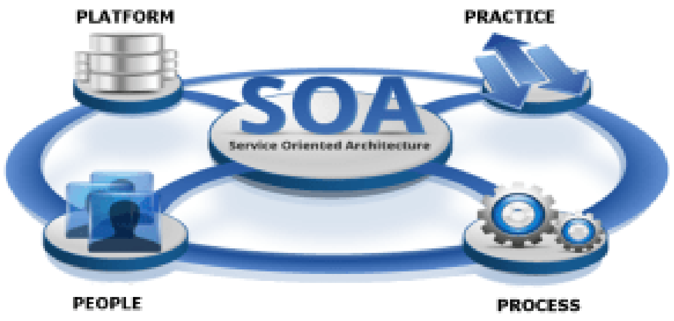
Actionable Strategies

Agile Approach to SOA



The technical benefits of a services-oriented architecture are significant and when applied properly, a SOA technology stack can yield true business benefits. However, the constraints of achieving an enterprise SOA model are often so daunting that the business is unwilling to make a full investment and wait for a complete transformation that may take years.

This paper outlines an approach to SOA that can make the business investment more attractive while lowering the technical risks associated with a shift in architecture. The following agile principles are critical to success.

- Collaboration with stakeholders
- Usable metrics to drive planning
- Tighter, managed iterations
- Frequent and early testing

Strategic Alignment

Projects with purely technical objectives are difficult for most organizations to justify and nearly impossible when budget constraints force aggressive prioritization of initiatives. By aligning projects to key strategic drivers, technology leaders achieve the dual goals of delivering quantifiable business value while creating a sustainable next generation technical infrastructure.

SOA tends to deliver the greatest value at inter-organization touchpoints such as customer or supplier interfaces. For larger enterprises, the elimination of intra-organizational seams and more agile response to customer needs are other critical business benefits. Collaboration between business and IT leaders to identify candidates where SOA can accelerate the delivery of business value is the key to obtaining and maintaining an appropriate level of funding.

Naturally, maintaining alignment is an ongoing process. Continuing to foster business buy-in is essential to prevent re-prioritization from stalling or canceling a SOA initiative.

Pragmatic Approach

The natural tendency of technologists is to create an entirely new environment and build from a greenfield. While certainly the most simple and ideal, the likelihood of an existing enterprise creating an entire architecture from scratch is unlikely. Even top-down SOA initiatives will need to encompass applications already deployed.

More often than not, a new initiative will need to fit into the existing portfolio of custom and packaged applications. It will likely consume certain centralized services and probably need to fit into some level of enterprise architecture. This is especially true to bottom-up SOA initiatives.

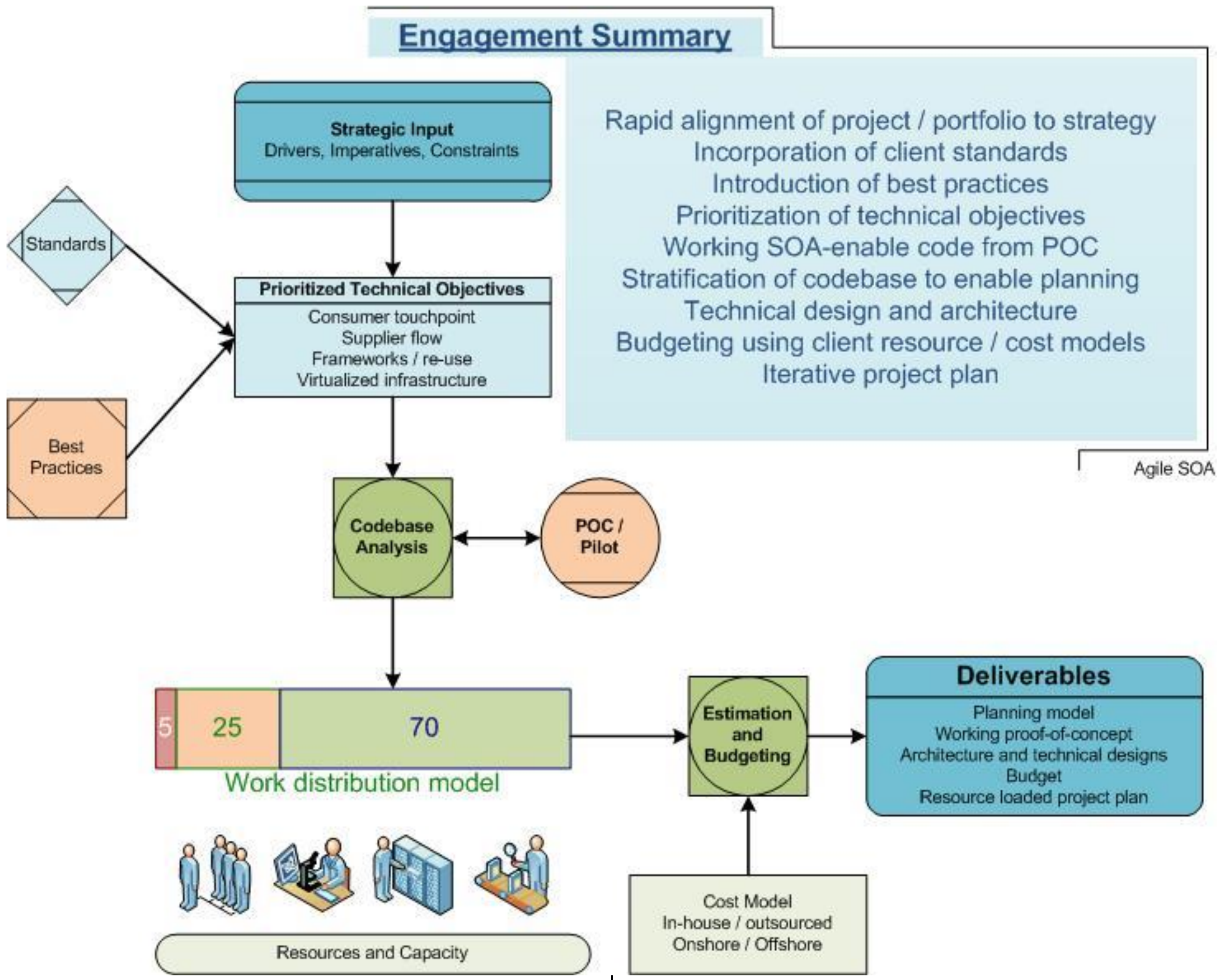
To realize a services-oriented architecture for a new initiative, a highly effective approach is to build upon an existing base of code. This code will need to be moved into a SOA and extended to deliver the new functionality. By applying agile techniques, functionality is delivered early and often with measured progress and risk. The business wins from the constant flow of value and IT moves further towards the goal of a modern and flexible architecture.

Metrics Driven Planning

Estimation and project planning can be as much art as science with the presence of new situations, many unknowns, potential for dramatic change or complexities related to large scale efforts. Usable, project-specific metrics can dramatically improve estimates and create more realistic and achievable plans. When migrating a codebase to SOA, it is important to understand the level of effort required as well as the constraints. There are two interlocking activities that allow technology leaders to calibrate their plans and estimates.

<u>**Codebase Analysis**</u>: The first activity is to determine the anticipated level of effort required to deliver a SOA. We suggest a simple stratification of the codebase into three categories.

1. The first category is the most problematic and challenging code that cannot be migrated effectively. Typically, this is the oldest, most tightly coupled and difficult to test code.


Engagement Summary
Rapid alignment of project / portfolio to strategy
Incorporation of client standards
Introduction of best practices
Prioritization of technical objectives
Working SOA-enable code from POC
Stratification of codebase to enable planning
Technical design and architecture
Budgeting using client resource / cost models
Iterative project plan
Agile SOA
Strategic Input
Drivers, Imperatives, Constraints
Standards
Best Practices
Prioritized Technical Objectives
Consumer touchpoint
Supplier flow
Frameworks / re-use
Virtualized infrastructure
Codebase Analysis
POC / Pilot
5
25
70
Work distribution model
Resources and Capacity
Estimation and Budgeting
Cost Model
In-house / outsourced
Onshore / Offshore
Deliverables
Planning model
Working proof-of-concept
Architecture and technical designs
Budget
Resource loaded project plan

2. The next category is code that can be refactored with some initial guidance from technical staff with prior experience.

3. The third category is code that can be very simply wrapped into services by virtue of loose coupling and other characteristics. This work could easily be sent offshore, leaving domain knowledgeable staff to focus on the first two categories.

Proof-of-Concept: The second major activity is a proof of concept. Ideally, the worst-case examples would be chosen both to rewrite and also to refactor. Experienced practitioners should pair off with developers who have familiarity with the domain and code. The actual durations and effort levels should be compared against initial estimates. This activity will accomplish some major objectives.

1. Proof of the approach
2. Baselines for better estimation
3. Immediate knowledge transfer into the organization
4. Assessment of the current staff's SOA skills

Estimation and Budget

Using actuals from the POC, a full project estimate can be prepared with a fair degree of confidence. By using the categorized work distribution model, the refined estimates can also drive project risk management. Note that the estimates may make the project non-viable, but it is best to know early and possibly re-scope.

The estimation and budgeting process should take into account the initial use of outside expertise to work on the more difficult categories. Moreover, there may be the potential for broad use of offshore resources to wrap the third category of code. The offshore resources may be captive or outsourced as the activity is contained to within the scope of the project and maintenance costs can be limited as noted below under automated testing.

By utilizing a stratified codebase to perform resource allocation, the most effective resource and cost model can be employed. The work distribution model has durability in that it can be applied to other Agile SOA initiatives where wrapping a codebase is an alternative.

Architecture and Design

While an iterative approach to development will require emergent design, a baseline architecture must be established. This architecture may feature core services that are initially stubs, but clearly the decisions around functionality and ownership must be addressed early in the planning process. Moreover, architectural decisions will also drive cost estimates and resourcing, especially when there are enterprise considerations and multiple organizations involved.

Iterative Delivery

One of the benefits of a services-oriented architecture is the ability to incrementally deploy services in a building block fashion. By supporting iterative delivery, SOA allows technology leaders to demonstrate business benefits early and often. Maintaining buy-in and alignment are thus more readily fostered. Clearly project governance is critical to enable the communication of benefits, acknowledgement of the value from business and reaffirmation of the mandate to proceed with the resulting commitment of resources and budget.

Quality and Test Enablement

In a loosely coupled, distributed environment not only is an investment in quality sound, but it is also essential. Because many services are not exposed to end users, traditional testing becomes expensive as both test coverage and traceability become challenging. While automated regression test coverage is still recommended, other testing is required not only for initial success, but for the ability to maintain the software without incurring excessive and accelerating costs.

Agile practitioners understand thoroughly the need for automated unit testing. These benefits have been well documented. Additionally, we recommend test harnessing around services to ensure that the interaction of services works as the services are linked, extended and modified. Knowledge of the orchestration of processes via services requires test generation that includes domain knowledge, coding depth and quality experience.

While such testing investments seem high, the ability to rapidly deploy new functionality based on services is predicated by the ability to rapidly test and promote the functionality. If integration and regression testing are performed manually with long cycles, any agility in deployment is lost. Costs also quickly become prohibitive as the inventory of services grows.

Anecdotally, our experience with loosely coupled cross organizational SOA applications is that "hot spots" quickly emerge. Testing takes longer with each iteration and often becomes single threaded through those parts of the organization that do not adequately wrap the services in orchestrated test harnesses. Correspondingly, when test frameworks are well implemented rapid change without quality degradation is the result.

As SOA environments mature and the network of interconnected services grows, the ability to test becomes critical to keep maintenance costs low. Likewise, the ability to quickly keep pace with needed updates and changes is predicated on quick turnaround in testing. This is only possible with automated testing at differing levels.

Conclusion

While it is ideal to build technology solutions free from the constraints of existing architectures and applications, the reality in most existing enterprises is that too much investment has been made in existing portfolios. To move to a services-oriented architecture, technologists must begin by aligning to key business problems on the executive agenda. By securing funding and buy-in, they can then define a pragmatic path to building a SOA while delivering a stream of business value. Appropriate planning and measuring, combined with suitable levels of expertise and investment in testing will ensure that the value creation is sustainable. These steps will also ensure that costs do not accelerate over time and that agility and pace of delivery can actually increase as the organization grows more proficient in the use of SOA.