Supports asynchronous operations

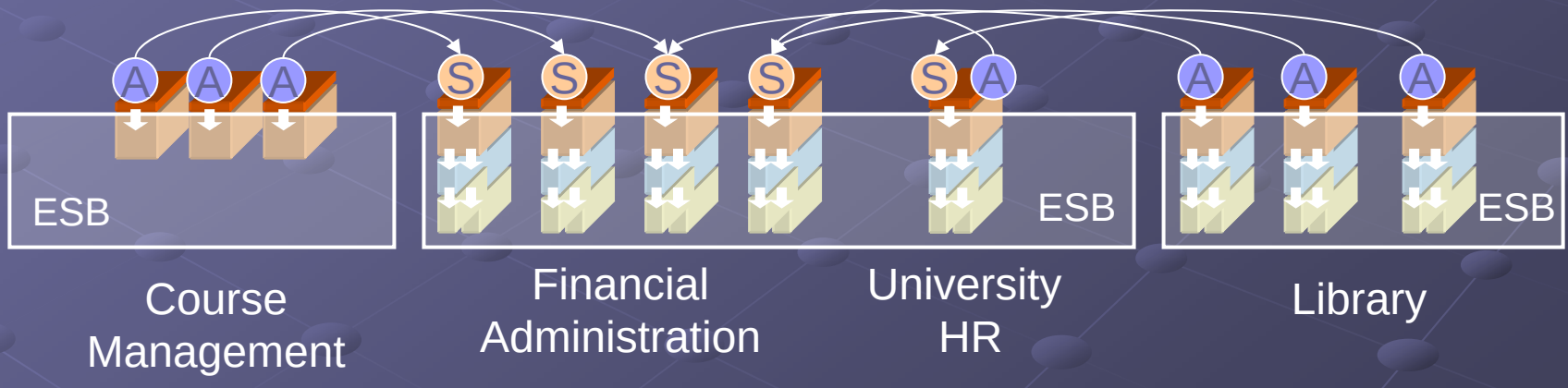


ESB Within an Higher Education Organization

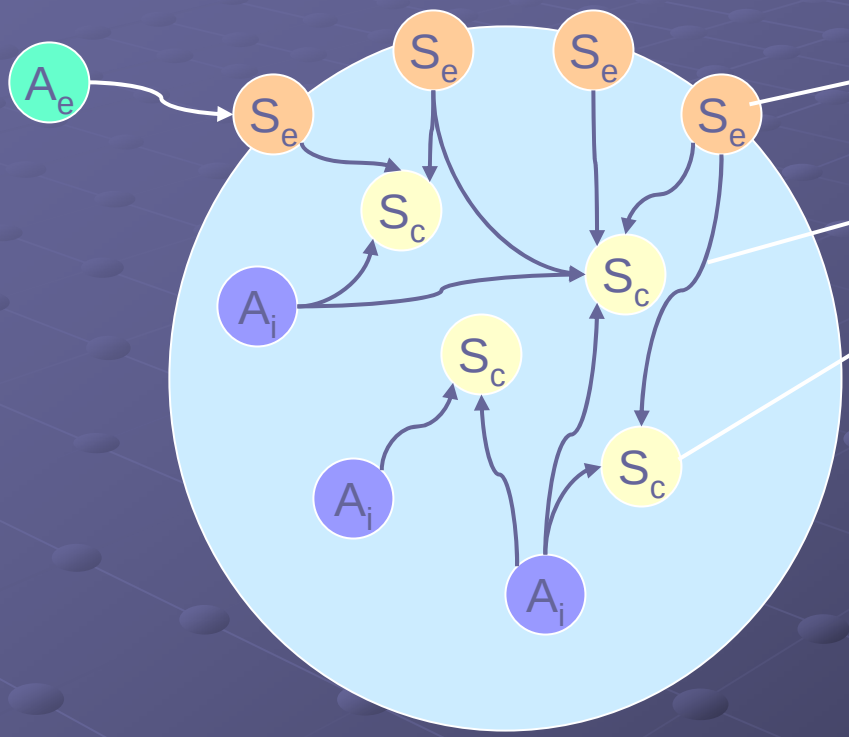
ESB Goals – can they be achieved?

- Repurpose/Reuse
  - Achievable with a single dev/admin domain
  - Challenging across multiple dev/admin domains
- Rapid Development – achievable
- Common Data and Process Policies
  - Achievable with a single dev/admin domain
  - Challenging across multiple dev/admin domains

Service Oriented Architecture



ESB Within An Enterprise



Expose Services
- Service Adaptors

- Messaging/Transactions
- Orchestration

Core Bus Services
- Process/Work Flow
  - Non-human related
- Protocol Adaptors
- Management & Diagnostics
- Transformation Libraries

External applications A_e

Service Adaptors for external services S_e

ESB service domain

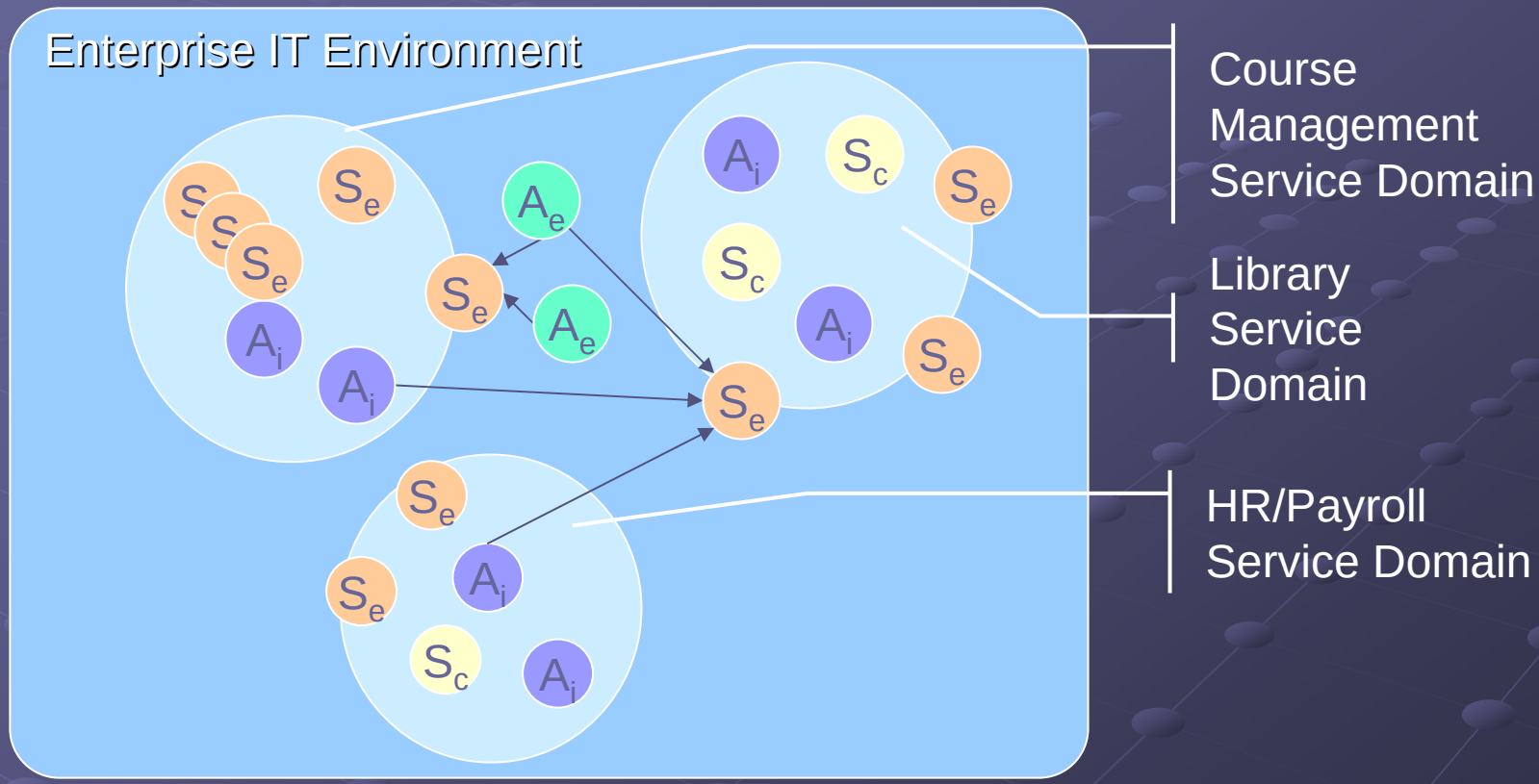
Internal ESB applications A_i

Internal ESB core services S_c

Multiple ESBs Within an Enterprise

- Common reasons for splitting the ESB
  - Separate development domains
  - Administrative control
    - Multiple campuses
    - E.g. central vs. departmental IT
  - Limits to feasible governance
  - Boundaries within institutional culture
  - Data access and management policies

Multiple ESBs in an Enterprise



External applications A_e

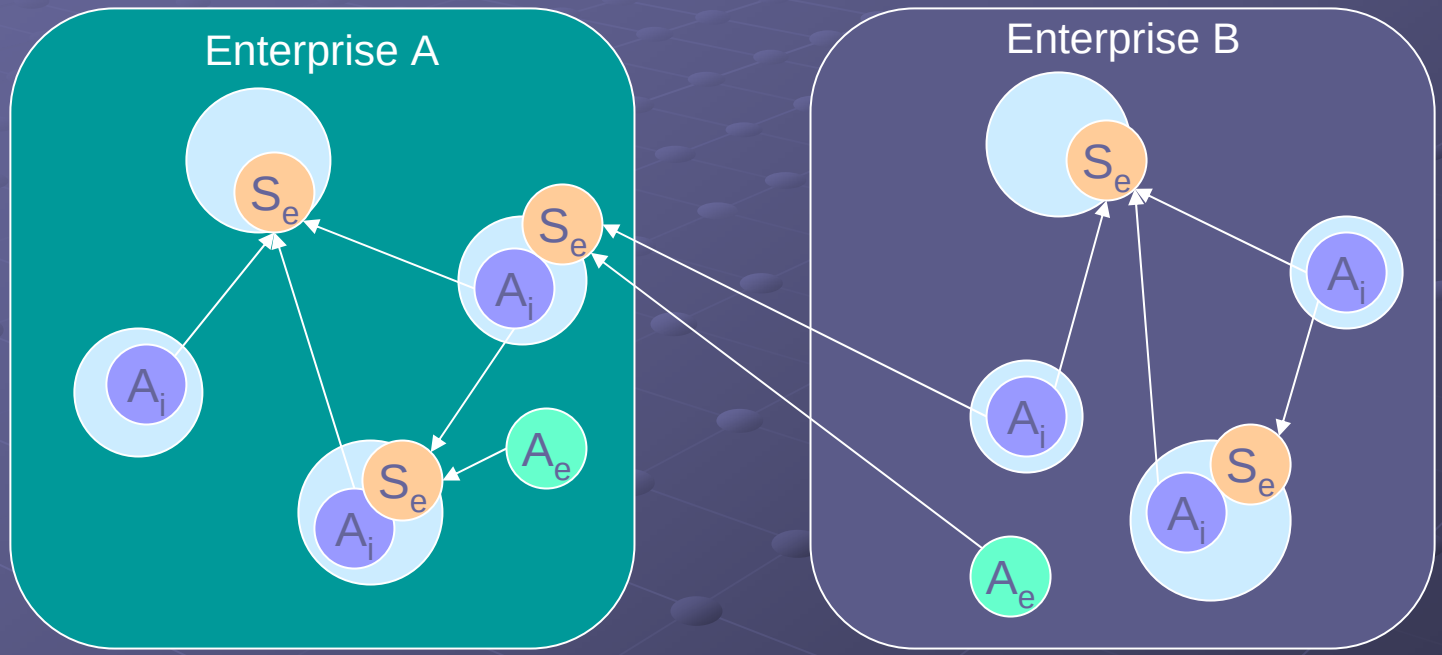
Service Adaptors for ESB external services S_e

ESB

Internal ESB applications A_i

Internal ESB core services S_c

Multiple Enterprise Domains



External applications A_e

Service Adaptors for external services S_e

ESB service domain

External ESB applications A_i

Internal ESB core services S_c

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

Use Case Criteria

- Driven from higher education
- Wide variety of applications and scenarios
- Included experts from each use case
- Requirements generated from the perspective of the application developer and technical manager

Use Cases

- Cambridge University
  - Institutional repository
  - Student identity management
- Carnegie Mellon University
  - Student information data feeds
- Cornell University
  - Fedora – “big ingest”
  - Fedora – “academic information space”

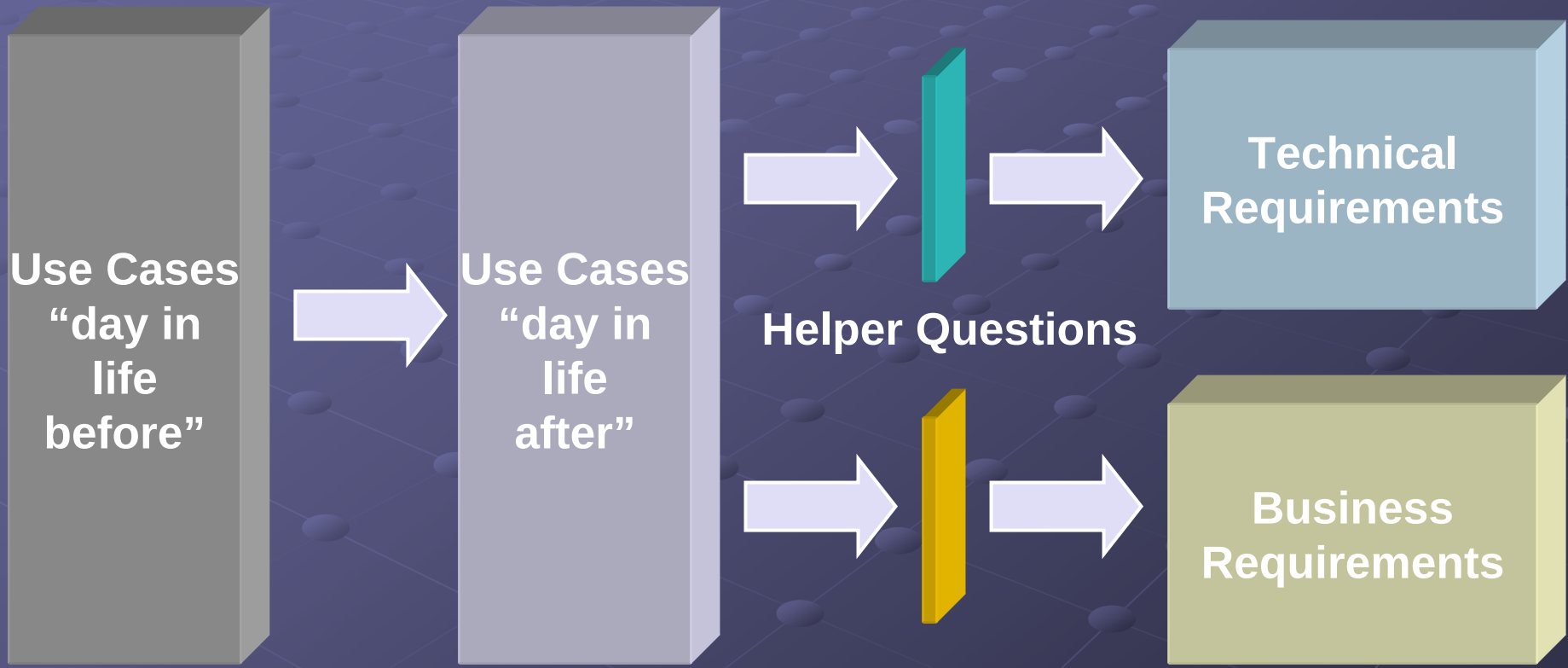
Use Cases Continued

- Indiana University
  - HR Hiring
- Kuali Student Development Team
  - Kuali Student
- Massachusetts Institute of Technology
  - Integrating a “standalone” performance review product with SAP HR using an ESB
  - Real-time data integration with departmental systems

Use Cases, continued

- Stanford University
  - Integrate course services with library, student, IdM & Infrastructure
  - Digital collections process framework
- University of Chicago
  - Interdisciplinary digital scholarship project

Requirements Process



Requirements Gathering: Before

- "Day in the life" *without* an ESB
- Describe in a narrative how you develop now
  - Detail the pain
  - What doesn't work
  - Technology, process and economic
- Define actors, owners and economic stakeholders
- Describe present outcome

Requirements Gathering: After

- "Day in the life" process *with* an ESB
- Describe in a narrative how you would develop with an (idealized) ESB
  - Detail the pleasure
  - What works
  - Technology, process and economic
- Define desired outcome

Defining An ESB for Higher ED

- Business Requirements
  - Licensing
  - Support
- Technical Requirements
  - Architecture
  - Development Platform
  - Operational Specifics

Different License Requirements

- Internal Use Only: No need to redistribute ESB source code nor any applications that are built from it
  - Some groups expressed a desire to modify/change ESB core code for internal use
- Institutional Bundling of ESB with Apps for Distribution: Need to redistribute ESB source code and applications that are built from it. Several models are possible:
  - No rights to modify App or ESB, just to use
  - No right to modify ESB, but can modify App
  - Can modify both ESB and App
- Third party redistribution after source modification
  - Of application only
  - Of both ESB and application

Show Stoppers

- What would cause the twelve academic use cases to **NOT** choose a specific vendor?
  - Little commonality among use cases
  - Top business reasons to say NO
    - Not an open source license (6/12)
    - Vendor has little experience in market (3/12)
  - Top technical reasons to say NO
    - Complex and hard to learn development environment (4/12)
    - Feature-poor or inflexible message infrastructure (3/12)

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

ESB Candidate Selection Criteria

- Best of breed
- Both open source and commercial
- Solicited opinions of use case institutions
- Industry research
- New and promising

ESB Candidates

Open Source

- Apache Synapse
- IONA
- JBoss
- Kuali Rice
- Mule
- Open EAI
- Open ESB
- ServiceMix

Commercial

- BEA
- Cape Clear
- Chain Builder
- Fiorano
- Oracle ESB
- Polar Lake
- Sonic
- WebSphere

Open Source Support Choices

- Self support
- Community based
- Work closely with the core development group
- Third party (vendor) contract support
  - 24x7 technical support
  - Custom development
  - Maintenance
  - Training

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

Use Case ESB Rating (yes/no)

Open Source

- ServiceMix (11/1)
- Mule (10/2)
- JBoss (8/4)
- Kuali Rice (6/6)
- Open EAI (6/6)
- Apache Synapse (4/8)
- IONA (4/8)
- Open ESB (3/9)

Commercial

- Cape Clear (5/7)
- Chain Builder (5/7)
- Polar Lake (5/7)
- Sonic (5/7)
- WebSphere (5/7)
- BEA (4/8)
- Oracle ESB (4/8)
- Fiorano (3/9)

Next Steps

- Bake-off of final three candidates
  - Dedicated resources and detailed plans
  - Specific goals and objectives that define the define the key success factors
- Identify (or catalyze) two real world reference implementations
  - Defining a reference process for governance with respect to higher education
- ESB for Higher Education symposium?

Outline

- Introduction and Objectives
- Definitions and State of the Practice
- Requirements Gathering Process
- Vendor Selection and Analysis
- Recommendations and Next Steps
- Reflections

Guidelines for Choosing the ESB Path

- Choose an ESB when you are
  - A software development shop (new or morphing old)
  - An owner of complex applications with many interdependencies (HR, student records, library)
  - An owner of data (student records, financial records)
- If you are not part of these three you are a candidate for an SOA, but may not need an ESB
  - Expose/consume existing services (e.g., an academic unit may access central computing student records but may not be part of Central IT's underlying ESB or legacy system infrastructure)

Guidelines for Choosing the ESB Path (continued)

- If you are distributing software that needs an ESB then pick one that has
  - Large community following
  - Simplicity over complexity
- If you need to ship an ESB with your software distribution today, keep configuration simple and plan ahead against a possible future need to move to another ESB

Internal Considerations

- Choose an ESB based on
  - Your existing/desired application servers
  - Experience with development platform
  - Skills and core competencies of development and operational staff
  - Initial and ongoing operational cost
  - Licensing issues

What to Take From This Effort

- Requirements of higher education institutions are extremely varied
- Lots of religion around technology
- Licensing issues are large
- Institutional culture plays a large role

Lessons Learned from the Study

- Deciding how to decide is much more work than deciding
- Good range of expertise about the topic
- Good collaboration support tools are essential
- Use case driven process was extremely helpful
- In an evolutionary project like this, managing scope is a crucial and ongoing challenge
- A dedicated leader/project manager is essential to drive the process

Acknowledgements

- Cambridge University
  - Ian Boston – use case author
- Carnegie Mellon University
  - Chas DiFatta – project leader, core discussion and analysis team, vendor analysis, 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

Acknowledgements, continued

- Indiana University
  - Brian McGough – use case author, core discussion and analysis team, vendor analysis
- Ithaka Harbors Inc.
  - Keith Kiser – Wiki and Email services, IT support
- Kuali Student Team
  - Parviz Dousti (representative) – use case author, core discussion and analysis team
- Massachusetts Institute of Technology
  - Scott Thorne – use case author

Acknowledgements, continued

- Mellon Foundation
  - Ira Fuchs – funding sponsor
  - Chris Mackie – solicitor of the project
- University of Chicago
  - Kaylea Hascall – vendor analysis, core discussion and analysis team
  - Chad Kainz – use case author, core discussion and analysis team

Acknowledgements, continued

- 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 – use case author, vendor analysis, core team, final report author
  - Minh Nguyen - vendor analysis
  - Mike Olive - vendor analysis