As the ever-changing business landscape continues to evolve, have you considered updating your major applications so that they are up to par with your future business needs?

In this essential guide, uncover what you need to know before modernizing legacy apps as well as how a service approach can support application modernization.

App design: Taking a services approach to application modernization
By Tom Nolle

According to a survey of 277 businesses conducted in the spring of 2013 by CIMI Corporation, about a third run major applications more than 10 years old, and a fifth have at least one major application that's largely unchanged for 15 years or more. Fully half of the applications reported in the survey were based on obsolete monolithic architectures or used Web services only as a front end to a large application.

Since the same survey showed 88% of CIOs believed service orientation was the single most important architectural key to a successful application, it's not surprising they thought application modernization should focus on a service approach. To make that happen, they should align application services to enterprise architecture (EA) functional/process designs. In addition, they must both think in Business Process Execution Language (BPEL) for high-level composition of services and componentize services below BPEL, based on software architecture goals.

Modern software should be broken down into components for reuse and functional agility, and made distributable for optimum resource efficiency in virtualization and the cloud. For more than a decade, this has been
accomplished by defining functional software elements as services that provide support to business operations.

The most common blunder has been failing to align services to operating processes—something EA integration can provide. At the top of an as-a-service model is the functional and process design portion of EA, which (if the work is done properly) will define the high-level services that support each critical business process.

When a functional and process design review is conducted, it will identify the highest level of function aggregation in an enterprise—the system of services that combine to help workers do something specific. Most companies find this level corresponds with the notion of an application, which is a workflow composed of a number of software services or components and a set of binding rules. Your modernization of applications begins by defining the business-driven boundaries the applications must conform to.

**Using BPEL in app design**

Application component or service binding rules are often expressed for SOA applications in BPEL, and even companies who aren’t enforcing strict SOA discipline can benefit from using BPEL-like app design of their services to help identify the way in which services and components will be grouped to create applications.

The goal in BPEL is to express business logic as workflow steering, which orders and identifies services used to create applications. As an application architect, if you think of a business process as being a set of steps, then each step is a service at the BPEL level, and that defines your first level of making a modernized application serviceable of a modernized application. Business functions decompose into applications, which decompose into BPEL services.

Below the BPEL layer, the business drivers of service and application structure diminish in comparison to the technical factors that guide service decomposition and deployment. Too much componentization of services
creates excessive overhead in workflow management that can significantly impact application response times and worker quality of experience.

It's critical to test workflow or service bus engines to see how they perform at various levels of transaction activity, for each level of componentization under consideration. In some cases, lower-level services can be directly coupled to avoid bus overhead, though this will make the application less flexible.

One factor that can offset efficiency concerns at this next layer is component reuse benefits. A large monolithic service may not be as bad as a monolithic application, but the distinction isn't very meaningful. Another factor to consider is the impact of further component creation on application lifecycle management (ALM).

Increasing the number of services per application is justified if component reuse generates a significant benefit in simplification of development, and on through the whole ALM cycle. Obviously, one factor is the extent to which component reuse is possible.

Whether component reuse creates a benefit depends on how often the components are changed. If components are largely static and combined in only a few ways, it may be smarter to leave them as multiple higher-level services than to break them down. If components are highly dynamic and used in multiple places, breaking them out can generate efficiencies in ALM.

Can is the operative word because many users don’t have ALM practices reflecting componentization well. Some of this is due to the fact that ALM tends to focus on testing against business processes, which demands it be elevated to the highest level and obscures componentization. If services are not properly designed, they can’t be assumed to work in all contexts and thus have to be tested in context for each application that uses them.

Well-configured services have properly designed APIs. This means they have an ability to define specific services, are strongly typed, have explicit error handling and feature checked data models. In addition, they have failsafe
interactions based on either RESTful processes or on processes with very clear state control.

If the service interfaces are "tight," ALM can do the following:

- fully test changed services
- check basic interface integrity on the derived applications
- be passed to production

Otherwise, a full test will be required for every application where service change is found and benefits of service reuse will be reduced.

Remember that service may mean SOA in its most general sense, but not necessarily SOAP-WS coupling. Service orientation can be created and sustained using purely RESTful interfaces, even if application architects use a BPEL-like description of high-level coupling of the business process to the application.

That description can be used to guide workflow, but can also be seen as a documentation tool to systematize the design of RESTful Web-like processes. The best result in application modernization is always created by the most structured approach.

What you need to consider before modernizing legacy apps
By Maxine Giza

Relying on poor documentation and glitzy tools, rather than the most appropriate, are just a couple of missteps that can send an application modernization project on a detour. Carefully analyzing the situation by talking to users, and being mindful of things like data migration, can help reduce the number of snags endured when modernizing legacy apps.
Communicate openly and often

Collecting user feedback is a first step in application modernization project planning, and helps architects to elicit requirements and improve legacy documentation, according to Ovum senior analyst Alan Rodger. "There can be a lack of understanding of the source system because documentation is often very poor for all the systems," he said. "If relying on [poor] documentation, there are other things likely to come out of the woodwork when they [architects] start working with the code."

By speaking with users, Rodger said it's possible to gain insight into how a system operates that wouldn't be obtained otherwise. In fact, he said, because of overturns in IT staff, experienced users might have more information than those in the IT department.

To help get mentally prepared for taking on an application modernization project, Hamala recommends architects seriously look at the most "painful" part of the application, and ask themselves several questions:

- Will the value of the application increase if modernized?
- What and how much effort will be needed?
- Are there potential risks?
- Are there areas that will generate more value than others if modernized?

Examine what is really needed

If an organization is going to modernize legacy apps, developers have to determine what the code quality is like. Rodger said it's important to look at the code and determine if all of it is being used. It's not uncommon for legacy apps, he said, to be built off redundant code.

The project management team might need to test application efficiency, but not have the in-house diagnostic tools or expertise to do those tests. Staff with dated skills will have to be trained in the new technology and language used in the new system, which according to Rodger, could introduce more time delays and risk.
Turning to an outside provider that specializes in assessing code can result in lower costs and greater efficiency. "I've seen outsources [outside consultants] be able to come in and have a fast conversation about doing modernization job, produce that kind of analysis," Rodger said.

Keep your hype shields on high when choosing tools. For example, Hamala said, an architect might think, "'We need to cope with big data, therefore the new architecture must be based on Hadoop!'" Instead of gravitating towards the latest "it" product, he said, it's best to choose a tool based on the specific situation and desired outcome.

Moving data and the information that lives with it is another step that architects need to be mindful of. That is because an application can start off looking one way and end up being entirely different upon completion. "Data migration is a huge issue," Rodger said. "It is almost as big of an exercise as migrating an application. Don't get that wrong!"

**Data migration checklist**

One of the simplest ways to overcome the challenge of data migration is to come up with a plan, advised Rodger. Planning often starts with nailing down how data is represented on the old system; Rodger said architects will have an idea of how to correctly document in the new system.

Rodger has four pieces of advice to help smoothly move information:

1. Take time to completely understand existing data and the information it represents.
2. Cull information obtained from system users.
3. Look into all modifications made to updated applications.
4. Test the migration of trail batches of data to see how they look in the new application.

While the decision to update legacy apps may always be one of contention in some organizations, having an action plan in place, and adhering to it, will help make it easier to see the project through completion.
Contents

App design: Taking a services approach to application modernization

What you need to consider before modernizing legacy apps

Free resources for technology professionals
TechTarget publishes targeted technology media that address your need for information and resources for researching products, developing strategy and making cost-effective purchase decisions. Our network of technology-specific Web sites gives you access to industry experts, independent content and analysis and the Web’s largest library of vendor-provided white papers, webcasts, podcasts, videos, virtual trade shows, research reports and more —drawing on the rich R&D resources of technology providers to address market trends, challenges and solutions. Our live events and virtual seminars give you access to vendor neutral, expert commentary and advice on the issues and challenges you face daily. Our social community IT Knowledge Exchange allows you to share real world information in real time with peers and experts.

What makes TechTarget unique?
TechTarget is squarely focused on the enterprise IT space. Our team of editors and network of industry experts provide the richest, most relevant content to IT professionals and management. We leverage the immediacy of the Web, the networking and face-to-face opportunities of events and virtual events, and the ability to interact with peers—all to create compelling and actionable information for enterprise IT professionals across all industries and markets.

Related TechTarget Websites

SearchSOA