The Open Group Service Integration Maturity Model (OSIMM) Version 2 – Example Assessment

This appendix illustrates the use of the OSIMM by describing an example assessment. The company described – HEALTHCO – is fictional. The assessment is described at an outline level, showing the main features of the method, but not the details of the specific indicators, attributes, weightings, and assessment questions.

Business Objective

HEALTHCO, a company providing healthcare services, envisioned an SOA to drive integration, promote a common business and IT vision, and optimize IT spending to support its business goals. To accomplish this vision, HEALTHCO needed to identify gaps in its current IT environment from the service integration maturity perspective. The OSIMM was used to assess the current state, determine the target state, and develop recommendations across the OSIMM dimensions.

Analysis

In the example, a number of applications were divided into two groups – front-end and legacy – and an OSIMM analysis was performed. The steps, focusing on the Business dimension, are summarized below:

- A pain-point is that the business perceives IT as not being sufficiently agile to support the introduction of new business processes.
- By analyzing the maturity indicators, it is determined that the business sees IT as applications rather than composite services that can be created from other services.
- This places the organization currently at Level 2 on the Business dimension.
- The applications are monolithic and are not integrated with other systems.
- By considering the characteristics of different levels as defined in the OSIMM, it can be seen that business at Level 5 will alleviate the pain-point by facilitating the design of new business processes from services.
- The need to go from Level 2 to (at least) Level 5 in the Business dimension suggests a step in the roadmap of introducing business processes and services to structure the functionality.

Recommendations

The recommendations are summarized in the following table, together with the current and target maturity levels for each of the dimensions:

OSIMM Dimension	Current Maturity Level	Summarized Assessment	Target Maturity Level	Recommendations
Business View	2	Strengths: • Business has good understanding of IT capabilities. Weaknesses: • Business views IT as a set of applications that deliver capabilities to support business processes. • Business capabilities are not aligned with IT. • Application inter-dependencies and complexities affect business agility.	6	A componentized view of the business capabilities should be developed in which business views services as assets that transcend the current application-centric views.
Organization	3	Strengths: • Cross-application organization is in place. • Responsibility for service enablement is managed. Weaknesses: • The IT organization is mostly application-centric with specialized skills to manage the development and support for specific applications.	4	Business owners should drive changes to services, business processes, and the component architecture to meet changing business needs. IT owners should be assigned to support specific business capability areas and their business owners. Business capability owners should be enabled to focus more on sustaining and improving specific capabilities.
Methods	2	Strengths: • IT planning and governance process in place. • Consistent development methodology followed. • Object-oriented design and development practices in place for front-end applications. • Services standards and guidelines are published. Weaknesses: • Planning and development process does not support services modeling or code re-use, with limited support for business process modeling. • Planning and development process is heavyweight and waterfall-based.	4	Enhance planning and development methods to support services identification, design, and development. Introduce services governance process. Enhance planning and development processes to encourage and promote code re-use. Enhance planning and development processes to support iterative development. Enhance the software development method to support business process modeling and implementation.
Infrastructure	3	Strengths: • System management software is in place. • Security infrastructure is in place. Weaknesses: • SOA-specific infrastructure (services management, BPM) is absent.	5	Deploy web services management infrastructure to support enterprise-scale SOA deployment. Deploy Business Process Management (BPM) infrastructure. Deploy SOA security infrastructure to be able to support security policies defined at the service level.
Applications (Front-end)	3	Strengths: • Architecture is componentized and layered. • Object models used. Weaknesses: • Minimal code re-use. • Object models not shared and are developed independently. • BPM/workflow is custom or not in place. • Application architecture is not standardized.	5	Implement enterprise domain object model. Introduce code re-use at the component and library level. Standardize reference application architecture, design patterns, and best practices. Implement business rules engine. Modernize and componentize COBOL applications.
Applications (Legacy)	2	Strengths: • Efforts are in place to modernize the application architecture. • Legacy access views provide a consistent approach for code re-use. Weaknesses: • COBOL legacy architecture difficult to change. • No consistent approach for system componentization. • BPM/workflow is custom or not in place. • Application architecture is not standardized and does not address back-end applications.	5	Implement enterprise domain object model. Introduce code re-use at the component and library level. Standardize reference application architecture, design patterns, and best practices. Implement business rules engine. Modernize and componentize COBOL applications.
Architecture Integration & Services (Front-end)	3	Strengths: • Most applications consume legacy access views using standard approach. • Some applications act as service providers. • WSDL files published within each application. Weaknesses: • Point-to-point integration. • Different protocols and translation mechanisms used for mainframe integration. • Security architecture is inconsistent.	5	Implement re-usable business services. Implement Enterprise Integration Data Model (Canonical Data Model). Implement uniform transport protocol for web services. All communications with internal and external systems should be handled by ESB. Support legacy consumers using ESB. Implement some of the application components as coarse-grained service components where component's interfaces are exposed using web services. All applications, including mainframe back-end systems, communicate via web services as opposed to re-using copybooks and legacy access views directly. COBOL applications should be able to act as consumers of web services provided by other back-end systems. SOA must provide the support for batch processing; batch processing should be implemented "on the side". Design and implement security policies at the service level.
Architecture Integration & Services (Legacy)	3	Strengths: • Legacy access views are used to provide services to other applications and are documented. • An approach is in place to make the legacy access views available to consuming systems. • ESB implemented. Weaknesses: • Back-end systems tightly coupled. • Some legacy access views not generic. • No enterprise data model for system integration. • Business functions are duplicated in multiple systems. • Heavy reliance on batch feeds. • Security architecture is inconsistent.	5	Implement re-usable business services. Implement Enterprise Integration Data Model (Canonical Data Model). Implement uniform transport protocol for web services. All communications with internal and external systems should be handled by ESB. Support legacy consumers using ESB. Implement some of the application components as coarse-grained service components where component's interfaces are exposed using web services. All applications, including mainframe back-end systems, communicate via web services as opposed to re-using copybooks and legacy access views directly. COBOL applications should be able to act as consumers of web services provided by other back-end systems. SOA must provide the support for batch processing; batch processing should be implemented "on the side". Design and implement security policies at the service level.

⇒ *Benefits of Moving to Higher Maturity Levels*