White Paper: Modernization with Service Architecture & Engineering and the Agile Service Factory

The urgent need to embrace digital systems strategies is commonly seen as an existential challenge. The aims and objectives of modernizing projects or programs are highly likely to prioritize creating inherently agile business and IT capabilities, able to respond to unpredictable requirements.

Contents

Introduction

Service Architecture & Engineering (SAE^TM^) provides a coherent reference framework (model, architecture, process and service factory) for enterprise scale software services. Many enterprises adopting SAE are using it as the backplane for modernization of existing systems. This paper provides introductory guidance on SAE in support of core enterprise modernization projects.

Modernization Context

Why are the existing systems being modernized?

What are the goals of modernization?

Conventionally modernization has focused on “application” modernization. Frequently this has focused on platform and technology issues caused by end of life or support issues triggering compelling events that must be responded to. In many cases however, although modernization is clearly required because of end of life skills, excessive complexity, cost etc. there is no compelling case for action. In today’s world modernization usually has a very different trigger - the urgent need to embrace digital systems strategies that enable response to profound changes in business models and markets that often represent existential challenges.

Approaches to modernization are therefore undergoing fundamental change. Application modernization projects commonly adopted a “baseline” strategy; delivering new systems with new technology and improved application architecture but providing exactly the same functionality as the existing systems.

In responding to digital systems challenges frequently the solution requirements are uncertain. The business model is undergoing change, perhaps driven by external forces, but the only certainty is that the solutions will need to be able to respond to as yet unforeseen needs. The aims and objectives of modernization projects or programs are highly likely to prioritize creating inherently agile business and IT capabilities, enabling rapid response to unpredictable requirements.

Consequently, today’s modernization projects will usually have radically different aims and objectives to application modernization. These might include:

- Dramatic reduction in change cycle time
- Dramatic reduction in cost of change
- Significant reduction in operational cost
- Ability to adopt new technologies without major functional reengineering
- Transparency of (regulatory or legislative) compliance
- Increased quality of delivered products and services
- Organizational neutrality or ease of cross ecosystem operations
- Minimal organizational impact of modernization

- Ability to respond to continuous organizational change (M&A, divestiture in part or whole, etc.)
- Process flexibility, able to support multiple concurrent process models, batch, realtime, event driven etc.
- Ease of introducing new (technology and business) channels
- Ease of introducing new products and services
- Enable certain business changes to be effected by non IT resources

Given the broad nature of the above list it might be appropriate to cease referring to “application modernization”. A more suitable term might be enterprise modernization, encapsulating business and IT concerns.

Modernization Strategy

What does modernization mean for the enterprise specific situation?

The SAE framework including the Agile Service Factory is an approach that directly and indirectly supports many of the above modernization aims and objectives. Core principles that are integral to SAE include

- formal reference architecture
- separation of concerns
- rigorous specification of business services and rules independent of implementation
- everything delivered as a service
- design by contract
- automation of infrastructure and common code through service factory concepts
- late (generation time) bound technology
- design by exception
- scalable Agile delivery process
- continuous modernization process

These are foundational principles that establish and crucially maintain a high level of modularity, consistency, flexibility, productivity and quality that are fundamental to the inherently agile business. Typically, users of SAE will take this baseline framework and customize and extend to refine particular areas of need in a process that involves all stakeholders, business and IT. The “business” goals for modernization should be articulated and mapped to the modernization strategy, and principles further developed specific for the enterprise needs. Table 1 provides an example.

	Elements of Modernization Strategy							
Business Modernization Goals	Everything is a Service	Separation of data and application	Implementation Independent Service Specification (SAE)	Implementation Independent Rules Specification (DM)	Event Driven Processes (EDA)	Core patterns delivered by Service Factory	Design by Exception; Minimize custom coding	Standards based interoperation
Implement regulatory changes at last minute			H	H				
Improve Product/Service cost curve trend		H						
Minimal organizational impact				H	H		H	
Enable future change be addressable by non-IT resources				H	H			
Support new product introduction	H	H	H	H	H	H	H	H
Align Logical and Implemented Process			H	H	H			
Reduce cost of future change							H	H
Absolute confidence of current rules executed	H	H		H	H	H		
Introduce new business channel/partner	H	H		H	H			H

Table 1: Example Mapping of Business Modernization Goals and Modernization Strategy

Transition Strategy & Architecture

The reason most large-scale modernization projects are never completed is because transition has not been thoroughly planned.

Modernizing one or more core business systems supporting a major enterprise is always viewed as significant risk. The transition strategy is frequently part of larger efforts that includes data rationalization and clean-up; portfolio rationalization addressing long standing issues and inefficiencies resulting from mergers and acquisitions over many years; process improvement; customer service/product contract rationalization; new product introduction; etc.

Frequently therefore, there will need to be consideration of multiple applications and their dependencies, multiple data domains and a staged transition strategy that introduces capabilities on a progressive basis that minimizes risk to business continuity.

Risk minimization considerations typically include two dimensions of componentization:

a) Componentization of capabilities; identification of modules that can establish integrity units that minimize the level of integration necessary between old and new systems.
b) Separation and phasing by data domains; for example, introducing change for certain types of product, service, type of customer or geography.

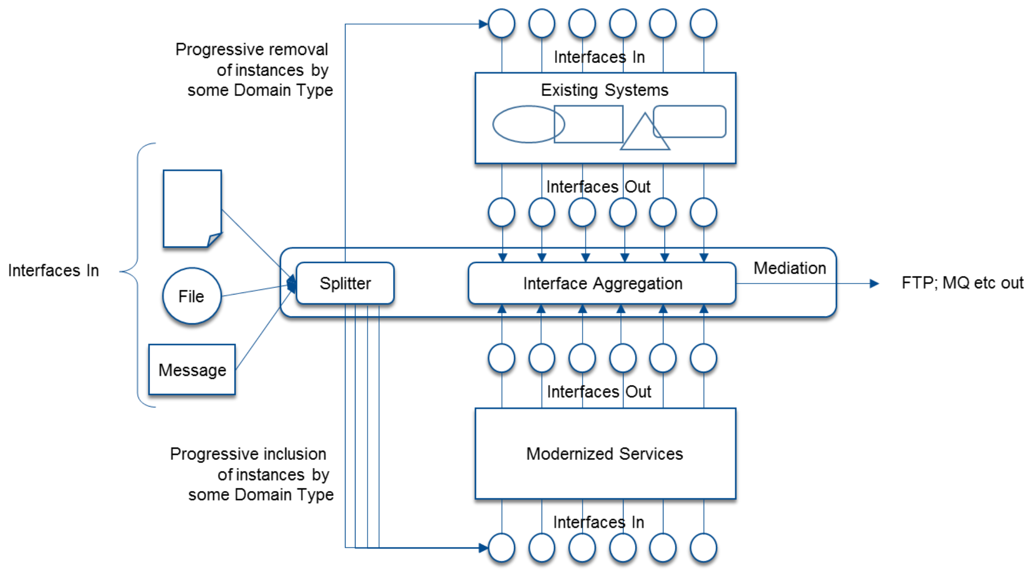

Figure 1: Example Transition Architecture

Figure 1 illustrates an example of a transition architecture using the splitter aggregator pattern which may be applicable to both types of componentization strategy, in which incoming transactions of various types and behaviors are split and diverted to old or new systems or components by some type of domain characteristic as discussed above, and then aggregated in the outbound channels to provide business continuity and transparency for system users. This is an example illustrating how an effective transition architecture is often necessary to allow progressive change and minimum risk in highly complex portfolios, leveraging the To-Be service architecture.

Establishing the transition strategy and architecture is therefore a critical first step that will strongly influence program and project planning.

Solution Architecture and Modernization Plan

A defined solution architecture is required to translate overall program goals into implementable solutions that work smoothly and consistently together, and in the process facilitate traceability and governance of the business goals as well as increasing program productivity, quality and effectiveness.

Solution architecture is a key level of detail that articulates how the To-Be solutions will collaborate at portfolio level and provides guidance for detailed design. The As-Is architecture, see Table 2 below, establishes a baseline of the current capabilities and provides key inputs into the To-Be architecture. These inputs guide capability boundaries, metrics and service requirements that are common to old and new capabilities, aspects that will be reused such as relevant non-functional requirements, common components, key patterns, utilities etc. Core functionality discovery provides guidance on harvesting opportunities.

Perspective	View	Content
As-Is	Core Functionality Discovery	Abstract As-Is architecture into logical view
As-Is	Integration Discovery	Validate existing inventory of integration points
As-Is	Implementation View	Identify design & implementation standards / patterns used; Identify design solutions implemented for: Security (authentication & authorization); Exception handling; Auditing / Logging; Code/Reference/Look up tables; Identifiers; Concurrency control; Integration; Batch; UI; Reporting; Persistence; Session state; Re-use; Other
As-Is	Quality	Reference to any existing QA standards or test assets Reference to any compliance requirements, Reference to types of legislative compliance requirements
As-Is	Non Functional	Service levels Sizing and performance Schedule constraints
As-Is	Metrics	Counts and complexities including mapping of perceived patterns to pattern instances Metrics for planning purposes

Table 2: As-Is Solution Architecture Content

The To-Be architecture, see Table 3 below, defines the modernized solution architecture views for both the transition and business operational capabilities with mapping of how the business goals will be met in the modernized solutions.

Perspective	View	Content
To-Be	Business	Business motivation for modernization Business agility requirements Business delta between AS-IS and TO-BE - Gap Analysis, Business process improvements. High level Use Case and Data models
To-Be	Transition	Initial transition strategy
To-Be	Enterprise	Architecture vision and principles summary. Identify TO-BE architecture patterns and high level mapping from AS-IS architecture
To-Be	Specification	Service Specification Architecture
To-Be	Implementation	Identify exceptional TO-BE design standards, design patterns, design solutions that will not be part of the factory Identify capabilities (may be utility, underlying and core services) to service common functional requirements in the solution/portfolio *
To-Be	Technology	Solution Platform requirements
To-Be	Deployment	Deployment requirements
To-Be	Non Functional	Service levels Sizing and performance Schedule constraints
To-Be	Transition	Detailed transition strategy and plan
To-Be	Transition	Outline transition solution, service and data architecture and design considerations including audit and parallel execution comparison capabilities
To-Be	Transition	Outline data migration plan

Table 3: To-Be Solution Architecture Content

Note the Solution Architecture is developed using an Agile architecture approach that delivers just enough clarity to coordinate the portfolio for consistency and facilitate planning of delivery projects that will detail the solution architecture for specific program increments. This topic is further addressed in the Agile Modernization Process section below.

Externalization of Rules

As a matter of principle all business logic that is not immutable is specified independently of implementation and technology concerns. Everware-CBDI recommends the Decision Model Notation (DMN) standard that guides a consistent decision/rule architecture and fully normalized rule specifications. The DMN based rules architecture establishes a linkage to business and service architecture to ensure the rules specification activity can be carried out to meet product backlog priorities, while ensuring rule integrity across the program and portfolio. Similarly, care must be taken in harvesting rules from existing systems to ensure that existing design constraints are not repeated in the To-Be solutions and the rules architecture in combination with the DMN techniques ensure integrity and reusability of the rules.

Rules are called by services to render them loosely coupled.

Rule changes happen at the specification level. Allows variant rule delivery engines.

Governance & Compliance

SAE governance is about ensuring realization of the business goals of inherent agility and ongoing integrity in delivered solutions. Agile governance is achieved through automation, visibility and traceability of the common capabilities while encouraging innovation and delegated responsibility.

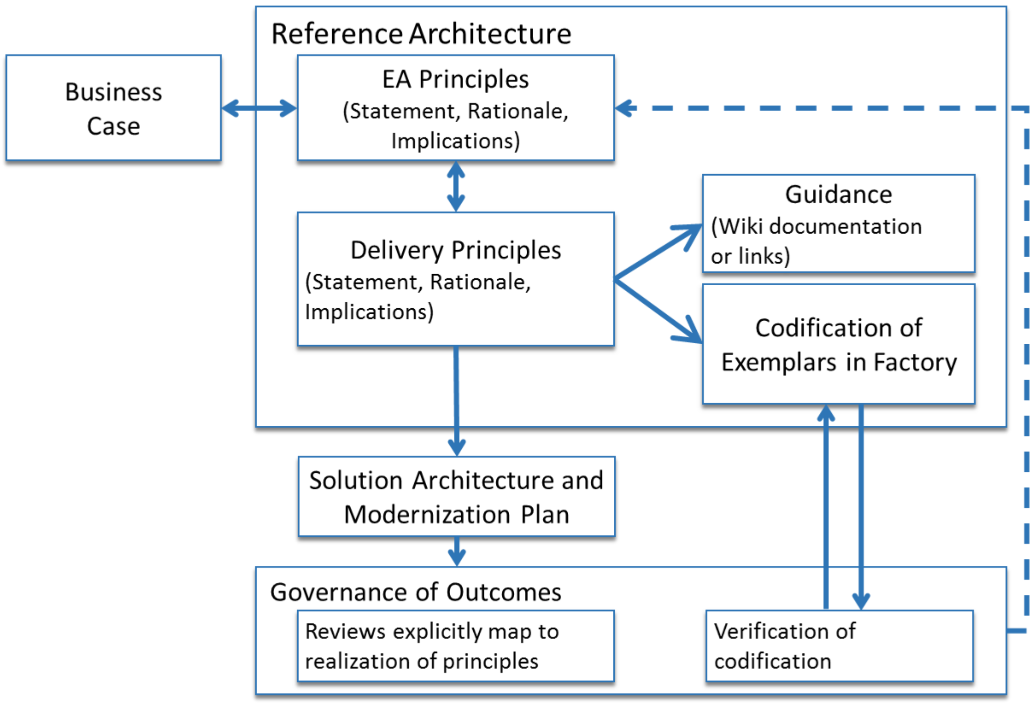
Reference Architecture
Business Case
EA Principles (Statement, Rationale, Implications)
Delivery Principles (Statement, Rationale, Implications)
Guidance (Wiki documentation or links)
Codification of Exemplars in Factory
Solution Architecture and Modernization Plan
Governance of Outcomes
Reviews explicitly map to realization of principles
Verification of codification

Figure 2: Modernization Governance

The SAE framework provides guidance on governance relating to the achievement of business agility and integrity goals. This might be referred to as a subset of the overall governance task addressing the realization of architecture in support of business goals.

Figure 2 illustrates how architecture principles, discussed above in Modernization Strategy, are realized, firstly as detailed delivery principles that define and document specific technologies, standards and policies that are then delivered as exemplars to be codified in the Service Factory to enable pattern based generation. In this way a significant proportion of governance over non-functional and common functional code is embedded in the factory, and once the codification has been verified once, no further review is necessary. There will always be exceptions and for these conventional Agile reviews and code inspections will be undertaken.

The factory based governance practice both reduces governance effort and increases productivity, while increasing compliance, traceability and visibility as follows:

- Business goals (for future agility) can be mapped to the reference and solution architecture to show how the various business goals are addressed in the modernized solutions.
- The reference architecture, plus policies and standards are embedded in the Agile Service Factory, ensuring that a high proportion of code is compliant.
- Exception designs that do not use the factory (for whatever reason) will be highly visible and tracked to ensure that full governance is carried out as appropriate.
- Specific legislative and regulatory compliance requirements can be identified in service and rule specifications that provide visibility and traceability of compliance requirements in the operational solutions.
- Test automation is driven by the service and rule specifications that ensures comprehensive test coverage.
- Comparison services generated from service specifications and existing systems interface specifications provide empirical evidence of outcomes.

Agile Modernization Process

The SAE and factory process can execute using a more purist Agile development method, enabling effective inter team dependency management with high levels of delegated responsibility with less risk of compromised reference architecture or unacceptable levels of technical debt.

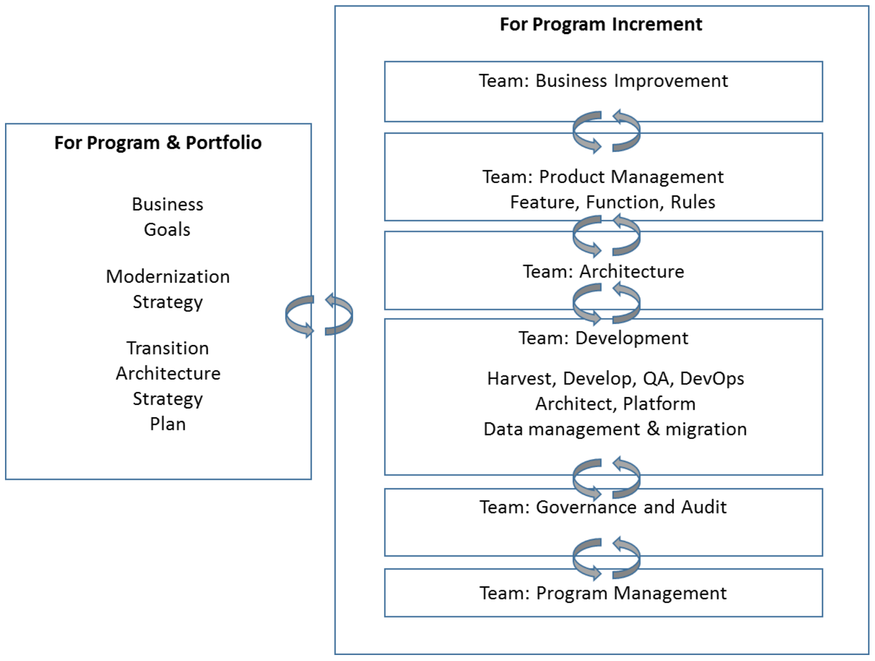

Figure 3: Agile Modernization Process

In principle the SAE and Factory modernization process is a conventional Agile process, with some important variations to address modernization specifics. As shown in Figure 3, it is likely SAE modernization will be a scaled Agile program/project given the size and complexity of enterprise solutions and services, with minimum necessary planning being undertaken for the portfolio and program, and then detailed delivery activities organized as Program Increments (PI) that deliver a significant business capability in its entirety.

Unlike well-known scaled agile processes, the SAE and factory based approach is less consumed with largescale and frequent ceremonies to coordinate and manage inter-team dependencies because the SAE framework provides consistent usage of reusable artifacts such as services and operations. In addition all of these are under management as specifications and factory managed code. There is therefore a high level of visibility and consistency of approach that makes coordination much more straightforward. In consequence the SAE and factory process can execute a more purist Agile development method, enabling effective inter team dependency management with high levels of delegated responsibility and less risk of compromised reference architecture or unacceptable levels of technical debt.

As discussed earlier, Business Goals for the Program and Portfolio must be clearly articulated to ensure the modernization strategy is designed accordingly; similarly, Transition Architecture and Strategy must be undertaken at the Program and Portfolio level in order to identify and outline units of business integrity that can be delivered in PIs. As Figure 3 indicates it is highly likely the strategy and plan will evolve in the light of the PI detailing.

For each PI, Agile teams use largely purist Agile methods, avoiding big-upfront design as follows:

- All teams are involved in Program and Portfolio planning efforts as appropriate.
- The Business Improvement & Implementation team provide subject matter experts and coordinate delivery and realization of the modernized business. Specific modernization processes will include harvesting and validating business knowledge, carrying out comparisons between old and new solutions, and retirement of existing systems.
- The Product Management team acts as the voice of the customer and coordinates the realization of objectives and goals through the product backlog. A key responsibility is to coordinate the product backlog together with the modernization and transition plan in conjunction with the Architecture and PMO teams. Modernization specific processes include collaboration on the transition strategy and specification of future capabilities that in addition to enabling the inherently agile business also provide units of integrity that can be transitioned into the business with minimum risk and impact.
- The Architects are organized as a virtual team distributed into Development teams, operating collectively as “architecture owners” with the responsibility to guide the development of the reference architecture and the transition architecture. In addition, the architects develop the As-Is architecture and are responsible for guiding harvesting and reuse efforts. A key collaboration of the Architecture team is with the Product and PMO teams to develop the product backlog in context with modernization and transition dependencies.
- The Development teams are cross functional involving QA, architecture, devops, and data. In addition, harvesting specialists will be required. Much harvesting will need to have been undertaken separately and preparatory to the development activity, prioritized and sequenced by PI demands, but Development Teams need access to harvesting skills in order to advise and supplement. Also data migration is a major effort with the Data Team working very closely with the Development Teams.
- The Governance team is a virtual team comprising architecture, PMO and product roles. As discussed above, architecture governance for common patterns and infrastructure is embedded in the factory. Only exceptions are subject to review and the exceptions governance process is integrated into the Agile development life cycle. A key aspect of governance is the audit of the delivered functionality in providing integrity and continuity with the existing rules, solutions and data. In most enterprise situations is a major preoccupation, and may be undertaken by automated comparison of the old and new solutions, rules and data. Commonly the delivery governance is undertaken by a separate team, and potentially by an independent third party.
- Although Program Management might be seen as mutually exclusive with Agile development, in largescale enterprise modernization situations there is a clear requirement for PMO specialists to manage planning, dependency coordination, risk

and cost governance and metrics. In the modernization context, the PMO processes also include all aspects of coordination with current portfolio team from knowledge acquisition to cutover and existing systems retirement.

About Everware-CBDI

Everware-CBDI is a technology consulting company with world leading expertise in service architecture and engineering, software factories and enterprise modernization. The company has led best practice development in service architecture and factory based development providing clients in all industry sectors and government with facilitation, skills transfer, implemented solutions, enhanced capabilities, automated tooling as well as documented, repeatable processes. www.everware-cbdi.com