



**An Introduction to
Total Business Integration (TBI)
Methodology**

**By Manish Varma, Chair, Integration Maturity Model Committee
Technology Manager, eBusiness Architecture and Technologies
Johnson & Johnson**

1 Overview

Enterprise Application Integration (EAI) technology is based on “Change Nothing, Integrate Everything” philosophy. This means that the architects/designers do not have the luxury of building applications from scratch, they have to integrate applications that were previously developed with or without integration in mind. This paradigm makes EAI technology extremely complex. In order to reduce the complexity and ease the integration efforts, one needs a robust, time-tested and a proven methodology for easy, cost-effective and rapid implementation of integration projects within an organization.

A methodology is an organized, documented set of procedures and guidelines, a step-by-step “cookbook” approach of carrying out the procedure and an objective set of criteria for determining whether the results of the procedure are of acceptable quality. The Integration Consortium realizes that effective application integration strategy requires a robust integration methodology along with a robust integration tool. The IC with our members have developed a Total Business Integration (TBI) Methodology to meet its enterprise integration needs.

The Total Business Integration (TBI) Methodology is a business process oriented methodology designed for projects that are integrating data, applications and processes within or across multiple business units using heterogeneous systems, throughout the enterprise. The methodology provides a foundation to maximize re-use opportunities by providing a common design approach, reusable templates and processes that can be leveraged by other projects within Enterprises.

1.1 About the TBI Methodology

TBI methodology covers the full lifecycle of integration projects and is independent of the source and target applications and technologies. This methodology is to be used as a top-down approach i.e. driven by business process analysis. The objective is to address the integration needs of entire business processes rather than individual applications or data sources. However, the methodology can also be used in a bottom-up approach where projects can select and use templates from the methodology that meet their requirements.

TBI methodology is derived from industry standards and contains best practices and templates from one or more of the following sources:

Business Process Analysis worksheets from ebXML

ebXML is a public standards body created by UN/CEFACT (United Nations/Center for Trade Facilitation and Electronic Business) and OASIS (Organization for Structured Information Standards) collaboration in 2001. The business process analysis worksheets assist in understanding the business process, business events, transactions and information flow.

DMAIC (Six Sigma) & DMADV (Design Excellence) Methodologies

Used for PE projects

GEAR Methodology from Webmethods

GEAR (Goals, Explore, Assemble, Rollout) is a four phased methodology recommended by Webmethods.

UML from Object Management Group

UML is Object Management Group's public standards visual modeling language for application development.

SEI Capability Maturity Model (CMM) & Best Practices from System Integrators.

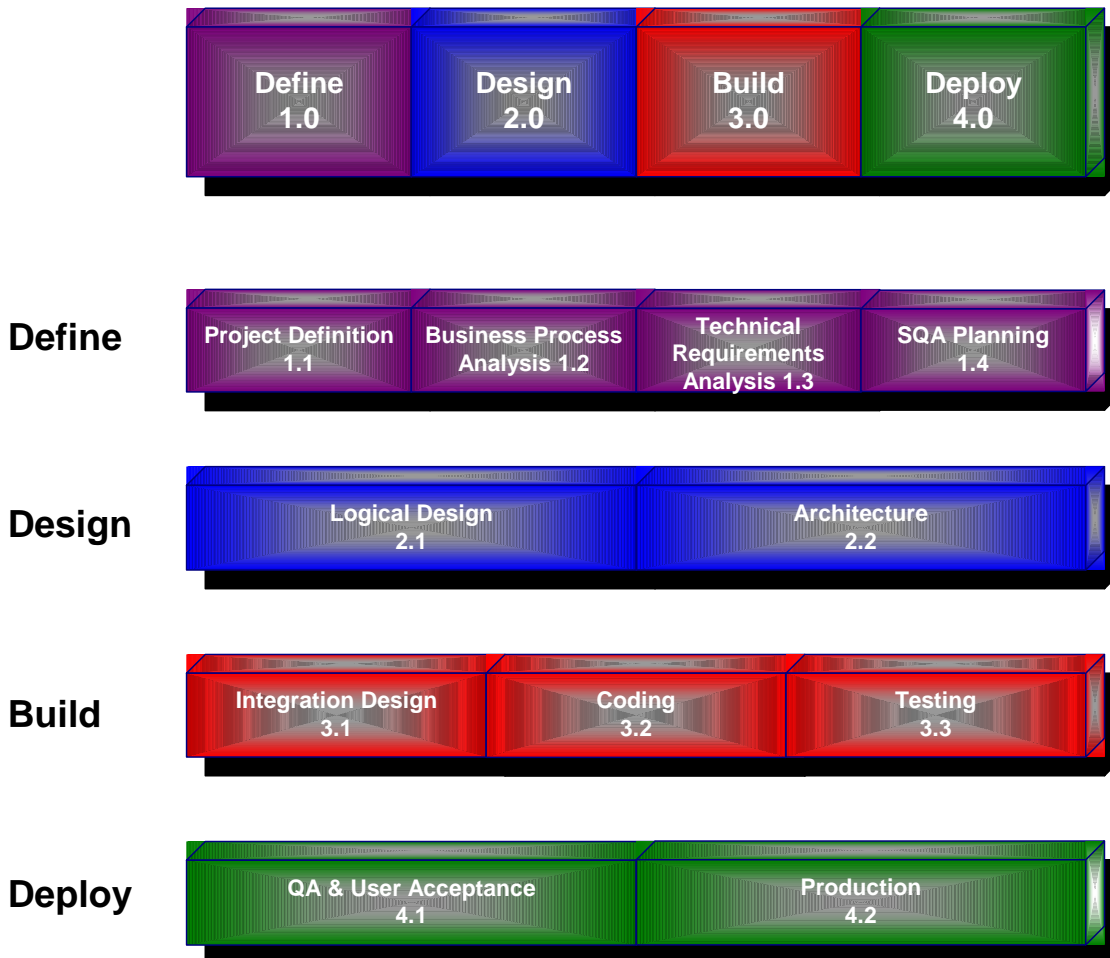
The SEI Capability Maturity Model (CMM) is a model for judging the maturity of software processes of an organization. Through the CMM, the SEI and process improvement community established an effective means of modeling, defining, and measuring the maturity of the processes used by organizations developing and maintaining software-intensive systems.

2 Benefits of TBI methodology

- Creates a standardized approach to integration across the Enterprise.
- Promotes reuse and leveraging of integration services and canonicals, which in turn will reduce total cost of ownership and increase the speed to market for projects.
- Enables Business process level integration to create future-proof solutions. It ensures that the changes to integration layer is minimal when the underlying business process or application architecture changes.
- Creates an “integration knowledge repository” within the organization to help jumpstart new integration projects.

3 Phases of TBI Methodology

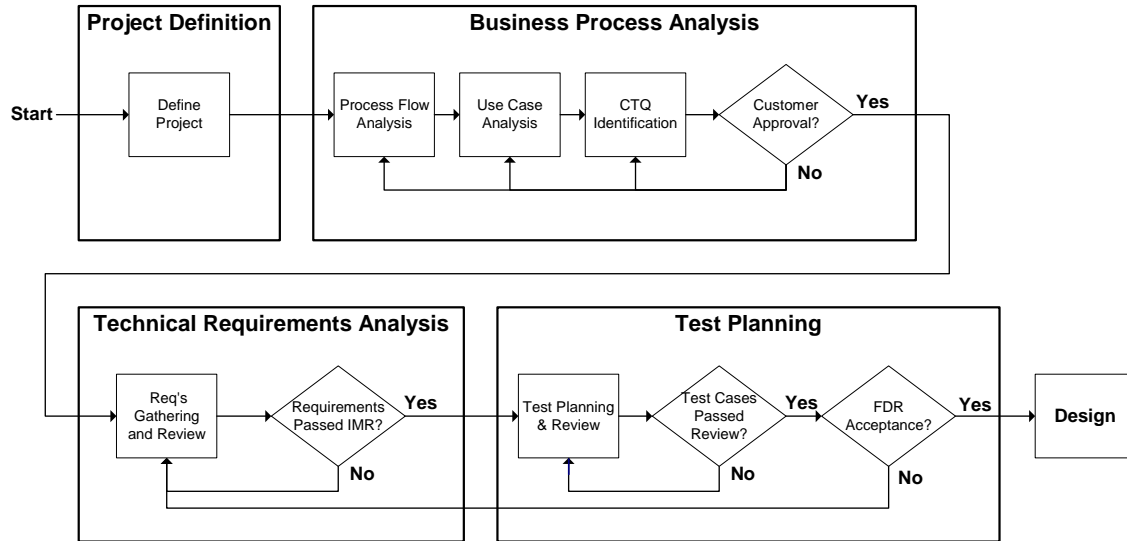
Different phases of TBI methodology along with description of each activity are described below. A high-level process flow, templates and deliverables for each phase are documented. For further details, please refer to the TBI Methodology document.



3.1 Define

The main purpose of “define” phase is to identify the scope, goals, objectives and technical requirements of the project and plan for the software quality assurance activities. During this phase the Change Management approach and the Testing Methodology is established for the project.

3.1.1 Define Phase – Process Flow



3.1.2 Define Phase – TBI Templates

TBI Integration Methodology Templates	
Define	
Project Definition	<i>TBI Template - Project Definition.doc</i>
Business Process Analysis Document	<i>TBI Template - Business Process Analysis.doc</i>
Requirements Document	<i>TBI Template - Technical Requirements.doc</i>
Requirements Walk-Through Report	<i>TBI Template - Requirements Walk-Through Report.doc</i>
System Test Plan	<i>TBI Template - Software Quality Assurance Plan.doc</i>
System Test Cases	<i>TBI Template – System Test Case.xls</i>
FDR Report	<i>TBI Template - FDR Report.doc</i>

3.1.3 Define Phase – Deliverables

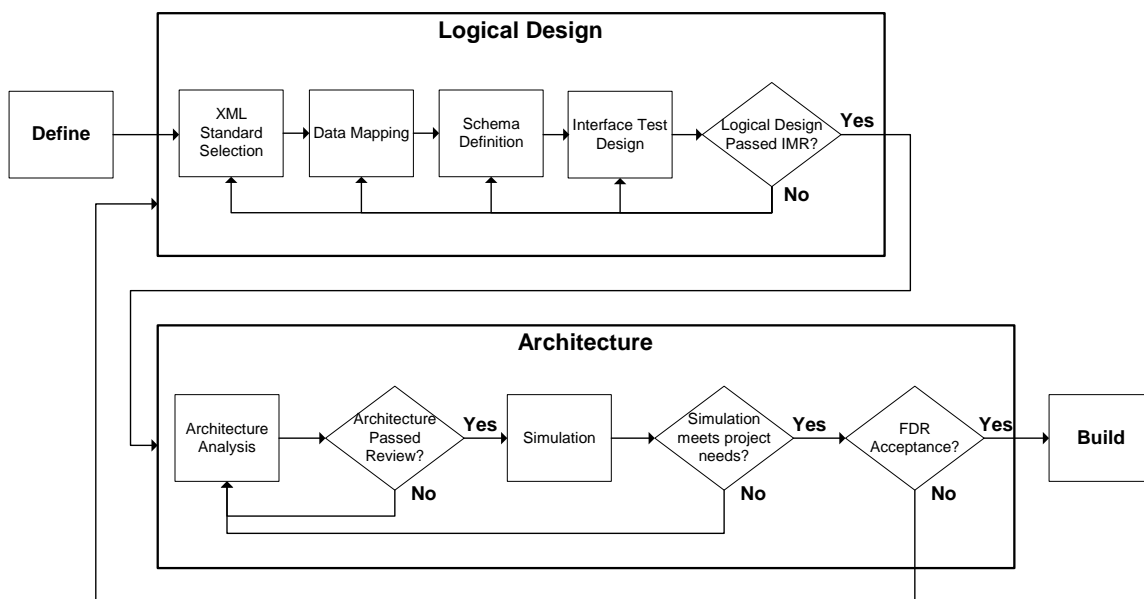
Deliverables	Description
Business Process Analysis Document	The business process analysis document is a compilation of a number of things: <ul style="list-style-type: none"> ➤ Project Definition ➤ Process and Functional Areas ➤ SIPOC Diagrams ➤ Use Cases ➤ CTQ's
Technical Requirements Document	The technical requirements document outlines the specific requirements for all the integrations that are identified. This document includes:

Deliverables	Description
	<ul style="list-style-type: none"> ➤ Functional Requirements ➤ Non-Functional Requirements such as Performance ➤ Data Requirements
Software Quality Assurance Plan	The system test plan is the overall plan for the quality assurance checkpoints and testing of all the integrations.
System Test Cases	The system test cases are written based on the requirements to ensure complete coverage of all the requirements

3.2 Design

Design phase includes logical design and architecture components of a project. Logical Design involves activities like selection of XML standard, data mappings and schema definition. Architecture analysis involves selection of interface coordination patterns, adapters, error handling, monitoring, logging, reprocessing etc.

3.2.1 Design Phase – Process Flow



3.2.2 Design Phase – TBI Templates

TBI Integration Methodology Templates	
Design	
Logical Design Document	<i>TBI Template - Logical Design.doc</i>
Logical Design Walk-Through	<i>TBI Template - Logical Design Walk-Through Report.doc</i>
Integration Test Cases	<i>TBI Template - Integration Test Case.xls</i>
Architecture and Deployment Document	<i>TBI Template - Architecture & Deployment.doc</i>
FDR Report	<i>TBI Template - FDR Report.doc</i>

3.2.3 Design Phase – Deliverables

Deliverables	Description
Logical Design	The high level design document containing the XML schema standard selection decision, data level mappings, message schemas, and a list of valid values for each message field.
Architecture Analysis document	The architecture analysis document summarizes the high-level system topology, adaptor selection, architecture recommendation, and approach for error handling, monitoring, logging and auditing.
Simulation document	This document presents the simulation approach, details, and results.
Integration test cases	The integration test cases are written based on the logical design to ensure complete coverage of all logical design elements (data mappings, etc.)

3.3 Build

The purpose of this phase is to develop the integration services, complete the technical documentation, and to execute test cases.

3.3.1 Build Phase – Process Flow



3.3.2 Build Phase – TBI Templates

TBI Integration Methodology Templates	
Build	
Integration Design	<i>TBI Template - Integration Design.doc</i>
Integration Walk-Through Report	<i>TBI Template - Integration Design Walk-Through Report.doc</i>
Code Reviews	<i>TBI Template - Code Review.doc</i>
Error Handling Guide	<i>TBI Template - Error Handling Guide.doc</i>
Unit Test Cases	<i>TBI Template - Unit Test Case.xls</i>
Unit Test Results	<i>TBI Template - Unit Test Results.xls</i>
Integration Test Results	<i>TBI Template - Integration Test Results.xls</i>
System Test Results	<i>TBI Template - System Test Results.xls</i>
FDR Report	<i>TBI Template - FDR Report.doc</i>

3.3.3 Build Phase – Deliverables

Deliverables	Description
Integration Design	Details the physical code design of the interface point(s). In addition, includes naming standards, error handling, and security settings
Source Code and Executables	The source code for the integrations and any executables (run-time code that may have been created).
Code Review	This document summarizes the results, issues, and follow-ups that come out of a formal code review.
Unit Test Cases	The unit test cases are written based on the integration physical design to ensure that the interface point adhere to the integration physical design.
Test Results	This document presents a summary of all of the tests that were run, and the results of these tests. Test Cases for unit, integration and system testing are all run in this phase.

3.4 Deploy

The purpose of the deploy phase is to ensure that the integrations meet the requirements documented in the define phase. Once the integration code meets the quality standards through quality assurance/user acceptance testing, it is deployed in production.

TBI methodology provides a checklist to help the project team to get their integration code ready for deployment in QA and Production.

Deployment Check List – QA	
	All code related to the integrations that needs to be promoted is in the appropriate configuration control.
	Instructions on how to deploy all the integrations in the QA environment must be documented. These documents must also be stored in the appropriate configuration control repository.
	System settings and configuration parameters required for the integrations must be documented and stored in the appropriate configuration control repository.
	System Test cases for testing the integrations in the QA environment must be provided to the team that will perform the testing.
	Error handling guide must be provided to the team that will perform all the testing.
	All updated project documentation must be in the configuration control repository.

Deployment Check List – Production	
	Sign-off from quality assurance/user acceptance testing
	All code related to the integrations that needs to be deployed in production is in the appropriate configuration control.
	Instructions on how to deploy all the integrations in the production environment must be documented. These documents must also be stored in the appropriate configuration control repository. (These documents would be the same as the ones created for the QA acceptance.)
	System settings and configuration parameters required for the integrations must be documented and stored in the appropriate configuration control repository. (These documents should be very similar to the ones created for the QA acceptance.)
	Error handling guide must be provided to the operations team that will support the integrations.
	System settings and configuration parameters required for the integrations must be documented and stored in the appropriate configuration control repository. (These documents should be similar to the ones created for the QA acceptance.)
	Error handling guide must be provided to the operations team that will support the integrations in the production environment.
	All updated project documentation must be in the configuration control repository

3.4.1 Deploy Phase – TBI Templates

TBI Integration Methodology Templates	
Deploy	
FDR Report	<i>TBI Template - FDR Report.doc</i>
Lessons Learned	

3.4.2 Deploy Phase – Deliverables

Deliverables	Description
Implemented Integration Solution	The interfaces that are deployed in the production environment.
Test Results	This document presents a summary of all of the tests that were run, and the results of these tests in the QA/User acceptance environment.
Lessons Learned	A listing of the significant learning's that were made during the completion of the TBI Integration Project. This can include but is not limited to: Project Management, Technical Implementation, Quality Assurance, Testing, Communications, etc.

4 Roles and Responsibilities

TBI methodology recommends following roles and responsibility for a typical integration project.

4.1 Project Manager

The Project Manager provides overall project management, vendor management, strategy and vision, project planning, staffing, change management, and risk management. In addition the project manager is involved in formal deliverables review (review and sign-off of deliverables by the business users).

Project Manager	
Define Activities	Deliverables
Define Project	Project Definition
Formal Deliverables Review	Formal Deliverables Review Report
Design Activities	Deliverables
Formal Deliverables Review	Formal Deliverables Review Report
Build Activities	Deliverables
Formal Deliverables Review	Formal Deliverables Review Report
Deploy Activities	Deliverables
Formal Deliverables Review	Formal Deliverables Review Report
Project Closing	Lessons Learned

4.2 Quality Manager

The Quality Manager is responsible for developing and coordinating all testing, coordination with Customer's testing and quality personnel, and final review of all deliverables.

Quality Manager	
Define Activities	Deliverables
CTQ Identification	CTQ Document
Functional Requirements Gathering	Requirements Walk-Through Report
Non-Functional Requirements Gathering	Requirements Walk-Through Report
Data Requirements Gathering	Requirements Walk-Through Report
System Test Design	System Test Plan System Test Cases
Design Activities	Deliverables
Data Mapping	Logical Design Walk-Through Report
Interface Test Design	Integration Test Plan Integration Test Cases
Simulation	Simulation Document
Build Activities	Deliverables
Integration Design	Integration Design Walk-Through Report
Unit Test Design	Unit Test Plan Unit Test Cases
Test Execution	Unit Test Results Integration Test Results System Test Result
Formal Deliverables Review	CTQ Acceptance Signoff
Deploy Activities	Deliverables

Quality Manager	
Facilitate User Acceptance testing	Test Results

4.3 Architect

The Architect is responsible for architecture design, integration development consistency, technical design reviews, and assistance with strategy and vision of integration throughout the Customer's enterprise.

Architect	
Define Activities	Deliverables
Business Process Flow Analysis	Project Definition
Define Project	Process and Functional Areas
Use Case Analysis	Use Cases
CTQ Identification	CTQ Document
Functional Requirements Gathering	Functional Requirements
Non-Functional Requirements Gathering	Non-functional Requirements
Data Requirements Gathering	Data Requirements Error Handling Requirements Monitoring Requirements
Design Activities	Deliverables
XML Standard Selection	XML Standards
Data Mapping	Data Map
Schema Definition	XML Schemas UDM Composition Document
Architecture Analysis	Architecture Document
Simulation	Simulation Document

4.4 Business Analyst

The Business Analyst is responsible for reviewing the business processes, gathering of business requirements, data mapping activities, and the creation of the Requirements and Logical Design documentation.

Business Analyst	
Define Activities	Deliverables
Process Flow Analysis	Project Definition Process and Functional Areas SIPOC Diagrams
Use Case Analysis	Use Cases
CTQ Identification	CTQ Document
Functional Requirements Gathering	Functional Requirements
Non-Functional Requirements Gathering	Non-functional Requirements Error Handling Requirements Monitoring Requirements Performance Requirements Constraining Requirements

Business Analyst	
Data Requirements Gathering	Data Requirements
System Test Design	System Test Cases
Design Activities	Deliverables
Data Mapping	Data Map
Interface Test Design	Integration Test Cases
Build Activities	Deliverables
Test Execution	System Test Results Integration Test Results

4.5 Designer

The Designer is responsible for completing the design and documentation of the integration design, technical leadership, and data mapping activities.

Designer	
Define Activities	Deliverables
Functional Requirements Gathering	Functional Requirements
Non-Functional Requirements Gathering	Non-functional Requirements Error Handling Requirements Monitoring Requirements Performance Requirements Constraining Requirements
Data Requirements Gathering	Data Requirements
Design Activities	Deliverables
XML Standard Selection	XML Standards
Data Mapping	Data Map
Schema Definition	XML Schemas UDM Composition Document
Interface Test Design	Integration Test Plan Integration Test Cases
Architecture Analysis	Architecture Document
Simulation	Simulation Document
Build Activities	Deliverables
Integration Design	Integration Design
Interface Development	Source Code / Executables Code Reviews Error Handling Guide
Unit Test Design	Unit Test Plan Unit Test Cases
Test Execution	Unit Test Results

4.6 Developer

The Developer is responsible for middleware development and configuration, language-specific development (such as Java), and database development.

Developer	
Design Activities	Deliverables
Architecture Analysis	Architecture Document
Simulation	Simulation Document
Build Activities	Deliverables
Integration Design	Integration Design
Interface Development	Source Code / Executables Code Reviews Error Handling Guide
Unit Test Design	Unit Test Plan Unit Test Cases
Test Execution	Unit Test Results