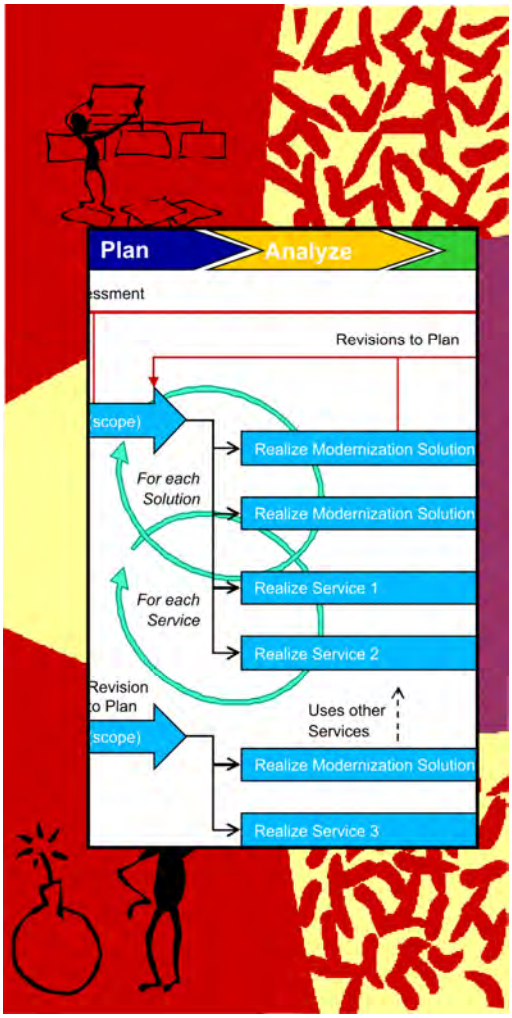
CBDI Journal



Practice Guide

The Agile Application Modernization Project (extract from)

In a previous report we introduced the Application Modernization process decomposing it into Disciplines, Process Unit and Tasks. In this report, we discuss an agile project structure and organization and provide a detailed breakdown of the Application Modernization process in terms of Project Phases and Work Packages. This approach to application modernization will allow an escalation from a sponsored modernization effort to an enterprise SOA effort over time. It can be viewed as the pragmatic middle ground between a difficult to motivate enterprise level SOA and successive SOA projects that will inevitably lead to service anarchy.

By Lawrence Wilkes and Denzil Wasson

Originally published in the January 2010 edition of the CBDI Journal

Independent Guidance *for* Service Architecture and Engineering

The Agile Application Modernization Project

In a previous report we introduced the Application Modernization process decomposing it into Disciplines, Process Unit and Tasks. In this report, we discuss an agile project structure and organization and provide a detailed breakdown of the Application Modernization process in terms of Project Phases and Work Packages, starting with the Assess and Plan phases.

This approach to application modernization will allow an escalation from a solution specific modernization effort to an enterprise SOA effort over time. It can be viewed as the pragmatic middle ground between a difficult to motivate enterprise level SOA and successive SOA projects that will inevitably lead to service anarchy.

Lawrence Wilkes and Denzil Wasson

Introduction

In CBDI-SAE we use Disciplines as a way of separating out the different capabilities required by an organization. For example the capabilities required to deliver the Service Architecture – such as skills or tools - will be different to that required to deliver Service Implementations. Disciplines are also responsible for one or more key deliverables in Application Modernization and SOA. Application Modernization Planning produces the Application Modernization Plan; Service Oriented Architecture & Design produces the Service Portfolio Plan, and so on.

The process decomposition of Disciplines, Process Units and Tasks forms the basic 'building blocks' from which various types of project plans can be assembled to meet different needs. Think of Disciplines as the 'service providers' to an IT process.

Most projects will be focused on delivering a solution to a business requirement. The project will be tasked with understanding the requirement, mapping out the architecture, provisioning and implementing the various components and services, and finally assembling and deploying the solution. Consequently it will be using the 'services' provided by a number of Disciplines.

However, not all Application Modernization projects are exactly the same.

- A project that deals with a larger unit of scope may require iteration through some activities, whereas a smaller one may not.
- One project may have a strategic focus that requires significant investment in planning and architecture, whereas another may be more tactical and start with an existing architecture that needs little refinement.
- Most projects should be business-driven in response to new business requirements, whereas some may be more IT-driven, for example retiring an obsolete platform, and consequently have less need to identify business improvements or build business models.

Mapping the process decomposition to appropriate project phases and into work packages enables us to explore these different options and to construct various types of project plans.

In this report we use a generic project lifecycle. We recognize that readers may follow frameworks such as RUP or various agile approaches that provide their own lifecycle phases. Mapping to these should be straightforward.

In context with Application Modernization, the phases cover:

- Assess: Obtain initial business requirements and assess current system status, structure, issues and opportunities, functional backlog and agility potential.
- Plan: Produce plans and architectures including the Business improvement plan, Service and Solution architectures (at an outline level), the Application modernization plan, and any Reference Architecture that will underpin the work.
- Analyze: Produce detailed To-Be models and architectures as well as models of current systems, for the scope of the modernization project.
- Deliver: Realize the plan by provisioning new or reengineering existing assets, and assembling and deploying the solution.
- Evolve: Transition (go live) to the new business and solution, then measure/monitor and produce refinements (to the assessment, plans or realizations).

Agile Approach

For reasons discussed in Business Requirements for Application Modernization[3], an agile approach is required. The approach described in this report addresses agility requirements via a number of mechanisms:

- Scoping and decomposing activities into manageable Work Packages.
- Iteration through the various Work Packages leading to revisions at all levels to fine-tune the next iteration.
    - Continuous improvement of the assessment and plan
- Separation of Solution and Service Architecture, and Solution and Service Delivery, to ensure tight focus of activities, and to deliver a layered Service Architecture that is not too tightly bound to a single solution and is inherently flexible.
- The incremental delivery of an inventory of shared services and solution components that reduce time and effort in subsequent projects.
- Agile methods used within Work Package (or within phase, depending on how broad the scope is)
    - With necessary planning and architecture to provide some coherence across and within the Work Packages

Work Packages

Figure 1 shows the identification of several Work Packages that roughly follow the project phases. In a large project there may be the execution of:

- For each Business Improvement, one Assessment Work Package, that leads to the identification of . . .

- Several Planning Work Packages for each unit of scope. In turn these identify . . .
- Several Realization (Analyze and Deliver) Work Packages that deliver:
    - each solution, or significant solution component
    - each service, or part of (e.g. certain operations)
- And a Deployment Work Package for each of those
- Finally, there is one Operations Work Package and one Measure Work Package for the Plan.

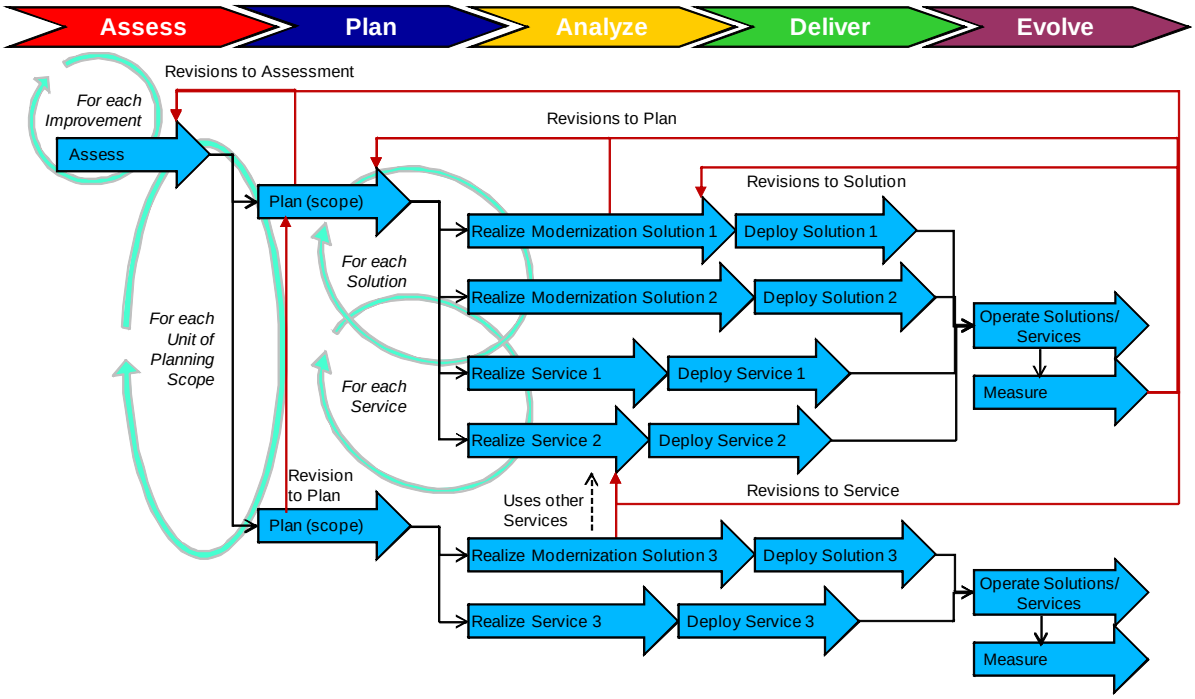


Figure 1 - Work Packages and Iteration

As shown in Figure 1, revisions may be surfaced at the end of each of these Work Packages. The revisions may lead to changes in the assessment, the plan, or to the solution itself.

For example, at the Plan phase architectures are only delivered to an outline level. As a consequence of the detailing of the architectures in the Analyze and Deliver phases it is therefore likely that some refinement will be necessary to the assumptions that we made in the assessment and planning. Making refinements to the architectures and plans ensures that subsequent projects will be better scoped, and can start with a higher level of precision in the outline architecture that is their starting point.

Figure 1 also illustrates that a Solution may use Services already delivered in previous projects. Clearly not all Services and Solution Components are necessarily shared. Many will be solution specific. However, correctly identifying the opportunities for shared Services and Solution Components is important in ensuring agility and reduction in effort.

Considering the Analyze and Deliver stages in more detail, figure 2 shows how further iteration and refinement takes place. Though we do not advocate any particular agile approach, figure 2 shows a SCRUM-like approach to realization.

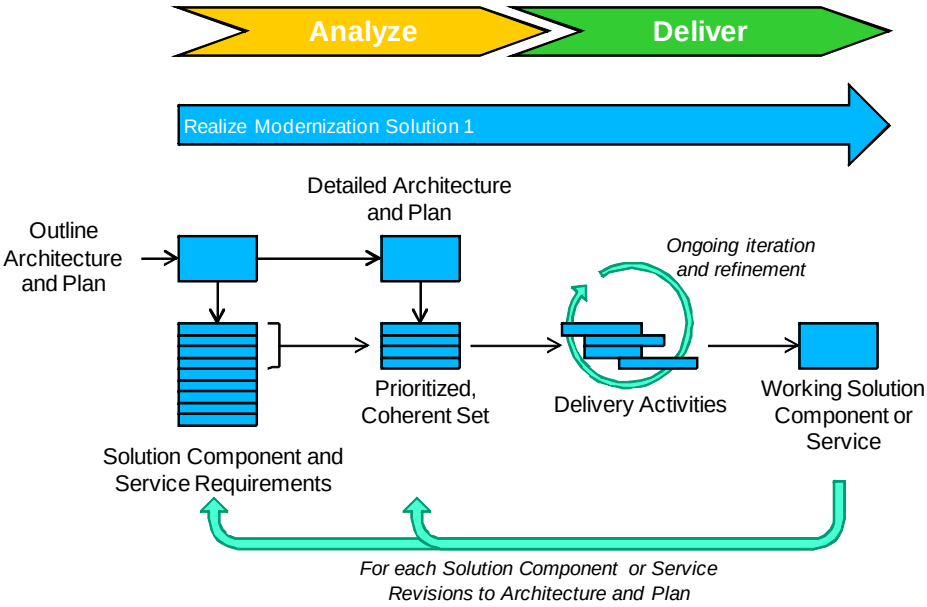


Figure 2 – Agile Approach to Realization

The complete set of Solution Component and Service requirements is prioritized into a coherent set, reflecting a coarse-grained solution component or service (from a functionality point of view – the software may be further componentized). The architecture and plan is then detailed for this unit of scope.

Disciplines

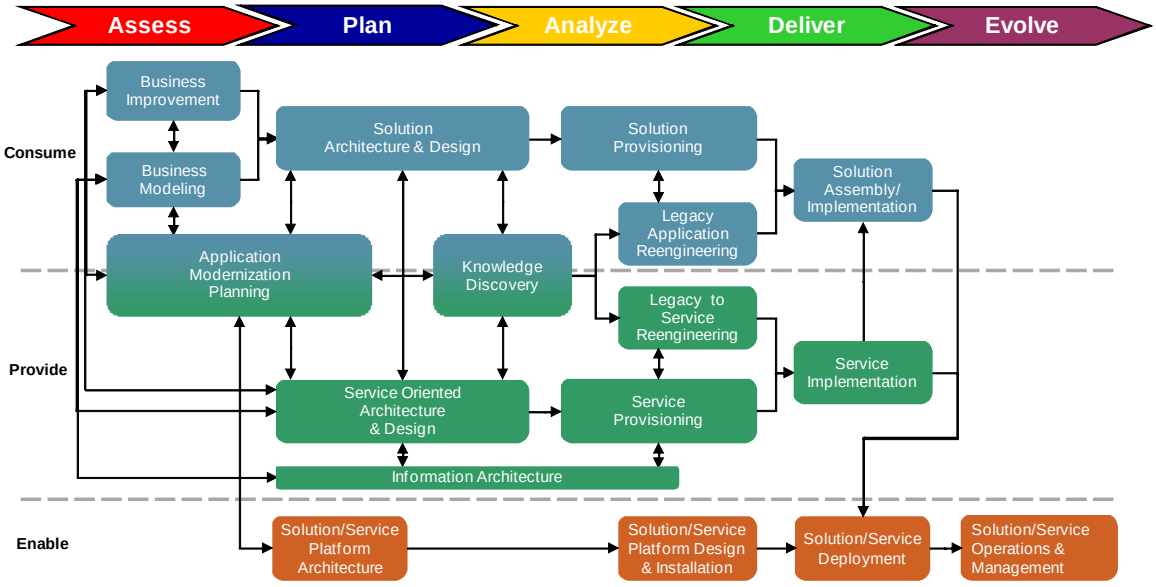


Figure 3 – High Level Mapping of CBDI-SAE Disciplines to Project Lifecycle Phases

Figure 3 provides a high level mapping of CBDI-SAE Disciplines to project lifecycle phases. As discussed some Disciplines span phases, reflecting that architectures for example are produced at an outline level in the Plan phase, but then detailed in the Analyze phases.

Scoping

Correct scoping is crucial to the success of any iterative method and becomes more so when the complexity of service dependencies are thrown into the mix. The SAE modernization approach calls for a philosophy of "Just enough and Just in time". For example, even though an outline understanding of the To-Be and As-Is architecture is necessary for assessment and planning in order to determine scope and impact, the key is to keep this at a high level; since subsequent phases and iterations will deliver the required detail

In Service Portfolio Planning we advise scoping the Service Architecture based on business domains[4]. The coupling between domains should be low, facilitating concurrent activity to proceed in relative isolation. New discoveries about the coupling across domains may lead to refinement of the architecture for domains that have already been analyzed.

However, in Application Modernization, the project scope of a business improvement is more likely to be determined by the impact on the current systems or the business processes requiring modernization. As illustrated in Figure 3, more often than not these will span multiple business domains. In a well-ordered enterprise, one might hope that Applications are aligned with business domains. However, an ERP application is a prime example of where this is typically untrue.

This is not to say that the Service Architecture should be scoped by a current system or business process, and business domains ignored. In order to achieve future agility the To-Be architecture should not be constrained by only considering the scope of the As-Is. However a pragmatic balance needs to be achieved so that the modernization effort is not turned into an enterprise level SOA effort, where the requirements of every stakeholder in a business domain are analyzed in-depth.

Instead what we suggest is that (where appropriate) a portion of the solution being modernized participates in one or more domains and that we are producing an increment of service capability for those domains. From an agile modernization perspective we explicitly use this stepwise refinement technique and constrain the scope of the service architecture to that slice of the business domain that is under modernization.

So for example if we are modifying an application that deals with Customers and we know that Customer is a future domain service, we will initially scope the Customer service to only the known requirement (i.e. the piece under modernization).Critical to success with this approach is that we deliver the partial Customer service explicitly as an SOA deliverable and not as a dedicated solution component.

There are a few other principles at work here that will allow us to grow this tactical service into strategic service over time:

- we actually have funding for the current increment scope.

- this particular scope has been prioritized by the business and is our first view of Customer.
- we believe, based on a high level Business domain model and the Service Portfolio Plan, that Customer is worth factoring out as a service.
- since all the deliverables of customer will be Service Oriented, the end of this increment will give us a starting point for a Customer service from an enterprise perspective.
- we know we are going to iterate and even though that iteration may drive necessary changes into the current service, we do know that the current service does support part of the business and we have also made that portion of the architecture more agile by having it loosely coupled and producing SOA deliverables for it.
- the service is still analyzed and designed in a 'top down', contract-first approach, based on the Business Type Model, and not 'bottom' up from the implementation.

Determining project scope can therefore be complex. In project planning it is necessary to understand where responsibilities for different business domains lie, and how complete understanding of a business domain might evolve iteratively across several projects. Does a project that is initiated to modernize current system B in figure 4 have responsibility to fully analyze a business domain that it only partially encroaches upon? If not, which project has? Or does the project perform just enough analysis to cover those aspects of the business domain that are relevant? Even though, controlling that is in itself very difficult. How do you know you have done 'just enough' correctly without first going beyond it?

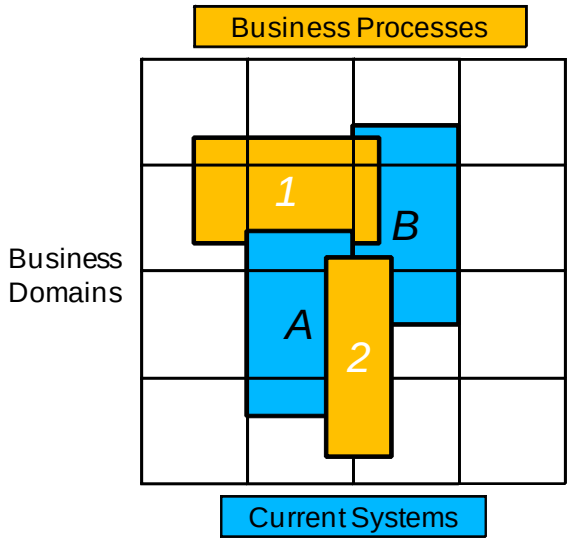


Figure 4 - Identifying Units of Scope

When scoping, it is necessary to achieve the right balance. Constraining scope to a manageable size is key to on-time delivery. But at the same time, there needs to be sufficient scope to enable agility, and ensure consistency of the architecture across several projects.

This is why even within the scope of an Application Modernization project it is still desirable to separate the Service and Solution architectures, and still build the Service

Architecture on a business domain basis, so that it is not hardwired to a single business process or solution.

Project Phase Breakdown

In the following sections we provide a detailed breakdown of the process for each project phase. The process shown assumes a strategic, business-driven Application Modernization scenario.

Defining a broad and deep process like this enables us to show most of the Process Units and deliverables that are required in Application Modernization.

Your project plan will depend on your requirements and your particular scenario. It may well be a subset of this strategic, business-driven scenario. Your units of scope and number of iterations will inevitably vary.

However, that said, the basic process will always apply. Often it is not the case that Process Units are omitted, but more a question of the degree of effort to which they are performed. A tactical project may assume minimal requirements to develop a Service Architecture. But *some* architecture is still surely required.

This is why the initial planning and scoping is so important. Solution project leaders may be reluctant to invest in architecture because of uncertain scope and the fear of introducing external factors into the discussion that are beyond their control. However, the result, as discussed in Business Requirements for Application Modernization, may be a lack of agility, or lack of consistency, or duplicated effort.

(continued in journal)

About CBDI

CBDI Forum is the Everware-CBDI research capability and portal providing independent guidance on best practice in service oriented architecture and application modernization.

Working with F5000 enterprises and governments the CBDI Research Team is progressively developing structured methodology and reference architecture for all aspects of SOA.

CBDI Products

The CBDI Journal is freely available to registered members. Published quarterly, it provides in-depth treatment of key practice issues for all roles and disciplines involved in planning, architecting, managing and delivering business solutions.

Visit www.cbdiforum.com to register.

Platinum subscription – A CBDI Forum subscription provides an enterprise or individual with access to the CBDI-SAE Knowledgebase for SOA delivering ongoing practice research, guidance materials, eLearning, tools, templates and other resources.

Everware-CBDI Services

At Everware-CBDI we enable large enterprises and governments to become more agile by modernizing their business systems. We have repeatable processes, resources, tools and knowledge-based products that enable enterprises to transform their current applications in an efficient, low risk manner, into an optimized service-based solutions portfolio that supports continuous, rapid and low cost evolution. Our consulting services range from providing practices and independent governance to architecture development, solution delivery and service engineering.

Contact

To find out more, and to discuss your requirements visit www.everware-cbdi.com or call

USA and Americas: 703-246-0000 or 888-383-7927 (USA)

Europe, Middle East, Africa, Asia, and Australasia: Telephone: +353 (0)28 38073 (Ireland)

www.everware-cbdi.com