Open Group Standard

The Open Group IT4IT™ Reference Architecture, Version 2.0

Open Group Standard

The Open Group IT4IT™ Reference Architecture, Version 2.0

Document Number: C155

ISBN: 1-937218-69-0

Published by The Open Group, October 2015.

Comments relating to the material contained in this document may be submitted to:

The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom

or by electronic mail to:

it4it-comments@opengroup.org

Contents

List of Figures

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through IT standards. With more than 450 member organizations, The Open Group has a diverse membership that spans all sectors of the IT community – customers, systems and solutions suppliers, tool vendors, integrators, and consultants, as well as academics and researchers – to:

- Capture, understand, and address current and emerging requirements, and establish policies and share best practices
- Facilitate interoperability, develop consensus, and evolve and integrate specifications and open source technologies
- Offer a comprehensive set of services to enhance the operational efficiency of consortia
- Operate the industry's premier certification service

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Open Group Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/bookstore.

Readers should note that updates – in the form of Corrigenda – may apply to any publication. This information is published at www.opengroup.org/corrigenda.

The IT4IT™ Forum

The IT4IT Forum, a Forum of The Open Group, was created when its predecessor, the IT4IT Consortium, transferred its activities to The Open Group. The IT4IT Consortium came into being in 2011 as a practitioner-driven initiative. The Consortium was comprised of IT professionals from multiple industry segments and several IT vendors who agreed to share their experiences for the purpose of developing and publishing future-safe prescriptive guidance for implementing end-to-end an IT4IT architecture with full insight. Past and present members include Enterprise Architects and IT department leaders or industry consultants from: Accenture, Achmea, AT&T, HP IT, ING Bank, Munich RE, PwC, Royal Dutch Shell, and University of South Florida.

The Consortium formed a strategy board that spent thousands of hours sharing their insights and analyzing their experiences to develop a future-safe IT operating model. This model includes a prescriptive architecture for implementing the IT Value Chain to optimize the efficiency and effectiveness of IT. The strategy board is backed by an executive board sponsor from each member company.

"Cloud services and multi-provider outsourcing are adding new degrees of complexity to IT service management. The Consortium will use its real-life, cross-industry expertise to define a new operating model for IT."

Dr. Dirk Heiss, Global Infrastructure Services Officer, Munich RE

This Document

The Open Group IT4IT™ Reference Architecture refers to the capability or capabilities required to manage the business of IT, covering IT end-to-end from plan, through build and operate. It assumes the principle that the business of running IT is industry-agnostic and that IT leaders share the same problems and opportunities in managing the service lifecycle effectively. At the core, these problems are rooted in IT structure, competencies, and capabilities and the missing link has been the lack of an IT operating model. The IT4IT Reference Architecture proposes that it is possible to establish an IT operating model standard mapped to the existing IT landscape yet flexible enough to support the volatility inherent in the IT industry and accommodate changing IT paradigms (composite apps, agile development, mobile technology, multi-sourcing, etc.).

The IT Value Chain and IT4IT Reference Architecture represent the IT service lifecycle in a new and powerful way, providing the missing link between industry standard best practice guides and the technology you need to select and execute those processes. The IT Value Chain and IT4IT Reference Architecture are a new foundation on which to base your IT4IT operating model and provide a welcome blueprint for the CIO to accelerate IT's transition to becoming a service broker to the business.

This document is The Open Group IT4IT Reference Architecture, Version 2.0, an Open Group Standard. It has been developed and approved by The Open Group.

This document is structured as follows:

- Chapter 1 (Introduction) introduces this document and the purpose of the IT4IT work.
- Chapter 2 (Definitions) lists important definitions needed in order to read the document.
- Chapter 3 (Overview) is an introduction for executives and others introducing the IT Value Chain and IT4IT Reference Architecture concepts.
- Chapter 4 (IT4IT Core) defines the structure of the IT4IT standard as well as the process and document structure used by the IT4IT standard.
- Chapter 5 (Strategy to Portfolio (S2P) Value Stream) explains the S2P Value Stream in detail.
- Chapter 6 (Requirement to Deploy (R2D) Value Stream) explains the R2D Value Stream in detail.
- Chapter 7 (Request to Fulfill (R2F) Value Stream) explains the R2F Value Stream in detail.
- Chapter 8 (Detect to Correct (D2C) Value Stream) explains the D2C Value Stream in detail.

- Appendix A (Rationale (Informative)) contains background information on the standard.

How to Use this Standard

It is recommended that the reader start by familiarizing themselves with Chapter 3 (Overview) which introduces the concepts of the IT Value Chain. This should then be followed by the IT4IT Core (Chapter 4), and the four IT Value Streams. These are:

- Strategy to Portfolio (S2P) Value Stream (Chapter 5)
- Requirement to Deploy (R2D) Value Stream (Chapter 6)
- Request to Fulfill (R2F) Value Stream (Chapter 7)
- Detect to Correct (D2C) Value Stream (Chapter 8)

Documentation Structure of the IT4IT Reference Architecture

Figure 1 is a graphical representation of the data objects associated with the IT4IT Reference Architecture. The architecture is comprised of a set of normative documents[1] and a formal model described using the ArchiMate® modeling language and UML. These define "what" the architecture is. The normative documentation includes:

- IT4IT Reference Architecture Overview
- IT4IT Value Stream Overview
- IT4IT Reference Architecture diagrams
- IT4IT meta-model diagram
- Glossary

The Reference Architecture diagrams, in ArchiMate notation, will be provided as a set of web pages, and also as a downloadable zip file of those pages. When available, they will be found at: https://collaboration.opengroup.org/data/IT4IT/RA_2.0.

A set of guidance documents is being developed to accompany the architecture, intended to describe "how" to apply the architecture in practice. Planned guidance documents include:

- Multi-Supplier Management White Paper
- Definition of IT Service White Paper
- Service Model Management White Paper
- Scenario White Papers (see Section 4.2.6.1)

[1] A normative document is one that provides rules, guidelines, or characteristics for activities or their results. The term "normative document" is a generic term that covers such documents as standards, technical specifications, codes of practice, and regulations; e.g., European Cooperation for Space Standardization (ECSS).

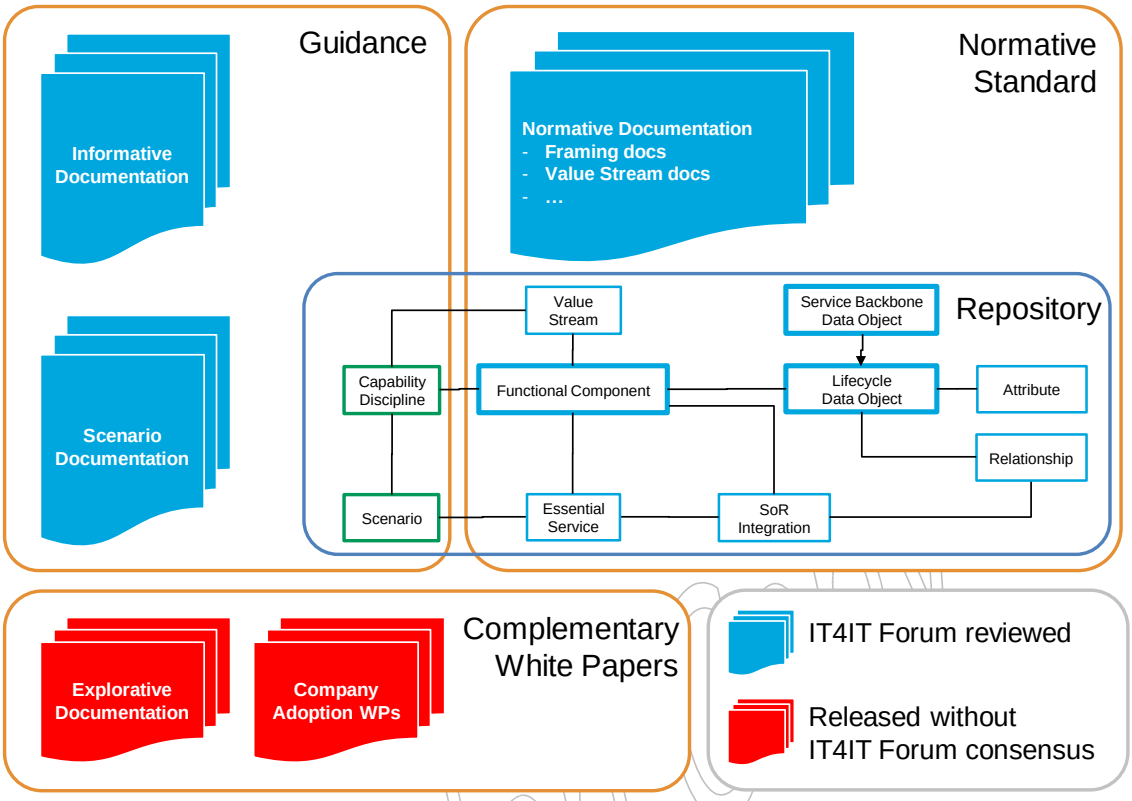


Figure 1: Documentation Structure of the IT4IT Reference Architecture

Documents/artifacts that fall into these two categories are governed by The Open Group Standards Process. In addition, The Open Group will maintain a set of White Papers that complement the architecture and elaborate on its applicability and use in various settings.

Related Industry Standards

Most IT management standards fall into one of two categories: process and/or method-focused technology and/or implementations-centric. There are no standards that prescribe both the operating model and automation guidelines for running the IT function. Therefore, the IT4IT Reference Architecture fills this gap and as such complements a number of existing standards and best practices such as:

- ISO/IEC 19770:2012: Information Technology – Software Asset Management
- ISO/IEC 20000:2011: Information Technology – Service Management
- ISO/IEC 38500:2008: Corporate Governance of Information Technology
- ISO/TC 258: Project, Program, and Portfolio Management
- Information Technology Infrastructure Library (ITIL)
- Control Objectives for Information and Related Technology (COBIT)
- Business Process Framework (eTOM)

- The TOGAF® standard
- The ArchiMate modeling language
- The Scaled Agile Framework (SAFe)
- The Project Management Body of Knowledge (PMBOK)

Trademarks

ArchiMate®, DirecNet®, Making Standards Work®, OpenPegasus®, The Open Group®, TOGAF®, UNIX®, and the Open Brand ("X" logo) are registered trademarks and Boundaryless Information Flow™, Build with Integrity Buy with Confidence™, Dependability Through Assuredness™, FACE™, IT4IT™, Open Platform 3.0™, Open Trusted Technology Provider™, UDEF™, and the Open "O" logo and The Open Group Certification logo are trademarks of The Open Group in the United States and other countries.

CMMI® is registered in the US Patent and Trademark Office by Carnegie Mellon University.

COBIT® is a registered trademark of the Information Systems Audit and Control Association (ISACA) and the IT Governance Institute.

eTOM® is a registered trademark of the TM Forum.

ITIL® is a registered trademark of AXELOS Ltd.

OASIS™ and TOSCA™ are trademarks of OASIS.

OMG®, Unified Modeling Language®, and UML®, are registered trademarks of the Object Management Group, Inc. in the United States and/or other countries.

All other brands, company, and product names are used for identification purposes only and may be trademarks that are the sole property of their respective owners.

Acknowledgements

This standard was prepared by The Open Group IT4IT™ Forum.

When The Open Group approved the IT4IT™ Reference Architecture, Version 2.0, an Open Group Standard, on October 5, 2015, the membership of the IT4IT™ Forum was as follows:

Chris Davis, University of South Florida, Chair
Karel van Zeeland, Shell Information Technology International BV, Vice-Chair
Martin Kirk, The Open Group, Forum Director
Andrew Josey, The Open Group, Director, Standards
Cathy Fox, The Open Group, Technical Editor

Contributors

The Open Group gratefully acknowledges the contribution of the following members of the IT4IT™ Forum in the development of this standard:

Richard Aarnink
Rob Akershoek
Chris Armstrong
Charles Betz
Georg Bock
Mark Bodman
Paul Buckley
James Caruso
Eran Cohen
Sam Courtney
Sukumar Daniel
Dwight David
Christopher Davis
Sue Desiderio
Ulrich Feyer
Mike Fulton
Philippe Geneste
Ohad Goldfarb
Mark Gray
Claudia Guli
Trey Harris
Rob Hengeveld
Brian Hodgdon
Keith Jahn
Jim Johnson
Linda Kavanagh
Mark Luchtmeijer
Sylvain Marie
Lakshmi Malhotra
Gunnar Menzel
Satya Misra
Brian Ng
Dan Rosenzweig
Lars Rossen
Vasu Sankhavaram
Ryan Schmierer
Rick Solis
Ken Street
Mary Street
Etienne Terpstra
Richard Tian
Erik van Busschbach
Kees van den Brink
Karel van Zeeland
Gerlan Verlouw
Prafull Verma
Floris Verschoor
Ulrich Wanka
Erik Witte

Technical Reviewers

Technical reviewers are those individuals who have submitted comments during the company review, or participated in the resolution process during the development of the IT4IT Reference Architecture, Version 2.0.

Sam Courtney	Andrew Josey
Chris Davis	Martin Kirk
Sue Desiderio	Randall Ramsey
Thorbjörn Ellefsen	Lars Rossen
Ohad Goldfarb	Mark Smalley

IT4IT™ Forum Members

The following organizations were members of the IT4IT™ Forum at the time of approval:

Accenture Limited, USA
Achmea, The Netherlands
Action Research Foundation, India
Agency for Public Management and eGovernment (Difi), Norway
Arismore, France
Armstrong Process Group, Inc., USA
AT&T IT Architecture Solutions, USA
ATE Enterprises Ltd., UK
Biner Consulting, Sweden
BP Oil International Limited, UK
CA, Inc., USA
Capgemini S.A., The Netherlands
CC and C Solutions, USA
Conexiam Solutions, Inc., USA
Dux Diligens S.A de C.V, Mexico
EA Principals, Inc., USA
Ernst & Young, UK
ExxonMobil, USA
Fujitsu, Japan
HCL Technologies Ltd., India
Hewlett-Packard Company, USA
Huawei Technologies, Co. Ltd., China
IBM, USA
Logicalis SMC, The Netherlands
Microsoft, USA
Ministerie van Financien, The Netherlands
Munich Re Group, Germany
National Healthcare, USA
Nationwide Mutual Insurance Company, Inc., USA
Oracle Corporation, USA
Origin Energy, Australia
PricewaterhouseCoopers LLP, USA
Raytheon Company, USA

Real IRM Solutions (Pty) Ltd., South Africa
ServiceNow, Inc., USA
Shell Information Technology International B.V., The Netherlands
Shift Technologies LLC, United Arab Emirates
Stichting ASL BiSL Foundation, The Netherlands
Tata Consultancy Services Ltd., India
The Boeing Company, USA
UMBRiO, The Netherlands
University of South Florida, USA
Westbury Software, The Netherlands

Referenced Documents

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

Normative References

Normative references for this standard are defined in Section 1.4.

Informative References

The following documents are referenced in this standard:

- Agile Alliance: Agile Manifesto and Principles (2001); retrieved 4/13/2011, from http://agilemanifesto.org/principles.html.
- J. Allspaw, J. Robbins: Web Operations, Beijing China, Sebastopol CA, O'Reilly (2010).
- ASL Foundation: Application Services Library (2005); retrieved 11/13/2005, from www.aslfoundation.org.
- K. Behr, G. Kim et al: The Visible Ops Handbook – Implementing ITIL in Four Practical and Auditable Steps, Eugene OR, Information Technology Process Institute (2005).
- R.J. Benson, T.L. Bugnitz et al: From Business Strategy to IT Action – Right Decisions for a Better Bottom Line, New York; Chichester, Wiley (2004).
- C.T. Betz: Architecture and Patterns for IT: Service and Portfolio Management and Governance (Making Shoes for the Cobbler's Children), 2nd Edition, Amsterdam, Elsevier/Morgan Kaufman (2011).
- P. Bourque, R.E. Fairley, Eds.: Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society (2014).
- J.A. Carbone: IT Architecture Toolkit, Upper Saddle River, NJ, Prentice Hall (2004).
- CMMI Product Team: CMMI for Acquisition, Version 1.3. Pittsburgh PA, Carnegie Mellon Software Engineering Institute (2010).
- CMMI Product Team: CMMI for Development, Version 1.3, Pittsburgh PA, Carnegie Mellon Software Engineering Institute (2010).
- CMMI Product Team: CMMI for Services, Version 1.3, Pittsburgh PA, Carnegie Mellon Software Engineering Institute (2010).
- A. Cockburn: Writing Effective Use-Cases, Boston, Addison-Wesley (2001).

- M.A. Cook: Building Enterprise Information Architectures – Re-Engineering Information Systems, Upper Saddle River, NJ, Prentice Hall (1996).
- P.M. Duvall, S. Matyas et al: Continuous Integration – Improving Software Quality and Reducing Risk, Upper Saddle River, NJ, Addison-Wesley (2007).
- J. Humble, D. Farley: Continuous Delivery, Boston, Addison-Wesley (2011).
- IEEE 730-2014: IEEE Standard for Software Quality Assurance Processes.
- ISACA: Control Objectives for Information and Related Technology (COBIT 5); refer to www.isaca.org.
- ISO/IEC 2005: Regional or National Adoption of International Standards and Other International Deliverables (Guide 21-2).
- ISO/IEC 2008: Uncertainty of Measurement (Guide 98-3).
- ISO/IEC 2013: ISO/IEC Directives.
- ISO/IEC 19770:2012: Information Technology – Software Asset Management.
- ISO/IEC 20000:2011: Information Technology – Service Management.
- ISO/IEC 27002:2013: Information Technology – Security Techniques – Code of Practice for Information Security Controls.
- ISO/IEC 38500:2008: Corporate Governance of Information Technology.
- ISO/TC 258: Project, Program, and Portfolio Management.
- J.D. Kaplan: Strategic IT Portfolio Management – Governing Enterprise Transformation, US, Pittiglio Rabin Todd & McGrath Inc. (2005).
- H. Kern, R. Schiesser et al: IT Production Services, Upper Saddle River, NJ, Prentice Hall Professional Technical Reference (2004).
- L. Klosterboer: Implementing ITIL Change and Release Management, Upper Saddle River, NJ, IBM; London: Pearson Education [distributor] (2009).
- D. Leffingwell, A. Yakyma et al: Scaled Agile Framework, from http://scaledagileframework.com (2014).
- T.A. Limoncelli, S.R. Chalup et al: The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems, Vol. 2, Pearson Education Ltd. (2014).
- D.C. Luckham: The Power of Events – An Introduction to Complex Event Processing in Distributed Enterprise Systems, Boston, MA; London, Addison-Wesley (2002).
- B. Maizlish, R. Handler: IT Portfolio Management Step-By-Step: Unlocking the Business Value of Technology, Hoboken, MJ, John Wiley & Sons (2005).
- J. Martin: Great Transition: Using the Seven Disciplines of Enterprise Engineering, ISBN: 978-0814403150, Amacom (January 1995).

- F.W. McFarlan: Portfolio Approach to Information Systems, Harvard Business Review 59(5): 142-150 (1981).
- OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA), Version 1.0 (2013).
- G. O'Donnell, C. Casanova: The Configuration Management Database (CMDB) Imperative: How to Realize the Dream and Avoid the Nightmares, Upper Saddle River, NJ, Prentice Hall; London: Pearson Education [distributor] (2009).
- Office of Government Commerce: Application Management, London, The Stationary Office (2002).
- M. O'Loughlin: The Service Catalog: A Practitioner Guide, Zaltbommel, The Netherlands, Van Haren Publishing (2009).
- M. Porter: Competitive Advantage: Creating and Sustaining Superior Performance, ISBN: 978-0684841465, Free Press; 1st Edition (June 1998).
- Project Management Institute: A Guide to the Project Management Body of Knowledge (PMBOK) (2013).
- T.A. Quinlan: Chargeback and IT Cost Accounting, Santa Barbara, CA, IT Financial Management Association (2003).
- T.A. Quinlan, S.J. Quinlan: Readings in IT Financial Management, Santa Barbara, CA, IT Financial Management Association (2003).
- Rational Software: Rational Unified Process: Best Practices for Software Development Teams (2011).
- D. Remenyi, A.H. Money et al: The Effective Measurement and Management of ICT Costs and Benefits, Oxford; Burlington, MA, CIMA (2007).
- R. Schiesser: IT Systems Management, Upper Saddle River, NJ, Prentice Hall (2010).
- R. Schiesser, H. Kern: Enterprise Computing Institute: IT Systems Management, Upper Saddle River, NJ, Prentice Hall (2002).
- SOA Reference Architecture, Open Group Standard (C119), December 2011, published by The Open Group; refer to: www.opengroup.org/bookstore/catalog/c119.htm.
- S.H. Spewak, S.C. Hill: Enterprise Architecture Planning – Developing a Blueprint for Data, Applications, and Technology, Boston, QED Pub. Group (1993).
- R. Sturm, W. Morris et al: Foundations of Service-Level Management, Indianapolis, IN, SAMS (2000).
- The Stationery Office: ITIL Continual Service Improvement, Norwich UK (2011).
- The Stationery Office: ITIL Service Design, Norwich UK (2011).
- The Stationery Office: ITIL Service Operation, Norwich UK (2011).

- The Stationery Office: ITIL Service Strategy, Norwich UK (2011).
- The Stationery Office: ITIL Service Transition, Norwich UK (2011).
- TOGAF Version 9.1 (English version), Open Group Standard, available online at www.opengroup.org/architecture/togaf9-doc/arch, and also available as TOGAF Version 9.1 "The Book" (ISBN: 978 90 8753 6794, G116) at www.opengroup.org/bookstore/catalog/g116.htm.
- W. Ulrich, N. McWhorter: Business Architecture: The Art and Practice of Business Transformation, Tampa, FL, Meghan-Kiffer (2010).
- Unified Modeling Language (UML), Object Management Group (OMG); refer to: www.uml.org.
- W. Van Grembergen: Strategies for Information Technology Governance, Hershey, PA; London, Idea Group Publishing (2004).
- W. Van Grembergen, S. Haes: Enterprise Governance of Information Technology – Achieving Strategic Alignment and Value, New York, Springer (2009).
- E.A. Van Schaik: A Management System for the Information Business – Organizational Analysis, Englewood Cliffs; London, Prentice Hall (1985).
- P. Weill, J.W. Ross: IT Governance – How Top Performers Manage IT Decision Rights for Superior Results, Boston, MA, Harvard Business School; [London: McGraw-Hill] (2004).

1 Introduction

1.1 Objective

This standard is the specification of The Open Group IT4IT Reference Architecture, Version 2.0, an Open Group Standard. It describes a reference architecture and value chain-based operating model for managing the business of IT.

1.2 Overview

The Open Group IT4IT Reference Architecture is a standard reference architecture and value chain-based operating model for managing the business of IT. It uses a value chain approach to create a model of the functions that IT performs to help organizations identify the activities that contribute to business competitiveness. This value chain framework, called the IT Value Chain and specified in this document as part of the IT4IT Reference Architecture (see Section 3.1), applies this concept to IT by defining an integrated IT management framework focusing on the lifecycle of services. It identifies the key things that IT must do – and do well. It allows IT to achieve the same level of business predictability and efficiency that supply chain management has allowed for the business, and was designed by practitioners to be industry, product, and vendor-independent.

1.3 Conformance

Readers are advised to check The Open Group website for any conformance and certification requirements referencing this standard.

1.4 Normative References

The following standard contains provisions which, through references in this standard, constitute provisions of the IT4IT Reference Architecture. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standard listed below.

- ArchiMate® 2.1 Specification, Open Group Standard (C13L), December 2013, published by The Open Group; refer to: www.opengroup.org/bookstore/catalog/c13l.htm.

1.5 Terminology

For the purposes of this standard, the following terminology definitions apply:

Can	Describes a possible feature or behavior available to the user or application.
May	Describes a feature or behavior that is optional. To avoid ambiguity, the opposite of "may" is expressed as "need not", instead of "may not".
Shall	Describes a feature or behavior that is a requirement of the standard. To avoid ambiguity, do not use "must" as an alternative to "shall".
Shall not	Describes a feature or behavior that is an absolute prohibition of the standard.
Should	Describes a feature or behavior that is recommended but not required.
Will	Same meaning as "shall"; "shall" is the preferred term.

1.6 Future Directions

Work is currently underway to produce a roadmap for how IT can move from the familiar IT capability-based understanding to implementing the new service-centric IT Value Chain model.

Scenarios currently being discussed to be constructed in future releases include the following:

- Multi-vendor Availability & Capacity Management
- Multi-vendor Service-Level Management (SLM)
- Hybrid Requirements Management in the enterprise
- Change Management, including the relationship with Configuration Management
- S2P alignment with the TOGAF standard, Service Model Management, and Service Definition
- Service Request (Self-service & Knowledge Management)
- Risk Management
- Asset Management
- Alignment to Cloud
- Intelligence and Reporting
- IT Financial Management

2 Definitions

For the purposes of this standard, the following terms and definitions apply. Merriam-Webster's Collegiate Dictionary should be referenced for terms not defined in this section. Note that the following definitions are ordered to reflect the hierarchy of the definitions.

2.1 Service Lifecycle Data Object (Data Object)

Data or records produced and/or consumed to advance or control the service model as it progresses through its lifecycle phases. Data objects can take a physical or digital form and are produced, consumed, or modified by functional components. Within the IT4IT Reference Architecture there are two classifications of data objects:

- Key – those that are essential to managing or advancing the service lifecycle.
- Auxiliary – those that are important but not essential.

Note: Early versions of the IT4IT Reference Architecture used the term "artifact" instead of "data object". There may remain some inconsistency in some documentation, but the terms should be viewed as the same.

See also Section 2.5.

2.2 IT Value Chain

A classification scheme for the set of primary and supporting activities that contribute to the overall lifecycle of creating net value of a product or service offering provided by or through the IT function. Within the IT4IT framework it is used to describe the operating model for the IT business function. It includes primary activities such as planning, production, consumption, fulfillment, and support. It also includes supporting activities such as finance, human resource, governance, and supplier management.

See also Section 2.3.

2.3 Value Chain

A classification scheme for the complete set of primary and supporting activities that contribute to the lifecycle of creating net value of a market offering. Originates from Michael Porter's book Competitive Advantage.[2]

2.4 Value Stream

Describes the key activities for a discrete area within the IT Value Chain where some unit of net value is created or added to the service as it progresses through its lifecycle. The IT4IT framework describes four value streams (Strategy to Portfolio, Requirement to Deploy, Request to Fulfill, Detect to Correct).

2.5 Functional Component

A software building block. The smallest unit of technology in the IT4IT Reference Architecture that can stand on its own and be useful as a whole to an IT practitioner (or IT service provider). Functional components must have defined inputs and outputs that are data objects and it must have an impact on a key data object.

2.6 Service Model Backbone Data Object

Key data objects that annotate an aspect of the service model in its conceptual, logical, or physical state. These data objects and their relationships form the Service Model Backbone which provides a holistic view of a service.

2.7 Relationship

Primarily used to depict the connections between (or interactions with) data objects. In the IT4IT Reference Architecture, relationships are based on three design principles:

- System of record – used to depict the relationships used to control authoritative source data via a system-to-system interface. These relationships are prescriptive in that they must be maintained to ensure the integrity of the IT4IT Reference Architecture.
- System of engagement – used to describe the relationships between data objects and humans or functional components via a user experience interface.
- System of insight – used to describe relationships between data objects for the purpose of generating knowledge, information, or analytics.

[2] Reference Michael Porter: Competitive Advantage: Creating and Sustaining Superior Performance.

Note: The current version of the IT4IT Reference Architecture places its primary emphasis on defining the system of record relationships. Some (but not all) system of engagement relationships are described. System of insight relationships have not been defined in the current version of the standard.

2.8 System of Record

A synonym for a system that contains and/or controls authoritative source data.

Note: This term can be easily confused with system of record relationships.

See also Section 2.7.

3 Overview

This chapter introduces the IT Value Chain and IT4IT Reference Architecture. Together these provide the foundation for an IT operating model where IT is the service broker to the lines of business.

3.1 What is the IT Value Chain?

A value chain is a series of activities that an organization performs in order to deliver something valuable, such as a product or service. Products pass through activities of a chain in order and, at each activity, the product gains some value. A value chain framework helps organizations to identify the activities that are especially important for competitiveness – for the advancement of strategy and attainment of goals. The IT Value Chain is grouped into two main categories of activities:

- Primary activities, which are concerned with the production or delivery of goods or services for which a business function, like IT, is directly accountable.
- Supporting activities, which facilitate the efficiency and effectiveness of the primary activities.[3]

With services as the center of gravity, a value-chain based model for IT has been constructed by identifying the critical activities associated with the planning, sourcing, delivery, and management of services. The IT Value Chain content details the series of activities that every IT department performs that add value to a business service or IT service. The IT4IT Reference Architecture breaks these activities down further to a Service Model and the essential functional components and data objects that IT produces or consumes in the IT Value Chain in order to advance the service lifecycle.

[3] For more on the value chain concept, read Michael Porter's "Competitive Advantage: Creating and Sustaining Superior Performance".

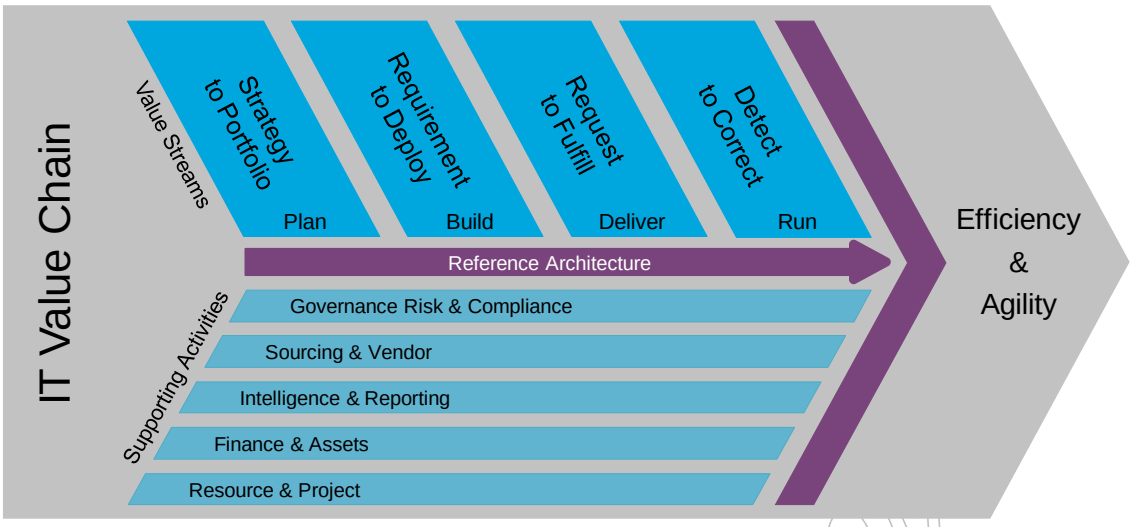


Figure 2: IT Value Chain

The functional components in the IT Value Chain are grouped into four primary IT Value Streams and five supporting activities, as follows.

The primary value streams for the IT Value Chain are:

- Strategy to Portfolio
- Requirement to Deploy
- Request to Fulfill
- Detect to Correct

The primary activities are core to the IT function and have a vital role in helping to holistically run the full service lifecycle. These are usually hosted within IT.

The supporting activities for the IT Value Chain are:

- Governance Risk & Compliance
- Sourcing & Vendor
- Intelligence & Reporting
- Finance & Assets
- Resource & Project

The supporting activities help ensure the efficiency and effectiveness of the IT Value Chain and primary value streams. These can be corporate or administrative functions that are hosted in the lines of business and/or IT.

This standard defines the value streams, functional components, and associated data objects that are critical to the service lifecycle.

The guiding principles of the IT4IT Forum state that the framework shall:

- Be flexible enough to support frequent changes in business models while sturdy enough to track compliance and cost controls.
- Be defined in practical terms for immediate application in "real-world" IT environments.
- Be able to be implemented in a phased approach that avoids any requirement for "rip and replace".
- Be technology and vendor-agnostic.
- Be accessible to anyone who wishes to make use of it. This provides the opportunity for IT departments to work towards the same standard and to benchmark performance data against a body of data from all participants.
- Be, wherever possible, complementary to current industry standard best practices.

3.2 IT Value Chain and IT4IT Reference Architecture

The IT Value Chain is the series of activities that IT performs to add value to a business service or IT service. The IT4IT Reference Architecture breaks down the activities in the IT Value Chain into four key pillars for IT – the Service Model, the Information Model, the Functional Model, and the Integration Model. Together these provide the prescription for the essential elements that IT must control to manage a service through its lifecycle.

Each IT Value Stream is centered on an essential element of the Service Model and the constellation of key data objects (Information Model) and functional components (Functional Model) that support it. Together, the four value streams play a vital role in helping IT holistically manage the full service lifecycle:

- The **Strategy to Portfolio (S2P) Value Stream** receives strategic demands for new or improved services from the business or IT itself and develops the Conceptual Service Blueprint to represent the new or enhanced business/IT service that is requested. The Conceptual Service Blueprint is the bridge between business and IT in that it provides the business context for the service along with the high-level architectural attributes.
- The **Requirement to Deploy (R2D) Value Stream** receives the Conceptual Service Blueprint and designs and develops the Logical Service Model with more detailed requirements that describe how the newly requested business/IT service and its components shall be designed. The Logical Service Model can be thought of as what is traditionally expressed in IT terms as the "system design". The Logical Service Model together with the Service Release make up the Logical Service Blueprint. The R2D Value Stream builds, tests, and delivers the deployable service (Service Release Blueprint) to the R2F Value Stream.
- The **Request to Fulfill (R2F) Value Stream** receives the Logical Service Blueprint after it has gone through development, test, and release approval. For repeatedly consumable services, the R2F Value Stream is responsible for the tasks to transition the service into production and make it consumable by the business including creating the Service Catalog Entry.

- The **Detect to Correct (D2C) Value Stream** provides a framework for integrating the monitoring, management, remediation, and other operational aspects associated with realized services and/or those under construction. It also provides a comprehensive overview of the business of IT operations and the services these teams deliver.

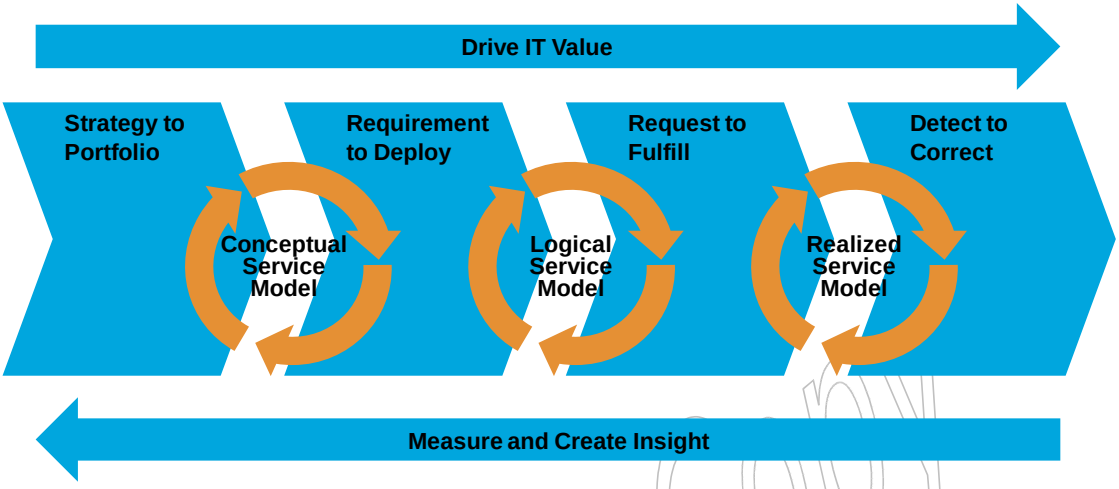


Figure 3: IT Value Streams and Service Models

Each value stream encapsulates the capabilities that are necessary to manage the service lifecycle. These capabilities are realized as a set of functional components and data objects. Functional components are the smallest technology unit that can stand alone and be useful as a whole to a customer. Functional components must have defined input(s) and output(s) that are data objects and it must change or advance a key data object (e.g., state change). Data objects represent tangible, non-trivial data items that are owned, consumed, produced, or modified by the functional components. Key functional components drive core activities within a value stream. Auxiliary functional components are not dedicated to a single value stream and provide relevant data objects to key functional components. Typically, functional components control a single data object or Service Model entity.

Relationships between functional components are controlled through input and output data objects. While the IT4IT Reference Architecture supports various types of integration, this document focuses only on the data-centric integrations referred to as “system of record”. System of record integrations ensure the consistent management of the lifecycle for individual data objects, as well as ensuring that the data objects are consistently named and cross-linked through prescriptive data flows between functional components.

The IT Value Chain and IT4IT Reference Architecture are focused on which functional components and data objects are on the critical path for IT departments to have in place to manage the service lifecycle. The IT Value Chain context is not an exercise in improving IT processes, IT capabilities, or existing industry standard best practice models. Instead, the purpose of the value chain framework is to drill down into the supporting framework upon which those processes and capabilities run and then detail the key elements within and across the value streams that will become critical in the tooling required by IT. The processes in use may change but the underpinning data objects and functional components required to define, develop, and release a service remain the same. Ultimately, IT departments can leverage this functional

component-based and data object-based model to define common tooling, drive more predictable quality and delivery, and ultimately realize these goals faster than trying to "go it alone".

While the IT4IT Reference Architecture is represented as an end-to-end lifecycle, its actual implementation supports iterative development. There is no assumption that any given development initiative follows all steps in the lifecycle; a stable IT service platform may receive repeated iteration on the R2D lifecycle, for example.

3.3 IT Value Streams

The IT Value Chain describes the IT service lifecycle. That service lifecycle is captured in the four IT Value Streams. The functional components within each value stream are responsible for creating, refining, and tracking key data objects across the service lifecycle. The relationships between the data objects that pass between the four value streams during the service lifecycle are well defined.

The following sections provide a short overview of the primary IT Value Streams in the IT Value Chain. Chapter 5 through Chapter 8 provide detailed descriptions for each value stream.

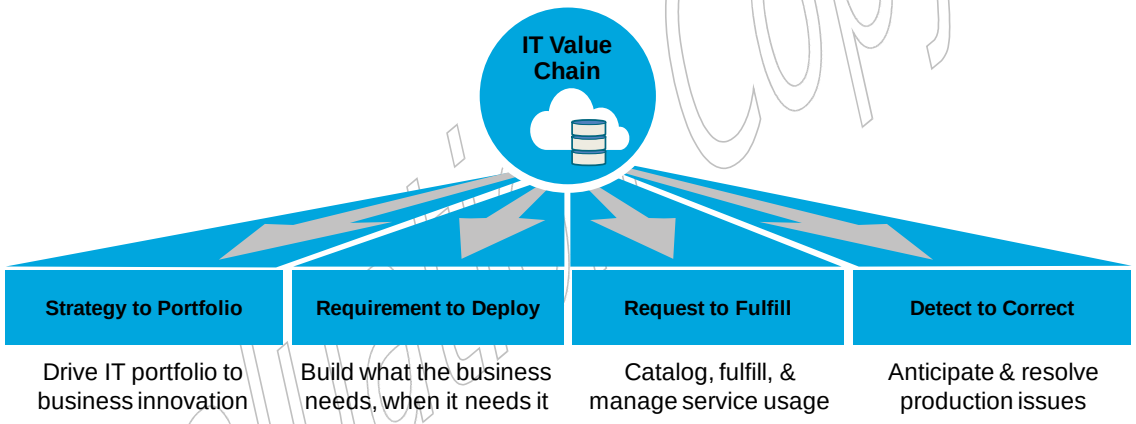


Figure 4: Value Stream Overview

3.3.1 Strategy to Portfolio

The Strategy to Portfolio (S2P) Value Stream provides IT organizations with the optimal framework for interconnecting the different functions involved in managing the portfolio of services delivered to the enterprise. Activities such as capturing demand for IT services, prioritizing and forecasting investments, Service Portfolio Management, and Project Management require data consistency and transparency in order to maintain alignment between the business strategy and the IT portfolio.

Traditional IT planning and Portfolio Management activities put emphasis on capturing and tracking a collection of *projects* that represent the "orders" from the business for technology enablement. The S2P Value Stream places emphasis on the *service* and aims to provide a more holistic view of the IT portfolio to shape business investment decisions and connect IT costs with business value.

The key value propositions for adopting the S2P Value Stream are as follows:

- Establish a holistic IT portfolio view across the IT PMO, and the Enterprise Architecture and Service Portfolio functional components so IT portfolio decisions are based on business priorities.
- Use well-defined system of records between the key areas that contribute to the IT Portfolio Management function to support consistent data for accurate visibility into business and IT demand.
- Endorse a Service Model that provides full service lifecycle tracking through conceptual, logical, and physical domains so it is possible to trace whether what was requested actually got delivered.

Typical activities include:

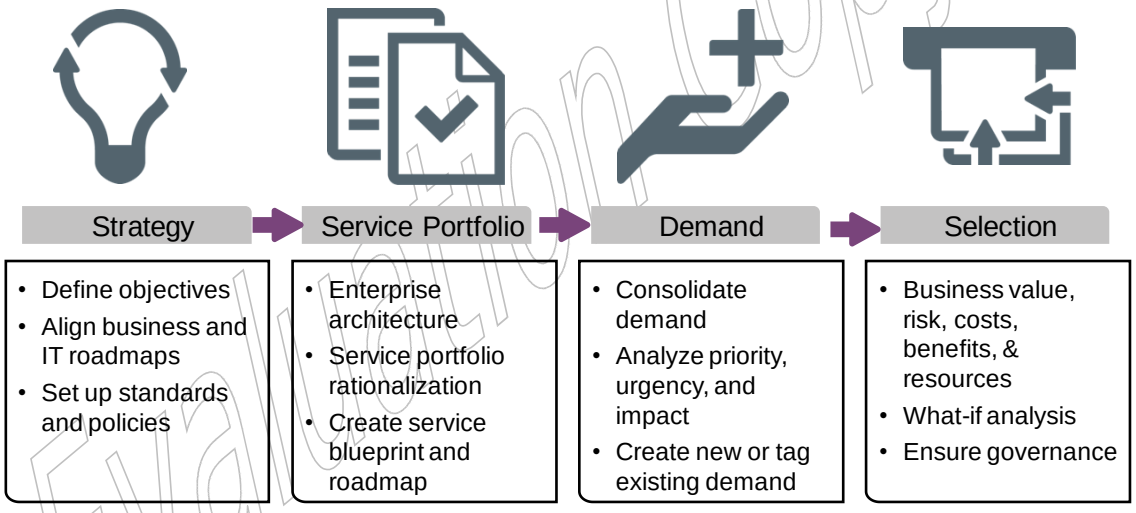
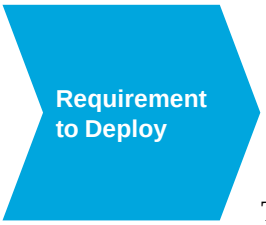


Figure 5: Strategy to Portfolio Activities

The end-to-end IT portfolio view provided by the S2P Value Stream is accomplished by focusing on the service as the desired business outcome and exposing key data objects often unavailable using traditional planning methods. Defining the key data objects, the relationships between them, and their effect on the Service Models is core to the value stream approach. In addition, it provides inter-dependent functions such as Portfolio Demand, Enterprise Architecture, Service Portfolio, and Proposal functional components with data consistency and predefined data object exchanges in order to optimize the organization's IT Portfolio Management and service lifecycle management capability.

3.3.2 Requirement to Deploy



The Requirement to Deploy (R2D) Value Stream provides the framework for creating/sourcing new services or modifying those that already exist. The goal of the R2D Value Stream is to ensure predictable, cost-effective, high quality results. It promotes high levels of re-use and the flexibility to support multi-sourcing. The R2D Value Stream is process-agnostic in that, while methods and processes may change, the functional components and data objects that comprise the value stream remain constant. Therefore, it is complementary to both traditional and new methods of service development like agile, SCRUM, or DevOps.

The R2D Value Stream consumes the Conceptual Service Blueprint produced in the S2P Value Stream and through a series of design, development, and testing functions enables the development of the Logical Service Model for the service. The Logical Service Model is elaborated on until it represents a release that can be commissioned into a production state using standard deployment methods or in an on-demand manner using a user-driven catalog experience. Once deployed into a production state, the Physical Service Model is generated that is comprised of the physical elements that comprise the service.

The key value propositions for adopting the R2D Value Stream are:

- Ensure that the Service Release meets business expectations (quality, utility).
- Make service delivery predictable, even across globally dispersed teams and suppliers, and multiple development methodologies while preserving innovation.
- Standardize service development and delivery to the point where re-use of service components is the norm.
- Build a culture of collaboration between IT operations and development to improve Service Release success.

Typical activities include:

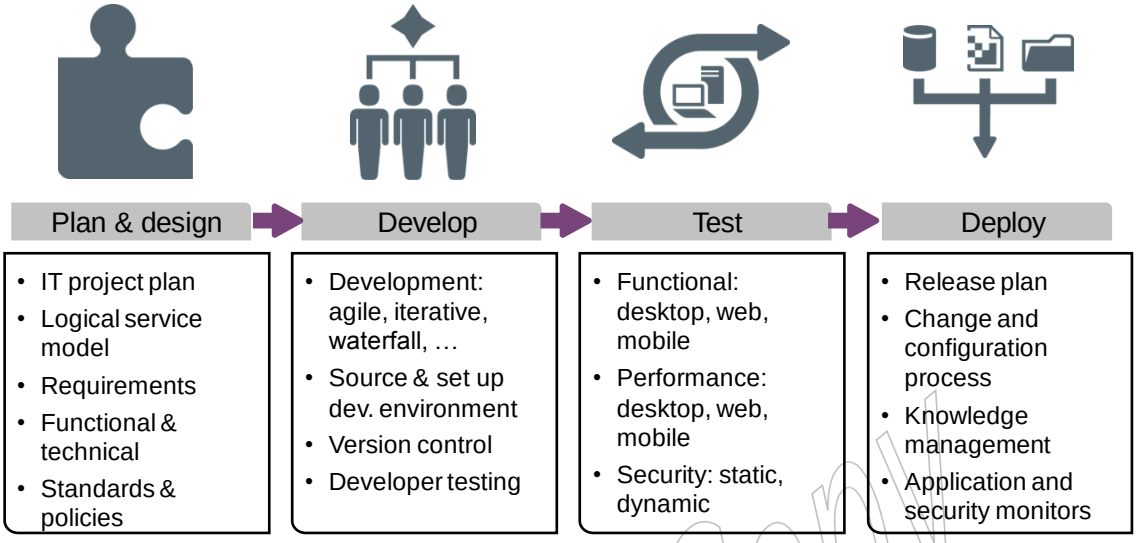
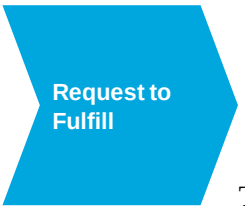


Figure 6: Requirement to Deploy Activities

3.3.3 Request to Fulfill



The Request to Fulfill (R2F) Value Stream is a framework connecting the various consumers (business users, IT practitioners, or end customers) with goods and services that are used to satisfy productivity and innovation needs. The R2F Value Stream places emphasis on time-to-value, repeatability, and consistency for consumers looking to request and obtain services from IT. The R2F Value Stream helps IT optimize both service consumption and fulfillment experiences for users by delineating functions for an Offer Catalog and Catalog Composition. The R2F Value Stream framework provides a single consumption experience to consumers for seamless subscription to both internal and external services, as well as managing subscriptions and routing fulfillments to different service providers using the R2F Value Stream framework.

The R2F Value Stream plays an important role in helping IT organizations transition to a service broker model. Enterprise customers have been using external suppliers for goods and services for many years. The IT multi-sourcing environment will accelerate as companies adopt cloud computing offerings like Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

The key value propositions for adopting the R2F Value Stream are:

- Provide a portal and catalog blueprint for facilitating a service consumption experience that allows consumers to easily find and subscribe to services through self-service, regardless of sourcing approach.

- Establish the model for moving from traditional IT request management to service brokerage.
- Increase fulfillment efficiency through standard change deployment and automation.
- Leverage the common Service Model to reduce custom service request fulfillments and design automated fulfillments.
- Facilitate a holistic view and traceability across service subscription, service usage, and service chargeback as applicable.

Typical activities include:

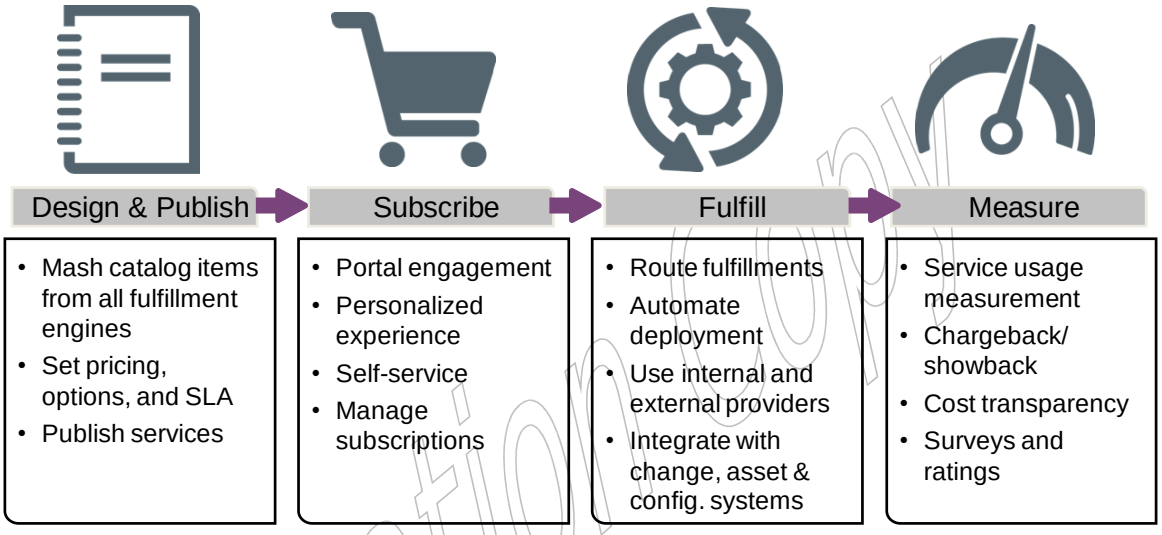
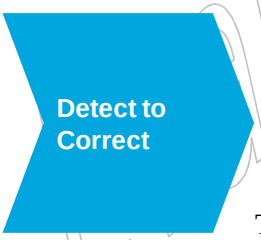


Figure 7: Request to Fulfill Activities

3.3.4 Detect to Correct



The Detect to Correct (D2C) Value Stream provides a framework for integrating the monitoring, management, remediation, and other operational aspects associated with realized services and/or those under construction. It also provides a comprehensive overview of the business of IT operations and the services these teams deliver. Anchored by the Service Model, the D2C Value Stream delivers new levels of insight which help improve understanding of the inter-dependencies among the various operational domains; including Event, Incident, Problem, Change Control, and Configuration Management. It also provides the business context for operational requests and new requirements. The D2C Value Stream is designed to accommodate a variety of sourcing methodologies across services, technologies, and functions. This value stream understands the inter-relationships and inter-dependencies required

to fix operational issues. It supports IT business objectives of greater agility, improved uptime, and lower cost per service.

The D2C Value Stream provides a framework for bringing IT service operations functions together to enhance IT results and efficiencies. Data in each operation's domain is generally not shared with other domains because they do not understand which key data objects to share and do not have a common language for sharing. When projects are created to solve this, it is often too difficult and cumbersome to finish or there is an internal technology or organization shift that invalidates the result.

The D2C Value Stream defines the functional components and the data that needs to flow between components that enhance a business and service-oriented approach to maintenance and facilitates data flow to the other value streams.

The key value propositions for adopting the D2C Value Stream are:

- Timely identification and prioritization of an issue.
- Improved data sharing to accelerate ability to understand the business impact.
- Automation both within domains and across domains.
- Ensuring an operating model, capabilities, and processes that can handle the complexity of service delivery across multiple internal and external domains.
- Effective linkage of Events to Incidents to Problems to Defects in the R2D Value Stream.

Typical activities include:

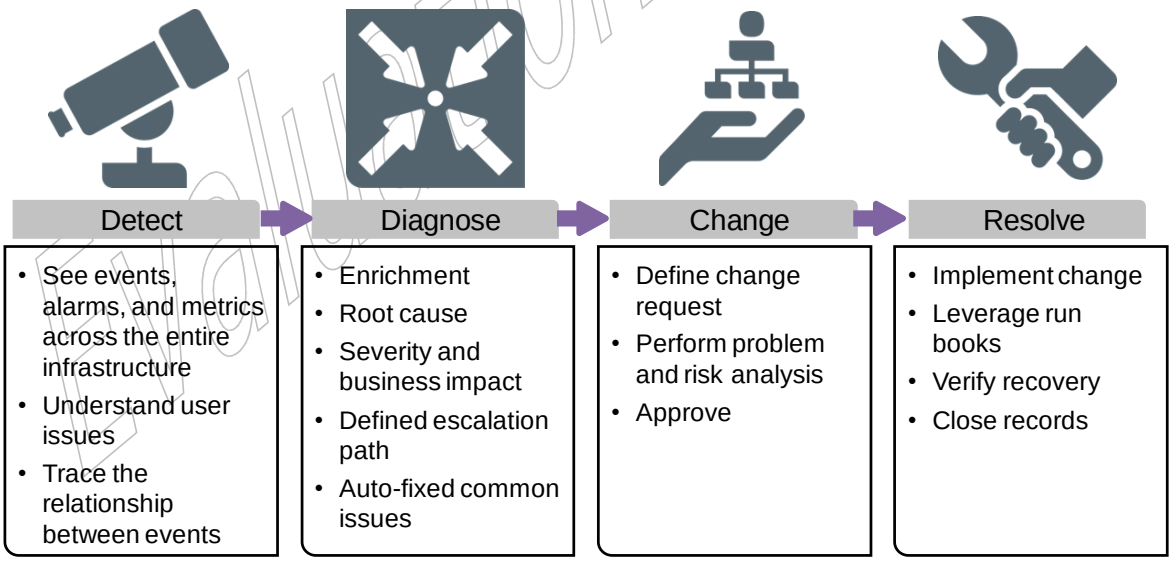


Figure 8: Detect to Correct Activities

3.4 IT4IT Reference Architecture

The IT Value Chain is supported by the IT4IT Reference Architecture. The IT4IT Reference Architecture provides a prescriptive framework to support the value chain-based IT operating model and service-centric IT management ecosystem. The service-centric IT management ecosystem comprises internal IT functions, external tool vendors, IT management improvement consultancies, external IT service providers (both XaaS and traditional consultancy and outsourcing services), external IT component provides (facilities, hardware, software, data), and training and certification providers. It can be thought of as describing the "IT for IT" (IT4IT) architecture and relationships.

Previous methods for creating an IT4IT Reference Architecture in the IT management domain have been oriented around processes, capabilities, and technology implementations. Unfortunately, processes can be implemented differently for each IT organizational archetype and capabilities are largely influenced by technology implementations. As a result, this architecture approach often results in a complex mesh of products and solutions requiring countless point-to-point integrations to accommodate the variations in process.

Instead, the IT4IT Reference Architecture approach for the IT Value Chain is anchored by four pillars – the Service Model, the Information Model, the Functional Model, and the Integration Model. These areas, when captured and modeled correctly, remain constant regardless of changes to process, technology, and/or capabilities.

3.4.1 Service Model

Without a clear understanding of both the business and technology attributes of a service, there is no way to be certain that the desired outcome can be consistently attained and that the most optimal sourcing strategy will be applied. The Service Model construct in the architecture captures, connects, and maintains these service lifecycle attributes as the service progresses through its lifecycle.

Traditional IT lifecycles are oriented around projects, used to govern technology deployments. Therefore, the plan, build, run lifecycle is the stalwart for most IT organizations and very little data is captured and maintained for a service. The provider/broker model for the new style of IT places its focus on services as the primary IT deliverable and requires a higher degree of flexibility, velocity, and adaptability. A service-centric lifecycle framework is one that supports a continuous cycle of assessing the portfolio, sourcing and integrating components, and offering services to consumers. This requires greater degrees of control over the data associated with a service in its various stages.

In addition, while traditional IT tends to place the majority of emphasis on delivery or implementation of technology resources, the service-centric IT must balance its focus on consumption and fulfillment of services along with the integration and delivery of the technology components and resources.

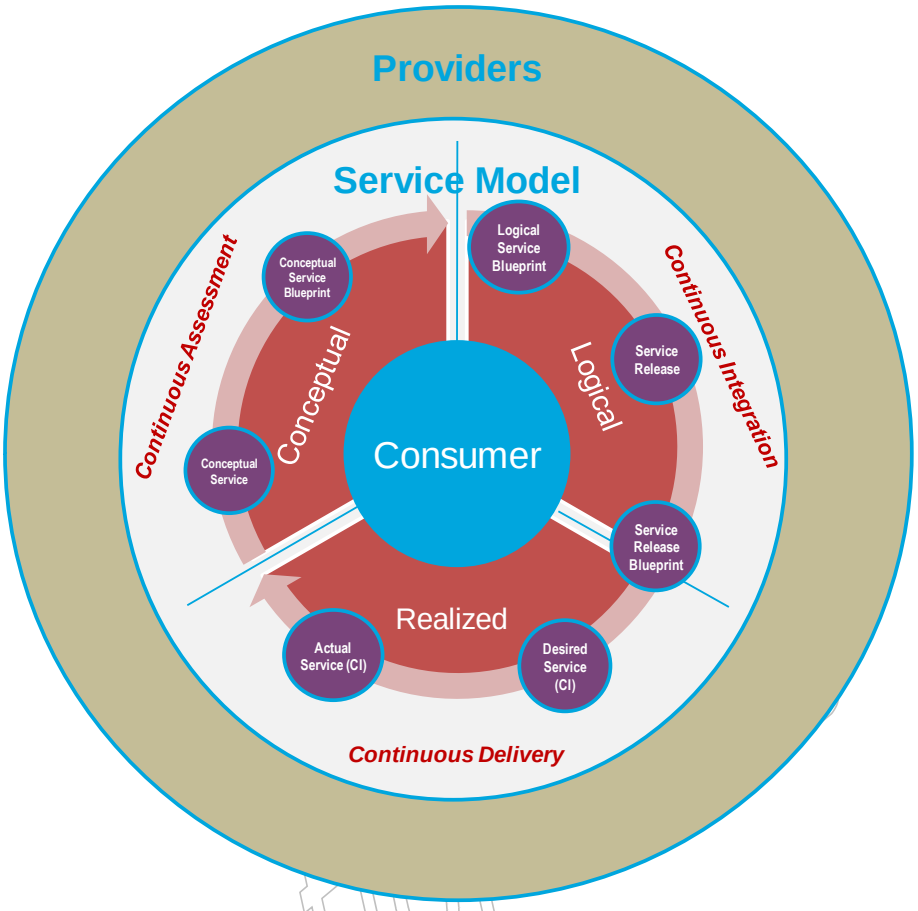


Figure 9: IT4IT Service Model

The structure that binds the different abstraction levels of the Service Model together is called the "Service Model Backbone" (shown in Figure 9). The Service Model Backbone provides the data entities, attributes, and necessary relationships between them to ensure end-to-end traceability of a service from concept to instantiation and consumption. This means that using this data-driven, model-based approach will ensure that what is required is actually what gets delivered or, in other words, what is offered will produce the outcome that the consumer desires. Further, it allows the IT organization to effectively utilize a service-centric approach in creating and packaging of its deliverables. This requires the creation of services from across resource and capability domains such as infrastructure, application components, database, middleware, monitoring, support, and so on. This enables the organization to improve their speed and consistency through higher re-use of pre-existing services and to embrace new technologies such as containers and micro-services in a more effective manner.

3.4.2 Information Model

The Information Model comprises the set of service lifecycle data objects and their relationships.

Each value stream produces and/or consumes data that together represents all of the information required to control the activities that advance a service through its lifecycle. This data has been referred to in prior releases of the IT4IT Reference Architecture as "service lifecycle *artifacts*";

however, as of Version 2.0 the term "service lifecycle *data objects*" (data objects in short form) has been adopted. Throughout the documentation there may still be some use of the term artifact, but when this occurs it should be interpreted as data object. Some data objects contribute directly to creating and/or advancing the Service Model while others serve as connectors, providing the linkage between functional components and across value streams.

Data objects have the following characteristics:

- They describe an aspect of an IT service.
- They are inputs or outputs associated with an IT4IT functional component or a service lifecycle phase.
- They are uniquely identified, and have a lifecycle of their own.
- They maintain structured information that allows for relationship tracking and automation.

Service lifecycle data objects in the IT4IT Reference Architecture are grouped into two categories: *key* and *auxiliary*.

Data Object Type	Description	Symbol
Key Data Objects	Key data objects describe aspects of "how" services are created, delivered, and consumed; they are essential to managing the service lifecycle. Managing the end-to-end service lifecycle and associated measurement, reporting, and traceability would be virtually impossible without them. The IT4IT Reference Architecture defines 32 key data objects and most are depicted as black circles.	
	Service models are a stand-alone subclass of key data objects that describe "what" IT delivers to its consumers. They represent the attributes of a service at three levels of abstraction: Conceptual, Logical, and Realized. These data objects are referred to as Service Model Backbone data objects (or service backbone data objects in short form) and depicted using a purple colored circle in the IT4IT Reference Architecture diagrams.	
Auxiliary Data Objects	Auxiliary data objects provide context for the "why, when, where, etc." attributes and, while they are important to the IT function, they *do not play a vital role in managing the service lifecycle*. The IT4IT Reference Architecture currently describes eight (8) auxiliary data objects and they are depicted using a gray colored circle.	

The essential relationships between data objects within and across value streams are defined in the IT4IT Reference Architecture. These relationships function as a prescriptive guide for ensuring the integrity of the Service Model as it progresses through its lifecycle and facilitate traceability across value streams. *The IT4IT Reference Architecture provides only the essential relationships and recognizes that there are other relationships that can exist but those are not part of the prescriptive guide.* Within the IT4IT Reference Architecture, the relationship between data objects is annotated as follows:

- 1 to 1 (1:1): implies that if there is a relationship, it is between two data objects. It does not imply that there will always be a relationship. For example, Events without Incidents or Incidents without Events are legitimate scenarios.
- 1 to many (1:n): implies that one data object relates (A) to one or more other data objects (B...) in scenarios where there is a relationship.
- Many to many (n:m): implies that both and A and B above can relate to zero, one, or many of the connected data objects.

Figure 10 provides an example of the relationship notation. The relationships and notation used here are for illustration purposes only and do not reflect the actual notation used or the relationship between these specific data objects.

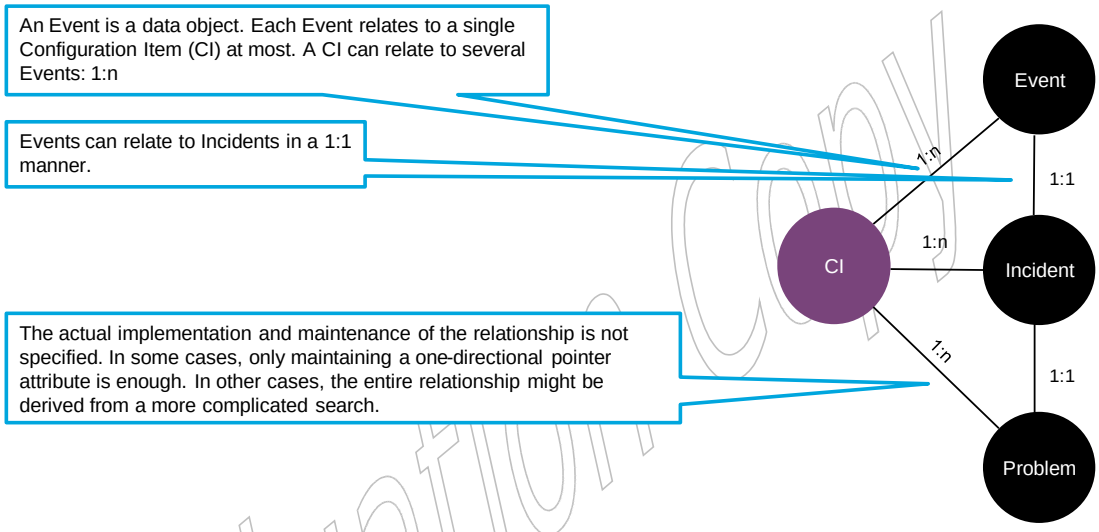


Figure 10: Data Objects and Relationships

Notes

1. Level 3 of the IT4IT Reference Architecture will utilize a full UML multiplicity specification for data object relationships (see Section 4.2).
2. In the IT4IT Reference Architecture notation, the multiplicity is always written horizontally (e.g., 1:1 in Figure 10). In some cases, the related entities are depicted vertically. When this occurs the general rules of mathematics should be applied to determine the relationship. This means the left position number/letter relates to the entity that is left or upward and the right position number/letter relates to the entity right or downward.

3.4.3 Functional Model

Based on real IT scenarios and use-cases, the IT4IT Reference Architecture identifies and defines one of the essential building blocks – functional components – which create or consume data objects and can be aligned with the appropriate value streams. The Functional Model is the set of functional components and their relationships.

The context for functional components starts with an IT "capability". A capability is the ability that an organization, person, or system possesses (function or activity it can perform) which produces an outcome of value through the utilization of a combination of people, process, methods, technology resources, and/or tools.

Functional components can be logically associated to IT capabilities for organizational clarity and will be underpinned with processes to drive uniformity and consistency.

While the definition of capabilities is used as a context for defining functional components, they are not the central focus within the IT4IT Reference Architecture. The documentation only refers to capabilities if there is a non-core capability that interacts with the architecture. In that case, using capability reduces the need to provide details for functional components and data objects that are outside the scope of the IT4IT Reference Architecture (for example, IT Financial Management).

Capabilities are supported by "building blocks" called functional components. A functional component is the smallest technology unit that can stand on its own and be useful as a whole to a customer. Functional components must have defined input(s) and output(s) that are data objects and must have impact on a key data object (for example, a state change). Functional components typically control a single data object. A grouping of one or more functional components represents the technology elements of an IT capability.

Capturing the architecture in this manner and constructing the ecosystem using this approach delivers on the agility promise and ensures end-to-end traceability by focusing on data rather than process as the design guide. Processes can change, organizational structures can shift (such as centralized/decentralized), and yet the Information Model remains constant and the architectural integrity of the IT management ecosystem is not compromised.

Functional Component Overview

Within the IT4IT Reference Architecture functional components are grouped into two categories: *primary* and *secondary*. In some of the IT4IT documentation primary functional components are also referred to as "key" and secondary as "auxiliary". The consistency around naming will be improved in future releases.

Functional Component Type	Description	Symbol
Primary Functional Component	A primary functional component is depicted using a blue colored rectangle and is core to a specific value stream. This means that the functional component plays a key role in the activities of a particular value stream. Without this functional component, the integrity of the data objects and thus the Service Model could not be maintained consistently and efficiently. Most IT4IT documentation will use language such as "a functional component is owned by or is core to a particular value stream" to represent a primary functional component.	

Functional Component Type	Description	Symbol
Secondary Functional Component	Secondary functional components are depicted using a gray colored rectangle and represent some level of dependency or interaction with a value stream and its data objects. While they interact with a value stream, they are not core to it and are either primary to another value stream or supporting function or represent a capability.	

Notes

1. There a few conditions when a functional component is core to or co-owned by more than one value stream (e.g., the Change Control functional component). When this occurs, the functional component is depicted using the blue colored rectangle in each value stream.

2. There is a uniqe condition in the R2F Value Stream where a relationship exists between functional components that is user experience-driven rather than data-driven. It is expected that this condition will manifest itself in other areas as well. To capture this condition, an informal notation using a gray box with a black outline is utilized, as depicted in Figure 11.

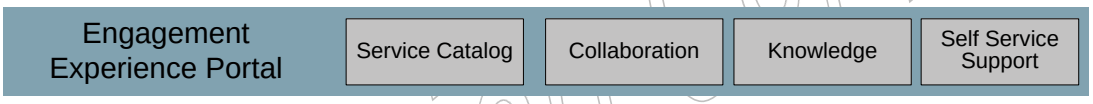


Figure 11: Engagement Experience Portal Functional Component

The relationships and dependencies between data objects controlled by functional components are depicted using a solid line along with cardinality mapping. In addition to the entity relationships, functional components interact and exchange data to form the relationship. The data exchange between functional components is depicted using a dotted-line arrow to represent the direction of the flow.

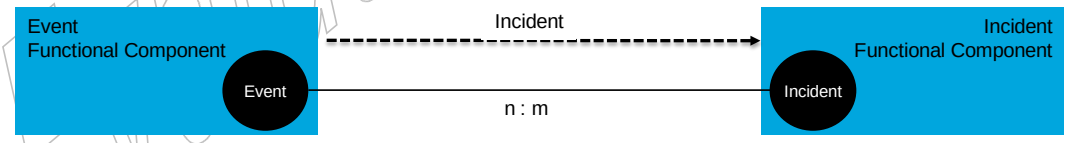


Figure 12: Data Flows in the IT4IT Reference Architecture

While the IT4IT Reference Architecture depicts data flow and relationships between functional components, it does not advocate or prescribe processes that make this happen. The IT4IT Reference Architecture is process-agnostic in that it provides a prescriptive framework that remains constant regardless of whether an organization utilizes ITIL, COBIT, eTOM, or internally developed processes.

3.4.4 Integration Model

Integration between components in the traditional IT management ecosystem is based on capability and processes. The interfaces needed to accommodate the requirements associated

with this approach are largely point-to-point between products to enable automation of inter-dependent workflows. Over time, this results in a complex web of connections that is virtually impossible to manage, making changes to any of the components a daunting task.

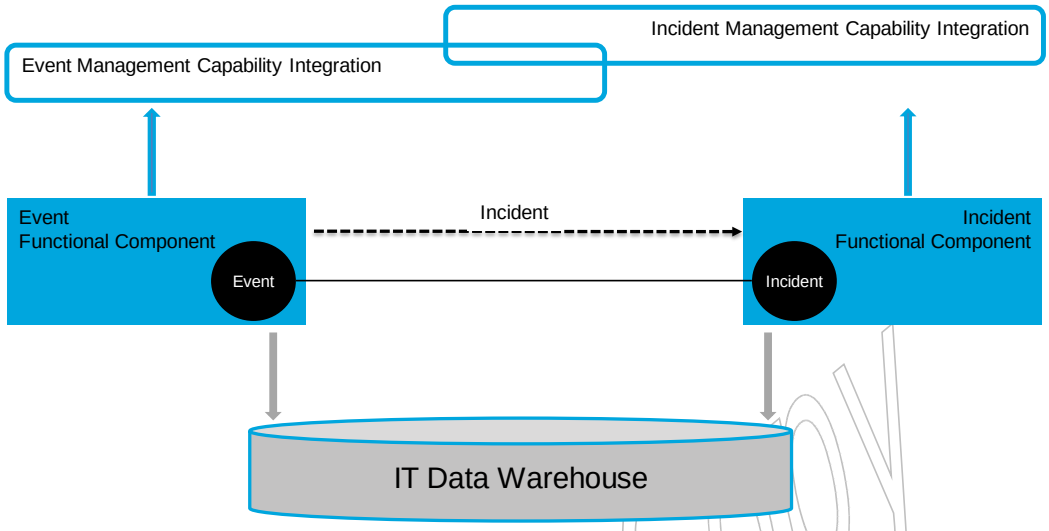


Figure 13: Engagement and Insight Information Flow

The IT4IT Reference Architecture defines an integration model composed from the following three types of integrations for simplifying the creation of an IT management ecosystem using functional components:

- System of record integrations (data-centric integration, SoR in short form): These entity relationship definitions ensure the consistent management of the lifecycle for individual data objects, as well as ensuring that the data objects are consistently named and cross-linked through prescriptive data flows between functional components to maintain the integrity of the Service Model. They are represented by a dotted black line like the one in Figure 13.
- System of engagement integrations (experience-centric integration, SoE in short form): These are user interface integrations derived from value stream use-cases and user stories. These integrations deliver the technology underpinning for a capability by combining several functional components into a single user experience to facilitate human interaction with data objects. In the IT4IT Reference Architecture system of engagement integrations are represented by the blue arrow in Figure 13. In the actual notation, system of engagement integrations are depicted using a dotted blue line.
- System of insight integrations (intelligence, analytics, and KPI-centric integrations, SoI in short form): These are data-centric integrations driven by the need to provide traceability, end-to-end visibility, transparency, and to capture measurements related to services (for example, performance) or the service lifecycle (for example, fulfillment time). Further, these integrations can accommodate the exchange of both structured and unstructured data that is created across the value chain. System of insight integrations are represented by the gray arrow in Figure 13. The actual notation for system of insight integrations has not yet been defined as the architecture to support them will be developed in future releases.

Therefore, while this integration is mentioned here, they do not appear in the current version documentation.

4 IT4IT Core

4.1 Introduction

The IT4IT Reference Architecture provides a "standard, repeatable model" for creating an IT management ecosystem, set in the context of a value chain-based IT operating model. It is intended to help organizations adapt to changes in technology, process, and methods without having to re-factor the management architecture to accommodate every shift. It is also intended to function as design guidance for suppliers of IT management products and services.

This chapter does not provide the details of the IT4IT Reference Architecture, but uses portions of it for illustrative and example purposes. For a comprehensive overview of the IT4IT Reference Architecture and associated content, please see the appropriate documents contained in the IT4IT website at www.opengroup.org/it4it.

4.2 IT4IT Abstraction Levels and Class Structure

4.2.1 IT4IT Abstractions

The IT4IT Reference Architecture is communicated using multiple levels of abstraction which is similar to the approach employed by other frameworks such as eTOM from the TM Forum. Each abstraction level expands on the prior to expose more details and prescriptive guidance. The upper levels (1-3) are vendor-agnostic and provide more generic views that are suitable for strategy and planning purposes as well as for creating IT management product roadmaps. The lower levels (4-5) provide more specific details, ultimately arriving at implementation level or vendor-owned/controlled information. Content at these levels is suitable for developing implementation plans and for facilitating product design. The IT4IT Reference Architecture defines five abstraction levels as depicted below:

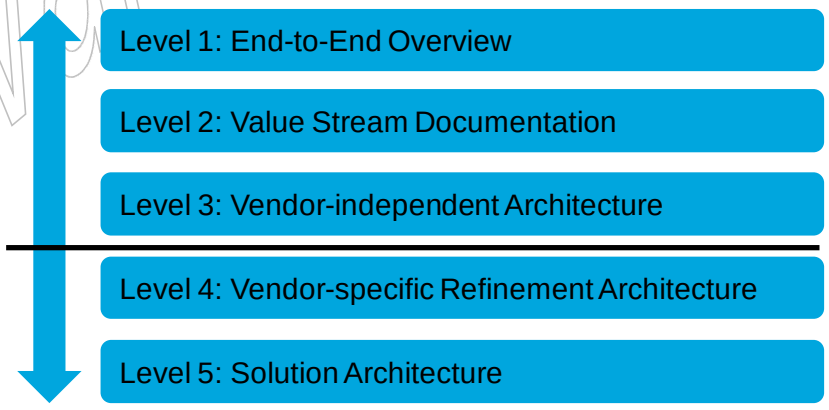


Figure 14: IT4IT Reference Architecture Levels

The balance of this chapter provides an overview for how to interpret the concepts and architectural constructs introduced at each level of abstraction for the IT4IT Reference Architecture. Levels 1 and 2 utilize a simplified class model and an informal notation to introduce and explain concepts. *An informal notation was selected at these levels so that the architecture would be easily understood by non-architects.* Where appropriate, the associated formal notation, expressed in the ArchiMate modeling language or UML terms, is also depicted. Level 3 and beyond are more applicable to architects and thus the formal notation is the preferred means of communicating concepts at this level. The IT4IT Reference Architecture focuses on documenting and governing Levels 1 to 3. Levels 4 and 5 are controlled by product and service providers or potentially other forums, and the IT4IT Reference Architecture merely provides exemplar guidance at these levels.

Most of the terminology used in the architecture is based on or derived from existing industry standard taxonomies. The architecture does introduce some new terms but this is only done when no other existing term could be utilized to express the intended meaning of the concept.

4.2.2 Concepts at Level 1: End-to-End Overview

Abstraction Level 1 is considered the "overview level". It provides a holistic model of the IT4IT Reference Architecture, introducing the core terms and concepts that underpin the architecture. It depicts, at a high level, the foundation controls an IT organization needs in place to standardize and automate and manage the IT Value Chain.

At this level five core concepts are introduced that are central to the IT4IT Reference Architecture body of work:

- Value Streams
- Functional Components
- Service Lifecycle Data Objects (key data objects)
- Service Model Backbone Data Objects (service backbone data objects)
- Relationships

The relationships among these five concepts are shown in Figure 15. (Note that the graphic uses UML notation to depict the multiplicity).

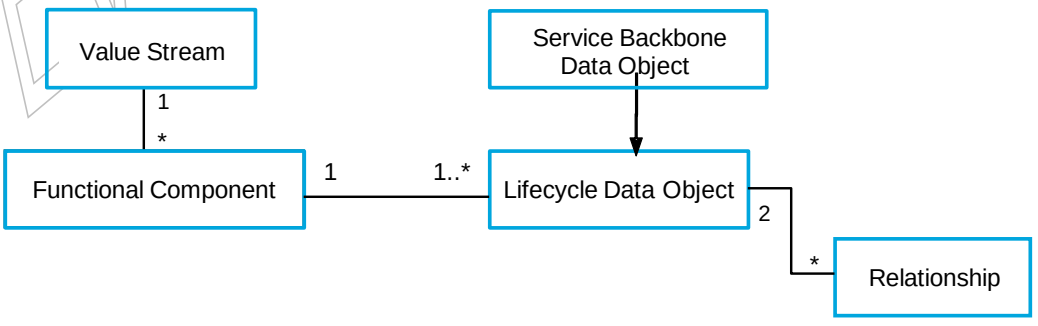


Figure 15: Level 1 Class Model

An important point to understand from the class model in Figure 15 is that all data objects are considered service lifecycle data objects. This means that the "service backbone data objects" that are used to represent the Service Model are a type of lifecycle data object. The IT4IT documentation treats service backbone data objects differently because they play a special role in the Reference Architecture.

Also notice that capability is missing from the diagram. While capability mapping remains an important activity for IT organizations, it is not included as part of the normative documentation. Instead, other documents (guidance documents) will provide this level of detail. The objective of the IT4IT Reference Architecture is to convey, in a prescriptive fashion, the key data objects, relationships, and components that are foundational for all IT organizations.

The following sections expand on each of the core concepts introduced in Level 1.

4.2.2.1 Value Streams

The IT4IT Reference Architecture uses the value stream construct as a way of grouping the functional components and data objects together to provide context for where value is being created/delivered. There are four value streams within the model:

- Strategy to Portfolio (S2P)
- Requirement to Deploy (R2D)
- Request to Fulfill (R2F)
- Detect to Correct (D2C)

There are also supporting functions within the value chain model such as Supplier Management, Asset Management, Human Resource Management, Legal, and IT Financial Management.

Note: While there are various opinions on the concept of process, this standard does not define the value streams to be a macro-process. As stated previously, the IT4IT Reference Architecture does not dictate a process or capability model. Further, the architecture has been tested to work equally well in lean, agile, and traditional waterfall process scenarios.

Notation and Naming

In the formal notation, value streams can be modeled in the ArchiMate language as "Business Functions". However, in the informal notation at Levels 1 and 2 they are only used to depict a grouping of the functional components (Figure 22).

4.2.2.2 Functional Components

A functional component is the smallest unit of technology that can stand on its own and be useful as a whole to an IT practitioner (or IT service provider). It must have defined input(s) and output(s) that are data objects and must have an impact on a key data object; for example, create, update, delete. Typically, a functional component controls and/or manages a single type of data object but this is not dictated by the architecture.

In the IT4IT Reference Architecture, functional components are aligned with specific value streams and supporting functions. Components aligned with a given value stream are considered its "primary" functional components. Functional components that affect key data objects for a given value stream but aren't primary to that value stream are considered "secondary" functional components. The Reference Architecture uses different colors to distinguish between primary and secondary functional components (see Section 3.4.3 for additional details).

Examples of functional components include Event, Project, Defect, etc. The IT4IT Reference Architecture contains only those functional components that have an impact on key data objects. There will be other technology components and management systems that are used by IT organizations in the normal course of business but those are not considered to be part of the prescriptive IT4IT Reference Architecture. Examples of these types of components could include corporate finance systems, human resource applications, and contracts management systems.

Notation and Naming

The IT4IT Reference Architecture uses both formal and informal notation style and a visual syntax to depict functional components. At Level 1 the informal notation renders primary functional components as blue rectangles (see Figure 16). Secondary functional components are rendered as gray rectangles. Functional components are represented formally in the ArchiMate language using the "Application Component" type.

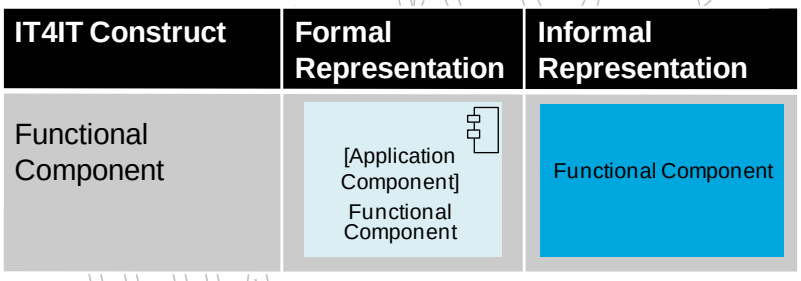


Figure 16: Functional Component Notation

While a functional component may control one or more data object types, a single data object can *only* be controlled by one functional component. To communicate this visually, an informal notation that embeds the "controlled" data object in the functional component is used, as depicted in Figure 17.

Figure 17: Functional Component to Data Object Notation

Diagrams can be complex and require clear "visual syntax" to help the reader interpret the elements and their relationships. Text is used to describe the emphasis of the functional

component. Figure 18 depicts the informal notation of the functional components and their associated data objects.

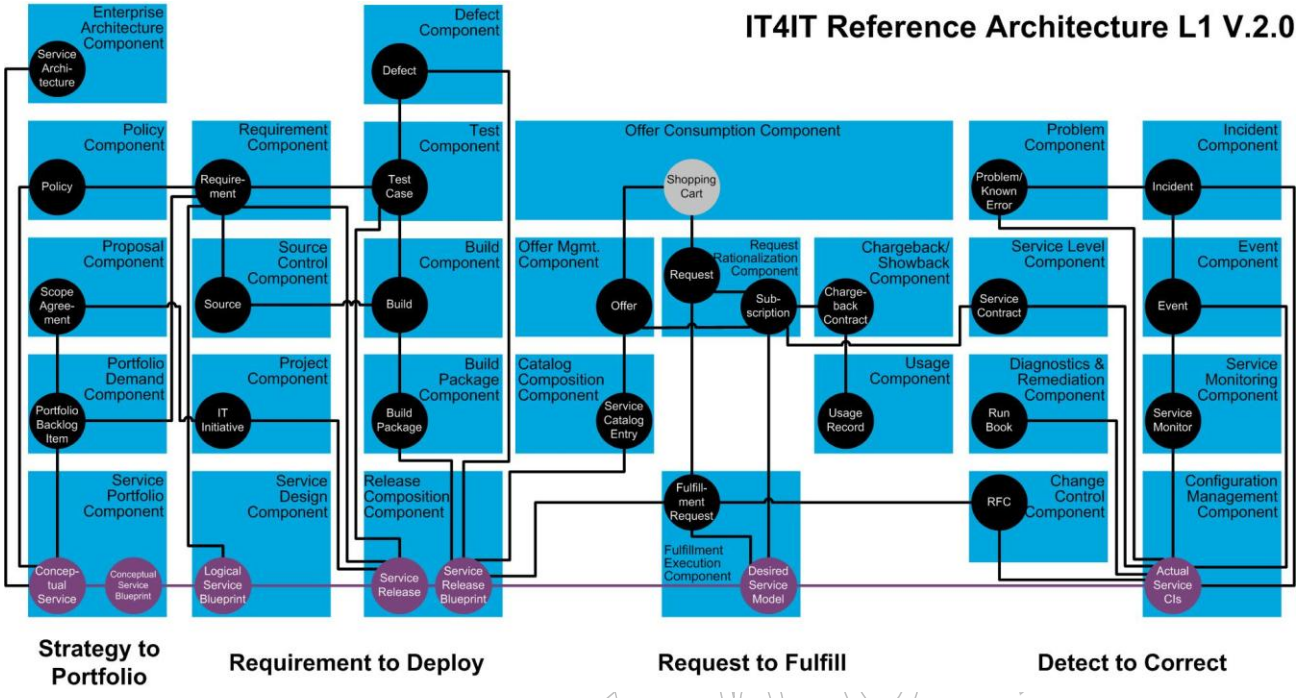


Figure 18: Functional Components and Data Objects

4.2.2.3 *Service Lifecycle Data Objects*

A service lifecycle data object (lifecycle data object) represents data (records, information, and so on) that annotate or model an aspect of a service being offered by IT. Data objects can take a digital or physical form and can be comprised of structured, semi-structured, or unstructured data. Our definition of service lifecycle data object is aligned contextually with the OMG definition of artifact. In UML, the OMG defines artifact as:

"… the specification of a physical piece of information that is used or produced by a software development process, or by deployment and operation of a system. Examples of artifacts include model files, source files, scripts, and binary executable files, a table in a database system, a development deliverable, or a word-processing document, a mail message."

Examples of data objects include incident records, training videos, requirements documents, project plans, etc. These types of data objects contribute in some way to the service over the course of its lifecycle. There are also "special" data objects that represent an abstraction or view of a specific service being offered. For example, a system diagram, Service Catalog Entry, or a binary executable file provide a "view" of a service to various personas at different stages of the service lifecycle. These special data objects and their relationships form a "backbone" that binds together all of the information needed to offer and manage a service, thus they are referred to as Service Model Backbone data objects (backbone data objects).

The central focus of the IT4IT work has been to identify the data objects that play a "key" role in the service lifecycle. In other words, the Reference Architecture focuses exclusively on the data

objects that are mandatory for ensuring end-to-end traceability of the service operationally and/or financially. There are other data objects used across the IT function to facilitate various needs or activities (e.g., company-specific processes), but these are considered to be secondary or auxiliary and outside the prescriptive control of the IT4IT Reference Architecture.

Notation and Naming

As with functional components, the IT4IT Reference Architecture utilizes both formal and informal notations for data objects so that the work can be easily understood by both architects and non-architects. In the Level 1 model an informal notation that depicts data objects as circles is used. Color is used to specify whether a data object is key (black) or a backbone data object (purple). Data objects are modeled formally in the ArchiMate language using the "Data Object" type.

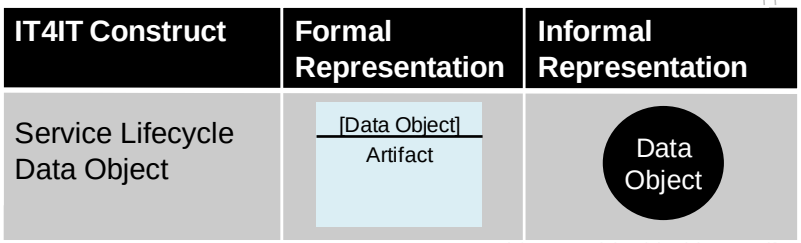


Figure 19: Service Lifecycle Data Object Notation

Text is also important in communicating the concepts represented in the model. According to dual-coding theory, using text and graphics together to convey information is more effective than using either on their own. Further, the IT4IT Reference Architecture attempts to avoid using text that creates collisions between architectural layers (e.g., business function, data, and implementation layers). Level 1 introduces naming conventions for data objects that are rooted in the well-known IT frameworks and standards.

4.2.2.4 Relationships

The service lifecycle data objects represent the authoritative source data and/or master data for managing the business of IT. In IT taxonomy, "system of record" is a term that is commonly used as a synonym for an authoritative source system.

An important aspect of the IT4IT Reference Architecture is defining not only the data objects, but the essential relationships between them. The data objects, combined with their relationships and inter-dependencies, form the "system of record fabric" for IT management (see Figure 20). These relationships are referred to as system of record integrations (record-centric, SoR in short form). Compliance with the prescribed system of record relationship mapping will ensure end-to-end traceability operationally and financially, and ensure Service Model integrity can be maintained across the lifecycle. This also eliminates the confusion and collisions that result from process re-engineering efforts that frequently occur within the various IT functions.

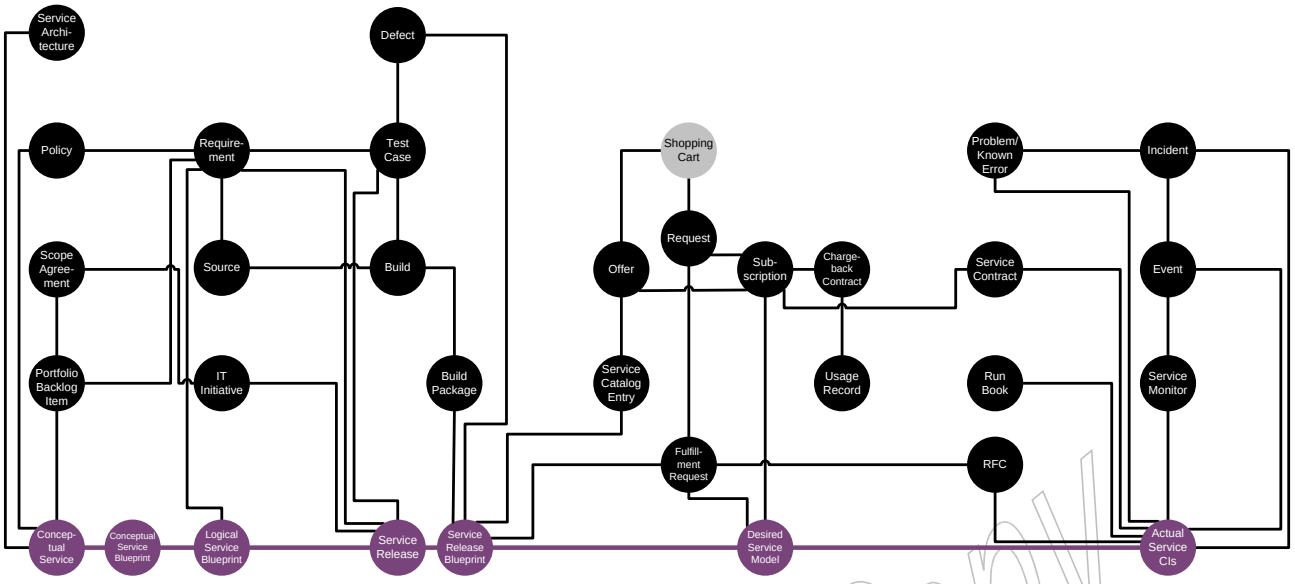


Figure 20: System of Record Fabric

Notation and Naming

Abstraction Level 1 expresses the essential relationships as lines between the participating data objects. Here, the formal and informal notation is identical; a line connecting one data object with another.

IT4IT Construct	Formal Representation	Informal Representation
Data Object Relations	[Data Object] Artifact — [Data Object] Artifact	Data Object — Data Object

Figure 21: Relationships Notation

4.2.3 Level 1 Reference Architecture Model

Figure 21 depicts the Level 1 model for the IT4IT Reference Architecture rendered using the informal notation. In discussions on the IT4IT Reference Architecture this graphic should be used to represent the content contained in abstraction Level 1.

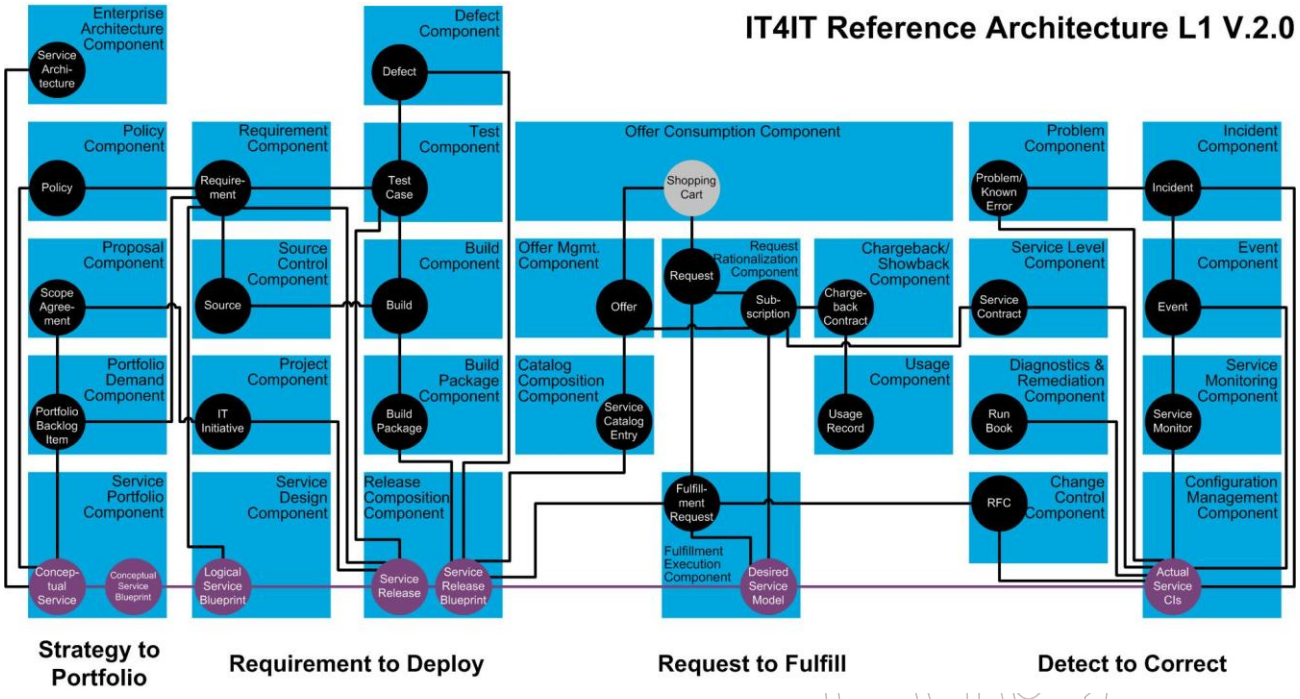


Figure 22: IT4IT Level 1 Reference Architecture Model

4.2.4 Concepts at Level 2: Value Stream Documentation

Abstraction at Level 2 expands on the concepts introduced in Level 1, providing definitions and details and introducing a few more concepts, including:

- Relationships between data objects are updated with multiplicity/cardinality attributes (e.g., one-to-one, one to many, many-to-many).
- The concept of data flow between functional components is introduced.
- The data flows are refined to depict integrations to build out the system of record fabric.
- The relationships between capability disciplines and functional components are introduced but they are not part of the normative reference and are presented as guidance.

The Level 1 class diagram in Figure 15 is expanded to reflect these additions in Figure 23 below. Note that the terms system of record and system of engagement integration are explained later in this section.

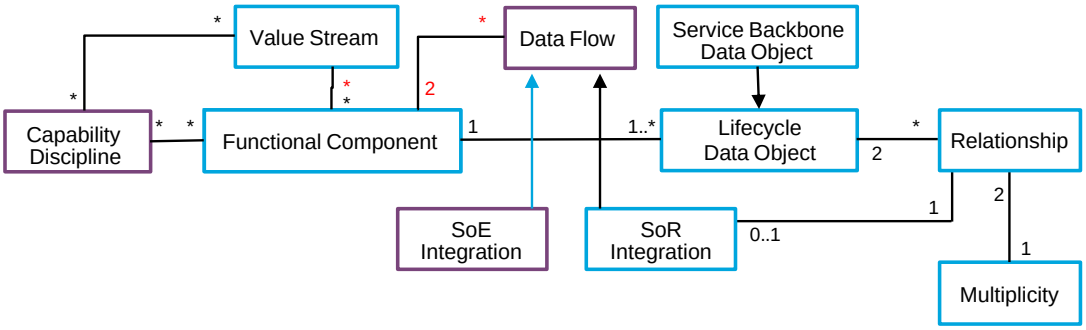


Figure 23: Level 2 Class Model

The Level 2 concepts are communicated through a set of normative documents and simplified views into the definitive vendor-independent model maintained at Level 3.

4.2.4.1 Multiplicity

Multiplicity (sometimes referred to as "cardinality") describes the number of allowable instances of an element. Multiplicity intervals have a lower bound and a (possibly infinite) upper bound. In the IT4IT Reference Architecture the term is used to describe the relationship between data object instances.

For example, there is a one-to-*n* (1:*n*) relationship between "request" and "request fulfillment" data objects. This indicates that a single request can result in multiple fulfillments. In the instance of a "laptop", a service request might require a request for the laptop and another for a mouse and one for a user account. Therefore, the one request generated multiple request fulfillment data objects.

The IT4IT Reference Architecture only defines the key relationships – those that contribute to the advancement of the service lifecycle. There are other relationships that may be needed to satisfy the specific policies, processes, or capabilities but they are not considered to be part of the prescriptive guidance.

Notation and Naming

For clarity, we are explicitly showing one-to-many relationships using UML notation. An information notation is also used to describe multiplicity at Levels 1 and 2. Again, the informal notation is used so that it can be understandable to non-architects.

IT4IT Multiplicity	Formal Representation	Informal Representation
One to no more than one	0..1:0..1	1:1
One-to-many	0..1:*	1:n
Many-to-many	*..*:*	n:m

4.2.4.2 *Data Flow*

A data flow describes the flow of information between functional components. The technique used to facilitate the flow is not specified. The reason for adding the data flow information to the Level 2 diagrams is to expose the need for integration and data sharing. It also helps to demonstrate the dependencies between functional components and how they can work together to deliver value.

The data flows are not considered part of the normative specification of the IT4IT Reference Architecture and are offered at Level 2 as guidance to help interpret the various value stream diagrams.

Notation and Naming

An informal notation is used for data flows, drawn as a dotted line with a solid end-arrow indicating the direction of the flow. The data/information being transferred should be depicted on the line.

Figure 24: Data Flow Notation

4.2.4.3 *System of Record (SoR) Integration*

In the Level 1 "Relationships" section (Section 4.2.2.4) the concept of authoritative source system or system of record was introduced. In order for the system of record relationships to be sustained over the course of the service lifecycle, integrations between functional components controlling the key data objects must be well defined. This requires that some data flows be "refined" into system of record integration specifications. These specifications become part of the normative IT4IT Reference Architecture.

For a data flow to be refined into a system of record integration, the following conditions must be satisfied:

- The data flow creates a relationship/dependency between two data objects in the functional components involved in the data flow.
- The two data objects must have a direct relationship to be maintained.
- The lifecycles of the two data objects are interlocked. This means that updates to one data object can have consequences on the other data object.

The reason for defining these integrations as part of the architecture is to ensure the integrity of the system of record fabric and the Service Model. IT organizations are likely to implement products from multiple suppliers for their IT4IT Reference Architecture and without this specification there is no way to ensure consistency in data flows.

A system of record integration is depicted in Figure 25. This figure provides an illustrative example that describes the state dependency between the "Event" and "Incident" data objects. Here, Events generate Incidents that results in a relationship and dependency between these two data objects.

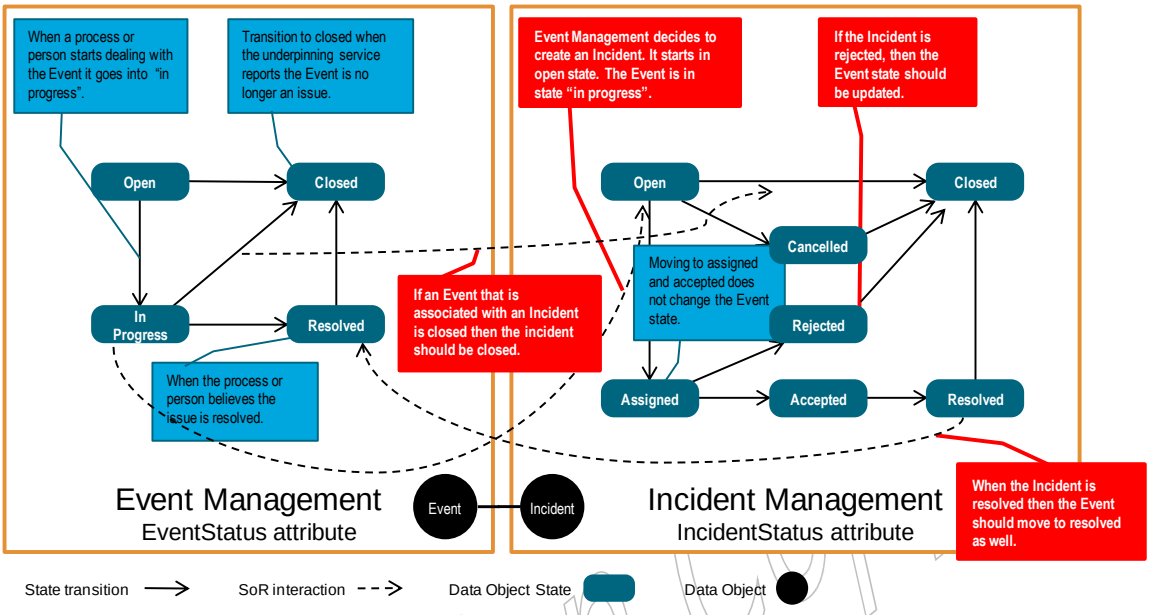


Figure 25: Data Object State Model Dependency Illustration

4.2.4.4 *System of Engagement (SoE) Integration*

Relationships are also formed between functional components to enable humans and machines to interact with and/or take action on the data objects. These relationships are called system of engagement integrations (experience-centric integration, SoE in short form). These integrations are derived from value stream use-cases and user stories and are offered as guidance rather than prescriptive practices. The intent is to demonstrate to the reader how data object relationships stimulate and/or are affected by human-to-machine and machine-to-machine interactions. Figure 26 provides an illustration of system of engagement integrations. It is merely an illustration and not an exact depiction of specific integrations in the IT4IT Reference Architecture.

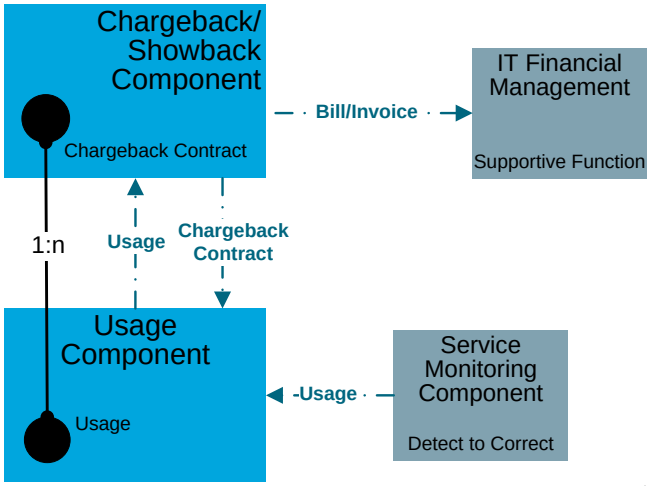


Figure 26: System of Engagement Integration Illustration

Naming and Notation

At Level 2 the informal notation depicts system of record integrations using a black dotted line with solid end-arrow accompanied by the name of the data object that is transferred to create the relationship. No subsequent information is shown as being passed on an ongoing basis to maintain or update either data object. The formal notation in the ArchiMate language uses the collaboration and interaction concepts. System of engagement integrations are only depicted using the informal notation and are rendered as a blue dotted line with a solid end-arrow accompanied by the name of the data object being acted upon.

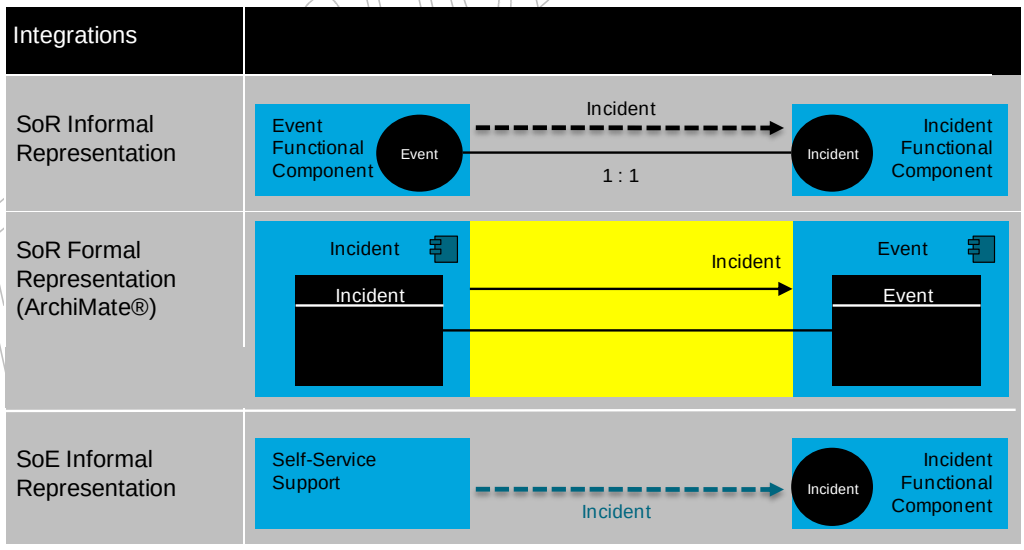


Figure 27: System of Record and Engagement Integration Notation

4.2.4.5 *Capability Discipline*

A capability is the ability of an organization to produce an outcome of value through the utilization of people, process, and technology. A list of capabilities at the discipline level of granularity (capability disciplines) is being compiled. These come from ITIL, COBIT, SAFe, PMBOK, and other industry sources. It is not the focus or intent of the IT4IT Reference Architecture to redefine or modify the existing work that has been done around IT capabilities; instead a comprehensive list is being compiled and it defines the relationships between functional components and capability disciplines.

Capability discipline documentation is considered "guidance" and not part of the normative IT4IT Reference Architecture.

Notation and Naming

While capability disciplines are not part of the normative reference, both an informal and formal notation are used to describe them for consistency across the architecture. The informal notation uses green boxes to depict primary capability disciplines (or discipline capability) and solid lines with solid end-arrows to show relationships and inter-dependencies. Secondary (or auxiliary) capability disciplines are depicted using gray boxes. Interfaces to "other" capabilities are depicted using white boxes.

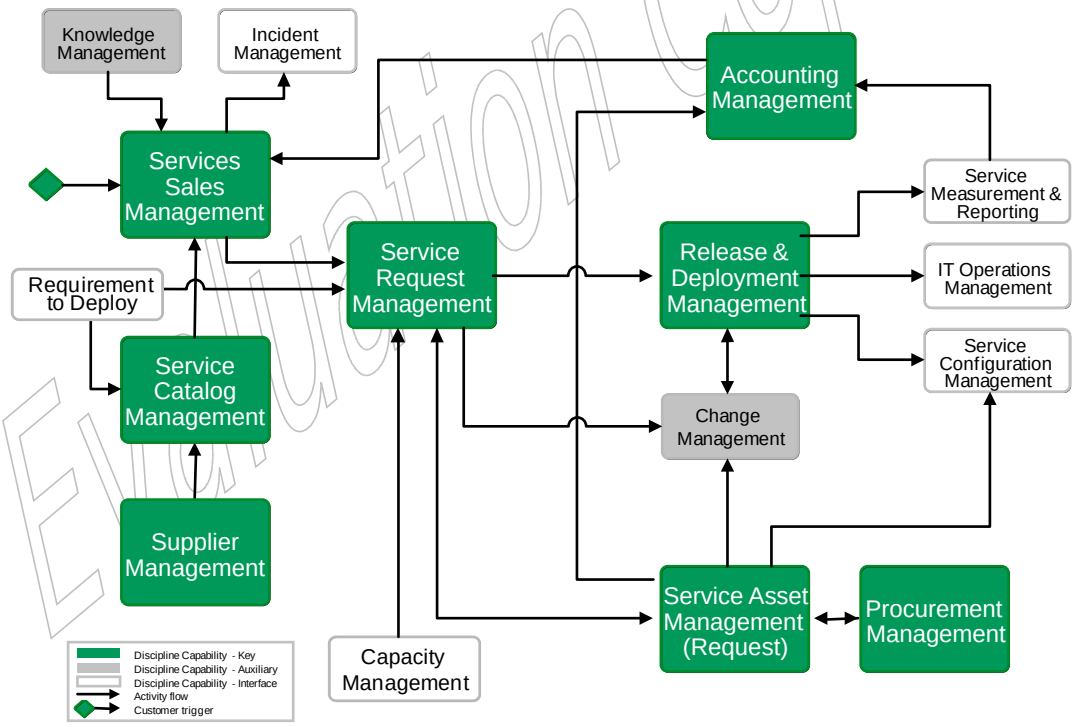


Figure 28: Capability Discipline Informal Notation

At the time of approval of this standard, the ArchiMate language does not currently recognize the concept of "capability" or "discipline", so for now in the formal notation, the "business function" construct has been utilized.

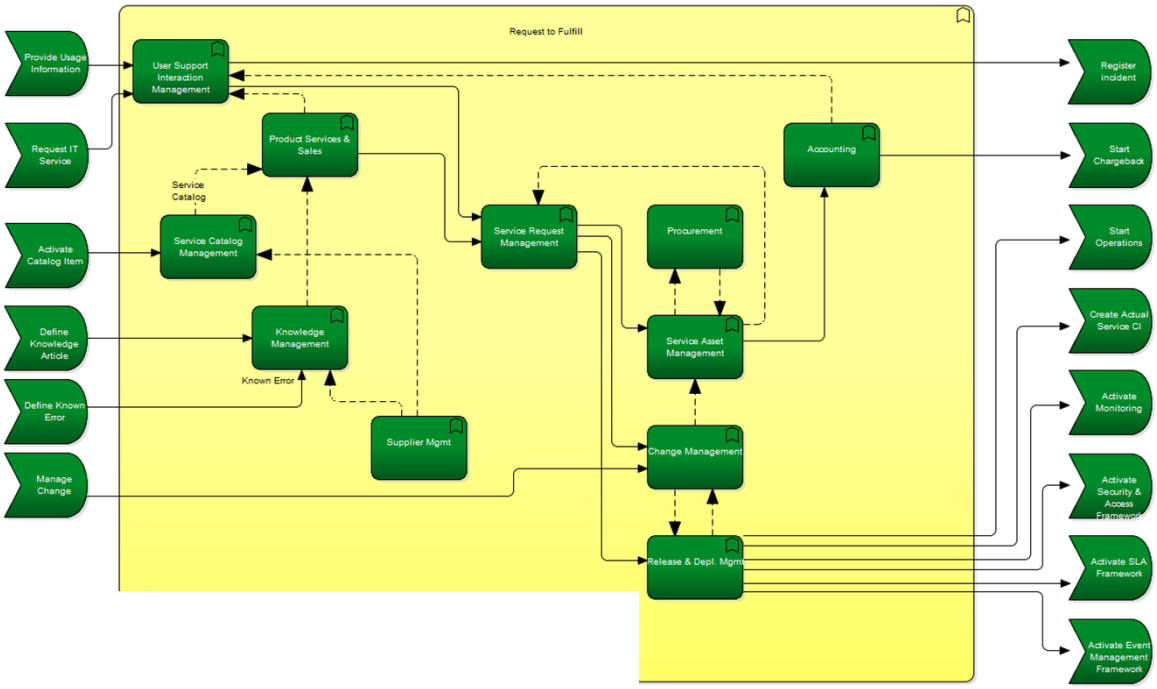


Figure 29: Capability Discipline Formal Notation

4.2.5 Level 2 Reference Architecture Diagram (Example)

Figure 30 provides an example of a Level 2 model/diagram using the informal notation. This example is the Request to Fulfill (R2F) Value Stream and in discussions on the IT4IT Reference Architecture this graphic represents the content contained in abstraction Level 2.

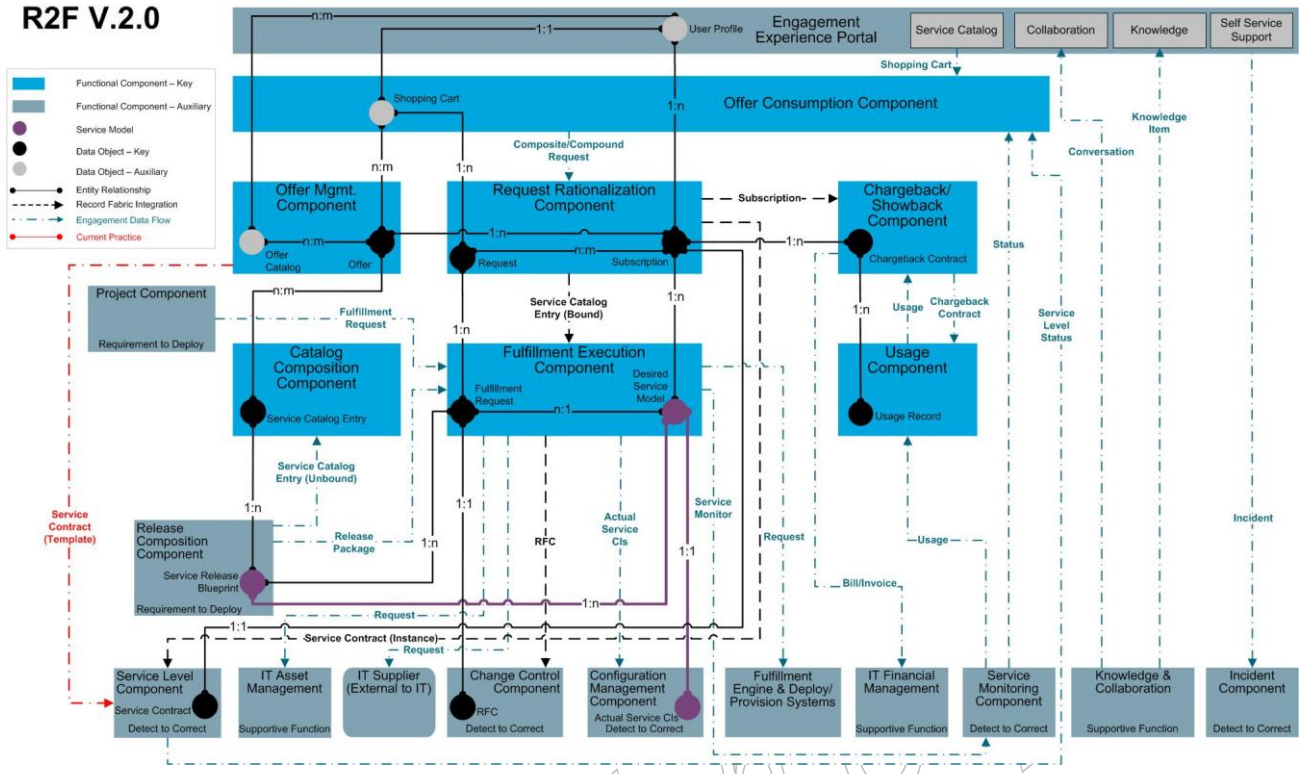


Figure 30: Example Level 2 Diagram (R2F Value Stream)

4.2.6 Concepts at Level 3: Vendor-Independent Architecture

Levels 1 to 2 use the informal notation as the primary means of communicating concepts to a general audience of non-architects; the formal notation being provided for the architect community. At abstraction Level 3 this changes and the formal notation (ArchiMate language and UML) is the primary method for communicating the IT4IT Reference Architecture specification. Level 3 adds more details for data object definitions, introducing essential attributes for key data objects. It also introduces the concepts of scenarios and essential services. Figure 31 shows the additions to the IT4IT Reference Architecture class model at Level 3.

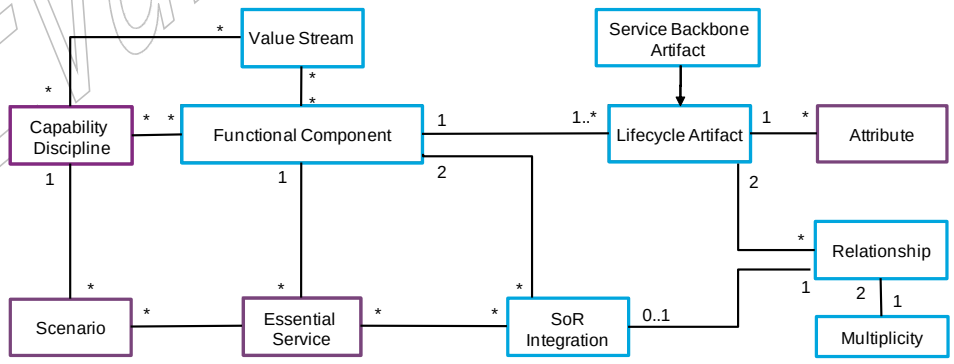


Figure 31: Level 3 Class Model

4.2.6.1 Scenarios

In the IT4IT Reference Architecture, a scenario is a narrative that describes foreseeable interactions of user roles (or "actors") and a system (or functional component). The term is analogous with "epics" or "theme" in agile development methodologies. Scenarios are used to explain, enhance, or modify the Reference Architecture and are described using a structured template that includes a formal notation that can be expressed in the ArchiMate language. This enables readers to consume the information more easily as multiple organizations may produce and/or contribute to scenarios over time.

The following list is an example of the "scenario master document content":

- **Introduction**: High-level description of the scenario to be described along with the goal to be achieved.
- **Requirements**: Requirements model the properties of the elements that are needed to achieve the "ends" that are modeled by the goals. In this respect, requirements represent the "means" to realize goals.
- **Process Flow**: An explanation on the process flow in the organization. This example explains how an Incident is handled using existing Knowledge.

Note: Process is not part of the normative reference in the IT4IT Reference Architecture.

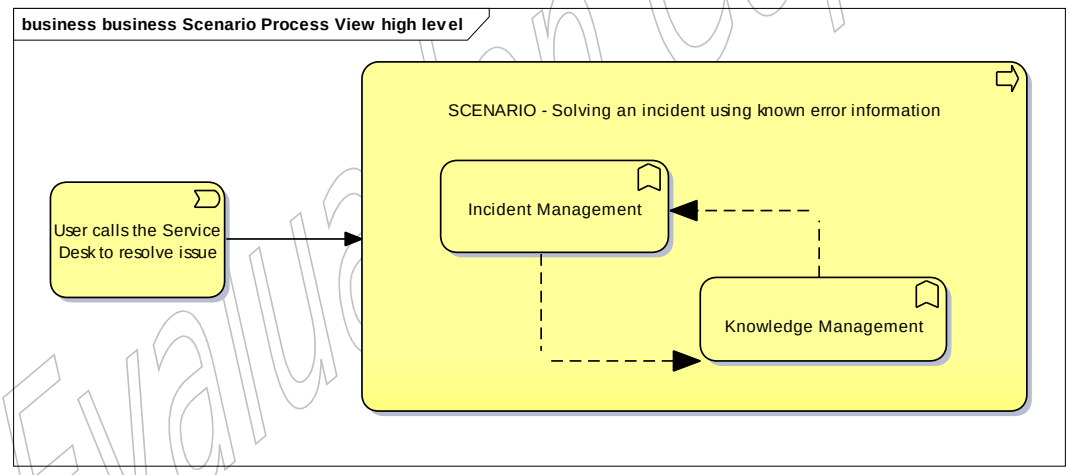


Figure 32: Scenario Process Flow Example

- **Automation Specification using the Reference Architecture**: Explains how the process is to be supported by the Reference Architecture. In the example two views would be created: one showing an Incident Management capability and another view for the Knowledge Management capability. The view should provide more information on how the functional components are aligned with the capabilities to automate the scenario.
- **Essential Services Supporting the Scenario**: See Section 4.2.6.3.
- **Data Objects and Essential Attributes**: Describes the data objects that are involved and the attributes of the data objects that are needed or impacted.

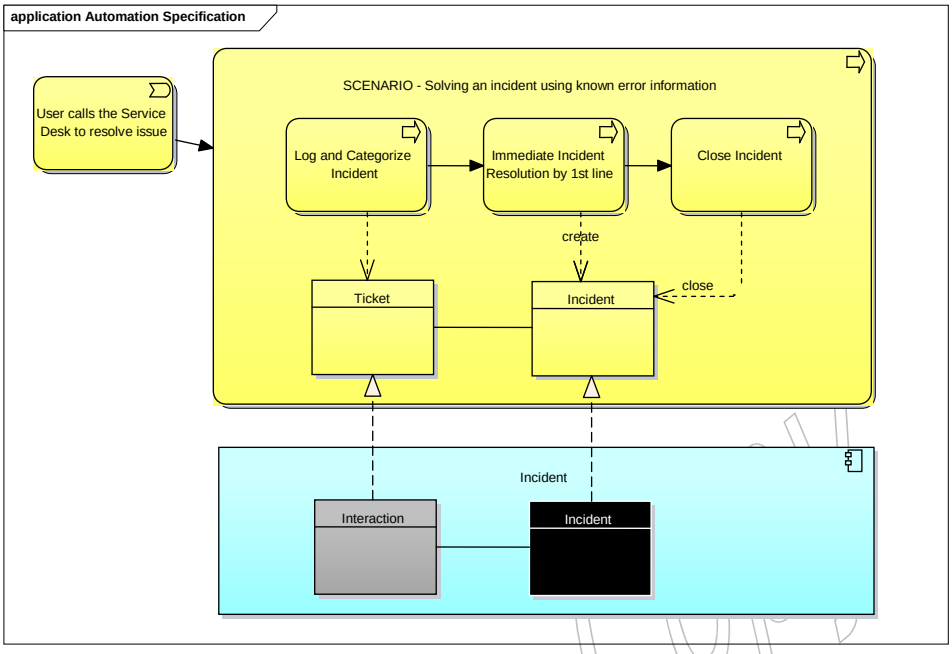


Figure 33: Essential Service Diagram Example

- **Proposed changes to the Reference Architecture**:

  Scenarios currently under construction as of Version 2.0 of the IT4IT Reference Architecture include the following:

  — IT Financial Management – IT service cost transparency

  — Agile Scenario – DevOps, also including the Kanban/SAFe Positioning White Paper

  — Multi-vendor Incident Management

  See also Section 1.6.

4.2.6.2 Essential Attributes

As described in Levels 1 and 2, data objects represent the data exchanged between functional components. At abstraction Level 3 it is important to specify the attributes of the objects that *must* be present (those deemed essential) in the exchange. Here, the minimum set of attributes that are required to maintain the integrity of the system of record fabric are defined. These also contribute to the formation of the canonical data model that underpins the IT4IT Reference Architecture. Often, IT management products will provide more information/attributes, but at Level 3 only the vendor-independent minimum that all should include is defined.

The most basic essential attributes are a “unique identifier” and the “data object lifecycle status”. Often there will be attributes that identify the relationships to other data objects in the IT4IT model.

For example, Figure 34 depicts the Event to Incident relationship. Here, the essential attributes that uniquely identify an Event and an Incident are shown. In addition there would be a data object lifecycle attribute that describes whether an Event is open, closed, in progress, or resolved. The Incident data object would have attributes that connect it with the related Event(s) and with potential problem records. It would also connect with the related Configuration Item (CI) data objects.

As of Version 2.0 of the IT4IT Reference Architecture, the specification is still being developed so there is no list of "approved" essential attributes and Figure 34 is merely an example included for illustrative purposes.

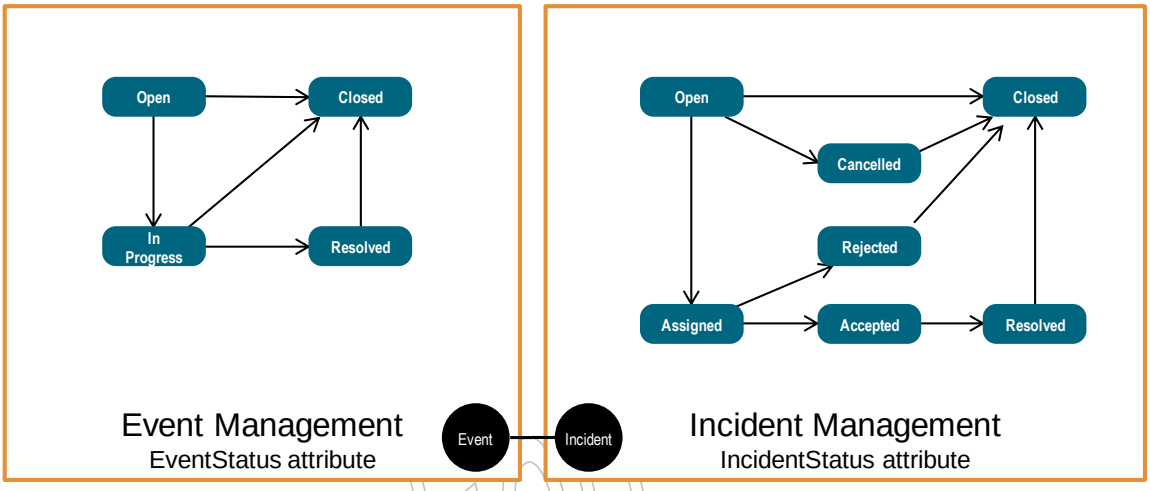


Figure 34: Essential Attributes Example

Notation and Naming

UML has been chosen as the notation for essential attributes (see Figure 35).

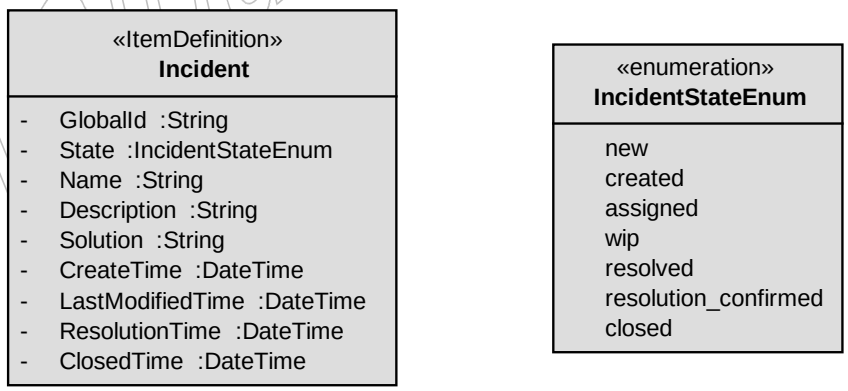


Figure 35: Essential Attributes Notation

4.2.6.3 *Essential Services*

In the IT4IT Reference Architecture, the essential service concept describes a means of facilitating integration between functional components or to take action on a data object (e.g., CRUD). Essential services are typically implemented as an API, web service, or micro-service with a well-defined set of parameters. The goal of essential services is to eliminate the need for point-to-point integrations between IT management products. They are defined by examining the data objects and specifying the attributes, parameters, etc. The result ultimately becomes part of the canonical data model.

As of Version 2.0, the specification is still being developed so there is no list of "approved" essential services. However, examples might include services such as: acknowledge event, register incident, manage incident, and so on.

Notation and Naming

Essential services can be modeled in the ArchiMate language using the "Application Service" construct. In the informal notation at Level 3 they are depicted as an oval inside a functional component element, as in Figure 36.

4.2.6.4 *System of Record Integrations and Essential Services*

Level 3 demonstrates how system of record integrations are implemented using essential services to maintain the relationship between data objects. Figure 36 depicts the essential services exposed by different functional components in the D2C Value Stream. Here how such integrations maintain the state and relationship between the Event and Incident data objects is shown.

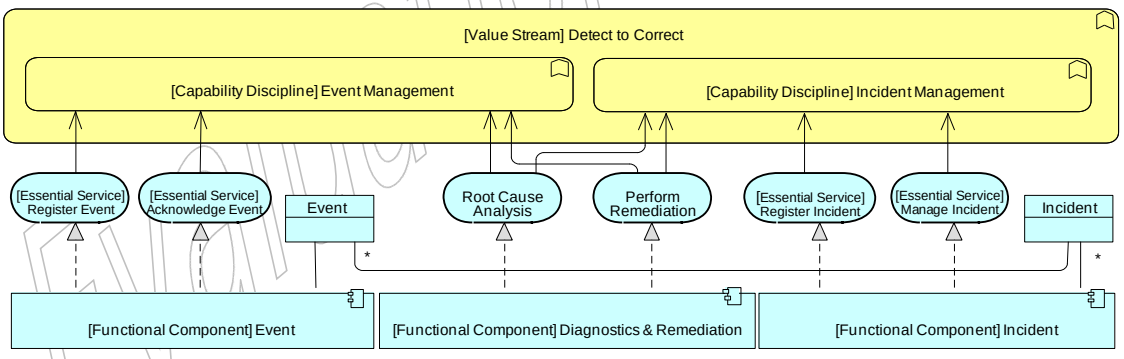
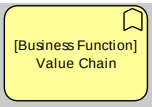
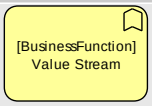
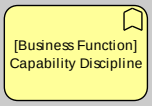
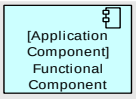


Figure 36: Example of Functional Components, Data Objects, and Essential Services

4.2.7 Level 3 Reference Architecture Diagram (Example)

Figure 37 provides an example of the Level 3 ArchiMate notation guidelines. Where constructs are used for multiple purposes the name reflects how it is being used. For example, the Business Function construct is used to represent Value Chain, Value Stream, and Capability Discipline and when this happens the name will reflect which entity is represented.

L3 Element	Representation
Value Chain	
Value Stream	
Capability (Discipline)	
Functional Component	

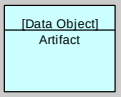
L3 Element	Representation
Artifact	
Essential Service	
SoR Integration	

Figure 37: Level 3 Notation Guide

4.2.8 Concepts at Levels 4 and 5 – Vendor Extensions

Levels 4 and 5 are owned and controlled by suppliers of IT management products and services. The IT4IT Reference Architecture has no direct control over defining and/or approving content at these abstraction layers. What is provided in this section are recommendations and examples of typical things would be expected at these levels.

For example, vendors might add essential services to the baseline or add functional components to differentiate their product or offering. The principle to be applied here is that whatever is added should build from and not change the prescriptive model defined in Levels 1 to 3.

4.2.8.1 Level 4

Abstraction Level 4 is where the architecture becomes more product design and implementation-oriented. Here, for example, providers of IT management products and services can design/specify their service, interface, and exchange models which should be derived from Level 3 content. Other examples of Level 4 content might include:

- Defining extensions to the standard – "*these essential attributes are being used, but these are added for the following reasons …*"
- Adding data objects – "*these non-key data objects were added, and are using the same notation style to reflect how they build off the baseline architecture*"
- Additional notations – "*the ArchiMate language is used for explaining scenarios and UML for the data models*"
- Introduction of process – might introduce/model practitioner-level processes within scenarios
- Canonical data model – might introduce the vendor-specific canonical data model for their IT management products

- Integrations – might specify the techniques/methods used in implementing system of record integrations

Regardless of the how the vendor chooses to adapt and implement the architecture, it must be able to be mapped back into what is specified at Levels 1 to 3.

4.2.8.2 Level 5

Abstraction Level 5 provides vendor-specific representations for an implementation of part or all of the Reference Architecture. It is the level at which the specifications and/or functionality for IT management products and services are provided. Deviations and adaptations to the Reference Architecture are also documented here. Examples of content at Level 5 might include:

- The structure used by vendor X to implement the D2C Value Stream.
- Vendor Y uses the "timestamp" essential attribute on the Event data object but names it "timeofday".
- Vendor Z uses a product called ServiceXchange as the platform for essential services.

Notation and naming at Level 5 is vendor-owned/controlled but should reflect adherence to the baseline architecture and prescriptive guidance in Levels 1 to 3.

5 Strategy to Portfolio (S2P) Value Stream

This chapter provides an overview of the Strategy to Portfolio (S2P) Value Stream – one of four IT Value Streams that comprise the IT Value Chain. It describes the business context, objectives, and details behind the S2P Value Stream.

5.1 Objectives

The S2P Value Stream allows IT to contribute to business strategy and planning enabling IT alignment with business plans. Traditional IT planning and portfolio activities put emphasis on evaluation, approval, and delivery tracking of project proposals. Discussion with business often becomes minimizing costs for IT initiatives or assets rather than the value of services provided to business from IT. The goal of the S2P Value Stream is to create an IT portfolio framework that allows IT organizations to optimize services provided to business by bringing together multiple functional areas. For example, traditional project proposal or investment Portfolio Management driven by an IT PMO needs to work in conjunction with Service (or Application) Portfolio Management driven by Enterprise Architects or Service Portfolio Managers.

The S2P Value Stream aims to provide holistic views of IT portfolio activities through data integrations within multiple areas of the IT portfolio. These views provide better understanding of the inter-relationships among IT's many sub-domains, including the IT PMO, Enterprise Architecture, Application Management, IT Operations Management, and Information Security Management. Today's IT needs accurate and point-in-time information to understand the inter-relationships and inter-dependencies required to truly orchestrate all the moving parts of IT in ways that can help the IT department support business objectives and goals.

Most IT organizations already have IT portfolio processes and solutions in place, but suffer from the following limitations:

- Poor data consistency and quality
- No holistic IT portfolio view across the IT PMO and the Enterprise Architecture and Service Portfolio functional components
- Inconsistent Service and IT Portfolio Management processes
- Poor tracking and correlation of service lifecycle across conceptual, logical, and physical domains

The S2P Value Stream provides a blueprint for optimizing service and investment IT Portfolio Management. The end-to-end IT portfolio view provided by the S2P Value Stream raises the visibility of key data objects often overlooked during IT portfolio planning activities. Defining key data objects and relationships between data objects and Service Models is easier with a proper framework.

5.2 Business Value Proposition

The S2P Value Stream enables IT organizations with the proper framework for interconnecting different functions supporting IT Portfolio Management. Functions such as the Portfolio Demand, Enterprise Architecture, Service Portfolio, and Proposal functional components need data consistency and proper data object hand-offs in order to optimize the organization's IT Portfolio Management.

The key value propositions for adopting the S2P Value Stream are:

- Holistic IT portfolio view across the IT PMO and the Enterprise Architecture and Service Portfolio functional components
- IT portfolio decisions based on business priorities
- Accurate visibility of business and IT demand
- IT portfolio data consistency
- Service lifecycle tracking through conceptual, logical, and physical domains
- Prioritized IT investment based on all IT portfolio facets including cost/value analysis, impacts on architecture, service roadmap, business priorities, etc.
- Re-balance IT investments between strategic and operational demand
- Solid communication with business stakeholders through roadmaps

5.3 Key Performance Indicators

The S2P Value Stream critical success factors and Key Performance Indicators (KPIs) are as follows:

Critical Success Factors	**Key Performance Indicators (KPIs)**
Business and IT Alignment	Ratio of new *versus* maintenance service.
Accurate Visibility into Overall Demands from Business	Demand requests, types, and delivery per service % of overall IT budget that can be traced to formalized demand requests. Structured and rationalized Demand Management with ongoing efforts to minimize the number of demand queues that staff must respond to.

Critical Success Factors	Key Performance Indicators (KPIs)
Service Portfolio Rationalization	A formal Service Portfolio functional component process exists under the ownership of the Service Portfolio Management process owner. Taxonomies for understanding functional and technical redundancy and business value of the IT service are implemented. Processes for consistently evaluating and tagging portfolio entries are implemented. Service portfolio is subject to ongoing rationalization using the taxonomies, implemented as continuous improvement. Service and IT Portfolio Management are themselves rationalized with clear scoping and relationship established.
Service Portfolio Financial Analysis	Accounting records are produced on a regular basis to show the ongoing "investment & spend" in each service/application. These are compared with business outcomes and financial objectives that have been achieved.
Service Portfolio Reporting and Analysis	A service portfolio exists and is used as the basis for deciding which services to offer.
Service Investment Tracking	The investment in each service is quantified in the service portfolio. Investment in each service is reported, starting with the initial investment, and followed by monthly, quarterly, or annual reporting of the ongoing budget spend (total cost of ownership).
Improve Customer Satisfaction	Satisfied customers per service/application.
Stewardship of IT Investment	CapEx *versus* OpEx. Software license percentage in use. Planned *versus* actual service costs. Average cost of IT delivery (per service/application) per customer.
Enterprise Security Alignment	Frequency of security assessments against latest standards and guidelines. Noted deficiencies against security standards and policies.

5.4 Value Stream Definition

The Strategy to Portfolio (S2P) Value Stream contains both key and auxiliary functional components. Key functional components are the core and drive key activities within a value stream. Auxiliary functional components are not dedicated to a single value stream and provide relevant data objects to key functional components. The S2P Value Stream includes the following key functional components that provide the technology necessary to support IT portfolio activities:

- Enterprise Architecture

- Policy
- Proposal
- Portfolio Demand
- Service Portfolio

A functional component utilizes an input data object to conduct key activities and may also provide output data objects to another functional component that requires that data.

The S2P Value Stream functional components are often owned and managed by different IT groups. For example, Enterprise Architects' primary tools and solutions are different than IT PMOs', but they must work together to optimize IT Portfolio Management. The S2P Value Stream provides a framework that ensures functional components used by IT groups can work together efficiently, through well-defined control points and data objects, to govern and model IT Portfolio Management.

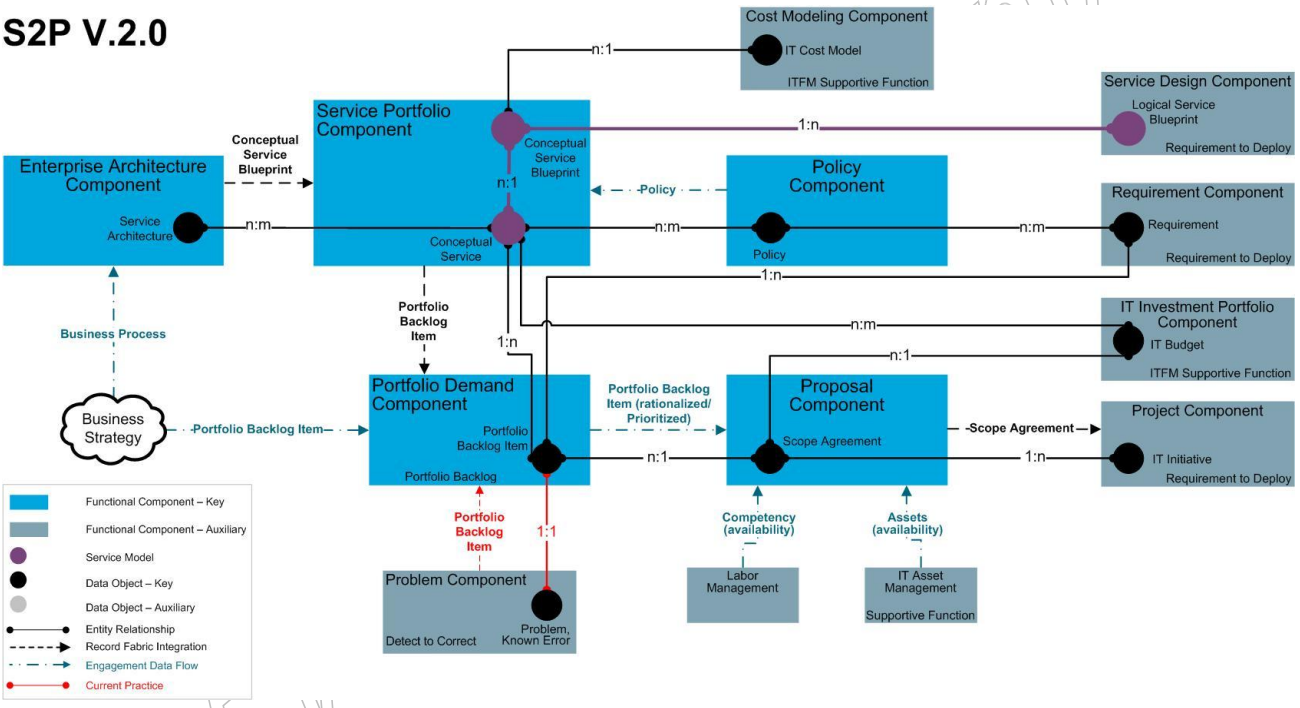


Figure 38: Strategy to Portfolio Level 2 Value Stream Diagram

5.4.1 Enterprise Architecture Functional Component

Purpose

Create and manage long-term IT investment and execution plan-of-action that are critical to business strategic objectives.

Key Data Objects

- **Service Architecture** (data object): A data object within the S2P Value Stream managed by the Enterprise Architecture functional component. It includes service blueprints, enterprise guiding principles, and technology roadmaps.

Key Attributes

The Service Architecture data object shall have the following key data attributes:

- **ServiceID**: Unique identifier of the IT service.
- **ServiceComponent**: Architectural components required to enable the desired IT service (e.g., application, infrastructure, etc.).
- **ServiceDiagram**: As-is builds of service diagram that may vary by location or service lifecycle stage.

Key Data Object Relationships

The Service Architecture data object shall maintain the following relationships:

- **Service Architecture to Conceptual Service** (n:m): This relationship helps track which service components and service diagrams are allocated to which IT service(s).

Main Functions

The Enterprise Architecture functional component:

- Shall identify strategic IT architectural components based on current business vision, strategy, goals, and requirements.
- Shall develop target state business, information, application, technology, and security blueprints based on strategies, principles, and policies.
- Shall develop IT roadmaps based on business roadmap and input.
- Shall develop and maintain enterprise guiding principles.
- Shall manage IT architecture guideline and standards.

Model

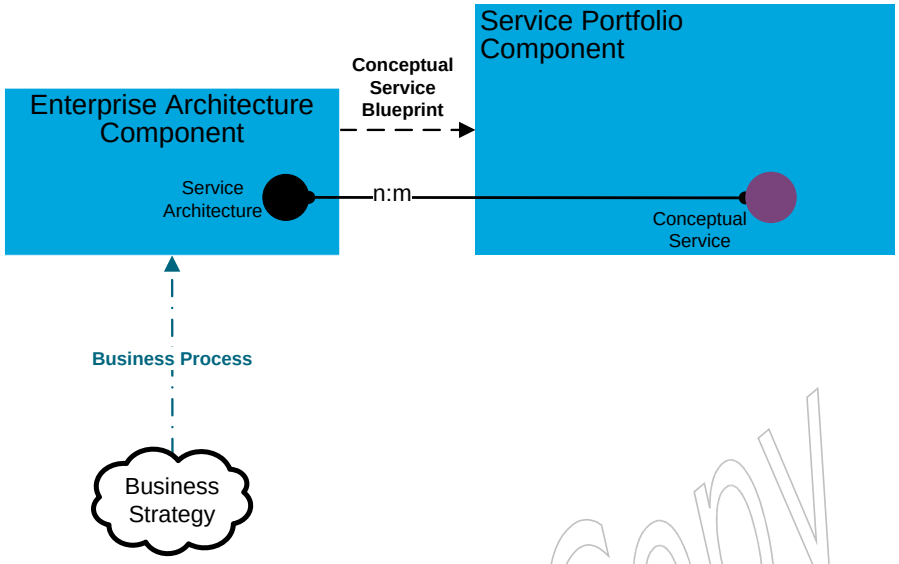


Figure 39: Enterprise Architecture Functional Component Level 2 Model

5.4.2 Policy Functional Component

Purpose

Manage creation, review, approval, and audit of all IT policies.

Key Data Objects

- **Policy** (data object): It is a central repository for storing and organizing all types of IT policies based on various templates and classification criteria.

Key Attributes

The Policy data object shall have the following key data attributes:

- **PolicyID**: Unique identifier; e.g., number of the Policy.
- **PolicyDescription**: Description/details of the Policy.
- **ApplicableGeography**: Which geographies does the Policy apply to; e.g., certain policies could be country-specific or may have country-specific customizations.

Key Data Object Relationships

The Policy data object shall maintain the following relationships:

- **Policy to Conceptual Service** (n:m): Multiple policies might be applicable for a single service or a single policy may be applicable for multiple services.

- **Policy to Requirement Functional Component** (n:m): Requirements may be sourced from policies or may reference policies in order to remain in compliance with previously agreed policies for an organization.

Main Functions

The Policy functional component:

- Shall align and map IT Policies to Service Architectures.
- Shall review and approve IT policies based on roles and responsibilities. It shall manage Policy distribution and acceptance based on predefined templates and schedules for designated IT stakeholders.
- Should maintain complete Policy revision history, and review period or obsolescence rules set for all Policies.
- May log and track IT Policy exceptions through an issue management mechanism. It may provide a consistent tracking feature for exception identification, evaluation, and status report leading to corrective action.
- Shall provide visibility into IT Policy attributes such as types, status, non-compliance, audit history, and issues.
- Shall manage overall IT governance Policies, and Policies applied to or associated with the particular services that may be managed downstream during service design.
- Shall manage IT security and regulatory Policies by incorporating external and internal security and regulatory compliances.
- Shall define pricing/costing Policies and capture information related to Service Contracts.

If a Service Portfolio functional component exists, the Policy functional component:

- Shall associate one or more policies to one or more Conceptual Services.

Model

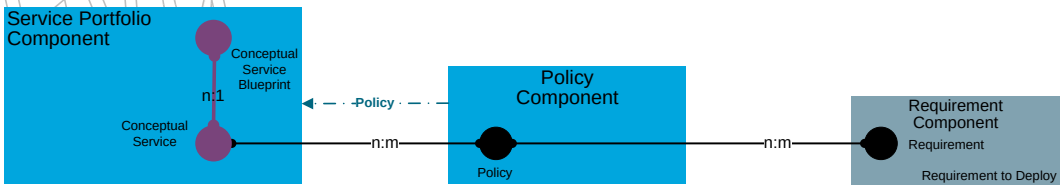


Figure 40: Policy Functional Component Level 2 Model

5.4.3 Proposal Functional Component

Purpose

Manage the portfolio of IT proposals that are proposed, approved, active, deferred, or rejected. This is the authoritative source for the list of IT proposals requested over a given time period that may result in the creation of Scope Agreements for projects.

Key Data Objects

- **Scope Agreement** (data object)**:** Proposals are created from rationalized Portfolio Backlog Items and, upon approval, Scope Agreements are created. Scope Agreements reflect budget, cost/benefit projections, scope, and other key attributes of proposed work. Views can be created for specific functions, such as line of business, or holistically, such as company-wide. Used for building the IT investment plan of record for the company or a specific line of business or function.

Key Attributes

The Scope Agreement data object shall have the following key data attributes:

- **ScopeAgreementID**: Unique identifier of the Scope Agreement.
- **ScopeAgreementDescription**: Details of the Scope Agreement.
- **BusinessEntity**: Business or geographic unit.

Key Data Object Relationships

The Scope Agreement data object shall maintain the following relationships:

- **Scope Agreement to Portfolio Backlog Item** (n:m): One Scope Agreement can be associated to one or more demand data objects.
- **Scope Agreement to IT Budget** (n:1): This relationship helps track budget allocated to which Scope Agreement.
- **Scope Agreement to IT Initiative** (1:n): This relationship helps track IT Initiative(s) to which Scope Agreement.

Main Functions

The Proposal functional component:

- Shall create a Scope Agreement from rationalized Portfolio Backlog Items in the data object repository. A Scope Agreement can follow an expedited analysis and approval for high priority urgent items or agile development proposals. A Scope Agreement can also follow a structured analysis and approval via IT annual planning activities.
- Activities for Scope Agreements requiring an expedited analysis and approval:

— Proposal created from a rationalized backlog item where the item requires high urgency due to business impact on an existing service.

— Quickly evaluate the proposal and decide on the approval. If rejected, then notify the Portfolio Demand functional component.

— Is there an existing IT Initiative that can be associated with the approved proposal? If yes, then create an updated Scope Agreement and update corresponding in-flight IT Initiative data object in the R2D Value Stream. In the context of the IT4IT Reference Architecture, an IT Initiative is any one of the class of temporary endeavors such as projects or programs with a defined beginning and end, undertaken to achieve an objective or outcome, at a specified cost.

— Create a new Scope Agreement. A new IT Initiative data object is created in the R2D Value Stream.

- Activities for proposals requiring structured analysis and approval:

— Proposals are created from rationalized Portfolio Backlog Items in the Portfolio Backlog Item data object repository. Rationalized items are grouped based on priority and themes for a proposal creation purpose. Not all rationalized items will be grouped within proposals as priority and cut-off must be decided.

— Proposals are created periodically throughout the year or once a year during annual planning activities.

— Create a high-level labor consumption model for a proposal (for example, one project manager, five developers, and two QAs for the proposal). Validate labor consumption model against available internal and external labor pools.

— Create a high-level asset (non-labor) consumption model for a proposal. Validate asset consumption model against available internal and external assets (for example, traditional/private cloud/managed cloud/public cloud).

— Model ongoing labor and non-labor budget for annual and future operations.

— Define tangible and intangible benefits for each proposal. Tangible benefit may be cost savings or revenue growth, whereas intangible benefit may be strategic initiative support, competitive advantage, or compliance achieved. Work with a finance organization to validate tangible benefits. This can involve utilizing industry-specific methods of measuring the value of business processes and estimating the impact of the proposal on performance metrics.

— Ensure proposal meets the technology policies.

— Rank proposals based on benefits and risks, labor and non-labor consumption models, and ROI or other defined evaluation criteria.

— Build proposal portfolio scenarios using proposals, conduct “what if” analysis on the proposal scenarios, and approve the optimal proposal scenario and its proposals.

— Create resulting Scope Agreement(s). Retain Scope Agreement information to compare approved baseline and actual resulting benefits derived from completing the IT Initiatives.

The Proposal functional component:

- Shall review the Scope Agreement change request from the R2D Value Stream. The IT Initiative team working to deliver on the approved Scope Agreement may ask for change requests related to budget, resource, or timeline. Evaluate the change request and take action to update the existing Scope Agreement.
- The R2D Value Stream project portfolio is the authoritative source for the list of IT deliverables or services that will be rendered during a project lifecycle. The project portfolio views can be created for specific organizations like line of business portfolio or functions like financial views. The project portfolio is used for rationalizing and tracking resources across projects to best deliver on all projects. The project portfolio entries are actuated through a Project Management system. The project portfolio reports back to the investment portfolio in order to accurately track progress and outcomes for a given Scope Agreement.
- Shall identify security controls necessary for protecting the various classifications of data.

Model

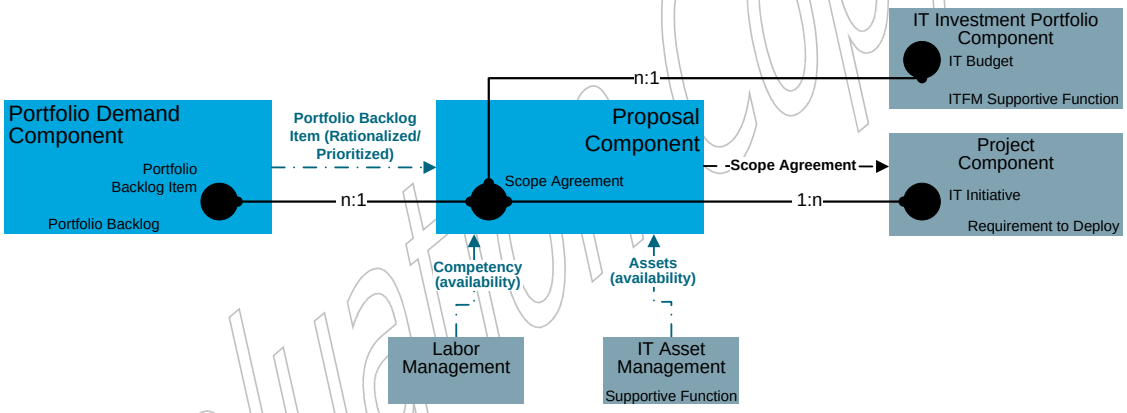


Figure 41: Proposal Functional Component Level 2 Model

5.4.4 Portfolio Demand Functional Component

Purpose

Log, maintain, and evaluate all demands (new service, enhancements, defects) coming into IT through a single funnel. Correlate incoming demand to similar existing demand or create new demand. The "single funnel" may be a virtual concept encompassing project ideation, service request management, Incident Management, continuous improvement, and other well-known demand channels.

Key Data Objects

- **Portfolio Backlog Item** (data object): Portfolio Backlog Items represent the repository of all incoming demands including but not limited to new requests, enhancement requests, and defect fix requests.

Key Attributes

The Portfolio Backlog Item data object shall have the following key data attributes:

- **DemandID**: Unique identifier of the Portfolio Backlog Item (demand request).
- **DemandDescription**: Description/details of the Portfolio Backlog Item (demand request).
- **Source**: Where did the demand originate from (e.g., person/department)?
- **ITServiceID**: Is demand related to a live service (e.g., enhancement requests)?
- **DemandFulfillmentStatus**: Status information.
- **DecisionMaker**: Information on who took the decisions related to the demand.

Key Data Object Relationships

The Portfolio Backlog Item data object shall maintain the following relationships:

- **Portfolio Backlog Item to Conceptual Service** (n:1): One Conceptual Service may be related to one or more Portfolio Backlog Items.
- **Portfolio Backlog Item to Requirement** (1:n): A Portfolio Backlog Item is mapped to one or more Requirements which will need to be delivered to successfully fulfill the demand.
- **Portfolio Backlog Item to Scope Agreement** (n:1): One or more Portfolio Backlog Items may be included in a Scope Agreement.

Main Functions

The Portfolio Demand functional component:

- Shall capture Portfolio Backlog Items from business.
- Shall capture Portfolio Backlog Items from Problem Management activities.
- Shall capture Portfolio Backlog Items from the Service Portfolio functional component activities.

If a Proposal functional component exists, then the Portfolio Demand functional component:

- Shall categorize and group the demands and push demands to the Proposal functional component.

If a Requirement functional component exists, the Portfolio Demand functional component:

- Shall associate one or more Requirements (user stories, use-cases, business rules, etc.) to a Portfolio Backlog Item.

The Portfolio Demand functional component may support backlog item data object backlog ranking, trending, and analysis based on requested services, timeline, business unit origination, etc.

Model

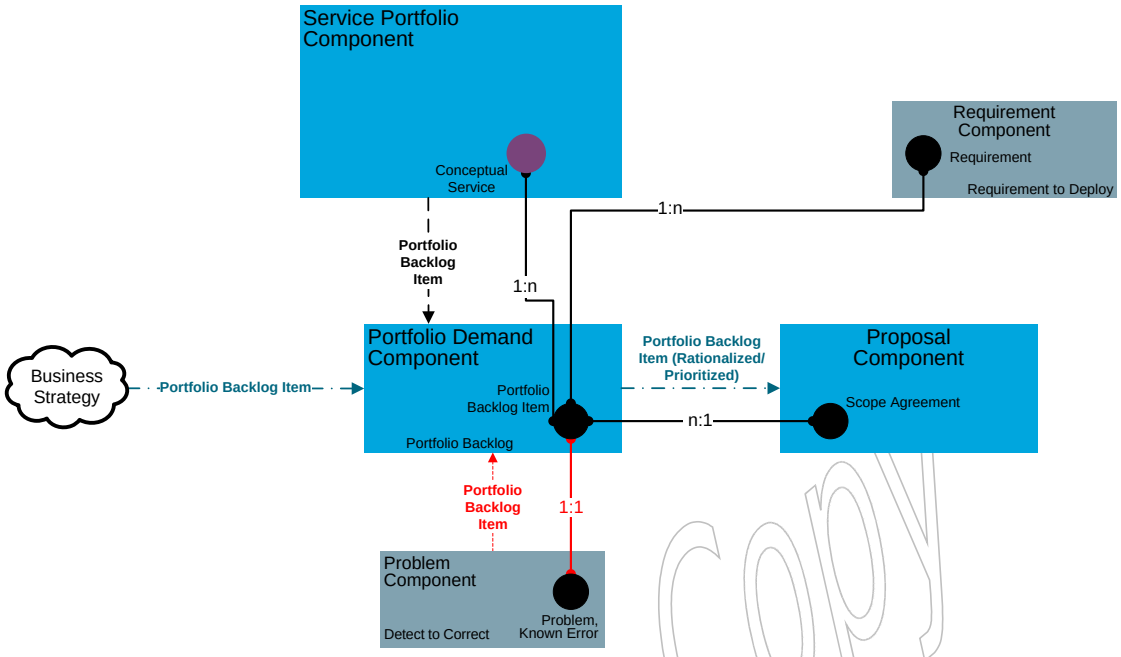


Figure 42: Portfolio Demand Functional Component Level 2 Model

5.4.5 Service Portfolio Functional Component

Purpose

Manage the portfolio of services in plan, transition, production, and retirement. Authoritative source for the list of services that IT delivers, has delivered in the past, or brokers to itself and business. Any IT service within the Service Portfolio functional component often corresponds to one or more entries in the Offer Catalog.

Key Data Objects

- **Conceptual Service** (data object): The Service Model is an authoritative source for the list of services that the enterprise consumes. It represents services planned, in transition, in production, or retired.
- **Conceptual Service Blueprint** (data object): The Conceptual Service Blueprint contains the list of all service blueprints associated with a given Conceptual Service. (Each Conceptual Service Blueprint has a comprehensive view of the service that depicts endpoints and interfaces that can be understood by Architects and BRMs).

Key Attributes

The Conceptual Service data object shall have the following key data attributes:

- **ServiceID**: Unique identifier of the Conceptual Service.

- **ConceptualServiceDetails**: Details of the Conceptual Service.
- **ServiceOwner**: Owner of the Conceptual Service.
- **ServiceStatus**: Status; e.g., planned, retired, etc.

The Conceptual Service Blueprint data object shall have the following key data attributes:

- **ConceptualServiceBlueprintID**: Unique identifier of the Conceptual Service Blueprint.
- **ConceptualServiceBlueprint**: Details of the Conceptual Service Blueprint.
- **ServiceID**: Unique identifier of the Conceptual Service to which Blueprint is associated.

Key Data Object Relationships

The Conceptual Service data object shall maintain the following relationships:

- **Service Architecture to Conceptual Service** (n:m): Traceability is maintained between one or more Conceptual Services and the Enterprise Architecture drawings, diagrams, and other documents that describe those services.
- **Conceptual Service to Portfolio Backlog Item** (1:n): One Conceptual Service may be related to one or more Portfolio Backlog Items.
- **Conceptual Service to IT Budget** (n:m): Budget for one Conceptual Service may be spread across multiple budget items and one budget item could hold budget for multiple Conceptual Services.
- **Conceptual Service to Policy** (n:m): Multiple Policies might be applicable for a single service or a single Policy may be applicable for multiple services.

The Conceptual Service Blueprint data object shall maintain the following relationships:

- **Conceptual Service to Conceptual Service Blueprint** (1:n): One Conceptual Service may have multiple Conceptual Service Blueprints.
- **IT Cost Model to Conceptual Service Blueprint** (1:n): One IT cost model (rule engine) can be applicable for multiple Conceptual Service Blueprints.
- **Conceptual Service Blueprint to Logical Service Blueprint** (1:n): One Conceptual Service Blueprint could have one or more Logical Service Blueprints.

Main Functions

The Service Portfolio functional component:

- Shall assess the effectiveness and efficiency of current services delivered to business.
- Shall manage all inventory information about services or applications; including business benefits, risk, quality, fit to purpose, etc.
- Shall compare similar services or applications to identify rationalization opportunities.

- Shall evaluate the portfolio with regard to value/cost performance and risk/criticality. These methods are used to maximize portfolio value, align and prioritize resource allocations, and balance supply and demand.
- Shall review proposed portfolio changes; decide whether to keep, retire, or modernize services or applications.
- Shall create, review, and update service roadmaps.
- Shall determine and track service budgets/actuals information.
- Shall create and maintain service blueprints and endpoints. A service blueprint is a set of service endpoints that support business processes. A service blueprint provides service process and delivery visualization from the customer's point of view. A service blueprint also maintains traceability of Logical and Physical (realized) Service Models.

If a Portfolio Backlog Item exists, the Service Portfolio functional component:

- Shall associate a Conceptual Service to one or more Portfolio Backlog Items.

If a Policy exists, the Service Portfolio functional component:

- Should comply with one or more applicable Policies.

Model

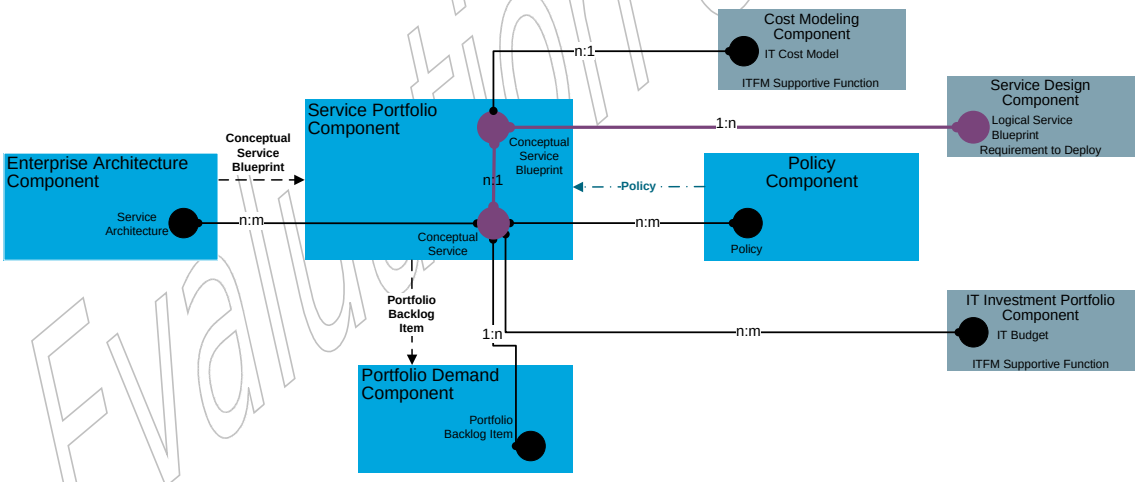


Figure 43: Service Portfolio Functional Component Level 2 Model

5.4.6 IT Investment Portfolio Auxiliary Functional Component

Purpose

This functional component is auxiliary to the S2P Value Stream and is primary in the IT Financial Management guidance document.

Its main purpose includes:

- Manage the authoritative list of all IT investments.
- Facilitate all activities required to effectively carry out forecasting and budgeting.
- Facilitate proposal managers with guidelines and information (e.g., unit costs) to be followed while estimating costs of proposed IT Initiatives.
- Establish a common governance and control mechanism for approval/rejections of any proposed IT Initiatives.

Key Data Objects

- **IT Budget** (data object):IT Budget is an authoritative list of approved IT investment pertaining to a proposed scope of work. This set of records will help to identify approved budget over different time periods; e.g., by financial year.

Key Attributes

The IT Budget data object shall have the following key data attributes:

- **FinancialPeriod** : Period expressed in MM-YYYY.
- **ServiceID**: Unique identifier of IT service.
- **ScopeAgreementID**: Unique identifier of the Scope Agreement.
- **BudgetType** : CapEx or OpEx.
- **RequestedBudget**: Budget that was originally requested.
- **ApprovedBudget**: Budget that was approved.
- **Spend**: Actual spend that was accounted.

Key Data Object Relationships

The IT Budget data object shall maintain the following relationships:

- **IT Budget to Conceptual Service** (n:m): This relationship helps track how much IT Budget is allocated to which IT service(s).
- **IT Budget to Scope Agreement** (1:n): This relationship helps track how much IT Budget is allocated to which Scope Agreement(s).

Main Functions

The IT Investment Portfolio functional component:

- Shall be the system of records for all IT investments.
- Shall manage the entire IT investment lifecycle.

- Shall provide labor & non-labor cost estimates and other guidelines to the Proposal functional component. This should be used by proposal managers to estimate the cost of proposed IT Initiatives.

  Various proposal managers working on their respective proposals should send the information on proposed IT investments to the IT Investment Portfolio functional component.

- Shall accept inputs from the Service Portfolio functional component to include OpEx budget requests for keeping live services operational.

- Shall take these proposals/budget requests for necessary approvals with the respective governing committee.

- Shall communicate the status of the final investment decisions back to the respective stakeholders.

Model

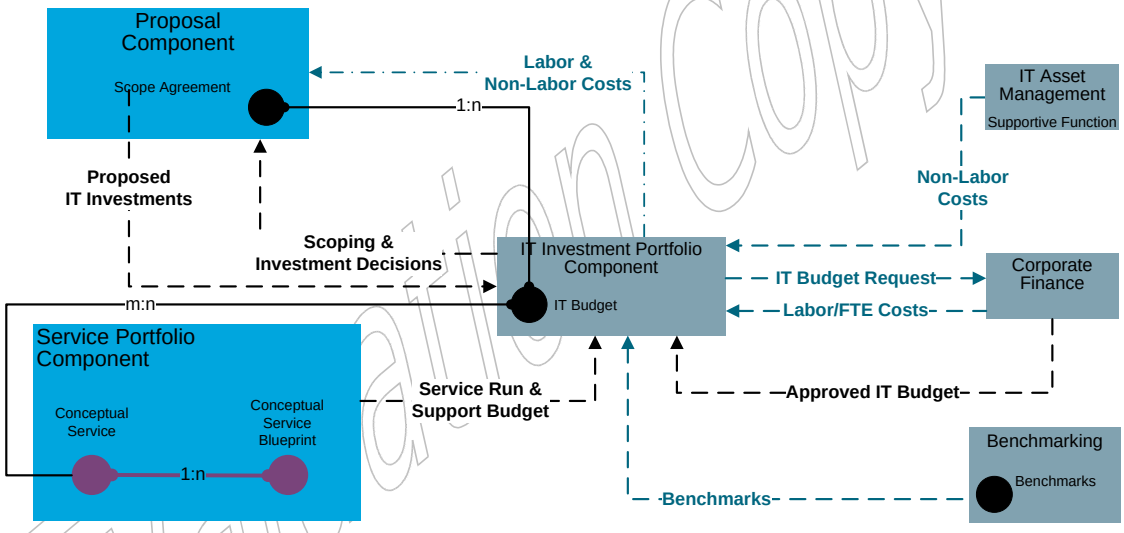


Figure 44: IT Investment Portfolio Auxiliary Functional Component Level 2 Model

6 Requirement to Deploy (R2D) Value Stream

This chapter provides an overview of the Requirement to Deploy (R2D) Value Stream – one of four IT Value Streams that comprise the IT Value Chain. It describes the business context, objectives, and details behind the R2D Value Stream.

6.1 Objectives

IT organizations today experience challenges in planning, sourcing, developing, and delivering applications and services that generate desired business outcomes. The R2D Value Stream is designed to ensure predictable, cost-effective, high quality results to the business while also promoting high levels of re-use, flexibility, speed, and collaboration across IT to support traditional and new methods for service creation and sourcing.

Key objectives to meet in support of giving the business what it needs are to:

- **Make service delivery predictable, even across geographically dispersed teams, multiple suppliers, and multiple development methodologies.**

  Applications and services today are sourced or developed in cooperation with many different parties. All parties are working with their own processes and tooling. IT must be able to provide a good overview of the planned activities, should speak a common language with all parties involved, and should provide a methodology for how to achieve the highest quality results.

  Cloud sourcing, agile development, and other innovations have created the need for IT to be able to manage development and delivery of services in a hybrid or multi-sourced environment. The R2D Value Stream provides a framework that can accommodate everything including:

  — Contracts for Software as a Service (SaaS)/Subscription services

  — Fast, lightweight application releases such as mobile applications or those built using agile methodologies (for example, SCRUM, extreme programming)

  — Traditional applications that utilize more structured project-driven release cycles

  — Everything in between

- **Ensure that each Service Release is high quality, fit-for-purpose, and meets customer expectations.**

  IT still experiences too many incidents immediately after release of an application or service into production. IT must establish control over the quality of a service regardless of the number of vendors involved in development and/or delivery.

- **Understand the evolving relationship between planning and building.**

  Historically, estimation has been framed as a trade-off, balancing a prior "planning" phase *versus* a later "building" phase. However, it is increasingly understood that planning in complex domains requires information only available through iterative attempts to fulfill requirements. In a sense, both planning and building must take place simultaneously in many cases.

- **Standardize service development and delivery to the point where re-use of service components is the norm.**

  IT operations and development must ensure increased quality and speed of service delivery while also lowering costs. In support of these efficiency and quality goals, IT must have a framework in which to drive the re-use of existing service components at multiple stages of the development lifecycle across multiple applications and services. IT must work successfully with multiple internal and external contributors and be able to integrate the data, process, and tools required to work with geographically dispersed teams, outsourcers, and traditional and cloud-based suppliers. Furthermore, IT must maintain control of the governance of the R2D Value Stream and be able to track and measure internal and vendor performance, costs, quality, and on-time delivery. The ability to re-use requirements, source code, documentation, test scripts, service monitors, and other data objects of the service development lifecycle is a key contributor to managing cost, increasing quality and predictability, and accelerating release cycles.

- **Build a culture of collaboration between IT operations and IT development to support Service Release success.**

  IT operations and development must improve collaboration between departments. Development organizations build and test services in a silo and "surprise" IT operations by "throwing release packages over the fence" for immediate delivery. IT operations may not be able to accommodate new technologies and environments fast enough to meet the requirements of developers. Inefficient manual processes are typical, and in high maturity shops are increasingly replaced by fully automated continuous delivery pipelines.

- **Put rigorous information management controls in place to lessen the impact of the IT reality – high staff turnover.**

  High turnover in IT means knowledge is lost and schedules are impacted. Particularly in low-cost labor markets where employers are suffering high employee turnover rates. The R2D Value Stream helps capture the knowledge that would otherwise be lost and cause schedules to be delayed.

- **Drive predictable outcomes without driving out innovation.**

  Innovation and process efficiency are two pillars of competitive advantage that IT departments bring to the business, yet these two pillars often have trouble co-existing. The emphasis on on-time project delivery tends to stifle innovation, creating a conflict between these two priorities. IT must continuously improve its ability to execute in such a way that on-time innovation is the norm. The R2D Value Stream identifies the core automation enablers and the key data exchanges required to accomplish this goal. For

example, focusing efforts on automation of test, release, and deployment provides more time and resource for innovation in service design and development.

6.2 Business Value Proposition

The R2D Value Stream describes a prescriptive framework of required functional components and data objects so IT organizations can better control the quality, utility, schedule, and cost of services regardless of the delivery model.

The key value propositions for adopting the R2D Value Stream are:

- Maximize the pipeline of projects and smaller grained demand requests for faster time-to-market in service realization.
- Predictable outcomes that ensure that the application or service delivered actually performs as requested, leading to higher rates of user acceptance and better business alignment.
- Establish control points to manage the quality, utility, security, and cost of services, independent of development method or delivery source.
- Increased management information for traceability and benchmarking of internal and external service developers and suppliers.
- Ensure that all services are designed in accordance with standards and policies (from sources including Corporate Compliance, Enterprise Architecture, Risk Management, IT Financial Management, and so on).
- Improved inputs to IT Financial Management on service cost.
- Relate applications and services with business value by creating and maintaining the service blueprint.
- Accelerate the sourcing and delivery of applications and services through best practices such as:
  - Re-use – manage, maintain, and leverage re-usable IT components and services.
  - Automation – identify the core functional components and data required to streamline the R2D Value Stream.
  - Collaboration – use data to institutionalize collaboration of teams involved in the development lifecycle to expedite releases, and reduce incidents and rework that might otherwise result. This lays the foundation for new paradigms such as DevOps.

6.3 Key Performance Indicators

The R2D Value Stream critical success factors and Key Performance Indicators (KPIs) are as follows:

Critical Success Factors	Key Performance Indicators (KPIs)
Improve Quality	Number of escaped defects % of actual *versus* planned executed tests % of critical defects found early in unit testing *versus* UAT
Improve Project and Feature Execution	% of projects (project tasks, stories, other demand requests) on time % of healthy projects (projects without unresolved urgent issues) Deviation of planned to actual work hours Number of identified issues Number of opened risks Amount of backlog/work-in-process Arrival and departure rate for work
Improve Stewardship of IT Investment	% of actual *versus* planned project cost % of change in project cost % of budget at risk
Increase Automation Adoption	% of automated tests
Achieve Development Process Excellence	% of requirements tested, authorized, completed % of requirements traced to tests % of reviewed requirements % of successful builds % of changes resulting in Incidents Ratio of detected to closed defects at release
Improve Early Life Success of Releases	% of Incidents during warranty period % of successful/unsuccessful deployments for the project % of emergency changes Pass rates on UAT/validated requirements
Operations and Development Collaboration	Trend on early life support/UAT success metrics % rework
Improve Financial Visibility	Planned cost *versus* actual cost

Critical Success Factors	Key Performance Indicators (KPIs)
Maintain a Linkage between Business Services and IT Initiatives	Aggregate (roll up) service development costs by business service
High Quality Service Design Specifications at the Outset	% reduction in the rework required for new or changed service solutions in subsequent lifecycle stages
Integration Test Success	Trend on the number of installation errors in all the packages in the integration environment Number of applications or services that require exceptions outside of the existing infrastructure portfolio
Design-Review to Ensure Application Design Complies with all Policies, including Security	Number of application designs that pass a security policy review
Early Testing of Applications for Security Vulnerabilities	% of severity 1 security defects fixed before application is released

6.4 Value Stream Definition

The Requirement to Deploy (R2D) Value Stream provides the framework for creating and sourcing a new or modifying an existing application or service. The R2D Value Stream is triggered if it receives a demand signal from a consumer or from other components such as Problem Management. This may take the form of an approved Scope Agreement and Conceptual Service Blueprint from the S2P Value Stream, or may be a smaller grained signal such as an individual development story, user story, defect, problem, usage data, or scenario for a specific application or service. The R2D Value Stream ends when the requested service or modification is packaged for immediate or future deployment through an R2F Value Stream Fulfillment Execution functional component.

Successful execution of the R2D Value Stream is dependent on the following key data objects:

- **Scope Agreements**, which originate from the S2P Value Stream and provide budget, cost/benefit projects, sizing/scoping of efforts, and other key attributes for proposed work.
- **Conceptual Service Blueprints**, which are generated and managed in the S2P Value Stream and provide the business view of the desired service or application including processes, architectural data objects, and other information.
- **Portfolio Backlog Items**, which are stored and managed in the S2P Value Stream and provide details on development needs that have not yet been met.
- **Standards & Policies**, which originate from the S2P Value Stream where the growing body of guidelines and best practice requirements are captured from IT and the business.
- **Problems**, which are generated from the D2C Value Stream and create one or many Defects.

The inputs above are assessed, rationalized, and elaborated with more detail and ultimately become content that is consumed and produced in the R2D Value Stream. The following are the key data objects and Service Model entities utilized in the R2D Value Stream:

Build	A Build is the assembly of Source. The term is also used to refer to a Build that is authorized for distribution; for example, software build, server build, or laptop build.
Build Package	A collection of one or more Builds together which will be included as part of a Release Package.
Defect	A flaw in a component or system that can cause the component or system to fail to perform its required function, such as an incorrect statement or data definition. A Defect, if encountered during execution, may cause a failure of the component or system.
IT Initiative	A temporary endeavor with a defined beginning and end undertaken to achieve an objective or outcome at a certain cost. In the context of the IT4IT Reference Architecture, the typical initiative outcome is the delivery of a new service or modification of an existing service or application.
Logical Service Blueprint	Provides the structure and behavior and design for the components that make up a new or changed service and describes how those components relate to one another. The Logical Service Blueprint can be thought of as what is traditionally expressed in IT terms as the design. The Logical Service Blueprint together with the Service Release and Service Release Blueprint make up the Logical Service Model. Along with the service design data objects, the Logical Service Blueprint introduces the "how" details; i.e., technology delivery methods (on-premise *versus* SaaS), and the technical interfaces necessary to access the service functionality.
Release Package	A collection of one or more Build Packages together with a Service Release Blueprint which describes how the Build Packages can be deployed (as one Build Package may exist for the app server, for the database, and for the front end). A Release Package might also contain other release objects that are not the result of a build process such as training material, Known Errors, or Run Books.
Requirement	A formal statement of what is needed. Such a statement identifies a necessary attribute, capability, characteristic, or quality of a system for it to have value and utility to a user. A product backlog or program backlog can be derived from requirements.
Service Design Package	Provides the necessary details that describe the design of the IT service, at least sufficient to a given stage in its development. It may be iteratively refined.

Service Release	Describes a specific release of an IT service which could include changes to the service systems and/or service offers for that IT service. One Service Release is defined by complying with a single Logical Service Blueprint. A new Service Release must be established whenever the Logical Service Blueprint for a service changes. A service is released according to a release schedule and cadence (e.g., annual, quarterly, daily) and communicates to external stakeholders of new functionality, improvements, and defect fixes as well as existing problems. A Service Release includes attributes such as a release manifest (that includes underlying technology details), description, dependencies, and one or more Service Release Blueprints (see below).
Service Release Blueprint	Provides the design for the components of an IT service. It contains the description and procedures in order to activate, deploy, and operate a service and its underlying components including applications and technology. Typically, there will be multiple blueprints for each Service Release deployed onto different environments such as development, QA, and production. This distinction is also useful to distinguish between delivery mechanisms, such as on-premise *versus* cloud.
Source	Represents all kind of sources used within the build process or packaged into a Build to make up an application or service. This may include but is not limited to: build scripts, configuration files, localization files, licenses and entitlements to external services, code in programming languages, code of monitors, HTML, CSS, and JS.
Test Case	One or a set of Test Cases (manual or automated) which contain a set of input values, execution preconditions, expected results, and execution post-conditions, developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement. A Test Case is developed based on a Requirement or a Defect and the same Test Case can be applied across many releases.

The following functional components in the R2D Value Stream support the definition, development, and governance of the data objects and Service Model entities above:

- **Build**: Create, manage, secure, and track Builds. Implement Build automation. Manage the delivery of Builds to the Build Package component.
- **Build Package**: Manage and store one or more Builds in a deployable package.
- **Defect**: Keep track of all Defects, including their origin, status, importance, and relation to Requirements and Known Errors.
- **Project:** Receive Scope Agreements and coordinate the creation and provide ongoing execution oversight of IT Initiatives aimed at the creation of new or enhancements to existing IT services.
- **Release Composition:** Keep track of the different Service Releases, define the structure and content of the Service Release and its underlying components, including the instructions of how each component can be deployed.
- **Requirement:** Manage Requirements through the lifecycle of the service. Maintain traceability of each Requirement to the original request that generated the Portfolio Backlog Item throughout the service lifecycle. Collect, refine, scope, and track progress of

Requirements. Provide views into product or program backlogs and team backlogs as a subset of the Requirements.

- **Service Design:** Create the Logical Service Blueprint for the service(s). Ensure these meet the requirements from the Scope Agreement, IT Initiative, and/or Portfolio Backlog Item and make it perform against the Key Performance Indicators (KPIs), Key Risk Indicators (KRIs), and Service-Level Agreements (SLAs). The output of the Service Design functional component is used by the Source data object to guide source, create, and secure the service.
- **Source Control:** Ensure that the service is developed in accordance with design specifications, organizational standards and policies, and both functional and non-functional requirements so that the service can be operated successfully and in line with customer expectations and requirements. Produce and manage source and documentation which is stored.
- **Test:** Plan, store, and execute tests which ensure that the service will support the customer's requirements at the agreed service levels, including system/integration testing, user acceptance testing, performance testing, and load testing.

The R2D Value Stream is process-agnostic in that while methods and processes may change (i.e., ITIL, COBIT, agile, waterfall, etc.) the functional components and data objects that comprise the value stream remain constant.

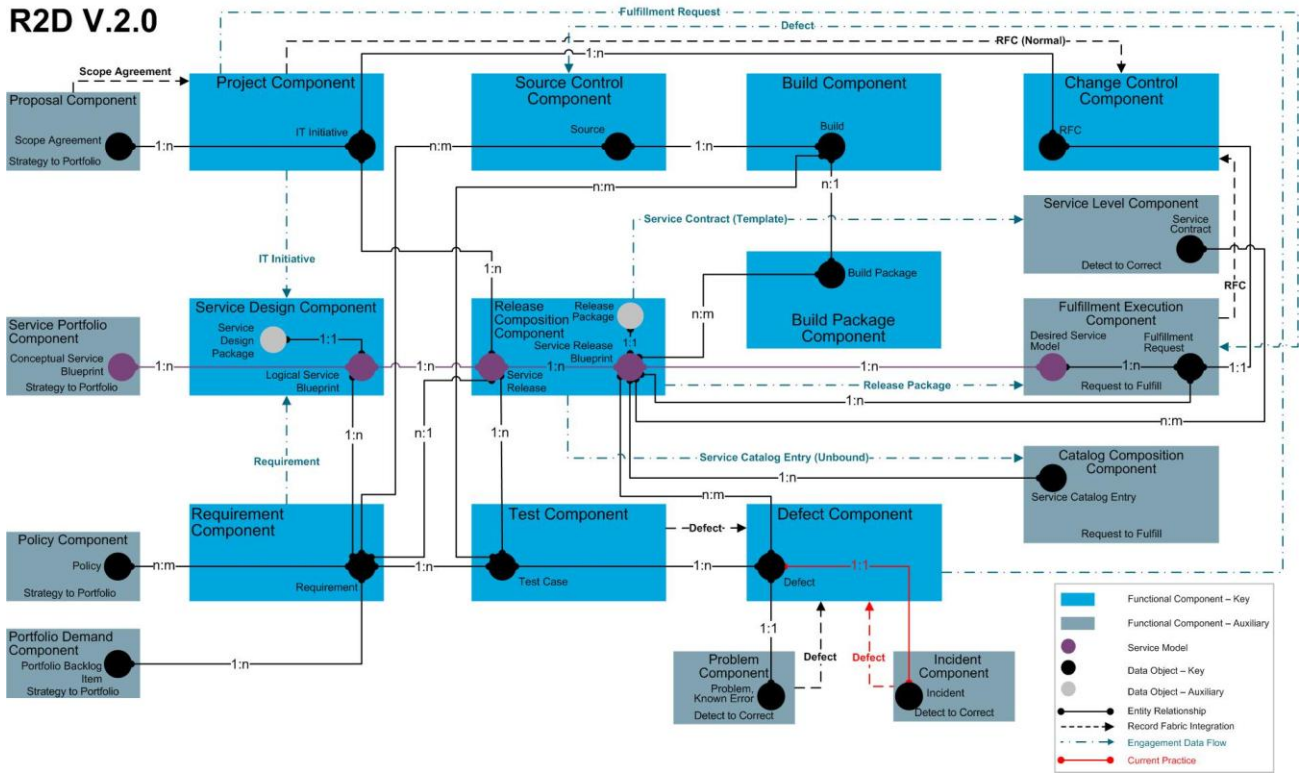


Figure 45: Requirement to Deploy Level 2 Value Stream Diagram

6.4.1 Project Functional Component

Purpose

- Coordinate the creation and provide ongoing execution oversight of IT Initiatives aimed at the creation of new or enhancements to existing services.
- Create IT Initiatives based on the specifications outlined in the Scope Agreement, including cost, time, scope, and quality.
- Aggregate, track, and report status, resources consumed against project plan, or project burn down, and communicate these to stakeholders via auxiliary functional components such as Resource Management, Supplier Management, and IT Financial Management.
- Govern, coordinate, influence, and direct initiative execution.
- Ensure financial goals and boundary conditions are adhered to.
- Maintain the connection between initiatives and associated applications and service(s) being developed.
- Maintain the linkage/traceability between Scope Agreements, IT Initiatives, and Service Releases.
- Produce various request artifacts associated with financial, human, and technology resources (that is, feedback to the S2P Value Stream when variances cross thresholds).
- Coordinate the acquisition of resources (hardware, software, and people) required to source/create a service in a particular project.

Key Data Objects

- **IT Initiative** (data object): Details the scope of the work to be performed and created from and associated with the Scope Agreement.

Key Attributes

The IT Initiative data object shall have the following key data attributes:

- **ITInitiativeID**: Unique identifier of the effort/initiative.
- **ITInitiativeName**: Full name of the effort/initiative.
- **ITInitiativeStatus**: Status of the effort/initiative.
- **ServiceReleaseID**: Identifier of one or more Service Releases.
- **RFCID**: Identifier of one or more RFCs.
- **ScopeAgreementID**: Unique identifier of the related Scope Agreement.

Key Data Object Relationships

The IT Initiative data object shall maintain the following relationships:

- **Scope Agreement to IT Initiative** (1:n): Maintain a linkage between the proposal which authorized one or more IT Initiatives.
- **IT Initiative to Service Release** (1:n): An IT Initiative will manage the creation of one or more Service Releases required to deliver the IT Initiative.
- **IT Initiative to Request for Change (RFC)** (1:n): An Initiative will be related to one or many RFC records in order to manage all changes resulting from a single work effort (initiative).

Main Functions

The Project functional component:

- Shall be the system of record (authoritative source) for all IT Initiatives.
- Shall manage the lifecycle of the IT Initiative.
- Shall manage the status of the IT Initiative.
- Shall allow recursive relationships between IT Initiatives.
- Shall associate an IT Initiative to a service.
- May associate an IT Initiative with IT budget in the IT Financial Management supporting function.

If a Change Control functional component exists, the Project functional component:

- Shall associate an IT Initiative to one or more RFCs.
- Can submit one or more RFCs required for the IT Initiative.

If a Fulfillment Execution functional component exists, the Project functional component:

- Can manage the Fulfillment Request data flow to the Fulfillment Execution functional component.
- Can send a request to the Fulfillment Execution functional component when resources are required for the IT Initiative.

If a Proposal functional component exists, the Project functional component:

- Shall associate a Scope Agreement to one or more IT Initiatives.
- Shall be able to receive the Scope Agreement from the Proposal functional component.

If a Service Design functional component exists, the Project functional component:

- Can provide IT Initiative information required for Service Design to the Service Design functional component.

Model

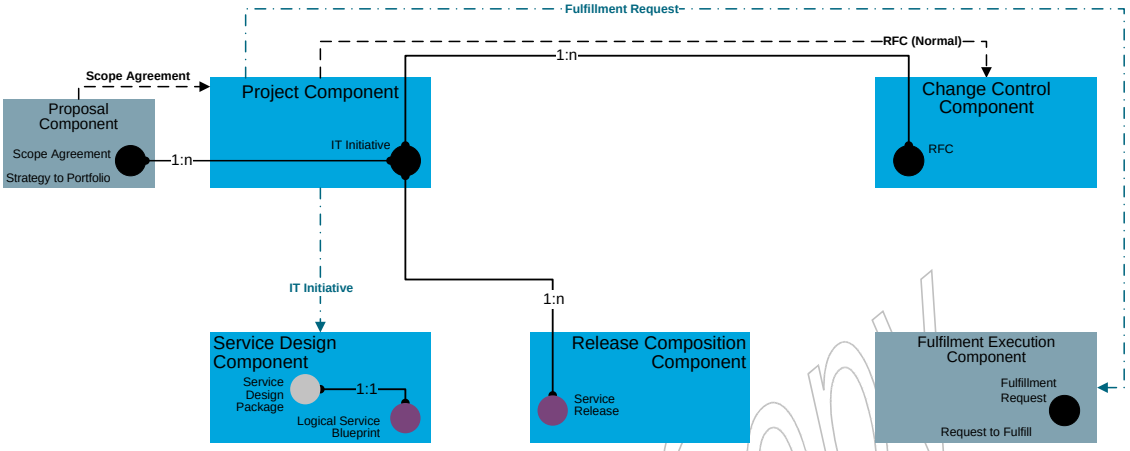


Figure 46: Project Functional Component Level 2 Model

6.4.2 Requirement Functional Component

Purpose

- Manage Requirements through the lifecycle of a service.
- Service-level requirements are captured as Requirements.
- Collect, refine, scope, and track progress of Requirements even before and after an IT Initiative has concluded.
- Maintain traceability of each Requirement to the original source (demand, IT or business standard or policy, and/or requestor) and to appropriate source and/or test cases throughout the service lifecycle.
- Derive product or program backlogs which will ultimately serve as queues for enhancing IT services.

Key Data Objects

- **Requirement** (data object): Record which details the needs or conditions to meet for a new or altered service.

Key Attributes

The Requirement data object shall have the following key data attributes:

- **RequirementID**: Unique identifier of a given Requirement.

- **RequirementType**: Identifies the category of a Requirement (i.e., epic, theme, feature, user story, non-functional, functional, etc.).
- **RequirementSummary**: Represents the short description/title/summary of a given Requirement.
- **LogicalServiceBlueprintID**: Unique identifier of the related Logical Service Blueprint.
- **PolicyID**: Unique identifier of the related Policy.
- **PortfolioBacklogID**: Unique identifier of the related Portfolio Backlog Item.
- **ServiceReleaseID**: Unique identifier of the related Service Release.
- **SourceID**: Unique identifier of the related Source.
- **TestCaseID**: Identifier of the related Test Cases.

Key Data Object Relationships

The Requirement data object shall maintain the following relationships:

- **Logical Service Blueprint to Requirement** (1:n): The Logical Service Blueprint is the Service Design which fulfills one or more Requirements.
- **Service Release to Requirement** (1:n): The Service Release delivers a service which fulfills one or more Requirements.
- **Requirement to Test Case** (1:n): A Requirement is traced to one or more Test Cases to ensure stated needs or conditions have been successfully delivered or met.
- **Portfolio Backlog Item to Requirement** (1:n): A Portfolio Backlog Item is mapped to one or more Requirements which will need to be delivered to successfully fulfill the demand.
- **Policy to Requirement** (n:m): Requirements may be sourced from policies or may reference policies in order to remain in compliance to previously agreed policies for an organization.

Main Functions

The Requirement functional component:

- Shall be the system of record (authoritative source) for all Requirements.
- Shall manage the lifecycle of the Requirement.
- Shall manage the state of a Requirement.
- Shall allow recursive relationships between Requirements.
- Shall allow hierarchical relationships between Requirements.
- Shall associate a requirement to a service.

If a Portfolio Demand functional component exists, the Requirement functional component:

- Shall associate one or more Requirements to a Portfolio Backlog Item that these Requirements originate from.

If a Service Design functional component exists, the Requirement functional component:

- Can manage the data flow to provide Requirement information to the Service Design functional component.
- Shall associate one or more Requirements to a single Logical Service Model.

If a Release Composition functional component exists, the Requirement functional component:

- Shall associate one or more Requirements to a Service Release that will fulfill these Requirements.

If a Test functional component exists, the Requirement functional component:

- Shall allow a Requirement to be traced to one or more Test Cases designed to test this Requirement.

If a Policy functional component exists, the Requirement functional component:

- Shall allow one or more Requirements to be associated to one or more policies that these Requirements originate from.

If a Source data object exists, the Requirement functional component:

- Shall allow one or more Requirements to be traced to one or more Sources.

Model

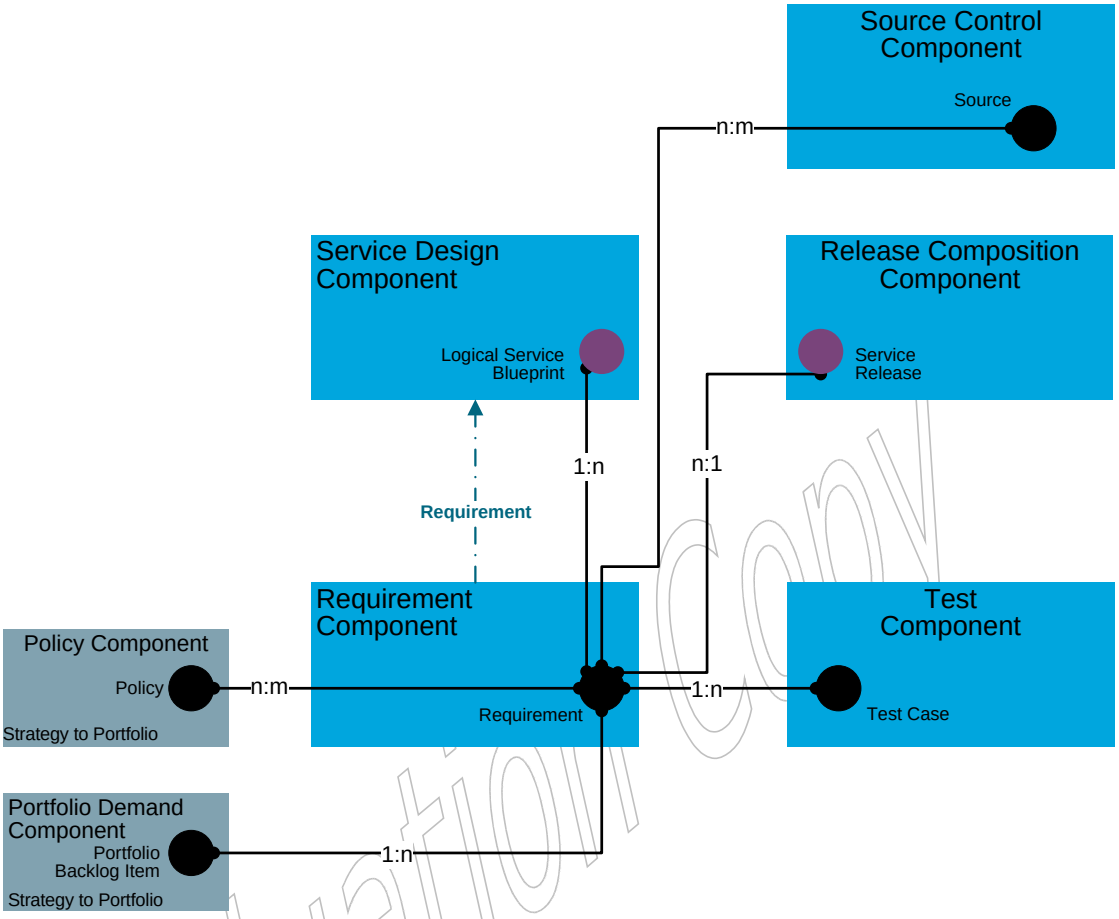


Figure 47: Requirement Functional Component Level 2 Model

6.4.3 Service Design Functional Component

Purpose

- Identify the new or existing services required to meet the needs of the Scope Agreement and IT Initiative, including both service systems and service offers.
- Leverage the Conceptual Service Blueprint and Portfolio Backlog Items from the S2P Value Stream along with requirements to produce a Logical Service Blueprint that describes the service structure and behavior considering both the service system and the service offer.
- Creation of various architectural artifacts (data flow diagrams, technical schematics, etc.) that comply with the IT Initiative specifications and boundaries.
- Creation of the service design specification document (Service Design Package).
- Identification of the service delivery model (in-source, outsource, etc.).

- Identification of service suppliers to meet the requirements within the chosen delivery model.
- Enable interaction with IT operations to develop support plan/requirements for an IT service.
- Ensure that the architecture and Service Design Package are compliant with all standards and policies, including security standards and policies.
- Ensure that the architecture and Service Design Package meets all requirements including security requirements to ensure the confidentiality, integrity, and availability of the service.
- Put instrumentation in place so that IT can capture empirical data about how IT services are performing rather than relying only on anecdotal input from the user community.
- Ensure that the service is architected to meet the KPIs and SLAs.
- The output of the Service Design functional component is used by the Source data object to source, create, and secure the service.

Key Data Objects

- **Logical Service Blueprint** (data object): Represents the logical design of the service based on the Requirements and Conceptual Service Blueprint.

Key Attributes

The Logical Service Blueprint data object shall have the following key data attributes:

- **LogicalServiceBlueprintID**: Unique identifier of the Logical Service Blueprint.
- **LogicalServiceBlueprintVersion**: Version of the Logical Service Blueprint.
- **ServiceDesignPackageID**: Unique identifier for the Service Design Package.
- **ServiceDesignPackageVersion**: Version of the Service Design Package.
- **ConceptualServiceBlueprintID**: Unique identifier of the related Conceptual Service Blueprint.
- **RequirementID**: Identifier of the related Requirement(s).
- **ServiceReleaseID**: Identifier of the related Service Release(s).

Key Data Object Relationships

The Logical Service Blueprint data object shall maintain the following relationships:

- **Conceptual Service Blueprint to Logical Service Blueprint** (1:n): The Conceptual Service Blueprint represents the high-level design of a service or changes to a service and leads to the creation of one or many Logical Service Blueprints.

- **Logical Service Blueprint to Requirement** (1:n): The Logical Service Blueprint is the service design that fulfills one or more Requirements.
- **Logical Service Blueprint to Service Release** (1:n): A Logical Service Blueprint can lead to the creation of one or more Service Releases.

Main Functions

The Service Design functional component:

- Shall be the system of record (authoritative source) for all Logical Service Blueprints.
- Shall associate a Logical Service Blueprint to a service.
- Can associate a Logical Service Blueprint to a Service Design Package.

If a Service Portfolio functional component exists, the Service Design functional component:

- Shall associate one or more Logical Service Blueprints to a Conceptual Service Blueprint.
- Can receive the Conceptual Service specification and design several Logical Service Blueprints that represent it.

If a Project functional component exists, the Service Design functional component:

- Can receive IT Initiative information which includes the scope and some content based on which the service is designed.

If a Requirement functional component exists, the Service Design functional component:

- Shall associate one or more Requirements to the Logical Service Blueprint.
- Can receive Requirement information from the Requirement functional component used to design the Logical Service Blueprint and create design specifications.

If a Release Composition functional component exists, the Service Design functional component:

- Shall associate a Logical Service Blueprint to one or more Service Releases which are detailed and designed to deliver the Logical Service Blueprint.

Model

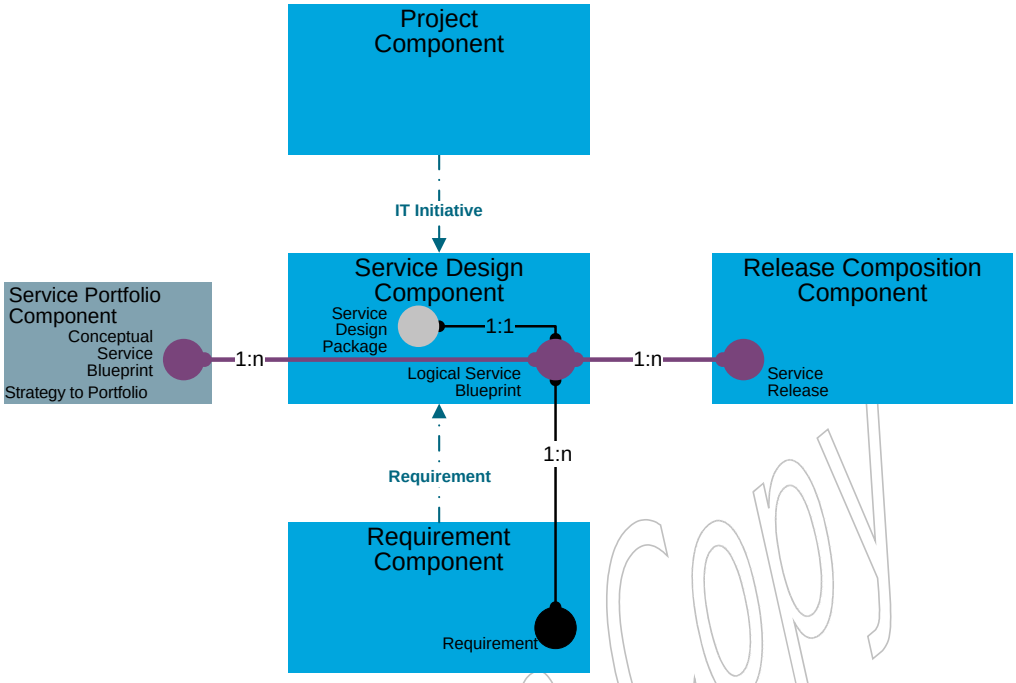


Figure 48: Service Design Functional Component Level 2 Model

6.4.4 Source Control Functional Component

Purpose

- Develop source code or infrastructure based on the Logical Service Blueprint, Service Design Package, and IT Initiative priorities.
- Ensure that the source code meets the design specifications, organizational policies, standards, and non-functional requirements so that the service can be operated successfully and meets customer expectations.
- Manage the development backlog of Requirements and Defects in accordance with the Service Design Package and Service Release.
- Receive Defects and input from the Defect functional component to enable the development of fixes or documented workarounds.
- Create automated test scripts including unit testing and scripts for static application security testing that follow a formal software security assurance methodology.
- On existing services being changed, run security tests on core code to identify existing security issues at the start of the development cycle so that assessment of scope/requirements set/schedule can be negotiated early.
- Manage source code images and store them in a Source data object repository.

- Develop automated Source compilation tools and procedures.
- Deliver the Source data object to the Build functional component to generate Builds.

Key Data Objects

- **Source** (data object): The created or purchased solution to meet the requirements for a particular Service Release.

Note: Source does not always equal "source code". Consider all use-cases such as "source code" for services produced on-premise, to contracts or entitlements for services simply subscribed to, to the purchase and implementation of a Commercial Off-The-Shelf (COTS) application.

Key Attributes

The Source data object shall have the following key data attributes:

- **SourceID**: Unique identifier of the Source.
- **SourceVersion**: Version of the Source.
- **BuildID**: Identifier of the related Build(s).
- **RequirementID**: Identifier of the related Requirement(s).

Key Data Object Relationships

The Source data object shall maintain the following relationships:

- **Source to Requirement** (n:m): Source will fulfill one or many Requirements, and for a given Service Release, there could be multiple Sources created/modified.
- **Source to Build** (1:n): Source can create one or many Builds.

Main Functions

The Source Control functional component:

- Shall be the system of record (authoritative source) for all Source.
- Shall manage the lifecycle of the Source.
- Shall allow recursive relationships between Source.
- Shall allow hierarchical relationships between Source.
- Shall associate Source to a service.

If a Requirement functional component exists, the Source Control functional component:

- Shall associate one or many Requirements to one or many Sources which includes the content that fulfills these Requirements.

If a Build functional component exists, the Source Control functional component:

- Shall associate one or many Builds to the related Source.

If a Defect functional component exists, the Source Control functional component:

- Can receive Defect information from the Defect functional component so Defects can be fixed in future versions of that Source.

Model

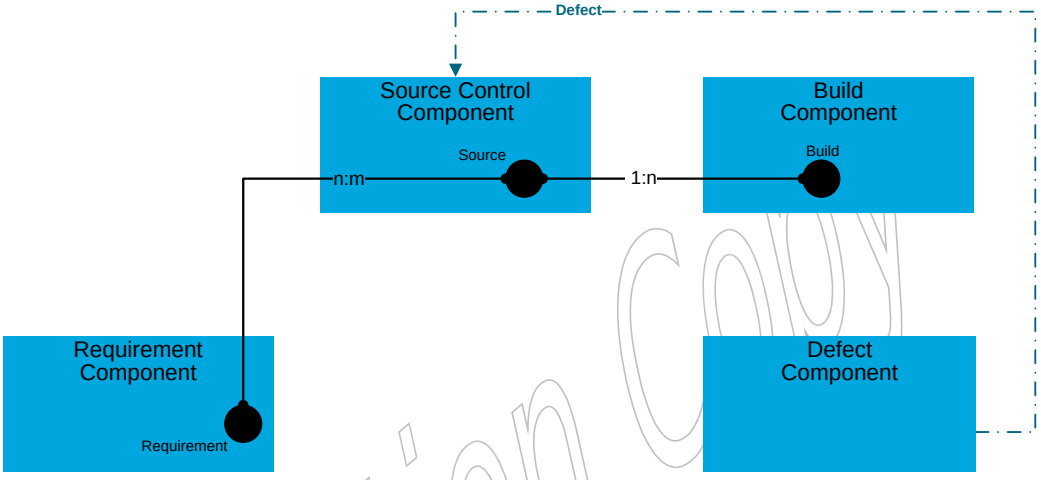


Figure 49: Source Control Functional Component Level 2 Model

6.4.5 Build Functional Component

Purpose

- Receive the Source data object from the Source Control functional component and manage the creation, implementation, automation, and security and storage of all Builds.
- Create Build from the Source data object for a particular service component.
- Automate the Build process to support the Build schedule and build frequency requirements in order to support daily Build and smoke test plans or continuous integration plans.
- Run dynamic application security testing no later than when the final Build data object is received and before the RFCs are created for moving the new or changed service into production.
- Manage Builds and versioning in a Definitive Media Library (DML).
- Develop automated Build storage procedures and automated compilation techniques and tools.
- Monitor and report on the results of each integration Build.

- Initiate or automate the delivery of Builds to the Build Package functional component for validation by the acceptance testing team as candidate release builds.

Key Data Objects

- **Build** (data object): Created from Source and versioned.

Key Attributes

The Build data object shall have the following key data attributes:

- **BuildID**: Unique identifier of the Build.
- **BuildVersion**: Version of the Build.
- **SourceID**: Identifier of the related Source.
- **TestCaseID**: Identifier of the related Test Case(s).
- **BuildPackageID**: Identifier of the related Build Package.

Key Data Object Relationships

The Build data object shall maintain the following relationships:

- **Source to Build** (1:n): Source can be built multiple times to create several Build versions.
- **Build to Test Case** (n:m): One or many Builds can be related to one or many Test Cases used as part of the Build creation.
- **Build Package to Build** (1:n): A Build Package is comprised of one or many Builds.

Main Functions

The Build functional component:

- Shall be the system of record (authoritative source) for all Builds.
- Shall manage the version of each individual Build.
- Shall associate a Build to a service.

If a Source Control functional component exists, the Build functional component:

- Shall associate Source to one or many Builds.

If a Test functional component exists, the Build functional component:

- Shall associate one or many Builds to one or many Test Cases which are executed as part of the Build creation.

If a Build Package functional component exists, the Build functional component:

- Shall associate one or many Builds to a Build Package.

Model

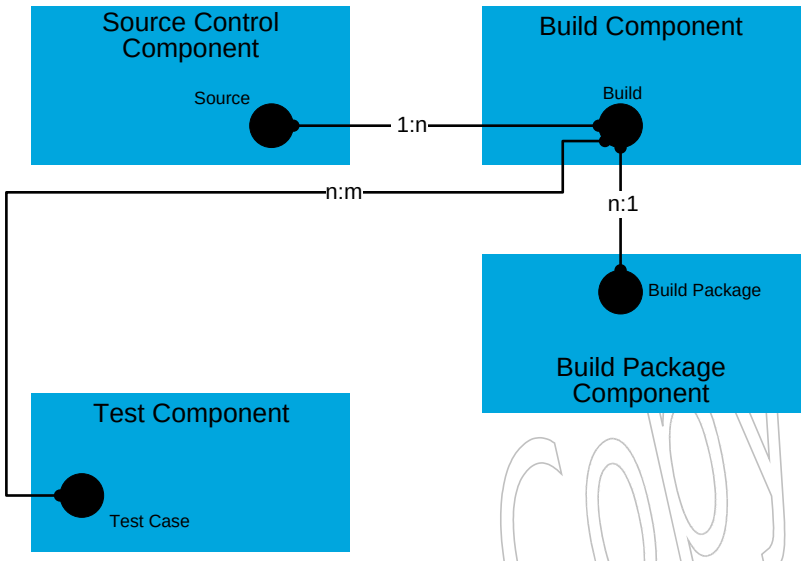


Figure 50: Build Functional Component Level 2 Model

6.4.6 Build Package Functional Component

Purpose

- Creation of a deployable package made up of one or many Builds.
- Manage the Build Packages and relationships to the Service Release Blueprints.

Key Data Objects

- **Build Package** (data object): A compilation of one or many Builds in a deployable package.

Key Attributes

The Build Package data object shall have the following key data attributes:

- **BuildPackageID**: Unique identifier of the Build Package.
- **BuildID**: Identifier of the related Build(s).
- **ServiceReleaseBlueprintID**: Identifier of the related Service Release Blueprint(s).

Key Data Object Relationships

The Build Package data object shall maintain the following relationships:

- **Build Package to Build** (1:n): The Build Package is comprised of one or more Builds.
- **Build Package to Service Release Blueprint** (n:m): One or more Build Packages can be associated to one or more Service Release Blueprints.

Main Functions

The Build Package functional component:

- Shall be the system of record (authoritative source) for all Build Packages.
- Shall associate a Build Package to a service.

If a Build functional component exists, the Build Package functional component:

- Shall associate one or more Builds to a Build Package.

If a Release Composition functional component exists, the Build Package functional component:

- Shall associate one or more Service Release Blueprints to one or more Build Packages.

Model

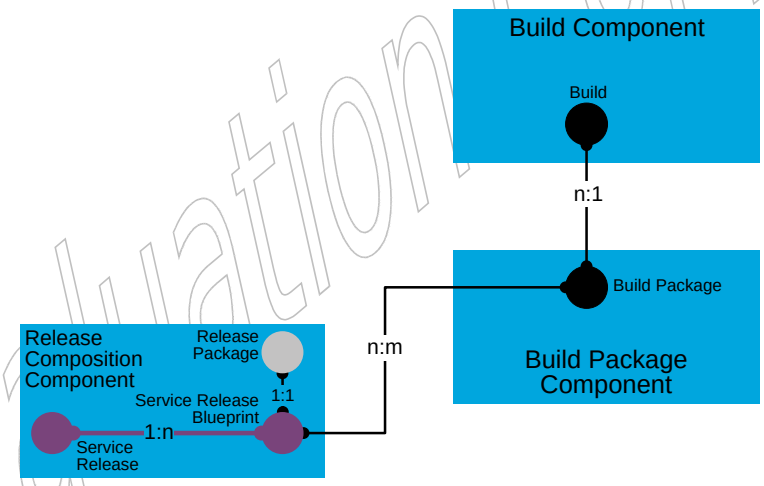


Figure 51: Build Package Functional Component Level 2 Model

6.4.7 Release Composition Functional Component

Purpose

- Manage the Release Package, Service Release, Service Release Blueprints, and overall Service Release for developing and delivering new or changed services to the R2F Value Stream Fulfillment Execution functional component to facilitate a smooth transition to IT operations.
- Create the Service Release, Service Release Blueprint, and Release Packages that will be utilized by the Test functional component and later the Fulfillment Execution functional

component (R2F Value Stream) to create a specific deployment for a specific IT service instance (including service system and/or service offer).

- Begin the creation of monitors, batch processing, backup/restore, etc. for the service, to ensure supportability as part of IT operations enablement.
- Manage the release artifacts within the Release Package by centralizing all elements of the Service Release Blueprint from the various functional components:
  - — Requirement functional component: requirements per Release Package
  - — Source functional component, as well as maintenance scripts: documentation
  - — Build Package functional component: Build Package
  - — Test functional component: test results as well as automated tests for validation post deployment
  - — Defect functional component: Known Errors (issues/defects)

Key Data Objects

- **Service Release** (data object): Represents the release of a given service.
- **Service Release Blueprint** (data object): The information and details related to a specific release to a specific environment.

Key Attributes

The Service Release data object shall have the following key data attributes:

- **ServiceReleaseID**: Unique identifier of the Service Release.
- **LogicalServiceBlueprintID**: Unique identifier of the related Logical Service Blueprint.
- **ITInitiativeID**: Unique identifier of the related IT Initiative.
- **ServiceReleaseBlueprintID**: Identifier of the related Service Release Blueprint(s).
- **RequirementID**: Identifier of the related Requirement(s).
- **TestCaseID**: Identifier of the related Test Case(s).

The Service Release Blueprint data object shall have the following key data attributes:

- **ServiceReleaseBlueprintID**: Unique identifier of the Service Release Blueprint.
- **ServiceReleaseBlueprintDescription**: Description of the Service Release Blueprint.
- **MasterServiceID**: Unique identifier of the related service.
- **ServiceReleaseID**: Unique identifier of the related Service Release.
- **BuildPackageID**: Unique identifier of the related Build Package.

- **DesiredServiceModelID**: Unique identifier of the related Desired Service Model(s).
- **FulfillmentID**: Identifier of the related Fulfillment Request(s).
- **ServiceContractID**: Identifier of the related Service Contract(s).
- **ServiceID**: Identifier of the related Service Catalog Entry(ies).
- **DefectID**: Identifier of the related Defect(s).

Key Data Object Relationships

The Service Release data object shall maintain the following relationships:

- **Logical Service Blueprint to Service Release** (1:n): A Logical Service Blueprint can lead to the creation of one or more Service Releases.
- **IT Initiative to Service Release** (1:n): An IT Initiative will manage the creation of one or more Service Releases defined to deliver the content of the IT Initiative.
- **Service Release to Service Release Blueprint** (1:n): A Service Release can be released to multiple environments based on the associated Service Release Blueprints.
- **Service Release to Requirement** (1:n): The Service Release delivers a service which fulfills one or more Requirements.
- **Service Release to Test Case** (1:n): A Service Release can be validated by one or many Test Cases/

The Service Release Blueprint data object shall maintain the following relationships:

- **Service Release to Service Release Blueprint** (1:n): A Service Release can be released to multiple environments based on the associated Service Release Blueprints.
- **Service Release Blueprint to Build Package** (n:m): One or more Build Packages can be associated to one or more Service Release Blueprints.
- **Service Release Blueprint to Desired Service Model** (1:n): One Service Release Blueprint can be translated to one or more Desired Service Models.
- **Service Release Blueprint to Fulfillment Request** (1:n): One Service Release Blueprint is used for service instantiation by one or many Fulfillment Requests.
- **Service Release Blueprint to Service Contract** (n:m): One or more Service Release Blueprints contain the template of one or more Service Contracts.
- **Service Catalog Entry to Service Release Blueprint** (1:n): Each Service Catalog Entry is created based on definitions of a Service Release Blueprint.
- **Service Release Blueprint to Defect** (n:m): One or more Service Release Blueprints can contain one or may Defects in the form of Problems/Known Errors.

Main Functions

The Release Composition functional component:

- Shall be the system of record (authoritative source) for all Service Releases.
- Shall associate a Service Release to a service.
- Shall allow a recursive relationship between Service Releases.
- Shall associate a Service Release to one or more Service Release Blueprints.
- Shall be the system of record for all Service Release Blueprints.
- Shall associate a Service Release Blueprint to a service.
- Shall associate a Service Release Blueprint to a Release Package.

If a Project functional component exists, the Release Composition functional component:

- Shall associate one IT Initiative to one or more Service Releases which are defined to deliver this IT Initiative.

If a Service Design functional component exists, the Release Composition functional component:

- Shall associate one Logical Service Blueprint to one or more Service Releases which are designed to deliver this Logical Service.

If a Requirement functional component exists, the Release Composition functional component:

- Shall associate one Service Release with one or more Requirements which are fulfilled in this release.

If a Test functional component exists, the Release Composition functional component:

- Shall associate one Service Release with one or more Test Cases.
- Can receive test-related information that should be included in the Release Package from Test Management.

If a Build Package functional component exists, the Release Composition functional component:

- Shall associate one or more Service Release Blueprints to one or more Build Packages.
- Can receive one or more Build Packages that should be included in the Service Release Blueprint.

If a Service Level functional component exists, the Release Composition functional component:

- Can provide service contract information for creating a Service Contract.
- Shall associate one or more Service Release Blueprints to one or more Service Contracts.

If a Fulfillment Execution functional component exists, the Release Composition functional component:

- Can provide information required for service instantiation to the Fulfillment Execution functional component.
- Shall associate a Service Release Blueprint to one or more Desired Service Models.
- Shall associate a Service Release Blueprint to one or more Fulfillment Requests.

If a Catalog Composition functional component exists, the Release Composition functional component:

- Can provide information required for creating a Service Catalog Entry to the Catalog Composition functional component.
- Shall associate a Service Release Blueprint to one or more Service Catalog Entry(ies).

If a Defect functional component exists, the Release Composition functional component:

- Shall associate one or more Service Release Blueprints to one or more Defects.
- Can receive Defect-related information that should be included in the Release Package.

Model

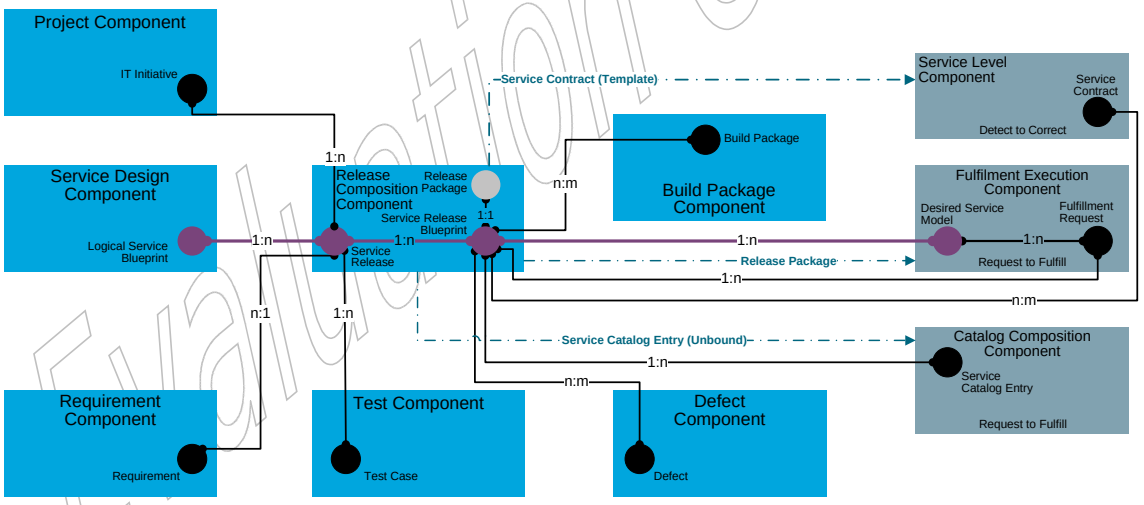


Figure 52: Release Composition Functional Component Level 2 Model

6.4.8 Test Functional Component

Purpose

- Trace to Requirements.
- Plan and execute tests that ensure the IT service will support the customer's requirements at the agreed service levels.

- Create Defect data objects that are consumed by the Defect functional component.
- Plan and design tests, including automated test scripts for both code images and monitors, using input from service development.
- Prepare test environment.
- Execute tests, including functionality, usability, acceptance, risk-based security (dynamic application security and infrastructure security testing), performance, and stress testing.
- Create Defects found during testing which are consumed by the Defect functional component.
- Manage test data, drive automation, and re-use test automation and test scripts where appropriate.
- Provide test execution reports for the tested Requirements.
- Ensure that the operations tooling works as expected (monitors, etc.).

Key Data Objects

- **Test Case** (data object): The Test Case is used to validate that the Service Release is fit for purpose.

Key Attributes

The Test Case data object shall have the following key data attributes:

- **TestCaseID**: Unique identifier of the Test Case.
- **TestCaseSummary**: Summary or short description of the Test Case.
- **TestCaseStatus**: Status of the Test Case.
- **ServiceReleaseID**: Unique identifier of the related Service Release.
- **BuildID**: Identifier of the related Build.
- **RequirementID**: Identifier of the related Requirement(s).
- **DefectID**: Identifier of the related Defect(s).

Key Data Object Relationships

The Test Case data object shall maintain the following relationships:

- **Requirement to Test Case** (1:n): A Requirement is associated to one or more Test Cases that validates this Requirement.
- **Service Release to Test Case** (1:n): A Service Release is associated to one or more Test Cases which are executed as part of this Service Release.

- **Test Case to Build** (n:m): One or more Test Cases can be associated with one or more Builds that uses this Test Case as part of the Build creation.
- **Test Case to Defect** (1:n): One Test Case can be associated to one or more Defects that are reported as a result of this test.

Main Functions

The Test functional component:

- Shall be the system of record (authoritative source) for all Test Cases.
- Shall manage the lifecycle of the Test Case.
- Shall allow recursive relationships between Test Cases.
- Shall associate a Test Case to a service.

If a Build functional component exists, the Test functional component:

- Shall associate one or more Test Cases to one or more Builds that uses this Test Case as part of the Build creation.

If a Requirement functional component exists, the Test functional component:

- Shall associate a Requirement to one or more Test Cases that validates this Requirement.

If a Defect functional component exists, the Test functional component:

- Shall associate a Test Case to one or more Defects that result from this test.
- Shall provide Defect information to the Defect functional component.

Model

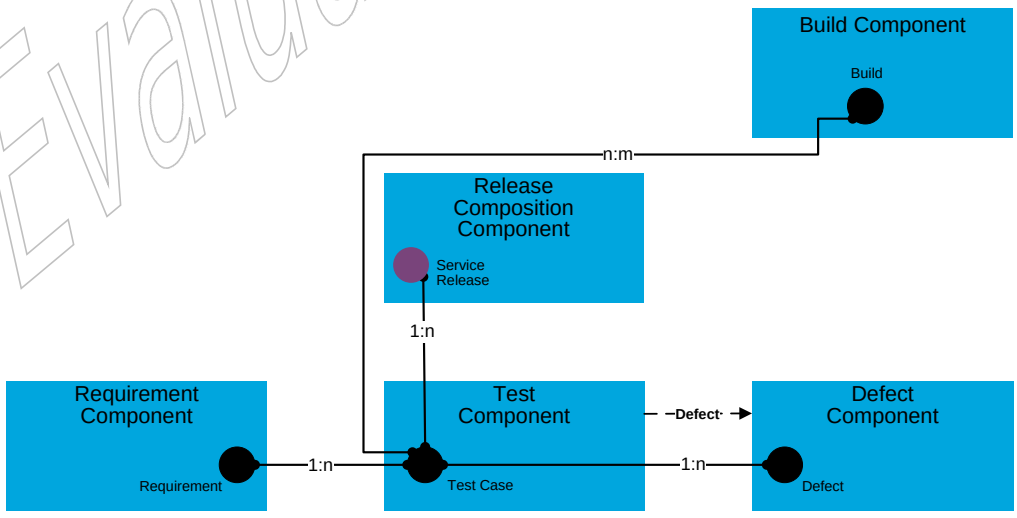


Figure 53: Test Functional Component Level 2 Model

6.4.9 Defect Functional Component

Purpose

- Keep track of all Defects; including their origin, status, importance, and relation to Requirements and Known Errors.
- Register Defects of all types (including security-related) with all relevant details such as description, severity, application version, related requirement, etc.
- Analyze Defects and find resolution.
- Associate Defects with Requirements.
- Document issues that should be communicated to the Release Composition functional component.
- Consume Defects from the D2C Value Stream Problem functional component as well as the Test functional component that are in turn consumed by the Source Control functional component for review and resolution.
- Update Defect details.
- Decide on target release.
- Report Defect status and provide Defect reports.
- Convert Defects not resolved by service development to Known Errors for Problem Management (D2C Value Stream) to document or develop work-around and report in knowledge management articles.

Key Data Objects

- **Defect** (data object): An issue with the Service Release Blueprint which should be remediated to fulfill the associated Requirements.

Key Attributes

The Defect data object shall have the following key data attributes:

- **DefectID**: Unique identifier of the Defect.
- **DefectDescription**: Description of the Defect.
- **DefectStatus**: Status of the Defect.
- **ServiceReleaseBlueprintID**: Unique identifier of the related Service Release Blueprint.
- **TestCaseID**: Identifier of the related Test Case.
- **KnownErrorID**: Identifier of the related Known Error.

Key Data Object Relationships

The Defect data object shall maintain the following relationships:

- **Test Case to Defect** (1:n): One Test Case can be associated to one or more Defects that results from the test.
- **Defect to Service Release Blueprint** (n:m): One or more Service Release Blueprints are associated to one or more Defects which are included in the Release Package as Problems/Known Errors.
- **Known Error to Defect** (1:1): A Known Error is associated to a Defect when the Known Error is found to be a Defect.

Main Functions

The Defect functional component:

- Shall be the system of record (authoritative source) for all Defects.
- Shall manage the lifecycle of the Defect.
- Shall associate a Defect to a service.

If a Release Composition functional component exists, the Defect functional component:

- Shall associate one or more Service Release Blueprints to one or more Defects which reflects Defects that should be published as Problems/Known Errors.

If a Source Control functional component exists, the Defect functional component:

- Can provide Defect information to the Source Control functional component.

If a Test functional component exists, the Defect functional component:

- Shall receive Defect information from the Test functional component.
- Shall associate a Test Case to one or more Defects.

If a Problem functional component exists, the Defect functional component:

- Shall associate a Known Error to a Defect.
- Shall receive Defect information from a Known Error.

Model

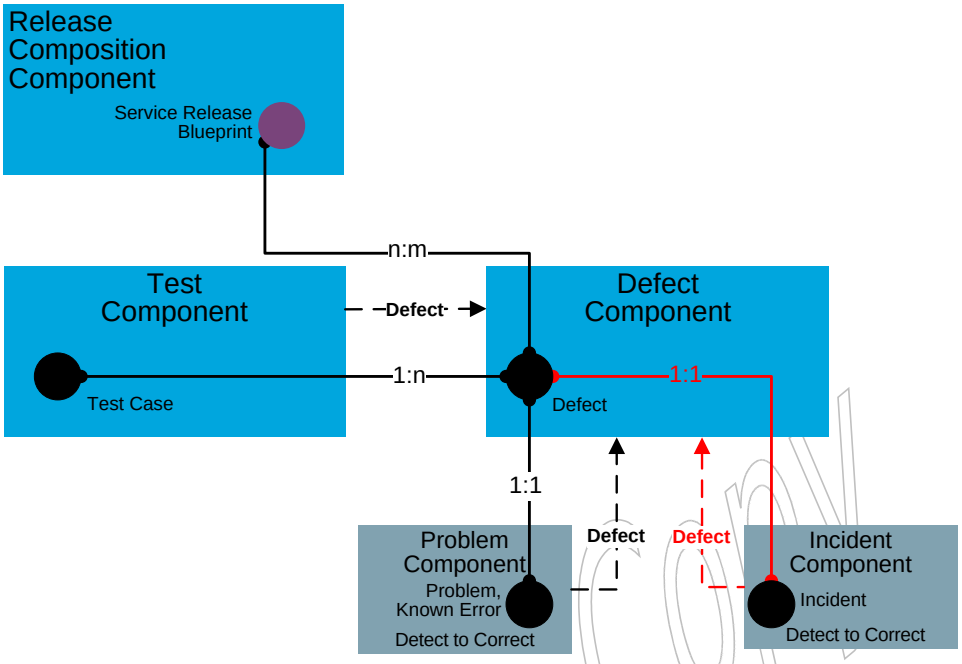


Figure 54: Defect Functional Component Level 2 Model

7 Request to Fulfill (R2F) Value Stream

This chapter provides an overview of the Request to Fulfill (R2F) Value Stream – one of four IT Value Streams that comprise the IT Value Chain. It describes the business context, objectives, and details behind the R2F Value Stream.

7.1 Objectives

The R2F Value Stream represents a modern, consumption-driven engagement model and goes beyond the traditional IT service request management. It is a framework for connecting the various consumers (business users, IT practitioners, or end customers) with goods and services that they need to drive productivity and innovation. It fosters service consumption and fulfillment, knowledge sharing, self-service support, and collaboration between communities of interest to improve the overall engagement experience with IT.

Many organizations use multiple IT requests and/or Service Catalogs to address the needs of their consumers. The R2F Value Stream brings these different catalogs and consumer personas into a single consumption experience thereby eliminating the complexity and confusion consumers experience today. The goal of the R2F Value Stream is to provide a blueprint for creating a streamlined consumption experience that consistently engages consumers and eliminates the need for them to avoid working with their IT organization.

The key to success with R2F is dependent upon two primary factors:

- The ability to package deliverables as offers that consumers recognize and value, abstracting away confusing technology choices and complex fulfillment processes. This is accomplished by leveraging the Service Model to create Service Catalog Entries that can be instantiated and consumable offers that can be requested/ordered.
- The ability to present and manage an inviting consumption experience that exposes a variety of opportunities to acquire services, goods, knowledge, and/or support. This requires organizations to go beyond the traditional IT request catalog solutions.

R2F data objects are realized through these primary functional components:

- The **Engagement Experience Portal** functional component represents the modern IT engagement/consumption experience which exposes a variety of opportunities to acquire services, goods, knowledge, and/or support.
- The **Offer Consumption** and **Offer Management** functional components present offers to consumers and enables the shopping experience. Offers are based on Service Catalog Entries developed in the Catalog Composition functional component; they are created from the consumer point of view and can be tailored to different personas, roles, or functions using profiling.

- The **Catalog Composition** functional component enables the aggregation of catalogs from multiple suppliers into a single Offer Catalog and the composition of Service Catalog Entries. Service Catalog Entries are created from the provider point of view and may have some level of fulfillment details exposed.
- The **Request Rationalization** functional component rationalizes the order/request into individual Fulfillment Requests and authorizes Subscriptions.
- The **Fulfillment Execution** functional component routes the individual Fulfillment Requests to the appropriate fulfillment engines.
- The **Usage** and **Chargeback/Showback** functional components track the service usage and control the chargeback contract. Actual chargeback and invoicing is handled by financial systems within most companies. Further, service usage data can be used to drive continuous improvement into service offerings. Using data to provide insight into how services are being used helps shape demand for improvements and new services. This helps minimize the dependency on intrusive and time-consuming "requirements gathering sessions" with consumers.

The common limitations for current R2F practices include:

- A service consumption experience that exposes technology resources and/or IT capabilities rather than valued services.
- Multiple catalogs required for consumers to navigate in order to find and request available services.
- Too many customer service requests requiring creation of fulfillment incidents, projects, and/or human intervention resulting in delays and an unfavorable experience overall.
- Lack of service Subscription, Usage, and chargeback traceability.

7.2 Business Value Proposition

The R2F Value Stream places emphasis on time-to-value, repeatability, and consistency for consumers looking to request and obtain services from IT. It optimizes both service consumption and fulfillment experiences by delineating between the creation of offers and catalog aggregation and Service Catalog Entry composition.

Today IT organizations struggle to increase the ratio of self-sourced services over workflow-based fulfillment requiring direct human intervention. Many fulfillments today require too much intervention that consumes valuable IT resources. By increasing self-sourcing, companies will see improved business velocity and reduction in friction. They will also be able to reduce "shadow-sourcing" within the lines of business because of a more responsive consumption experience. Today's IT is focused on delivery of technical capabilities – tomorrow's IT must be positioned to focus on facilitating consumption of multi-sourced services.

The R2F Value Stream emphasizes the importance of deploying standard changes (a form of request management) rather than normal changes for internal fulfillments. Normal, risk-assessed and individually approved changes are one of the most time-consuming and resource-intensive

activities in an enterprise-wide IT organization. Service fulfillments should be based on standard Service Models where request criteria and approvals are designed into the model and fully automated fulfillment processes orchestrate all necessary provisioning and Change Management.

The R2F Value Stream plays an important role in helping IT organizations advance toward a service broker model. Such a model is dependent on the organization's ability to leverage both internal and external sourcing options for satisfying consumer demand. Enterprise IT organizations have been using external suppliers for goods and services for many years. However, the multi-sourcing environment will become more complex as companies expand their use of cloud-based offerings. The R2F Value Stream enables the aggregation of catalogs and catalog entries from multiple providers into a single consumption experience. Therefore, while there is complexity on the delivery side in managing the various catalogs and catalog entries, it is not exposed to the consumer and the ordering experience is seamless and inviting.

The key value propositions for adopting the R2F Value Stream are:

- Provides a blueprint for increasing business innovation velocity by facilitating a service consumption experience that allows consumers to easily find and subscribe to goods and services through a self-service engagement model.
- Provides a functional framework that delineates between a single Offer Catalog and multiple Catalog Compositions to reduce complexity in the IT shopping experience.
- Provides an architectural foundation for moving from traditional IT request management to service brokerage that increases both business and IT effectiveness.
- Increased fulfillment efficiency and consistency through standard change deployment and automation.
- Provides holistic visibility and traceability across service Subscription, Usage, and chargeback to improve IT Financial Management.
- Enables increased cost optimization; for example, by canceling expired Subscriptions and reclaiming resources, Subscriptions, and/or licenses that are unused.

7.3 Key Performance Indicators

The R2F Value Stream critical success factors and Key Performance Indicators (KPIs) are as follows:

Critical Success Factors	Key Performance Indicators (KPIs)
Ability to Meet Customer Expectations	New or modified Subscriptions per time period % and number of Subscription requests complying or breaching SLA or OLA agreements Number of Subscription requests accepted and rejected by the requestor for the first time right delivery/fulfillment Variation in the average time to fulfill Subscription requests for the predictability of delivery Number of Incidents related to request fulfillment Arrival and departure rate of service requests
Reduce Costs	Costs (burned resources) per service and per fulfillment step Breakdown of self-source fulfillments *versus* one-off fulfillments % and number of fulfillments requiring human intervention to be completed Number of service request queues being managed
External Service Provider Compliance	Number of purchase orders per time period % and number of orders delivered and accepted complying with underpinning contract agreements % and number of delivered orders breaching underpinning contract agreements Number of Incidents related to the purchase order fulfillment Number of purchase orders unfulfilled at the end of a given period Number of orders delivered and accepted by the requestor per time period Number of purchase orders rejected via no delivery or cancelled purchase orders
Increase Speed/Agility/Flexibility (Operational Performance)	Completed service requests Service request work-in-progress Number of interactions with consumers per service during delivery % of work-in-progress within SLA % of completed work within SLA

7.4 Value Stream Definition

The Request to Fulfill (R2F) Value Stream contains primary and secondary functional components. Primary functional components are core to the value stream and are essential for managing the service at this stage of its lifecycle. Secondary functional components are not dedicated to R2F but provide relevant data objects to primary functional components. R2F includes the following primary functional components that provide the technology necessary to support IT portfolio activities:

- Offer Consumption
- Offer Management
- Catalog Composition
- Request Rationalization
- Fulfillment Execution
- Usage
- Chargeback/Showback
- Knowledge & Collaboration

A functional component utilizes an input data object to conduct key activities and may also provide output data objects to another functional component that requires that data.

The R2F Value Stream primarily focuses on system of record integrations between two functional components like other value streams. However, the R2F Value Stream also includes system of engagement integrations enabling a common user experience or graphical user interface mash-up. The R2F Value Stream functional components with system of engagement integrations are the Engagement Experience Portal and Offer Consumption functional components. The Offer Consumption functional component integrations include system of engagement integrations with the following functional components: Offer Management, Request Rationalization, and Chargeback/Showback.

The Engagement Experience Portal functional component is a secondary functional component since its scope can expand beyond the IT4IT Reference Architecture and into the business domain. It exposes and facilitates a unified engagement between consumers and the IT functions. The main objectives of the Engagement Experience Portal are to:

- Drive the consumption through the Offer Catalog
- Enable collaboration between communities of interest
- Obtain support through a self-service interface
- Access knowledge that enables them to be better informed about services offered by IT

There may be other engagement experiences presented through the portal but this is the core set called out explicitly in the R2F Value Stream.

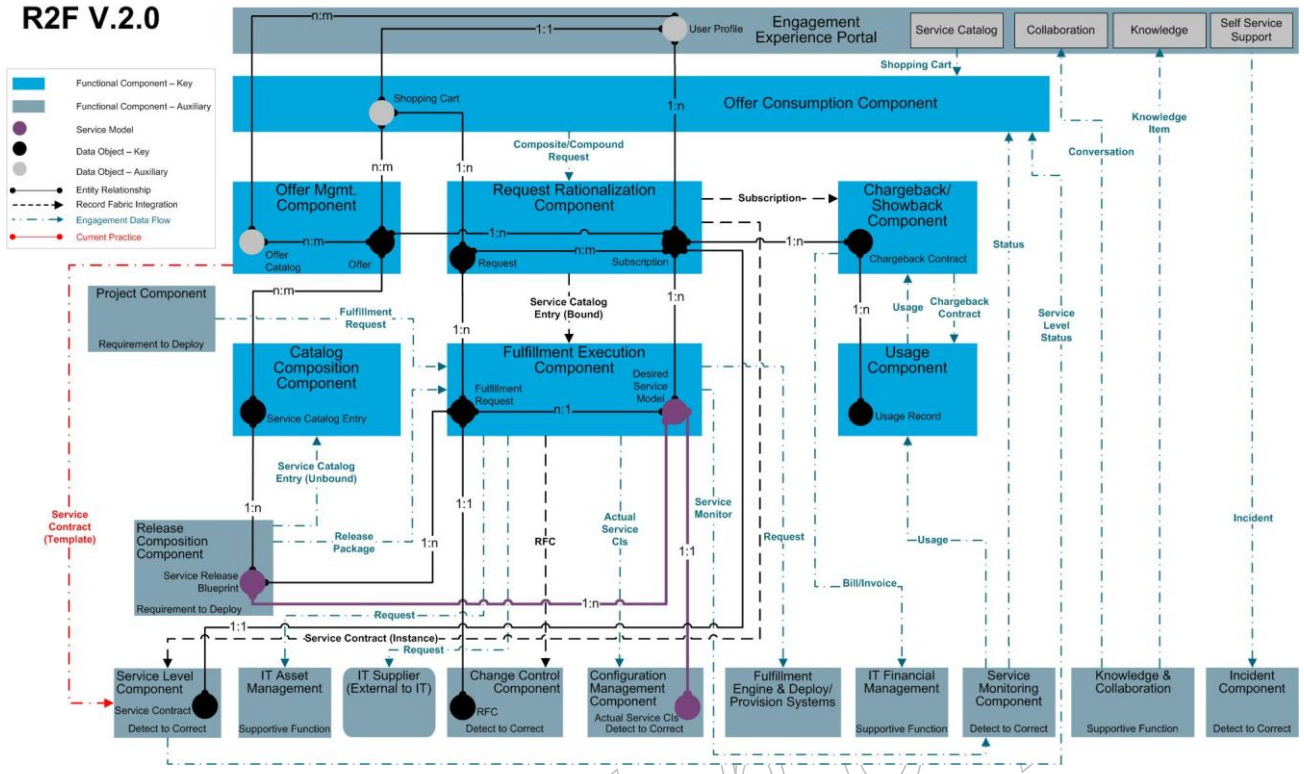


Figure 55: Request to Fulfill Level 2 Value Stream Diagram

7.4.1 Engagement Experience Portal (Secondary Functional Component)

Purpose

- The Engagement Experience Portal functional component is based on a system of engagement integration design pattern where consumers access different functional components through a common user experience.
- Consumers manage certain aspects of their profile; for example, localization, preferred method of communication, user interface settings, and so on.
- Facilitate service consumption by connecting any potential consumer with the right information, goods, services, or capability at the right time through a single experience, taking into account the consumer profile.
  - — Provide an intuitive experience that draws consumers in instead of being viewed as an inhibitor to productivity that is forced upon them.
  - — Provide a self-configurable experience (such as mash-ups) so that the different types of consumers can tune the experience to best suit their needs.
  - — Provide an interface supported across multiple devices such as smartphones, tablets, etc.

— Provide plug-and-play connectivity for components that need to be exposed through the portal.

— Components for connectivity include but are not limited to catalog-driven service consumption.

- Self-service support:

  — Community and collaboration

  — Knowledge and content associated with services and capabilities

  — Information about consumed services

  — Information from the Service Monitoring and the Service Level functional components (D2C Value Stream) such as the (current) service status

Key Data Objects

- **User Profile** (data object): Personal data associated with a specific user and the explicit digital representation of a person's identity. A User Profile consists of different attributes managed by a user or consumed by other authoritative sources. User Profile attributes must be secure, protected, and access restricted based on roles (e.g., HR Manager) or system.

Key Attributes

The User Profile data object shall have the following key data attributes:

- **UserID**: Unique Identifier of the user.
- **UserName**: Full name of the user.
- **Role**: Used to grant access to functionality and information.

Key Data Object Relationships

The User Profile data object shall maintain the following relationships:

- **User Profile to Offer Catalog** (n:m): Present a personalized list of offers from the catalog depending on the consumer profile.
- **User Profile to Shopping Cart** (1:1): The catalog items which are ordered need to link to the consumer that will receive the items. If an approval step is required, the User Profile helps to identify the authorized person by looking up information from the organizational hierarchy.
- **User Profile to Subscription** (1:n): A Subscription is created and linked to the user for every service that has been ordered and fulfilled where a Subscription is required.

Main Functions

The Engagement Experience Portal functional component:

- Shall be available to all users that desire to consume IT services.
- Shall expose various IT functions and capabilities in a single place, unifying the experience. These functions and capabilities may be exposed in a form similar to smartphone apps.
- May allow consumers to manage their User Profile (to varying degrees as some attributes may be provider-controlled).

Model

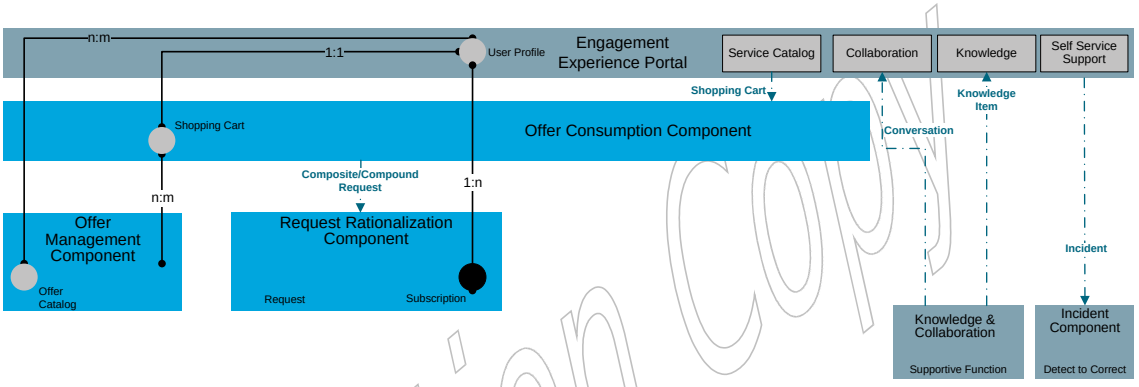


Figure 56: Engagement Experience Portal Level 2 Model

7.4.1.1 Service Catalog Functional Sub-Component

The Service Catalog functional sub-component enables consumers to engage with and consume services through the Offer Consumption functional component. This includes but is not limited to ordering new services, modifying Subscriptions, viewing the status of existing services or requests, and costs associated with services. For a detailed description of the functionality of the Service Catalog, refer to Section 7.4.2.

7.4.1.2 Collaboration Functional Sub-Component

The Collaboration functional sub-component provides the user front end for an enterprise collaboration experience, such as a chat capability.

7.4.1.3 Knowledge Functional Sub-Component

The Knowledge functional sub-component provides the interface for users to search and read Knowledge data objects of all types and sources. Knowledge data objects may include but are not limited to IT or supplier-created technical briefs, training videos, and user-created content.

7.4.1.4 Self-Service Support Functional Sub-Component

The Self-Service Support functional sub-component:

- Provides service consumers with a way to address more of their IT-related issues, as well as receive information regarding their existing records without necessarily engaging IT providers
- Reduces the load on the IT support organization by enabling and promoting self-help and self-healing behavior through the use of communities, knowledge sharing, content, etc.

The Self-Service Support functional sub-component contains the following data object:

- **Incident**: Create an Incident in the Incident functional component. For a detailed description, refer to Section 8.4.3.

The Self-Service Support functional sub-component:

- Shall enable users to create new support tickets for issues and/or questions that they were not able to resolve, or their questions that remain unanswered.
- Shall enable users to view and update their existing support tickets.
- Can route users to access the Knowledge data objects before a new support ticket is created.
- Can incorporate the Collaboration functional sub-component to enable communication with peers and IT staff in order to gain knowledge or resolve their IT-related issues.

7.4.2 Offer Consumption Functional Component

Purpose

- Present service offers to various service consumers; facilitate consumption/management of and payment for IT services rendered.
- Utilize the User Profile to present a personalized experience which includes consumer-specific information such as personal preferences, location, or job function.
- Present consumable offers derived from Service Catalog Entries with associated descriptions, pictures, prices, and purchasing options to prospective consumers.
- Enable consumers to manage their Subscriptions by:
  - — Viewing Subscription/service status, costs, usage, etc.
  - — Adding new Subscriptions, ordering new services, or service instances
  - — Modifying Subscription parameters (also known as upgrading/downgrading)
  - — Cancelling/ending service Subscriptions

Key Data Objects

- **Shopping Cart** (data object): Contains the IT services that the user wants to order; the object only exists during the actual shopping session. Upon submission, a request is generated that is comprised of the content contained in the Shopping Cart.

Note: Shopping Cart is an auxiliary data object (not key) but is being described here to help with the reader's understanding of the R2F Value Stream.

Key Attributes

The Shopping Cart data object shall have the following key data attributes:

- **ShoppingCartID**: Unique identifier for the Shopping Cart.
- **UserID**: Unique identifier for the user (from User Profile).
- **ApproverID**: Based on the user, the identifier of the approver (e.g., user ID of the manager).
- **Status**: Controls the status of the approval steps/workflow if needed.

Multiple services can be ordered for every item ordered in one Shopping Cart, therefore these attributes apply per item in the Shopping Cart:

- **LineItem**: Multiple items can be ordered in one go.
- **OfferID**: Identifier of the Offer (from Offer).
- **ReqValue**: Based on the Offer the user might need to provide values/options for a successful fulfillment (from Offer).

Key Data Object Relationships

The Shopping Cart data object shall maintain the following relationships:

- **Shopping Cart to User Profile** (1:1): The contents of the Shopping Cart relate to a specific user, who is actually ordering the services.
- **Shopping Cart to Offer** (n:m): The Shopping Cart only contains those items available to the end-user from the existing Offers.
- **Shopping Cart to Request** (1:n): Represents the relationship between the Shopping Cart and the Requests necessary to fulfill the services ordered in the shopping experience.

Main Functions

The Offer Consumption functional component:

- Can provide information on the existing Subscription to enable the user to change/cancel existing Subscriptions.
- Shall provide all necessary information to guarantee the fulfillment.
- Can provide functionality to order multiple offers in one transaction.
- Can enable consumers to order services on behalf of other consumers.

- Can provide visibility to information about the user's specific service consumption including pricing, usage, etc.

If the Service Level functional component exists, the Offer Consumption functional component:

- Shall expose information on the Service Level status for the services the user subscribed to.

Model

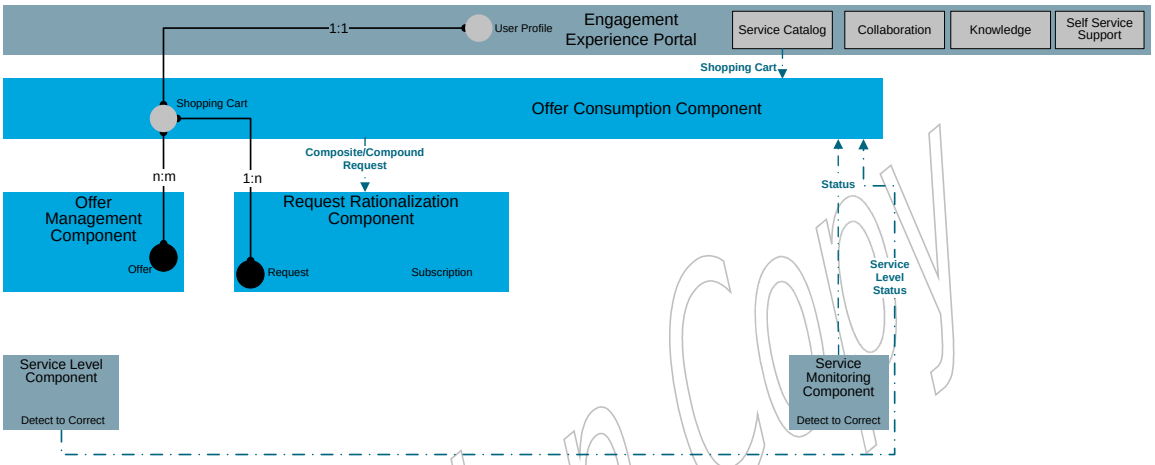


Figure 57: Offer Consumption Functional Component Level 2 Model

7.4.3 Offer Management Functional Component

Purpose

- Aggregate (mash-up) all Catalog Composition items and external supplier catalogs into consumable Offers that users can order through the Offer Consumption functional component.
- Build and publish the various offerings into Offer Catalogs for various populations to consume, determine prices, and valid options that consumers can select.
- Enable Offers to be grouped into an Offer Catalog to expose them as a collection of consumable items for a given group of consumers.
- Fulfill each Offer through numerous underlying Catalog Compositions as determined by this functional component.

Key Data Objects

- **Offer** (data object): Defines how a Service Catalog Entry will be instantiated and under what terms and conditions – price, deployment, approval, workflow, service level (contract), etc.

- **Offer Catalog** (data object): A set or collection of Offers that are grouped together as something that can be consumed by certain consumers or consumer groups.

Key Attributes

The Offer data object shall have the following key data attributes:

- **OfferID**: Unique identifier for every Offer in the catalog.
- **CatalogID**: Identify in which catalog this Offer is available (from Offer Catalog).
- **OfferName**: Description of the Offer for consumers to identify/search Offers.
- **StartDate**: Date/time on which the Offer may be consumed.
- **ExpiryDate**: Date/time on which the Offer is no longer available.
- **Status**: Indicates if the Offer is ready for consumption (e.g., draft, published, retired, etc.).
- **Price**: If applicable, the pricing information on the service, including the type of Subscription.
- **ReqValue**: Mandatory options or variables linked to the service which need to be provided by the consumer to prevent issues during the fulfillment. (Some of) these options or variables might not be selectable for customers, but are pre-filled by the Offer itself upon creation of the Offer.

The Offer Catalog data object shall have the following key data attributes:

- **CatalogID**: Unique identifier for the Offer Catalog.
- **CatalogName**: Offer Catalog name used for consumers.
- **ServiceID**: Identifier for the service (from Service Catalog Entry).
- **Roles**: Authorization role required for access (from User Profile).

Key Data Object Relationships

The Offer data object shall maintain the following relationships:

- **Offer to Service Catalog Entry** (n:m): Ensures all required information is captured for the fulfillment (deployment/delivery) of the service.

The Offer Catalog data object shall maintain the following relationships:

- **Offer Catalog to Offer** (n:m): Represents the collection of Offers that comprise each Offer Catalog.
- **Offer Catalog to User Profile** (n:m): Represents which users can access/consume each Offer Catalog.

Main Functions

The Offer Management functional component:

- Shall contain all of the Offers available to consumers and provide this information to the Offer Consumption functional component.
- Can group services from multiple providers (internal and external) into a single Offer (also known as a bundle of services).
- May create the Service Contract template and provide information to the Service Level functional component.

Model

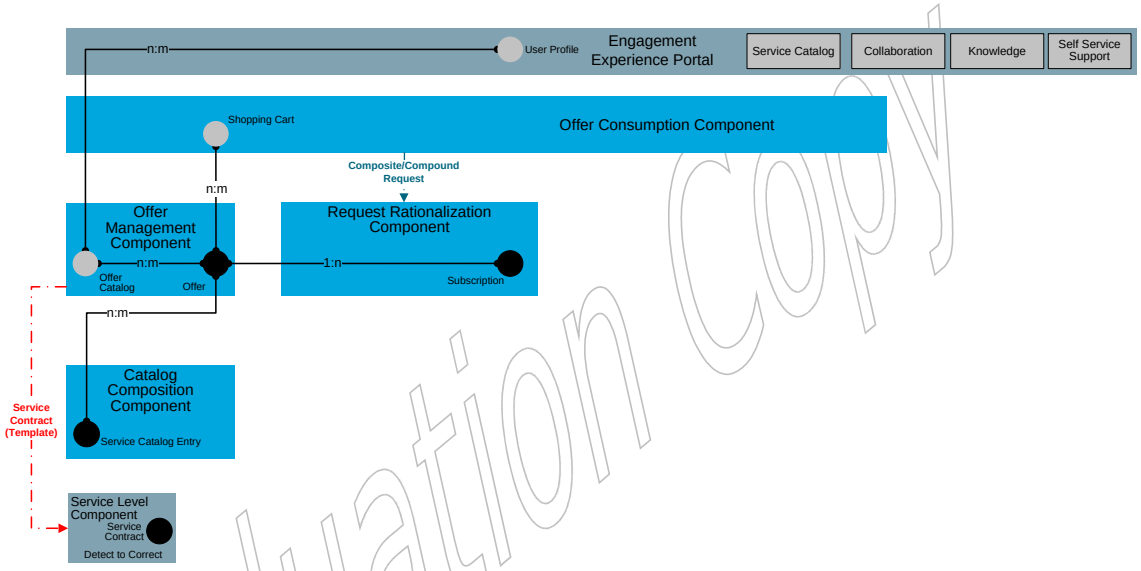


Figure 58: Offer Management Functional Component Level 2 Model

7.4.4 Catalog Composition Functional Component

Purpose

- Create and publish the Service Catalog Entries, including all of their dependencies, at the level at which these can be presented as Offers in the Offer Management functional component.
- Service Catalog Entries are created from the Service Release Blueprint in the Release Composition functional component (R2D Value Stream). Services, as well as their dependencies and details, are accurately defined, including supplying the necessary information for the service to be instantiated.
- Service Catalog Entries are created and updated to prepare them for consumption, including configurable options (e.g., pricing, subscription terms, bundles, service level, support conditions, etc.).

Key Data Objects

- **Service Catalog Entry** (data object): A Logical Service Blueprint within the R2F Value Stream managed by the Offer Management functional component. It is an authoritative source for the consolidated set of IT services that can be presented as Offers. An unbound part of the superset of IT service definitions is available for promotion to the status of Offer. A Service Catalog Entry would need to be bound to aspects relating to the delivery of the IT service such as price, deployment, approval, workflow, and service level in order to be promoted to Offer.

Key Attributes

The Service Catalog Entry data object shall have the following key data attributes:

- **ServiceID**: Unique identifier for every service in the catalog.
- **ServiceName**: Name of the service used for catalog and offer creators.
- **ReqValue**: Provide a set of values/variables that are needed upon fulfillment.

Key Data Object Relationships

The Service Catalog Entry data object shall maintain the following relationships:

- **Service Catalog Entry to Service Release Blueprint** (1:n): Ensure all catalog entries relate to the specific service definitions used for fulfillment.
- **Service Catalog Entry to Offer** (n:m): Ensure all information needed is captured during the order phase.

Main Functions

The Catalog Composition functional component:

- Shall manage inter-dependencies within the services.

Model

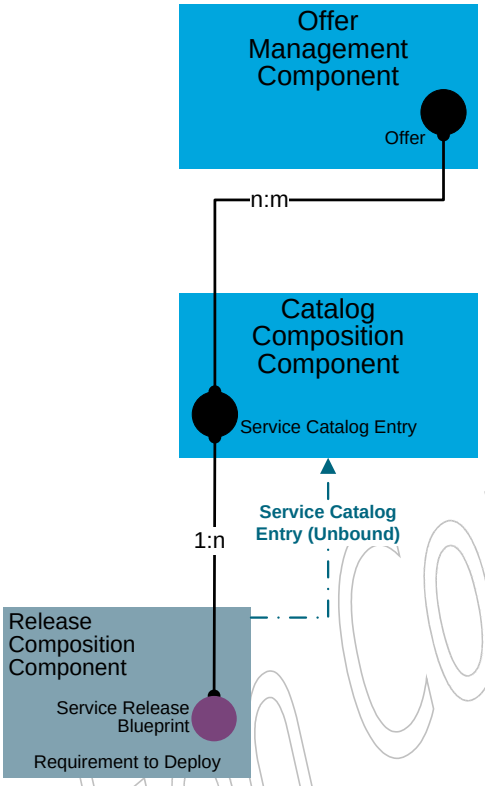


Figure 59: Catalog Composition Functional Component Level 2 Model

7.4.5 Request Rationalization Functional Component

Purpose

- Rationalize, break down, and route "clean order" requests (ready for fulfillment) to appropriate Fulfillment Execution engines or providers in order to deliver services to consumers. May involve breaking down a single order/request into multiple Fulfillment Requests.
- Ensure appropriate fulfillment-related Subscription information is kept up-to-date, such as approval/rejections, modifications, cancellations, and so on.
- Enable the recording of patterns of service consumption that can be used to shape demand for new and/or improved services.
- Break the request down into the IT services and provide these to the Fulfillment Execution functional component and create the Subscriptions for these services upon their successful fulfillment.
- Fulfillment status is tracked and completion notifications from fulfillment channel(s) are received. Consumers will be able to receive status updates at the Subscription level, not at the level of the underlying requests needed to fulfill the Subscription.

Key Data Objects

- **Request** (data object): Contains all Offers from the Shopping Cart which have been consumed and need to be fulfilled.
- **Subscription** (data object): Managed by the Request Rationalization functional component. This data object represents the rights to access a service that has been provided to a consumer.

Key Attributes

The Request data object shall have the following key data attributes:

- **RequestID**: Unique identifier for the Request.
- **UserID**: User identifier (from Shopping Cart).
- **Status**: Controls the status of the fulfillment.
- **RequestDate**: Date/time the Request was received.
- **MaxFulFillDate**: Maximum date/time on which the Request needs to be fulfilled.
- **ActFulFillDate**: Date/time on which the Request is fulfilled.
- **OfferID**: Identifier of the service (from Offer).
- **SubscriptionID**: Link to the Subscription for this service (from Subscription).
- **ServiceID**: Based on the Offer, the service(s) are identified (from Offer).
- **ReqValue**: Based on the Offer, the user might need to provide values/options for a successful fulfillment (from Offer).

The Subscription data object shall have the following key data attributes:

- **SubscriptionID**: Unique identifier for the Request.
- **UserID**: Unique identifier of the user (from Shopping Cart).

Key Data Object Relationships

The Request data object shall maintain the following relationships:

- **Request to Shopping Cart** (1:n): Enables the traceability of Requests to the originating order (in the form of the Shopping Cart) which is used to report the current status of the order.
- **Request to Subscription** (n:m): Enables traceability between the Request and the resulting Subscription. This helps to better understand how the current Subscription state was realized.
- **Request to Fulfillment Request** (1:n): Used for tracking fulfillment as well as to navigate between dependent Fulfillment Requests.

The Subscription data object shall maintain the following relationships:

- **Subscription to User Profile** (n:1): Enables the consumer to manage (view/modify) all of their Subscriptions. Also helps provide important information related to the specific consumer Subscription.
- **Subscription to Offer** (1:n): Provides traceability between the Subscription and the Service Contract (via the Offer).
- **Subscription to Chargeback Contract** (1:n): Facilitates the various chargeback/showback calculations that are dependent on Subscription details such as its contract duration and service status.
- **Subscription to Desired Service Model** (1:n): Enables traceability between the consumer, their Subscription, and the realized Service Model.

Main Functions

The Request Rationalization functional component:

- Shall provide information to the consumer on the fulfillment status.
- Shall provide Subscription information for the creation of the associated Chargeback Contract.
- Shall provide information on Request delivery times for SLA measurements.
- Shall break down the composite request (described by the Shopping Cart and consumer-selected values) into the individual Requests that need to be fulfilled.
- Shall send the bound Service Catalog Entry to the Fulfillment Execution functional component in order for it to create the Fulfillment Requests needed to satisfy the order.

Model

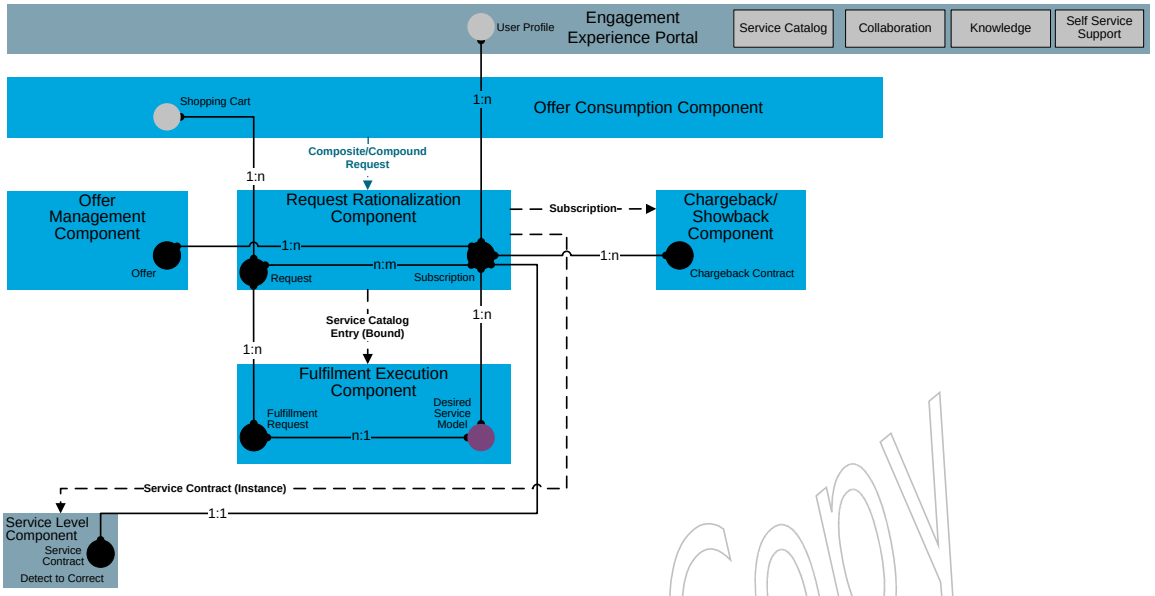


Figure 60: Request Rationalization Functional Component Level 2 Model

7.4.6 Fulfillment Execution Functional Component

Purpose

The Fulfillment Execution functional component orchestrates the delivery of the various requests amongst (one or more) fulfillment engines in order to deliver the IT service. Fulfillers may be systems that perform actions directly, or engage other systems in order to perform actions. They may also include external providers. In order to be able to engage the fulfillers, the Fulfillment Execution functional component needs to:

- Manage a registry of the available fulfillers. This registry captures:

  — What each fulfiller does (capabilities)

  — How to engage each fulfiller (where they are located and how to invoke them)

- Take the bound Service Catalog Entry and generate both the relevant Fulfillment Requests in order to realize/fulfill the originating consumer request and the Desired Service Model data object which represents the Service Model in its pre-configured or consumer configured state.

- Update the IT asset inventory as they are ordered.

- Request standard changes and update the Configuration Management functional component (if needed) on delivery of components.

- Maintain visibility into supplier capacity levels and raise alerts if capacity appears to be insufficient for immediate demand.

The Fulfillment Execution functional component can be used via two paradigms:

- **Consumer-driven** – In this paradigm a consumer request results in a bound Service Catalog Entry which is broken down into the necessary Fulfillment Requests needed to fulfill the originating request.
- **Direct access** (without a Service Catalog Entry) – In cases in which there aren't sufficient catalog entries to describe the fulfillment and no entries are planned to be created, the Fulfillment Execution functional component is directly engaged by the Release Composition functional component (R2D Value Stream). The Release Composition functional component needs to provide enough information in order for the Fulfillment Execution functional component to create the Fulfillment Request(s) necessary to perform the actions needed.

Key Data Objects

- **Fulfillment Request** (data object): Describes all fulfillment aspects of an IT service, which includes items such as provisioning, deploying, modifying, actions (i.e., start, stop, etc.), decommissioning, and so on.
- **Desired Service Model** (data object): This is an instantiation of the unbound Service Catalog Entry, which is the binding of the relevant parameters that determine how a service will be deployed/fulfilled. This results in a single realized deployment for the service. The parameters are set by the user's selections made in the Offer Consumption functional component, as well as the determinations made in the design of the service that are interpreted by the Fulfillment Execution functional component. This data object:
  - — Validates that the realized service matches the selections in the original request and notes exceptions or modifications where applicable
  - — Will comply with the definitions of the Service Catalog Entry

Note: The realized service (Actual Service CI) must also comply with the policies governing the IT environment.

Key Attributes

The Fulfillment Request data object shall have the following key data attributes:

- **FulfillmentID**: Unique identifier for the Request.
- **RequestID**: Refers to the originating Request (from Request).
- **DesiredServiceID**: Links to the service to be instantiated (from Desired Service Model).
- **RFCID**: If applicable, an RFC will be created (from RFC).
- **Status**: Status of the deployment/fulfillment workflow.

The Desired Service Model data object shall have the following key data attributes:

- **DesiredServiceID**: Unique identifier for the bound IT service to be delivered.

- **SubscriptionID**: Tracability to the Subscription (user of the IT service) (from Subscription).
- **ServiceReleaseBlueprintID**: Getting necessary information on the IT service (from Service Release Blueprint).
- **ActualServiceCI_ID**: Link to the Configuration Management functional component (from Actual Service CI).

Key Data Object Relationships

The Fulfillment Request data object shall maintain the following relationships:

- **Fulfillment Request to Request** (n:1): Inform on Fulfillment Request status.
- **Fulfillment Request to Service Release Blueprint** (n:1): The Service Release Blueprint supplies the Fulfillment Request with information needed to instantiate the service.
- **Fulfillment Request to Desired Service Model** (n:1): Acquire relevant information.
- **Fulfillment Request to RFC** (1:1): If applicable, the RFC created can be linked to the originating Request.

The Desired Service Model data object shall maintain the following relationships:

- **Desired Service Model to Subscription** (n:1): Create the traceability from service to Subscription.
- **Desired Service Model to Service Release Blueprint** (n:1): Acquire all necessary service information for fulfillment.
- **Desired Service Model to Actual Service CI** (1:1): Create traceability and enable verification of correct deployment/fulfillment.

Main Functions

The Fulfillment Execution functional component:

- Shall select the appropriate fulfillment mechanism.
- Shall coordinate if multiple fulfillment mechanisms are needed and manage the dependencies required to fulfill the IT service request.
- Shall provide the Subscription status to the Request Rationalization functional component.
- Shall create the Actual Service CIs within the Configuration Management functional component.
- May create an RFC associated with the service instantiation that is created within the Change Control functional component (D2C Value Stream). The RFC type, standard or normal, is determined within the Change Control functional component.

- Can create a new service monitor or modify an existing one for the service provided in the Request as part of fulfillment.
- Can create/route a Request to an external service provider to fulfill part or all of the service.
- Can request IT assets necessary for fulfillment (such as licenses). This also enables the tracking of assets being requested or procured and links them with the services that require them.
- Can trigger deployment engines to enable fulfillment of the service.

Model

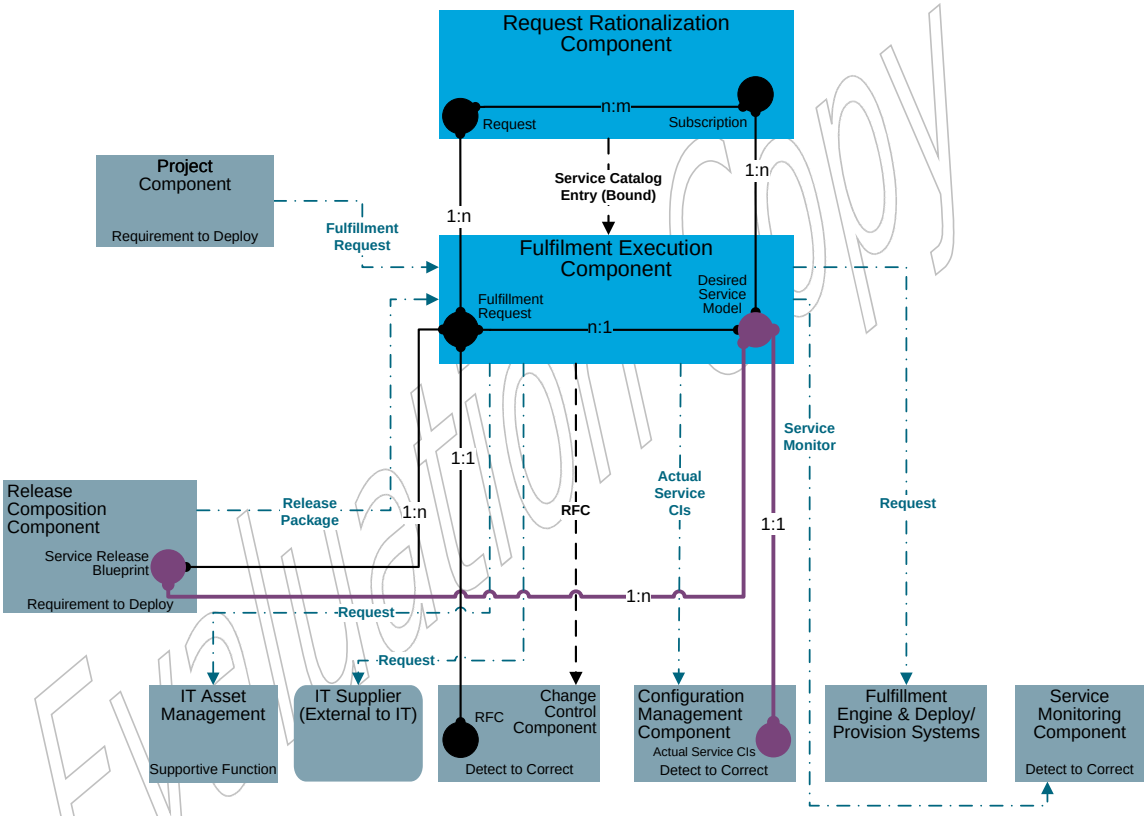


Figure 61: Fulfillment Execution Functional Component Level 2 Model

7.4.7 Usage Functional Component

Purpose

- Track actual usage of subscribed IT services by gathering IT service usage metrics, activity, and history for both internal and external sourced IT services.
- Process and break down usage information for each Subscription, its consumers (singular, group), provider, etc.

- Collect internally and externally-sourced IT service usage metrics from the Service Monitoring functional component (D2C Value Stream).

Key Data Objects

- **Usage** (data object): A data object within the R2F Value Stream managed by the Usage functional component. It is the measured use of a particular service or service component. An example usage can be composed of (internal) hours, system usage (capacity, CPUs, etc.), or external supplier usage.

Key Attributes

The Usage data object shall have the following key data attributes:

- **UsageID**: Unique identifier for the service Usage.
- **ChargebackContractID**: Relate the service Usage to the Chargeback Contract.

Key Data Object Relationships

The Usage data object shall maintain the following relationships:

- **Usage Record to Chargeback Contract** (n:1): Usage records are utilized to calculate chargeback amounts in cases in which the Chargeback Contract is dependent on Usage.

Main Functions

The Usage functional component:

- Shall encrypt sensitive usage information or set appropriate access controls.
- Can generate service usage history and activity reports.
- Can provide usage information to the Chargeback Contract component enabling usage-based showback or chargeback.

Model

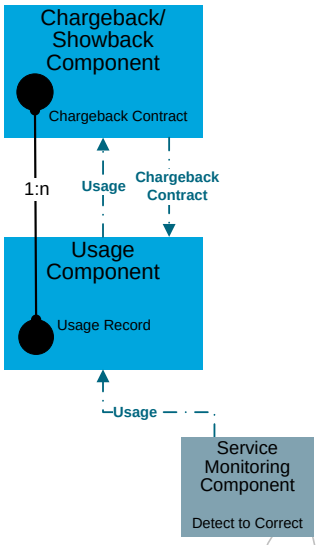


Figure 62: Usage Functional Component Level 2 Model

7.4.8 Chargeback/Showback Functional Component

Purpose

- Provide chargeback or showback for internal and external services taking into account service Subscription and Usage information.
- Break down chargeback or showback based on charge types such as CapEx/OpEx, direct/indirect, fixed/variable, and labor/non-labor, as well as by consumer or consumer groups.
- Charge for IT services rendered based on the pricing model associated with the catalog item, this may also be implemented via showback (no actual billing or cross-charging), which helps the consumer become more IT cost-aware.
- Trace chargeback or showback line items to individual cost drivers based on the cost model.
- Consolidate IT service Subscription (right to use) and actual Usage as the usage may differ from the right to use.

Key Data Objects

- **Chargeback Contract** (data object): Details the financial obligations between the service consumer and provider(s). The Chargeback Contract typically defines the unit of measure (price) for a given service and is often tightly linked to Usage.

Key Attributes

The Chargeback Contract data object shall have the following key data attributes:

- **ChargebackContractID**: Unique identifier for the Service Contract.
- **SubscriptionID**: Link to the Subscription for which this Service Contract is instantiated (from Subscription).

Key Data Object Relationships

The Chargeback Contract data object shall maintain the following relationships:

- **Chargeback Contract to Subscription** (n:1): This relationship provides the traceability between the service rendered (represented by the Subscription) and the expected charges for those services (described in the Chargeback Contract).

Main Functions

The Chargeback/Showback functional component:

- Shall provide the price of consuming/subscribing to a service.
- Can take actual usage into consideration when calculating the price of consuming a service.

If an IT Financial Management component exists, the Chargeback/Showback functional component:

- Can provide the necessary information in order for the IT Financial Management supporting function to produce invoices or bills for services rendered.

Model

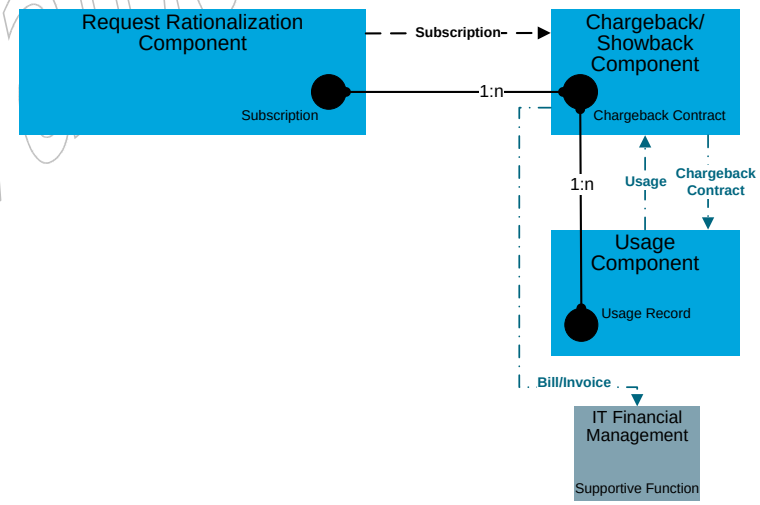


Figure 63: Chargeback/Showback Functional Component Level 2 Model

7.4.9 Knowledge & Collaboration Supporting Function

Purpose

Note: In the transition from Version 1.3 to 2.0 of the IT4IT Reference Architecture this functional component became a supporting component. However, it remains an important part of the R2F Value Stream in that it facilitates self-sufficiency across the consumer base. In future releases this section may be moved to later in the document but was kept here to facilitate readability.

- Provide knowledge in the form of content and Conversations that help to address the needs of IT service consumers. Knowledge includes structured IT/supplier produced articles, or unstructured Conversations from business/IT users, webinars, videos, training materials, etc. which are searchable by the IT service consumers.
- Increase the contribution to knowledge by providing all users with the ability to generate new content, either through informal Conversations, or by more formal submissions of knowledge.
- Encourage contributions by promoting a collaborative culture through various techniques such as gamification.
- Improve accessibility of knowledge in the organization by:
  - — Supporting key word search capabilities
  - — Providing filter capabilities based on various attributes of the knowledge, such as subject category, time range, source types (internal *versus* external), etc.
  - — Supporting natural language queries to reduce the complexity of finding relevant information
  - — Providing users with access to third-party knowledge and forums
  - — Providing natural language processing analytics so (for example) “trending topics” can be reported from service desk interactions
- Reduce the number of requests for information/knowledge that arrive at the IT service desk.
- Provide knowledge for consumption by additional value streams in general and specifically by the D2C Value Stream.
- IT staff participate in Conversations related to IT services that they plan, develop, or operate.
- IT service consumers and IT staff consume third-party knowledge through the same experience as the formal and informal forms of knowledge the company provides.

Key Data Objects

- **Knowledge** (data object): Structured and unstructured Knowledge from the Knowledge & Collaboration functional component.

- **Conversation** (data object): User Conversations from the Knowledge & Collaboration functional component.

Key Attributes

The Knowledge data object shall have the following key data attributes:

- **KnowledgeID**: Unique identifier of the Knowledge.
- **AuthorID**: Unique identifier of the responsible author.
- **RelatedService**: If provided, linked to the Actual Service CI in Configuration Management.
- **ProblemID**: If applicable, a link to a related Problem will be provided.
- **Status**: For example, draft, revise, published, review, archived.
- **PublishDate**: Date/time on which the item is available for publication.
- **ExpiryDate**: Date/time on which the item is no longer visible.
- **Title**: Title of Knowledge data object.
- **Body**: Body text, video, or any other content that is published.

The Conversation data object shall have the following key data attributes:

- **UserID**: Unique identifier of the responsible author.
- **KnowledgeID**: Related Knowledge data object(s).
- **Body**: Body text, video, or any other content.

Key Data Object Relationships

The Knowledge data object shall maintain the following relationships:

- **Knowledge to Problem** (n:m): If a link to a Problem exists, the Knowledge data object will need changing when a Problem is (partly) resolved.

The Conversation data object shall maintain the following relationships:

- **Knowledge to Conversation** (n:m): Conversations can be linked to the Knowledge data object(s). A Mnowledge manager is typically responsible for ensuring the content creates added value.

Main Functions

The Knowledge & Collaboration component:

- Shall enable SMEs to submit and/or approve Knowledge data objects (whether they are IT staff or service consumers).

- Shall provide functionality to enable the IT service consumers and IT staff to rank Knowledge data objects and Conversations, thus improving future knowledge consumption.
- Shall provide functionality to enable IT service consumers to participate in Conversations relating to the IT services they consume (collaboration).
- Can aggregate multiple (internal and external) Knowledge sources.

Model

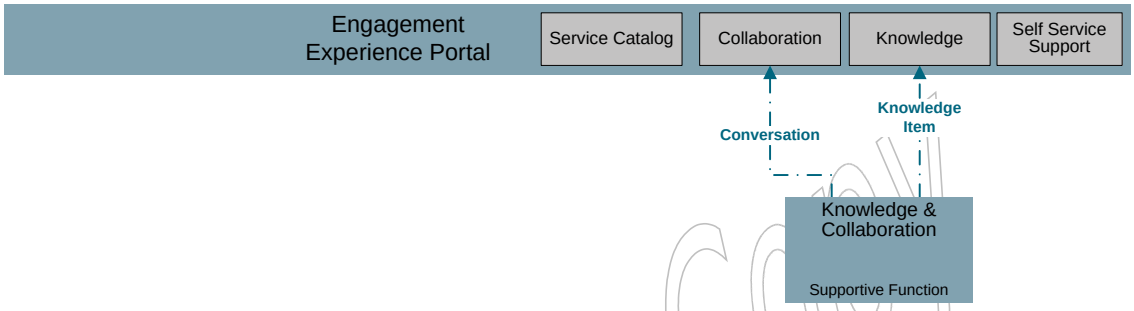


Figure 64: Knowledge & Collaboration Supporting Function Level 2 Model

8 Detect to Correct (D2C) Value Stream

This chapter provides an overview of the Detect to Correct (D2C) Value Stream – one of four IT Value Streams that comprise the IT Value Chain. It describes the business context, objectives, and details behind the D2C Value Stream.

8.1 Objectives

The D2C Value Stream provides a framework for the work of IT operations integrating Service Monitoring, Event, Incident, Problem, Change Control, Configuration Management, Service Remediation, and Service Level functions. It also provides a comprehensive overview of the business of IT operations and the services delivered by IT operations. This viewpoint provides understanding of the inter-relationships among its many domains; including Event, Incident, Problem, Change Control, and Configuration Management, and responsiveness to business requests and requirements. The D2C Value Stream is designed to incorporate a wide variety of sourcing methodologies across services, technologies, and processes. This value stream accommodates the technical inter-relationships and inter-dependencies required to fix operational issues and improve the ability of IT to support business objectives by providing agility, increased uptime, and lower per-service cost.

Today, all IT organizations have the ability to detect and resolve operational issues, but may have the following limitations:

- Timely identification of an issue before users of the service are impacted.
- Prioritization that understands business, IT operations, and technology impacts.
- Complexity of service delivery across multi-sourced environments exceeds the capabilities.
- Siloed organizational structure combines with process and technology misalignments.
- Slow troubleshooting from poor cross-domain teamwork due to poor data sharing, unreliable data, and inconsistent terminology.
- Slow remediation due to:
  - — Low levels of automation within domains
  - — Virtually no automation across domains
- Poor linkage of Incidents and Problems to the R2D Value Stream (providing Defect information about the service).
- Poor linkage of Problems to the S2P Value Stream (providing demand loop back information about the service).

- No end-to-end management of the service lifecycle across domains which causes more disconnects between the IT department's different groups, less focus on the business needs that are being served by the specific service, and a lack of coherent view of the service status at all times.

The D2C Value Stream provides a framework for bringing IT operations functions together to enhance IT services and efficiencies – thus reducing risk. Currently, data in each operation's domain is generally not shared with other domains because they do not understand which key data objects to share and do not have a common language for sharing. When projects are created to solve this, it is often too difficult and cumbersome to finish or there is an internal technology or organization shift that invalidates the result. The D2C Value Stream defines the functional components and the data objects that need to flow between components.

8.2 Business Value Proposition

The D2C Value Stream enables IT organizations to increase efficiency, reduce cost, reduce risk, and drive continuous service improvement by defining the data objects and data flow required to integrate the operations of multiple domains.

Increase efficiency and reduce cost by:

- Focusing response based on causal factor, priority, and business impact
- Increasing sharing of information and reduction of multiple entry of the same data
- Creating a prescriptive data flow between Event, Incident, Problem, and Change Control
- Centralized Event Management for faster analysis
- Automation between and across business functions
- Knowledge management and self-service linkage
- Driving Service Monitoring configuration, operating/Service Level targets, and predefined Knowledge linked to the R2F Value Stream
- Improving the speed at which issues with a business service are identified, including proactive identification before service impact is severe

Reduce risk by:

- Consistent data and configuration information shared between operational silos
- Prescriptive flow semantics and data objects
- Defined business impact
- Reducing the need for best-guess routing and clannish knowledge
- Increased uptime by reduced MTTR
- Creating a consistent way of managing SLM definitions, measurements, KPI calculations, and reporting back to the proper service owner or user (in the R2F Value Stream)

- Implementing network security to minimize intrusions that cause denial of service, viruses, and theft or corruption of data and minimize risk exposure
- Utilizing SIEM to identify complex attack signatures that can disrupt operations and affect compliance
- Performing threat and vulnerability assessments
- Audit trail
- Clear ownership

Continuous service improvement:

- Defined data objects to be shared with Problem Management
- Using this accumulated Knowledge as input into the S2P Value Stream
- Improved management information and decision-making

8.3 Key Performance Indicators

The D2C Value Stream critical success factors and Key Performance Indicators (KPIs) are as follows:

Critical Success Factors	Key Performance Indicators (KPIs)
Achieve Operational Excellence	Events: Increase in breadth and depth of monitoring endpoints, reduction of escalated events (via filtering/correlation/ automated resolution), reduction of false positives, and reduction of the number of security events that cause business disruption. Incidents: Incident reduction, reduction of escalated Incidents, reduction of false positives, reduction in the total number of security-related Incidents. Problems: Increase Problems identified, increase Problems eradicated. Changes: Reduction of change-related outages, reduction of emergency changes, reduction of unplanned changes, and reduction of security vulnerabilities introduced during Change Management. Knowledge: Increase Known Error availability (enrich Known Error database), increased usage.

Critical Success Factors	Key Performance Indicators (KPIs)
Improve Customer Satisfaction	OLA/SLA: Reduction of failed agreements. Availability of critical business systems: Increase uptime, decrease MTTR, increase MTBF. Performance (user experience) of critical business systems: Decrease user complaints. Incidents: Increased rate of first call resolution. Self-service: Increased success rate for user self-fix.
Improve Staff Effectiveness	Events: Increase automatically remediated Events, increase the percentage of Events correlated to a business service. Incidents: Reduction of re-opened Incidents, increase percentage of first call resolution, reduction in average time to close an Incident, increase automatically remediated Incidents, reduce average handling time, and reduce rejected Incidents. Changes: Increase automatically remediated changes.
Alignment with Business Strategy	Cost: Increase percentage of time invested on business-critical services. Services: Increase number of business services defined, decrease percentage of business-critical services, decrease number of CIs that are not linked to a business service, increase "quality of service" monitoring for internal and external business services. SLA/SLO: Increase percentage of business-critical services with defined Service Level targets. Security: Number of security-related outages to business-critical systems, number of security Incidents causing financial loss, business disruption, or public embarrassment, number of security Incidents resolved without business impact.

8.4 Value Stream Definition

The goal of the D2C Value Stream is to efficiently manage the IT operations by monitoring key services, correlating and appropriately escalating Events, managing (resolving) Incidents/Problems, running an effective Change Control, and doing all of that in an automated way. The D2C Value Stream is initiated when:

- A new or updated service is deployed resulting in the creation of a Physical Service Model, and Service Monitors which must now be managed and maintained.
- A Service Monitor generates an Event that must be handled (ignored, correlated, or escalated). This may be triggered when service discovery detects new CIs, which cause an existing Service Monitor to generate an Event.

- The service desk receives an Incident. In addition to escalated Events, Incidents can also be initiated by Self-Service Support requests from the R2F Value Stream or a user call to the service desk.

The above inputs trigger functional components into action that typically lead to the resolution and closure of the related Events, Incidents, or Problems.

The D2C Value Stream may interact with other value streams by creating:

- **Portfolio Backlog Item** (demand request) in the S2P Value Stream (Problems that can't be solved or are bigger than regular Defects).
- **Defects** in the R2D Value Stream (emergency fixes required to resolve the Incident or Problem).
- **Usage** in the R2F Value Stream (the Service Monitoring functional component provides usage for Subscription management, billing, chargeback, and showback).
- **Knowledge** data objects in the R2F Value Stream (problem resolution may capture knowledge of Known Errors).
- **Status and Service Level** status information for the R2F Value Stream (supplying monitoring status and SLA status to the Offer Consumption functional component).
- **Storing the Service Contract** (template) coming from the R2D and R2F Value Streams (as a base artifact for SLA management and creating the Service Contract instances) and also creating the **Service Contract** (instance) once the Subscription has been created in the R2F Value Stream.

As part of the D2C Value Stream, manual or automated Run Books may be executed to diagnose or remediate Events, Incidents, or Problems. When necessary, the Physical Service Model is updated and RFCs are created to record changes.

The D2C functional components are often owned and managed by different IT groups. The D2C Value Stream provides a framework that ensures that functional components used by IT groups can work together efficiently, through well-defined control points and data objects, to govern and run IT operations.

Figure 65 shows the functional components, data objects, and entity relationships for the D2C Value Stream.

Note: The red lines in Figure 65 represent recognized industry employed linkage variations. These variations and their associated implications will be detailed in future releases.

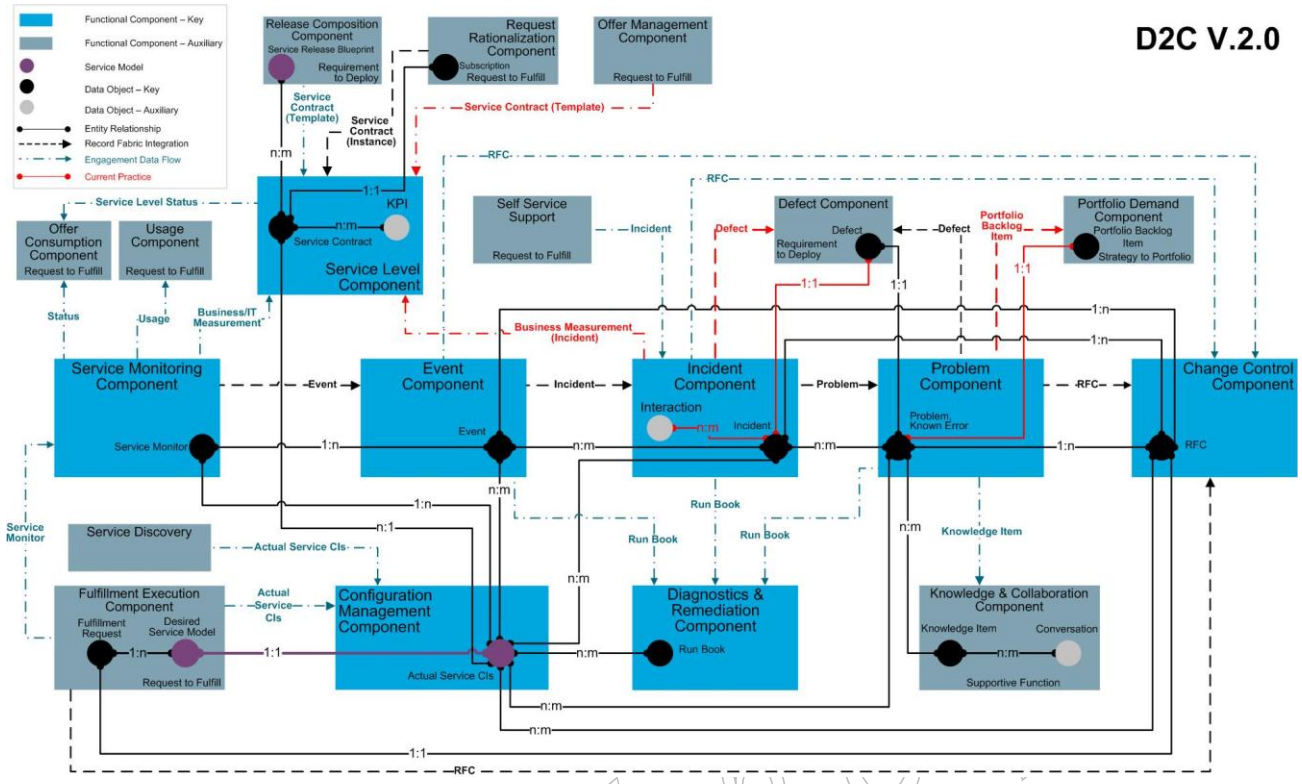


Figure 65: Detect to Correct Level 2 Value Stream Diagram

8.4.1 Service Monitoring Functional Component

Purpose

The Service Monitoring functional component is in charge of creating, running, and managing monitors which measure all aspects/layers of a service such as infrastructure (system and network), application, and security. It is also in charge of storing all measurement results and calculating compound measurements. The Service Monitoring functional component is not in charge of the monitor definitions. These are done earlier in the service lifecycle in the R2D Value Stream and are delivered to Service Monitoring by the Fulfillment Execution functional component in the R2F Value Stream.

Key Data Objects

- **Service Monitor** (data object): Performs the operational measurement aspects of a CI or an IT service in order to understand the current operational status of that CI or IT service.

Note: The Service Monitor definition created in the R2D Value Stream is moved with the Release Package as a part of the Service Release Blueprint transfer to the R2F Value Stream and is received from the Fulfillment Execution functional component (R2F Value Stream).

Key Attributes

The Service Monitor data object shall have the following key data attributes:

- **ServiceMonitorID**: Unique identifier for the Service Monitor.
- **Name**: Name of the Service Monitor.
- **Description**: Description of the Service Monitor.
- **Type**: Type of the Service Monitor (system, application, network, security, etc).
- **MeasurementDefinitions**: The definitions of the measurements that the Service Monitor is collecting about the monitored CI.
- **LastRunTime**: Date/time that the Service Monitor was last run.
- **LastRunStatus**: Was the last run of the Service Monitor successful or not?
- **ActualServiceCI_ID**: Identifier of related CI(s).

Key Data Object Relationships

The Service Monitor data object shall maintain the following relationships:

- **Service Monitor to Event** (1:n): Enables the traceability from the Events that are created to the Service Monitor they originated from.
- **Service Monitor to Actual Service CIs** (1:n): The Actual Service CI is the CI being monitored.

Event data flow integration between the Service Monitoring functional component and the Event functional component must manage the Event lifecycle state.

Main Functions

The Service Monitoring functional component:

- Shall be the system of record for all Service Monitors.
- Shall manage the lifecycle of the Service Monitor.
- Shall perform monitoring of all aspects of an IT service.
- Shall store all of the results of the measurements being done on the IT service.
- Shall calculate the results of compound Service Monitors from one or more simple measurements.
- Shall create an association between the Service Monitor data object and the related Actual Service CI(s).

If an Event functional component exists, the Service Monitoring functional component:

- Shall initiate the creation of an Event or alert that is passed to the Event functional component.

If an Offer Consumption functional component exists, the Service Monitoring functional component:

- Can provide service monitoring status.

If a Usage functional component exists, the Service Monitoring functional component:

- Can provide service usage measurements.

If a Service Level component exists, the Service Monitoring functional component:

- Can provide business/IT measurements.

If a Fulfillment Execution functional component exists, the Service Monitoring functional component:

- Can receive Service Monitor definitions.

Model

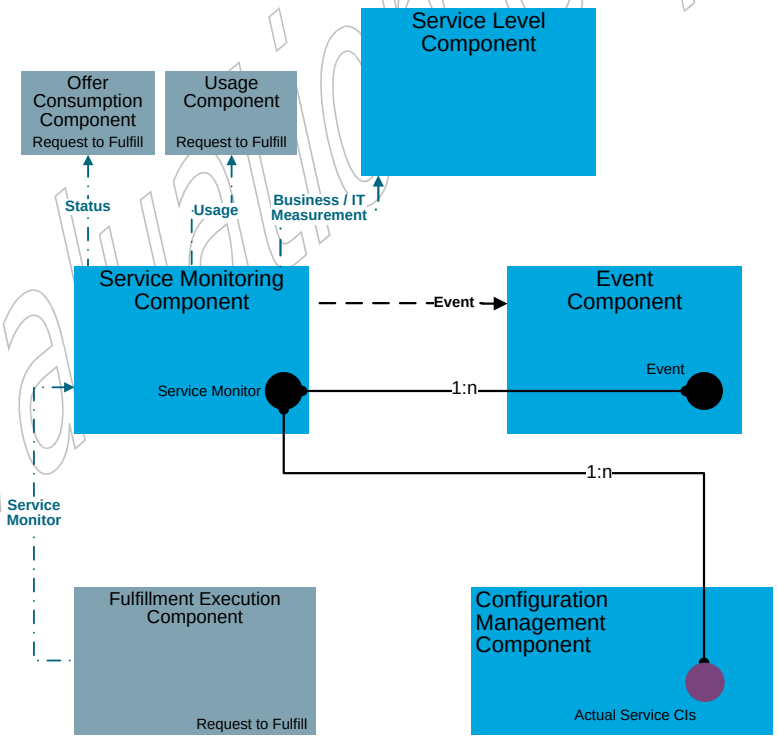


Figure 66: Service Monitoring Functional Component Level 2 Model

8.4.2 Event Functional Component

Purpose

The Event functional component manages Events through the Event lifecycle for events that occur on any IT service. The Event lifecycle includes but is not limited to detecting, categorizing, filtering, analyzing, correlating, logging, prioritizing, and closing events. During the event lifecycle some categories of events can serve as initiators of Incidents and for diagnostics and remediation activities.

Key Data Objects

- **Event** (data object): Represents an alert/notification signifying a change of state of a monitored CI.

Key Attributes

The Event data object shall have the following key data attributes:

- **EventID**: Unique identifier of an Event.
- **Name**: Name of an Event.
- **Category**: Category of an Event category (info, warning, error, etc.); aids in determining assignment and prioritization.
- **EventType**: Categorizing the Event to causal or symptom type.
- **EventStatus**: The current stage in the lifecycle of an Event.
- **EventStatusTime**: Time stamp for the EventStatus attribute.
- **Severity**: Severity of the Event.
- **ThresholdDefinitions**: The definitions of the thresholds by which the Event severity is determined.
- **AssignedTo**: Group or person that is assigned to handle the Event.
- **IsCorrelated**: Is the Event correlated to other Events?
- **ActualServiceCI_ID**: Related CI(s).

Key Data Object Relationships

The Event data object shall maintain the following relationships:

- **Event to Incident** (n:m): Enables the connection between the Incidents and Events and supports the integration lifecycle between them.
- **Event to RFC** (1:n): Associated Event is available for RFC processing.

- **Event to Actual Service CIs** (n:m): The Actual Service CI(s) associated with the Event(s).
- **Service Monitor to Event** (1:n): Enables the traceability from the Events that are created to the Service Monitor from which they originated.

Main Functions

The Event functional component:

- Shall be the system of record for all Events.
- Shall manage the state and lifecycle of the Events.
- Shall manage the correlation between Events.
- Shall categorize Event data.
- Shall forward Events categorized as Incidents to the Incident functional component.
- May initiate a change request (RFC) based on Event data to the Change Control functional component.
- Shall create an association between the Event data object and the related Actual Service CI(s).

If a Diagnostics & Remediation functional component exists, the Event functional component:

- May send Events for diagnostics and remediation processing.

Model

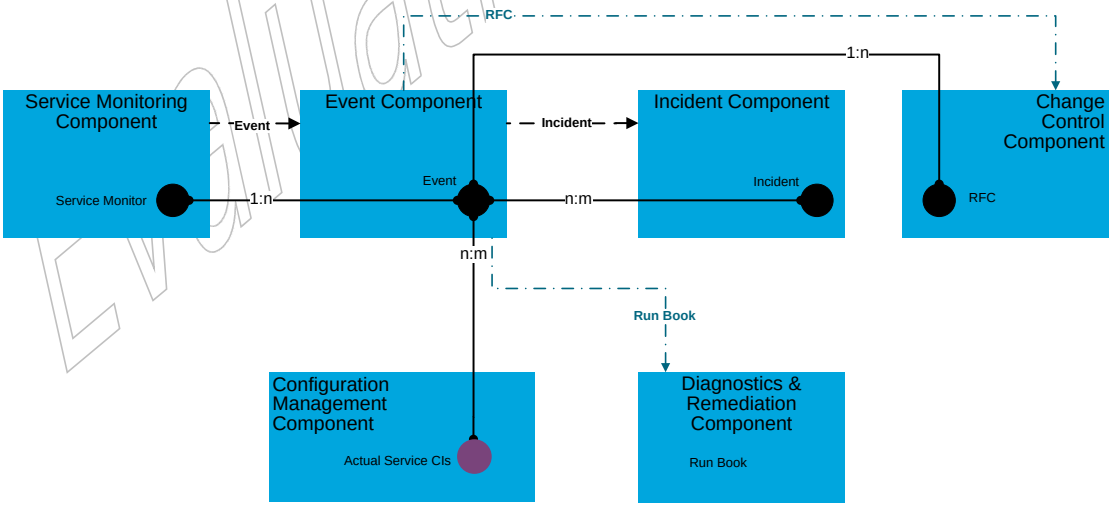


Figure 67: Event Functional Component Level 2 Model

8.4.3 Incident Functional Component

Purpose

The Incident functional component facilitates normal service operations restoration as quickly as possible and minimizes the impact on business operations, thus optimizing service quality and availability. Service restoration can be facilitated through the following means:

- In partnership with the Service Monitoring functional component, filter end-user Interactions and determine which ones should be associated with Incidents.
- Detect Incidents, investigate the impacts across all domains (server, network, security, etc.), and determine the correct action to take.
- Initiate change and/or remediation activity for some categories of Incidents.

An Incident is defined as an unplanned interruption to an IT service or reduction in the quality of an IT service as defined within the Service Contract related to the IT service. Failure of a CI that has not yet affected a service is also an Incident; for example, failure of one disk from a mirror set.

An Interaction is a record of any end-user contact with the service desk agent. In some cases, either the service desk agent or self-service knowledge can resolve the Interaction without creating and managing an Incident. In other cases, an Interaction can be associated with an existing Incident (eliminating the need for Incident creation and correlation). When an Interaction cannot be associated with an existing Incident because it requires additional clarification, diagnostics, or support actions, a new Incident will be created.

Key Data Objects

- **Incident** (data object): Hosts and manages Incident data.
- **Interaction** (data object): Hosts the record of an end-user's contact with the service desk.

Key Attributes

The Incident data object shall have the following key data attributes:

- **IncidentID**: Unique identifier for the Incident record.
- **Category**: The category aids in determining assignment and prioritization.
- **SubCategory**: The second level of categorization, following Category.
- **IncidentStatus**: The current stage in the lifecycle of an Incident.
- **IncidentStatusTime**: Time stamp for the IncidentStatus attribute.
- **Severity**: Severity of the Incident.
- **Priority**: Priority of fixing the Incident.

- **Title**: Title of the Incident.
- **Description**: Description of the Incident.
- **AssignedTo**: Group or person that is assigned to fix the Incident.
- **ActualServiceCI_ID**: Related CI(s).

The Interaction data object shall have the following key data attributes:

- **IteractionID**: Unique identifier for the Interaction record.

Key Data Object Relationships

The Incident data object shall maintain the following relationships:

- **Incident to Problem, Known Error** (n:m): Connection between the Incidents that are converted to Problems to permanently address severe/repeating Incidents.
- **Incident to RFC** (1:n): Connecting RFCs to the Incidents they originated from.
- **Incident to Defect** (1:1): Current practice: connecting Defects that are created when Incident diagnostics determines there is need for an emergency fix from development.
- **Incident to Actual Service CIs** (n:m): CI to which the Incident is associated and usually the main subject of.
- **Event to Incident** (n:m): Enables the connection between the Incidents and Events and supporting the integration lifecycle between them.

Incident data flow between the Event functional component and Incident functional component must manage the Incident lifecycle state.

Problem data flow between the Incident functional component and the Problem functional component must manage the Problem lifecycle state.

Main Functions

The Incident functional component:

- Shall be the system of record for all Incidents.
- Shall manage the state escalation paths and general lifecycle of the Incident.
- Shall allow an Incident to be initiated from an Event.
- Shall create an Incident when an Interaction cannot be associated with an existing Incident because it requires additional clarification, diagnostics, or support actions.
- Shall create an association between the Incident data object and the related Actual Service CI(s).

If a Defect functional component exists, the Incident functional component:

- May initiate the creation of a Defect when Incident diagnostics determines that an emergency fix is required from development for resolution. The Defect is created and forwarded to the Defect functional component in the R2D Value Stream.

If a Diagnostics & Remediation functional component exists, the Incident functional component:

- Can trigger the execution of a Run Book (either automated or manual) to provide diagnostic information or remediation steps.

If a Problem functional component exists, the Incident functional component:

- May create a Problem record when the Incident is severe, requires further deep investigation, or is repeating.

If a Change Control functional component exists, the Incident functional component:

- Can trigger the creation of an RFC in order to implement a fix to the Incident fault.

If a Self-Service Support functional component (R2F Value Stream) exists, the Incident functional component:

- Can allow the initiation of an Interaction or an Incident.

If a Service Level functional component exists, the Incident functional component:

- Can provide business measurements of Incident data.

Model

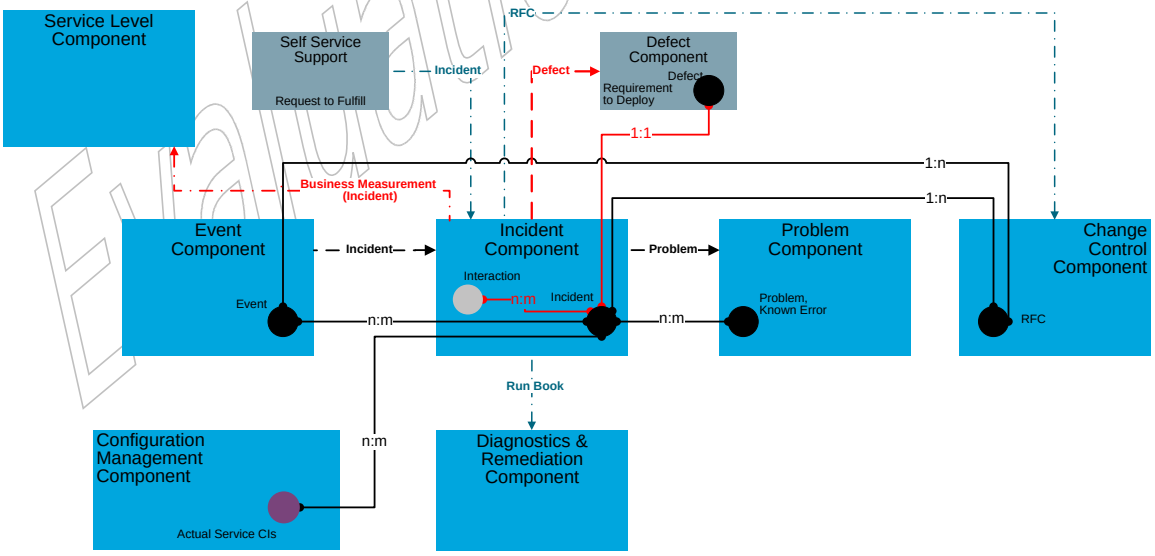


Figure 68: Incident Functional Component Level 2 Model

8.4.4 Problem Functional Component

Purpose

The Problem functional component is responsible for managing the lifecycle of all Problems. The objectives of the Problem functional component are to solve severe/repeating Incidents, prevent Incidents from happening, and minimize the impact of Incidents that cannot be prevented. The Problem cause is not usually known at the time of the Problem data object instance creation, and the Problem functional component is responsible for the investigation. The Problem functional component also serves as the main exit point from the D2C Value Stream for the feedback information about IT services issues. The feedback is reported to the R2D Value Stream in the form of Defects and to the S2P Value Stream in the form of Portfolio Backlog Items (demand request).

Key Data Objects

- **Problem, Known Error** (data object): Defines the Problem or Known Error and manages the Problem and Known Error lifecycle.

Key Attributes

The Problem data object shall have the following key data attributes:

- **ProblemID**: Unique identifier for every Problem record.
- **Category**: The Category aids in determining assignment and prioritization.
- **SubCategory**: The second level of categorization, following the Category attribute.
- **ProblemStatus**: The current stage in the lifecycle of a Problem.
- **ProblemStatusTime**: Time stamp for the ProblemStatus attribute.
- **Title**: Title of the Problem.
- **Description**: Description of the Problem.
- **AssignedTo**: Group or person that is assigned to fix the Problem.
- **ActualServiceCI_ID**: Related CI(s).

Key Data Object Relationships

The Problem data object shall maintain the following relationships:

- **Problem, Known Error to RFC** (1:n): This connection enables the relation of an RFC record that is created when problem resolution requires a change.
- **Problem, Known Error to Portfolio Backlog Item** (1:1): Ensures a Portfolio Backlog Item is created for Problems requiring a future fundamental/big fix/enhancement to the IT service.

- **Problem, Known Error to Defect** (1:1): Enables the creation of Defects when emergency/specific fixes require development.
- **Incident to Problem, Known Error** (n:m): Connection between the Incidents that are converted to Problems to permanently address severe/repeating Incidents.
- **Problem, Known Error to Actual Service CI** (n:m): Problem records are mapped to the affected CI(s).
- **Problem, Known Error to Knowledge** (n:m): Creating a relationship between the Knowledge data object and the Problem it originated from.

Defect data between the Problem functional component and the Defect functional component must manage the Problem to Defect lifecycle state.

RFC data between the Problem functional component and the Change Control functional component must manage the RFC lifecycle state.

Main Functions

The Problem functional component:

- Shall be the system of record for all Problem records.
- Shall manage the state and lifecycle of the Problem.
- Shall associate Problem(s) to CI(s).
- Shall create Known Error data object instances from unsolved Problems.

If a Diagnostics & Remediation functional component exists, the Problem functional component:

- Can push Problem data to trigger the execution of a Run Book data object (either automated or manual) to provide diagnostics information or remediation steps.

If a Change Control functional component exists, the Problem functional component:

- Shall create an RFC associated to a Problem in order to implement a fix to the issue that is documented by the Problem.

If a Knowledge & Collaboration functional component exists, the Problem functional component:

- Shall use existing Knowledge data to solve a Problem.
- Can create a new Knowledge data object based on Problem Management activities (from the Known Error data object instances).

If an Incident functional component exists, the Problem functional component:

- Shall associate Incident data to the corresponding Problem record and continue the investigation around the Incident reported fault within the Problem lifecycle.

If a Defect functional component exists, the Problem functional component:

- Shall push Problem data requiring emergency/specific development to the Defect functional component (R2D Value Stream).

If a Portfolio Demand functional component exists, the Problem functional component:

- May push a Portfolio Backlog Item to the Portfolio Demand functional component for backlog processing.

Model

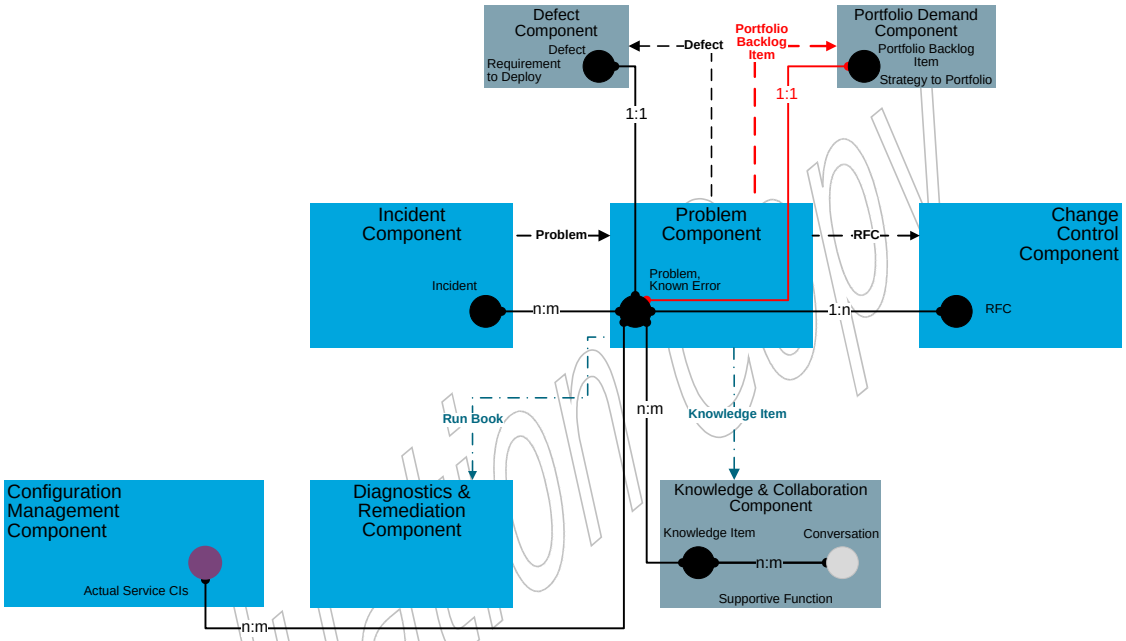


Figure 69: Problem Functional Component Level 2 Model

8.4.5 Change Control Functional Component

Purpose

The Change Control functional component is the system that is responsible for managing the lifecycle of all the RFCs in the IT environment. It makes sure that changes are done in a standardized and auditable way so that the business risk is minimized. It manages change through the following means:

- Facilitate communication with stakeholders.
- Assess risk of proposed changes and their implementation. Support the impact and risk assessments to minimize risk of production disruptions involved when rolling out changes
- Enable management of organizational changes and training needed for making a new release a success (which may include oversight that ensures that these tasks are included as part of the deployment package).

- Enable notification of all affected change stakeholders and facilitate their collaboration on change execution.
- Support automation of changes so that human participation is reserved for the highest added value and most complex change work. For example, the Event functional component or Incident functional component may use a manual or automated Run Book to resolve well understood issues without an active RFC. These classes of pre-approved RFCs will vary by company and by criticality of service. For these pre-approved RFCs, it is assumed that the RFC is recorded and that the Change Management process has access to that information. A typical example would be a Run Book automation script that both fixes the issue and logs what was changed as a closed RFC record. The CI relationship is maintained to allow Configuration Management to have access to change information.
- Enable RFC management against a change calendar and avoid change collisions.
- Check and steer conflict resolutions between parallel planned deployments.

Key Data Objects

- **RFC** (data object): Change data to manage the change lifecycle. An RFC includes details of the proposed change and may be recorded on paper or electronically.

Key Attributes

The RFC data object shall have the following key data attributes:

- **RFCID**: Unique identifier for the change record.
- **Category**: The category aids in determining assignment and prioritization.
- **SubCategory**: The second level of categorization, following Category.
- **ChangePhase**: The current step in the RFC workflow.
- **ChangePhaseTime**: Time stamp for the change phase.
- **ApprovalStatus**: The current status of the RFC approval.
- **ChangeRisk**: The probability that the change could cause harm or loss, or affect the ability to achieve objectives.
- **PlannedStartTime**: Date/time that RFC implementation is planned to start.
- **PlannedEndTime**: Date/time that RFC implementation is planned to end.
- **Title**: Title of the Incident.
- **Description**: Description of the Incident.
- **AssignedTo**: Group or person that is assigned to fix the Problem.
- **ActualServiceCI_ID**: Related CI(s).

Key Data Object Relationships

The RFC data object shall maintain the following relationships:

- **Fulfillment Request to RFC** (1:1): An RFC initiated from the Fulfillment Execution functional component (R2F Value Stream) is created on service implementation/instantiation.
- **RFC to Actual Service CIs** (n:m): Associates the RFC with affected CI(s).
- **Problem, Known Error to RFC** (1:n): Enables the relation of an RFC record that is created when problem resolution requires a change.
- **Incident to RFC** (1:n): Connecting RFCs to the Incidents they originated from.
- **RFC to Event** (1:n): Associated Event is available for RFC processing.

RFC data between the Fulfillment Execution functional component and Change Control functional component must manage the RFC lifecycle state.

RFC data between the Problem functional component and the Change Control functional component must manage the RFC lifecycle state.

Main Functions

The Change Control functional component:

- Shall act as an authoritative system of record for all change request information.
- Shall manage the state and lifecycle of the change.
- Shall associate change(s) to CI(s).
- Can provide change data to the Event and/or Service Monitoring functional components to facilitate root-cause analysis process in the context of the impact changes may have.

If an Incident functional component exists, the Change Control functional component:

- Shall associate changes with Incidents bi-directionally (changes in response to Incidents, and changes that actually cause Incidents; i.e., unsuccessful changes).

If an Event functional component exists, the Change Control functional component:

- May associate changes with Events when a change triggers an Event or an Event occurs during a change period.

If a Fulfillment Execution functional component exists, the Change Control functional component:

- Shall associate the Fulfillment Request with the RFC record as the overall framework that facilitates the IT service implementation/instantiation.

If a Problem functional component exists, the Change Control functional component:

- Shall associate RFCs to the Problem in order to implement a fix to the issue that is documented by the Problem.

Model

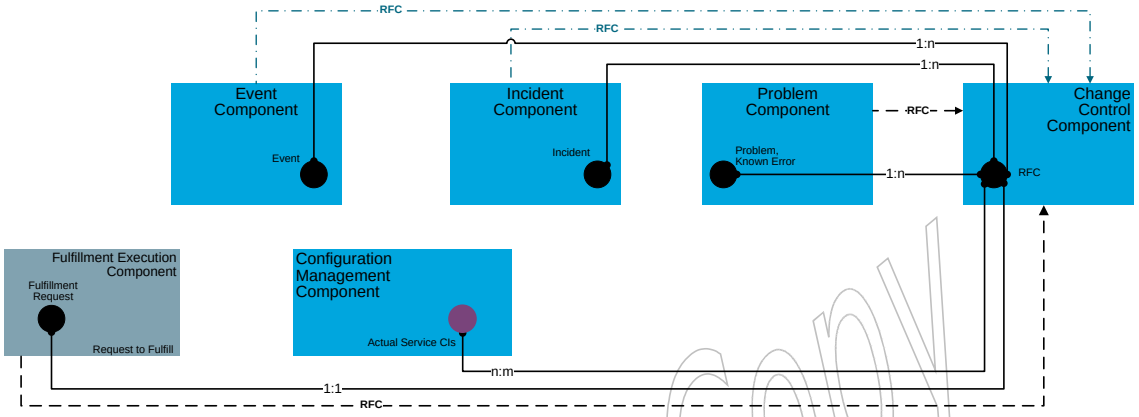


Figure 70: Change Control Functional Component Level 2 Model

8.4.6 Configuration Management Functional Component

Purpose

The Configuration Management functional component is focused on tracking the inventories of Actual Service CIs and their associated relationships. It identifies, controls, records, reports, audits, and verifies service CIs; including versions, constituent components, their attributes, and relationships.

Key Data Objects

- **Actual Service CI** (data object): Serves as the data store for the realization of the service in the production environment. CIs may be populated by service discovery, created by manual processes, or sourced from other processes such as IT Asset Management. A CI is defined as any component that may need to be managed in order to deliver an IT service. Typical CIs include but are not limited to: application services, infrastructure services, databases, message queues, batch jobs, logical transactions, servers (virtual and physical), network devices, storage devices, racks, power distribution units, laptops, software packages, and components.

Key Attributes

The Actual Service CI data object shall have the following key data attributes:

- **ActualServiceCI_ID**: Unique identifier of the CI.
- **Name**: Name of the CI.

- **Type**: Type of the CI (e.g., service, computer, network card, database, etc.).
- **CreateTime**: Date/time the CI was created.
- **LastModifiedTime**: Date/time the entry was last modified in a significant way.
- **Owner**: Group or person that is assigned to own the CI.
- **Location**: Location of the CI. Can vary from high-level country to city or low-level, such as building or room.

Key Data Object Relationships

The Actual Service CI data object shall maintain the following relationships:

- **RFC to Actual Service CIs** (n:m): The Actual Service CIs represent the CI(s) associated with the change activity.
- **Problem, Known Error to Actual Service CI** (n:m): Problem records are mapped to the affected CIs.
- **Run Book to Actual Service CI** (n:m): Run Book records are mapped to the associated CIs.
- **Incident to Actual Service CIs** (n:m): CI(s) to which the Incident is associated.
- **Actual Service CIs to Event** (n:m): The Actual Service CI(s) associated with the Event.
- **Actual Service CIs to Service Contract** (n:1): Connecting between the CIs and the Service Contract they are measured in.
- **Service Monitor to Actual Service CIs** (1:n): The Actual Service CI represents the CI being monitored.

Main Functions

The Configuration Management functional component:

- Shall be the system of record for all Actual Service CIs and their associated relationships.
- Shall manage the lifecycle of the CI.
- Shall allow hierarchical relationships between CIs.
- Shall serve as the data store for the realization of the service in the production environment.
- Can be populated by service discovery.

If a Diagnostics & Remediation functional component exists, the Configuration Management functional component:

- Shall associate a Run Book with the CI against which the Run Book is associated.

If a Change Control functional component exists, the Configuration Management functional component:

- Shall associate a CI with an RFC with which the change is associated.
- Shall calculate and provide the change impact based on the proposed change and the CI relationships.

If a Problem functional component exists, the Configuration Management functional component:

- Shall associate the CI with the Problem record against which the Problem is associated.

If an Incident functional component exists, the Configuration Management functional component:

- Shall associate the CI with the Incident with which the Incident is associated.
- Shall calculate and provide the business impact of the Incident to help in the prioritization process.

If an Event functional component exists, the Configuration Management functional component:

- Shall associate the CI with the Event with which the change is associated.
- Shall calculate and provide the business impact of the Event to help in the prioritization process.

If a Service Monitoring functional component exists, the Configuration Management functional component:

- Shall associate the CI with the Service Monitor with which the change is associated.

Model

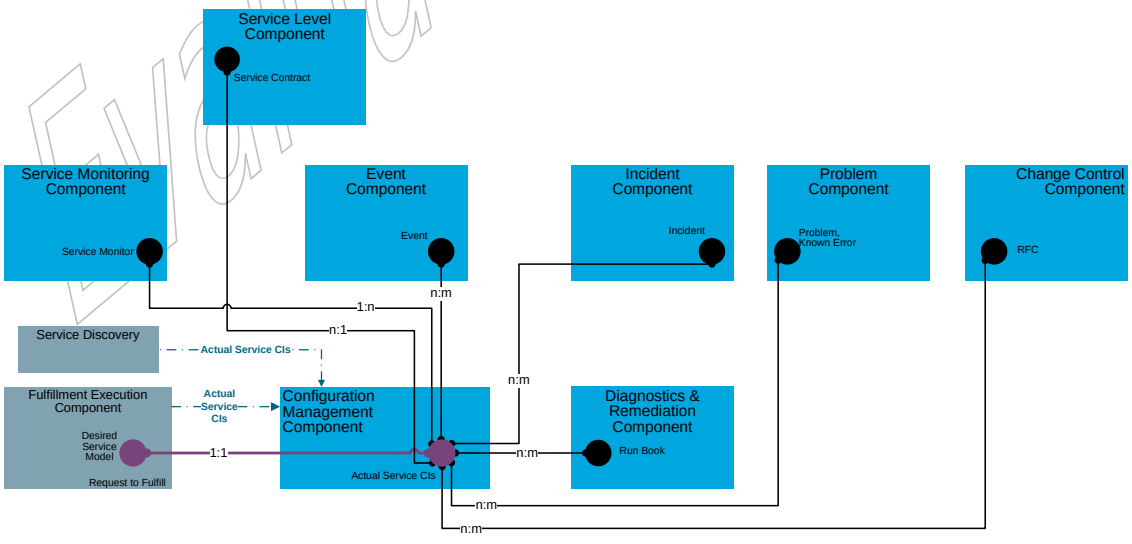


Figure 71: Configuration Management Functional Component Level 2 Model

8.4.7 Diagnostics & Remediation Functional Component

Purpose

Through the use of manual and automated Run Books, the Diagnostics & Remediation functional component provides diagnostics information and/or remediation steps to shorten the MTTR. Run Books help streamline diagnosis and remediation for service functions by applying knowledge solutions to service anomalies.

Key Data Objects

- **Run Book** (data object): A routine compilation of the procedures and operations which the administrator or operator of the system carries out. A Run Book can be either a manual process or an automated script.

Key Attributes

The Run Book data object shall have the following key data attributes:

- **RunbookID**: Unique identifier for the Run Book record.
- **Description**: Description of the Run Book.
- **Category**: The Category aids in determining assignment and prioritization.
- **ExecutionTime**: Date/time the Run Book was created.
- **ActualServiceCI_ID**: Related CI(s).

Key Data Object Relationships

The Run Book data object shall maintain the following relationships:

- **CI to Run Book** (n:m):Track Run Books and the CI with which they are associated.

Main Functions

The Diagnostics & Remediation functional component:

- Shall be the system of record for all Run Books.
- Shall manage the Run Book lifecycle.
- Shall allow hierarchical relationships between Run Books.
- Shall associate a Run Book with a CI.

If an Event functional component exists, the Diagnostics & Remediation functional component:

- Can allow an Event to trigger a Run Book mainly for diagnostics purposes.

If an Incident functional component exists, the Diagnostics & Remediation functional component:

- Can allow an Incident to trigger a Run Book for diagnostics or remediation purposes (remediation that does not require RFCs).

If a Problem functional component exists, the Diagnostics & Remediation functional component:

- Can allow a Problem to trigger a Run Book for remediation purposes (after an RFC has been opened).

Model

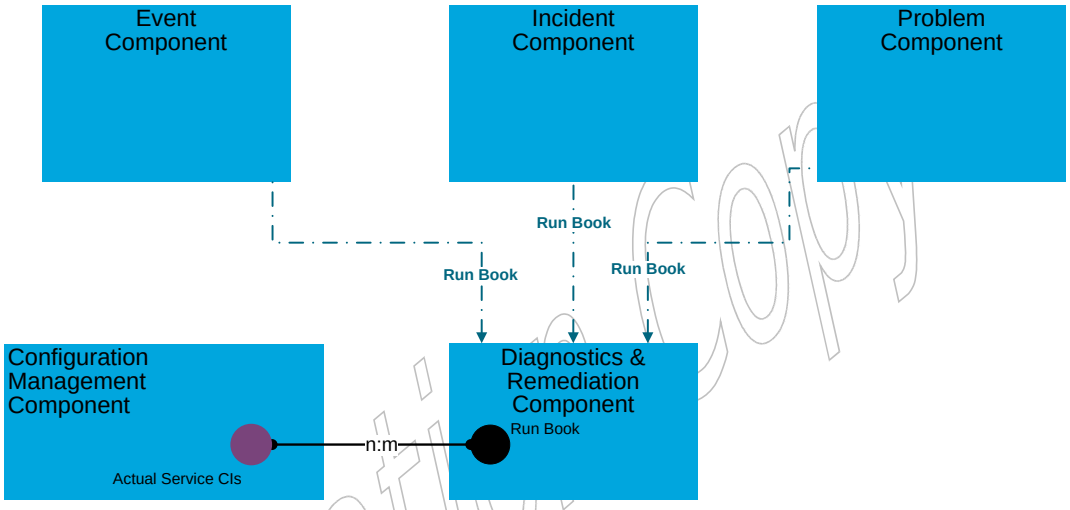


Figure 72: Diagnostics & Remediation Functional Component Level 2 Model

8.4.8 Service Level Functional Component

Purpose

The Service Level functional component enables the design and creation of Service Contracts (SLAs). It is also responsible for the management of all Service Contract data objects throughout their lifecycle including the governance of the Service Contract instances from the moment they are instantiated. This functional component is also responsible for collecting the relevant information in order to monitor compliance with the terms specified in the Service Contract and exposing data that reflects that actual performance against the defined Service-Level Objectives (SLOs).

The actual legal aspects of the Service Contracts are not handled by the Service Level functional component directly; however, these documents (usually created and managed by the legal department and not in IT) are used by the functional components in the S2P and R2D Value Streams as the main input for the demand and requirements definition stages.

Key Data Objects

- **Service Contract** (data object): Describes the service characteristics and supports service measurement tracking, governance, and audit. Service Contracts can be related to logical services as well as physical services. Service Contracts related to logical services are known as Service Contract templates, while Service Contracts related to physical services are known as Service Contract instances. Each Service Contract data object is comprised of two main parts: the General Contract definitions (*aka* the header) and the SLOs (the line items), which also enable nesting other Service Contracts that define Service Levels for different aspects of the service. These lines may need to be detailed due to the service being composed of multiple services, because there are multiple providers involved, or to cover different areas of Service Levels.
- **Key Performance Indicator** (data object): The definition of an objective that is measured, its requested thresholds, and the exact mathematical method in which measurement data items are used in order to calculate the KPI measurements.

Key Attributes

The Service Contract data object shall have the following key data attributes:

- **ServiceContractID**: Unique identifier of the Service Contract.
- **Name**: Name of the Service Contract.
- **Type**: Type of the Service Contract (SLA, OLA, UC).
- **Provider**: Provider of the service.
- **Consumer**: Consumer of the service.
- **StartDate**: Start date/time of the Service Contract.
- **EndDate**: End date/time of the Service Contract.
- **SupportCalendar**: Contracted support hours of the service.
- **AdherenceCalculationPeriodicity**: Service measurement calculation period.
- **MaintenanceWindow**: Service maintenance timeframes/blackout periods.
- **ActualServiceCI_ID**: Related CI(s).

The Key Performance Indicator data object shall have the following key data attributes:

- **KeyPerformanceIndicatorName**: The name of the Key Performance Indicator.
- **KeyPerformanceIndicatorDescription**: A short description of the Key Performance Indicator.
- **KeyPerformanceIndicatorThreshold**: The threshold of the Key Performance Indicator.

Key Data Object Relationships

The Service Contract data object shall maintain the following relationships:

- **Service Release Blueprint to Service Contract** (n:m): The Service Release Blueprint serves as the place where the Service Contract templates are being stored after their development and before they are shipped to the Service Level functional component.
- **Actual Service CIs to Service Contract** (1:n): CI relationships are maintained and updated to ensure functional component and data object traceability in the value stream.
- **Service Contract to KPI** (n:m): KPIs will track the measurements associated with Service Contracts. Service Contracts will have multiple KPIs.
- **Subscription to Service Contract** (1:1): Once a Subscription is instantiated it also triggers the instantiation of a Service Contract instance from the Service Contract template.

Service Contract (instance) data flow between the Request Rationalization functional component and the Service Level functional component must manage the Service Contract (instance) lifecycle state.

Main Functions

The Service Level functional component:

- Shall be the system of record for the Service Contract.
- Shall manage the Service Contract lifecycle (create, store, and maintain).
- Shall manage the lifecycle (create, store, and maintain) of KPIs.
- Shall manage the state of the Service Contract.
- Shall allow hierarchical relationships between Service Contracts.
- Shall manage the relations between the Service Contract data object and the KPI data object throughout their lifecycle.
- Shall receive measurements such as Incident data as well as other information that may be covered by the Service Contract and used for calculating the KPI measurements.
- Shall create reports on the Service Contracts to show the quality of service per SLO.

If a Service Monitoring functional component exists, the Service Level functional component:

- Can receive business/IT measurements from Service Monitoring.

If a Release Composition functional component exists, the Service Level functional component:

- Can instantiate a Service Contract from a Service Release Blueprint using the Service Contract (template).

If an Offer Management functional component exists, the Service Level functional component:

- May instantiate a Service Contract from a Service Contract (template) originating from the Offer Management functional component (R2F Value Stream).

If a Request Rationalization functional component exists, the Service Level functional component:

- Shall create a Service Contract (instance) and start measuring it once a Subscription is instantiated.

If an Incident functional component exists, the Service Level functional component:

- May receive Incident business measurements from the Incident functional component.

If an Offer Consumption functional component exists, the Service Level functional component:

- Can send reporting data on the Service Level status.

Model

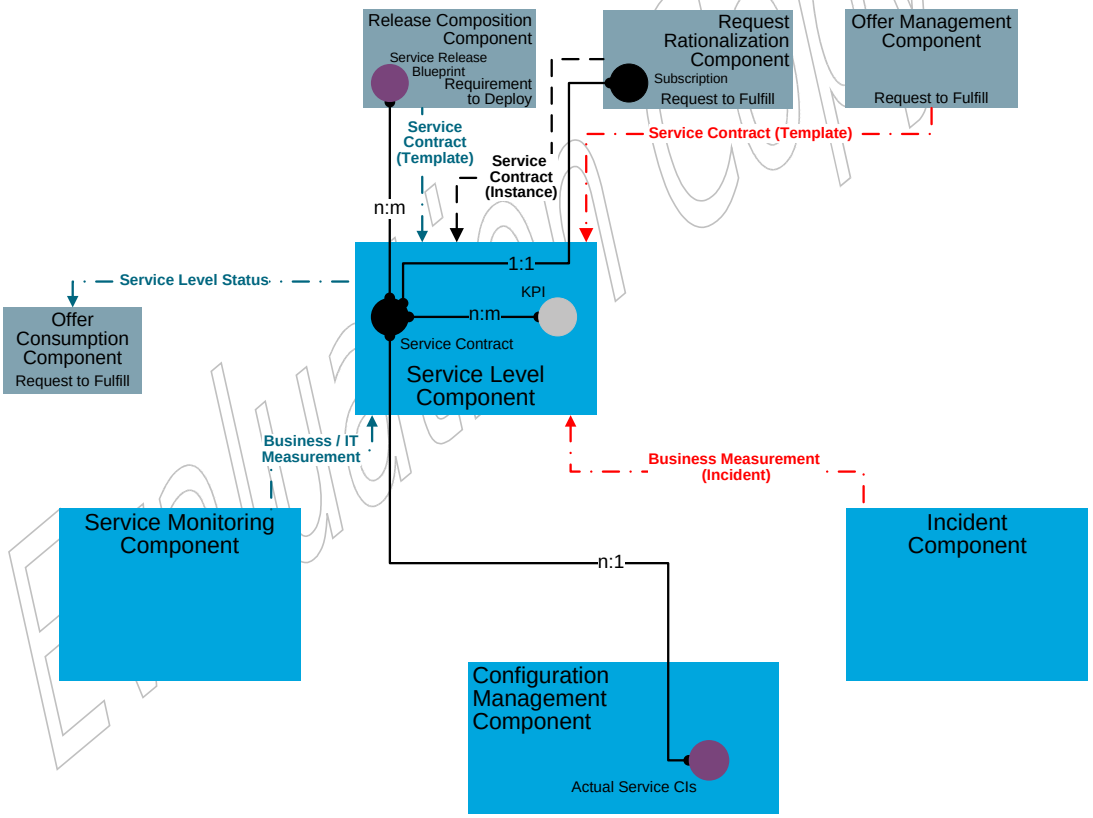


Figure 73: Service Level Functional Component Level 2 Model

8.4.9 Other IT Operations Areas

IT operations include a broad area of capability and many of these capabilities are not part of the D2C Value Stream.

- Capacity planning is currently not included in the D2C Value Stream; however, there is a definite relationship with the D2C Value Stream. Its impact and relationship will be reviewed in future releases.
- Availability Management, although not represented in the D2C Value Stream, is recognized as adding value to the D2C Value Stream. It will be reviewed in future releases.
- Intelligence, trending, proactive alerting (for example) are capabilities within the D2C Value Stream functional components Service Monitoring, Event, and Incident that may be sources of Events or Incidents.

A Rationale (Informative)

This informative appendix contains additional information concerning the contents of this standard.

The sections below parallel the chapters of the standard.

A.1 Introduction

The approach put forward in this standard is based on the long-standing thought experiment of "running IT as a business", a common theme in IT management discussions for the past 40 years (see Betz 2011, p.10 for extensive citations). It is based in the mainstream of IT management guidance including frameworks such as ITIL, COBIT, and PMBOK, standards such as ISO/IEC 20000, and other IT management professional literature. Drawing on this base, the IT4IT Reference Architecture provides practicable guides for the information and data and is a better approach to running IT – one that is business-centric and ultimately user-centric – for advancing strategy and attaining goals.

A.2 Definitions

Harmonization with ArchiMate definitions is planned for a future release.

A.3 Overview

The IT Value Chain and IT4IT Reference Architecture represent the IT service lifecycle in a new and powerful way. They provide the missing link between industry standard best practice guides and the technology framework and tools that power the IT management ecosystem. The IT Value Chain and IT4IT Reference Architecture are a new foundation on which to base your IT operating model. Together, they deliver a welcome blueprint for the CIO to accelerate IT's transition to becoming a service broker to the business.

The IT Value Chain and IT4IT Reference Architecture strategy and content was created by a consortium of enterprise IT customers, vendors, and partners to address strategic challenges brought about by mobility, cloud, big data, security, and Bring Your Own Device (BYOD). The IT Value Chain is a prescriptive model for how to manage the service lifecycle as well as broker services to the enterprise using a multi-sourced approach.

As you read this document, consider the proposition that this IT operating model is the "missing link" allowing the IT function to achieve the same level of business discipline, predictability, and efficiency as other functions in the business such as the supply chain function.

The absence of a service-centric IT operating model in the past helps explain why all the time and money invested in best practice process frameworks like ITIL and COBIT, working with consultants to optimize IT organizational structure, and investments in IT automation have fallen short of expectations. Are you still unable to deliver services in a way that is flexible, traceable, and cost-effective? Is your organization still unable to consistently deliver what the business asked for in the timeframe and cost requirements they need it? Is too much of your resource still spent on maintaining the IT4IT infrastructure instead of on innovation projects? If so, read on to learn how the IT Value Chain and IT4IT Reference Architecture together provide a usable model for standardizing the automation fabric for IT to support constant innovation and accelerate service delivery.

A.3.1 Business Drivers for an Improved IT Operating Model

Typical IT departments were built to deliver and manage technology assets and capabilities that power business processes. This capability-centric model depended on investing heavily in people, processes, and technology to deliver and maintain what has become the "system of record" fabric for the enterprise. This system of record fabric is where individual domains, or silos, operate with their own language, culture, and process irrespective of the organization in front of or behind them in the IT Value Chain. This leads to the loss of focus on the true role of IT: to deliver services that make the company more competitive. In order to compete in the service economy, IT must change from a technology and project-centric orientation to a new IT management ecosystem oriented around a "system of engagement" that is focused on connecting people in new ways and creating new experiences and innovation opportunities. This consumerization of IT, BYOD, and the explosion of cloud-based services has provided business professionals and IT practitioners with new means of sourcing business and technology capabilities. This disruption is having a significant impact on the IT organization and has become a forcing function for a new IT model – one that supports the multi-sourced service economy, and one that enables new experiences in driving the self-sourcing of services that power innovation. This is a new style of IT where IT is a business innovation center, measured by the innovation it delivers, not the cost it consumes.

What strategy does your IT organization have to support this new multi-sourced service economy? Will your IT department support self-sourcing of services to power innovation? The redesign of IT needs to be happening now to avoid IT being highlighted as a hindrance to business innovation or worse, marginalized.

"Building a new fully integrated approach for managing IT – going beyond the traditional process models and disjointed solution landscapes – based on a common industry data model will give an important boost to our effort of becoming a world class IT provider."

Hans van Kesteren, VP & CIO Global Functions, Royal Dutch Shell

Where should you start on your transformation journey? The required changes do not start with the people, process, and technology. Instead the changes must start with the structure and focus of the organization, which in turn will impact roles, processes, and the IT automation and technology landscape. What is required is a new IT operating model – one that modernizes IT in the following four key areas:

- **Engagement Model**: Moving from exclusively project-led fulfillment to a self-service experience that puts the control over the pace of innovation in the hands of the service consumer (IT or business consumers).
- **Deliverables**: Shifting from the traditional "system of record" focus driven by monolithic, process-focused applications to a new service-centric orientation powered by hybrid, composite applications and information that evolve rapidly and continually.
- **Lifecycle**: Moving from a technology-centric, project lifecycle to managing the end-to-end Service Model and lifecycle. This opens the door for using new development methodologies like SCRUM, agile, and models such as DevOps and DevOpsSec.
- **Operating Model**: Moving from a technology delivery organization focused on silos toward a value chain model that promotes value-based consumption, greater cost transparency, and multi-sourced delivery of a broad-based service portfolio.

The IT4IT Forum developed the IT Value Chain to share a prescriptive standard for end-to-end automation of the IT service lifecycle. When captured and modeled correctly, the IT Value Chain-based model remains constant regardless of changes to process, technology, organization, and/or capabilities. The remainder of this chapter describes the IT Value Chain, the IT Value Streams, and the IT4IT Reference Architecture that support it.

Supporting activities will be further articulated in a future release of the Reference Architecture.

A.3.2 The IT Value Chain

Many industries and organizations have already institutionalized the value chain in their operating models. But the IT function has not, because of its heritage of providing technology enablement to the enterprise. Instead, it has organized around capabilities and technology silos, with each team being responsible for delivering different components of a particular business system using a multitude of disparate tools and methods. So, the traditional IT approach evolved around technology and process. Over the course of time this legacy IT approach has become ineffective for delivering outcomes that satisfy business needs in a timely manner. It has also made IT less competitive with third-party suppliers who are organized to deliver technology from a service-centric point of view.

Frameworks such as ITIL, COBIT, and eTOM, and standards such as ISO/IEC 20000, ISO/IEC 27002, ISO/IEC 38500, and others have also been applied inside traditional IT organizations in an attempt to drive greater efficiency, predictability, and reliability in the deliverables. The IT Value Chain does not replace these existing industry frameworks, it is complementary to them. The IT Value Chain and IT4IT Reference Architecture provide the missing underpinning IT operating model that removes the costly guesswork out of how to associate these frameworks to technology and how to implement vendor products in alignment with them. Together, they help to ensure repeatability and predictability of IT outcomes.

Through hundreds of hours interviewing customers and thousands of hours of functional and product analysis in R&D, it became obvious that a better, more business-centric approach to running IT was possible. IT deliverables were re-examined – shifting from technical capabilities to technology-enabled services. The IT4IT Forum applied Porter's value chain framework to the IT function as the foundation on which to design a robust and versatile IT operating model.

The IT Value Chain construct and the integration in the IT4IT Reference Architecture permeate everything that goes on in IT, to enable full service lifecycle traceability, mitigate risk and compliance concerns, improve cross-silo collaboration (DevOps), and ensure continuous delivery of business innovation. It can reduce cycles that are focused on maintenance to return more time and resource to the CIO to invest in what makes the business competitive – new services and innovation. Security is designed into the functional components of each value stream to provide a secure and cost-effective system for managing risk and compliance and to provide a more streamlined integration with the security teams. Once in place, this modular framework provides the foundation for successful adoption of the "IT as service broker" model and provides the CIO with the flexibility needed to successfully navigate an IT management ecosystem.

"In our industry, control of the end-to-end IT value chain is mandatory. That particularly includes effective and dynamic management of a multi-sourced landscape, which can only become a cost-efficient and high-quality reality with the right level of standardization. We're committed to the Consortium initiative as we strongly believe in the community as the basis for sustainable competitiveness. We've done it in other areas and know that it works."

Ton van der Linden, CIO, Achmea

A.4 IT4IT Core

Once industries and professions reach a certain level of maturity, efforts arise to standardize key terminology and concepts to foster improved knowledge exchange and collaboration. When done using an Enterprise Architecture framework or method, such attempts may be termed "reference architecture" or "reference models". Notable examples of such efforts include:

- Frameworx (eTOM), by the TM Forum
- BIAN – the Banking Industry Architecture Network
- ARTS – the work of the Association for Retail Technology Standards, an affiliate of the National Retail Federation
- The ACORD framework – an Enterprise Architecture for the insurance industry
- EMMM – the work of The Open Group Exploration, Mining, Metals & Minerals Forum

Advocates of these models point to their utility in defining standard interfaces as being a key contributor to reducing complexity and implementation costs for their industries. More broadly, as forms of frameworks they assist in stabilizing terminology, educating newcomers, and establishing shared meaning between collaborating parties.

The IT4IT Reference Architecture is an attempt to standardize certain aspects of IT management and has strong parallels to the efforts described above.

This chapter functions as the reference text that describes the structure, notation, models, and other concepts used in the IT4IT Reference Architecture. It also provides the rationale underpinning the design choices made by the standards team in developing the body of work. It is intended to help readers correctly interpret the content within the architecture so that it may be

applied consistently by IT practitioners and by suppliers of IT management products and services.

A.4.1 Value Streams

The IT4IT Reference Architecture is fundamentally a Functional Model for the IT management ecosystem. However, function-oriented reference models are frequently published from vendors of IT management products. These models are often viewed as "marketecture" which is designed to demonstrate the strengths of a vendor's products but are too ambiguous to be implementable. Most of these models are not based on architectural principles and/or are not connected to the broader IT operating model that describes how the function works. Therefore, the IT4IT Forum made two deliberate decisions: first, to design a reference model based on architectural design principles that could be implemented, and second to connect the Reference Architecture with an IT operating model based on an industry standard construct: the value chain.

The value chain concept originated with Michael Porter in his book Competitive Advantage[4] and focuses on the activities by which a company adds value to an article as it flows through the production and post-production lifecycle. This was combined with the concept of value streams[5] to capture the customer-in perspective. The value stream concept is rooted in Lean Six Sigma and emphasizes customer-oriented results.

A.4.2 Functional Components

The term functional component may be unfamiliar to some IT practitioners and vendors because it is not widely used. It has its roots in SOA reference models (such as the SOA Reference Architecture from The Open Group) and is intended to convey the notion of a small building block of technology. In the IT4IT Reference Architecture a single IT management product might play the role of one or more functional components. Conversely, a functional component might instead be as small as a web service or micro-service.

A.5 Strategy to Portfolio Value Stream

A.5.1 Related Standards, Frameworks, and Guidance

The S2P Value Stream extends the best practices put forward by various authors and frameworks, including ISO/IEC, the Capability Maturity Model Integration (CMMI), AXELOS Ltd. in its ITIL series on IT Service Management, ISACA in its COBIT Framework, and others, by defining the technology components necessary to implement them. A range of organizational maturity is also accommodated by the S2P Value Stream.

A non-normative list of related capabilities addressed by the S2P Value Stream is provided here, with the industry sources supporting their existence. Overall industry sources not tied to any one capability are: ISO/IEC 2005; Betz 2011; ISACA; ITIL Continual Service Improvement; ITIL

[4] Reference Michael Porter: Competitive Advantage: Creating and Sustaining Superior Performance.
[5] Reference James Martin: Great Transition: Using the Seven Disciplines of Enterprise Engineering.

Service Design; ITIL Service Operation; ITIL Service Strategy; ITIL Service Transition; ISO/IEC 2008; CMMI for Services.

IT Strategy and Architecture

Planning and overall strategy for IT services and sourcing (Benson, Bugnitz et al 2004; Kaplan 2005; Spewak & Hill 1993; Cook 1996; Carbone 2004; Ulrich & McWhorter 2010).

IT Demand and Portfolio Management

Business relationship management and pipeline for new IT services, service changes, and ongoing IT investment optimization (McFarlan 1981; Kaplan 2005; Maizlish & Handler 2005; ITIL Service Strategy).

IT Financial Management

IT financial planning and control, including pricing, budgeting, accounting, and charging (Quinlan 2003; Quinlan & Quinlan 2003; Remenyi, Money et al 2007; ITIL Service Strategy).

IT Governance, Risk, Security, and Compliance

Ensuring well-governed IT (direct-monitor-evaluate), correct management of adverse scenarios, protecting against malicious actors, and adhering to relevant governmental laws and regulations (ISACA; Van Grembergen 2004; Weill & Ross 2004; Van Grembergen & Haes 2009; ISO/IEC 2013).

A.6 Requirement to Deploy Value Stream

A.6.1 Related Standards, Frameworks, and Guidance

The R2D Value Stream extends the best practices put forward by various authors and frameworks, including ISO/IEC, the Capability Maturity Model Integration (CMMI), AXELOS Ltd. in its ITIL series on IT Service Management, ISACA in its COBIT Framework, and others, by defining the technology components necessary to implement them. A range of organizational maturity is also accommodated by the R2D Value Stream.

A non-normative list of related capabilities addressed by the R2D Value Stream is provided here, with the industry sources supporting their existence. Overall industry sources for this value stream are: Agile Alliance; ISO/IEC 2005; Betz 2011; ISACA; ITIL Continual Service Improvement; ITIL Service Design; ITIL Service Operation; ITIL Service Strategy; ITIL Service Transition; ISO/IEC 2008; PMBOK 2013; CMMI for Development; CMMI for Services; Rational Software 2011; Leffingwell, Yakyma et al 2014; ISO/IEC 2013.

Project Management

Define, manage, and control a defined set of (typically) non-repeatable work, to a scope of time, resources, and quality (PMBOK 2013).

Release and Deployment Management

Organize and coordinate the overall transition of new product functionality to a market available or enterprise production state. Includes both technical and organizational Change Management concerns (Klosterboer 2009; ITIL Service Transition; Humble & Farley 2011; OASIS TOSCA).

Requirements Management

Elicit, capture, track, and analyze the business needs of IT-centric products (CMMI for Development; Cockburn 2001; Leffingwell, Yakyma et al 2014).

Software Engineering

The core effort of developing new products and features based on computing and IT for market and enterprise needs (Duvall, Matyas et al 2007; Bourque & Fairley 2014; CMMI for Development).

Test Management

The activity of developing and executing specific assessments of an IT product's quality, lack of errors, and satisfaction of requirements (CMMI for Development, IEEE 730-2014).

Change Management

The activities and concerns related to implementing new functionality in an IT product, including communication, risk assessment, and correct execution (Klosterboer 2009; ITIL Service Transition; Van Schaik 1985).

Configuration Management

The capability supporting the overall management and understanding of complex inventories and dependencies in IT products and ecosystems (Betz 2011; ITIL Service Transition; O'Donnell & Casanova 2009).

A.7 Request to Fulfill Value Stream

A.7.1 Related Standards, Frameworks, and Guidance

The R2F Value Stream extends the best practices put forward by various authors and frameworks, including ISO/IEC, the Capability Maturity Model Integration (CMMI), AXELOS Ltd. in its ITIL series on IT Service Management, ISACA in its COBIT Framework, and others, by defining the technology components necessary to implement them. A range of organizational maturity is also accommodated by the R2F Value Stream.

A non-normative list of related capabilities addressed by the R2F Value Stream is provided here, with the industry sources supporting their existence. Overall industry sources not tied to any one capability are: ISO/IEC 2005; CMMI for Services; ISACA; ITIL Continual Service

Improvement; ITIL Service Design; ITIL Service Operation; ITIL Service Strategy; ITIL Service Transition; ISO/IEC 2008; O'Loughlin 2009.

Service Request Management and Service Desk

The capability and associated processes for interacting directly with end-users to provide operational support and benefits by an IT capability (market or enterprise-facing) (O'Loughlin 2009; ITIL Service Operation).

Supplier Management

The activity of defining needs, assessing sources, and establishing new vendor relationships for supplying the IT Value Chain, or its end-users (CMMI for Acquisition).

Service-Level Management

The activity of defining and managing to agreed performance criteria for IT services (Sturm, Morris et al 2000; ITIL Service Operation).

Access Management

The activity of provisioning and authenticating end-users in IT systems that require control over the user base (ISO/IEC 2013).

A.8 Detect to Correct Value Stream

A.8.1 Related Standards, Frameworks, and Guidance

The D2C Value Stream extends the best practices put forward by various authors and frameworks, including ISO/IEC, the Capability Maturity Model Integration (CMMI), AXELOS Ltd. in its ITIL series on IT Service Management, ISACA in its COBIT Framework, and others, by defining the technology components necessary to implement them. A range of organizational maturity is also accommodated by the D2C Value Stream.

A non-normative list of related capabilities addressed by the D2C Value Stream is provided here, with the industry sources supporting their existence. Overall industry sources not tied to any one capability are: ISO/IEC 2005; CMMI for Services; Betz 2011; ISACA; ITIL Continual Service Improvement; ITIL Service Design; ITIL Service Operation; ITIL Service Strategy; ITIL Service Transition; ISO/IEC 2008; Van Schaik 1985; Schiesser 2010; Kern, Shiesser et al 2004.

IT Operations Management

The set of activities related to actually running systems, ensuring that they are available, and performing on a day-to-day basis (Allspaw & Robbins 2010; Limoncelli, Chalup et al 2014).

Event Management

The detailed tracking of state changes within a managed IT resource or collection of resources, and the application of business roles and workflow to those raw occurrences for further treatment by processes such as Incident and Problem Management (Luckham 2002; Allspaw & Robbins 2010).

Incident Management

The identification, tracking, and resolution of occurrences where a service is not meeting expectations (Schiesser & Kern 2002; Kern, Shiesser et al 2004).

Application Management

The overall development and maintenance of market or enterprise-facing solutions (Office of Government Commerce 2002; ASL Foundation).

Capacity, Availability, and Problem Management

The reactive and proactive tracking and resolution of recurring or emerging problems in an IT product, including design shortcomings, over-consumption of resources, or emergent requirements (Behr, Kim et al 2005; Allspaw & Robbins 2010; ITIL Service Operation; Limoncelli, Chalup et al 2014).

Acronyms and Abbreviations

API	Application Program Interface
ARTS	Association for Retail Technology Standards
BIAN	Banking Industry Architecture Network
BRM	Business Risk Management
BYOD	Bring Your Own Device
CapEx	Capital Expenditure
CI	Configuration Item
CIO	Chief Information Officer
CMMI	Capability Maturity Model Integration
COBIT	Control Objectives for Information and Related Technology
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheet
DevOps	Development and Operations
DevOpsSec	Applying DevOps principles to Security
DML	Definitive Media Library
ECSS	European Cooperation for Space Standardization
EMMM	Exploration, Mining, Metals & Minerals (The Open Group)
eTOM	Business Process Framework (TM Forum)
HTML	Hypertext Mark-up Language
IaaS	Infrastructure as a Service
ISO	International Standards Organization
ISACA	Information Systems Audit and Control Association

IT	Information Technology
ITIL	Information Technology Infrastructure Library
JS	Java Script
KPI	Key Performance Indicator
KRI	Key Risk Indicator
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
OLA	Operational-Level Agreement
OpEx	Operating Expenditure
PaaS	Platform as a Service
PMBOK	Project Management Body of Knowledge
PMO	Project Management Office
QA	Quality Assurance
R&D	Research and Development
RFC	Request for Change
ROI	Return On Investment
SaaS	Software as a Service
SAFe	Scaled Agile Framework
SIEM	Security Information and Event Management
SLA	Service-Level Agreement
SLM	Service-Level Management
SLO	Service-Level Objective
SME	Subject Matter Expert
SOA	Service-Oriented Architecture
TOSCA	Topology and Orchestration Specification for Cloud Applications (OASIS)
UAT	User Acceptance Testing
UC	Underpinning Contract

UML	Unified Modeling Language
URL	Uniform Resource Locator
XaaS	X as a Service

Index