University of California, Berkeley

Name: Tamer Sakr

Do you have a wiki or web site with a lot of this info? If so, link please.

We have two distinct efforts: API management and Enterprise integration infrastructure.

The API management portal is at https://api-central.berkeley.edu. This is where API providers publish documentation, it also includes the capabilities of an API registry, active documentation and a support forum.

The integration middleware documentation is available only on an internal wiki.

What API Management Infrastructure (Middleware) do you have in place or have in place? What are the other core systems? API Management, ESB, ERP , DataIntegration, and Security(Authz/Authn/Access)

We use a cloud-based platform for API management, provided by 3Scale. This is a central campus service and is used by most API providers on campus.

(API management is a solution for managing multiple tiers of access to the API, and allows controlling how an API is published and used. In this context, an API is a full programmable interface of a system, which is different from a set of web services.)

To support system integration components, the campus offers a central integration infrastructure, including an ESB (ServiceMix / Fuse ESB), a message broker (ActiveMQ / Fuse MQ), and a cloud-based messaging service using Amazon SQS and SNS. While this infrastructure is available centrally, campus units have opted to use their local solutions for most applications, and this has proved to be more efficient.

API security is handled partially through the API management platform, and partially through a home-grown authorization system that uses internal API keys.

Picture of current state and aspirational state?

What are the "poster child" use cases you're using to build momentum for SOA?

The best use cases so far have included APIs for student information and for research data and technologies.

Do you have low-hanging fruit, or "easy wins" that you can show reasonably quick value with?

How do you evolve the culture from file based to real-time based?

The campus has largely accepted the paradigm of real time access through programmable interfaces. The adoption is expanding across major systems at varying rates. The usefulness of having accessible data in real time has become sufficiently evident to impact the culture.

Do you have design documents that you could share?

We have several sets of documentation, from high level strategies to coding recipes. Some of these are intended for a campus audience, others are more general and can be shared.

How did you build data objects? What kind of governance did you put around the objects? How do you maintain them? How do you put security and authorization around them?

How are you gathering resources - people and funding to support the effort? Where do these resources live? What skills sets do you think you need to support this?

Funding has been tricky. Overall, there's some 'common good' funding that comes from central IT, and additional project-based funding from multiple sources. This is still evolving.

What is the technical stack you are running? Why did you pick that stack? Pros and Cons of the stack. What else did you look at and didn't pick?

The campus is decentralized, and there are hybrid tech stacks in different units. The central stack includes the following technologies: Nginx (HTTP & reverse proxy server), Apache ServiceMix, Apache ActiveMQ, Apache CXF, Apache Camel, Amazon SQS, Amazon SNS, and the 3Scale platform.

What is behind your API? How do you handle security?

Why are doing this? What is your vision? What do you see as the future state?

The vision is to build a mature API practice over time. This means that the various business domains within the campus would have a rigorous data model that is represented through a full interface (API). This would be the foundation for internal integration and business process management. In addition to that, the possibility of data access through the API could/should spur innovation in terms of building new apps and technologies, and generally instill an 'open data' culture.

What benefits you have achieved in your implementation?