The Open Group IT4IT™ Standard, Version 3.0.1

A Reference Architecture for Managing Digital

Table of Contents

The Open Group IT4IT Standard, Version 3.0.1

A Reference Architecture for Managing Digital

The Open Group IT4IT™ Standard, Version 3.0.1
A Reference Architecture for Managing Digital
ISBN: 1-957866-52-9
Document Number: C24A

Published by The Open Group, October 2024.
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:
ogspecs@opengroup.org

Built with asciidoctor, version 2.0.20. Backend: pdf Build date: 2024-10-22 16:52:31 UTC

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards and open source initiatives by fostering a culture of collaboration, inclusivity, and mutual respect among our diverse group of 900+ memberships. Our membership includes customers, systems and solutions suppliers, tool vendors, integrators, academics, and consultants across multiple industries.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

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 standards and guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details are available at www.opengroup.org/library.

The IT4IT™ Forum

The IT4IT Forum is a group of member organizations that work together to solve shared challenges in Digital Product Management in the digital enterprise.

The mission of the IT4IT Forum is to continuously develop and drive the adoption of an open standard that:

- Provides a vendor-neutral reference architecture that delivers value-driven improvement to business outcomes
- Accelerates the adoption and delivery of end-to-end management of Digital Products and services

A fundamental objective of the IT4IT Forum is to drive adoption of the IT4IT Standard through a variety of activities including publishing how-to guides in the IT4IT extended body of knowledge.

The IT4IT Forum is composed of a diversity of member organizations, such as technology vendors, service providers, consulting companies, end-user organizations, training companies, academic institutions, and other digital enterprises. All come together in a technology independent, industry independent, and vendor-neutral environment to work together in a non-competitive, consensus-driven environment governed by The Open Group Standards Process.

Member organizations and their employees that participate in the Forum activities can expect benefits, including:

- Gaining competitive advantage through early access to pre-publication thought leadership
- Realizing more reliable outcomes by solving shared challenges with other like-minded professionals
- Establishing personal and professional relationships and a network of contacts for use long into the future
- Expanding digital management business insight through collaboration with other member organizations
- Establishing credibility as a thought leader in the industry by becoming a named contributor or co-author on standards of The Open Group and other publications
- Growing professional capabilities and promotion through dynamic learning exchanges in Forum discussions with other members

Proposals from IT4IT Forum members drive the strategy and content for successive versions of the IT4IT Standard. If you would like to contribute to future versions of the IT4IT Standard, we invite you to explore membership in The Open Group IT4IT Forum.

For further information about membership in the IT4IT Forum, visit www.opengroup.org/it4it-forum. For further information about the IT4IT Standard itself, visit www.opengroup.org/it4it.

The IT4IT Name

The business is increasingly dependent upon IT to enable their business capabilities and optimize their business value streams. IT is part of any business process and/or business product.

As a result, IT management is becoming a critical capability to ensure sustainable business success. To manage the increasing complexity of IT and digital, an organization needs to optimize their end-to-end IT management activities involved in the planning, development, delivery, and operations of Digital Products.

A more integrated approach is needed to optimize these IT value streams. The name “IT4IT” refers to this integrated approach of managing the IT specifically needed to enable and automate IT itself, such as portfolio and product backlog management, source code management, testing, deployment, identity management, monitoring. “IT4IT” refers to all digital management capabilities and practices needed to manage the IT/Digital Product Portfolio and thus ultimately be efficient in optimizing business outcome.

This Document

This document is the specification of The Open Group IT4IT Standard, Version 3.0.1, a standard of The Open Group. Version 3.0.1 is a minor maintenance update and incorporates Technical Corrigendum 1 of the IT4IT Standard, Version 3.0.

The IT4IT Standard addresses a critical gap in the Digital Transformation toolkit: the need for a unifying architectural model that describes and connects the capabilities, value streams, functions, and operational data needed to manage a Digital Product Portfolio at scale.

Traditional management paradigms, in which the technology budget is a combination of one-off projects and keep-the-lights-on operations, have constrained the value that could be delivered by technology. A fundamentally different approach is needed.

In recent years, this need continues to evolve rapidly as business management itself has become digital management. In other words, as the business delivers Digital Products, IT becomes the business.

By showing how to shift the focus of digital investment from project expense to product-based value delivery, the IT4IT Standard provides a powerful model for standardizing the digital automation fabric to support constant innovation and accelerated digital service delivery.

The ultimate target is a new style of technology management – “managing digital” – in which the primary metric for measuring IT investment value (and for measuring the performance of IT leaders) is the level of innovation and measurable business value delivered by a well-managed Digital Product Portfolio.

The Transformation Journey

The principle of product centricity shifts the focus of technology management away from the details of frictional project delivery and operations silos to a more holistic model focused instead on value-based consumption, customer focus, strong collaboration with consumers on end-to-end journeys, scalable automation, greater cost transparency, and the multi-sourced delivery of a broad Digital Product Portfolio.

Crucially, the IT4IT Standard provides a practical roadmap and blueprint for moving away from traditional practices and transitioning to a modern ability to manage digital at scale. The transition to managing digital typically includes several relevant journeys, such as moving from:

- Project-based to product-based technology investment management
- Waterfall methodologies to Agile planning and development
- Silo-oriented automation models to integrated, automated DevSecOps at scale
- Reactive order-taking to effectively managed and measured service brokerage
- Opaque operational and financial reporting to effective full-lifecycle, end-to-end visibility, and control of technology investment outcomes

Who Benefits from the IT4IT Standard?

"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, at the launch of The Open Group IT4IT Forum

The IT4IT Standard provides an approach to making digital investment decisions and managing digital outcomes that is particularly useful for:

- C-level executives responsible for Digital Transformation, as a top-down view of digital value creation
- Product Managers whose portfolios include significant digital content, as a way to integrate marketing priorities with product delivery practices
- Governance, risk, and compliance practitioners, as a guide to controlling a modern digital landscape
- Enterprise and IT Architects, as a template for IT tool rationalization and for governing end-to-end technology management architectures
- Technology buyers, as the basis for Requests for Information (RFIs) and Requests for Proposals (RFPs) and as a template for evaluating product completeness
- Consultants and assessors, as a guide for evaluating current practice against a well-defined standard
- Technology vendors, as a guide for product design and customer integrations
- Technical support staff, as a guide for automating and scaling up support services to deal with modern technology deployment velocity

Evolution of the Standard

The approach put forward over the lifetime of this standard has been 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, p.10 for extensive citations).

A history of the IT4IT Standard, including references to related standards, concepts, and industry themes, is published as a separate case study in the IT4IT Body of Knowledge; see The Open Group Case Study: *On the Origin of the IT4IT™ Standard* [Y202].

As part of the ongoing evolution of the IT4IT Standard, the IT Value Chain concept from Version 2.1 of the IT4IT Reference Architecture has been retired in favor of a focus on Digital Product Portfolio Management and the set of associated IT4IT Value Streams.

The Value Network metaphor has been proposed to describe the broad collaboration needed to connect core practices described in the IT4IT Standard to non-technology business domains such as

Human Resources (HR), Finance, Vendor Management, Customers, Partners, and Suppliers. It is consistent with the approach taken in the release to describe the standard in those terms; however, the Value Network concept has not been formally adopted by the IT4IT Forum at this time.

The IT4IT Standard, Version 3.0 Release Highlights

The following topics have been included/enhanced in Version 3.0 of the IT4IT Standard:

- Introduction of Digital Product

  A standard definition for “Digital Product” has been introduced. The Digital Product concept underpins and strengthens the traditional emphasis of the IT4IT Standard on treating the enterprise portfolio of IT applications/services as the primary metaphor for understanding and managing IT investment. As this thinking has matured, a “shift to product” has become a mainstream objective in IT strategy.

  The updated terminology and extended Digital Product definition reflect and support this trend and its implications for financial planning, value management, organization around Agile/DevOps teams, and the exploitation of modern automation options across the Digital Product lifecycle, from strategy to support.

- Introduction of Digital Product Backbone

  The concepts of service and a service backbone have been significantly improved in two ways. First, as part of the shift to product semantics, the term “service” is used primarily to describe the delivery of products “as a service” when the Digital Product is purely an act that is performed. The service backbone found in prior versions of the IT4IT Standard has been renamed “Digital Product Backbone” to account for a larger variety of topics that includes smartphones and other physical products, automated workflows, and even Robotic Process Automation “bots”. Second, the backbone has been simplified and made more straightforward, with a single primary data object at each stage.

- Move from Value Chain to Digital Value Network

  The use of “Value Network” as a concept for managing IT has been introduced. In the move to Digital Product semantics, Value Network replaces the Porter Value Chain [Porter] as the top-level, business view of the IT4IT Standard.

- New value streams

  The introduction of seven new value streams has replaced the four value streams of the IT4IT Value Chain of the IT4IT Standard, Version 2.1. Essentially, two value streams, “Evaluate” and “Explore”, are derived from Strategy to Portfolio. Requirement to Deploy is replaced with the “Integrate”, “Deploy”, and “Release” value streams; the “Consume” value stream replaces Request to Fulfill; and Detect to Correct is replaced with the “Operate” value stream. These new value streams are much more consistently and formally defined.

A common question is: what is the relationship between the new value streams in Version 3 and the value streams in Version 2.1?

Although strongly connected by data integrations and data flows, the original four IT4IT Value Streams are aligned to traditional IT organizational structures, which in most companies represented functional and cultural silos.

As the IT4IT Standard evolved into Version 3, IT organizations were also evolving and the old silos were giving way to concepts such as cross-functional development teams, new IT investment models, and DevOps integrations of development, deployment, and operations.

The new value streams in Version 3 take this evolution of industry into account, and align with modern IT management directions that are moving ever more strongly away from silos and toward the end-to-end integration of managing digital.

A close examination of both versions of the standard will quickly reveal the relationship between the old and new value stream definitions, and point the way to a migration path for those who have already implemented against the older version:

- Four functional groups derived from the value streams of the earlier IT4IT Standard, Version 2.1

  In the IT4IT Standard, Version 2.1 the four value streams – Strategy to Portfolio, Requirement to Deploy, Request to Fulfill, and Detect to Correct – were also defined to represent the groupings of the IT4IT Functional Components. We have preserved the groupings, but no longer refer to the groups as value streams:

  - Updated Strategy to Portfolio functional components

    In Strategy to Portfolio, a Strategy functional component is introduced and significant updates have been made to the way strategy, architecture, and Digital Product work together.

  - Updated Requirement to Deploy functional components

    Requirement to Deploy has been upgraded significantly to reflect modern Agile and DevOps operating practices. This includes renaming some data objects and functional components to reflect the typical terms used in Agile.

  - Updated Request to Fulfill functional components

    Change Management has been moved from Detect to Correct to Request to Fulfill to reflect that change is an activity managed by the Deliver functions. Furthermore, Request to Fulfill sees the introduction of Identity Management, as well as the better formalization of the Service Offer Catalogs and Consumption Experience.

  - IT Financial Management (ITFM) Support functions

    The IT4IT Reference Architecture has been updated to improve the description of how Financial Management capabilities are supported by the standard. Financial Management is one of the

Supporting Functions in the overall Digital Value Network, and its impacts on core functions and data objects have been updated to more effectively describe these impacts and interactions.

- Use of the ArchiMate® modeling language as the standard notation

  The ArchiMate Specification has replaced most instances of the "informal notation" used in previous releases. This generally improves the rigor of the diagrams. It also enables the automatic creation of these diagrams from the data held in the ArchiMate model of the IT4IT Reference Architecture that is available for download with the IT4IT Standard, Version 3.0. This ensures a high level of consistency across the model.

- Removal of the Key Performance Indicator (KPI) lists

  The lists of KPIs associated with the four value streams in the previous release have been removed. The creation and management of appropriate metrics and KPIs for activities described in the IT4IT Standard are addressed at various points in the text of the standard. The Open Group Guide: *Intelligence & Reporting Supporting Activity in the IT4IT™ Reference Architecture* [G18E] describes a recommended way of approaching metrics and KPIs.

- General consistency and flow of the overall standard

  Inconsistencies of terminology and structure that were reported against prior versions of the IT4IT Standard have been resolved.

Related Industry Standards

The IT4IT Reference Architecture provides the overall framework for managing a “digital factory”, covering the value streams, capabilities, and data flows needed to manage the entire Digital Product lifecycle. The IT4IT Standard can be combined with other practices and standards providing additional guidance for specific capabilities or functions. Therefore, the IT4IT Reference Architecture can be complemented with other practices and standards, such as those listed below.

Enterprise Architecture

- The Open Group ArchiMate® Specification
- The Open Group Open Agile Architecture™ Standard
- The Open Group TOGAF® Standard

(Scaled) Agile Development

- Kanban
- Large Scale Scrum – LeSS
- Nexus™ for Scaling Scrum
- Scaled Agile Framework® (SAFe®)
- Scrum

Project Management

- PRINCE2® for Project Management
- The Project Management Body of Knowledge (PMBOK™) Guide

IT Service Management

- ISO/IEC 20000: Information Technology – Service Management
- ITIL® for IT Service Management from AXELOS
- The VeriSM™ Framework

IT Governance

- COBIT® for IT Governance by ISACA
- ISO/IEC 38500: Corporate Governance of Information Technology

Software Asset Management

- ISO/IEC 19770: Software Asset Management

Security and Risk Management

- ISO/IEC 27000 : Information Security Management systems
- NIST Cybersecurity Framework

Other Practices

- Capability Maturity Model Integration (CMMI®)
- DevOps

- OASIS™ Topology and Orchestration Specification for Cloud Applications (TOSCA™)
- Object Management Group® (OMG®) Unified Modeling Language™ (UML®)
- Site Reliability Engineering
- The Open Group Digital Practitioner Body of Knowledge™
- The Open Group FACE™ Technical Standard
- The Open Group Healthcare Enterprise Reference Architecture (HERA)

Referenced Documents

The following documents are referenced in this Standard.

(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 Normative References.

Informative References

The following documents are referenced in this standard or provide further information:

[Agile Manifesto]	*Manifesto for Agile Software Development*, by Beck et al. 2001; refer to: http://agilemanifesto.org/principles.html
[Allspaw & Robbins]	*Web Operations: Keeping the Data On Time*, by John Allspaw and Jesse Robbins, July 2010, published by O'Reilly Media
[Behr et al.]	*The Visible Ops Handbook: Implementing ITIL® in Four Practical and Auditable Steps*, by Kevin Behr, Gene Kim, and George Spafford, June 2005, published by Information Technology Process Institute
[Benson et al.]	*From Business Strategy to IT Action: Right Decisions for a Better Bottom Line*, by Robert J. Benson, Thomas L. Bugnitz, and William B. Walton, April 2008, published by Wiley
[Betz]	*Architecture and Patterns for IT Service Management, Resource Planning, and Governance (Making Shoes for the Cobbler's Children)*, by Charles T. Betz, November 2011, 2nd Edition, published by Morgan Kaufmann
[Bourque & Fairley]	*Guide to the Software Engineering Body of Knowledge (SWEBOK®): Version 3.0*, edited by Pierre Bourque and Richard E. Fairley, January 2014, published by IEEE Computer Society Press
[C119]	*SOA Reference Architecture (C119)*, a standard of The Open Group, December 2011, published by The Open Group; refer to: www.opengroup.org/library/c119
[C155]	*The Open Group IT4IT™ Reference Architecture, Version 2.0 (C155)*, a standard of The Open Group Standard, October 2015, published by The Open Group; refer to: www.opengroup.org/library/c155
[C171]	*The Open Group IT4IT™ Reference Architecture, Version 2.1 (C171)*, a standard of The Open Group, January 2017, published by The Open Group; refer to: www.opengroup.org/library/c171

[C19C] *ArchiMate® Model Exchange File Format for the ArchiMate Modeling Language, Version 3.1 (C19C)*, a standard of The Open Group, November 2019, published by The Open Group; refer to: www.opengroup.org/library/c19c

[C19E] *Open Messaging Interface (O-MI), The Open Group Standard for the Internet of Things (IoT), Version 2.0 (C19E)*, December 2019, published by The Open Group; refer to: www.opengroup.org/library/c19e

[C202] *O-DEF™, the Open Data Element Framework, Version 2.0 (C202)*, a standard of The Open Group, February 2020, published by The Open Group; refer to: www.opengroup.org/library/c202

[C207] *FACE™ Technical Standard, Edition 3.1 (C207)*, a standard of The Open Group, July 2020, published by The Open Group; refer to: www.opengroup.org/library/g207

[Carbone] *IT Architecture Toolkit (Enterprise Computing)*, by Jane Carbone, May 2004, published by Prentice Hall

[CMMI for Acquisition] *CMMI® for Acquisition, Version 1.3*, by CMMI Product Team, November 2010, published by Carnegie Mellon University Software Engineering Institute; refer to: https://resources.sei.cmu.edu/asset_files/technicalreport/2010_005_001_15284.pdf

[CMMI for Development] *CMMI® for Development, Version 1.3*, by CMMI Product Team, November 2010, published by Carnegie Mellon University Software Engineering Institute; refer to: https://resources.sei.cmu.edu/asset_files/technicalreport/2010_005_001_15287.pdf

[CMMI for Services] *CMMI® for Services, Version 1.3*, by CMMI Product Team, November 2010, published by Carnegie Mellon University Software Engineering Institute; refer to: https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15290.pdf

[COBIT] *ISACA: Control Objectives for Information and Related Technology (COBIT®)*; refer to: www.isaca.org

[Cockburn] *Writing Effective Use Cases*, by Alistair Cockburn, October 2000, published by Addison-Wesley

[Cook] *Building Enterprise Information Architectures: Reengineering Information Systems*, by Melissa A. Cook, February 1996, published by Prentice Hall

[Duvall et al.] *Continuous Integration: Improving Software Quality and Reducing Risk*, by Paul M. Duvall, Steve Matyas, and Andrew Glover, June 2007, published by Addison-Wesley

[G160] *IT4IT™ for Managing the Business of IT, A Management Guide (G160)*, The Open Group Guide, January 2016, published by The Open Group; refer to: www.opengroup.org/library/g160

[G18E] *Intelligence & Reporting Supporting Activity in the IT4IT™ Reference Architecture (G18E)*, The Open Group Guide, October 2018, published by The Open Group; refer to: www.opengroup.org/library/g18e

[G18F] *Service Brokering with the IT4IT™ Standard (G18F)*, The Open Group Guide, December 2018, published by The Open Group; refer to: www.opengroup.org/library/g18f

[G191] *Tool Rationalization using the IT4IT™ Reference Architecture Standard (G191)*, The Open Group Guide, April 2019, published by The Open Group; refer to: www.opengroup.org/library/g191

[Humble & Farley] *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, by Jez Humble and David Farley, August 2010, published by Addison-Wesley

[IBM Rational Software] Rational Software: Rational Unified Process Best Practices for Software Development Teams, revised January 2011, published by Rational Software; refer to: ftp://ftp.software.ibm.com/software/rational/web/whitepapers/2003/rup_bestpractices.pdf

[IEEE 730-2014] *IEEE 730-2014: IEEE Standard for Software Quality Assurance Processes*, June 2014; refer to: https://standards.ieee.org/standard/730-2014.html

[ISO/TC 258] *ISO/TC 258: Project, Program, and Portfolio Management*; refer to: https://www.iso.org/committee/624837.html

[ISO/IEC 21] *ISO/IEC Guide 21-2:2005 Regional or National Adoption of International Standards and Other International Deliverables – Part 2: Adoption of International Deliverables other than International Standards*; refer to: https://www.iso.org/standard/39800.html

[ISO/IEC 98] *ISO/IEC 98-3:2008 Uncertainty of Measurement – Part 3: Guide to the Expression of Uncertainty in Measurement (GUM:1995)*; refer to: https://www.iso.org/standard/50461.html

[ISO/IEC 19770] *ISO/IEC 19770-1:2017: Information Technology – Software Asset Management – Part 1: IT asset management systems — Requirements*; refer to: https://www.iso.org/standard/68531.html

[ISO/IEC 20000] *ISO/IEC 20000-1:2018: Information Technology – Service Management – Part 1: Service Management System Requirements*; refer to: https://www.iso.org/standard/70636.html

[ISO/IEC 27001] *ISO/IEC 27001: Information Security Management*; refer to: https://www.iso.org/isoiec-27001-information-security.html

[ISO/IEC 27002] *ISO/IEC 27002:2022: Information security, cybersecurity and privacy protection — Information security controls*; refer to: https://www.iso.org/standard/75652.html

[ISO/IEC 38500] *ISO/IEC 38500:2015: Corporate Governance of Information Technology*; refer to: https://https://www.iso.org/standard/62816.html

[ITIL] *ITIL® Foundation, ITIL 4 Edition*, 2019, published by AXELOS_; refer to: https://www.axelos.com/certifications/itil-service-management

[Kaplan] *Strategic IT Portfolio Management: Governing Enterprise Transformation*, by Jeffrey Kaplan, May 2009, published by Jeff Kaplan

[Kern et al.]	*IT Production Services*, by Harris Kern, Rich Schiesser, and Mayra Muniz, July 2011, published by Prentice Hall
[Leffingwell et al.]	*Scaled Agile Framework®* (SAFe®), by D. Leffingwell, A. Yakyma, et al, 2014; refer to: http://scaledagileframework.com
[Limoncelli et al.]	*The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems, Volume 2: DevOps and SRE Practices for Web Services*, by Thomas A. Limoncelli, Strata R. Chalup, and Christina J. Hogan, September 2014, published by Addison-Wesley Professional
[Luckham]	*The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, by David C. Luckham, May 2002, published by Addison-Wesley
[Maizlish & Handler]	*IT Portfolio Management Step-By-Step: Unlocking the Business Value of Technology*, by Bryan Maizlish and Robert Handler, October 2010, published by Wiley
[Martin]	*Great Transition: Using the Seven Disciplines of Enterprise Engineering*, by James Martin, October 1995, published by Amacom
[McFarlan]	*Portfolio Approach to Information Systems*, by F. Warren McFarlan, September 1981, published in the Harvard Business Review 59(5); refer to: https://hbr.org/1981/09/portfolio-approach-to-information-systems
[Merriam-Webster]	*Merriam-Webster Dictionary*; refer to: www.merriam-webster.com/
[O'Donnell & Casanova]	*The Configuration Management Database (CMDB) Imperative: How to Realize the Dream and Avoid the Nightmares*, by Glenn O'Donnell and Carlos Casanova, February 2009, published by Pearson
[OGC]	*Application Management*, by Office of Government Commerce, June 2002, published by The Stationary Office
[O'Loughlin]	*The Service Catalog: A Practitioner Guide*, by Mark O'Loughlin, March 2010, published by Van Haren Publishing
[Porter]	*Competitive Advantage: Creating and Sustaining Superior Performance*, by Michael E. Porter, January 2004, published by Free Press
[PMBOK Guide]	*A Guide to the Project Management Body of Knowledge (PMBOK™ Guide)*, January 2013, published by Project Management Institute
[Quinlan]	*Chargeback and IT Cost Accounting*, by Terence A. Quinlan, January 2003, Published by IT Financial Management Association
[Quinlan & Quinlan]	*Readings in IT Financial Management*, by Terence A. Quinlan and Susan J. Quinlan, 2003, published by IT Financial Management Association
[Remenyi et al.]	*The Effective Measurement and Management of ICT Costs and Benefits*, by Dan Remenyi, Frank Bannister, and Arthur Money, February 2007, published by CIMA Publishing

[Ross et al.] *Designed for Digital: How to Architect your Business for Sustained Success (Management on the Cutting Edge)*, by Jeanne W. Ross, Cynthia M. Beath, Martin Mocker, September 2019, published by MIT Press

[S182] *Healthcare Enterprise Reference Architecture (HERA) (S182)*, The Open Group Snapshot, April 2018, published by The Open Group; refer to: www.opengroup.org/library/s182

[S210] *IT4IT™ Reference Architecture, Version 3.0: Managing Digital Excerpt (S210)*, The Open Group Snapshot, January 2021, published by The Open Group; refer to: www.opengroup.org/library/s210

[Schiesser 2001] *IT Systems Management: Designing, Implementing, and Managing World-Class Infrastructures*, by Rich Schiesser, January 2001, published by Prentice Hall

[Schiesser 2010] *IT Systems Management*, by Rich Schiesser, January 2010, published by Pearson

[Spewak & Hill] *Enterprise Architecture Planning – Developing a Blueprint for Data, Applications, and Technology*, by Steven H. Spewak and Steven C. Hill, October 1993, published by Wiley-QED Publications

[Sturm et al.] *Foundations of Service Level Management*, by Rick Sturm, Wayne Morris, and Mary Jander, April 2000, published by Sams Publishing

[TOGAF Standard] *The TOGAF® Standard, 10th Edition (C220)* a standard of The Open Group, April 2022, published by The Open Group; refer to: www.opengroup.org/library/c220

[TOGAF Value Streams] *TOGAF® Series Guide: Value Streams (G178)* a guide of The Open Group, October 2017, published by The Open Group; refer to: www.opengroup.org/library/g178

[OASIS TOSCA] *OASIS™: Topology and Orchestration Specification for Cloud Applications (TOSCA™), Version 1.0*, November 2013; refer to: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[Ulrich & McWhorter] *Business Architecture: The Art and Practice of Business Transformation*, by William Ulrich and Neal McWhorter, October 2010, published by Meghan-Kiffer Press

[UML] *Unified Modeling Language™ (UML®)*, Object Management Group® (OMG®); refer to: www.uml.org

[Van Grembergen] *Strategies for Information Technology Governance*, by Wim Van Grembergen, 2004, published by Idea Group Publishing; refer to: https://www.researchgate.net/profile/Reima-Suomi/publication/314501133_Governance_Structures_for_IT_in_the_Health_Care_Industry/links/5519414a0cf2d241f355f085/Governance-Structures-for-IT-in-the-Health-Care-Industry.pdf

[Van Schaik] *A Management System for the Information Business: Organizational Analysis*, by Edward A. Van Schaik, January 2006, published by Prentice Hall

[W17B] *Defining the IT Operating Model, White Paper (W17B)*, September 2017, published by The Open Group; refer to: www.opengroup.org/library/w17b

[W180] *IT4IT™ Business Value: Providing Operational Value with the IT4IT Standard, White Paper (W180)*, January 2018, published by The Open Group; refer to: www.opengroup.org/library/w180

[W181] *IT4IT™ Business Value: Delivering Business Value with IT, White Paper (W181)*, January 2018, published by The Open Group; refer to: www.opengroup.org/library/w181

[W183] *Seamless Service Delivery and the IT4IT™ Standard, White Paper (W183)*, May 2018, published by The Open Group; refer to: www.opengroup.org/library/w183

[W185] *How to Use the TOGAF® and IT4IT™ Standards Together, White Paper (W185)*, May 2018, published by The Open Group; refer to: www.opengroup.org/library/w185

[W188] *Driving Business Outcomes Using the IT4IT™ Standard and SAFe®, White Paper (W188)*, December 2018, published by The Open Group; refer to: www.opengroup.org/library/w188

[W205] *The Shift to Digital Product: A Full Lifecycle Perspective, White Paper (W205)*, December 2020, published by The Open Group; refer to: www.opengroup.org/library/w205

[Weill & Ross] *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*, by Peter Weill and Jeanne W. Ross, June 2004, published by Harvard Business Review Press

[Y202] *On the Origin of the IT4IT™ Standard*, a Case Study of The Open Group, April 2020, published by The Open Group; refer to: www.opengroup.org/library/y202

Trademarks

ArchiMate, FACE, FACE logo, Future Airborne Capability Environment, Making Standards Work, Open Footprint, Open O logo, Open O and Check certification logo, OSDU, The Open Group, TOGAF, UNIX, UNIXWARE, and X logo are registered trademarks and Boundaryless Information Flow, Build with Integrity Buy with Confidence, Commercial Aviation Reference Architecture, Dependability Through Assuredness, Digital Practitioner Body of Knowledge, DPBoK, EMMM, FHIM Profile Builder, FHIM logo, FPB, IT4IT, IT4IT logo, O-AA, O-DA, O-DEF, O-HERA, O-PAS, O-TTPS, Open Agile Architecture, Open FAIR, Open Process Automation, Open Subsurface Data Universe, Open Trusted Technology Provider, Sensor Integration Simplified, Sensor Open Systems Architecture, SOSA, and SOSA logo are trademarks of The Open Group.

Amazon Web Services is a trademark of Amazon Technologies, Inc.

Android is a trademark of Google LLC.

Apple is a registered trademark of Apple, Inc.

Azure, Microsoft, and Windows are registered trademarks and Microsoft Teams is a trademark of Microsoft Corporation.

Cisco WebEx is a registered trademark of Cisco Systems, Inc.

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

COBIT and ISACA are registered trademarks of the Information Systems Audit and Control Association (ISACA) and the IT Governance Institute.

eTOM is a registered trademark and Frameworx is a trademark of the TM Forum.

GitLab is a registered trademark of GitLab BV.

ITIL is a registered trademark of AXELOS Ltd.

Java is a registered trademark and JavaScript is a trademark of Oracle and/or its affiliates.

OASIS and TOSCA are trademarks of OASIS.

Object Management Group, OMG, and UML are registered trademarks and Unified Modeling Language is a trademark of the Object Management Group, Inc.

PMBOK is a trademark of Project Management Institute, Inc.

SAFe and Scaled Agile Framework are registered trademarks of Scaled Agile, Inc.

SLACK is a trademark of Slack Technologies, Inc.

SWEBOK is a registered trademark of the IEEE.

VeriSM is a trademark of IFDC.

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 document was prepared by The Open Group IT4IT™ Forum. The Open Group gratefully acknowledges the leadership and contributions of the following individuals (where appropriate, arranged by elected officers, followed by alphabetical order by organization affiliation) in the development of The Open Group IT4IT Standard, Version 3.0 and Version 3.0.1.

NOTE For the company name of each individual included here, we endeavored to list the employer affiliation they had at the time they contributed to the standard.

Lead Architects, Version 3.0

The Open Group gratefully acknowledges the following individuals for their contributions to the IT4IT Standard, Version 3.0.

IT4IT Core Standard Standing Committee topic leaders:

- Lars Rossen, Micro Focus, Chair of the IT4IT Core Standard Standing Committee
- Philippe Geneste, Accenture
- Sylvain Marie, Accenture
- Richard Aarnink, Achmea
- Guido de Jong, Achmea
- Mark Luchtmeijer, Achmea
- Jason Thurmond, Boeing
- Dan Warfield, CC and C Solutions
- Rob Akershoek, DXC
- Satya Misra, HCL Technologies
- Etienne Terpstra-Hollander, Micro Focus
- Erik van Busschbach, Micro Focus
- Michael Fulton, Nationwide
- Stephanie Ramsay, Raytheon Technologies
- Mark Bodman, ServiceNow
- Michelle Supper, ServiceNow
- Sue Desiderio, Invited Expert

ArchiMate® Model of the IT4IT Reference Architecture Work Group topic leaders:

- Kees van den Brink, ServiceNow, Chair of the ArchiMate Model of the IT4IT Reference Architecture Work Group

- Martin Tax, DAIN
- Søren Hansen, Micro Focus
- Lars Rossen, Micro Focus
- Etienne Terpstra-Hollander, Micro Focus

Active Participant Contributors

In addition to the Lead Architects above, The Open Group gratefully acknowledges the following individuals who contributed by regularly attending the live face-to-face quarterly members-only meetings or live virtual weekly meetings of the IT4IT Forum and providing content, use-cases, models, GitLab® merge requests, or votes on decisions during discussions of iterative drafts of the IT4IT Standard, Version 3.0:

- Sri Srinivasan, Accenture
- Lars Kristian Larsen, BusinessNow
- Jan Stobbe, Capgemini
- Dave Hornford, Conexiam
- Didier Beyens, DXC
- Ben Noordzij, DXC
- Roan van Helten, DXC
- Karel van Zeeland, DXC
- Andrew Platt, Fujitsu
- Prafull Verma, HCL Technologies
- David Morlitz, IBM
- Sukumar Daniel, ING Bank
- Xavier Mayeur, ING Bank
- Soumajit Das, Micro Focus
- Søren Hansen, Micro Focus
- Peter Vollmer, Micro Focus
- Altaz Valani, Security Compass
- Jeannine McConnell, ServiceNow
- Sam Courtney, Sparx Services North America
- Charles Betz, University of St. Thomas
- Chris Madden, ValueFlow
- Luke Sorensen, ValueFlow

IT4IT Forum Steering Committee

The Open Group gratefully acknowledges the leadership of these IT4IT Forum Officers who were in office at the time the standard was approved after Company Review:

- Rob Akershoek, DXC, Forum Co-Chair
- Satya Misra, HCL, Forum Co-Chair
- Etienne Terpstra-Hollander, Micro Focus, Forum Vice-Chair and The Open Group Governing Board Member
- Linda Kavanagh, The Open Group, Forum Director
- Lars Rossen, Micro Focus, Chair of the IT4IT Core Standard Standing Committee and Chair of The Open Group Governing Board
- Jan Stobbe, Capgemini, Co-Chair of the IT4IT Guidance Standing Committee
- Andrew Platt, Fujitsu, Co-Chair of the IT4IT Guidance Standing Committee
- Michael Fulton, Co-Chair of the IT4IT Training and Certification Standing Committee
- Sylvain Marie, Co-Chair of the IT4IT Training and Certification Standing Committee
- Prafull Verma, HCL Technologies, IT4IT Adoption Standing Committee and The Open Group Governing Board member
- Stephanie Ramsay, Raytheon Technologies, Chair of the Strategy to Portfolio Work Group and The Open Group Governing Board member
- Dan Warfield, CC and C Solutions, Chair of the IT4IT Product Lifecycle Management Standing Committee
- Mark Bodman, ServiceNow, Chair of the IT4IT IT Reference Operating Model Work Group
- Kees van den Brink, ServiceNow, Chair of the ArchiMate Model of the IT4IT Reference Architecture Work Group

Reviewers

The Open Group gratefully acknowledges the reviewers below who submitted written comments or otherwise participated in comment resolution during The Open Group Company Review process or during the draft development process of the IT4IT Standard, Version 3.0 or the Snapshot of the IT4IT Reference Architecture, Version 3.0: Managing Digital Excerpt [S210]. In addition, we thank those who gave significant feedback in content reviews held at live or virtual members-only meetings:

- David Stefferud, Accenture
- Antonio diPerna, Bank United
- Leonard Wiens, Beniva Consulting
- Enoch Andrade, CC and C Solutions
- Dave Hornford, Conexiam

- Myles Suer, Dell Technologies (Boomi)
- Alain Lord, Desjardins
- Justin Mann, Digital Business Consulting, LLC
- Trey Harris, ExxonMobil
- Rosemary Peh, Hawaiian Electric Company
- Varun Vijaykumar, HCL Technologies
- Emmanuel Amamoo-Otchere, Huawei Technologies, Co. Ltd.
- James Doss, IT Management & Governance, LLC
- Andreas Hartmann, Leipzig University of Applied Sciences
- Steven Jarman, Micro Focus
- Ben Rabau, Micro Focus
- Gerben Verstraete, Micro Focus
- Muhammad Suleman, Microsoft
- David Gilmour, Mundo Cognito Ltd.
- Miranda Gidderon, NASA SEWP
- Fei F. Peng, Oracle Corporation
- Ronald Israels, Quint
- John Wagemaker, Quint
- Kevin Kaiser, Raytheon Technologies
- Matthew Kehret, Raytheon Technologies
- Brian Moore, Raytheon Technologies
- David Mosko, Raytheon Technologies
- Chris Armstrong, Sparx Services North America
- Andrew Josey, The Open Group
- David Lounsbury, The Open Group
- Dave Favelle, ValueFlow

Version 3.0.1

The Open Group gratefully acknowledges the following individuals who contributed by providing content, use-cases, models, GitLab® merge requests, or votes on decisions during discussions of Technical Corrigendum 1 for The Open Group IT4IT Standard, Version 3.0, incorporated here to form Version 3.0.1:

- Rob Akershoek, DXC
- Satya Misra, HCL Software
- Etienne Terpstra-Hollander, Eenpool B.V.
- Dan Warfield, Managing Digital
- Mark Bodman, ServiceNow
- Lars Rossen, OpenText
- Kees van den Brink, ServiceNow
- Eric D Choi, NASA SEWP
- Andreas Hartmann, Leipzig University of Applied Sciences
- Stephanie Ramsey, RTX
- Maarten Keverkamp, OpenText

Previous Versions

The Open Group gratefully acknowledges The Open Group IT4IT Forum members past and present for developing the previous versions of the IT4IT Standard. Their names can be found in the Acknowledgements pages of the previous versions of this standard: the IT4IT Reference Architecture, Version 2.0 [C155] and Version 2.1 [C171].

The Open Group gratefully acknowledges these members of The Open Group – all employees of DAIN sro. – for the provision of and/or technical support on the ArchiRepo tool which allowed multiple member authors to simultaneously collaborate on building the ArchiMate model of the IT4IT Reference Architecture, as well as for their guidance in the usage of the ArchiMate modeling language:

- Roman Hak
- Leo Mates
- Martin Tax

Elected Officers

The Open Group gratefully acknowledges those past and present members of the IT4IT Forum who have served as its member-elected officers (Chairs, Co-Chairs, and Vice-Chairs) since its inception. In alphabetical order:

- Rob Akershoek
- Mark Bodman
- Chris Davis
- Michael Fulton
- Satya Misra
- Stephanie Ramsay
- Etienne Terpstra-Hollander
- Erik Witte

IT4IT Forum Member Organizations

An up-to-date list of Forum members can be found at: http://www.opengroup.org/it4it-Forum.

[This page is intentionally blank]

Chapter 1. Introduction

1.1. Objective

The Open Group IT4IT™ Standard, Version 3.0.1, describes a reference architecture that can be used to manage the business of Information Technology (IT) and the associated end-to-end lifecycle management of Digital Products.

It is intended to provide a prescriptive Target Architecture and clear guidance for the transformation of existing technology management practices for a faster, scalable, automated, and practical approach to deploying product-based investment models and providing an unprecedented level of operational control and measurable value.

This foundational IT4IT Reference Architecture is independent of specific technologies, vendors, organization structures, process models, and methodologies. It can be mapped to any existing technology landscape. It is flexible enough to accommodate the continuing evolution of operational and management paradigms for technology. It addresses every Digital Product lifecycle phase from investment decision-making to end-of-life.

1.2. Conformance

Refer to The Open Group website for conformance requirements for this document.

1.3. Normative References

This document contains provisions which, through references in this standard, constitute provisions of The Open Group IT4IT Standard. 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® 3.2 Specification (C226), a standard of The Open Group, October 2022, published by The Open Group; refer to: www.opengroup.org/library/c226

1.4. Terminology

For the purposes of this document, 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. To avoid ambiguity, do not use “must” as an alternative to “shall”.

Shall not

Describes a feature or behavior that is an absolute prohibition.

Should

Describes a feature or behavior that is recommended but not required.

Will

Same meaning as “shall”; “shall” is the preferred term.

1.5. Future Directions

It is expected that this document will need to be revised from time to time to remain current with both practice and technology.

Chapter 2. Definitions

For the purposes of this document, the following terms and definitions apply. Merriam-Webster's Collegiate Dictionary should be referenced for terms not defined in this document.

2.1. Contract

The relationship between provider and consumer when an Offer is accepted by a consumer and an instance of the Digital Product created.

Note: Contract is defined in detail in Contract Definition.

2.2. Digital Product

A service, physical item, or digital item that provides an agreed and specific outcome for a consumer, and that incorporates and requires software to realize that outcome.

Note: Digital Product is defined in detail in Digital Product Definition.

2.3. Digital Product Backbone Data Object

A key data object in the IT4IT Reference Architecture that annotates an aspect of the Digital Product model in its planning, development, consumable, or running state.

Note: Digital Product Backbone data object is defined in detail in Digital Product Backbone Data Objects.

2.4. Functional Component

The smallest unit of technology that can stand on its own and be useful as a whole to a Digital Practitioner (or IT organization)

Note: Functional component is defined in detail in Functional Component.

2.5. Key Data Object

Data (records, information, and so on) that annotate or model an aspect of a Digital Product being planned, developed, offered, or consumed.

Note: Key data object is defined in detail in Key Data Object.

2.6. Service Offer

Possible Service Contract terms as perceived by the potential consumer of a Digital Product Instance.

Note: Service Offer is defined in detail in Service Offer Definition.

2.7. Service Offer Backbone Data Object

A key data object in the IT4IT Reference Architecture that controls and defines the consumption of a Digital Product.

Note: Service Offer Backbone data object is defined in detail in Service Offer Backbone Data Objects.

2.8. System

Technology components of a Digital Product Instance that will be managed by the provider.

Note: System is defined in detail in System Definition.

2.9. System of Record

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

Note: System of record is defined in detail in System of Record.

2.10. Value Network

A construct that defines the relationships across the value-creating functions that deliver value.

Note: Value Network is defined in detail in Value Network.

2.11. Value Stream

The value-creating activities for discrete areas within the Value Network where some unit of net value is created or added to the Digital Product as it progresses through its lifecycle

Note: Value stream is defined in detail in Value Stream.

Chapter 3. Digital Management

This chapter provides an overview of all the IT4IT Reference Architecture. In the IT4IT Standard this is referred to as Level 1.

3.1. Foundational Concepts

At the most basic level, the IT4IT Standard defines the “Digital Management” of “Digital Products” in order to create an efficient value delivery. This is illustrated in Figure 3-1.

Figure 3-1The IT4IT Value Delivery Framing

A basic understanding of a few foundational IT4IT concepts is necessary for any detailed understanding of the material that follows: the standard describes how **data objects**, **functional components**, and **value streams** interact to deliver business value. Each is discussed briefly here.

Formal definitions of these and other terms can be found in IT4IT Concepts and Metamodel.

Data objects: The most critical element of the IT4IT Standard is the identification of which information must exist and be effectively integrated for the effective end-to-end management of the Digital Product lifecycle.

The information described by each data object may be stored in a database table but the storage method is not specified by the IT4IT Standard. Likewise the format of an instance of a data object can be any relevant format (e.g., model file, source file, script, executable file, a development deliverable, a word processing document, a mail message) that can be indexed by its associated functional component.

In many organizations, much of this information is held in informal, poorly managed, or poorly integrated systems including manual filing systems or as static documents. Part of the IT4IT Standard is about ensuring what data should be identified and which functional component should control the data object through its lifecycle.

Some key data objects make up the Digital Product Backbone: the full description of a Digital Product over its lifecycle.

Other key data objects make up the Service Offer Backbone: the data about the consumption of service delivered by a Digital Product.

Functional components: These define the minimum set of operational functionality essential for effective end-to-end technology management (e.g., Strategy, Product Backlog, Defect, Identity, Incident, and so on). Each is modeled as an Application Function whose job is to control one or more key Data Objects.

The definition of each functional component includes specific functional criteria, which are the minimum operational capabilities, data integrations, and data flows required for the coherence of the full IT4IT model.

Value streams: The architecture describes seven prescriptive value streams (see The IT4IT Functionality Groups and IT4IT Value Streams). These business views of Digital Product lifecycle management are broken down into a series of value stream stages that make use of functional components and data objects to deliver a defined outcome. For each value stream, a set of practical examples is provided in the form of business scenarios.

How do these concepts work together? To achieve the high-automation, large-scale vision of the IT4IT model, the functional components would be software components that participate in a fully integrated automated toolchain – a “Digital Product factory” – that maintains full data traceability from concept to deployed Digital Product and delivered service, and for which all value streams and associated data flows and integrations are automated based on a common IT4IT architecture.

3.2. Top-Down Decomposition of the IT4IT Architecture

The IT4IT Standard is about defining the value stream, the associated functions, and the data objects needed to control the planning, development, delivery, and run management of Digital Products. This can be seen as a factory for digital.

In the rest of this chapter we will outline:

- The groups of functionality needed for managing digital
- The decomposition of the digital value stream
- The necessary underlying functionality and data models

The detailed normative definition of this can be found in Chapters 5 to 10.

3.3. The IT4IT Functionality Groups

Functionality groups provide an abstraction of what is needed to manage the new digital reality in a way that helps to simplify conversations between interested stakeholders. In the IT4IT Standard, we use it to describe a collection of functionality and data that is responsible for a well-defined part of the overall value creation. We describe the groups at a high level, and we decompose one level to have well-defined categories for defining the Supporting Functionality.

The IT4IT Standard defines four IT4IT groups of functions that deliver the outcome for the high-level IT4IT Value Streams. These are known from previous versions of the IT4IT Standard as making up the IT4IT Value Chain:

- Strategy to Portfolio Functions defines the functionality used to manage a digital strategy and portfolio

  This was formerly known as the Strategy to Portfolio (S2P) value stream.

- Requirement to Deploy Functions defines the functionality used to build a Digital Product from its defined requirements

  This was formerly known as the Requirement to Deploy (R2D) value stream.

- Request to Fulfill Functions defines the functionality used to deploy and consume a Digital Product

  This was formerly known as the Request to Fulfill (R2F) value stream.

- Detect to Correct Functions defines the functionality used to detect issues and correct them in order to ensure products run as expected

  This was formerly known as the Detect to Correct (D2C) value stream.

In addition to these four IT4IT functional groups, digital management depends on a set of Supporting Functions. The IT4IT Standard defines six of these in Supporting Functions.

The Supporting Functions are only defined in terms of the minimum functionality they need to deliver in order to support the core of the IT4IT Reference Architecture. Other standards define these capabilities in much more detail and it is not our aim to compete with these standards.

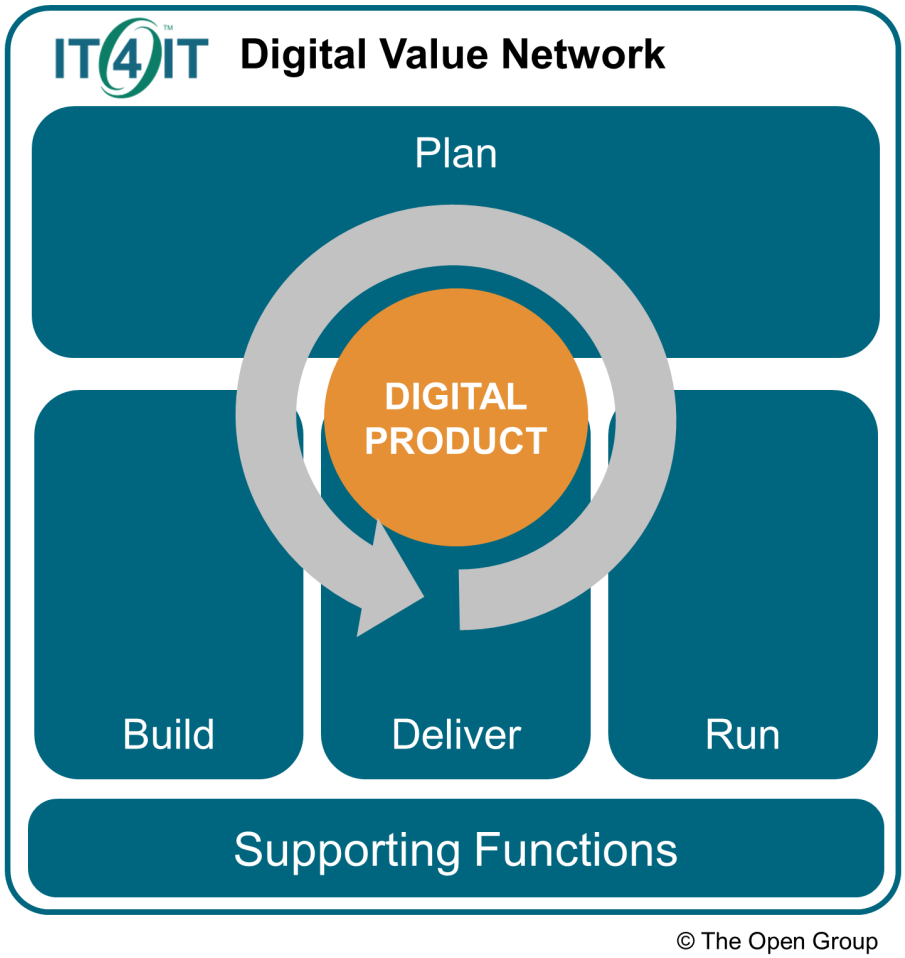

Figure 3-2. The IT4IT Value Delivery Framing Across the High-Level Capabilities

Figure 3-2 illustrates how the capabilities work together, with the IT4IT Value Streams and the associated decomposition of the concept of a Digital Product.

3.4. The Digital Value Network

A traditional model for defining factories is Porter's Value Chain concept [Porter]. This revision of the IT4IT Standard breaks with the Value Chain model and moves to the more recent concept of the Value Network. This reflects that value creation is a complex interaction within many systems and organizations.

In Figure 3-3, it is recognized that in addition to the core value stream for managing digital, a complex interaction with a number of supporting capabilities in an enterprise is needed. These functions are not at the heart of the IT4IT Standard, as other standards and frameworks have these functions and capabilities as a focal point.

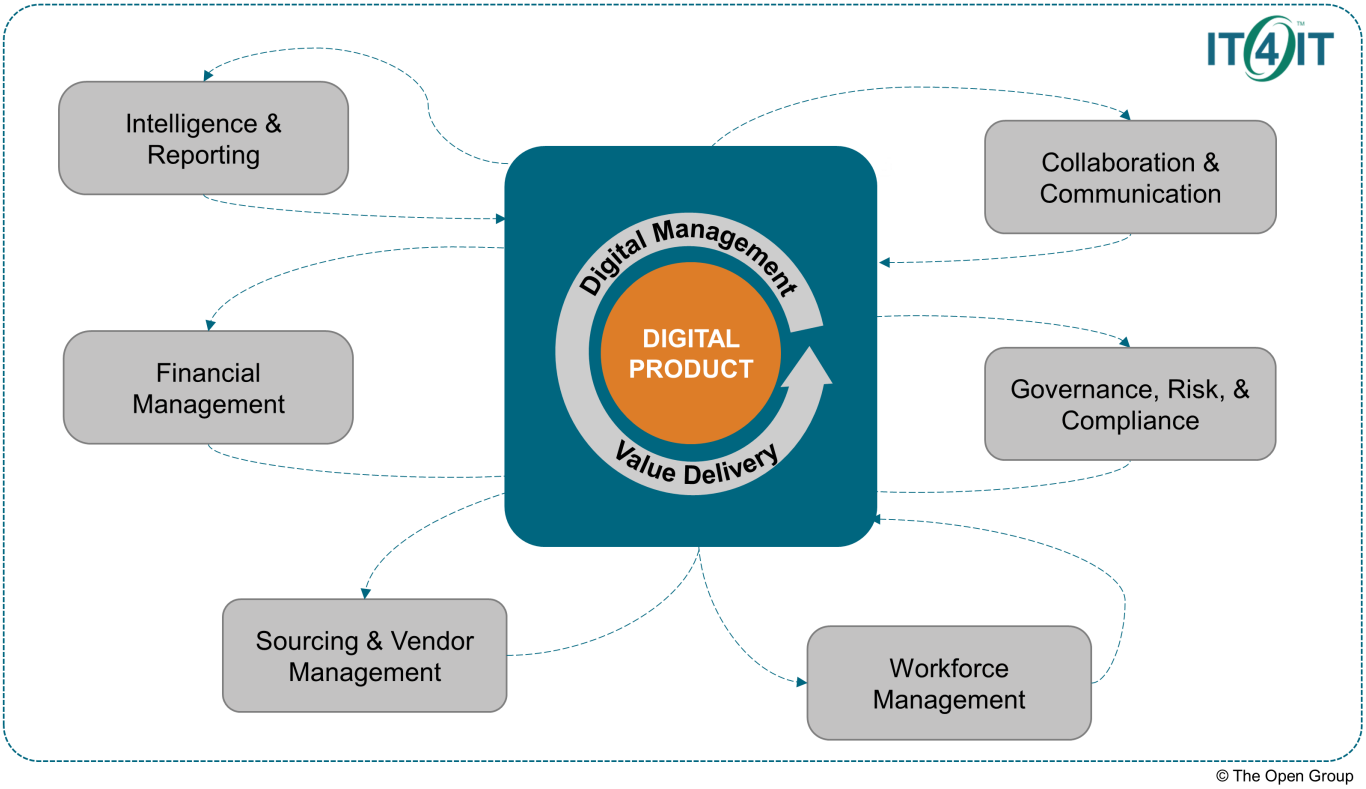

Figure 3-3. The IT4IT Supporting Capabilities

The Value Network is further defined through the interaction of the Digital Product with customers, business partners, vendors, as well as government and regulatory entities. The Digital Product is creating value for its consumers. The business partners and service providers can both provide resources for delivering Digital Products as well as creating their own value using the Digital Products delivered through the IT4IT Standard. As you develop Digital Products you must comply with government and regulatory rules and thus participate in the value creation of society; see Figure 3-4.

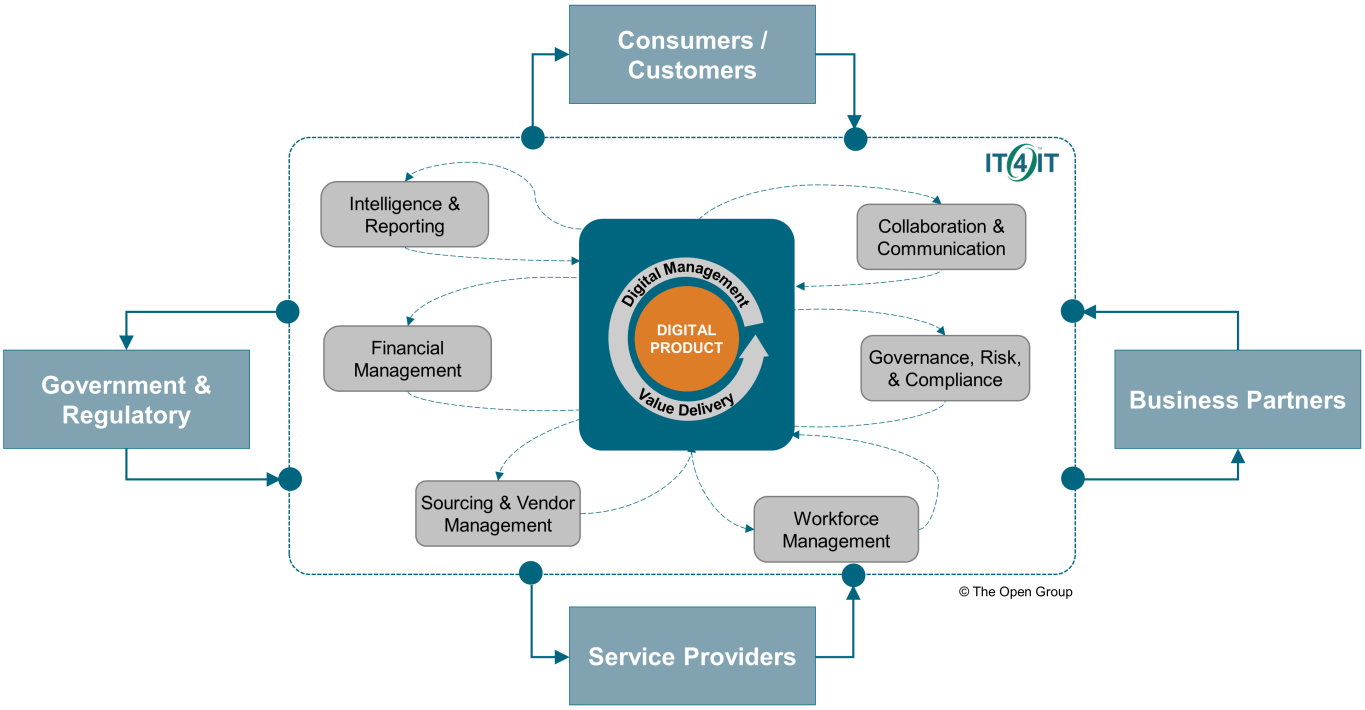

Figure 3-4. The IT4IT Value Network Interactions

3.5. The Seven IT4IT Value Streams

The high-level Digital Product concept (illustrated in Figure 3-4) can be decomposed into four data objects (shown in Figure 3-5) that represent the data managed in each of the four functional groups. A further decomposition is represented by the nine data objects shown in Figure 3-6. Taken together, these nine interlinked data objects comprise the formal concept of the Digital Product Backbone. Each of the three views of the Digital Product – as one, four, or eight objects – are decompositions of the same unifying concept of the Digital Product expressed as data. This is complemented with seven new value streams that, together, explain how to define and manage the delivery of digital value.

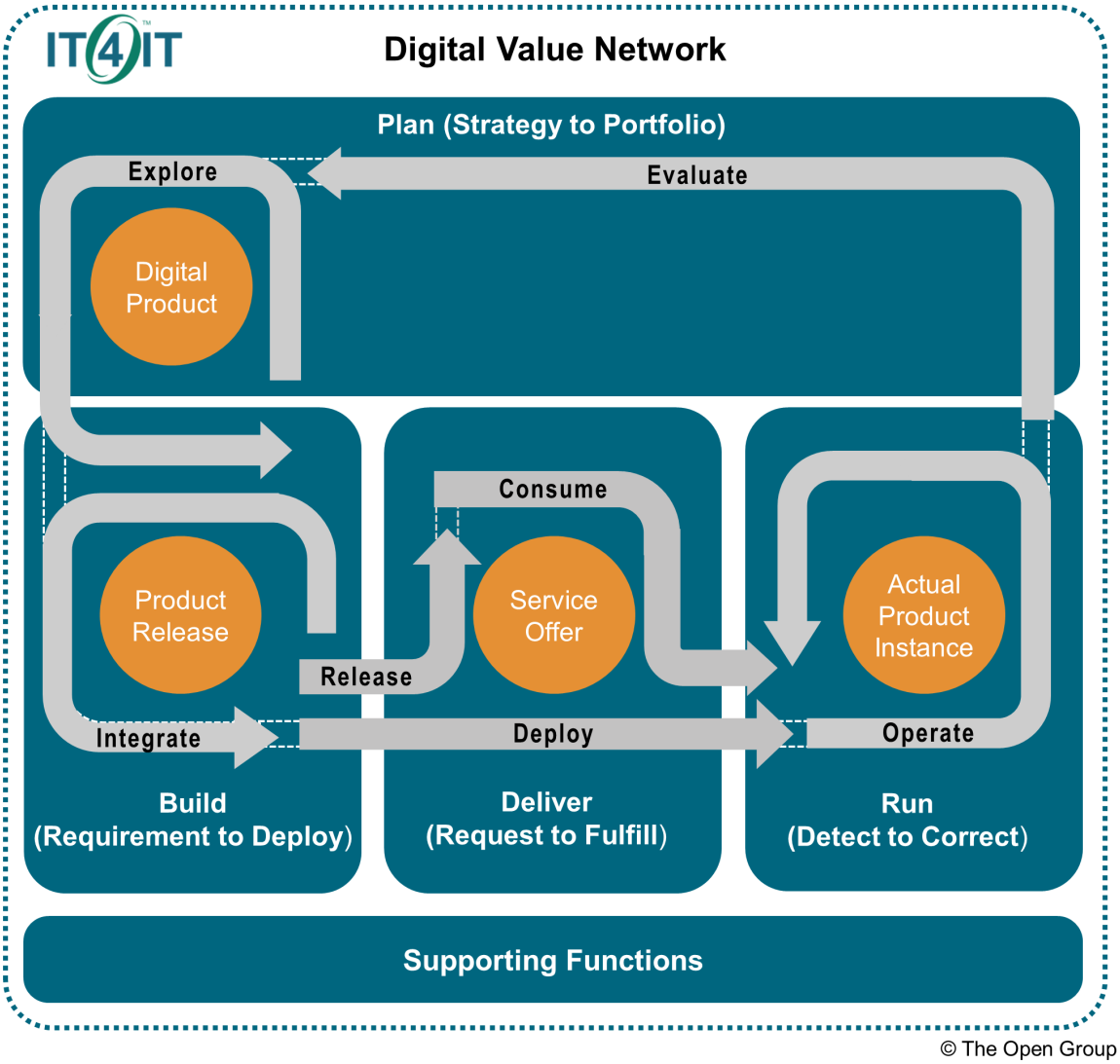

Figure 3-5. The IT4IT Value Streams

A value stream is a representation of an end-to-end collection of value-adding activities that create an overall result for a customer, stakeholder, or end user. For more detailed treatment of the definition of value streams; see the *TOGAF Series Guide: Value Streams* [G178].

As shown in Figure 3-5, the complete set of value streams depicts the various ways in which an organization orchestrates its capabilities to create stakeholder value through Digital Products. The value stream has a direct linkage to an organization's business model (specifically to the value proposition). As an organization translates its business model to an operating model, those value stream stages can be translated into digital delivery processes as well as related data objects, tools, integrations.

This document defines the following seven value streams: Evaluate, Explore, Integrate, Deploy, Release, Consume, and Operate.

These value streams are briefly introduced and described in the following sections.

3.5.1. Evaluate

The Evaluate value stream is about the continuous assessment and evaluation of the entire Digital Product Portfolio to optimize the alignment of the Product Portfolio with the business strategy, including the identification of new Digital Product opportunities that support business and technology objectives.

The Evaluate value stream is described formally in Evaluate Value Stream.

3.5.2. Explore

The Explore value stream continuously explores new features and/or future directions of a Digital Product aligned with strategic direction and business needs. It ensures the Product Design evolves to facilitate innovation and optimize business outcomes.

The Explore value stream is described formally in Explore Value Stream.

3.5.3. Integrate

The Integrate value stream continuously designs and builds new Product Releases and makes them ready for deployment to the market or the business. The Product Release is not limited to software development; this value stream is also applicable for the development of infrastructure building blocks and workplace services. The Product Release may be as large as an entire new product or as small as a hotfix or a patch.

The Integrate value stream is described formally in Integrate Value Stream.

3.5.4. Deploy

The Deploy value stream deploys a Product Release into a production/operating environment (creating a new instance or updating/removing an existing instance). The scope of this value stream includes many different strategies for deploying Product Releases.

The Deploy value stream is described formally in Deploy Value Stream.

3.5.5. Release

The Release value stream provides consumers with digital Service Offers that represent a new instance of a Digital Product or a change (including decommissioning) to an existing instance. A Service Offer defines how to subscribe to a Digital Product and all other lifecycle interactions the instance will have with its consumers, stakeholders, and support. The Service Offer can be made available for consumption through a self-service portal or through Application Program Interfaces (APIs).

The Release value stream is described formally in Release Value Stream.

3.5.6. Consume

The Consume value stream is triggered when an entitled human or system actor accepts a Service Offer related to a Digital Product. The value stream should support customer self-service and is not limited to a single engagement channel. It orchestrates the actions required to fulfill consumption of the offered service to ensure that the desired Digital Product is delivered within the agreed terms.

The Consume value stream is described formally in Consume Value Stream.

3.5.7. Operate

The Operate value stream manages the continuous operations of a deployed instance of a Digital Product in a continuous manner. It is responsible for maintaining availability and performance within the boundaries of agreed Service Contracts. The scope of this value stream includes managing any compliance and security aspects of running Digital Product Instances and underlying systems.

The Operate value stream is described formally in Operate Value Stream.

3.6. Introducing Functional Components and the Data Model

The seven IT4IT Value Streams are defined in IT4IT Value Streams. Each IT4IT Value Stream depends upon an end-to-end flow of work – via data objects – through multiple functional components to deliver the expected result for the customer, stakeholder, or end user.

The IT4IT Standard defines functional components as the application building blocks for managing Digital Products. Related functional components are associated together into four functional groups. A further decomposition is made from the four high-level functional groups into eight (sub-functional groups that must be established in order to manage Digital Products: Strategy, Portfolio, Develop, Test, Consume, Fulfill, Support, and Assure.

While the IT4IT Value Streams orchestrate, consume and/modify multiple data objects in the flow of work to deliver value, we provide a high-level visualization in Figure 3-6 where we highlight only the backbone data objects.

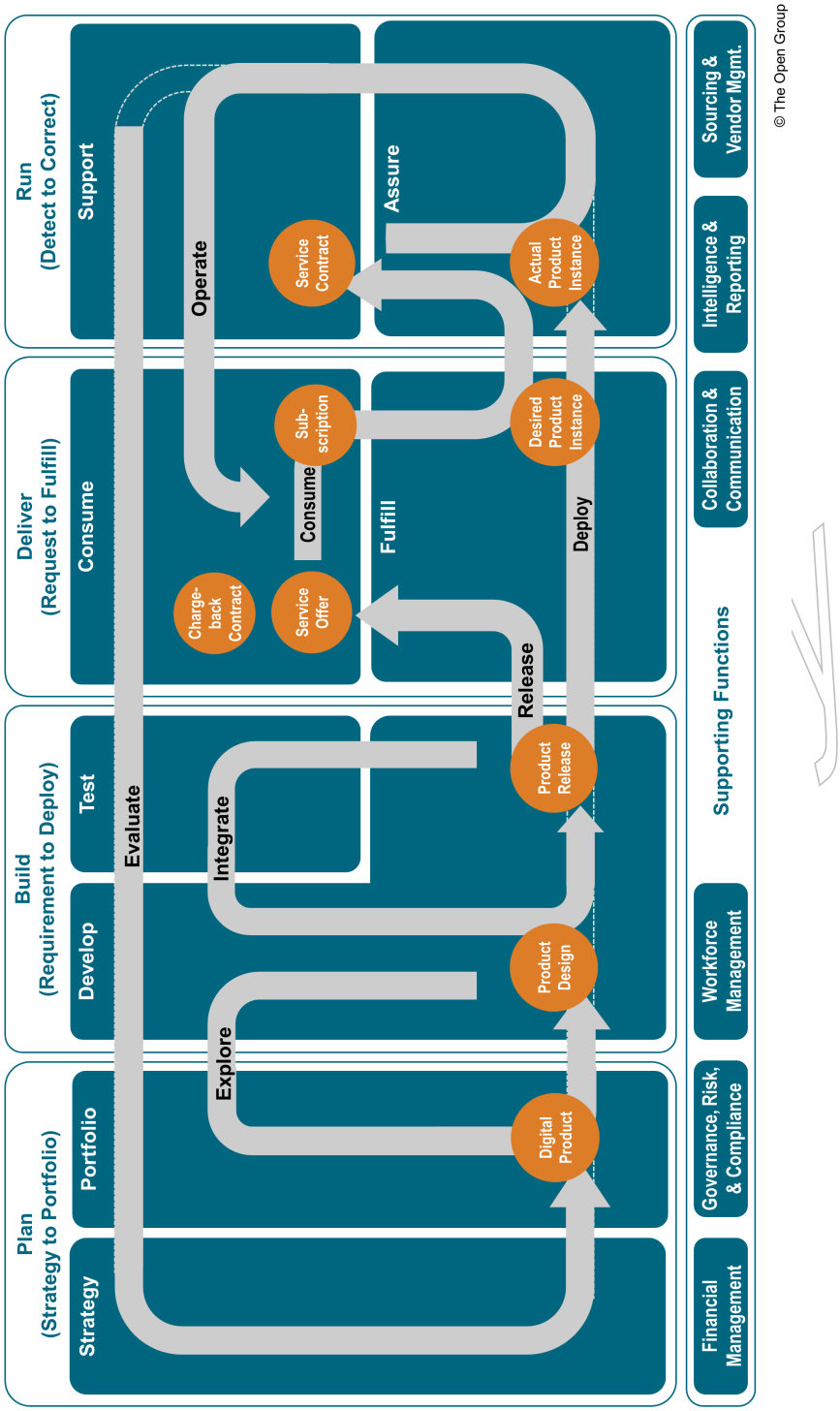

Figure 3-6. The IT4IT Functional Groups, Value Streams, and Associated Digital Product and Service Offer Backbone Data Objects

These eight functional groups shown in Figure 3-7 are then populated by 33 functional components that control 44 key data objects. The 44 data objects are manipulated as the seven value streams are exercised: exploring, integrating, deploying, releasing, consuming, operating, and evaluating the Digital Products. The data objects are related to each other and in Figure 3-7 we show some of the most important relations. For the full list of relations and their associated cardinality see Chapters 6 to 10.

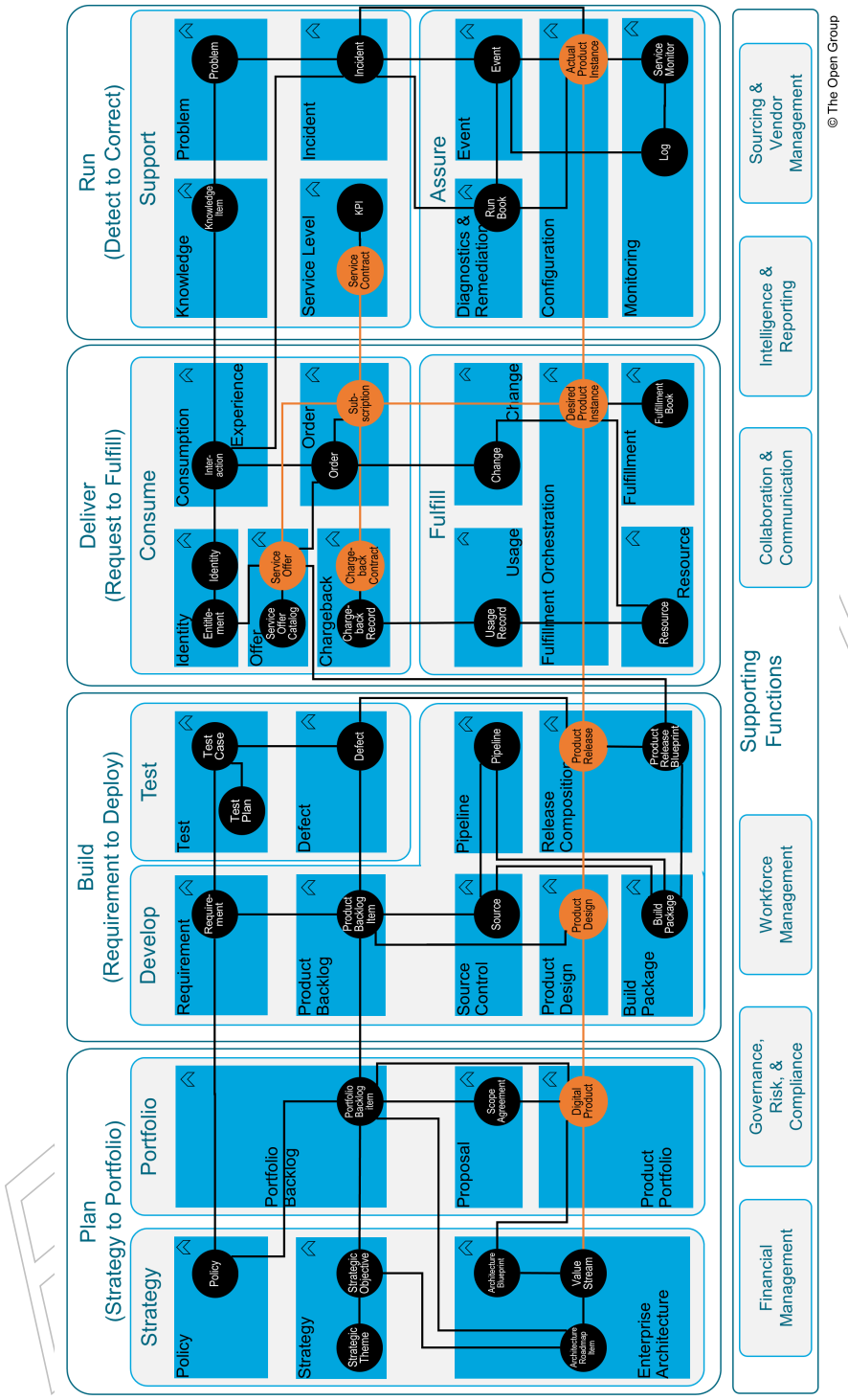

Figure 3-7. The IT4IT Level 1 Functional Diagram

3.7. Concepts Recap

The IT4IT Standard defines seven value streams that manage the Digital Products. Information on the Digital Products is tracked through a set of interconnected data objects managed by functional components. These are the components at the center of delivering the digital management capabilities, as illustrated in Figure 3-8.

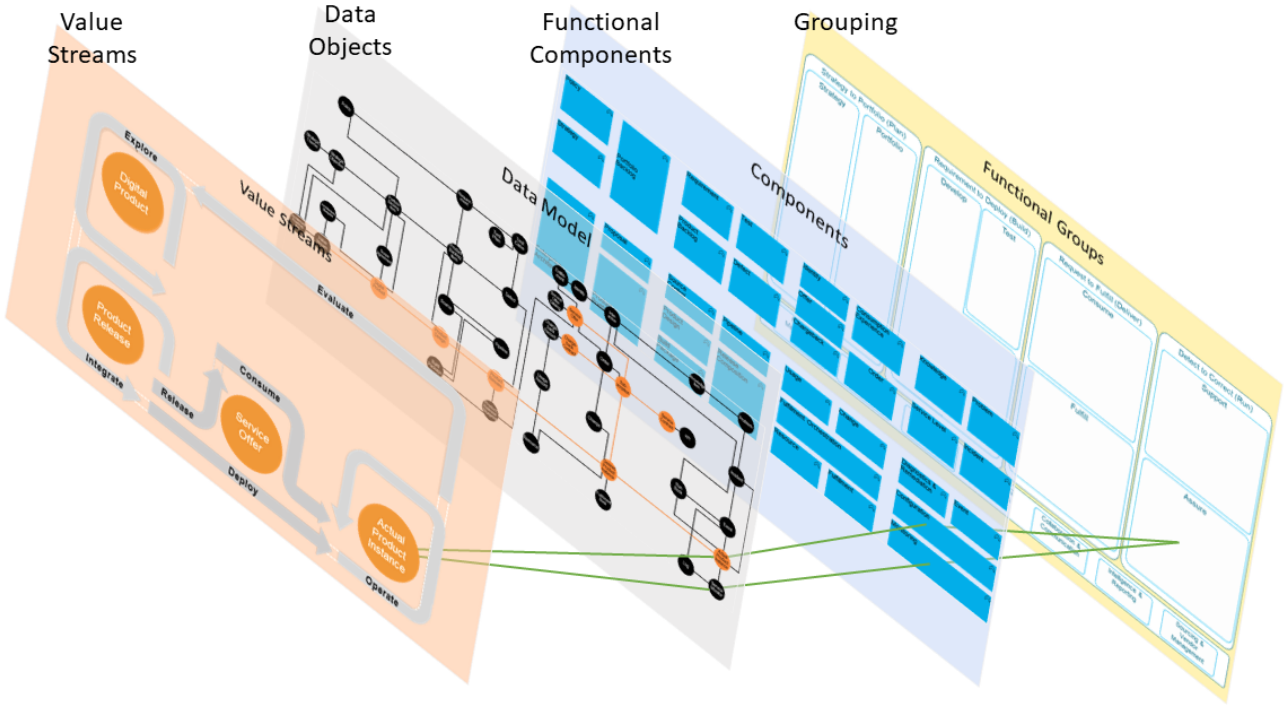

Figure 3-8. The IT4IT Concepts and their Relationships

Combining the value streams with the four high-level functional groups results in The IT4IT Value Streams. Combining the value streams with the eight sub-level functional groups results in The IT4IT Functional Groups, Value Streams, and Associated Digital Product and Service Offer Backbones. Combining the data objects with functional components and functional groups results in The IT4IT Level 1 Functional Diagram.

For details on the formal object types, metamodels, and associated ArchiMate solution patterns used in IT4IT Concepts and Metamodel, see Figure 3-9.

Figure 3-9. The IT4IT Simplified Notation for the IT4IT Model

Chapter 4. Digital Product

This chapter describes the Digital Product: the single, simple, unifying entity referenced over the full management lifecycle and across all the value streams in the IT4IT Standard.

The rationale behind the Digital Product-oriented thinking is The Open Group White Paper *The Shift to Digital Product: A Full Lifecycle Perspective* [W205], where a proposed Digital Product definition, with examples, was initially documented and a proposal was made to evolve the IT4IT Standard in order to better address its full lifecycle management intent with a product-oriented technology management model, which replaces the traditional practice of viewing IT investment as a combination of project and run costs spread across loosely-coupled silos.

Managing the Digital Product Portfolio becomes the focal point of all IT investment. The Digital Product owner becomes accountable for all value-creation activities associated with a product, from funding through support activities.

This approach integrates traditional Product Management acumen with IT management disciplines. It applies to internally and externally consumed digital offerings. In Enterprise Architecture terms, the consumer of a Digital Product is an actor – individual, organization, or machine.

Only the most granular Digital Products are simple: what seems like a single product to the consumer will usually be made up of a recursive array of sub-products.

The Digital Product is a unifying concept that aligns end-to-end management activities ranging from business strategy through Product Management, development, consumer interactions, and operational support. It promotes cooperation across the disparate roles needed to support cross-functional automation and scaling up of Lean, Agile, and DevOps approaches to software delivery. Team members can better understand what they are working on, and what consumers they serve.

The integrated data objects that comprise the Digital Product Backbone provides the full end-to-end description of a Digital Product. The backbone ensures traceability through the lifecycle, which is required to understand all phases of managing the Digital Product, its impact on consumers, and how consumer experience is traced back to initial investments; see The IT4IT Value Delivery Framing Across the High-Level Capabilities.

4.1. Merging “Application”, “Service”, and “Products”

In the IT4IT Standard, Version 2.1 [C171] and in typical IT Service Management (ITSM) practices, the term “service” has been used to describe the core consumable output that the IT organization will manage throughout its lifecycle.

“Application” is a common term that overlaps “service” in common IT usage. The word “product”, often used to describe the same concept, has also become popular in Agile practice to reference development teams’ “code” delivery.

While acknowledging the ambiguity arising from choosing any of these terms, a single, simplified approach using one abstract unifying term, “Digital Product”, is used to clarify the intention of how the term “service” was used in previous IT4IT versions, and to consolidate the management aspects required for “application,” “service,” and “product” as described above.

The term “Digital Product” aligns well with recent trends and business semantics (such as those documented in DevOps and Agile practices and illustrated in IT management “Project to Product” directions and associated shifts in Financial Management), and with reorganizing the work of separate silos into single cross-functional teams.

Product Management disciplines have become a focus for traditional IT organizations, and in startup Digital Product teams that need to scale up as they grow. In both scenarios, the IT4IT Standard provides value. Product Management is a proven, mature competency with well established roles, activities, frameworks, lifecycle management approaches, and so on. Adopting “product” as the governing metaphor within the IT4IT Standard brings this acumen to the fore. The IT4IT Standard does not formally define the role of a Product Manager; instead we focus on defining Digital Product in the context of the main value streams, functional components, and data models required to manage them.

The IT4IT Standard is part of a solution set that supports Digital Transformation, connecting proven management practices with newer methods. For many successful organizations, Product Management and IT management roles will converge and unite into a new competency we call “Digital Product Management” a fundamental capability for managing digital.

4.2. Digital Product Definition

The Digital Product definition used in the IT4IT Standard highlights the distinction between Digital and non-Digital Products and provides clarity about the use of the term “digital” throughout the rest of the IT4IT Standard.

This definition flows from widely accepted definitions of both “digital” and “product”, brought together to form a unique type of product that incorporates IT management. Merriam-Webster defines *digital* as “characterized by electronics, especially computerized technology”.

The Business Dictionary defines *product* as “a good or service that most closely meets the requirements of a particular market and yields enough profit to justify its continued existence”.

The Economic Times defines *product* in more detail as:

> “A product is the item offered for sale. A product can be a service or an item. It can be physical or in virtual or cyber form. Every product is made at a cost and each is sold at a price. The price that can be charged depends on the market, the quality, the marketing, and the segment that is targeted. Each product has a useful life after which it needs replacement, and a lifecycle after which it has to be re-invented.”

A “Digital Product” can be defined as:

> “A service, physical item, or digital item that provides an agreed and specific outcome for a consumer; that incorporates and requires software to realize that outcome; that is expected to require active management of the software and its required resources over its lifecycle, in a manner prescribed by the provider; and that is described by a formal offer of the outcome to be provided in exchange for an explicit price.”

The criteria of a Digital Product:

- A Digital Product **must** include one or more Service Offers which define Service Contract options for consumers
- A Digital Product **may** be delivered as a Digital Product Instance (defined below) as described in the Service Offer
- The Digital Product Instance **may** include a system containing IT, non-IT resources, and software
- A Digital Product **may** be consumed within an organization or externally
- A Digital Product **may** have dependencies on other Digital Products
- A Digital Product **may** be comprised of other Digital Products
- A Digital Product **may** be a resource for other Digital Products
- A Digital Product **must** provide interactions via machine and/or human interfaces

A Digital Product refers primarily to the perspective of a Product Manager and represents a complete product line over its lifetime, to include product variants, types and locations of consumers, annual and lifetime financial models, dependencies on other products and services, supplier relationships, capacity, service-level options, distribution models, pricing and usage rules, delivery and management of Digital Product Instances, and so on.

4.2.1. System Definition

The system comprises those technology components of a Digital Product Instance that will be managed by the provider.

Software, compute, and networking hardware are the most common system resources integrated into Digital Products. This may include custom and externally sourced software that may change frequently through Agile processes.

The system recursively includes any supporting resources upon which the system relies, including other products. Resources might be as simple as a few intangible lines of code to be deployed on hosted infrastructure in the cloud, or might be comprised of a large and complex number of components.

4.2.2. Service Offer Definition

The Service Offer (sometimes shortened to Offer) describes possible Service Contract terms as perceived by the potential consumer of a Digital Product Instance. The Offer must include all the elements that will become part of the contract if the Offer is accepted.

The Offer formalizes the value statement and contract terms for consumer/provider interactions. The promises articulated in the Service Offer are recorded in a Service Contract as part of the agreed criteria for how the system will be managed and supported throughout its lifecycle. Longer-term contracts may be defined as subscriptions that track ongoing consumption and billing.

The Offer is derived from the Digital Product definition, which includes all available consumable variants, and all possible contract terms. A product may in theory provide as many unique Offers as there are prospective types of consumer.

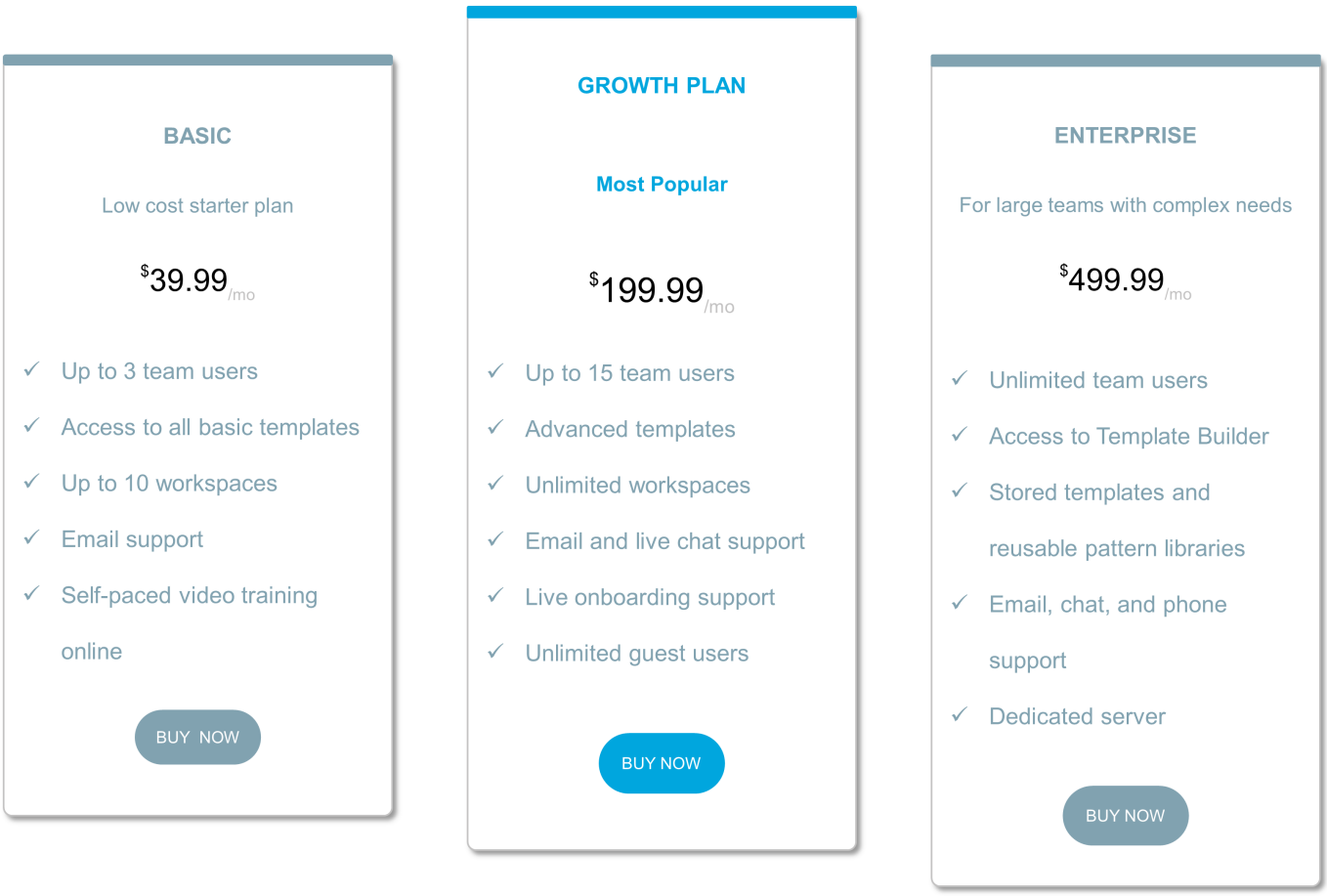

Figure 4-1. Pricing Table

A consumer can either be a human or internal employee actor, a paying customer, or alternatively a machine actor such as another Digital Product. The structure of both the Offer and the Service Contract may vary significantly based on the type of actor involved, and how a particular product is to be consumed.

4.2.3. Contract Definition

When an Offer is accepted by a consumer, an Instance of the Digital Product is created in which the relationship between provider and consumer is captured in a contract based on the relevant Offer. The contract includes all the details required for financial recording, governance, measurement, charging, support, and servicing of the Digital Product Instance.

A contract for a Digital Product may include a subscription that describes the ongoing usage rights for a consumer, with periodic charges.

For example, a Wi-Fi network in a public place is made available to consumers. When a consumer logs on to the network using the credentials supplied by the network owner, a Digital Product Instance is created. The product is not the Wi-Fi network itself. The product is the provision of access to the network and its resources. This product incorporates other Digital Products including, at a minimum, the Wi-Fi router. The contract may be implicit or explicit, and will likely be governed by a range of legal and technical constraints, all of which form part of the contract.

In its simplest form, a Service Contract may record a single transaction that is not normally expected to have recurring charges, usage charges, or to require support. However, even such a one-off transaction will typically include mutual commitments to rules and metrics about possible future circumstances governed by the contract terms, such as regulatory compliance, warranty, repair, refunds, Service-Level Agreements (SLAs), consumer support, product recalls, offers on related items, commitments to keep the software up-to-date, and so on.

The contract structure includes Offer terms as agreed upon for the Digital Product Instance. The interaction process described in the Consume value stream addresses all the concerns for this interaction with a consumer, to include the following:

- Warranties and SLAs
- A record of the transactional commitments made by both parties

  For a consumer Digital Product sold online, this may be an identifier for a series of web page interactions in which terms are accepted and money is taken. For an internal machine-to-machine contract, this may be the digital record of a “handshake” acknowledgement between two systems.

- Usage and financial rules, especially where there is a recurring financial payment

  In the IT4IT Standard, this contract data forms the basis for managing chargeback/showback, and penalties for the breach of SLAs.

- A description of the usage monitors needed for governing the Digital Product Instance to ensure operational status

  Depending on the relationship between the provider and the consumer of the product, this may include legally binding terms with provisions for events such as returns, subscription renewal, refunds, warranty claims, recourse for breach, and so on.

- A right for the provider to audit the consumer for such things as license compliance audits or conformance to usage rules
- Long-term ownership rights to physical components or assets
- Lifecycle management expectations such as support, updates, replacement, or retirement

4.2.4. Price Definition

Each Service Offer and Service Contract describes a price which may require direct payment by a consumer through the Chargeback Record and may include an ongoing subscription.

Price need not be related to cost and is merely relayed to the customer as showback. In some cases, the price may consist entirely of indirect benefits anticipated by the provider with no financial charge at all.

The element of chargeback and showback at a minimum is meaningful for effective reporting and Product Management. In every offering, a Product Manager should be able to describe the Return on Investment (ROI) over the product lifecycle. Offering a Digital Product Instance with no direct price should always be a deliberate pricing decision that is explained in the product's lifetime financial model.

In cases where the Digital Product is perceived by the consumer primarily as software, it will always include other components defined by the Product Manager, including those required by the business or legal context of product delivery, as well as the provision of a supporting infrastructure.

These other associated components – which may not be explicit in the Offer or Service Contract – may include hardware in which software is embedded (disk drive, smartphone, Internet of Things (IoT) doorbell, automobile). They may also include contractual elements (SLAs, chargeback terms, warranties, software maintenance commitments, customer support).

4.3. From IT Service to Digital Product

In the IT4IT Reference Architecture, Version 2.1, the Service Model Backbone represents the full lifecycle of an IT service, which had previously included services from both a labor and a digital perspective. The Digital Product, however, changes this focus to address technology or code as the main actor in the delivery of an outcome of a service, removing those services performed by human labor.

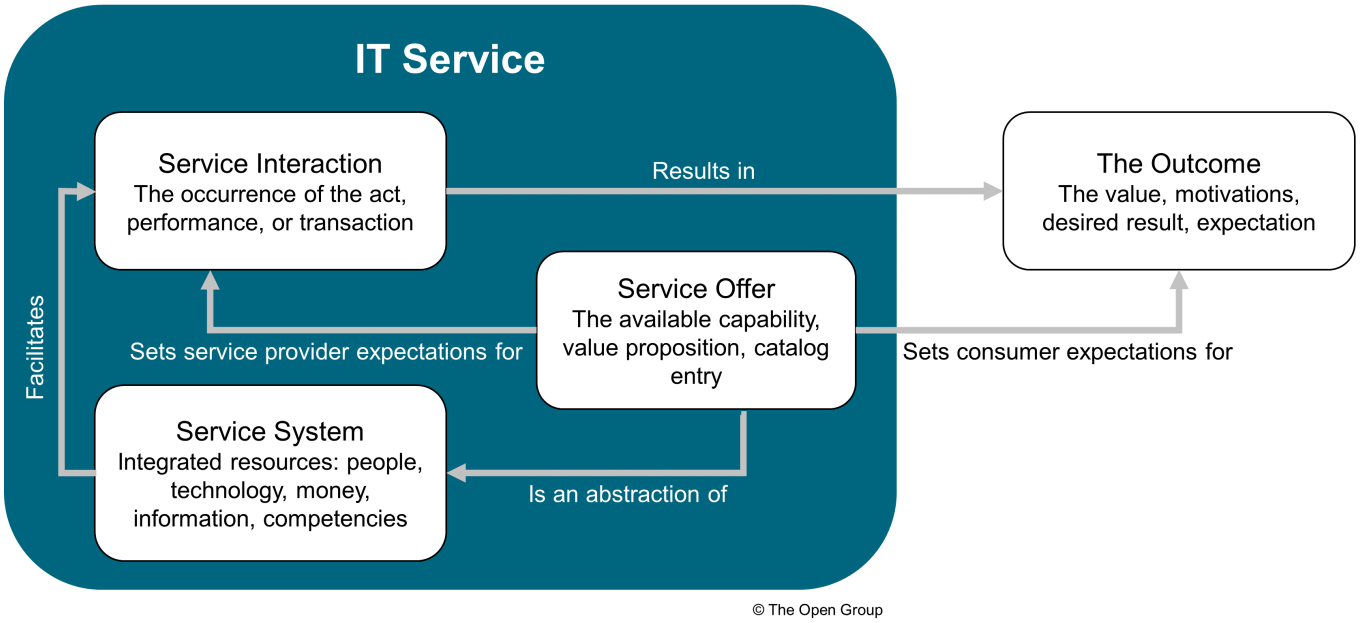

Figure 4-2. Original IT Service Model

Figure 4-2 describes service relationships and interactions from the IT4IT Reference Architecture, Version 2.1. This conceptual view has evolved in this document with a redefinition of the service to a Digital Product concept, exclusively addressing digital type services. In Figure 4-3, the IT service model has been refactored by:

- Replacing “IT Service” with “Digital Product” as the main entity managed across the IT Value Network
- Narrowly defining Service System to focus on technical and data components that deliver outcomes
- Changing “Service System” to simply “System”

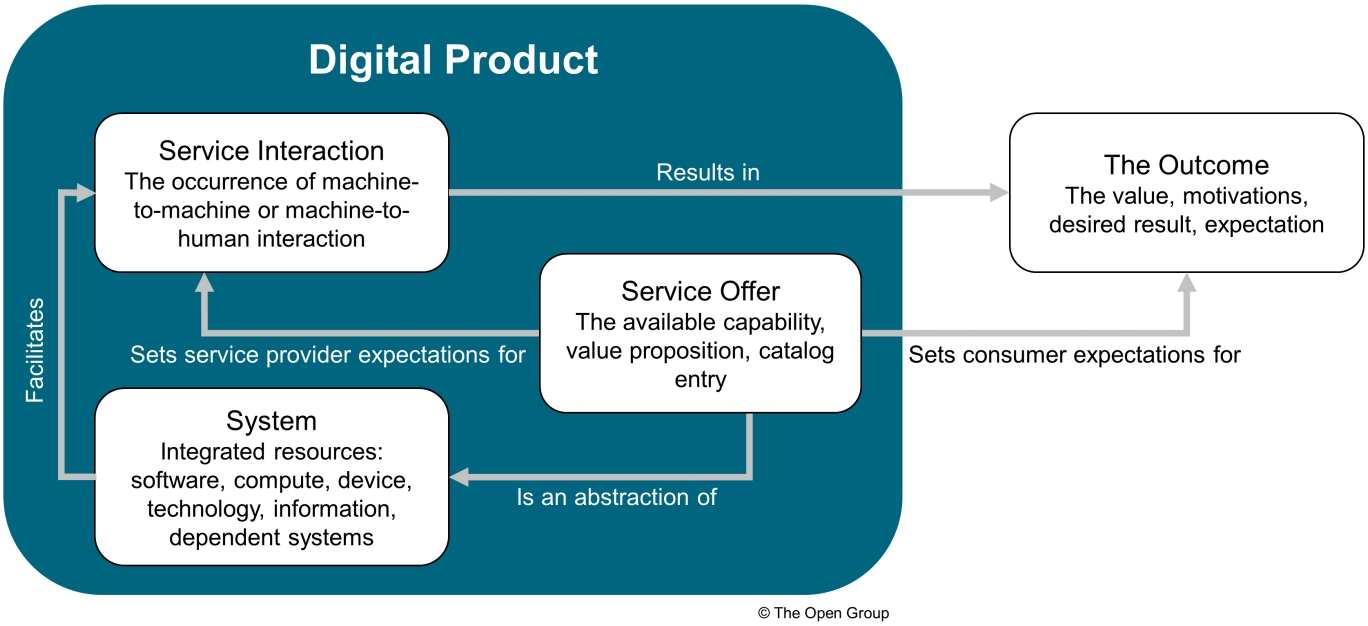

Figure 4-3. Digital Product Interaction

Figure 4-3 provides a view for how the Digital Product's software and supporting technical components interact with consumers, following practices described in the IT4IT Value Streams and functional components:

- The **Service Interaction** is a process or event, and depicted in the Consume value stream
- The **System** is a set of configured software, compute, devices, technology, information, and subscriptions to dependent Digital Product systems, as depicted in the Actual Product Instance
- The **Service Offer** is a description of the service consumable, based on product rules and in some cases may be dynamically created as described in the Offer functional component

Figure 4-3 can be used as a template for the instantiation of a Digital Product Instance, in which a specific Service Offer becomes the basis for a Contract, and a specific subset of system resources is allocated per the Offer and supported in the Operate value stream for the term of the Service Contract.

4.4. Examples of Digital Products

The examples listed here are consumer products that are easy to recognize. However, the majority of Digital Products are primarily designed to be incorporated within another product. When thinking about a Digital Product we need to recognize that the critical boundary of a product is what a supplier provides to the consumer.

4.4.1. eCommerce Websites

A website that provides functionality for shoppers and vendors to reach a commercial agreement is a typical Digital Product. The consumer of this Digital Product would be the vendor's Sales Department(s), not the shopper.

The eCommerce website may be comprised of several Digital Products. These may have different functionalities to support other users, such as those who maintain site data or manage financial transactions. The different functionality may also be different Digital Products.

4.4.2. Mobile Applications

Mobile applications are Digital Products. Most are provided through a store managed by the digital platform provider. They may also be linked to other products, physical and digital. Fitness mobile applications typically require a separate physical device to track motion and location. Other mobile applications act as channels to Digital Products, such as a newspaper subscription or online banking.

Digital Products are often comprised of other Digital Products, or are dependent on other products to provide a complete outcome.

4.4.3. Operational Technology

Digital Products can also automate manufacturing, warehouses, or other activities outside of typical back-office applications. This more industrial and embedded use of automation technology is commonly known as Operational Technology (OT).

For example, applications are now typically used to manage wind farms, monitor and adjust factory equipment, create tasks for planned and unplanned maintenance, or change operational characteristics due to weather events and forecasts. There may be a network connection to support communications and software updates to be applied on an ongoing basis, much in the same way data centers are managed.

In factory automation, there may be dozens or hundreds of smart OT devices with embedded computers, for which software may be updated periodically to update code and change configurations. The possibility of such a continuous improvement feedback loop means that software design can be more Agile and at the same time more complex than the design of traditional embedded systems, for which firmware updates may be expensive and complicated, or even impossible.

4.4.4. Smart Devices with Digital Interfaces

This is a rapidly growing category of internet-connected products ranging from smart watches to automobiles that combine a traditional physical product format with software to create Digital Products. These smart products do not contain only the physical device and its embedded software; the core product definition may also include cloud-based software and software running on other platforms, such as smartphones. For example:

- A digital oven that can be programmed by scanning a barcode on a food package using a mobile app, providing the oven with cooking instructions via a cloud-based cooking instruction look-up system; alerts can be sent to a smart watch to inform the user that cooking is complete
- A modern car, containing more lines of code than model desktop operating systems, for which updates are routinely pushed from the car manufacturer, and which can be monitored and controlled from miles away using a smartphone app

  The product definition includes software in the car and the app, software elsewhere in the supporting ecosystem, and all the mechanisms needed to keep the software up to date.

Figure 4-4. Digital Watch

4.4.5. Digital Platforms

An important category of Digital Product is the digital platform. This product is designed to enable the creation or hosting of other Digital Products which can be sold or shared with consumers.

Common examples of how this works in practice include the app stores available to Apple® and Android™ customers, or the Azure® and Amazon Web Services™ platforms that provide a range of easily orderable plugins, and also the ability for users to create their own.

There are many types of platforms that use the same model:

- Platform as a Service (PaaS) raw compute and virtual compute
- Software as a Service (SaaS) platforms accessed in the cloud
- Low code/no code development
- Content platform, such as for music, videos, and documents
- Digital Services, such as connecting drivers, passengers, and deliveries
- Software applications that rely on shared platform capabilities, such as smartphones, TVs, and platforms built for Sales, Human Resources (HR), or IT Management

Many digital platforms incorporate shared resources, such as platform-specific development tools, an integrated store for configuring or adding features through plugins, platform-dependent application management, self-service interactions for purchase, and knowledge and support features.

4.4.6. Interplay Among Digital Products

The digital oven example in Smart Devices with Digital Interfaces illustrates an accelerating trend in which collaboration between interacting Digital Products adds greater value than when they function independently.

The oven example of an IoT-based product reflects marketing collaboration among multiple Product Managers of the oven, the watch app, the phone app, and the food-cooking instruction suppliers. All this functionality is embodied in software created by each company involved and supported by

interoperability standards such as *O-DEF™, the Open Data Element Framework, Version 2.0, a standard of The Open Group* [C202] and the *Open Messaging Interface (O-MI), The Open Group Standard for the Internet of Things (IoT), Version 2.0* [C19E]. The result is a value-added ecosystem which others can join – or from which competitors can be excluded.

This interplay creates opportunities and data for all parties. The data and ongoing consumer feedback received helps the Product Manager to make decisions such as adding features, improving performance, adjusting pricing models, and updating market strategies.

4.5. Granularity and Dependency of Digital Products

A Digital Product may be composed of, or relies upon, other Digital Products. For example:

- An app that offers room booking or transportation may rely on a cloud-based Digital Product providing mapping-as-a-service
- A web or mobile app-based streaming video service may rely on a separate platform of web service products to manage and provide streaming content

The network of dependencies of interconnected Digital Products may be deep, and may form very complex systems.

Each component in the system may be another Digital Product. Each Interaction may have a Service Offer, though sometimes they are not well defined.

4.5.1. Examples of Digital Product Granularity

Consider an insurance policy (a type of Digital Product) that relies on software to calculate premium charges.

Case A
The insurance company uses a third-party software package and signs a license agreement allowing use of the software. The agreement is a contract for a Digital Product Instance and defines the exact details of how the insurance company can use the software, including pricing and payment terms. The contract will also include vendor guarantees of calculation accuracy and system performance, and may define penalties and liability for failures. Information about this contract should be easy to incorporate into the design of the insurance product to meet stipulations of compliance and audit, consumer needs, and warranty. The cost of the calculation system's software license – as well as risk management for possible errors – will be a factor in setting valuation and pricing rules for the insurance "book" that represents the affected product line.

Case B
The software used to calculate premiums is created, managed, and supported by a separate in-house department. Is it a “product”? Should it still be managed as a Digital Product? If not, will there be the same confident understanding of long-term cost and risk for this software over the life of the insurance “book”? Does using in-house software make it any less important to track the cost of the calculations, to define and manage expected performance, or to analyze the risk and liability of incorrect calculations?

4.6. Benefits of Formalism between Internal Digital Product Teams

Some Digital Products are expected to be shared and used broadly as components or dependencies in more complex Digital Products. Microservices architectures are a common example. For a Digital Product that is an “internal” sub-component or dependency of another Digital Product, some formalism in understanding pricing and contracts helps in three ways:

1. It may create an objective basis for understanding financial outcomes and value for the sub-component products

   Increased formalism and consequential measurement practices provide continuous visibility of the operational and financial data needed to support full lifecycle Product Management, providing the same level of insight and routine control provided by Enterprise Resource Planning (ERP) systems or by cloud platform providers.

2. It may provide the data collection processes needed to understand internal products by the same standards used for externally sourced products

   This can result in a more objective comparison of options. In some cases, an internally sourced product may be a candidate external product in its own right. Alternatively, an apples-to-apples comparison of well-defined internal and external products could provide the case for replacing internal products with more viable external options.

3. It may support long-term planning of the product lifecycle and the management of technology risks

   Roadmapping and investment planning can become more proactive and long-term (as opposed to reactive and *ad hoc*). The use of automated reports can reveal concerns about the full stack of components such as end-of-life, security vulnerabilities, and compliance.

4.7. Managing the Digital Product

Digital Product Portfolios are managed from three perspectives, as depicted in Figure 4-5:

- **Market-facing:** primarily for use by external customers
- **Internal-facing:** primarily to improve employee experience or productivity; may replace employees with fully autonomous processes
- **Foundational:** a combination of both

Jeanne Ross et al. made these perspectives clear in *Designed for Digital: How to Architect your Business for Sustained Success* [Ross et al.]. In this book, the authors explain that there are two areas in which organizations should focus their Digital Transformation:

1. Digitizing internal processes that improve employee productivity and experience
2. Creating digital offerings for the market to improve customer experience

Figure 4-5 captures these two perspectives as subsets of the overall Digital Product Portfolio. A third portfolio captures those shared, foundational Digital Products used in both contexts.

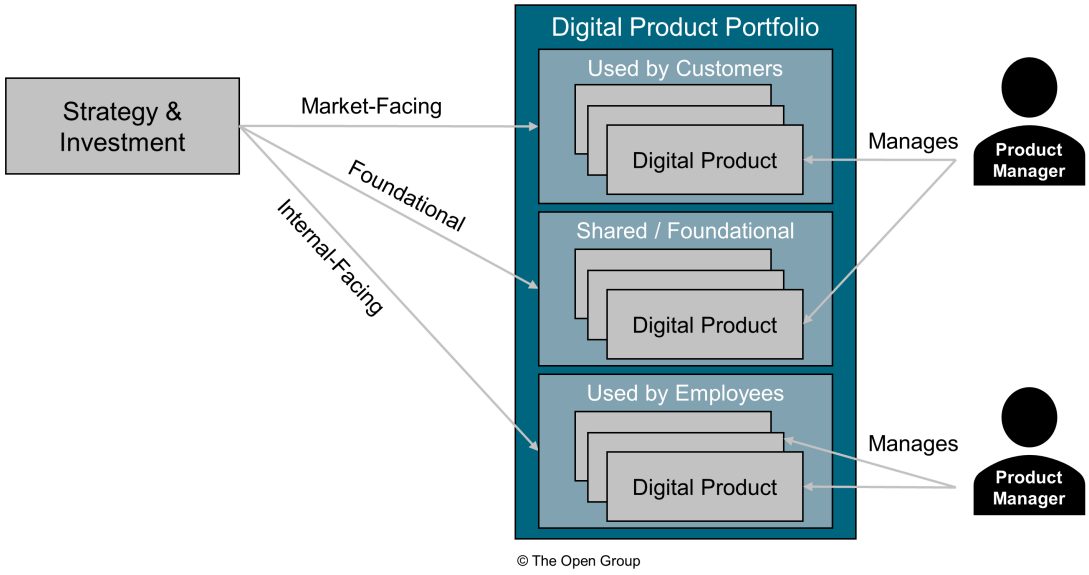

Figure 4-5. Digital Product Portfolios

Directly linking Digital Product Portfolios to Strategy in this way provides clarity on investment priorities and should reduce bad outcomes in competition for funding. Management agility is improved when analyzing costs, and investment decisions become strategically aligned.

4.8. The Digital Product Management Competency

A career and competency of “Digital Product Management” has emerged in which two distinct professional competencies are combined:

- **Technology competency** based on a functional understanding of relevant technologies, including a meaningful understanding of how those technologies are created and managed over time
- **Product Management marketing competency** that includes an understanding of markets, consumers, distribution, logistics, customer value propositions, and financial aspects of products

Digital Products inherently rely on technology resources and infrastructure such as networking, computers, databases, and software to realize value, each with their own lifecycle and contract and/or subscription.

4.9. Shared Resources

Digital Products may have shared resources, such as people, marketing, and lifecycle processes assigned to them. Digital Products may have tightly integrated, shared resources dedicated to facilitating delivery, support, and lifecycle management processes for systems and consumer interactions. Shared functionality could include self-updating apps, in-app support processes, in-app usage monitoring, and update processes within a smart device, and so on.

4.10. Digital Product Lifecycle Concerns

Digital Product Management must address two distinct lifecycle concerns that map to various IT4IT Value Streams (see Figure 4-6):

- **System lifecycle** (the managed IT component of the product)
- **Contract lifecycle** (governs consumed instances, warranties, commitments, and dependent systems)

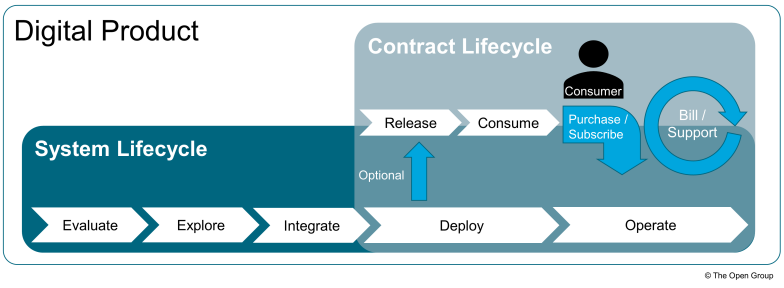

Figure 4-6. Digital Product Lifecycle Management

The system lifecycle is a familiar concept in IT management. For example, an internal application system remains a subject of active concern for as long as it is in use.

After such a system is decommissioned, we will quickly lose interest in most of the hardware and software. We still must address any residual data retention and disposal requirements, sometimes for many years. This is included in traditional understandings of technology lifecycles.

The contract lifecycle for a Digital Product may be more complex, especially regarding residual or implied lifecycle management concerns for external consumers or for a product that is a sub-unit of another product:

- The lifecycle of the contract depends on the existence and management of the system used by the consumer

  The underlying system(s) should not be retired until all contractual obligations are met.

- The contract lifecycle concept depends partly on maintaining enough information and related capability to ensure that the contractual obligations for every consumer are fully discharged

As our societal understanding of digital rights evolves this may include obligations defined after the fact by legislation or court cases.

When the provider ends a contract, thus ending support for the underlying system and explicit support or warranty obligations, terminated consumers may be unable to retrieve critical data, to migrate to alternative systems, or to maintain the system. In these situations, consumers have banded together to continue to support Digital Products long after support from the provider has ceased.

For example, modern farming relies on Global Positioning System (GPS) modules supplied by tractor manufacturers that are used to ensure crops are planted or cultivated in straight rows. Data generated by the system has a financial value to the farmer. Ongoing updates and patches to the software are essential to ongoing product value.

Such a tractor is a Digital Product for which the contractual context can be murky, and this gives rise to litigation and pressure on legislatures about data ownership, rights to use the data, and the “right to repair”:

- Can the manufacturer keep the GPS component design and code secret, so that no one else can service the product?
- Who owns the data created while the product is being used?
- What happens if the manufacturer discontinues support – can the farmer be forced to buy a new module or even a whole new tractor to stay in business?
- Can consumers who band together to continue to support Digital Products after support has ceased insist on being given enough information to be able to do so?

The manager of a Digital Product must consider end-of-life scenarios such as these and provide consumers with suitable recourse in end-of-life situations. Managing contract risks such as the pressure for “right to repair” becomes part of the Product Management function.

4.11. Code, Dependencies, and Instance Resource Management

Until a system is retired, its underlying resources and dependencies must be managed until no active consumers depend on it and all instances have been retired, and possibly longer, depending on factors such as regulatory requirements for data retention.

To maintain the operational integrity of a Digital Product, the code, hardware, third-party software (each has their own support lifecycle), and integrations must be managed. Often, dependencies have their own upgrade and maintenance cycles that should be tracked as well.

Digital Products rely on a supply chain that is often real-time: code-executed and data-driven functionality. The Product Manager is accountable for ensuring that dependencies with other Digital Products are managed along with any contractual, financial, and technical requirements.

The rapid cycle time for changes in Digital Products requires a level of automation not needed for more traditional products. Many products receive software updates weekly or even daily. This can only be managed using an automated toolchain and support process. The IT4IT Standard describes how this can be done.

4.12. Data-Driven Opportunities and Concerns

Data, both generated by and used within a Digital Product, presents unique opportunities and advantages, as well as risks if improperly managed. Data can be collected from a wide range of sources to include the consumer, external sources, automatic capture through embedded sensors, or related Digital Products in complex systems. Insights for the consumer or Product Manager can be generated as data is analyzed to provide reports and recommendations, and to take proactive action.

For effective full lifecycle management, policies and controls should be put in place to ensure the information created in the design, creation, management, and support of the Digital Product, and operational support of systems deployed to support Digital Product Instances remain available for a much longer period than is typical in a project-oriented investment model.

Data governance must be effectively applied across the full lifecycle of the product, including data retention management and compliance with privacy and other applicable policies.

The design, planning, source code, and other information created to define and manage the system should be retained for future planning and support purposes. Without this, it is impossible for the Product Manager to track technical, operational, and financial aspects of the Digital Product over time.

4.13. Service Contract Lifecycle Concerns

The Service Contract exists separately from the digital and physical components of the Digital Product, and may require management for reporting and regulatory purposes long after active users or Digital Product Instances have gone.

A simple example would be a regulatory requirement to retain financial and customer records for a period of time based on laws and regulations in the country of operation.

4.14. Digital Product Fulfillment and Lifecycle Management

The fulfillment flow illustrated in Figure 4-7 demonstrates consumer integration with the Offer. This often results in allocating devices and resources specific to the consumer, shared resources, and resources from an external dependency, such as another Digital Product.

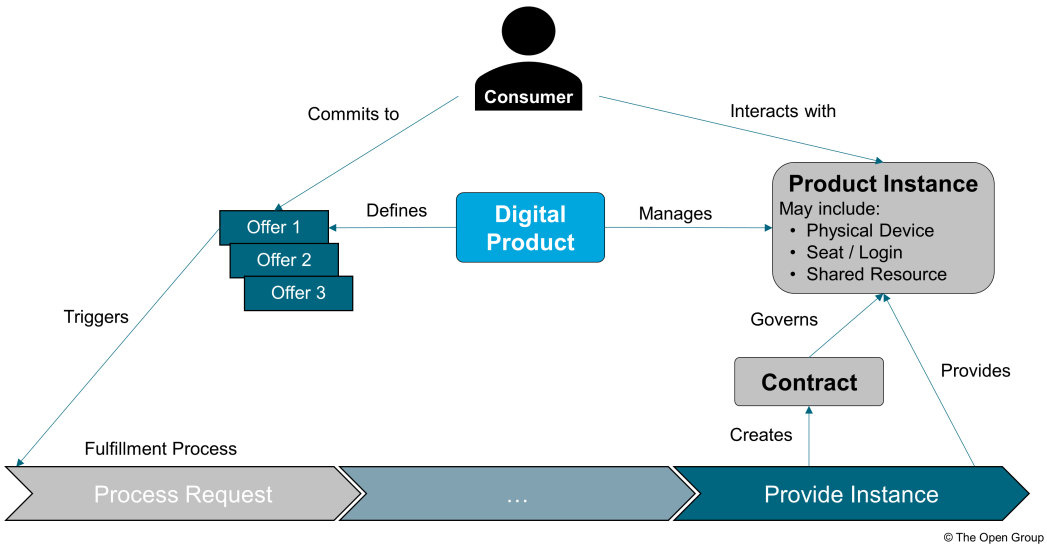

Figure 4-7. Digital Product Fulfillment Event

4.15. The Digital Product Instance

Figure 4-8 more fully shows the elements of a Digital Product Instance. This Product Manager's perspective includes the contract as well as common concerns such as dependencies on other Digital Products, the product team, market and distribution factors, and lifecycle management processes.

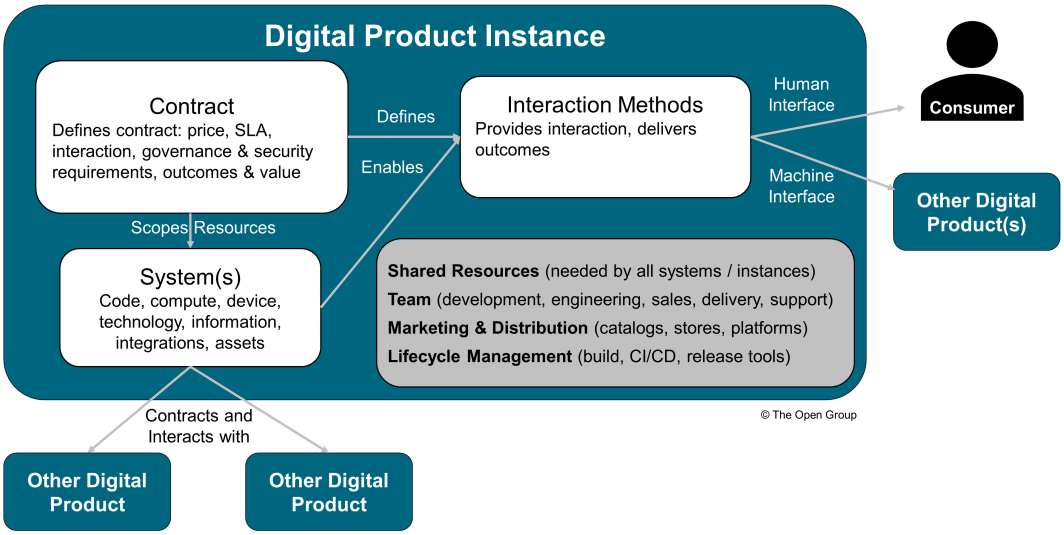

Figure 4-8. Digital Product Instance Structure

4.16. Consumer Types and Interaction Methods

Different types of consumer have different expectations and ways to interact with the Digital Product. A robust Digital Product includes a well-defined, detailed Service Offer structure that defines all anticipated consumer interaction methods, and that can be expressed in terms of the consumer contract.

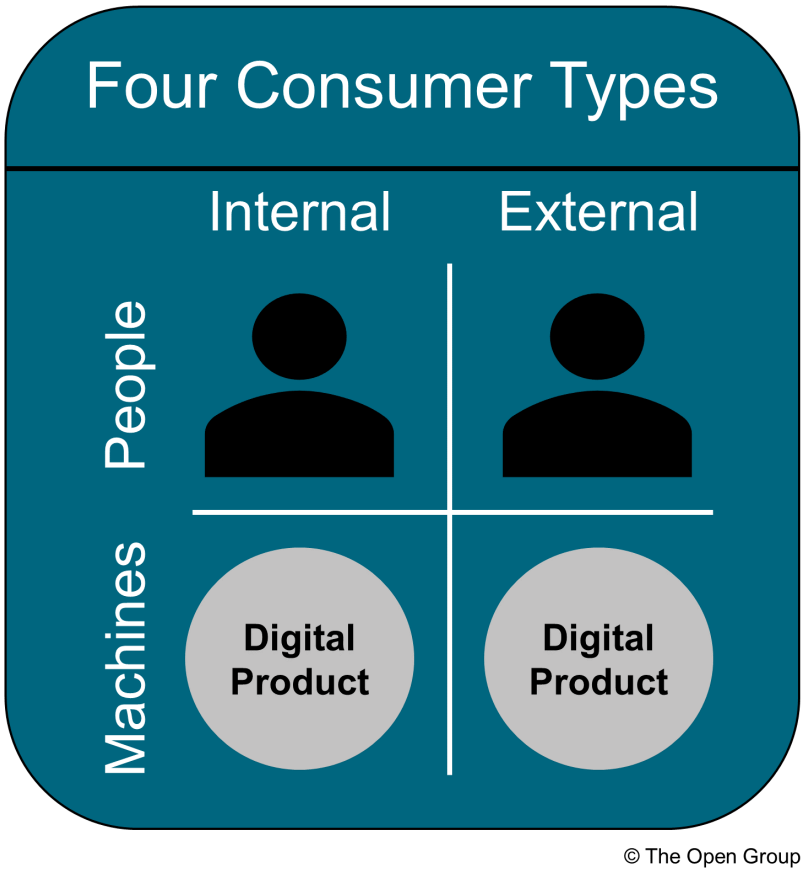

Figure 4-9. Consumer Types

Figure 4-9 suggests a classification of consumer types to consider in the Digital Product Offer, interaction methods, and system design: consumers of a Service Offer may be internal or external (or both), and may be machine or person/organization (or both).

Criteria for Service Offer types are:

- May have internal consumers, such as employees or departments
- May have external consumers or corporate customers and partners
- May have internally exposed APIs, integrations, and processes
- May have externally exposed APIs, integrations, and processes

It is important to set proper expectations for both human and machine consumers in the Service Offer, such as response time, resolution to issues, notification of planned down-time as described in the Offer functional component. Characteristics may vary across this landscape, requiring different Cost Models, SLA templates, consumption rates, security requirements, and business criticality commitments, resulting in more complex Service Offer definitions, and ongoing Subscription management.

4.17. Complex Digital Product Systems

As mentioned above, the resources within the system may include contracts to use other internal or external Digital Products, with both technical and financial dependencies. These third-party dependencies and their explicit or implied contracts are an integral part of product definition, and a consideration in pricing and other management concerns.

Figure 4-10 shows a more complex view of the possible dependency mesh of a Digital Product.

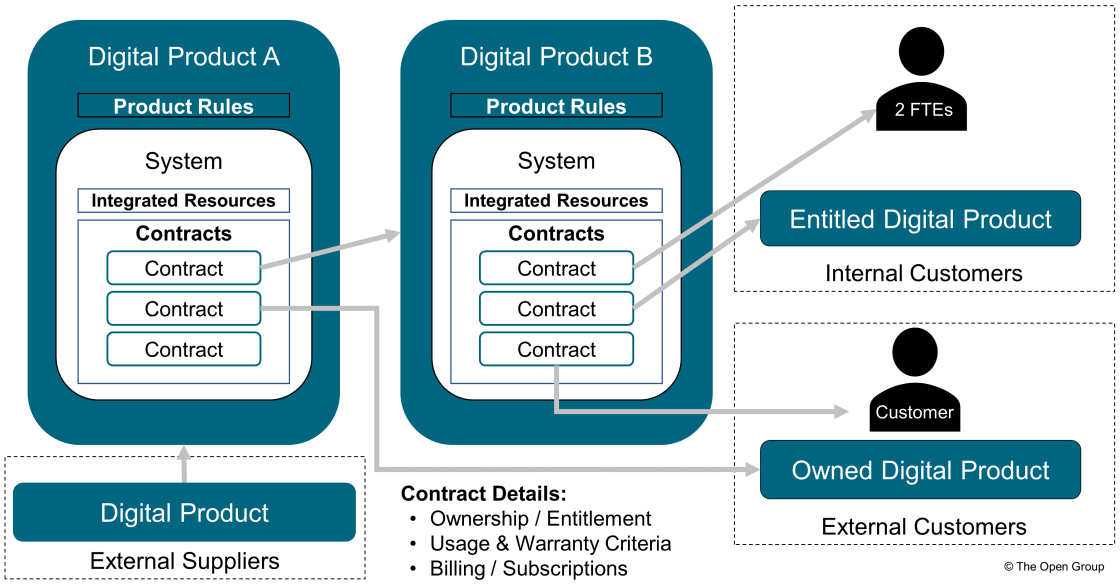

Figure 4-10. Typical Contractual Mesh Among Digital Products

When more elaborate Digital Product systems are deployed and connected using Service Management models such as ITIL®, some resources can be managed as Configuration Items (CIs), representing the

system's underlying and integrated resources. Each CI represents a device or software object (such as an executable file for an application, operating system) that is discoverable when active on the enterprise network, or that has been manually recorded.

A Configuration Management Database (CMDB) is used by most large organizations to manage a subset of possible CIs that has been defined as interesting for management of the IT landscape. Discovery tools or manual data entry are used to capture and update CI information.

More sophisticated CMDBs and CMDB users are able to aggregate individual CIs into a data set that represents more complex and interesting management objects, such as licensed software instances, in-house applications recognizable with, for example, a "signature". A challenge for Digital Product Management is to define the extent to which the Digital Product footprint can or should be recorded using an extended CMDB metamodel.

For example, it may be sensible for each instance of a Digital Product found in the CMDB to include a foreign key pointing to the Subscription record created at instantiation, or to the consumer represented by that record. This or some other mechanism should enable system resources identified in the Configuration functional component to be traceable to those consumers, their Service Contracts, and Subscriptions.

As suggested in Figure 4-10, interaction among collaborating Digital Products, through Service Contracts and Subscriptions, pertains to both human and machine interfaces, entitlements, ownership, and access to systems.

Underlying code executes within interconnected systems to complete interactions for consumers, and may involve external suppliers as well.

Chapter 5. IT4IT Value Streams

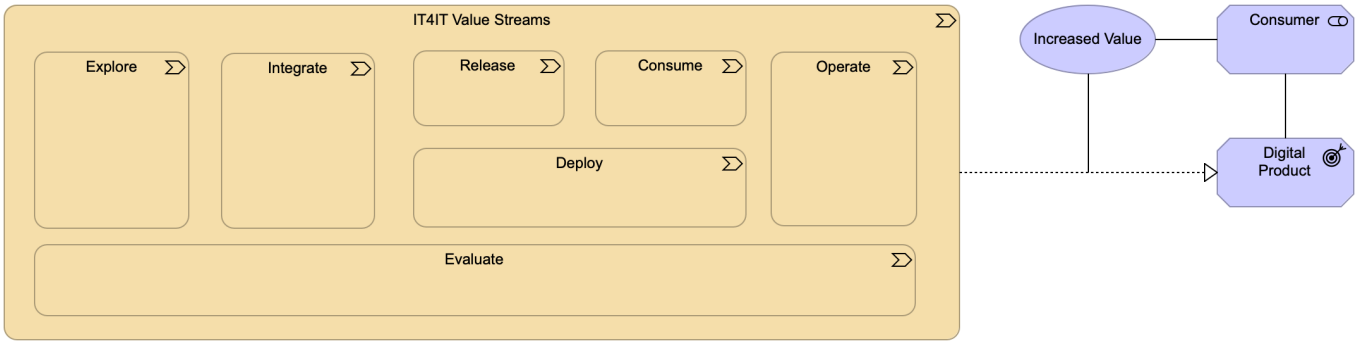

Figure 5-1. IT4IT Value Streams Model

The IT4IT Value Streams, as shown in Figure 5-1, are integrated streams of capabilities within the Architecture Reference Model. They support a holistic lifecycle approach for the planning, creation, delivery, modification, operation, support, and retirement of a Digital Product, combining all of the necessary capabilities to deliver value and support to the dependent parts of the Digital Product lifecycle.

There are seven value streams defined:

- Evaluate
- Explore
- Integrate
- Deploy
- Release
- Consume
- Operate

Despite being highly integrated, value creation with regards to Digital Products requires a specific set of capabilities, functions, processes, and data per value stream. In the following sections therefore, for every value stream, the delivered value, main stakeholder(s), a set of scenarios and value stream stages are described in more detail. Value streams interact or depend on other value streams to support the Digital Product throughout the entire lifecycle.

5.1. Evaluate Value Stream

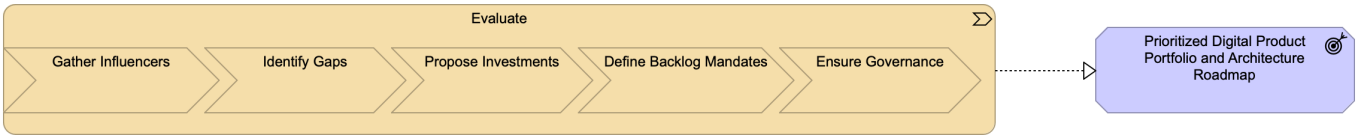

Figure 5-2. Evaluate Value Stream Model

Overview

The value stream “Evaluate”, as shown in Figure 5-2, contributes to the business strategy and portfolio planning activities. It provides a blueprint for optimizing products, services, and investment Portfolio Management. This value stream is focused on the continuous assessment and evaluation of the entire Digital Product Portfolio to optimize co-creation and alignment of business and technology Strategic Objectives.

Many organizations have portfolio processes and solutions in place but suffer from the following limitations:

- Poor data quality and consistency
- No holistic portfolio view across the enterprise
- Inconsistent Portfolio, Service, and Product Management
- Poor tracking and correlation of the product lifecycle

Organizations need accurate and point-in-time information to understand the inter-relationships and inter-dependencies required to truly orchestrate all the moving parts of Digital Products in ways that can help support business objectives and goals.

The Evaluate value stream provides holistic views of Product Portfolio activities. These views offer an understanding of the inter-relationships among many sub-domains, including Portfolio Management, Enterprise Architecture, Application Management, Information Management, Operations Management, and Information Security Management.

The Evaluate value stream is triggered by a portfolio planning cycle. Portfolio planning can be done in weekly, monthly, or annual cycles.

The overall funding of a Digital Product consists of portfolio epic-based funding (often referred to as Grow the Business (GtB), or discretionary) and product specific investments (often referred to as Run the Business (RtB), or non-discretionary). The product-specific Scope Agreement is defined in the Explore value stream and aligned with the prioritized portfolio level Scope Agreements. This funding methodology enables decentralized decision-making by the Product team.

Primary Stakeholder

The primary value stream stakeholder of the Evaluate value stream is the Product Portfolio Manager. The Product Portfolio Manager is responsible for ensuring consistent services and portfolio balance through rationalization and sound investments. The Product Portfolio Manager works closely with Strategists, Business Relationship Managers, Architects, and Technologists to align with strategic drivers, consumer demand, and policy standards. They must work together to optimize the portfolio through well-defined control points, data objects, and governance.

Value

The outcome of the Evaluate value stream is a portfolio investment plan with funding and resources to conduct product exploration activities and refine the plan. Here, resources are referred as both IT-related resources; for example, IT infrastructure, database, software packages, and non-IT-related resources such as HR. This value stream delivers an optimized Product Portfolio with secure and quality Digital Products that are consumer friendly and cost effective, accomplished through continuous portfolio improvement and refinement.

Cross-Value Stream Dependencies

The Evaluate value stream depends on:

- All value streams:
  - To continually bring in ideas, improvements, and demand to the Evaluate value stream
  - To capture data that ensures the quality of the Digital Product will meet the requirements

    Quality is an aspect of each task within each value stream. Quality data may also be used to help analyze the Portfolio Backlog.

Value streams that depend on the Evaluate value stream are:

- All value streams:
  - Improvements start with drivers that are expected to arise from all seven value streams

    All value streams will capture data to ensure the quality of the Digital Product will meet the requirements.
- Explore:
  - Requires qualified Portfolio Backlog Items that will be used to realize the Product Design, either a new or next Product Design version, and the necessary changes to Enterprise Architecture and Strategic Themes

The Evaluate value stream, as shown in Figure 5-3, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

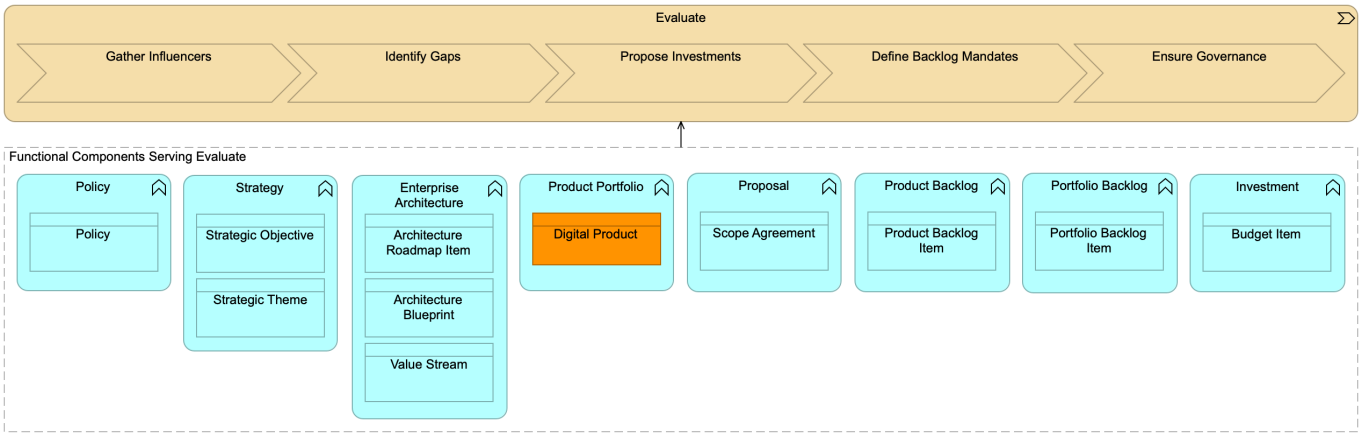

Figure 5-3. Evaluate Value Stream Details Model

In the following sections we document the scenarios and stages of the Evaluate value stream:

- Scenario: Consider a New Digital Product
- Scenario: Perform Digital Product Governance
- Scenario: Rationalize the Product Portfolio
- Scenario: Plan Product Portfolio Investments
- Stage: Gather Influencers
- Stage: Identify Gaps
- Stage: Propose Investments
- Stage: Define Backlog Mandates
- Stage: Ensure Governance

5.1.1. Evaluate Scenarios

The Evaluate value stream is applicable for the following four scenarios:

Consider a New Digital Product Scenario

The trigger for this scenario is a business requirement for a new Digital Product. The outcome is a Scope Agreement and resource allocation focused on Portfolio Backlog Items and Strategic Objectives, indicating there is a strong business case for a new Digital Product. New Digital Products are often considered when a new technology is introduced in the marketplace that may solve a business problem; when technology becomes obsolete and needs to be modernized; or when a new business niche is desired.

Perform Digital Product Governance Scenario

The trigger for this scenario is the Change Management of Digital Products. The outcome is a secure, quality, consumer-friendly, cost-effective Digital Product that aligns with Strategic Objectives, Architecture Principles, technology standards, and financial and policy constraints. Governance ensures the traceability of the Digital Product lifecycle information from Strategy to Development and into Operations.

Rationalize the Product Portfolio Scenario

The trigger for this scenario is a business requirement to rationalize and balance the Digital Product Portfolio. The outcome is a Scope Agreement and resource allocation focused on Portfolio Backlog Items for the portfolio rationalization types: Retain, Replace, Rebuild, Retire. Based on the rationalization type a Digital Product is left as-is, substituted, modernized, or decommissioned. Rationalization is imperative to maintain a healthy and balanced portfolio by minimizing Digital Product technical debt and capability redundancy.

Plan Product Portfolio Investments Scenario

The trigger for this scenario is a portfolio planning increment. A portfolio planning cycle can be in weekly, monthly, or annual increments. The Product Portfolio provides information about the health of existing Digital Products and new product strategy. This information supports decisions for investing or reinvesting in products and services. The outcome of a portfolio planning cycle is a Scope Agreement that provides the required scope and funding for portfolio-related concerns. Digital Product-specific concerns are delegated to a Product Manager and Digital Product stakeholders for refinement in the Explore value stream. Big strategic initiatives and cross-product content are funded and governed at the portfolio level in the Evaluate value stream; while smaller product investments are better managed at the product level in the Explore value stream.

5.1.2. Gather Influencers Stage

The purpose of the value stream stage “Gather Influencers” is to bring together inputs for a planning cycle. Typical inputs include: consumer demand, improvement ideas, new technology opportunities, technology lifecycle events (e.g., end-of-life), and any known policy (legal and regulatory demands) or budget constraints for the Digital Product Portfolio. Other inputs may include an internal and/or external environmental scan to identify strengths, weaknesses, opportunities, and threats. Discussion with the business is centered on Digital Product value delivery for consumers and corresponding policy requirements. The deliverable for this value stream stage is new or updated Strategic Themes and/or Strategic Objectives.

An important part of this stage is to manage the strategic roadmap of the Digital Enterprise. Throughout the IT4IT Value Streams, this roadmap will be refined from the Strategic intent to the Architecture Roadmap to the Digital Product Roadmap to the Release Roadmap.

Table 5-1. Gather Influencers Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Consumer demand • Improvement ideas	• Defined vision • Strategic Themes • Strategic Objectives
Value Item: • Technology investment plan input	
Activities: • Shall co-create vision and strategic roadmap (Business + Technology) • Should consider existing strategic roadmap, consumer demand, improvement ideas, and budget constraints • Should review standards and policies for compliance gaps and new requirements • May conduct an environmental scan: ◦ Internal analysis to assess strengths and weaknesses ◦ External analysis of the market and competitors (political, environmental, social, technical, economic, and legal) to assess opportunities and threats • Shall define Strategic Themes and Strategic Objectives	
Examples of Participating Stakeholders: • Business Analyst • Business Stakeholder • Chief Security Officer • Compliance Manager • Consumer • Enterprise Architect • External Stakeholder • Product Manager • Vendor Manager	**Participating Data Objects** (*Component*): • Policy (*Policy Component*) • Strategic Objective (*Strategy Component*) • Strategic Theme (*Strategy Component*)

5.1.3. Identify Gaps Stage

The purpose of the value stream stage “Identify Gaps” is to evaluate the current state and identify opportunities that align to the vision, Strategic Themes, and Strategic Objectives. Inputs for the current state evaluation should include the consideration of recent consumer survey information and capability assessments. Gaps are identified by comparing the current state to the desired future state, and opportunities are derived. The deliverable for this value stream stage is a list of gaps and opportunities expressed in proposed Scope Agreements, with any related Portfolio Backlog Items.

Table 5-2. Identify Gaps Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Vision • Strategic Themes • Strategic Objectives	• Defined list of gaps and opportunities • Proposed Scope Agreements with related Portfolio Backlog Items
Value Item: • Consumer satisfaction and efficient/Agile operations	
Activities: • Shall assess the current state of the Product Portfolio • Should consider consumer survey information • Should research consumer needs (consumer journey, business model, capabilities) • Should perform a capability assessment • Shall identify gaps in the Product Portfolio • May identify opportunities for improvement and Product Portfolio rationalization • Shall develop proposed Scope Agreements with identified opportunities and related backlog items	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Business Analyst • Business Stakeholder • Chief Security Officer • Data Protection Officer • Enterprise Architect • Product Architect • Product Manager • Product Portfolio Manager • Risk Analyst • Security Architect	• Digital Product (*Product Portfolio Component*) • Policy (*Policy Component*) • Product Backlog Item (*Product Backlog Component*) • Scope Agreement (*Proposal Component*) • Strategic Objective (*Strategy Component*) • Strategic Theme (*Strategy Component*) • Value Stream (*Enterprise Architecture Component*)

5.1.4. Propose Investments Stage

The purpose of the value stream stage “Propose Investments” is to prioritize a list of opportunities and associated Scope Agreements. The prioritization should be based on a scoring method that considers business value, risk, costs, time, and resource availability. Scope Agreements are evaluated to determine if there is already a solution in the Product Portfolio that can be used or modified to satisfy the business need. If it is determined that there is not an existing solution and the new demand is in alignment with the digital strategy, the Scope Agreement is assessed against other work in the Portfolio Backlog and prioritized accordingly. The prioritization criteria should also consider factors like urgency and the impact of opportunities. The deliverable for this value stream stage is updated Scope Agreements with clarifications discovered during a prioritization of opportunities.

Table 5-3. Propose Investments Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• List of gaps and opportunities and proposed Scope Agreements with related Portfolio Backlog Items	• Updated list of prioritized opportunities and associated Scope Agreements
Value Item: • Portfolio decisions based on business priorities	

Activities:
• Shall consider business value, risk, costs, time, and resource availability for opportunities • May complete a what-if analysis • May consolidate demand and ideas • Shall classify investment/divestment strategic themes (try something new, invest, divest, continue to support) • Should analyze priority considering factors such as urgency and impact of opportunities

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Enterprise Architect • Product Manager • Product Portfolio Manager • Security Architect • Vendor Manager	• Budget Item (*Investment Component*) • Digital Product (*Product Portfolio Component*) • Policy (*Policy Component*) • Portfolio Backlog Item (*Portfolio Backlog Component*) • Scope Agreement (*Proposal Component*) • Strategic Objective (*Strategy Component*) • Strategic Theme (*Strategy Component*) • Value Stream (*Enterprise Architecture Component*)

5.1.5. Define Backlog Mandates Stage

The purpose of the value stream stage “Define Backlog Mandates” is to determine if there are any mandates or directives that will impact the Product Portfolio, and to ensure mandates are clearly defined in backlog items and related in Scope Agreements. This should include a review of the existing portfolio and managed service provider agreements. Portfolio rationalization to balance the portfolio should be completed regularly and may generate backlog items that are bundled into Scope Agreements. Upon approval to proceed on a Scope Agreement, initial funding is allocated to the Explore value stream to identify concerns specific to a Digital Product and determine feasibility. The deliverables for this value stream stage are refined Scope Agreements to reflect mandates and defined Architecture Roadmap Items.

Table 5-4. Define Backlog Mandates Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• List of prioritized opportunities and associated Scope Agreements	• Refined Scope Agreements with initial funding and defined Architecture Roadmap Items

Value Item:

- The right digital technology for the business

Activities:

- Should consider existing portfolio and managed service provider agreements
- Shall further define Scope Agreements
- Should develop the Architecture Roadmap
- Should rationalize and balance the Product Portfolio (Retain, Replace, Rebuild, Retire)
- May create Architecture Blueprints
- May identify architecture enablers (reference architectures, building blocks, templates, patterns)

Examples of Participating Stakeholders:

- Business Analyst
- Business Stakeholder
- Chief Security Officer
- Compliance Manager
- Data Protection Officer
- Enterprise Architect
- External Stakeholder
- Product Manager
- Product Portfolio Manager

Participating Data Objects (*Component*):

- Architecture Blueprint (*Enterprise Architecture Component*)
- Architecture Roadmap Item (*Enterprise Architecture Component*)
- Digital Product (*Product Portfolio Component*)
- Policy (*Policy Component*)
- Portfolio Backlog Item (*Portfolio Backlog Component*)
- Product Backlog Item (*Product Backlog Component*)
- Scope Agreement (*Proposal Component*)
- Strategic Objective (*Strategy Component*)
- Strategic Theme (*Strategy Component*)
- Value Stream (*Enterprise Architecture Component*)

5.1.6. Ensure Governance Stage

The purpose of the value stream stage “Ensure Governance” is to provide methods that guide consistent and compliant Digital Product deliverables. This should include the use of reference architectures, building blocks, patterns, and templates. The deliverables for this value stream stage are guardrails, integrated repositories, enablers, audit definitions, and Architecture Principles.

Table 5-5. Ensure Governance Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Scope Agreements with initial funding and Architecture Roadmap Items	• Defined guardrails, integrated repositories, enablers, audit definitions, and Architecture Principles
Value Item:	
• Secure, high quality, cost effective, consumer-friendly Digital Product	
Activities:	
• Should develop guardrails for adherence to technology standards and policies • Should create repositories that ensure the traceability of the Digital Product artifacts from Strategy to Development and into Operations to Retirement • Should confirm the use of enablers (reference architectures, building blocks, patterns, templates)	
Examples of Participating Stakeholders: • Business Analyst • Business Stakeholder • Chief Security Officer • Enterprise Architect • External Stakeholder • Product Manager • Product Portfolio Manager • Vendor Manager	**Participating Data Objects** (*Component*): • Digital Product (*Product Portfolio Component*) • Policy (*Policy Component*) • Portfolio Backlog Item (*Portfolio Backlog Component*) • Scope Agreement (*Proposal Component*) • Strategic Objective (*Strategy Component*) • Strategic Theme (*Strategy Component*) • Value Stream (*Enterprise Architecture Component*)

5.2. Explore Value Stream

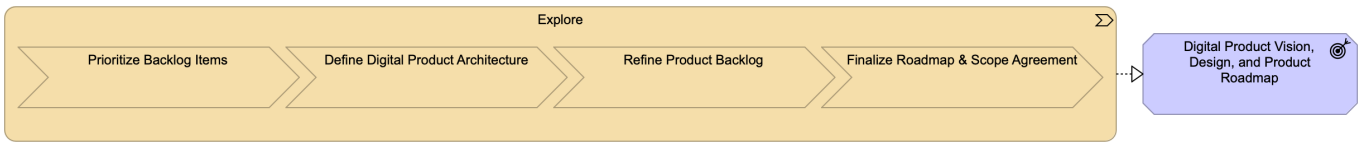

Figure 5-4. Explore Value Stream Model

Overview

The value stream “Explore”, as shown in Figure 5-4, is performed on the Scope Agreements developed in the Evaluate value stream. It ensures that all planned and accepted Portfolio Backlog Items are further detailed, budgeted, and sourced for Digital Products. This value stream continuously explores new features and/or future directions of a Digital Product aligned with strategic direction and business

needs. It ensures that the Product Design evolves to facilitate innovation and optimize business outcomes.

Digital Product Portfolio Backlog Items for investigating a new idea, optimizing existing ideas, or retirement may be resolved by different funding, management, reporting, and contractual obligations for various types of consumers. Differences in budgeting, technology, contracts, business domains, security, or organizational/management structures will influence the portfolios required for an organization.

The Explore value stream is intended to be executed in quick cycles to validate release iterations. The Portfolio Backlog Items should include the generation of a proof-of-concept, conceptual service, or Minimum Viable Product (MVP) with the high-level architectural attributes and activities to begin gathering requirements for the build stage.

The Explore value stream stage ensures that the allocation of resources, budget, and phases for Digital Product releases remain aligned with agreed, planned, and accepted Portfolio Backlog Items and Architecture Blueprints. The primary stakeholder of this value stream is a Product Manager who approves Scope Agreements and release plans for Digital Products.

This value stream can be triggered by a changed Scope Agreement.

Primary Stakeholder

The primary value stream stakeholder of the Explore value stream is the Product Manager. The Product Manager is responsible for the development and success of a Digital Product for an organization. Product Managers own the business strategy, manage the requirements, and launch features for a product. The Product Manager coordinates the work done by other functions, including: Business Relationship Manager, Architect, Strategist, Technologist, Business Owners, and DevSecOps.

Value

The outcome of the Explore value stream is an accepted and planned Product Backlog with a detailed Scope Agreement, Release Roadmap, and resource allocation. During the Explore stages, Scope Agreements are refined to include product specific details. The value is a Digital Product roadmap with allocated resources.

Cross-Value Stream Dependencies

The Explore value stream depends on:

- Evaluate:
  - Delivers qualified, as accepted and planned, Portfolio Backlog Items that will be used to realize the Digital Product and refine the Architecture Blueprint(s) (all product lifecycle stages, scaling and maturing the product, as well as retracting/retirement) and necessary changes to the Enterprise Architecture and Strategic Themes according to the Architecture Roadmap Items

- Integrate:
  - Provides information on the progress of the Scope Agreement during execution back to the Explore value stream to enable governance on progress and planned dependencies with other Scope Agreements
  - If an agreed Scope Agreement cannot be met, the Explore value stream is responsible for changing the Scope Agreement; e.g., change the covered Portfolio Backlog Items (to include more or go for less), change the time or budget, or change the sourcing party

Value streams that depend on the Explore value stream are:

- Evaluate:
  - Improvements start with drivers, that are expected to arise from all seven value streams

    All value streams will capture data to ensure that the quality of the Digital Product will meet the requirements
- Integrate:
  - Will receive an approved Scope Agreement together with an agreed set of Product Backlog Items to start work on a Product Release

The Explore value stream, as shown in Figure 5-5, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

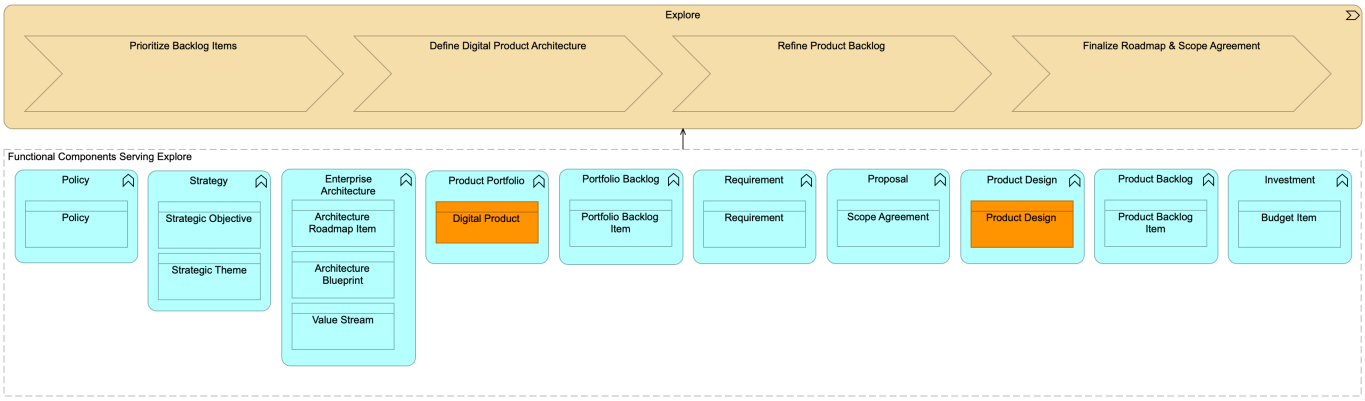

Figure 5-5. Explore Value Stream Details Model

In the following sections we document the scenarios and stages of the Explore value stream:

- Scenario: Investigate a New Digital Product Idea
- Scenario: Optimize a Digital Product
- Scenario: Refine Digital Product Feasibility
- Scenario: Retire Digital Product
- Stage: Prioritize Backlog Items
- Stage: Define Digital Product Architecture
- Stage: Refine Product Backlog
- Stage: Finalize Roadmap & Scope Agreement

5.2.1. Explore Scenarios

The Explore value stream is applicable for the following four scenarios:

Investigate a New Digital Product Idea Scenario

The trigger for this scenario is a Scope Agreement and an initial budget allocation to invest in a new Digital Product idea or enhancement that has a strong value proposition and alignment to one or more Strategic Objectives. The outcome is validation of whether the new product idea has a strong value proposition, alignment to one or more Strategic Themes, and will comply with technology standards and policy. The scope incorporates Portfolio Backlog Items for testing assumptions that may include a proof-of-concept, conceptual service, or an MVP to prove that consumer needs and problems can be addressed. Results retrieved from the proof-of-concept may lead to the refinement of Portfolio Backlog Items to continue researching the new product idea, or may result in a decision to reconsider, or rethink, a decision, and the reallocation or discontinuation of further investments and resources. This scenario may only be intended to validate a hypothesis and may not include the full operationalization of a Digital Product (for example, its inclusion in the Service Offer Catalog, automation using the Operate functions, or marketing it to consumers).

Optimize a Digital Product Scenario

The trigger for this scenario is a Scope Agreement and an initial budget allocation that incorporates Portfolio Backlog Items for optimizing a Digital Product. The outcome is an enhanced Digital Product. The scope includes activities to optimize a Digital Product based on innovating new features and addressing Defects, mandates, and technical debt. The outcome is a Scope Agreement and resource allocation priority focused on Portfolio Backlog Items for implementing a release for an existing Digital Product. The scope includes the creation of an MVP/Minimum Marketable Product (MMP) release together with testing assumptions about the business model.

Results retrieved from the MVP may result in a decision to continue and refine, to pivot and reallocate, or to discontinue further investments and resources. This may lead to a request to increase or decrease investment, a change to resource allocation, a change in scope, or a change in timeline. An MVP may

only include quick enhancements to Digital Products, as small changes, resulting in immediate business value (time, resources, and scope).

Refine Digital Product Feasibility Scenario

The trigger for this scenario is a portfolio level Scope Agreement with a decision to invest in a new or existing product. Product planning cycles are conducted to further detail the budget, resource allocation, and feasibility for planned and accepted Portfolio Backlog Items. The outcome is a product level Scope Agreement that further details the scope and budget for a Product Design and the optimization of business outcomes. While big strategic initiatives and cross-product content are funded and governed on the portfolio level in the Evaluate value stream; smaller product investments are better managed at the product level by the Product team in the Explore value stream.

Retire Digital Product Scenario

The trigger for this scenario is a Scope Agreement and an initial budget allocation that incorporates Portfolio Backlog Items for retiring a Digital Product. The outcome is a decommissioned Digital Product. The scope includes activities to mitigate negative effects on consumers, impacts to dependencies, the re-allocation of resources, and the archival of data. Reasons for retirement may be driven based on technical debt, cost, duplication in the environment, lack of interest or use, or other valid reasons to no longer maintain a Digital Product. It is important to retire Digital Products that are no longer in use to reduce the complexity of technical environments.

5.2.2. Prioritize Backlog Items Stage

The purpose of the value stream stage “Prioritize Backlog Items” is to refine qualified Portfolio Backlog Items and mandates for new or existing Digital Product investment decisions. The budget allocated in the Evaluate value stream for the Product Portfolio is used to explore Digital Product ideas and determine feasibility. This should include ensuring that scoped Product Backlog Items deliver value, mitigate risks, and meet governance criteria (e.g., technology, security, reliability, supportability). The deliverables for this value stream stage are a Scope Agreement with prioritized Product Backlog Items; scope and timing refined.

Table 5-6. Prioritize Backlog Items Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Digital Product is selected for exploration and to determine feasibly	• Refined Scope Agreement with prioritized Product Backlog Items, scope, and timing
Value Item:	
• Digital Product investments that meet budget, compliance, and supportability requirements	
Activities:	
• Shall prioritize and refine qualified Portfolio Backlog Items and mandates for new or existing Digital Product investment decisions • May consider new revenue opportunities, Strategic Themes, rationalization effects, customer demands, improvements, emergency fixes, directives related to compliance or regulatory, Digital Product dependency changes, end-of-life and decommissioning, transition to next version • May adjust funding and resource allocations for Scope Agreements • May group, correlate, and rationalize qualified Product Backlog Items to avoid redundancy and maximize resources to gain efficiencies and scale • Should ensure scoped Product Backlog Items deliver value, mitigate risks and meet governance criteria (e.g., technology, security, reliability, supportability) • Should plan program increments (e.g., quarterly planning) • Should align portfolio and product roadmap	
Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Business Analyst • Compliance Manager • Enterprise Architect • External Stakeholder • Product Manager • Product Portfolio Manager	• Digital Product (*Product Portfolio Component*) • Policy (*Policy Component*) • Portfolio Backlog Item (*Portfolio Backlog Component*) • Scope Agreement (*Proposal Component*) • Strategic Objective (*Strategy Component*) • Strategic Theme (*Strategy Component*)

5.2.3. Define Digital Product Architecture Stage

The purpose of the value stream stage “Define Digital Product Architecture” is to collaborate with stakeholders to validate product viability. This should include industry research and a Business Impact Analysis (BIA). Product Portfolio funding and resource allocation may be adjusted based on findings. The deliverable for this value stream stage is a conceptual product architecture.

Table 5-7. Define Digital Product Architecture Value Stream Stage

<table>
<tr><td>Entrance Criteria:
Scope Agreement with prioritized Product Backlog Items, scope, and timing</td><td>Exit Criteria:
Defined conceptual product architecture</td></tr>
<tr><td colspan="2">Value Item:
Feasible Product Architecture and impact assessment</td></tr>
<tr><td colspan="2">Activities:
Should define the Digital Product conceptual architecture or high-level Product Design
Shall create Architecture Blueprints
Shall collaborate with stakeholders to validate product viability
Should research industry best practices, the consumer needs, including the business model, customer journey, and capabilities
May re-validate the Digital Product Scope Agreement against risk, cost, time, value, and other parameters
Should establish business impacts and resource (people, process, data, technology) availability
Address delivery and supportability requirements to maintain Digital Product Instances
May adjust Product Portfolio funding and resource allocations</td></tr>
<tr><td>Examples of Participating Stakeholders:
Business Stakeholder
Data Protection Officer
Enterprise Architect
Product Architect
Product Manager
Security Analyst
Security Architect
Vendor Manager</td><td>Participating Data Objects (Component):
Architecture Blueprint (Enterprise Architecture Component)
Policy (Policy Component)
Portfolio Backlog Item (Portfolio Backlog Component)
Product Design (Product Design Component)
Strategic Objective (Strategy Component)
Strategic Theme (Strategy Component)</td></tr>
</table>

5.2.4. Refine Product Backlog Stage

The purpose of the value stream stage “Refine Product Backlog” is to translate a Scope Agreement into smaller pieces of work that can be assigned to product level features and linked to the product backlog. This includes organizing assigned backlog items into work packages and adjusting the Scope Agreement, business case, and budget as needed. The deliverables for this value stream stage are established Architecture Roadmap Items.

Table 5-8. Refine Product Backlog Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Conceptual product architecture	• Established Architecture Roadmap Items
Value Item:	
• Incremental Digital Product work packages with dependencies and priorities	
Activities:	
• Shall translate a Scope Agreement into “smaller” pieces of work that can be assigned to the “product” level; e.g., features (linked to the product backlog) • Shall organize/assign backlog items into work packages • Should determine and accommodate dependencies • Shall size and prioritize the work • Shall update the Scope Agreement with new findings • May re-confirm the product work package aligns with the vision, strategy, policies • May create Architecture Blueprint(s) per product work package in line with the overall Enterprise Architecture • May align the product work packages with the Architecture Roadmap • May refine the business case and adjust budget	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Business Analyst • Business Stakeholder • Data Protection Officer • Enterprise Architect • Product Manager • Security Analyst	• Architecture Blueprint (*Enterprise Architecture Component*) • Architecture Roadmap Item (*Enterprise Architecture Component*) • Digital Product (*Product Portfolio Component*) • Policy (*Policy Component*) • Portfolio Backlog Item (*Portfolio Backlog Component*) • Product Backlog Item (*Product Backlog Component*) • Product Design (*Product Design Component*) • Requirement (*Requirement Component*) • Scope Agreement (*Proposal Component*)

5.2.5. Finalize Roadmap & Scope Agreement Stage

The purpose of the value stream stage “Finalize Roadmap & Scope Agreement” is to assign work and begin to conduct planning meetings on a regular cadence. Stakeholder buy-in and agreement to the Release roadmap and outcomes must be obtained. This includes setting up tracking and reporting on progress. The deliverables for this value stream stage are an approved Architecture Roadmap and outcomes.

Table 5-9. Finalize Roadmap & Scope Agreement Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Architecture Roadmap Items	• Approved Architecture Roadmap and outcomes
Value Item:	
• Architecture Roadmap with work packages outlining resource requirements and budget allocation	

Activities:

- Shall assign/allocate resources to work packages
- May assign/allocate resources to work on subsequent work packages
- May conduct planning meetings on a regular cadence
- Shall obtain stakeholder review and agreement of the Architecture Roadmap and outcomes
- Shall track and report on the progress of the Scope Agreement, work packages, and related IT initiatives
- May update a Scope Agreement and work packages
- May allocate / assign resources for build and deliver activities (link to value streams)
- Shall update Portfolio Backlog Items to reflect the build and deliver progress status

Examples of Participating Stakeholders:

- Development Team
- Enterprise Architect
- Product Manager
- Risk Analyst
- Security Analyst
- Vendor Manager

Participating Data Objects (*Component*):

- Architecture Roadmap Item (*Enterprise Architecture Component*)
- Digital Product (*Product Portfolio Component*)
- Policy (*Policy Component*)
- Portfolio Backlog Item (*Portfolio Backlog Component*)
- Product Backlog Item (*Product Backlog Component*)
- Product Design (*Product Design Component*)
- Scope Agreement (*Proposal Component*)

5.3. Integrate Value Stream

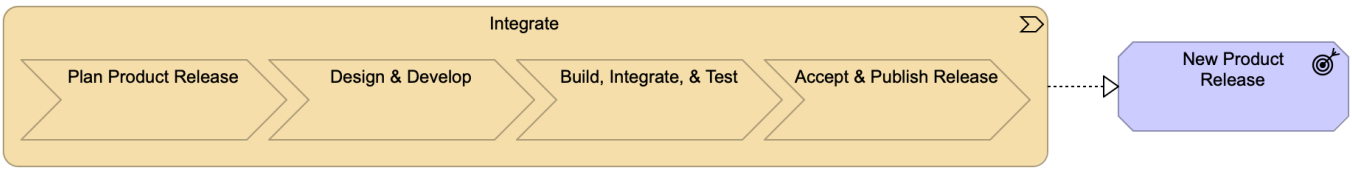

Figure 5-6. Integrate Value Stream Model

Overview

The value stream “Integrate”, as shown in Figure 5-6, creates a new Product Release. It covers the planning, design, development (or configuration), and testing of a new Product Release (in non-production environments), and makes the Product Release available for deployment and release into production. A new Product Release can also be related to an emergency fix needed to resolve a production issue (such as resolving vulnerabilities and critical bugs).

The Integrate value stream covers the creation and modification of all types of Digital Products; for example, custom-build software, configuration of SaaS applications, Commercial Off-The-Shelf (COTS) software packages, PaaS platforms, Infrastructure as a Service (IaaS) products. This value stream is not limited to application development, but also subject to the development and maintenance of infrastructure products (and platforms).

Work is initiated and managed in the Product Backlog (and related team backlogs), which can receive demand signals from consumer or business stakeholders (e.g., new features or business requirements), or from other functional components such as Portfolio Backlog Management, Policy Management, and Problem Management. This may take the form of an approved Portfolio Backlog Item, or may be a smaller-grained signal such as an individual feature, user story, Defect, or Problem.

The Integrate value stream continuously pulls work from the Product Backlog based upon various triggers and inputs; such as:

- New features and requirements raised by a consumer (or a business stakeholder) and/or other product teams
- New or modified non-functional requirements such as those related to Service-Level Objectives (SLOs), security, risk, and compliance
- New portfolio epics from the portfolio backlog which need to be refined into Product Backlog Items
- New versions, updates, or fixes provided by the vendor (e.g., software updates and patches) and/or other technology refresh activities
- Problems detected in operational environments (e.g., functional issues, performance issues)
- Emergency changes to resolve major incidents and/or mitigate major risks (e.g., security issues and vulnerabilities)

The Integrate value stream delivers the Product Release (for a new or modified product) which is packaged for immediate or future deployment through the Deploy value stream.

Note that although the stage is shown as a linear process, the actual work in the value stream consists of multiple iterations, continuous cycles, and flow of work with feedback loops (e.g., Continuous Integration/Continuous Delivery (CI/CD) and continuous testing).

Primary Stakeholder(s)

The typical primary stakeholders involved in the Integrate value stream are the Product Manager and associated Development Teams (also referred to as Product Teams, or DevOps Teams if Development and Operations are combined into one team), responsible for the planning, design, creation, and testing of a Product Release.

Other primary stakeholders involved are business stakeholders and customer representatives involved in defining business needs (voice of the customer), validating (and testing) the product, and those who ultimately benefit from the value created by using the product.

Value

The outcome of the Integrate value stream is a new Product Release, accepted and ready for deployment and release into production. This is a release for a new or existing Digital Product which provides additional value (and/or reduced risk) once made available and used by consumers in production, as a result of:

- New features (e.g., providing additional value by fulfilling new consumer needs)
- Non-functional improvements (e.g., improving performance, improve user experience, compliance, security, availability, improved resilience)
- Reducing technical debt (and reducing risks), such as those related to technology upgrades/patches and code quality improvements
- Resolving issues/problems (e.g., improve reliability, reduce Incidents in production, improve performance)
- Reduce costs (e.g., improve utilization of resources)

Cross-Value Stream Dependencies

The Integrate value stream depends on:

- Explore:
  - Delivers Portfolio Backlog Items and an approved Scope Agreement to start creating or updating the Product Backlog
  - If the agreed Scope Agreement cannot be met, the Explore value stream is responsible for changing the Scope Agreement within the boundaries of the Architecture Roadmap and Product Portfolio; for example, to allocate more time or a higher budget
  - Delivers an updated product vision and high-level roadmap
- Consume:
  - Deliver the environments to work on, as a request can be used to fulfill the provisioning of a needed development, test, and staging environment
- Deploy:
  - Deployment of the Product Release into a production environment
- Operate:
  - Delivers an inflow for Defects, as Problems related to Actual Product Instances may require a fix (or patch) to be created through the Integrate value stream

Value streams that depend on the Integrate value stream are:

- Evaluate:
  - The Evaluate value stream uses all lifecycle information from all value streams to identify potential improvements in the product (and/or way of working)

- Explore:
  - Provides development progress and team capacity

    The status of the Product Backlog Item is fed back so it can be taken into account when planning the Portfolio Backlog Items for sourcing.

  - If an agreed Scope Agreement cannot be met, the Explore value stream is responsible for changing the Scope Agreement; for example, by changing the covered Portfolio Backlog Items (to include more or go for less), changing the time or budget, or changing the sourcing party
- Deploy:
  - Provides the approved Product Release, which in turn triggers the deployment scheduling of the Product Release to the production environment
- Release:
  - Due to a new Product Release, one or more Service Offers need to be modified (or created) to make the new Product Release available for consumption
- Operate:
  - The Operate value stream depends upon the development of operations features, such as monitoring (and observability), logging, and automated remediation Runbooks

    The Integrate value stream needs to develop the required functionality to manage the product in an operational environment.

The Integrate value stream, as shown in Figure 5-7, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

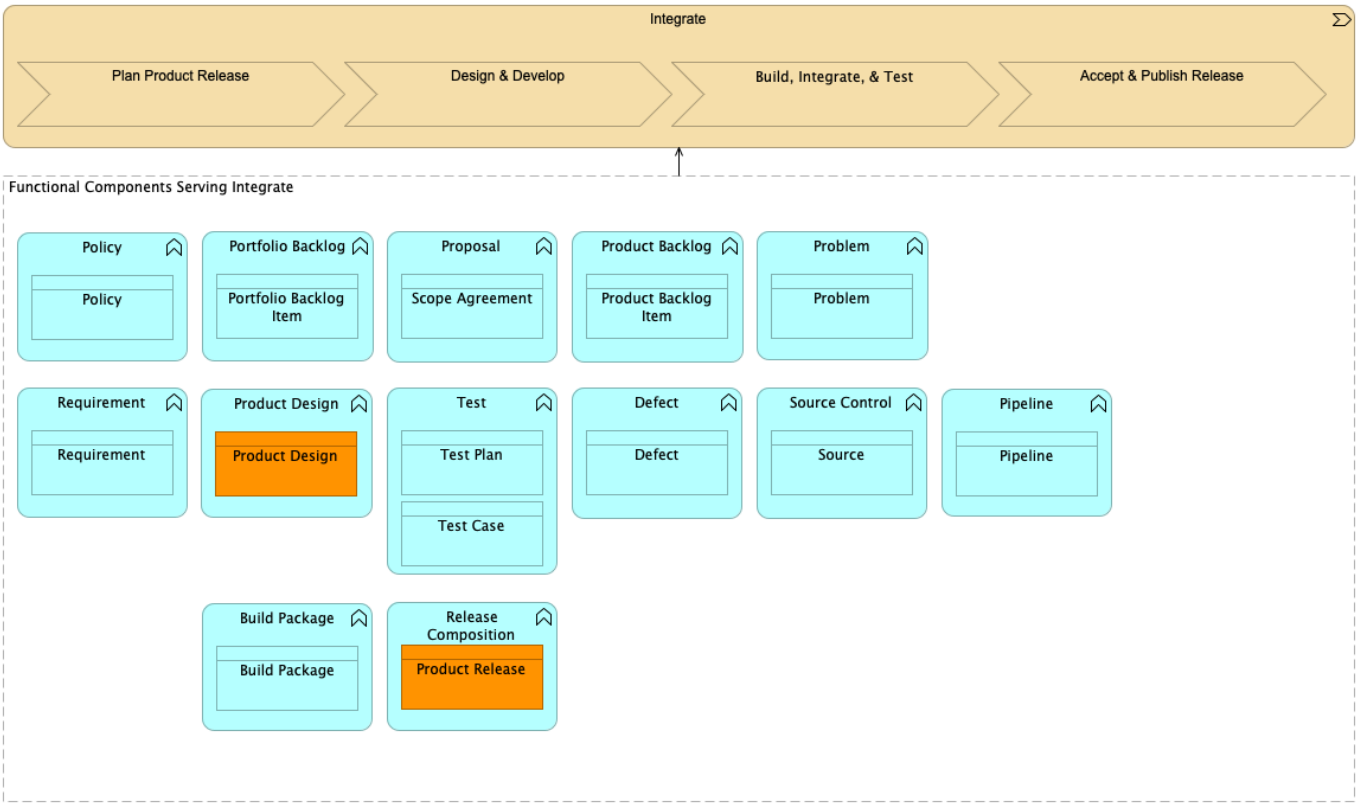

Figure 5-7. Integrate Value Stream Details Model

In the following sections we document the scenarios and stages of the Integrate value stream:

- Scenario: Develop a New or Initial Product Release
- Scenario: Configure an Off-the-Shelf or SaaS Product for Use
- Scenario: Deliver an Emergency Change or Hotfix
- Scenario: Update of a Vendor Product
- Stage: Plan Product Release
- Stage: Design & Develop
- Stage: Build, Integrate, & Test
- Stage: Accept & Publish Release

5.3.1. Integrate Scenarios

The Integrate value stream is applicable for the following four scenarios:

Develop a New or Initial Product Release Scenario

In this scenario a Product Release is developed which may be software and/or infrastructure and is developed in-house or (partly) sourced with an external partner. The approach may either be traditionally waterfall or iterative, which would imply that iterations have taken place inside the Design & Develop value stream stage.

Configure an Off-the-Shelf or SaaS Product for Use Scenario

Off-the-Shelf applications and SaaS solutions need to be integrated in a business Value Stream and supporting infrastructure environments, which might include the configuration of the application software. The Scope Agreement contains the mandate to contract the software or SaaS solution with the vendor and configure the product to fit in the specific business Value Stream and supporting infrastructure.

Deliver an Emergency Change or Hotfix Scenario

The Integrate value stream can be triggered by the Operate value stream in case a Defect needs to be created. The Build Package of the Digital Product in Operations is changed to fix the Defect, after which a new Product Release is delivered. It will be investigated whether this fix also needs to be applied (backported) to the newest Product Release under development (to prevent regression issues).

Update of a Vendor Product Scenario

A Digital Product can be based upon multiple third-party services and software components (e.g., packaged-based software, SaaS, PaaS). The Integrate value stream should therefore also cover the technology refresh of new releases delivered by the software vendor and/or the SaaS service provider. It is important to include these technology lifecycle events into the Product Backlog so they can be planned and executed accordingly.

A new release of the vendor product could, for example, include new features, patches/fixes for problems, or vulnerabilities and performance improvements. Changes from the vendor need to be integrated into the product build and release of the Digital Product, and tested to ensure the updates are successfully applied.

5.3.2. Plan Product Release Stage

Description

The purpose of the value stream stage “Plan Product Release” is to plan all development and testing activities required for creating the Product Release. Multiple teams may be needed, and different development methodologies may be used. This value stream stage is revisited as it oversees the other value stream stages in this value stream to cater for changes in the planning and iterations of the Design & Develop value stream stage.

Table 5-10. Plan Product Release Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• New or modified Product Backlog Item/Requirement (e.g., linked to a portfolio epic, Scope Agreements) • New problem (to be resolved) • Emergency change request (e.g., to resolve a major product issue or vulnerability) • Modified/new policies	• Prioritized and refined Product Backlog • Iteration/sprint plan created
Value Item: • Defined, prioritized, and planned Product Backlog Items	
Activities: • Shall analyze the needs of stakeholders • Shall capture/document Product Backlog Items (e.g., derived from portfolio epics, problems) • Shall refine Product Backlog Items • Shall prioritize Product Backlog Items (including value, risk, and effort estimation) • Shall create the test strategy and Test Plan • Shall create a sprint/iteration plan	
Examples of Participating Stakeholders: • Business Analyst • Business Stakeholder • Consumer • Development Team • Product Architect • Product Manager • Risk Analyst • Scrum Master • Test Specialist	**Participating Data Objects** (*Component*): • Defect (*Defect Component*) • Policy (*Policy Component*) • Portfolio Backlog Item (*Portfolio Backlog Component*) • Problem (*Problem Component*) • Product Backlog Item (*Product Backlog Component*) • Requirement (*Requirement Component*) • Scope Agreement (*Proposal Component*) • Test Plan (*Test Component*)

5.3.3. Design & Develop Stage

Description

The purpose of the value stream stage “Design & Develop” is to analyze the requirements, create (or update) the design artifacts, and develop the changes (and associated Test Cases) for the Product Release. This includes the coding and configuration of software and related infrastructure (e.g., infrastructure as code).

This value stream stage covers the development of the changes that are part of the new release of the Digital Product based on the requirements, architecture boundaries, and policies set by the organization. In addition, requests might be needed to cater for setting up a development environment for the developers.

Table 5-11. Design & Develop Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Prioritized and refined Product Backlog • Iteration/sprint plan created • Identified Defects	• Committed code/configuration changes (stored in the source code repository) • Test Cases created • Requirements and acceptance criteria defined • Updated Product Design
Value Item:	
• Development or configuration of changes have been completed and committed to the source code repository (ready to be merged)	
Activities:	
• Shall define/analyze requirements (e.g., engage and collaborate with stakeholders) • Shall create and approve the Product Design, including data model design, User Experience (UX) design, and interface design • Shall define and create the Test Plan and associated Test Cases • Shall develop/configure the required changes (actual software development and/or configuring a product) • Shall commit code (into the source code repository)	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Business Analyst • Business Stakeholder • Consumer • Data Protection Officer • Development Team • Product Architect • Product Manager • Scrum Master • Security Analyst • Security Architect • Test Specialist • Vendor Manager	• Defect (*Defect Component*) • Policy (*Policy Component*) • Product Backlog Item (*Product Backlog Component*) • Product Design (*Product Design Component*) • Requirement (*Requirement Component*) • Source (*Source Control Component*) • Test Case (*Test Component*) • Test Plan (*Test Component*)

5.3.4. Build, Integrate, & Test Stage

Description

The purpose of the value stream stage “Build, Integrate, & Test” is to perform the build, integrate, and test activities to ensure the release package is ready for deployment into production.

Table 5-12. Build, Integrate, & Test Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Committed code/configuration changes (in the source code repository) • Test Cases created • Requirements and acceptance criteria defined • Updated Product Design	• New or updated Build Package • New or updated release package • Tests executed • Identified Defects/issues
Value Item: • Created and tested Product Release with associated Build Package	

Activities:

- May compile code (if applicable)
- Shall create the build artifacts as part of the Continuous Integration pipeline
- Shall perform code reviews (and static code analysis)
- Shall validate compliance and security aspects of the build created (and associated components)
- Shall create the release package
- Shall deploy to test (and other environments such an acceptance/staging environment)
- Shall perform tests and verify against policies and compliance requirements, this includes security testing, performance testing, regression testing, integration testing
- Shall identify potential Defects/issues (feedback loop to plan stage)

Examples of Participating Stakeholders:

- Business Stakeholder
- Consumer
- Consumer
- Development Team
- Product Architect
- Security Officer
- Test Specialist
- Vendor Manager

Participating Data Objects (*Component*):

- Build Package (*Build Package Component*)
- Fulfillment Book (*Fulfillment Component*)
- Pipeline (*Pipeline Component*)
- Product Release (*Release Composition Component*)
- Requirement (*Requirement Component*)
- Source (*Source Control Component*)
- Test Case (*Test Component*)
- Test Plan (*Test Component*)

5.3.5. Accept & Publish Release Stage

Description

The purpose of the value stream stage “Accept and Publish Release” is to accept the Product Release as a candidate for deployment into production. The required change authority, such as the Product Manager, accepts the Product Release for deployment and/or release into production. The Product Manager accepts the Product Release after engaging with relevant stakeholders, which includes reviewing test results to accept or reject the list of remaining Defects, and reviewing the collateral produced. In a DevOps model, Operations has been participating all along but at this point an approval is given for the Product Release to be deployed into production.

This value stream stage ensures only validated and approved release packages can be deployed into production. The release package is tagged and secured to prevent any modification at a later stage (e.g., to prevent unauthorized changes being made).

Table 5-13. Accept & Publish Release Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Tested and validated release package (ready for release)	• Accepted and published new Product Release (ready for deployment to production) • Securely stored and tagged Product Release • Release notes available (including outstanding Defects and associated Problems)
Value Item: • The Product Release is ready for deployment; the future consumer is informed and able to prepare for the release	
Activities: • Shall review readiness of the release (e.g., verify test results, verify outstanding Defects, code quality, deployment Runbooks, data migration, and backout procedure) • Shall accept the new Product Release (by the Product Manager and required stakeholders) • Shall inform stakeholders of the new release • Shall publish the final release package (make the release available for consumption)	
Examples of Participating Stakeholders: • Business Stakeholder • Consumer • Product Architect • Product Manager • Risk Analyst • Security Analyst • Security Officer • Support Specialist	**Participating Data Objects** (*Component*): • Defect (*Defect Component*) • Desired Product Instance (*Fulfillment Orchestration Component*) • Fulfillment Book (*Fulfillment Component*) • Log (*Monitoring Component*) • Order (*Order Component*) • Product Backlog Item (*Product Backlog Component*) • Product Release (*Release Composition Component*) • Requirement (*Requirement Component*) • Scope Agreement (*Proposal Component*) • Subscription (*Order Component*) • Test Case (*Test Component*) • Test Plan (*Test Component*)

5.4. Deploy Value Stream

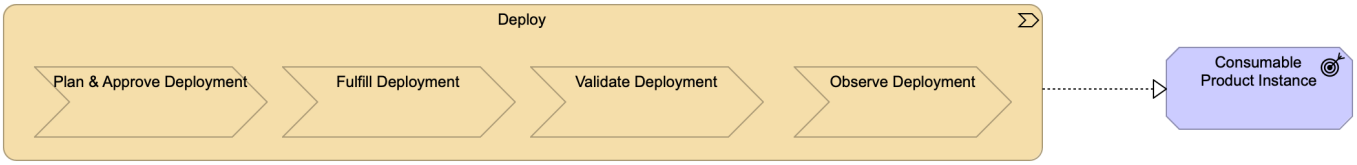

Figure 5-8. Deploy Value Stream Model

Overview

The value stream “Deploy”, as shown in Figure 5-8, is installing a Product Release into production. It can also imply the removal, disposal, or updating of an existing installed Product Release, so it is made available or unavailable to the customer. Many possible strategies for deploying a Product Release can be scenarios within the Deploy value stream; for example, deployment to a targeted audience, or deployment using a dark launch in which functionality is not directly released to consumers. If the user requirements and/or the characteristics of the Digital Product demand that the installed Product Release is improved, released, and deployed frequently, a drive to automate these value streams can emerge. To comply with regulatory requirements, organizations can require approvals and an audit trail of all the Changes made in production environments. The Deploy value stream ensures that all changes are tracked and related to a specific Product Release.

The Deploy value stream delivers the initial deployment, update, or removal of an Actual Product Instance, and will put the Service Contract into operation.

Primary Stakeholder

The main stakeholder of the Deploy value stream is the Product Manager, who can authorize deployment of Actual Product Instances to the customer in the production environment.

Value

The outcome of the Deploy value stream is that new Product Releases can be installed safely without causing unexpected outages to the current Actual Product Instances in use.

Cross-Value Stream Dependencies

The Deploy value stream depends on:

- Integrate:
  - Provides approved Product Releases that require deployment to the production environment
- Consume:
  - Delivers a foundation to work on, as an Order can be used to fulfill the underlying setup of a needed production environment

Value streams that depend on the Deploy value stream are:

- Evaluate:
  - Improvements start with drivers, that are expected to arise from all seven value streams

    All value streams will capture data to ensure that the quality of the Digital Product will meet the requirements.
- Release:
  - Deploy a context in the form of an Actual Product Instance in which the Service Offer can be consumed and supported
- Operate:
  - Actual Product Instances can be enabled, changed, disabled, or removed by the deployment of the Product Release

The Deploy value stream, as shown in Figure 5-9, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

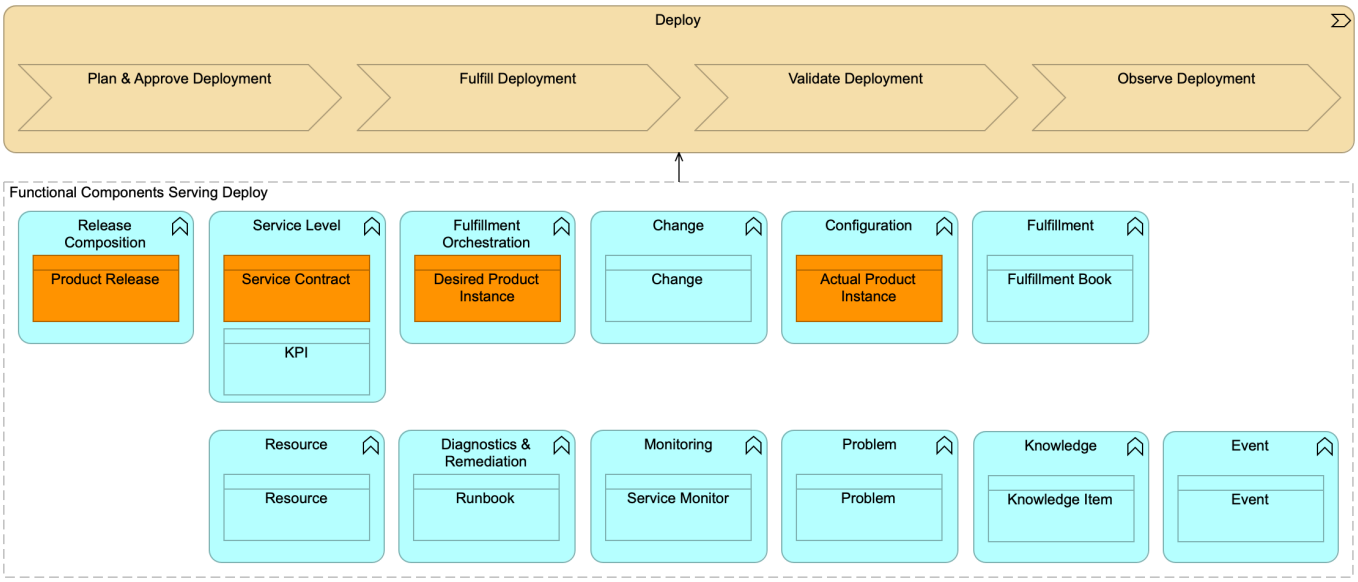

Figure 5-9. Deploy Value Stream Details Model

In the following sections we document the scenarios and stages of the Deploy value stream:

- Scenario: Deploy a First Product Release for a New Digital Product
- Scenario: Deploy a New Product Release Version for a Digital Product
- Scenario: Disable or Remove Existing Product Release Instances of a Digital Product
- Stage: Plan & Approve Deployment
- Stage: Fulfill Deployment

- Stage: Validate Deployment
- Stage: Observe Deployment

5.4.1. Deploy Scenarios

The Deploy value stream is applicable for the following three scenarios:

Deploy a First Product Release for a New Digital Product Scenario

The goal of this scenario is to plan, prepare, and execute the deployment of the initial version of the Digital Product. Part of the planning is determining the priority, assessing the risk and dependencies, and scheduling the time slot. The information used for the risk assessment will be limited as this is the first release of the Digital Product. However, during development, information from earlier deployments in non-production environments should provide the necessary information to minimize risk. Since no customer is using the Digital Product, available time slots might include office hours. The available capacity for dependent Digital Products can be checked, and IT assets requested if needed. The deployment schedule must consider any dependencies on the Orders required to allocate the necessary Resources.

During execution of the deployment, the Service Contract will be activated; this requires instrumentation of the data for the metrics that will be collected to measure the Key Performance Indicator (KPI) targets. Additional training of specialists involved in the Operate value stream are performed, although it is advisable to train specialists before the Product Release is finalized. Knowledge Items are created based on the Defects and test results (e.g., metrics from performance tests).

Deploy a New Product Release Version for a Digital Product Scenario

The goal of this scenario is to plan, prepare, and update the existing Actual Product Instances. This ensures a stable Product Instance after the Deploy value stream is completed. The deployment strategy may depend on the Digital Product: some might require a maintenance window in which deployment can be executed, while for others a completely new version will be installed alongside an existing version. As every deployment should result in a stable Product Instance in production, there are some points to address:

- To mitigate risk, the plan should include a “point of no return” in time, and a “rollback to the last known good configuration” plan

  These should be addressed and practiced during the development of the Product Release.

- During the execution of the deployment, stakeholders should be informed; for example, the monitoring events can be suppressed to prevent disrupting the Operate value stream

Disable or Remove Existing Product Release Instances of a Digital Product Scenario

The goal of this scenario is to completely or partly disable or remove Actual Product Instances of a Digital Product. The deployment design (which is part of the Product Release) should provide

information on the active Product Release that needs to be removed or undeployed, whereas the Operate value stream has the actual information on which related Actual Product Instances are to be disabled or removed.

Removing a Digital Product might require changes to the Product Design, and, in some cases, customer data may have to be retained for a while and made available for the customer to view. Some of the components of the Digital Product can be removed, but new components may also be installed or deployed.

5.4.2. Plan & Approve Deployment Stage

Description

The purpose of the value stream stage “Plan & Approve Deployment”` is to create a deployment plan for a Digital Product and, based on this plan, to sign it off with all relevant stakeholders.

The details of the plan and the extent of approval required depends on the complexity of the deployment.

Table 5-14. Plan & Approve Deployment Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• A Product Release in a fully-tested form	• An approved deployment plan
Value Item:	
• Ensuring that production deployment is not a surprise to any part of the organization, and that the resources needed for deployment are available	
Activities:	
• Shall estimate the time for deployment and associated service downtime if any • Shall ensure resources are available for the full deployment, including the validate and observation stage • Shall plan a deployment time based on effort, resources, and corporate policies • May validate Knowledge Items exist for open, accepted Defects and work-arounds that are part of the Product Release • Shall obtain necessary approval for deployment as per the organization policy This typically includes but is not limited to approval from: ◦ Operations teams for increased operations support ◦ Support teams for the potential increase of support needed, as well as training ◦ Security operations for sign-off on product security mandates ◦ Legal teams for any potential license and copyright issues with product	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Operations Manager • Product Manager • Release & Deployment Manager	• Actual Product Instance (*Configuration Component*) • Change (*Change Component*) • Desired Product Instance (*Fulfillment Orchestration Component*) • Product Release (*Release Composition Component*) • Service Contract (*Service Level Component*)

5.4.3. Fulfill Deployment Stage

Description

The purpose of the value stream stage “Fulfill Deployment” is to execute the Fulfillment Orchestration when the change window for deployment is open. The Fulfillment Orchestration is documented in the change records and can be coded as part of the Product Release to automate the Fulfillment Orchestration.

Table 5-15. Fulfill Deployment Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Signed-off preconditions for deployment • Validated Knowledge Item for Product Release	• Actual Product Instance is deployed in production
Value Item:	
• Deployed Actual Product Instance	

Activities:

- Shall create the Desired Product Instance topology from the Product Design
- Shall reserve the required Resources (e.g., hardware capacity, cloud resources, licenses, ...)
- Shall create Order(s) to cater for the deployment of any dependent services required
- May inform support specialists from the Monitoring bridge and the service desk
- Shall verify the orchestration for the Desired Product Instance and associated Fulfillment Books, and validate if all preconditions are met
- May suppress monitoring Events to prevent them from being escalated during the change window
- Shall deploy artifacts from the Build Package as part of the Product Release on the Actual Product Instance(s), and update the topology
- Shall deploy new or updates to Runbook(s)
- Shall deploy new or updates to Service Monitor(s)
- Shall deploy/activate the Knowledge Item(s) as part of the Product Release
- May start the Actual Product Instance(s)

Examples of Participating Stakeholders:

- Operations Manager
- Product Manager
- Release & Deployment Manager
- Release & Deployment Manager

Participating Data Objects (*Component*):

- Actual Product Instance (*Configuration Component*)
- Change (*Change Component*)
- Desired Product Instance (*Fulfillment Orchestration Component*)
- Fulfillment Book (*Fulfillment Component*)
- Knowledge Item (*Knowledge Component*)
- Problem (*Problem Component*)
- Product Release (*Release Composition Component*)
- Resource (*Resource Component*)
- Runbook (*Diagnostics & Remediation Component*)
- Service Monitor (*Monitoring Component*)

5.4.4. Validate Deployment Stage

Description

The purpose of the value stream stage “Validate Deployment” is to ensure that all aspects of the deployment activities were correctly executed. This includes checking that the Product Instance is indeed running and configured according to plan, and that the monitoring and documentation associated is configured and made available.

Table 5-16. Validate Deployment Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Deployed new or updated Product Instance	• It is validated that the Actual Product Instance is running
Value Item:	
• Ensure that the deployment went OK	
Activities:	
• Shall inspect/discover the deployed CIs registered in the Actual Product Instance Model • Shall validate that the discovered Actual Product Instance model is compatible with the Desired Product model • Shall validate that all the components of the Product Instance are running, and that the Service Monitor and Log systems are working for the Product Instance • Shall validate that the Service Level monitoring for the Product Instance is registered, and Monitoring is working • Shall validate that the known errors/Defects are registered in the known error system	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• External Stakeholder • Operations Manager • Product Manager • Release & Deployment Manager • Support Specialist	• Actual Product Instance (*Configuration Component*) • Change (*Change Component*) • Desired Product Instance (*Fulfillment Orchestration Component*) • Problem (*Problem Component*) • Product Release (*Release Composition Component*) • Resource (*Resource Component*) • Runbook (*Diagnostics & Remediation Component*) • Service Contract (*Service Level Component*)

5.4.5. Observe Deployment Stage

Description

The purpose of the value stream stage “Observe Deployment” is to ensure that the Actual Product Instance (= deployed product instance) is delivering expected functional and non-functional requirements.

Table 5-17. Observe Deployment Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Running and validated Actual Product Instance	• New or updated Actual Product Instance can be handed over to the Operations team
Value Item: • Actual Product Instance is operational	
Activities: • Shall observe Logs, Events, and service levels, and ensure they are stable and acceptable • May observe if consumption of the product is working through test consumption and the monitoring of help desk interactions • Shall consider if any issues require a quick patch forward or a rollback of deployment	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Business Stakeholder • Development Team • Product Manager	• Actual Product Instance (*Configuration Component*) • Actual Product Instance (*Configuration Component*) • Event (*Event Component*) • KPI (*Service Level Component*) • Log (*Monitoring Component*) • Service Contract (*Service Level Component*)

5.5. Release Value Stream

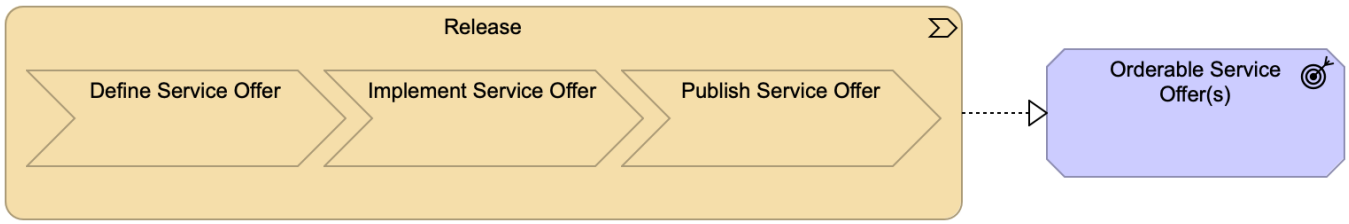

Figure 5-10. Release Value Stream Model

Overview

The value stream “Release”, as shown in Figure 5-10, ensures that the Service Offers for ordering a new, a change in, or the decommissioning of an Actual Product Instance or its support to them are offered to consumers. The Release value stream maintains the Service Offer Catalog, the catalog views, and the Service Offer promotion and information of available Service Offers to the consumer base so that the individual consumer can find and order Digital Products, or support for them. Service Offers should be made available for viewing and requests for access over multiple channels. This value stream publishes the Service Offers based on various triggers and inputs, such as:

- The Product Release, as a result of the Integrate value stream
- The decision of the Product Manager to provide a new (bundled) Service Offer based on existing Service Offers and releases
- The decision to change or terminate an existing Offer

The Release value stream delivers a digital service by means of a Service Offer that is made available and can be requested.

Primary Stakeholder

The primary stakeholder of the Release value stream is the Catalog Manager, who is primarily responsible for defining, implementing, and publishing the Service Offers so that they are available to be consumed by the target consumers.

Value

The outcome of the Release value stream is to ensure the management of the consumable Service Offers for the Digital Product. These Service Offers are for new service consumption; there are Service Offers for support or to change/retire an Actual Product Instance. This also provides the following additional value to the consumers but is not limited to:

- Centralization of the catalog (e.g., providing access control to all the products and services being offered to the consumers)
- Enabling self-service (e.g., the aggregated Service Offer Catalog facilitates user self-service capabilities by providing detailed information about the Service Offer Catalog item and its detailed attributes)
- Providing traceability (e.g., the Service Offer Catalog allows traceability of service, from ordering to fulfillment of the service)
- Reducing cost (e.g., the Service Offer Catalog reduces the time that the consumer requires from ordering to fulfillment of the service)

Cross-Value Stream Dependencies

The Release value stream depends on:

- Deploy:
  - Released Offers for the support or consumption of an Actual Product Instances (new, changed, or terminated) can depend on a deployed Product Release

Value streams that depend on the Release value stream are:

- Evaluate:
  - Improvements start with drivers, that are expected to arise from all seven value streams

    All Value Streams will capture data to ensure that the quality of the Digital Product will meet the requirements.

- Consume:
  - Service Offers that can be fulfilled are based on released Service Offers, and can only be activated after deployment of the release
  - The Consume value stream feeds the result of the deployment back into the Release value stream stages for reporting

The Release value stream, as shown in Figure 5-11, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

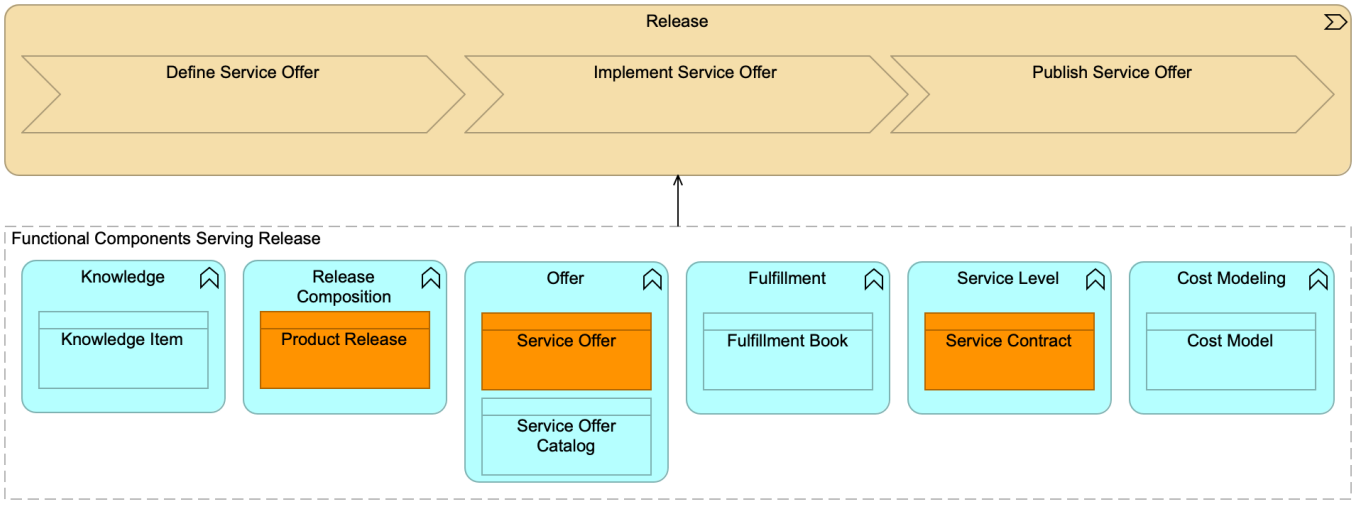

Figure 5-11. Release Value Stream Details Model

In the following sections we document the scenarios and stages of the Release value stream:

- Scenario: Release a New Service Offer
- Scenario: Change an Existing Service Offer
- Scenario: Bundle Existing Service Offers
- Scenario: Terminate a Service Offer
- Stage: Define Service Offer
- Stage: Implement Service Offer
- Stage: Publish Service Offer

5.5.1. Release Scenarios

The Release value stream is applicable for the following four scenarios:

Release a New Service Offer Scenario

The outcome of this scenario is to provide the consumers with the possibility to request a new Actual Product Instance of a Digital Product or to request a new form of support on an Actual Product Instance. The following activities are part of this scenario:

- Link the Service Offer to existing (via deploy/deployed) support/fulfillment plan(s); this includes linkage of the Service Offer to its internal and external support or fulfillment parties
- Provide customer access to the Offer (views)
- Promote and inform the availability of the new Service Offer

Change an Existing Service Offer Scenario

The outcome of this scenario is to maintain an up-to-date Service Offer for the consumers. The reason for the change of the Service Offer can vary. Recurring Change activities are:

- Change of charged service price; e.g., annual service fee recalculation
- Change of underlying terms and conditions; e.g., when the SLA changes
- Change of the fulfillment or support plan; e.g., to include more automation
- Change of access to the Service Offer; e.g., to change who can consume the service

Bundle Existing Service Offers Scenario

The outcome of this scenario is to combine multiple existing Service Offers into a single Service Offer, so that customers only need to order a single Service Offer, instead of multiple Service Offers. Activities in this scenario are:

- Combining and chronologically orchestrating/sequencing the Service Offers
- Providing customer access to the Service Offer (views)
- Promoting and informing the availability of the new bundled Service Offer

Terminate a Service Offer Scenario

The outcome of this scenario is to remove a Service Offer from the catalog and stop it from being ordered in the future. This scenario can be executed when the all-encompassing Digital Product is terminated, but it can also be executed in a normal release cycle of a Digital Product, when new versions of the Digital Product become available. Activities in this scenario are:

- Marketing and promotion of the termination of the Service Offer; this could include promotion to, for example, a new or replacement Product Release
- Change the Service Offer validity
- Remove customer access to the Service Offer (views)

5.5.2. Define Service Offer Stage

Description

The purpose of the value stream stage “Define Service Offer” is to obtain the Service Offer information from the Integrate value stream and to plan the creation and publication of Service Offers to Consumers. Offers can also represent the consumption of cloud services making the Service catalog an Aggregation catalog between their own and cloud services; planning should take place to identify which cloud Service Offers would be made available.

Table 5-18. Define Service Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• New or modified Product Release (from the Integrate value stream) • Request for Service Offer improvements (prices, Quality of Service (QoS), bundles, ...)	• Approved Service Offer update plan
Value Item: • The Service Offer demand is planned for its release	
Activities: • Shall take in new demand for Service Offers: ◦ New Service Offers ◦ New Service Offer bundles • May be based on consumer need, existing Service Offers, and demand captured: ◦ Service Offer changes ◦ Service Offer terminations • Shall validate completeness of offering information: ◦ Including availability of fulfillment and support plans, self-service Runbooks, additional Knowledge Management documents, or required customer training • Shall take in the catalog structure and view changes • May plan Service Offer release activities and the release date	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Catalog Manager • Release & Deployment Manager	• Cost Model (*Cost Modeling Component*) • Fulfillment Book (*Fulfillment Component*) • Knowledge Item (*Knowledge Component*) • Product Release (*Release Composition Component*) • Service Contract (*Service Level Component*) • Service Offer (*Offer Component*)

5.5.3. Implement Service Offer Stage

Description

The purpose of the value stream stage “Implement Service Offer” is to onboard the Service Offer into the Service Offer Catalog and to prepare all that is needed so that the Service Offer can be published.

Table 5-19. Implement Service Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Approved Service Offer release plan	• New, updated, or terminated Service Offer ready for publication
Value Item:	
• The Service Offer is ready for its release, and the future Consumer and Operations teams are informed of the release dates and can prepare for the release	

Activities:

- Shall create or update the Service Offer, where the Service Offer can have:
  - A description and validity period
  - A defined cost
  - A price and margin
  - Delivery terms and conditions
  - Payload options needed for the fulfillment (including feature toggles; all options for delivery are asked up front)
  - A defined target consumer audience
  - Additional training/knowledge documents for the target consumers
  - For delivery, references to a:
    - Service request fulfillment plan
    - Support request handling plan
    - An Actual Product Instance, system, or Subscription
    - Working alignment with Monitoring
- Shall maintain the following attributes of the Service Offer:
  - Service Offer information, including price list
  - Service access
  - Service Offer Catalog structure
  - Service Offer Catalog views
- May terminate the validity of the Service Offer
- Can inform relevant stakeholders (Consumer and Operations Managers) on the upcoming Service Offer release changes

Examples of Participating Stakeholders:

- Catalog Manager
- Development Team
- Operations Manager
- Product Manager
- Release & Deployment Manager
- Support Specialist
- Test Specialist

Participating Data Objects (*Component*):

- Knowledge Item (*Knowledge Component*)
- Service Offer Catalog (*Offer Component*)
- Service Offer (*Offer Component*)

5.5.4. Publish Service Offer Stage

Description

The purpose of the value stream stage “Publish Service Offer” is to activate or release the Service Offer so that it can be ordered by consumers.

Table 5-20. Publish Service Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• New, updated, or terminated Service Offer ready for publication	• Active new or updated Service Offer • Terminated Service Offer
Value Item: • Service Offer is released, and the future Consumer and Operations teams are informed on the release and can start using the Service Offer	
Activities: • Shall activate Offer at the valid-from date • May inform relevant stakeholders (Consumer and Operations Managers) on the released changes • Shall ensure the Service Offer is created in the catalog and: ◦ Shall be accessible by the target consumer group (supporting canary releases) ◦ Can be selected, configured, priced, requested, delivered, adopted, supported, modified, and terminated ◦ May be promoted through targeted informative activities to drive consumption ◦ May be sold through targeted business sales activities to drive consumption	
Examples of Participating Stakeholders: • Catalog Manager • Consumer • Development Team • Operations Manager • Product Manager • Release & Deployment Manager • Test Specialist	**Participating Data Objects** (*Component*): • Cost Model (*Cost Modeling Component*) • Knowledge Item (*Knowledge Component*) • Service Offer Catalog (*Offer Component*) • Service Offer (*Offer Component*)

5.6. Consume Value Stream

Figure 5-12. Consume Value Stream Model

Overview

The value stream “Consume”, as shown in Figure 5-12, manages the lifecycle of consuming a Digital Product. Digital Products are consumed as services and the lifecycle is managed as a Subscription to the service.

The Consume value stream covers the entire set of activities of a consumer, from selecting a Service Offer, ordering the service, fulfillment of the service, getting support, and status on the service.

The consumer can be a person internal to the business or an external customer. The Consumer can also be a system actor that on behalf of an organization consumes the service.

The Consume value stream ensures the ordered Desired Product Instance will be delivered as an Actual Product Instance within the agreed terms. The value stream is responsible for all activities needed to deliver the Actual Product Instance(s) successfully or provide support for them to Consumers.

If ordered, the Consume value stream will create or update the Service Contract. To increase its delivered value and identify areas for improvement, the value stream will also measure the consumption of the Service Contract, and assess the ongoing needs of the stakeholder. These actions support the offering, deployment, and monitoring of the Actual Product Instance, ensuring commitment to the Service Contract.

Primary Stakeholder

The main stakeholder of the Consume value stream is a customer and/or consumer (human or system actor) in need of a Digital Product.

Value

The outcome of the Consume value stream is to have customers able to locate desired Digital Product service offerings, and obtain them in a timely manner as deployed and instantiated Digital Products that will be supported by the Operate value stream, according to a Service Contract.

Cross-Value Stream Dependencies

The Consume value stream depends on:

- Release:
  - Offers are based on releases, and can only be activated after deployment of the release
  - The Consume value stream feeds the result of the deployment back into the Release value stream stage for reporting
- Operate:
  - Fulfillment feedback of a Service Offer that resulted in an Incident and needed to be handled in the Operate value stream

Value streams that depend on the Consume value stream are:

- Evaluate:
  - Improvements start with drivers, that are expected to arise from all seven value streams

    All value streams will capture data to ensure that the quality of the Digital Product will meet the requirements.
- Integrate:
  - A request can be used to fulfill the setup of a test or build environment
- Deploy:
  - A request can be used to fulfill the underlying platform setup of a needed production environment
- Operate:
  - Actual Product Instances can be enabled, changed, or disabled by the fulfillment of the Service Offer
  - Consumption of a Service Offer can result in an Incident that needs to be handled in the Operate value stream

The Consume value stream, as shown in Figure 5-13, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

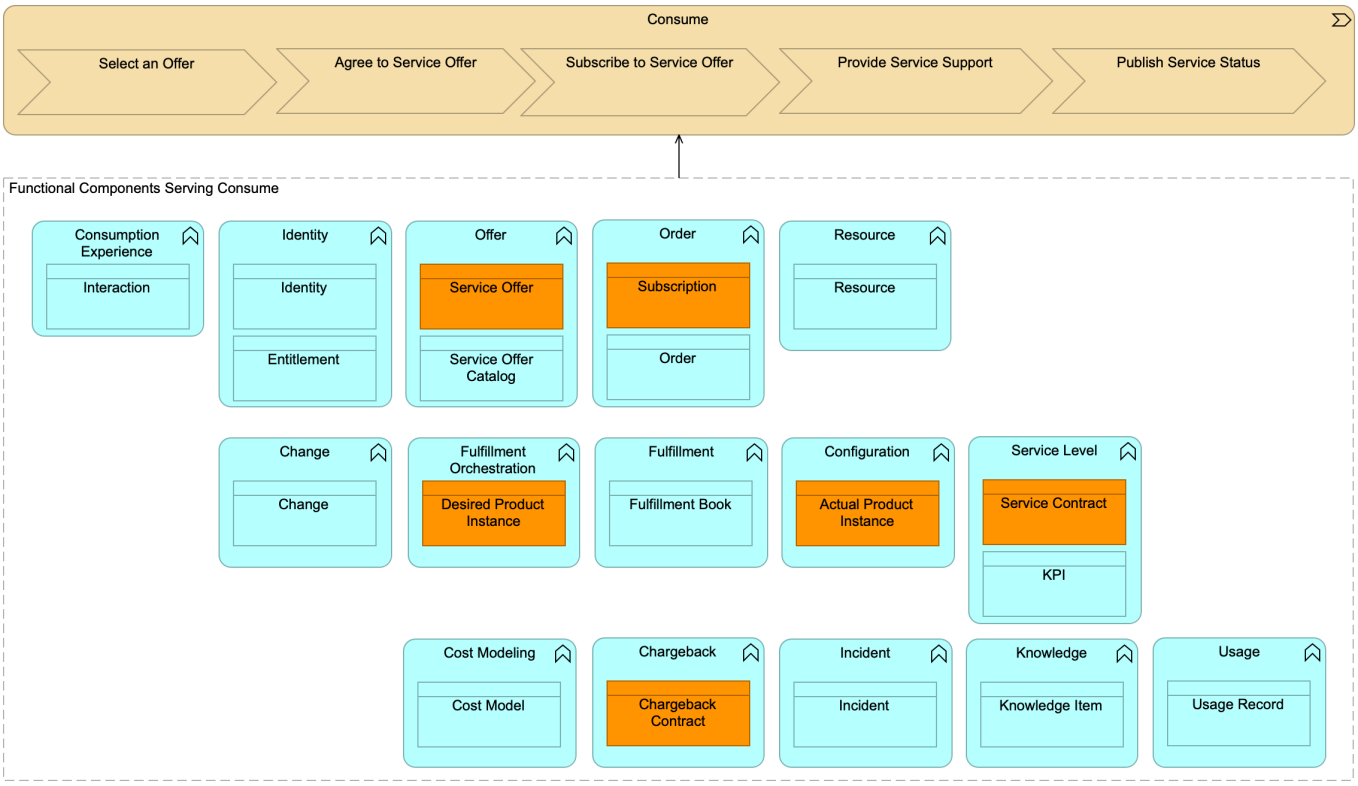

Figure 5-13. Consume Value Stream Details Model

In the following sections we document the scenarios and stages of the Consume value stream:

- Scenario: Order a New Desired Product Instance
- Scenario: Order an Actual Product Instance Modification
- Scenario: Order an Actual Product Instance Termination
- Scenario: Order an Actual Product Instance Support
- Stage: Select an Offer
- Stage: Agree to Service Offer
- Stage: Subscribe to Service Offer
- Stage: Provide Service Support
- Stage: Publish Service Status

5.6.1. Consume Scenarios

The Consume value stream is applicable for the following four scenarios:

Order a New Desired Product Instance Scenario

The outcome of this scenario is to look for, select, and obtain a new Actual Product Instance by means of ordering a new Desired Product Instance based on an available Service Offer; for example:

- New Subscription
- Stage a Build Package to an Actual Product Instance
- Access to a business application

Order an Actual Product Instance Modification Scenario

The outcome of this scenario is to look for, select, and obtain a modification to an Actual Product Instance; for example:

- Changing a Subscription
- Changing an asset
- Changing an Actual Product Instance
- Lifecycle management actions to an Actual Product Instance (e.g., start, stop, scale up)
- Changing access to a business application

This scenario as such will influence the usage (and chargeback) of the Actual Product Instance.

Order an Actual Product Instance Termination Scenario

The outcome of this scenario is to terminate a running Actual Product Instance; for example:

- End a Subscription
- Return or decommission an asset
- Decommission an Actual Product Instance
- End access to an application

Order an Actual Product Instance Support Scenario

The outcome of this scenario is to look for, select, and obtain support for a running Actual Product Instance; like a support request. This scenario supports omni-channel principles to get access to the desired support. Next to that, self-service concepts are used to limit the interaction flow to people and provide direct remediation.

5.6.2. Select an Offer Stage

Description

The purpose of the value stream stage “Select an Offer” is to help the customer to select the Digital Product Service Offers(s) best suited to their needs.

Table 5-21. Select an Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Customer searches for a Service Offer	• Service Offer selected
Value Item:	
• Consumption Experience channel available to the customer • Service Offers available to the customer	
Activities:	
• May advertise across channels and making customers aware of the Digital Products • Shall display Service Offers to present Digital Products in a searchable form • Shall enable selection filtering and assessments of the best product(s) matched to customer needs	
Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Catalog Manager • Consumer • Product Manager • Release & Deployment Manager	• Entitlement (*Identity Component*) • Identity (*Identity Component*) • Interaction (*Consumption Experience Component*) • Service Offer Catalog (*Offer Component*) • Service Offer (*Offer Component*)

5.6.3. Agree to Service Offer Stage

Description

The purpose of the value stream stage “Agree to Service Offer” is to get agreement from the customer to the terms and conditions of the Service Contract.

Table 5-22. Agree to Service Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Service Offer selected	• Service Contract agreed
Value Item: • Active Service Contract • Active Subscription • Delivery commitment	
Activities: • Shall finalize the contractual agreement, and have the customer accept the terms and conditions of the Service Contract as part of the Service Offer and delivered Product Instance	
Examples of Participating Stakeholders: • Business Stakeholder • Product Manager • Vendor Manager	**Participating Data Objects** (*Component*): • Interaction (*Consumption Experience Component*) • Order (*Order Component*) • Service Contract (*Service Level Component*)

5.6.4. Subscribe to Service Offer Stage

Description

The purpose of the value stream stage “Subscribe to Service Offer” is to make the Actual Product Instance, as result of the Order and Desired Product Instance, available to the customer, in such a way that the customer is able to consume the Digital Product.

Table 5-23. Subscribe to Service Offer Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Agreed Service Contract	• The Actual Product Instance is ready to be consumed • Customer consumes the Actual Product Instance
Value Item: • The Actual Product Instance available to and used by the customer	

Activities:	
• Shall orchestrate the Desired Product Instance fulfillment by orchestrating all necessary changes that enable the customer to consume the Digital Product Note: Orchestration can trigger Fulfillment Books that are also used within the Deploy value stream. • Shall order and deliver the Actual Product Instance and provide the customer access to the Digital Product • Can measure the Service Contract against the agreed terms and conditions	
Examples of Participating Stakeholders: • Consumer • Product Manager • Service Desk Agent • Vendor Manager	**Participating Data Objects** (*Component*): • Actual Product Instance (*Configuration Component*) • Change (*Change Component*) • Chargeback Contract (*Chargeback Component*) • Cost Model (*Cost Modeling Component*) • Desired Product Instance (*Fulfillment Orchestration Component*) • Fulfillment Book (*Fulfillment Component*) • Order (*Order Component*) • Resource (*Resource Component*) • Service Contract (*Service Level Component*) • Subscription (*Order Component*)

5.6.5. Provide Service Support Stage

Description

The purpose of the value stream stage “Provide Service Support” is to deliver support to a consumer of an active Subscription.

Table 5-24. Provide Service Support Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Subscription created or changed for a service	• Support provided
Value Item: • Support for Subscriptions	

Activities: • Shall provide self-help access to knowledge related to the service • Shall provide a mechanism to request for help with using the service • Shall provide a mechanism for reporting issues • The service support shall provide: ◦ Resolutions to reported issues ◦ Escalation to the Operations team for underpinning operational issues ◦ Feedback to the Development team for issues that identify product improvement opportunities	
Examples of Participating Stakeholders: • Consumer • Support Specialist • Vendor Manager	**Participating Data Objects** (*Component*): • Incident (*Incident Component*) • Interaction (*Consumption Experience Component*) • Knowledge Item (*Knowledge Component*) • Subscription (*Order Component*)

5.6.6. Publish Service Status Stage

Description

The purpose of the value stream stage "Publish Service Status" is to deliver status to a consumer on an active Subscription.

Table 5-25. Publish Service Status Value Stream Stage

Entrance Criteria: • Subscription created or changed for a service	**Exit Criteria:** • Status calculated and communicated to the consumer
Value Item: • Status of consumed service is up-to-date	

Activities:	
• Shall collect and calculate usage statistics • Shall provide KPI calculations on service delivery according to the Service Contract • Shall provide notification to the relevant service consumer to inform them if there are any ongoing service issues • May proactively predict if a service will have any impacting event in the future, and communicate this to the consumer	
Examples of Participating Stakeholders: • Consumer • Product Manager	**Participating Data Objects** (*Component*): • Chargeback Contract (*Chargeback Component*) • Chargeback Record (*Chargeback Component*) • Incident (*Incident Component*) • KPI (*Service Level Component*) • Service Contract (*Service Level Component*) • Subscription (*Order Component*) • Usage Record (*Usage Component*)

5.7. Operate Value Stream

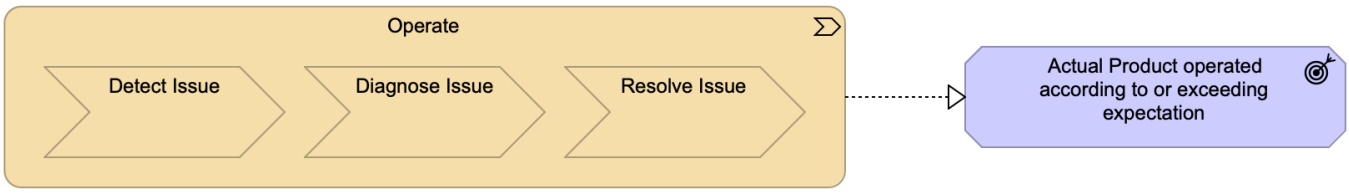

Figure 5-14. Operate Value Stream Model

Overview

The value stream “Operate”, as shown in Figure 5-14, monitors and ensures that the availability and performance of Actual Product Instances are within the boundaries of their agreed Service Contract and Key Performance Indicator (KPI) targets. The scope of this value stream includes managing any compliance and security aspects of running Digital Product Instances and underlying systems that may result from deployment and/or fulfillment. The IT4IT Standard describes how to achieve this using integrated, automated, or fully autonomous data flows between the event monitoring (Event, Incident, Change, Configuration, and Problem Management) functions of the digital services. In addition, the data flows may be designed to be reactive, proactive, or predictive, and include a retrospective.

The Operate value stream delivering the Actual Product Instance is operated in a sustainable way within the agreed terms and conditions of a Service Contract.

Primary Stakeholder

The primary stakeholder of the Operate value stream is the Consumer, who can use an Actual Product Instance or system within the boundaries of the Service Contract.

Value

The outcome of the Operate value stream is to ensure that the Actual Product Instance operates, and can be consumed as agreed in the terms of the Service Contract.

Cross-Value Stream Dependencies

The Operate value stream depends on:

- Deploy:
  - Actual Product Instances can be enabled, changed, or disabled by the deployment of the Product Release
- Consume:
  - Actual Product Instances can be enabled, changed, or disabled by the fulfillment of the Service Offer
  - Consumption of a Service Offer can result in an Incident that needs to be handled in the Operate value stream

Value streams that depend on the Operate value stream are:

- Evaluate:
  - Improvements start with drivers that are expected to arise from all seven value streams

    All value streams will capture data to ensure the quality of the Digital Product will meet the requirements. Quality is an aspect of each task within each value stream. The Evaluate value stream uses data produced in all value streams to evaluate drivers.
- Integrate:
  - Problems related to Actual Product Instances can require a fix from development (system defect)
- Consume:
  - Fulfillment feedback of a Service Offer that resulted in an Incident and needed to be handled in the Operate value stream

The Operate value stream, as shown in Figure 5-15, is the more comprehensive view for included scenarios and detailed value stream stages for the value stream, and will be described in detail in the following sections.

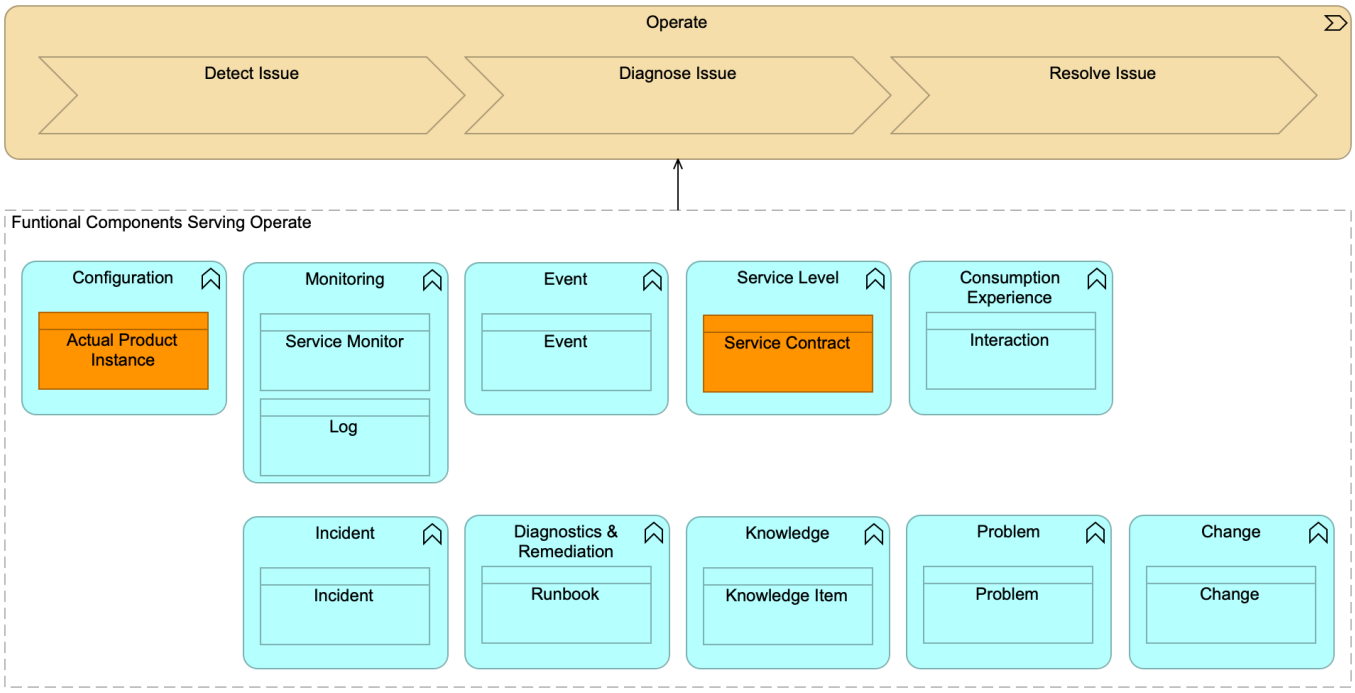

Figure 5-15. Operate Value Stream Details Model

In the following sections we document the scenarios and stages of the Operate value stream:

- Scenario: Remediate an Identified Back-End Issue
- Scenario: Ensure Disaster Recovery Objectives
- Scenario: Remediate a major Event or Incident
- Scenario: Remediate Consumer Front-End Issue
- Scenario: Perform Scheduled Maintenance
- Stage: Detect Issue
- Stage: Diagnose Issue
- Stage: Resolve Issue

5.7.1. Operate Scenarios

The Operate value stream is applicable for the following five scenarios:

Remediate an Identified Back-End Issue Scenario

In this scenario, the Operate value stream is triggered by an issue (a breach and/or vulnerability) or a potential future issue (triggered by a threshold breach) in the infrastructure, platforms, applications, user transactions, and/or behavior of the back-end network. The scenario ends with the outcome that the service remains operational without affecting consumer experience and within the boundaries of the Service Contract.

The activities undertaken in the scenario include the detection, diagnosis, evaluation, and resolution of issues, alerts, and anomalies related to the performance, compliance, and security aspects of the organization's back-end.

- Performance remediation examples include: "reactive" event detection and correlation through the monitoring of the Digital Product Instance, system, and topology; "pro-active" event detection through test transactions or built-in self-healing capabilities; or "predictive" event detection, done by generating test events from the analysis of historical monitoring data and other data sources (e.g., business performance, social media, weather) to identify potential degradation based on predicted consumer behavior
- Security remediation examples include responses to different types of attack, including unauthorized system access, phishing, password cracking, spyware, malware, Denial of Service (DoS) attacks, and more
- Compliance remediation examples include inspections to detect compliance with the rules; for example, adherence to legislation, statutes, regulations, standards, policies, contracts, processes, procedures, and operational controls

Ensure Disaster Recovery Objectives Scenario

This scenario is about ensuring that any capabilities listed in the business continuity/disaster recovery plan can work within the boundaries of the Service Contract. The activities in this scenario, which are planned and triggered by the business continuity and/or disaster recovery plan activities calendar, include: detection, diagnosis, and remediation tasks related to the plan. The purpose of the activities is to ensure that recovery objectives (e.g., Maximum Acceptable Outage (MAO), Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO)) remain within the boundaries of the agreed Service Contract and BIA. Real-world examples of these activities include performing backups, ensuring plans and mechanisms for graceful degradation, removing Single Points Of Failure (SPOFs), validating recovery libraries, performing redundancy validations, testing backup sites, and performing running (scheduled or random) disaster recovery simulations.

Remediate a Major Event or Incident Scenario

This scenario is triggered by a major event (e.g., full system outage, significant application failure); its outcome is that the service is restored within the boundaries of the Service Contract. The activities undertaken in the scenario include the correlation of Events/Incidents to identify (detect) that a major Event/Incident has occurred, collaboration with multiple teams to perform research and analysis to determine the underlying cause of the issue and to establish a remediation plan (diagnose), and the execution of remediation plans by those involved to resolve the major Event/Incident and restore services.

Remediate Consumer Front-End Issue Scenario

This scenario is triggered when an issue is raised by or through the service desk. The outcome of the scenario is restored consumer experience, normal service operations, and service delivery to the consumer within the boundaries of the Service Contract. The activities of the scenario include

capturing/detecting, diagnosing, resolving, and evaluating any service interactions raised and escalated by consumers or the service desk. This might include service interactions which have been generated by users or by the service desk, and have been subsequently escalated into an Incident, requiring second or third-line handling for remediation. In cases where handling is not or cannot be performed due to operational decisions or lack of capacity, the risk of working outside of agreed boundaries will be logged and communicated with the consumer and can be input into the Evaluate value stream.

Perform Scheduled Maintenance Scenario

This scenario is triggered when a planned activity is scheduled or when certain thresholds are met which trigger an alert/event for proactive maintenance. The planned activity or maintenance is reviewed and scheduled. Manual or automated action is taken to complete the activity/maintenance.

5.7.2. Detect Issue Stage

Description

The purpose of the value stream stage “Detect Issue” is to detect issues in an Actual Product Instance or system that has caused or may cause degradation of the service or breach the agreed service levels, then to prioritize and assign them for further diagnosis in a structured and repeatable manner.

Table 5-26. Detect Issue Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Monitoring and analytical tooling continuously monitors an Actual Product Instance or system for issues or triggers for proactive or self-healing maintenance activities • The service desk identifies and raises issues based on support interactions	• Prioritized list of issues and/or maintenance activities assigned to a resolver group to diagnose or review
Value Item: • The Actual Product Instance is continuously monitored for anomalies, issues, or alerts for maintenance activities • The service desk supports consumers of the Actual Product Instance and assigns Incidents	

Activities:

- Should proactively monitor the Actual Product Instance to:
  - Discover the Actual Product Instance and system topology
  - Track usage to support charging and licensing
  - Detect current or potential future issues in the Actual Product Instance or system that may impact:
    - Agreed service performance levels
    - Service security
    - Compliance policies
  - Identify and report patterns and trends
  - Detect incoming issues from the service desk or reported through a self-service mechanism
- Shall create Events and/or Incidents when defined policies or thresholds are breached
- Should correlate Events and/or Incidents with other Events and/or Incidents to determine impact for the Digital Product Instance or system
- Shall assign prioritized Events and/or Incidents to a resolver group considering the impact on value realization based on a complete and up-to-date overview of all managed Events or Incidents

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Compliance Manager • Consumer • Development Team • Security Analyst • Service Desk Agent • Support Specialist	• Actual Product Instance (*Configuration Component*) • Event (*Event Component*) • Incident (*Incident Component*) • Interaction (*Consumption Experience Component*) • Log (*Monitoring Component*) • Service Contract (*Service Level Component*) • Service Monitor (*Monitoring Component*)

5.7.3. Diagnose Issue Stage

Description

The purpose of the value stream stage “Diagnose Issue” is to prepare the best/quickest course of action to retain or restore normal operations. Further diagnosis can be performed to identify the root cause(s) of a potential or detected issue.

Table 5-27. Diagnose Issue Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Prioritized list of issues and/or maintenance activities assigned to a resolver group to diagnose or review	• Validated course of action as a (first) measure for issue prevention or issue resolution is defined and assigned • Actual Product Instance or system problem areas identified by the problem analyst, as input for the Development team Product Backlog
Value Item: • Course of action defined for issue prevention or remediation and assigned to the resolution group	
Activities: • Should understand severity, business impact, escalation paths, and find fixes by consulting the knowledge base or accessing up-to-date information from the Digital Product lifecycle • Should find quick fixes and/or work-arounds • Should identify root causes • Shall propose a course of action or activities needed to start, change, restore, or stop service operation levels • Shall approve the proposed course of action steps as a valid measure to resolve an Incident or to retain/resolve service operation	
Examples of Participating Stakeholders: • Chief Security Officer • Consumer • Development Team • Risk Analyst • Security Analyst • Service Desk Agent • Vendor Manager	**Participating Data Objects** (*Component*): • Actual Product Instance (*Configuration Component*) • Event (*Event Component*) • Incident (*Incident Component*) • Knowledge Item (*Knowledge Component*) • Log (*Monitoring Component*) • Problem (*Problem Component*) • Runbook (*Diagnostics & Remediation Component*) • Service Contract (*Service Level Component*)

5.7.4. Resolve Issue Stage

Description

The purpose of the value stream stage “Resolve Issue” is to execute the (prepared) course of action to retain, restore, and confirm normal service operations.

Table 5-28. Resolve Issue Value Stream Stage

Entrance Criteria:	**Exit Criteria:**
• Validated course of action as a (first) measure for issue prevention or issue resolution is defined and assigned	• Resolved issue for the Actual Product Instance or system • Resolution acceptance is asked to the consumer • The consumer provides feedback based on the resolution by the team or team member • Updated Knowledge Item usability rating
Value Item: • Service level for the Actual Product Instance or system is maintained or restored • Consumers have not been impacted or can continue to use the Actual Product Instance or system	
Activities: • Shall ensure everything is in place to resolve Incidents and/or related Events • You should collaborate on the execution of the course of action (e.g., a Pipeline or Runbook) and implementation of changes for the Actual Product Instance or system where applicable • Should validate success by performing post-execution checks and obtaining consumer acceptance • Shall close related Incidents, Interactions, and/or Events	

Examples of Participating Stakeholders:	**Participating Data Objects** (*Component*):
• Chief Security Officer • Consumer • Data Protection Officer • Development Team • Security Analyst • Service Desk Agent • Vendor Manager	• Actual Product Instance (*Configuration Component*) • Change (*Change Component*) • Event (*Event Component*) • Incident (*Incident Component*) • Interaction (*Consumption Experience Component*) • Knowledge Item (*Knowledge Component*) • Log (*Monitoring Component*) • Runbook (*Diagnostics & Remediation Component*) • Service Contract (*Service Level Component*)

Chapter 6. Strategy to Portfolio Functions

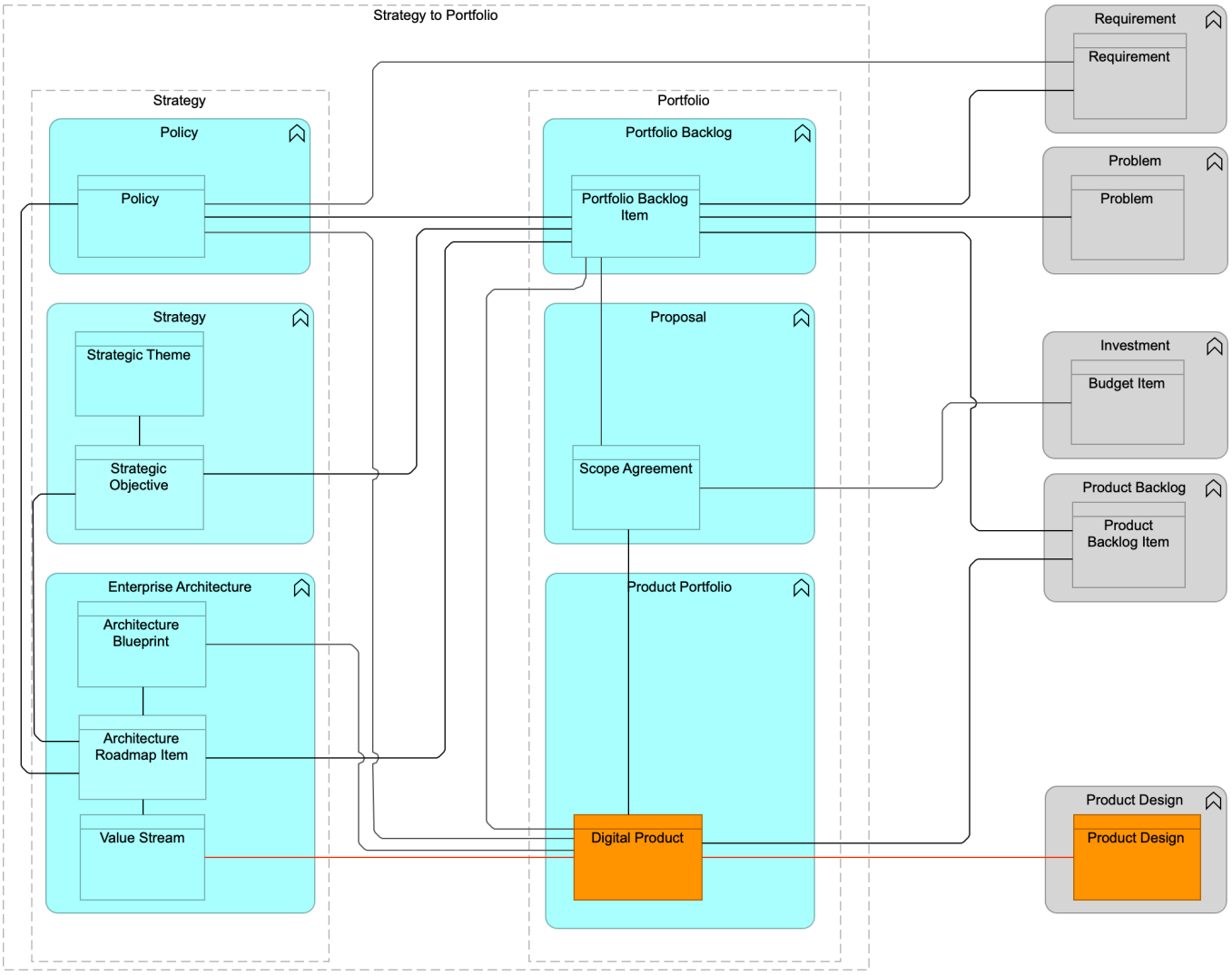

Figure 6-1. Strategy to Portfolio Functions Model

Description

The Strategy to Portfolio functions, informally referred to as “Plan”, provide a framework for interconnecting the functions and data involved in managing a portfolio of Digital Products, enabling effective prioritization of proposed investments. The IT4IT approach replaces project-based IT investment practices in which unfiltered user demands often trigger suboptimal projects and most investment is for run costs that cannot be allocated to any particular user outcome.

The Digital Product data object is the business description of a Digital Product and lies at the heart of Strategy to Portfolio functions. It records financial and operational outcomes, both as planned and as monitored.

The Product Portfolio manages this data over the lifecycle of the Digital Product. Strategy to Portfolio functions manage the data governance, consistency, flows, and integrations needed to assure that Digital Product definitions are complete, robust, and actionable, and that they remain aligned and traceable to the organization's Strategies, Policies and Roadmaps.

Strategy to Portfolio functions enable the filtering and rationalization of demand signals into consistently defined Portfolio Backlog Items associated with new or updated Digital Product definitions.

Evaluation of proposals to implement specific sets of Portfolio Backlog Items packaged with appropriate funding are formulated as Scope Agreements. Decision-makers can easily compare and prioritize these because Strategy to Portfolio functions enforce consistent data structures as well as traceability to Strategy and Policy value drivers. The Proposal function records these Scope Agreements and manages them over the life of the affected Digital Product(s).

The information created and managed by the Strategy to Portfolio functions enables the effective design, creation, traceability, and management of strategy-aligned Product Releases by the Requirement to Deploy functions over the life of each Digital Product.

The Strategy to Portfolio functions contain the following functional components:

- Strategy:
  - Policy component
  - Strategy component
  - Enterprise Architecture component
- Portfolio:
  - Portfolio Backlog component
  - Proposal component
  - Product Portfolio component

Related Value Streams

The following value streams use one or more functional components from the Strategy to Portfolio functions:

- Evaluate
- Explore

Business Benefits

The Strategy to Portfolio functions describe a prescriptive framework of required functional components and data objects so organizations can better control Strategy alignment to the Investment and Product Portfolio functional components.

The main benefits of using the Strategy to Portfolio functions are:

- Holistic portfolio view across the Strategy, Enterprise Architecture, Portfolio Backlog, Proposal, and Product Portfolio functional components
- Portfolio decisions based on business priorities
- Product lifecycle tracking through conceptual, logical, and physical domains
- Re-balance investments between strategic and operational demand
- Prioritized investment based on all portfolio facets including cost/value analysis, impacts on architecture, product/service roadmap, business priorities, and feasibility
- Solid communication with business stakeholders through Scope Agreements and roadmaps

6.1. Strategy Function

The Strategy Function is centered around managing and evolving the digital strategy.

6.1.1. Policy Functional Component

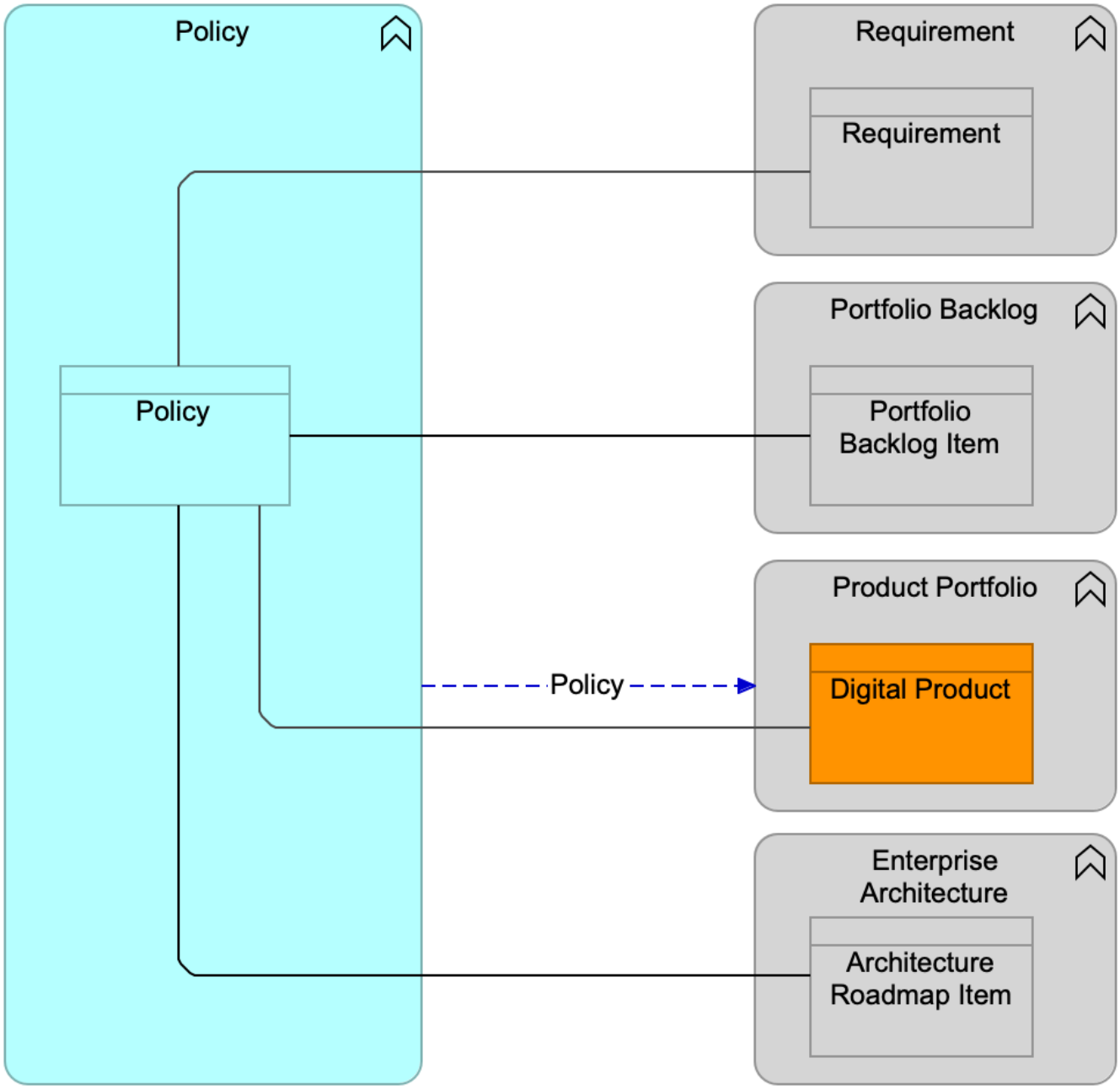

Figure 6-2. Policy Functional Component Model

Purpose

The Policy functional component manages the creation, review, approval, and audit of Policies.

The Policy functional component supports the value streams:

- Evaluate
- Explore
- Integrate

Functional Criteria

The Policy functional component:

- Shall align and map to Enterprise Architecture
- Shall enable the review and approval of Policies based on roles and responsibilities
- Shall provide visibility into Policy attributes such as types, status, non-compliance, audit history, risks, and issues
- Shall manage the governance of Policies associated with downstream value stream stages of the product lifecycle
- Shall manage both internal and external Policies (e.g., security, pricing/costing, resources, regulatory, risk, information) and history of revisions
- Shall track Policy exceptions and waivers with consistent identification, evaluation, and status reports for managing corrective actions

6.1.1.1. Policy Data Object

Purpose

The Policy data object defines and maintains a Policy to the digital organization.

Key Attributes

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

- **Id:** unique identifier of the Policy
- **Name:** name of the Policy
- **Description:** description/details of the Policy
- **Applicable Geography:** to which geographies the Policy applies; 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 Digital Product (n:m): multiple Policies might be applicable for a single product or service, or a single Policy may be applicable for multiple products or services
- Policy to Architecture Roadmap Item (n:m): multiple Policies might be applicable to the Architecture Roadmap
- Policy to Portfolio Backlog Item (n:m): multiple Portfolio Backlog Items may be sourced from Policies or may reference Policies to demonstrate compliance
- Policy to Requirement (n:m): multiple Requirements may be sourced from Policies or may reference Policies to demonstrate compliance

6.1.2. Strategy Functional Component

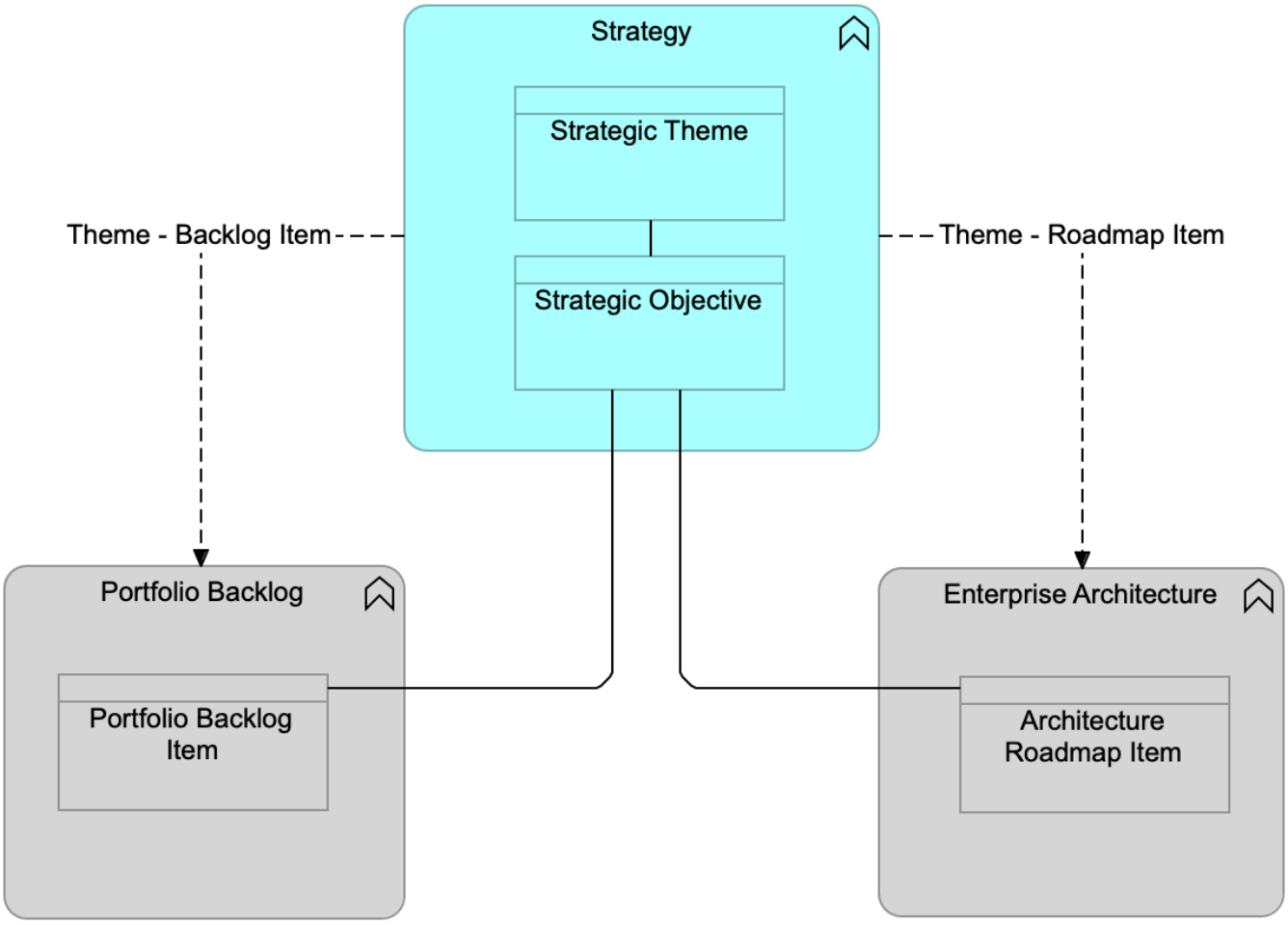

Figure 6-3. Strategy Functional Component Model

Purpose

The Strategy functional component manages the creation and maintenance of the portfolio strategy in alignment with the overall business/enterprise strategy.

The Strategy functional component supports the value streams:

- Evaluate
- Explore

Functional Criteria

The Strategy functional component:

- Must align to business objectives and goals
- Shall enable periodic review of the portfolio Strategic Themes
- Shall manage the creation and closure of Strategic Objectives as required to address Strategic Themes
- Shall provide visibility of both Strategic Themes and initiatives to as broad an audience as possible
- Shall ensure traceability of the strategic initiative to the impacted value streams
- Must maintain alignment with the Architecture Roadmap(s)

6.1.2.1. Strategic Theme Data Object

Purpose

The Strategic Theme data object is a statement of direction and priorities that summarize Strategic Objectives to provide business context for portfolio decision-making.

Key Attributes

The Strategic Theme data object shall have the following key data attributes:

- **Id:** unique identifier; e.g., number of the Strategic Theme
- **Name:** descriptive name of the Strategic Theme
- **Description:** description/details of the Strategic Theme

Key Data Object Relationships

- Strategic Theme to Strategic Objective (n:m): a Strategic Theme may contribute to the creation of many strategic initiatives; a strategic initiative could be created to address objectives associated to more than one Strategic Theme

6.1.2.2. Strategic Objective Data Object

Purpose

The Strategic Objective data object defines specific actionable goals and measurable outcomes for identified stakeholders and should be related to/or provide the most important results for the business case.

Key Attributes

The Strategic Objective data object shall have the following key data attributes:

- **Id:** unique identifier; e.g., number of the Strategic Objective
- **Name:** descriptive name of the Strategic Objective
- **Description:** description/details of the Strategic Objective

Key Data Object Relationships

- Strategic Objective to Strategic Theme (n:m): a Strategic Objective defines details of Strategic Themes; in some instances a given objective will deliver on several themes
- Strategic Objective to Portfolio Backlog Item (n:m): a Portfolio Backlog Item can be part of delivering a Strategic Objective and thus this relation can indicate the importance of a Portfolio Backlog Item as well as indicating how an objective is to be delivered
- Strategic Objectives to Architecture Roadmap Item (n:m): define what activities will deliver a given objective

6.1.3. Enterprise Architecture Functional Component

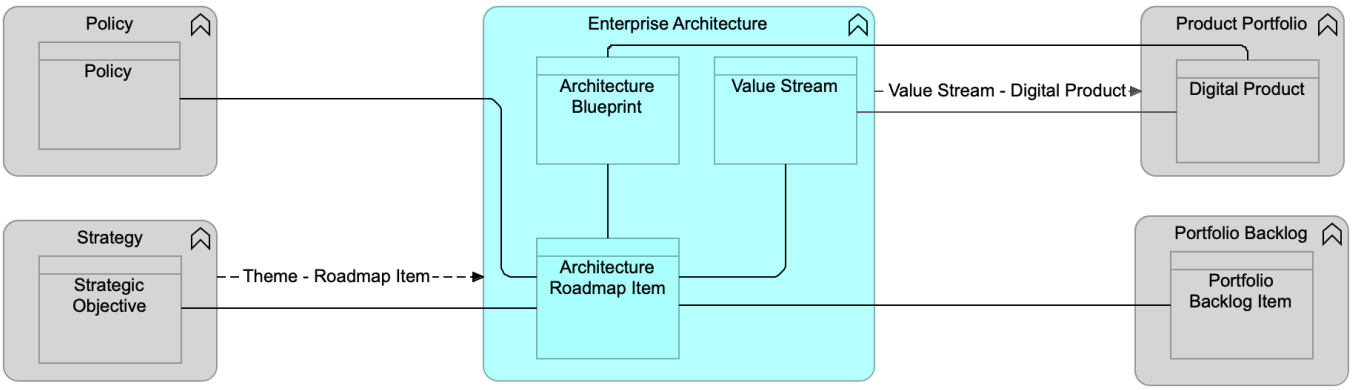

Figure 6-4. Enterprise Architecture Functional Component Model

Purpose

The Enterprise Architecture functional component manages Enterprise Architectures that will ensure a coherent and optimized portfolio of Digital Products that address enterprise business needs on a long-term basis.

The Enterprise Architecture functional component supports the value streams:

- Evaluate
- Explore

Functional Criteria

The Enterprise Architecture functional component:

- Shall be the system of record (authoritative source) for all Architecture Roadmaps and Architecture Roadmap Items
- Shall develop an Architecture Vision, based on business strategy, goals, and high-level Portfolio Backlog Items that define the capabilities and business value to be delivered
- Shall develop or use existing baseline state Business, Information, Application, Technology, and Security architectural artifacts based on stakeholder assessments of the current state
- Shall develop target state Business, Information, Application, Technology, and Security architectural artifacts based on strategies, principles, and policies
- Shall identify gaps between Baseline and Target Architectures
- Shall create an architecture definition document (or Architecture Blueprint)
- Shall identify the migration strategy
- May identify intermediate Architecture Roadmap Items that will deliver continuous business value, if an incremental approach is required
- Shall identify reusable building blocks, which are a package of functionality designed to meet business needs across the organization

  The building blocks can either be Architecture Building Blocks (ABBs) or Solution Building Blocks (SBBs), where ABBs are realized through the application of the ADM and guide the creation of SBBs; see the TOGAF Standard.

- Shall identify Digital Products (SBB) that realize the Target Architecture
- Shall ensure that the architectural artifacts are compliant with all standards and policies, including security standards and policies
- Shall ensure that the realized architecture meets the KPIs and SLAs
- Shall ensure that each Architecture Blueprint addresses Portfolio Backlog Items and stakeholder concerns
- Shall ensure that the output of the Enterprise Architecture functional component is used by the Product Portfolio functional component to define the Digital Products that need to be created
- Shall ensure the compliance of created Digital Products to the Architecture Roadmap and Architecture Blueprint artifacts
- Can receive Strategic Objective information which includes the objectives and some content based upon which the Enterprise Architecture is developed if a Strategy functional component exists
- Can receive Portfolio Backlog Item information from the Portfolio Backlog functional component used to create the Architecture Roadmap and Architecture Blueprint if a Portfolio Backlog functional component exists

6.1.3.1. Architecture Roadmap Item Data Object

Purpose

The Architecture Roadmap Item data object is produced in the context of architecture work and lists individual work packages in a timeline that will realize the Target Architecture.

- The Architecture Roadmap Item data object is a planning item that refers to a cohesive set of architectural artifacts
- The Architecture Roadmap Item defines a stable step towards the Target Architecture that will achieve the business goals and respond to the strategic drivers

Key Attributes

The Architecture Roadmap Item data object shall have the following key data attributes:

- **Id:** unique identifier of the Architecture Roadmap Item
- **Name:** name of the Architecture Roadmap Item
- **Status:** lifecycle status of the Architecture Roadmap Item
- **TargetDate:** planned date of the realized architecture of this specific Architecture Roadmap Item
- **Roadmaps:** name of the roadmap(s) that will contain one or several Architecture Roadmap Items
- **Description:** description of the Architecture Roadmap Item

Key Data Object Relationships

The Architecture Roadmap Item data object shall maintain the following relationships:

- Architecture Roadmap Item to Product Backlog Item (n:m): establish what major program increments will implement the Architecture Roadmap Item
- Architecture Roadmap Item to Value Stream (n:m): establish which value stream will be created or improved by implementing the Architecture Roadmap Item
- Architecture Roadmap Item to Architecture Blue Print (n:m): establish which blueprint will be supported by the work in the Architecture Roadmap Item
- Architecture Roadmap Item to Strategic Objectives (n:m): establish the strategic rational for a given Architecture Roadmap Item
- Architecture Roadmap Item to Policy (n:m): multiple Policies might be applicable to the Architecture Roadmap

6.1.3.2. Architecture Blueprint Data Object

Purpose

The Architecture Blueprint data object represents the development of the Enterprise Architecture which details the building blocks (including Digital Products) and how these relate to each other. It

contains a list of architecture views or architectural artifacts that depict a stable and approved state of the architecture.

Key Attributes

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

- **Id:** unique identifier of the Architecture Blueprint
- **Name:** meaningful name of the Architecture Blueprint
- **Description:** short description of the purpose of the Architecture Blueprint
- **Version:** version of the Architecture Blueprint
- **Artifact List:** list of architectural artifacts that describe Business, Data, Application, and Technology aspects of the Architecture Blueprint

Key Data Object Relationships

The Architecture Roadmap Item data object shall maintain the following relationships:

- Architecture Roadmap Item to Architecture Blueprint (n:m): one Architecture Roadmap Item may be described by one or several Architecture Blueprints
- Architecture Blueprint to Digital Product (n:m): traceability is maintained between one or more Architecture Blueprints and Digital Products

6.1.3.3. Value Stream Data Object

Purpose

The Value Stream data object represents an end-to-end view of how value is achieved for a given stakeholder. A Value Stream is depicted as an end-to-end collection of value-adding activities that creates a result for a customer, stakeholder, or end-user.

Key Attributes

The Value Stream data object shall have the following key data attributes:

- **Id:** unique identifier of the Value Stream
- **Name:** name of the Value Stream
- **Description:** description of the Value Stream
- **Stakeholder:** name of the Stakeholder that triggers and gets value from the Value Stream
- **Value Stream Id (N):** unique identifiers of the related Value Streams

Key Data Object Relationships

The Value Stream data object shall maintain the following relationships:

- Value Stream to Architecture Roadmap Item (n:m): establish which Value Stream will be created or improved by implementing the Architecture Roadmap Item
- Value Stream to Digital Product (n:m): define which Digital Products contribute to delivering the value specified in the Value Stream

6.2. Portfolio Function

The Portfolio function is centered around managing the portfolio of Digital Products.

6.2.1. Portfolio Backlog Functional Component

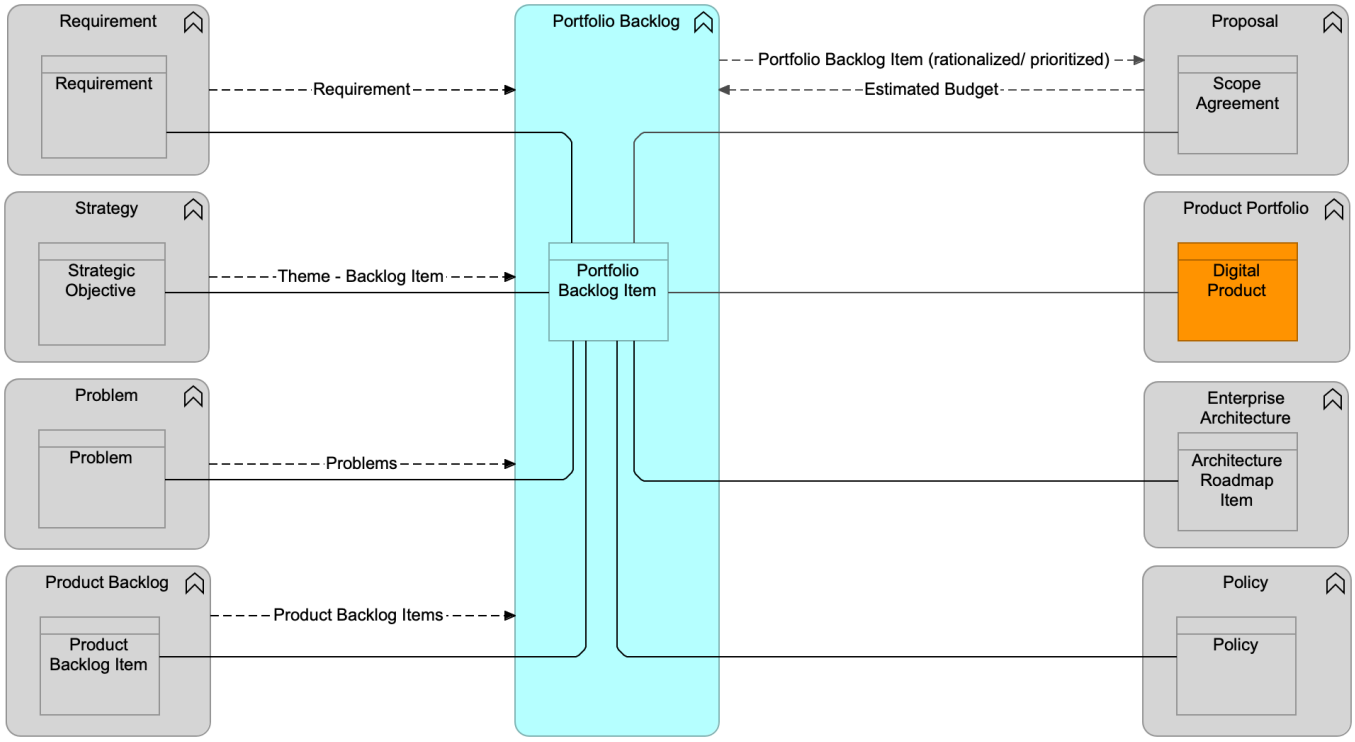

Figure 6-5. Portfolio Backlog Functional Component Model

Purpose

The Portfolio Backlog functional component provides for the lifecycle management and visibility of the Portfolio Backlog Items as grouped into collections of work and prioritized.

The Portfolio Backlog functional component supports the value streams:

- Evaluate
- Explore
- Integrate

Functional Criteria

The Portfolio Backlog functional component:

- Shall provide for the transparency of proposed, accepted, rejected, and prioritized Portfolio Backlog Items
- Shall execute the processes necessary for reviewing, accepting, creating, prioritizing, updating, and closing Portfolio Backlog Items

6.2.1.1. Portfolio Backlog Item Data Object

Purpose

The Portfolio Backlog Item data object is a discrete and specific statement of work with acceptance criteria, funding requirements, and detailed work definitions. It may be a collection of many smaller work items, such as work packages or Product Backlog Items.

Key Attributes

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

- **Id:** unique identifier; e.g., number of the Portfolio Backlog Item
- **Name:** descriptive title of what the Portfolio Backlog Item represents
- **Description:** short description of the work contained within the Portfolio Backlog Item
- **Prioritization:** current priority on the Portfolio Backlog
- **Acceptance Criteria:** what must be accomplished by the work in order to consider the item complete and accepted
- **Related Products:** products that are impacted by the execution and completion of this work
- **Demand Profile:** detailed list of how this Portfolio Backlog Item will consume resources, such as labor, contracts, infrastructure
- **Cost Estimation:** estimated cost derived from the demand profile

Key Data Object Relationships

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

- Portfolio Backlog Item to Strategic Objective (n:m): a strategic initiative may drive the creation and prioritization of many Portfolio Backlog Items; the Portfolio Backlog Item may satisfy the objectives of many strategic initiatives
- Portfolio Backlog Item to Architecture Roadmap Item (n:1): a Portfolio Backlog Item must align to the authoritative Architecture Roadmap Item
- Portfolio Backlog Item to Policy (n:m): Portfolio Backlog Items may be sourced from Policies or may reference Policies to demonstrate compliance

- Portfolio Backlog Item to Scope Agreement (1:1): a Portfolio Backlog Item can have one corresponding Scope Agreement
- Portfolio Backlog Item to Digital Product (n:m): a Portfolio Backlog Item may be executed or impact many Digital Products; a Digital Product may be executing work against many Portfolio Backlog Items
- Portfolio Backlog Item to Product Backlog Item (1:m): Portfolio Backlog Items may contain many Product Backlog Items; a Product Backlog Item is related to one Portfolio Backlog Item
- Requirement to Portfolio Backlog Item (n:m): Requirements can be associated with one or more Portfolio Backlog Items

6.2.2. Proposal Functional Component

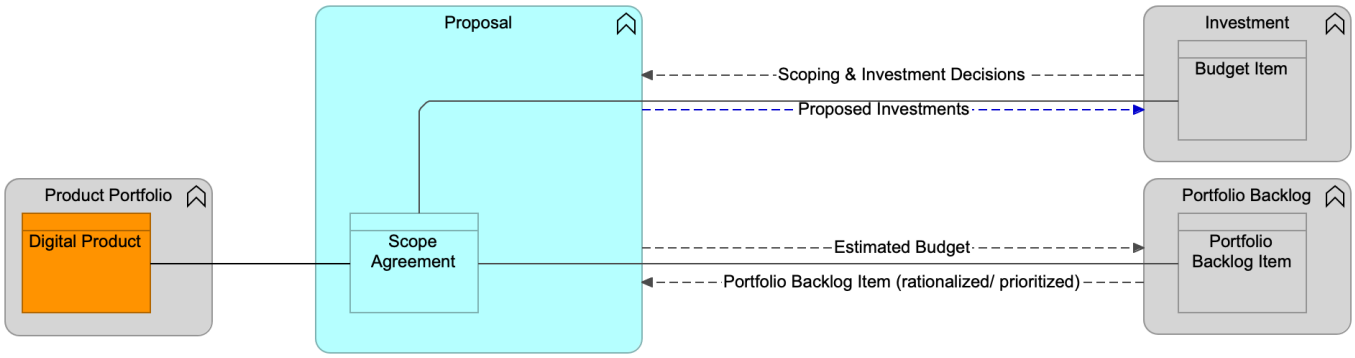

Figure 6-6. Proposal Functional Component Model

Purpose

The Proposal functional component manage proposals to fund the implementation of a package of one of more Portfolio Backlog Items. Each package is described as a Scope Agreement. The Proposal functional component manages these Scope Agreements in various states: proposed, approved, active, deferred, or rejected. It can create views of Scope Agreements for specific functions, company-wide or for a line of business, such as for building an investment plan of record or for comparison of proposals.

The Proposal functional component interoperates and maintains alignment with the Portfolio Backlog, Product Portfolio, and Investment functional components. It supports and may automate evaluation of the feasibility and resources needs for each Scope Agreement. This evaluation includes refinement and rationalization of cost and resource estimates held in any associated Portfolio Backlog Items.

The Proposal functional component maintains approved Scope Agreement data objects over the life of all associated Digital Products. It retains unapproved Scope Agreement records according to the organization's data management and retention rules.

The Proposal functional component supports the value streams:

- Evaluate
- Explore
- Integrate

Functional Criteria

The Proposal functional component:

- Shall analyze and refine Proposals consisting of rationalized Portfolio Backlog Items and associated funding, create a Scope Agreement representing approved Proposals, and manage the portfolio of all Proposals and Scope Agreements in the data object repository

A Scope Agreement:

- May follow an expedited analysis and approval for high-priority urgent items or Agile development proposals
- May follow a structured analysis and approval; for instance, via annual planning activities

A Scope Agreement following an expedited analysis and approval:

- Shall create a proposal from a rationalized Portfolio Backlog Item where the item requires high urgency due to business impact on an existing service/Digital Product
- Shall allow a quick evaluation of the proposal and a decision on its approval; and if rejected, will then notify the Portfolio Backlog functional component
- Shall create an updated Scope Agreement and update the corresponding in-flight Product Backlog Item data object for all activities associated with a newly approved demand

Scope Agreements following a structured analysis and approval:

- Shall create proposals 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 a priority and a cut-off must be decided.

- May periodically produce proposals throughout the year or once per planning period
- Shall create a high-level labor consumption model for a proposal (for example, one project manager, five developers, and two quality assurers for the proposal) based on labor estimates

  If a Resource Management Offer exists in the Offer catalog, then the Proposal functional component shall receive the resource price from the mentioned Offer and validate the labor consumption model against available internal and external labor pools.

- Shall create a high-level asset (non-labor) consumption model for a proposal based on asset estimates

  If an Asset Management Offer exists, then the Proposal functional component shall receive the asset price from the mentioned Offer.

- Should validate the asset consumption model against available internal and external assets; for example, traditional/private cloud/managed cloud/public cloud
- Should model the ongoing labor and non-labor budget for annual and future operations
- May define tangible and intangible benefits for each proposal

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

- Shall ensure the proposal meets the technology policies
- Should rank proposals based on benefits and risks, labor and non-labor consumption models, ROI, or other defined evaluation criteria
- May build proposal portfolio scenarios using proposals, conduct “what if” analysis on the proposal scenarios, and approve the optimal proposal scenario and its proposals
- Shall send proposed investments to the Investment functional component for scoping and investment decisions
- Shall update Scope Agreement(s) to compare approved baseline and actual resulting benefits derived from completing the Digital Product delivery, thus comparing planned with actual delivery

In addition, the Proposal functional component:

- Shall review the Scope Agreement change request from the Integrate value stream

  The Digital Product team working to deliver the approved Scope Agreement may ask for change requests related to budget, resource, timeline, or scope, and may evaluate the change request and take action to update the existing Scope Agreement.

- Shall consider the Product Portfolio as the authoritative source for the list of deliverables or services that will be rendered during a product lifecycle
- May create Product Portfolio views for specific organizations such as a line of business portfolio or functions like financial views; the Product Portfolio is used for rationalizing and tracking resources across Product teams to best deliver on all products
- Should actuate the Product Portfolio entries through a Project Management system

  The Product Portfolio functional component shall report back to the Investment functional component in order to accurately track progress and outcomes for a given Scope Agreement.

- Shall identify the security controls necessary for protecting the various classifications of data

6.2.2.1. Scope Agreement Data Object

Purpose

The Scope Agreement data object represents a proposal to implement one or more rationalized Portfolio Backlog Items and Budget Items. Scope Agreements reflect budget, cost/benefit projections, scope, status, and other significant aspects of the proposed work. They can be defined in the context of portfolio- or product-level concerns. The Scope Agreement represents the agreement between the requesting party (usually the Digital Product portfolio manager) and the delivery party (such as a Digital Product delivery team) on the work to be performed to implement the specified Portfolio Backlog Items and the funding available for the implementation.

Key Attributes

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

- **Id:** unique identifier of the Scope Agreement
- **Description:** details of the Scope Agreement
- **Business Entity:** business or geographic unit
- **Anticipated Business Value (optional):** projected business value, over a given time period, anticipated once the corresponding products and capability have been delivered
- **Proposed Budget:** requested budget of the Scope Agreement; this should be broken down into a build and a run budget
- **Approved Budget:** approved budget of the Scope Agreement; this should be broken down into a build and a run budget
- **Status:** Scope Agreement (e.g., created, proposed, approved, rejected, deferred)
- **Context:** portfolio level, product level

Key Data 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 Budget Item (n:1): this relationship helps track the budget allocated to each Scope Agreement
- Scope Agreement to Product Backlog Item (1:n): this relationship helps track Digital Product(s) to each Scope Agreement

6.2.3. Product Portfolio Functional Component

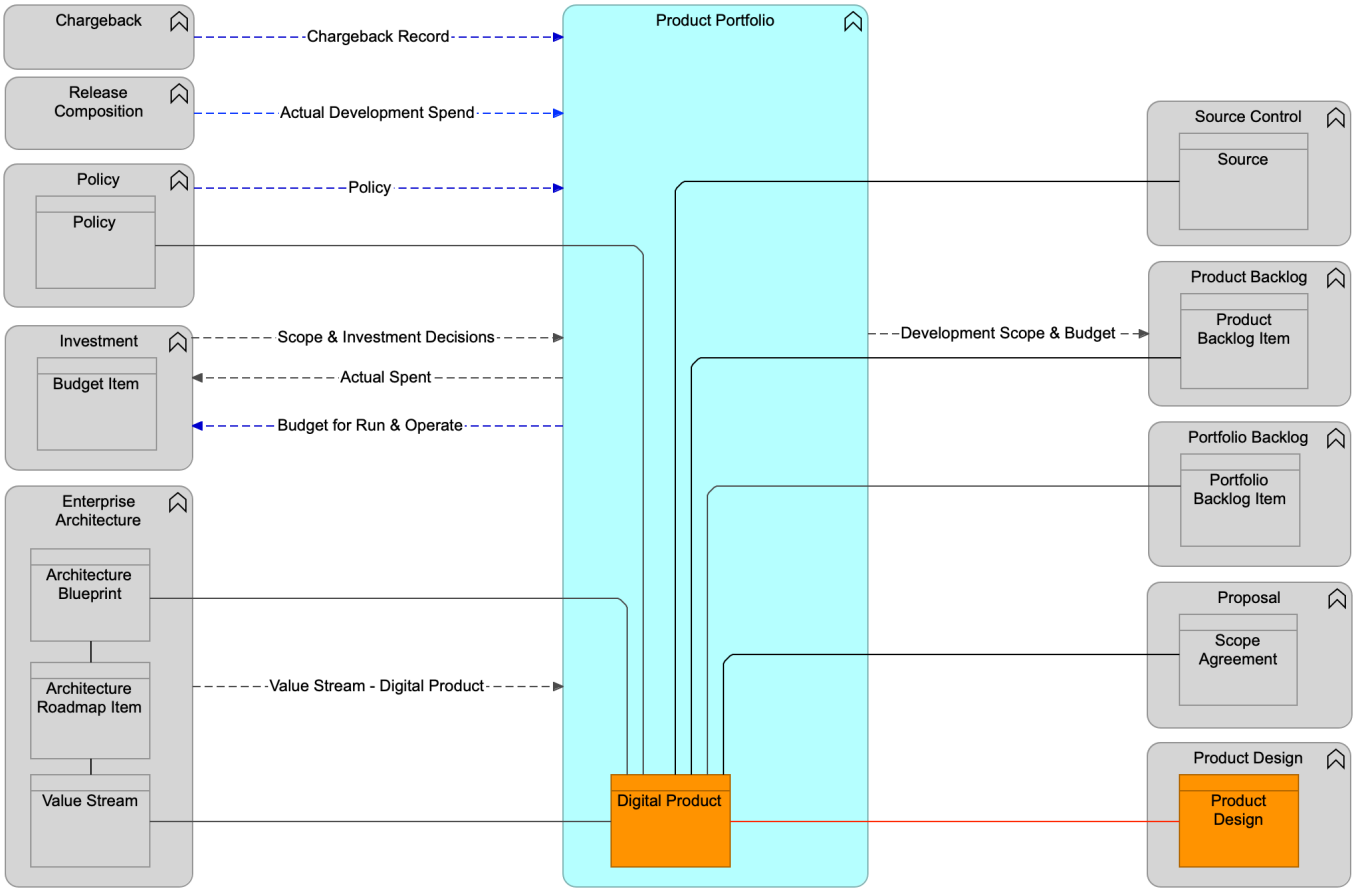

Figure 6-7. Product Portfolio Functional Component Model

Purpose

The Product Portfolio functional component structures and organizes all of the different types of Digital Products managed by the organization. Digital Products are grouped into portfolios that allow for the grouping of like Digital Products based on ownership, value stream alignment, product lines, strategic investment groups, and prioritization.

Each Digital Product is managed based on common criteria and attributes that provide the data to properly manage them throughout their lifecycle. Managing dependencies and dependents of the Digital Products is a critical aspect of the Digital Product, as well as making portfolio decisions.

While the Product Portfolios may be arranged in any way an organization requires, there are three basic top portfolio tiers to facilitate strategic planning:

- **Market-facing** – Digital Product offerings used by customers, organized by product lines, versions, or models accessed by a human or machine interface
- **Internal-facing** – Digital Product offerings used by employees, manufacturing, and other automaton tasks

- **Foundational** – common and foundational Digital Products often used to build or facilitate other Digital Products such as infrastructure, networking, platform, and other commodity building blocks

These three portfolios allow for relevant strategic discussion and funding decisions to be made more readily, without needing to identify every specific Digital Product affected.

The Product Portfolio functional component supports the value streams:

- Evaluate
- Explore

Functional Criteria

The Product Portfolio functional component:

- Must include all Digital Products managed by the organization
- May support a taxonomy of portfolios with sub-portfolios, and ultimately contained Digital Products
- Must provide, via specific Product Designs and Product Releases, Service Offers describing the contract, rules, warranty, and other consumer-based attributes used to establish a contract
- Must track consumer contracts, which may be simple promises, acceptance of usage terms, Subscriptions, or binding legal agreements
- Shall support strategic funding and de-funding activities at a portfolio level, and infer decisions on all contained Digital Products
- Shall review input from organizational strategy, consumers, and market sources to propose portfolio changes that decide whether to keep, retire, increase, or decrease investment
- Shall track all backlog items used for prioritizing investment and potential changes
- Shall assess effectiveness, efficiency, and satisfaction of the Service Offers being consumed
- Should aggregate all metrics from dedicated resources, dependencies, and consumption to include, for example, total costs, performance, consumption, capacity, resources, risks, benefits, quality, fitness-for-purpose
- Shall manage an inventory of underlying shared resources, systems, Service Offers, and instances
- Shall track all consumers such as users, instances, and machine/API dependencies
- Shall compare Digital Products with similar ones to identify rationalization opportunities
- Shall create, review, and update Digital Products
- Shall create and maintain service blueprints and end points
- May share roadmaps with market, consumers, peers, partners, and dependent Product Managers
- May have roadmaps that include major and minor versions/models

- Should ensure compliance with all applicable Policies if a related Policy exists
- Shall aggregate charges from the Chargeback functional component if a Chargeback functional component exists
- May send the operation charge acceptance to the Chargeback functional component if a Chargeback functional component exists

6.2.3.1. Digital Product Data Object

Purpose

A Digital Product data object is an entity representing the Digital Product throughout its entire lifecycle. It is used to organize activities across all value streams.

A more elaborate description of Digital Product is provided in the Digital Product chapter, to include types of Digital Products, examples, and rationale for definitions as supported by this data object.

Key Attributes

- **Id:** unique product identifier
- **Name:** descriptive, common name of the Digital Product
- **Description:** short description of the Digital Product
- **Product Manager:** name of the Product Manager
- **Investment Status:** status of the Digital Product; e.g., invest, divest, sustain, retired
- **Portfolio:** portfolio to which the Digital Product belongs
- **Business Criticality:** indication of the importance of the Digital Product for the business
- **Security Risk:** indication of the business risk if the Digital Product is compromised
- **Budget:** approved budget for ongoing resources, development, maintenance, and services
- **Total Cost of Ownership (TCO):** the total cost of ownership of the Digital Product (includes all development, run, enhancement, and overhead charges for systems and shared costs)
- **Profit/Loss:** calculated chargeback/showback against TCO for the Digital Product

Key Data Object Relationships

The Digital Product data object shall maintain the following relationships:

- Digital Product to Policy (n:m): multiple Policies might be applicable for a single Digital Product or a single Policy may be applicable for multiple Digital Products
- Digital Product to Value Stream (n:m): traceability is maintained between Digital Products and the Value Stream it supports
- Digital Product to Portfolio Backlog Item (1:n): one Digital Product may be related to one or more Portfolio Backlog Items

- Digital Product to Architecture Blueprint (n:m): traceability may be maintained between one or more Digital Products and the Architecture Blueprint drawings, diagrams, and other planning documents
- Digital Product to Scope Agreement (n:m): Digital Products may have one or more Scope Agreements associated with them
- Digital Product to Product Design (1:n): Digital Products will have one or more Product Designs associated with each version/model of the Digital Product
- Digital Product to Digital Product (n:m): a Product can depend on services being delivered by other Products

Chapter 7. Requirement to Deploy Functions

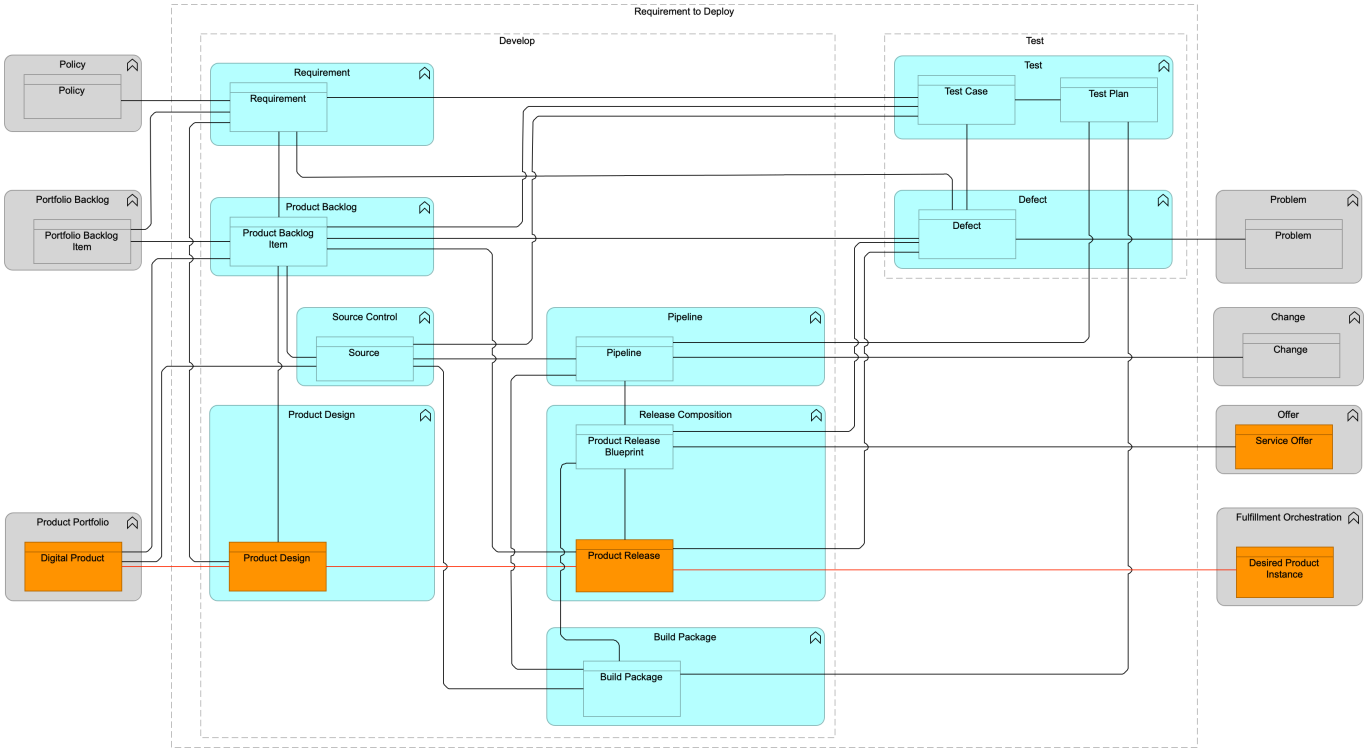

Figure 7-1. Requirement to Deploy Functions Model

Description

The Requirement to Deploy functions, informally referred to as “Build” enable the creation of high-value Product Releases for new and existing Digital Products that:

- Are aligned with, and tested against, all new or modified requirements and features, including non-functional requirements such as security, risk, and regulatory requirements
- Are deployable into standard catalogs, instantiated automatically for any kind of consumer
- Are instrumented to support effective operational and financial measurement and management throughout the product lifecycle
- Are packaged for immediate or future deployment through the Deploy value stream
- Include all changes made to the Digital Product throughout its lifecycle

The Requirement to Deploy functions support the creation of automatable Release Packages for all types of market-facing, internal-facing and foundational products including custom-built software, cloud-native applications, configuration of vendor solutions such as package-based software (COTS), SaaS, infrastructure products, and platforms used to host or support other Digital Products.

The Requirement to Deploy functions support all product engineering activities including technical design, requirements management, backlog and implementation management, sourcing, solution development and configuration, software maintenance, testing, defect management, and release packaging.

The Requirement to Deploy functions are methodology-independent, supporting all approaches to creating Product Releases. The Product Backlog Items (e.g., feature, business requirement, non-functional) are abstractions of implementation tasks. They may originate as a Requirement, Portfolio Backlog Item, Policy, or Problem; or may arise directly within the implementation team during solution development work. Regardless of methodology, the Requirement to Deploy functional components and data objects remain constant.

The Requirement to Deploy functions contain the following functional components:

- Develop functionality:
  - Product Backlog component
  - Requirement component
  - Product Design component
  - Source Control component
  - Pipeline component
  - Build Package component
  - Release Composition component
- Test functionality:
  - Test component
  - Defect component

Related Value Streams

The following value streams use one or more functional components from the Requirement to Deploy functions:

- Integrate
- Release
- Deploy

Business Benefits

The Requirement to Deploy functions describe a prescriptive framework of required functional components, integrations and data objects so organizations can deliver better value, sooner, and safer with lower costs while improving the productivity of the product teams.

The main benefits of using the Requirement to Deploy functions are:

- Reduced lead time of delivering Product Backlog Items (e.g., new features or resolving Defects or Problems)
- Increased deployment frequency of new Product Releases

  By using the same automatable architecture across teams (and by implication, standardized tools for the most automatable parts of the development and release chains), release frequency can be made more predictable and manageable. By ensuring that the Product Release includes all the content needed for automated instantiation, the Requirement to Deploy functional and data models provide an architecture able to decrease the time from committed code to live systems to zero.

- More complete and more traceable testing capability, which should result in higher change success rates and reduced security risks
- End-to-end transparency and traceability from requirement and/or backlog item to Product Release
- Reduced risk due to Security and Compliance by Design

  The Requirement to Deploy functions maintain the association between Policy and Requirement data objects throughout the product lifecycle. This persistent traceability enables designers to ensure that all non-functional requirements are accounted for, so that products are designed in accordance with standards and policies from sources such as Security Management, Governance, Risk, & Compliance, Legal & Regulatory, Enterprise Architecture, and Financial Management.

- Improved interoperability, communication, and collaboration among involved stakeholders and teams (including external vendors)

Applications and services may be sourced or developed in cooperation with many different parties, all of which work with their own processes and tooling. The Requirement to Deploy functional criteria define a standard for interoperability that enables digital organizations to enforce a consistent, standardized description of planned activities and interoperability of functions and data.

- Standardized product development and delivery to the point where the reuse of service components is the norm

The Requirement to Deploy functions enable the creation of consistently defined libraries of reusable artifacts by prescribing how to manage Requirements, Source, documentation, test scripts, Service Monitors, and other artifacts of the service development lifecycle as standardized data objects. The explicitly defined interfaces of the Requirement to Deploy functions means that access to such libraries can be extended across multiple organizations and locations. When access to these libraries is integrated with implementation tools such as Integrated Development Environments (IDEs), and supported by management direction, a dramatic increase in reuse becomes a realistic goal.

- Accelerate the sourcing and delivery of products and services through best practices such as:
  - Reuse – manage, maintain, and leverage reusable components and services
  - Automation – identify the core functional components and data required to streamline the Integrate value stream
  - Collaboration – use data to institutionalize collaboration of teams involved in the development lifecycle

7.1. Develop Function

The Develop functionality is centered around the development, enhancement, and maintenance of Digital Products.

7.1.1. Product Backlog Functional Component

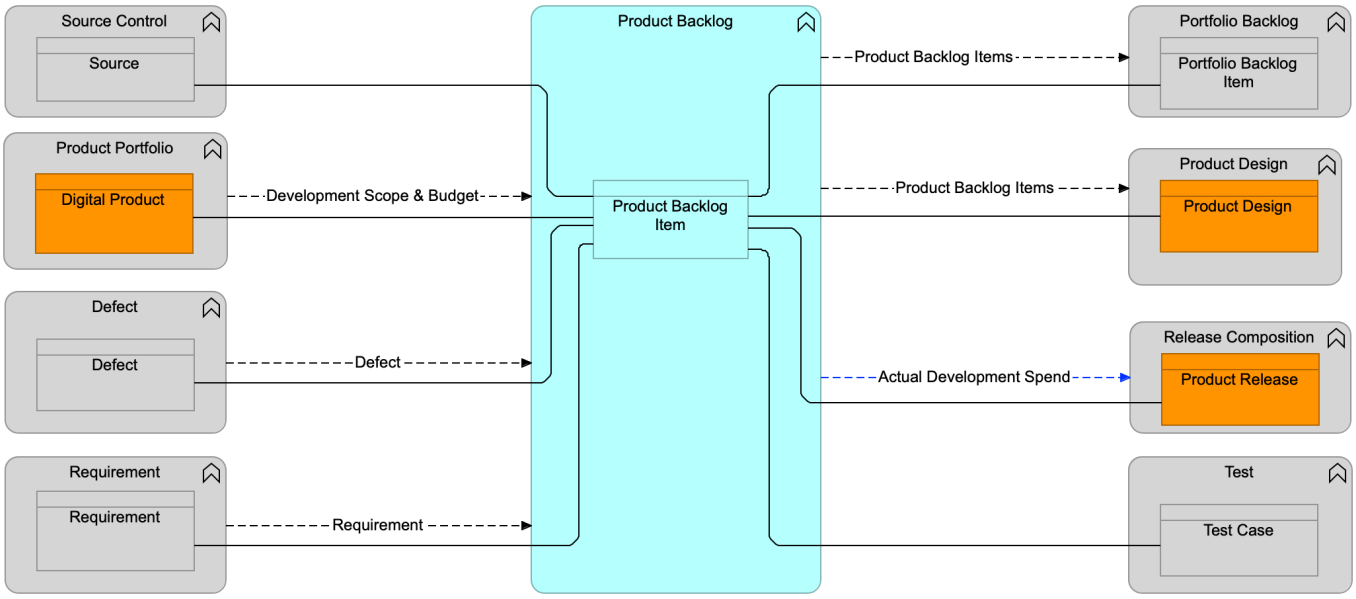

Figure 7-2. Product Backlog Functional Component Model

Purpose

The Product Backlog functional component manages the creation of, and provides the ongoing execution, oversight, and management of all work items (Product Backlog items) involved in the creation, modification, or improvement of a Digital Product throughout its entire lifecycle, delivered through Product Releases.

Product Backlog functional component manages the lifecycle of Product Backlog Items including organization of Product Backlog Items into prioritized and rationalized team backlogs. It maintains the linkage / traceability from Portfolio Backlog Items to Requirements and Product Releases. It aggregates, tracks, and reports status, resources consumed against the plan (e.g., project, iteration or sprint plan), and communicates these to stakeholders.

The Product Backlog functional component also receives Defects from the Defect functional component to enable the development of fixes and/or temporary workarounds.

The Product Backlog functional component supports the value streams:

- Evaluate
- Explore
- Integrate

Functional Criteria

The Product Backlog functional component:

- Shall be the system of record (authoritative source) for all Product Backlog Items for a specific Digital Product
- Shall define standard types of Product Backlog Items such as product epics, product features, and user stories
- Shall include security, risk, and compliance Product Backlog Items to ensure security and risk management stories are fully embedded into each Product Release
- Shall have the ability to organize Product Backlog Items into groups associated with a specific Digital Product or implementation team.
- Shall prioritize each of the Product Backlog Items
- Shall define the value for the customer of each Product Backlog Item
- Shall have the ability to capture and maintain the size and effort of each Product Backlog Item
- Can have associated requirements, including non-functional requirements
- Shall define acceptance criteria for each Product Backlog Item
- Shall define the “Definition of Done” for each Product Backlog Item

The Definition of Done is an agreed set of criteria that must be completed before a Product Backlog Item can be considered as “done.” This may include administrative tasks not directly related to acceptance criteria.

- Shall have the ability to calculate the business impact and cost of delay for each Product Backlog Item
- Shall have the ability to record risks associated with Product Backlog Items
- Shall manage the lifecycle and status of the Product Backlog Items
- Shall coordinate the planning (e.g., sprint plans) of Product Backlog Items and provide ongoing execution oversight of Product Backlog Items
- Shall aggregate, track, and report status, resources consumed against the plans, or burn down charts, and communicate these to stakeholders such as Financial Management

- May integrate with the Collaboration & Communication functional component to engage with different stakeholders during design, development, deploy, and test management activities
- Shall maintain traceability between Product Backlog Items and associated products and service(s) being developed
- Can relate a Product Backlog Item to a Portfolio Backlog Item
- Can decompose Product Backlog Items into sub-Product Backlog items (e.g., features with related user stories)
- Can use different methods or practices to manage and visualize the backlog items such as Scrum, Kanban, or traditional Project Management methodologies
- Shall maintain the linkage/traceability from Portfolio Backlog Items to actual realization in Product Releases

7.1.1.1. Product Backlog Item Data Object

Purpose

The Product Backlog Item represents a work item associated with a Digital Product that when fulfilled will be associated with a Product Release.

Types of work item vary depending on development methodology and may include epics, features, stories, story tasks, fixes, enhancements, and so on. Product Backlog Items describe the full range of work and artifacts required to complete the Product Release. This includes not only code and configuration information, but also the creation of other deployable artifacts under version control such as training material, user guides, and pricing tables.

A Product Backlog Item can be decomposed into a set of more granular Product Backlog Items; for example, a feature can be decomposed into user stories and related tasks. A group of Product Backlog Items can be assigned to a specific implementation phase, such as an iteration or sprint.

Key Attributes

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

- **Id:** unique identifier of the Product Backlog Item
- **Title:** short description of the Product Backlog Item
- **Type:** type of Product Backlog Item such as epic, feature, and story
- **Description:** detailed description of the Product Backlog Item
- **Status:** status of the Product Backlog Item
- **Parent Id:** unique identifier of the parent Product Backlog Item (decomposition/refinement)
- **Budget:** budget for the Product Backlog Item
- **Actual Spend:** actual spend on the Product Backlog Item

- **Start Date:** Product Backlog Item start date
- **End Date:** Product Backlog Item completion date
- **Size:** size of the Product Backlog Item; for example, in story points
- **Definition of Done:** list of criteria which must be met before a Product Backlog Item is considered "done"
- **Priority:** priority of the Product Backlog Item
- **Value:** scoring of the Product Backlog Item in terms of business value

Key Data Object Relationships

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

- Product Backlog Item to Digital Product (n:1): a Product Backlog Item belongs to one Digital Product
- Portfolio Backlog Item to Product Backlog Item (1:n): a Portfolio Backlog Item can be decomposed into one or more Product Backlog Items (refinement of Portfolio Backlog Items)
- Product Backlog Item to Product Backlog Item (1:n): a Product Backlog Item can be refined into one or more sub-Product Backlog Items (parent and child Product Backlog Items)
- Product Backlog Item to Product Backlog Item (n:m): a Product Backlog Item can be dependent upon or related to other Product Backlog Items
- Product Backlog Item to Product Release (n:1): a Product Backlog Item will be associated with one Product Release; larger Portfolio Backlog Items which cannot be delivered in one release will need to be decomposed into sub-Product Backlog Items (refinement)
- Product Backlog Item to Product Design (n:1): a Product Backlog Item can modify and/or realize a (part of the) Product Design
- Product Backlog Item to Source (n:m): modifications in the source code need to be linked to a Product Backlog Item (with a merge request)
- Product Backlog Item to Defect (n:m): a Product Backlog Item can be related to one or many Defect records in order to investigate and resolve the Defects
- Product Backlog Item to Requirement (n:m): a Product Backlog Item must be associated with one or more Requirements (such as non-functional requirements); a new or changed Product Backlog Item may require creation or modification of one or more Requirements
- Product Backlog Item to Test Case (n:m): a Product Backlog Item can have one or more Test Cases defined in the Test functional component

7.1.2. Requirement Functional Component

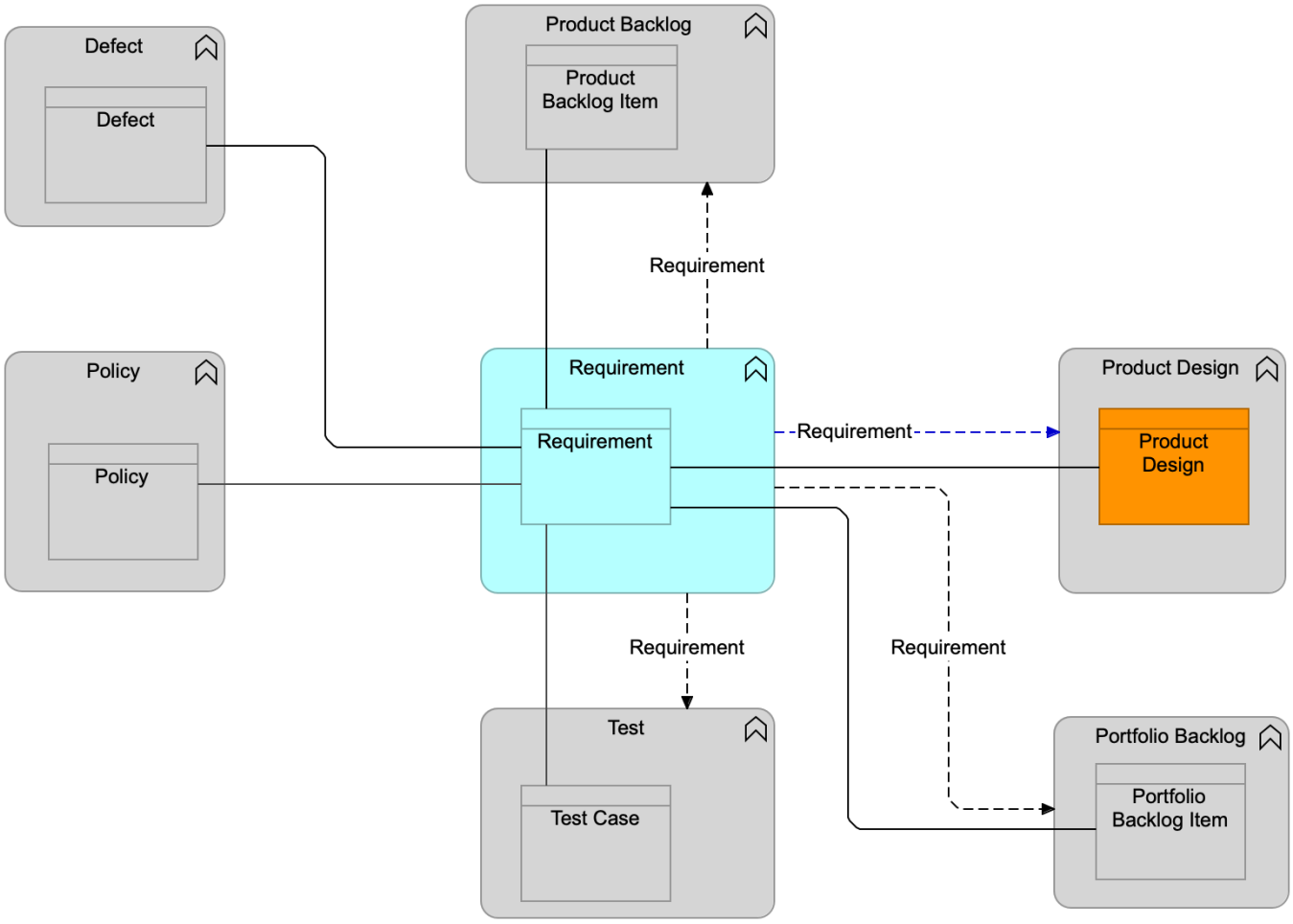

Figure 7-3. Requirement Functional Component Model

Purpose

The Requirement functional component manages the Requirements of a product throughout its lifecycle. Requirements are associated with the Product Backlog Items managed in the Product Backlog. The Product Backlog manages all epics, features, and stories, which are short-lived, and closed once completed, while the Requirements are long-lived and provide the definitive specifications of the product. This definitive list of Requirements is also used for regression testing.

NOTE

The Requirement functional component also tracks non-functional requirements, such as those related to security, risk, privacy, and compliance. These non-functional requirements are derived from the Policy functional component and are selected based upon the outcome of risk assessments such as a Security Impact Assessment (SIA) and Data Privacy Impact Assessment (DPIA). The relevant security and compliance Requirements should be added to the Requirement functional component. These Requirements are then linked to the Product Design, Product Backlog Items, and associated Test Cases. This ensures the product is designed according to the agreed Policies and enables full traceability on the compliance of the product against defined Policies and controls.

The Requirement functional component manages all the Requirements (both functional and non-functional requirements) for the different stakeholders such as those related to:

- Business requirements
- Security requirements (derived from Policies)
- Requirements derived from SLOs such as those related to availability, performance, and capacity (to define the expected performance of the product)
- Legal, regulatory, and compliance requirements (such as those related to data privacy)
- Data management requirements
- Business continuity/disaster recovery requirements (related to business and service continuity)
- Architectural requirements
- User experience requirements (such as those related to Experience-Level Agreements (XLAs)
- Operations requirements (such as those related to monitoring and logging)

All Requirements need to be collected, refined, scoped, and have their progress tracked across the full development lifecycle.

The Requirement functional component supports the value streams:

- Explore
- Integrate

Functional Criteria

The Requirement functional component:

- Shall be the system of record (authoritative source) for all Requirements
- Shall manage the lifecycle of the Requirements
- Shall manage the state of a Requirement
- Shall track all changes to Requirements
- Shall manage Requirements through the lifecycle of a product
- Shall capture service level Requirements such as those related to availability, reliability, and performance
- Shall capture Requirements related to user experience (such as those related to user experience design, usability)
- Shall capture Requirements related to security, risk, privacy, and compliance (for example, those derived from risk assessments and policies), and shall ensure that these Requirements can be traced back to originated Policies and associated legal and regulatory frameworks
- Shall capture Requirements related to data management, such as those related to data integrity

- Shall capture Requirements related to monitoring and logging (ensuring the deployed product can be monitored)
- Shall collect, refine, scope, and track the progress of Requirements linked to the Product Backlog Items
- Shall maintain traceability of each Requirement to the original Source (Product Backlog Item, a Policy, and/or requestor) and to the appropriate Source and/or Test Cases throughout the product lifecycle
- Can manage the data flow to provide Requirement information to the Product Design functional component if a Product Design functional component exists
- Shall allow a Requirement to be traced to one or more Test Cases designed to test the Requirement if a Test functional component exists
- Shall allow one or more Requirements to be associated to one or more Policies that these Requirements originate from if a Policy functional component exists
- May relate Requirements to specific stakeholders and personas
- Shall rank Requirements to define the importance of each of the Requirements (e.g., must have, should have, could have)
- May link Requirements to Portfolio Backlog Items (e.g., defining the high level requirements related to portfolio epics)

7.1.2.1. Requirement Data Object

Purpose

The Requirement data object records which functionality or non-functional conditions to meet for a product.

Key Attributes

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

- **Id:** unique identifier of a given Requirement
- **Name:** short description/title of a given Requirement
- **Description:** represents the more detailed description and statements of a given Requirement
- **Purpose:** purpose of the Requirement
- **Type:** identifies the category of a Requirement, such as functional, security, compliance, data privacy, service level, performance, and availability
- **Status:** status of the Requirement
- **Reference:** reference to the related authoritative sources; for example, Policies, regulatory requirements, security standards
- **Priority:** the priority of a Requirement

- **Owner:** person or team that owns the Requirement statement
- **Version:** version of the Requirement specification

Key Data Object Relationships

The Requirement data object shall maintain the following relationships:

- Product Backlog Item to Requirement (n:m): a Product Backlog Item is mapped to one or more Requirements (e.g., feature or user story) to track how the Requirement is fulfilled/realized
- Requirement to Product Design (n:m): one or more Requirements will be used to define the required behavior from the Product Design
- Requirement to Requirement (n:m): Requirements can be mapped to other Requirements (recursive and/or hierarchical)
- 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
- Policy to Requirement (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
- Requirement to Portfolio Backlog Item (n:m): Requirements can be associated with one or more Portfolio Backlog Items
- Requirement to Defect (n:m): a Defect can be associated with one or more Requirements

7.1.3. Product Design Functional Component

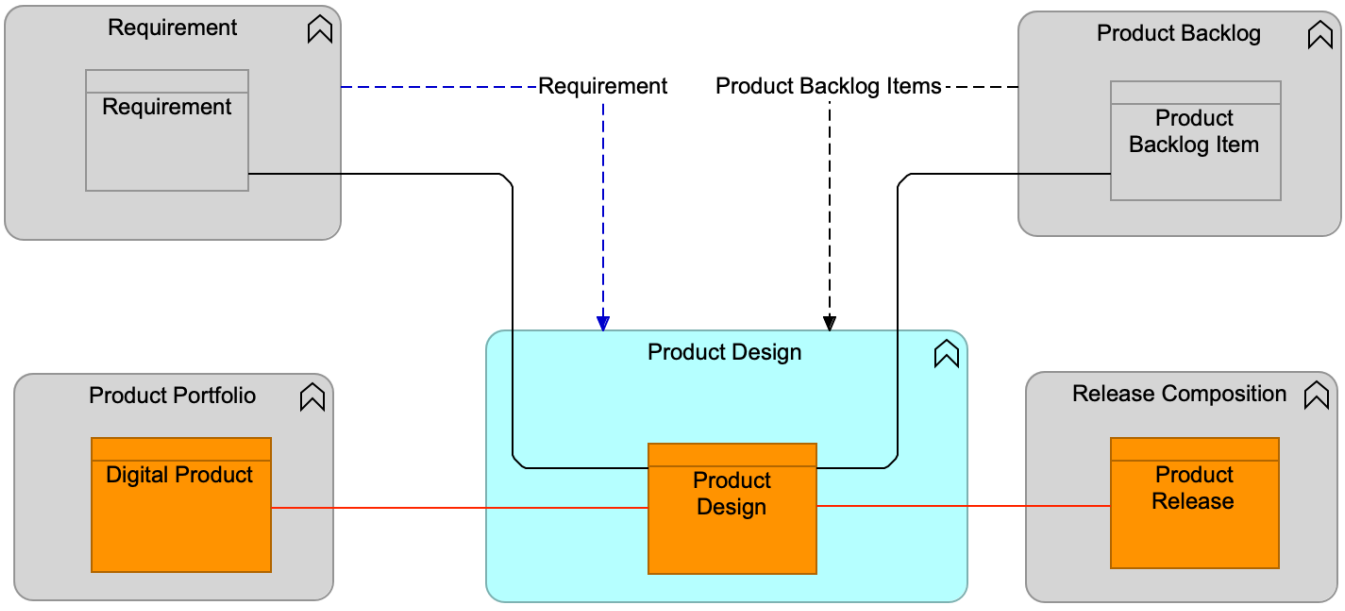

Figure 7-4. Product Design Functional Component Model

Purpose

The Product Design functional component creates and maintains the Product Design associated with a Digital Product. This functional component ensures the architecture and Product Design of the Digital Product meets specified requirements, is in line with the Enterprise Architecture, and adheres to standards and polices.

The Product Design functional component describes the structure and behavior of the services offered by the product, taking into consideration both the desired product and the associated Service Offers. Various design artifacts (e.g., UML® diagrams, API specifications, data models, process designs, user experience design) can be created that comply with the Product Backlog Item specifications, Requirements, and product vision. The service delivery model (e.g., in-source, outsource) can be identified, together with the service providers who will meet the Requirements within the chosen delivery model.

The Product Design should include instrumentation so that objective data can be captured on how the digital services are performing rather than relying only on consumer feedback. The service should also be architected to meet the KPIs and SLAs (and associated SLOs) and to ensure that appropriate Service Monitors are included in the Product Release.

The output of the Product Design functional component will be used by the Source data object to source, create, and secure the service, and the traceability will be done through the Requirement functional component.

The Product Design functional component supports the value streams:

- Explore
- Integrate

Functional Criteria

The Product Design functional component:

- Shall be the system of record (authoritative source) for all Product Designs
- Shall identify the required changes in the Product Design to meet the needs of the Product Backlog Items and associated Requirements
- Shall maintain the design of the Digital Products and its sub-components/parts
- Shall maintain the design of the data model (mapped to the Enterprise Data catalog to understand what data is managed and/or used by the Digital Product)
- Shall maintain the design of the integrations (and related data flows) of the Digital Product with other Digital Products (to understand dependencies and integrations with other products)
- Shall maintain the design related to the security model, covering user profiles, roles, and associated access rights

- Shall maintain various design artifacts (such as use-case diagrams, data models, data flow diagrams, interaction diagrams) that comply with the Product Backlog Item specifications and boundaries
- Shall maintain the design of how the product enables/supports the customer journeys and business processes (if applicable)
- Shall cover the user experience in the design
- Shall identify the product delivery model (e.g., buy or build, in-source, outsource)
- Shall maintain the infrastructure design for hosting the Digital Product (with the associated technology products)
- May identify service suppliers to meet the Requirements within the chosen delivery model
- Shall maintain the design of operations and support features of a product, including the support model, monitoring, and design for self-service (designing service interactions through the Service Offer Catalog)
- Shall document instrumentation so that empirical data can be captured to measure how the digital services are performing, rather than relying only on anecdotal input from the user community
- Shall document how the Digital Product complies with the non-functional requirements, including security, privacy, legal (Secure, Privacy, and Compliance by Design)
- Shall document how the product is architected to meet the SLOs, including availability and performance, as well as XLAs
- Shall receive the Digital Product specifications and product vision from the Product Portfolio functional component
- Shall provide the ability to collaborate on the design artifacts, including reviews, capturing feedback, and comments

7.1.3.1. Product Design Data Object

Purpose

The Product Design data object is the collection of design artifacts related to a Digital Product, together with the changes to the Product Design for a specific Product Release. It represents the grouping of logical components necessary to provide the expected outcome or service interaction. This includes the design for non-functional requirements such as those related to security, privacy, availability, and performance.

Key Attributes

The Product Design data object shall have the following key data attributes:

- **Id:** unique identifier of the Product Design
- **Name:** meaningful name of the Product Design
- **Description:** short description of the purpose of the Product Design

- **Version:** version of the Product Design
- **Artifacts:** reference to the artifacts as part of the Product Design (which are also under version control); this includes, for example, UML diagrams, data models, user experience designs, prototypes

Key Data Object Relationships

The Product Design data object shall maintain the following relationships:

- Digital Product to Product Design (1:n): one or more Product Designs represents the logical components necessary to provide the expected outcome of the Digital Product; a Product Design belongs to one Digital Product
- Product Design to Requirement (n:m): one or more Requirements will be used to define the required behavior from the Product Design
- Product Design to Product Backlog Item (1:n): Product Backlog Items involved in changing and/or realizing the Product Design
- Product Design to Product Release (1:n): a Product Release is based upon one Product Design in order to deliver the required service outcomes

7.1.4. Source Control Functional Component

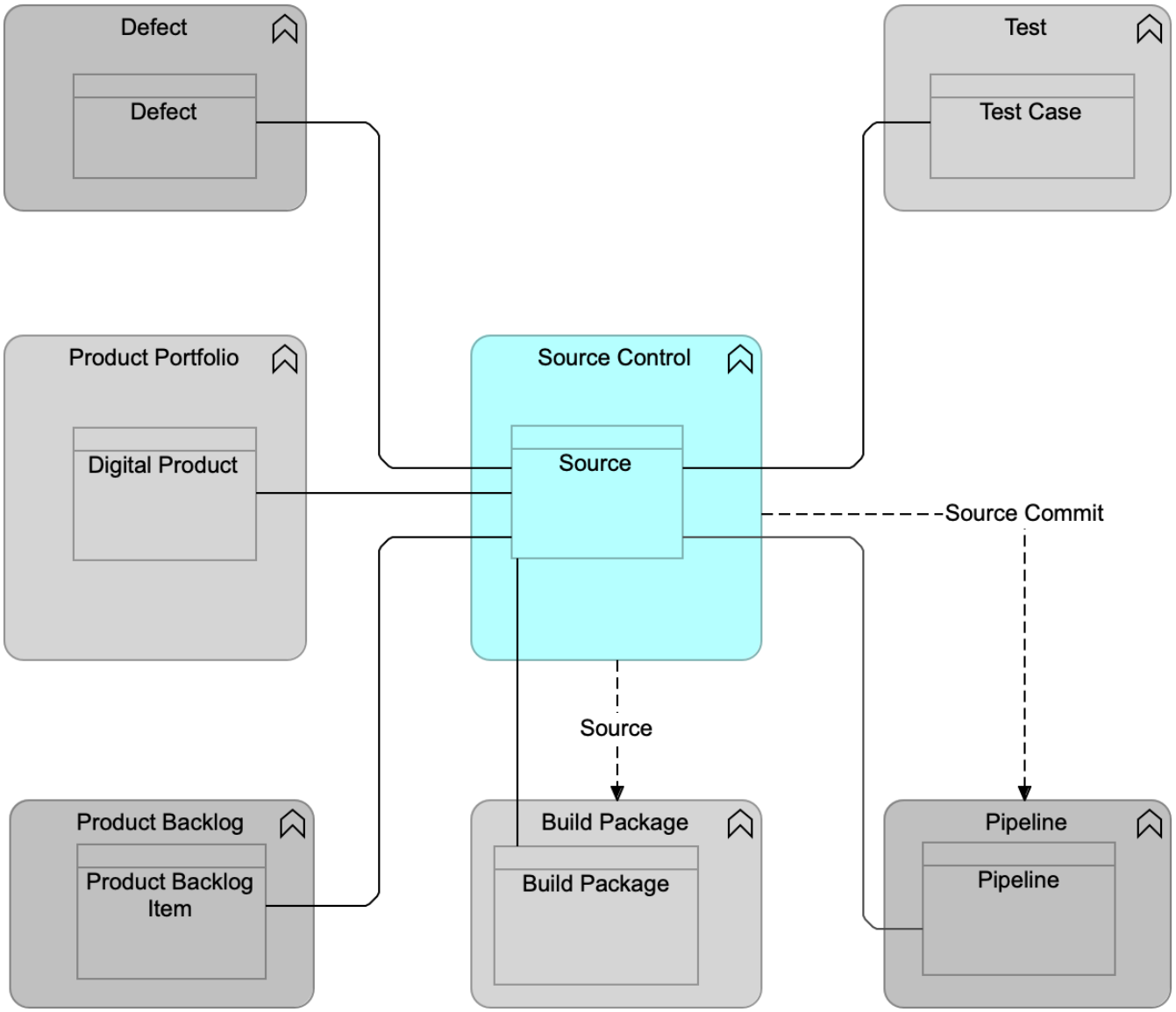

Figure 7-5. Source Control Functional Component Model

Purpose

The Source Control functional component manages all the source code (and tracks all changes to the source) for a Digital Product. It is the system of record for all sources and configuration changes which fall under version control. The Source Control functional component ensures that access and updates to the source code are securely controlled.

In this context, “Source” also includes code that captures changes and configurations made to package-based software and SaaS applications; for example, configuring or customizing standard packages such as modifying business rules and workflows, configuring APIs, database configuration. The Source Control functional component additionally manages the source code related to infrastructure (referred to as infrastructure as code).

Since all modifications in the source code repository are under change and version control, any source code changes need to be traced back to associated Product Backlog Items. As part of product development, various other automation routines are created such as test scripts, deployment templates, or workflows and monitors (e.g., monitoring as code). These automation scripts are also maintained by the Source Control functional component.

The Source Control functional component supports the Integrate value stream.

Functional Criteria

The Source Control functional component:

- Shall be the system of record (authoritative source) for all Sources
- Shall associate the source code repository to the Digital Product (owning the Source)
- Shall implement the necessary security controls to protect and manage access to source code
- Shall manage the lifecycle of the Source
- Shall ensure Source updates are traced back to one or more Product Backlog Items (they understand the origin of changes)
- Shall allow and promote the reuse of code between products and teams
- Shall cover all source codes including infrastructure as code, as well as source code for automated Test Cases, monitoring (monitoring as code)
- Can integrate with external sources such as open-source libraries and third-party vendor repositories
- Shall integrate with collaboration and communication tools to inform relevant stakeholders of Source updates
- Shall allow for peer reviews of source code changes/commits
- Shall allow analytics and statistics of source code changes over time
- Shall allow the ability to compare Source updates/changes (between the different branches and versions)
- Shall verify source code quality and conformance to coding and security standards and policies (using Test functional components such as Code Quality Scanning)
- Shall analyze the code (including all related open-source/third-party source codes) for security issues, vulnerabilities, and risks
- Shall code or configure Test Cases as part of product development (linked to the Test functional component); automated test scripts are maintained in the Source Control functional component as well
- Shall integrate with code editors to perform the creation and/or updates of source codes

7.1.4.1. Source Data Object

Purpose

The Source data object represents a collection of source code and configuration settings under version control as part of the configuration of a product. The Source data object fulfills the Requirements associated with the product.

NOTE The Source data object does not only represent the source code such as .NET, JavaScript™, or Java®, but also infrastructure as code, web pages, as well as configuration settings to define; for example, business rules, APIs, database schemas. Consider all use-cases including customizations or configuration changes to SaaS applications or COTS packages. Source represents all changes made to realize the design of the Digital Product which falls under version control and is used to build the Product Release.

Key Attributes

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

- **Id:** unique identifier of the Source
- **Version:** version of the Source
- **Repository:** link to the repository where the Source is maintained
- **Owner:** reference to the organizational owner of the repository and Source
- **Date/Time:** date and time of last update (and commit)
- **Person Id(s):** references to the identities who have created/modified the Source
- **Tags:** one or more tags associated with the Source
- **Comments:** comments related to the Source update

Key Data Object Relationships

The Source data object shall maintain the following relationships:

- Source to Pipeline (n:m): a Pipeline can be triggered by updates of the source code; a Pipeline is associated with one or more Sources
- Source to Product Backlog Item (n:m): the Source is created or updated with a reference to one or more Product Backlog Items fulfilling one or many Requirements, and for a given Product Backlog Item, there could be multiple Sources created/modified
- Source to Build Package (n:m): the Source can be built multiple times to create several build versions
- Source to Test Case (n:m): one or more Test Cases can be defined related to the source code, such as unit tests, code quality scans, and reviewing security coding standards

- Source to Defect (n:m): one or more Defects can be related to the source code; for example, a Defect can be related to code quality issues identified by the Test functional component
- Source to Digital Product (n:1): the Source repository is related to the Digital Product owning the Source

7.1.5. Pipeline Functional Component

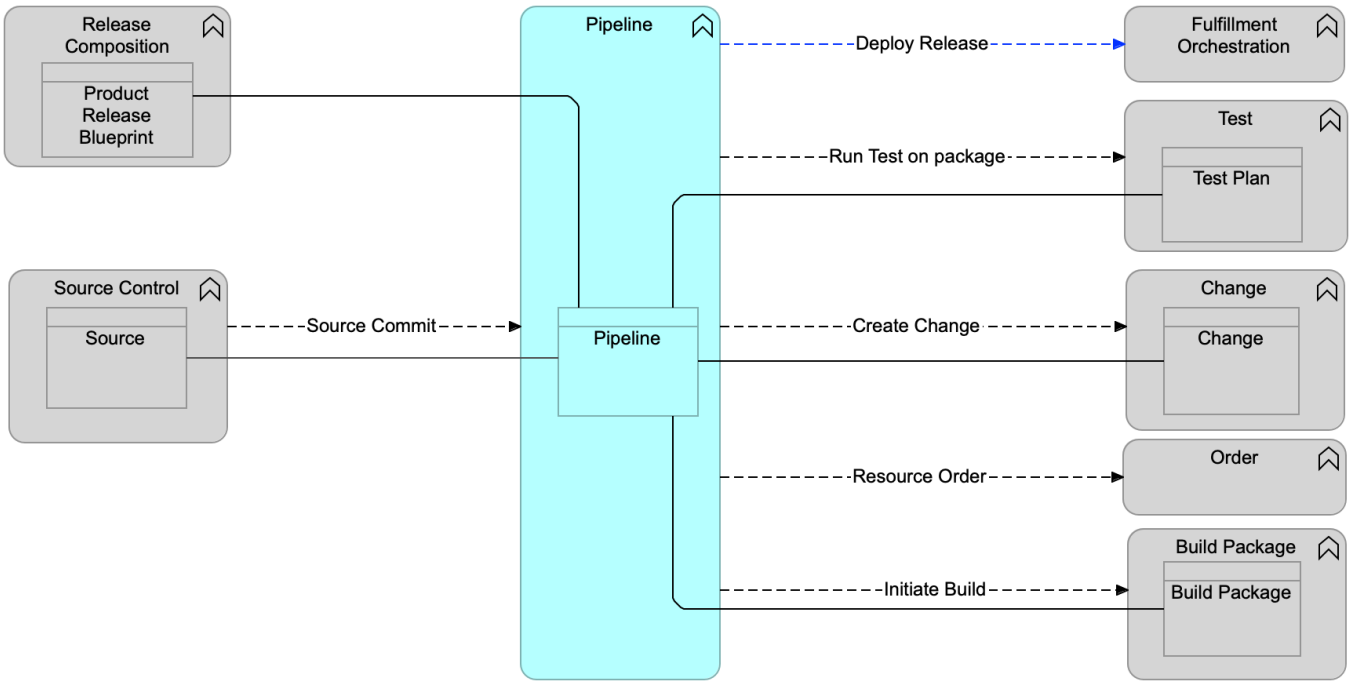

Figure 7-6. Pipeline Functional Component Model

Purpose

The Pipeline functional component orchestrates the activities needed to deliver a tested, validated, and compliant Product Release. Its purpose is to orchestrate and coordinate the tasks associated with building, verifying, testing, and deploying software changes, and the Pipeline functional component can manage different types of Pipelines, such as:

- Continuous Integration or build Pipelines
- Release Pipelines orchestrating the release across the different environments (such as test, acceptance, and production)

A Pipeline defines an orchestration workflow consisting of several steps (jobs or tasks) and stages to be performed, including:

- Execute build activities and create the Build Package (compiling and packaging) triggered by code changes
- Initiate code quality checks (executed by the Test functional component)

- Trigger the execution of Test Plans such as those related to unit testing, functional testing, and performance and security testing
- Initiate deployments to different environments such as test, stage/acceptance, and production (via the Order functional component)
- Validate the outcome of build, test, and deployment activities against agreed controls and policies

The Pipeline controls the movement of code from a commit in the Source Control functional component, through its development via the different environments such as test, acceptance, and production. The Pipeline functional component triggers the Fulfillment Orchestration functional component to perform the actual deployment. A Pipeline also embeds the required controls to ensure releases are processed through the Pipeline in a secure and compliant manner. The configuration of a Pipeline should be managed as code and placed under version control.

The Pipeline functional component supports the Integrate value stream.

Functional Criteria

The Pipeline functional component:

- Shall ensure that the Pipeline is under version control and secured (for unauthorized changes)
- May manage the Pipeline as code in the Source Control functional component
- Shall associate the Pipeline to the source code repository (e.g., a merge request can initiate the Pipeline to execute build activities)
- Shall manage the executions of the Pipelines (and keep track of all activities and outcomes)
- Can associate the Pipeline to a Digital Product
- Shall ensure that policies are embedded into the Pipeline so that security, risk, and compliance checks are executed as part of build and release activities
- Shall track the progress and status of the Pipeline activities/tasks (a Pipeline consists of one or more tasks)
- Shall associate a Pipeline to one or many Test Plans which are initiated as part of the Pipeline workflow
- Shall register the Build Package in the Build Package functional component
- Shall trigger the execution of one or more Test Plans
- Shall update the status of the associated Build Package reflecting the outcome of the test activities
- Shall associate the Product Release with the Pipeline (which orchestrated the creation of the release)
- Shall update the status of the associated Product Release based upon the outcome of the deployment and activities across the different environments
- Shall create a Change to log the planned and executed deployment of a Product Release
- Shall create a Change to inform stakeholders about the (planned) Change

- Shall use the Change functional component to perform a change impact assessment and determine potential conflicts (identifying potential conflicts of multiple changes)
- Can use the Change functional component to schedule the actual deployment in the appropriate change window (e.g., in case a deployment results into downtime)
- Shall wait for the approval of the Change functional component to continue with the workflow of the Pipeline
- Shall initiate the deployment of the Product Release through the Fulfillment Orchestration functional component (e.g., to a test, acceptance, or production environment)

7.1.5.1. Pipeline Data Object

Purpose

The Pipeline data object defines and controls the execution of all the activities (and stages) to orchestrate and validate the build, and the deployment and testing activities required to securely deliver a modified Product Release.

Key Attributes

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

- **Id:** unique identifier of the Pipeline
- **Name:** name of the Pipeline
- **Version:** version of the Pipeline

Key Data Object Relationships

The Pipeline data object shall maintain the following relationships:

- Pipeline to source code (n:m): the Pipeline is linked to a source code repository

  Multiple Pipelines can be defined for one repository. Source code updates trigger the Pipeline to perform a number of tasks such as initiate static code analysis and build activities to create the Build Package (consisting of one or more build artifacts).

- Pipeline to Test Plan (n:m): a Pipeline triggers the execution of several Test Plans (and associated Test Cases), such as those related to unit testing, performance testing, security testing
- Pipeline to Build Package (1:n): a Build Package is created by one Pipeline (using one or more build tools)
- Pipeline to Product Release Blueprint (1:n): the creation of a Product Release Blueprint is initiated and governed by one Pipeline
- Pipeline to Change (1:n): a Pipeline can raise one or more Changes to inform stakeholders about new releases/changes, perform change impact assessment, and get formal approval (manually or automated)

7.1.6. Build Package Functional Component

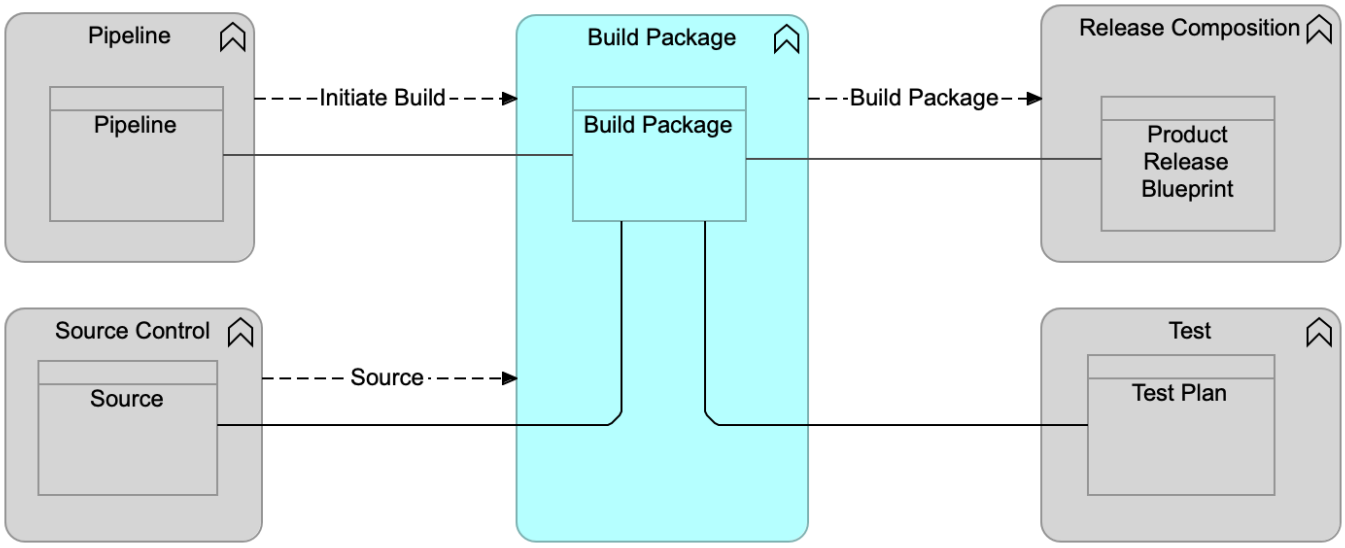

Figure 7-7. Build Package Functional Component Model

Purpose

The Build Package functional component is the authoritative source for all Build Packages and associated build artifacts. It receives the Build Package and any associated artifacts from the Pipeline functional component.

The Build Package functional component is the single source for the various types of Build Packages including software builds, infrastructure images, container images.

The Build Package functional component supports the Integrate value stream.

Functional Criteria

The Build Package functional component:

- Shall be the authoritative source for all Build Packages (often referred to as an artifact/build repository) including, for example, container images
- Shall manage the lifecycle of all Build Packages from registration to archiving and clean-up of build artifacts
- Shall secure the Build Packages for unauthorized access and unauthorized changes
- Shall maintain a full audit trail of all additions and modifications in Build Packages and their components
- May receive Build Packages and artifacts from external sources, such as third-party vendors and open-source libraries
- Shall receive the Build Package from the Pipeline for a particular Digital Product or product component

- Shall associate the Build Package with the Pipeline through which the Build Package is created
- Shall ensure that all Build Packages (e.g., container images) are safe to deploy
- Shall maintain a bill of materials for all components (and artifacts) that are part of a Product Release (including dependencies and references to third-partly libraries and components)
- Shall scan and verify all Build Packages (and their components and dependencies) for potential security issues, compliance issues, and vulnerabilities (prior to deployment)
- Shall associate one or many Build Packages to one or many Test Plans which are executed as part of the Build Package registration; for example, to perform a Software Composition Analysis (SCA) for all software packages and containers for vulnerabilities and software license compliance issues

7.1.6.1. Build Package Data Object

Purpose

The Build Package data object represents a collection of build artifacts (components) packaged into a deployable unit. The Build Packages are securely stored in a build/artifact repository.

Key Attributes

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

- **Id:** unique identifier of the Build Package
- **Name:** meaningful name of the Build Package
- **Description:** short description of the Build Package
- **Type:** define the Build Package type; e.g., container image
- **Status:** lifecycle status of the Build Package
- **Compliance Status:** compliance status of the Build Packages (and their components)
- **Version:** version of the Build Package
- **Tags and Metadata:** a build can have one or more tags (or properties) to provide additional metadata

Key Data Object Relationships

The Build Package data object shall maintain the following relationships:

- Source to Build Package (n:m): Source can be built multiple times to create several build versions
- Build Package to Test Plan (n:m): one or many Build Packages can be related to one or many Test Plans used as part of the build creation (e.g., perform security and vulnerability scanning on the new or modified Build Packages)
- Build Package to Product Release Blueprint (n:m): multiple Build Packages, which represent the deployable units or components, can be assembled into a Product Release Blueprint which is used for the deployment of the Product Release into the different environments

- Build Package to Pipeline (n:1): a Build Package is associated with the Pipeline through which the Build Package is created (or modified)
- Build Package to Build Package (n:m): a Build Package can have dependencies with other Build Packages (or components)

7.1.7. Release Composition Functional Component

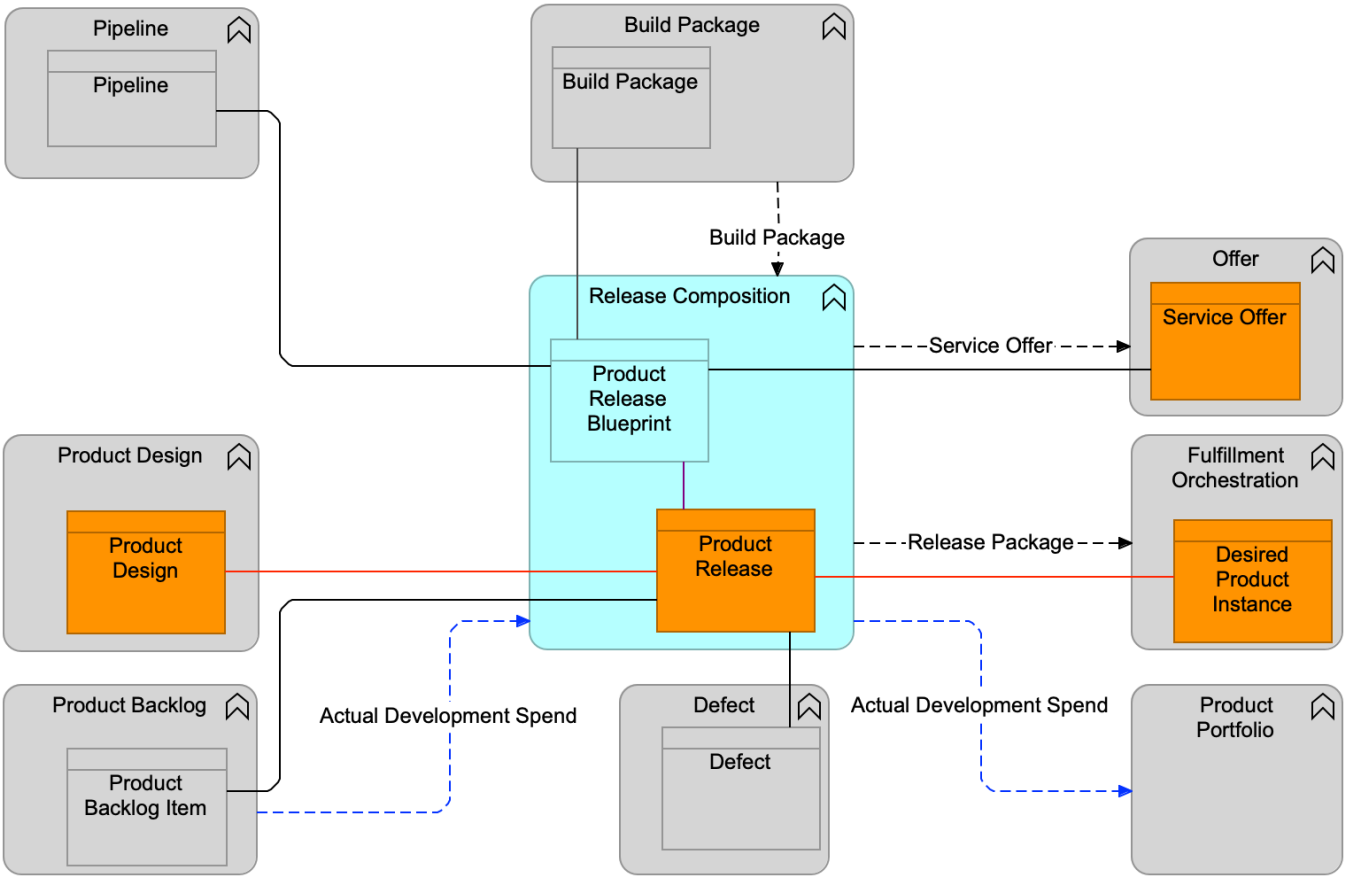

Figure 7-8. Release Composition Functional Component Model

Purpose

The Release Composition functional component manages the Product Release and creates the associated Product Release Blueprint for delivering new or changed products to the Fulfillment functional component. The Fulfillment functional component uses the Product Release Blueprint to deploy and configure an instance of a Digital Product; for example, an instance used for testing, staging, or production. To ensure that the release can be operated and supported as part of digital operations enablement, the creation of, for example, monitors, Service Offers, Knowledge Items, backup/restore procedures for the product also form part of this component.

The Product Release should include all components and artifacts to be delivered as part of a release, such as:

- Build Packages used to create the Product Release Blueprint (from the Build Package functional component), which is consumed by the Fulfillment functional component

- Definitions (and templates) for the new or modified Service Offers with associated request fulfillment automation logic
- Monitor definitions (to be consumed by the Monitoring functional component) to ensure the new Product Release can be monitored
- Test results (and full Log of executed tests) from the Test functional component
- Outstanding Defects from the Defect functional component (as input for Knowledge Management to capture known errors)
- Definitions of the underlying infrastructures and platforms; e.g., infrastructure as code (and associated parameters)
- Release notes, as well as new or modified Knowledge Items and associated user guides/support information

The Release Composition functional component supports the value streams:

- Integrate
- Deploy
- Release

Functional Criteria

The Release Composition functional component:

- Shall be the system of record (authoritative source) for all Product Releases
- Shall be the system of record for all Product Release Blueprints
- Shall create the Product Release and the associated Product Release Blueprint used by the Fulfillment functional component to create an instance (or deployment) of the Digital Product that, for example, will be utilized by the Test functional component
- Shall manage the Product Release and Product Release Blueprints for delivering new or changed services to the Fulfillment functional component to facilitate a smooth transition to digital operations
- Shall include the templates for the associated Service Offers (as part of the Product Release Blueprint)
- Shall include the templates and definitions for the configuration of Service Monitors executed by the Monitoring functional component (monitoring as code)

7.1.7.1. Product Release Data Object

Purpose

The Product Release data object represents a candidate or actual release of the Digital Product. A release relates to a single Product Design. It represents a version of the Digital Product with all associated documentation and artifacts required to fully support the product, including, for example,

release notes, user guides, training materials, updated Knowledge Items. It must contain a set of features that are sufficient to produce an outcome aligned to the value expectations of the Digital Product.

Key Attributes

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

- **Id:** unique identifier of the Product Release
- **Name:** meaningful name of the Product Release
- **Description:** description of the Product Release
- **Status:** lifecycle status of the Product Release
- **Version:** version of the Product Release
- **Tags:** a Product Release can have one or more tags to further categorize and classify the release
- **Deployment Model:** a representation of the deployable service components and their various configuration options
- **Actual Spend:** actual spend for the development of a Product Release

Key Data Object Relationships

The Product Release data object shall maintain the following relationships:

- Product Design to Product Release (1:n): a Product Design can lead to the creation of one or more Product Releases in order to deliver the required service outcomes
- Product Backlog Item to Product Release (n:1): a Product Backlog Item represents the changes and work performed on a specific Product Release; a Product Backlog Item should be decomposed into sub-product items which can be fulfilled within one release
- Product Release to Product Release Blueprint (n:1): each Product Release has one associated Product Release Blueprint
- Product Release to Defect (n:m): identified Defects (in the form of Problems/known errors) associated with the Product Release
- Product Release to Desired Product Instance (1:n): one Product Release can be translated to one or more Desired Product Instance(s)
- Product Release to Product Release (n:m): a Product Release is dependent on other products being available in a particular version

7.1.7.2. Product Release Blueprint Data Object

Purpose

The Product Release Blueprint data object provides the definition and specification of a release with all related components, dependencies, and required configurations as part of the Digital Product. It

contains the description and procedures in order to activate, deploy, and operate an instantiation and its underlying components including software components and required resources. It may have many variations for a given Product Release as considerations such as environmental (development, stage, and production) or hosting (on-premise or cloud) could be specified differently. Iterative development methods may reuse a Product Release Blueprint multiple times.

Key Attributes

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

- **Id:** unique identifier of the Product Release Blueprint
- **Name:** meaningful name of the Product Release Blueprint
- **Description:** description of the Product Release Blueprint
- **Version:** a version of the Product Release Blueprint
- **Deployment Model:** a definition of the various parameters and configuration options (e.g., used for deployment/provisioning)

Key Data Object Relationships

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

- Product Release Blueprint to Pipeline (n:1): a Product Release Blueprint is created by a Pipeline
- Product Release to Product Release Blueprint (1:n): a Product Release has one Product Release Blueprint
- Product Release Blueprint to Build Package (n:m): multiple Build Packages, which represent the deployable content of the service components, can be deployed using the instructions contained in the Product Release Blueprints
- Service Offer to Product Release Blueprint (n:m): a Service Offer is created based on definitions of one or more Product Release Blueprints

7.2. Test Function

The Test function is centered around verifying and testing the Source, Build Package, Product Release, and associated documentation to ensure the product meets the Requirements and adheres to standards and policies, such as those related to security and compliance.

7.2.1. Test Functional Component

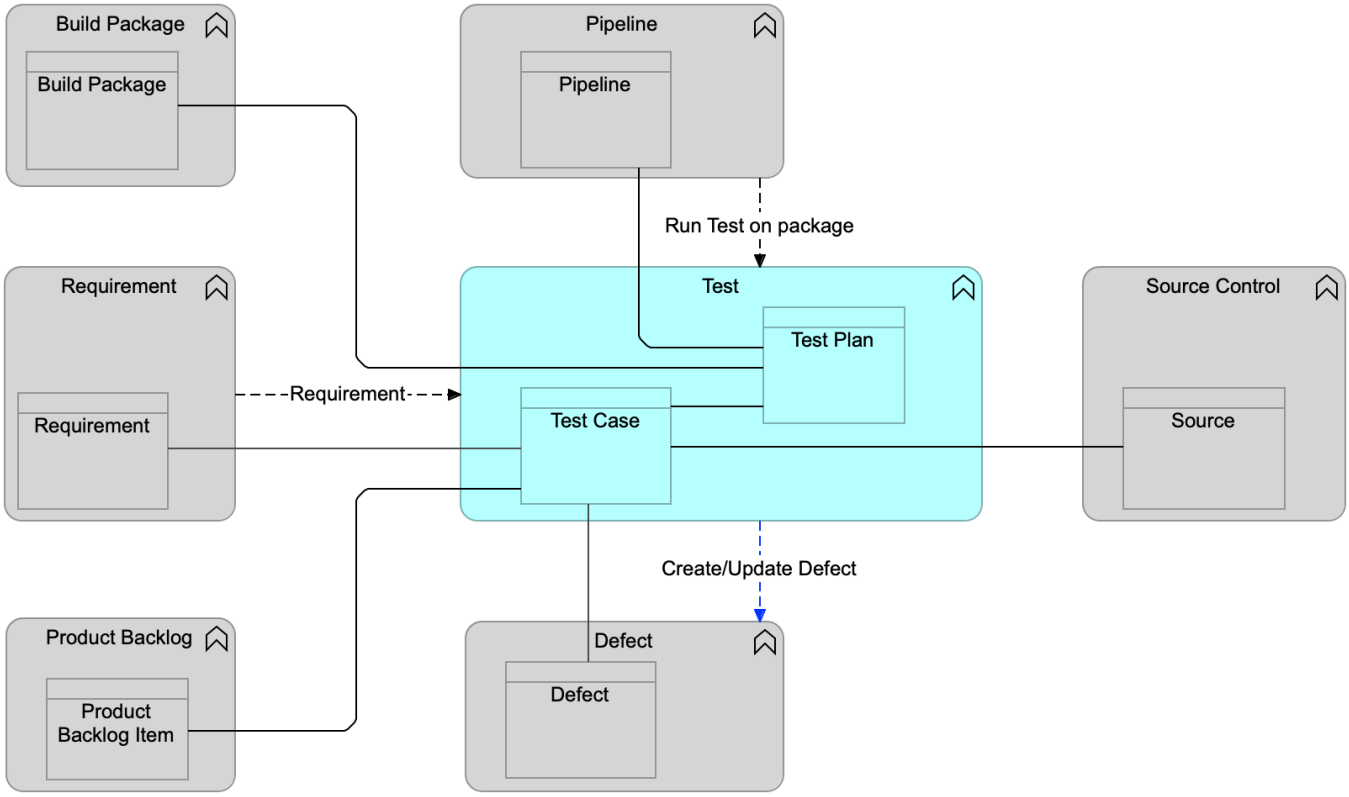

Figure 7-9. Test Functional Component Model

Purpose

The Test functional component plans, orchestrates, and executes tests (including reviews) that ensure the Product Release will meet stakeholder expectations, and support all the Requirements at the agreed service levels. Using the Test functional component, Defect data objects are created to be consumed by the Defect functional component. Tests Cases are designed, planned, and executed, covering both manual and automated testing.

The Test functional component covers all verifications, validations, and test activities to ensure the product meets the Requirements and specifications (as part of the Product Design) including non-functional requirements and adherence to policies and standards.

Issues and exceptions found during testing are recorded as Defect data objects, which are consumed by the Defect functional component.

Various tests are planned and executed across the different environments, including:

- Unit testing
- Peer reviews
- Code quality testing (to verify the conformance of the code to policies and coding standards)
- Performance and stress testing

- User Interface (UI) testing
- Functional testing
- Usability testing
- Security and compliance testing covering various security tests such as Static and Dynamic Application Security Testing (referred to as SAST and DAST) to validate compliance to security requirements and policies, as well as vulnerability scanning
- Testing of operations functions, including backup/restore, monitoring, logging, and Runbooks (e.g., used for diagnostics and remediation)
- Business process testing (testing an end-to-end business process)
- Mobile testing (for the different mobile platforms)
- API and integration testing
- Regression testing
- Continuity testing (e.g., tests the disaster recovery/continuity plan)
- Monkey testing (tests the product by providing random inputs and checking the behavior)
- User Acceptance Testing (UAT)
- Continuity (and failover) test

The Test functional component should support different test methods, including Test-Driven Development (TDD) in which each Product Backlog Item begins with writing a test (instead of writing a test after the code has been written). Test Cases can be created through use-cases and user stories to cover the requirements and exception conditions.

The Test functional component supports the Integrate value stream.

Functional Criteria

The Test functional component:

- Shall maintain the test strategy and Test Plans
- Shall link Test Cases to one or more Test Plans
- Shall leverage the Product Backlog functional component to plan and develop Test Cases, both manual and automated tests
- Shall manage the automated Test Cases in the Source Control functional component
- Shall ensure traceability between tests and Requirements
- Shall plan and execute tests that ensure the Product Release will support the Requirements at the agreed service levels
- Shall plan and execute tests to ensure non-functional requirements are addressed
- Shall create Defect data objects that are consumed by the Defect functional component

- Shall log all test runs and capture the test results (including test logs)
- Shall include abuse cases to be tested for the product; misuse and abuse cases describe how users could potentially misuse or exploit the weaknesses of controls in software features
- Shall execute tests such as:
  - Unit tests
  - Code quality scanning
  - Functionality tests
  - Security tests (static and dynamic application security and infrastructure security testing)
  - Vulnerability scanning
  - Penetration tests
  - Performance and stress tests
  - Business process tests
  - Regression tests
  - Integration tests
- Shall create Defects found during testing which are consumed by the Defect functional component
- Shall define and manage representative test data for all data sets needed to execute the Test Cases
- Shall provide test execution reports per tested Requirements and/or Product Backlog Items
- Shall track all executed Test Cases (as part of a Product Release)
- Shall test operations activities as well; for example, monitoring, request fulfillment
- Shall be the system of record (authoritative source) for all Test Cases (and associated test data)
- Shall manage the lifecycle of Test Cases
- Shall calculate test coverage of a new Build Package and Product Release
- Shall measure and report upon code quality and compliance to coding standards
- Shall scan all builds against potential vulnerabilities (e.g., third-party libraries and open-source components)
- Shall create automated test scripts, including unit testing and scripts for security testing that follow a formal software security assurance methodology
- Shall run security tests on core code to identify existing security issues at the start of the development cycle so that the assessment of scope/requirements set/scheduled can be scheduled early for existing services that are undergoing change

7.2.1.1. Test Case Data Object

Purpose

The Test Case data object defines how to validate a particular aspect of a Product Release. It is used to document what makes a product fit-for-purpose.

Key Attributes

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

- **Id:** unique identifier of the Test Case
- **Name:** meaningful name for the Test Case
- **Description:** summary or short description of the Test Case
- **Status:** lifecycle status of the Test Case
- **Result:** last known Test Case execution outcome

Key Data Object Relationships

The Test Case data object shall maintain the following relationships:

- Test Case to Requirement (n:m): a Test Case is associated to one or more Requirements (which are covered by the Test Case)
- Test Case to Product Backlog Item (n:m): a Test Case can be associated to one or more Product Backlog Items
- Test Case to Test Plan (n:m): one or more Test Cases are related to one or more Test Plans
- Test Case to Build Package (n:m): a Build Package is validated by executing one or more Test Cases as part of the build creation
- Test Case to Defect (1:n): a Test Case can be associated to one or more Defects that are reported as a result of this test
- Test Case to source code (1:n): a Test Case is associated to the source code which is executed as part of verifying code changes (e.g., code quality analysis)
- Test Case to Test Case (1:n): a Test Case can be associated to another Test Case

7.2.1.2. Test Plan Data Object

Purpose

The Test Plan data object is a dynamic plan defining the test scope, test coverage, and associated Test Cases to be executed as part of new builds and/or releases. Multiple Test Plans can be created for a Digital Product. The Test Plan ensures that all required Test Cases are executed and verified as part of a Product Release.

Key Attributes

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

- **Id:** unique identifier of the Test Plan
- **Name:** meaningful name for the Test Plan
- **Description:** summary or short description of the Test Plan
- **Status:** lifecycle status of the Test Plan
- **Version:** version of the Test Plan
- **Test Runs:** overview of Test Plan executions

Key Data Object Relationships

The Test Case data object shall maintain the following relationships:

- Test Case to Test Plan (n:m): one or more Test Cases are related to one or more Test Plans
- Product Release to Test Plan (n:m): a Product Release can have one or more associated Test Plans
- Test Plan to Pipeline (n:1): a Test Plan can be associated to and initiated by one Pipeline
- Test Plan to Test Plan (n:m): a Test Plan can link to another Test Plan (decomposition or relationship)
- Test Plan to Build Package (n:m): a Test Plan can be created to scan and verify one or more Build Packages (e.g., for software compliance and potential vulnerabilities)

7.2.2. Defect Functional Component

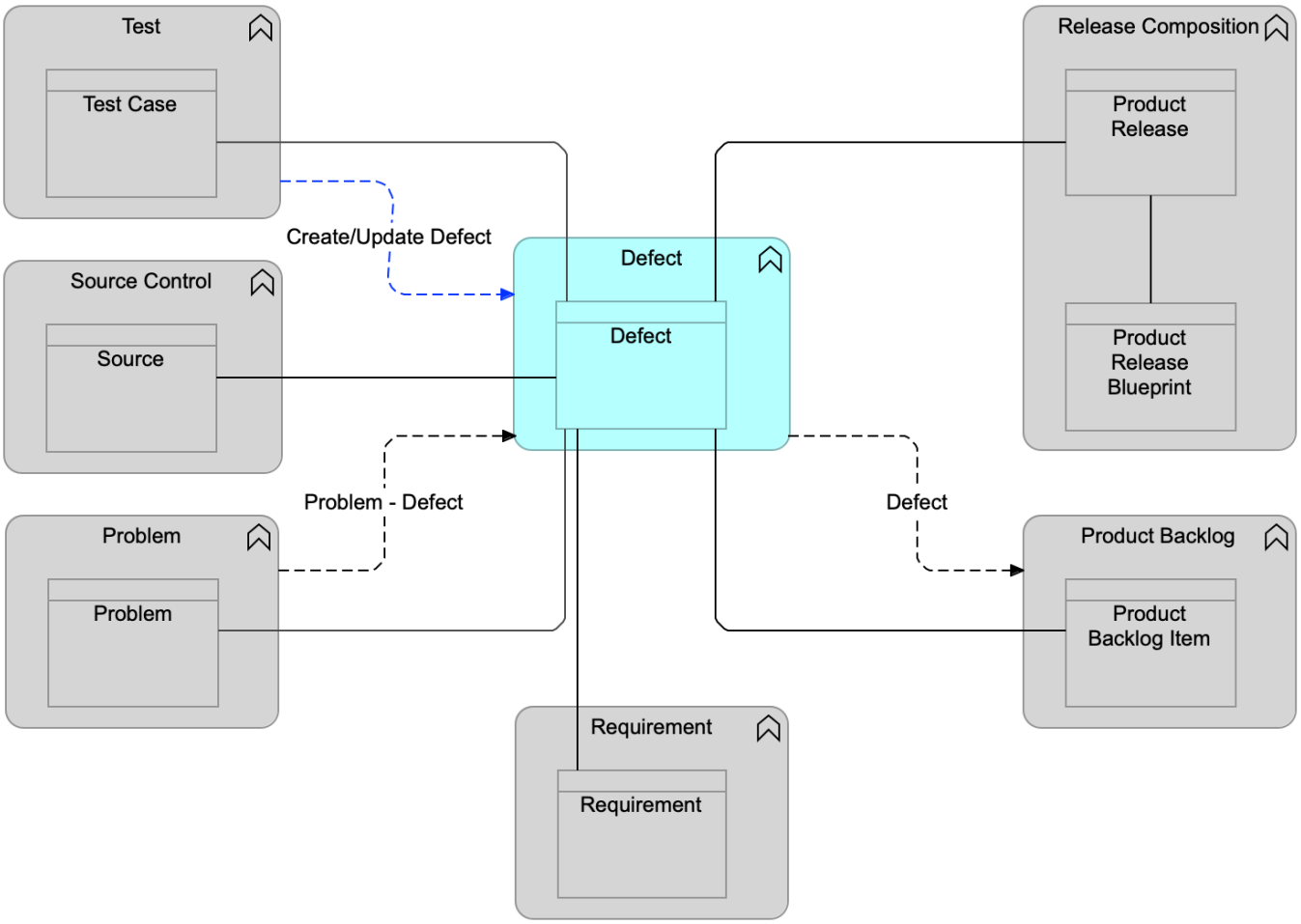

Figure 7-10. Defect Functional Component Model

Purpose

The Defect functional component keeps track of all Defects; including their origin, status, importance, and relation to Requirements and Problems (or known errors). Defects of all types are registered (including security-related Defects), together with all relevant details such as description, severity, product version, related requirement. Defects can be analyzed, and resolutions found. Defects can be associated with Requirements.

NOTE

A Defect in this context can be any issue that impacts the product in delivering the expected value and outcome for the consumer as described in the product definition and requirements, such as poor user experience, potential vulnerabilities, maintainability issues, technical debt. This includes unfulfilled nonfunctional requirements mandated by policy even if they were omitted from the formal requirements process. In this case defect management should include updating any missing product requirements and design.

The Defect functional component documents issues that should be communicated to the Release Composition functional component. The Defect functional component consumes Defects from the

Problem functional component and the Test functional component, and these are in turn consumed by the Source Control functional component for review and resolution. Defect details can be updated, and used to decide on target release times. The Defect status can be monitored, and Defect reports provided. Defects that remain unresolved by service development can be converted to known errors for Problem Management to document, develop work-arounds for, and/or report in Knowledge Management articles.

The Defect functional component supports the Integrate value stream.

Functional Criteria

The Defect functional component:

- Shall be the system of record (authoritative source) for all Defects
- Shall manage the lifecycle of the Defect
- Shall keep track of all Defects, including their origin, status, importance, and relation to Requirements and known errors
- Shall register Defects of all types (including security-related) with all relevant details such as description, severity, application version, related requirement
- Shall analyze Defects and find resolutions
- May document issues that should be communicated to the Release Composition functional component
- May consume Defects from the 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
- Shall report Defect status and provide Defect reports
- Shall create known errors for unresolved Defects (including the potential work-around and associated Knowledge Items) which can be published to consumers and used by the Operate value stream to provide remediation in case of Incidents
- Shall receive Defect information from the Test functional component
- Shall receive Defect information from a Problem (or known error)
- Shall provide Defect information to the Product Backlog functional component (to plan and work on the Defect resolution in the Product Backlog)

7.2.2.1. Defect Data Object

Purpose

The Defect data object records an issue with the Product Release that should be remediated to fulfill the associated Requirements or designated as a Known Error, including unfulfilled nonfunctional requirements mandated by policy even if they were omitted from the formal requirements process. In this case defect management should include updating any missing product requirements and design.

Key Attributes

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

- **Id:** unique identifier of the Defect
- **Title:** short description of the main issue discovered
- **Description:** description of the Defect
- **Status:** status of the Defect

Key Data Object Relationships

The Defect data object shall maintain the following relationships:

- Defect to Test Case (n:1): a Defect can be associated with the Test Case that detected the Defect
- Defect to Product Release (n:m): a Defect can be associated with one or more Product Releases
- Problem to Defect (1:n): a Problem (or known error) may be the source for submitting one or more Defects (or updating existing ones)
- Defect to Product Backlog Item (1:n): a Defect is added to the Product Backlog as a Product Backlog Item (for fixing the Defect)
- Defect to Requirement (n:m): a Defect can be associated with one or more Requirements

Chapter 8. Request to Fulfill Functions

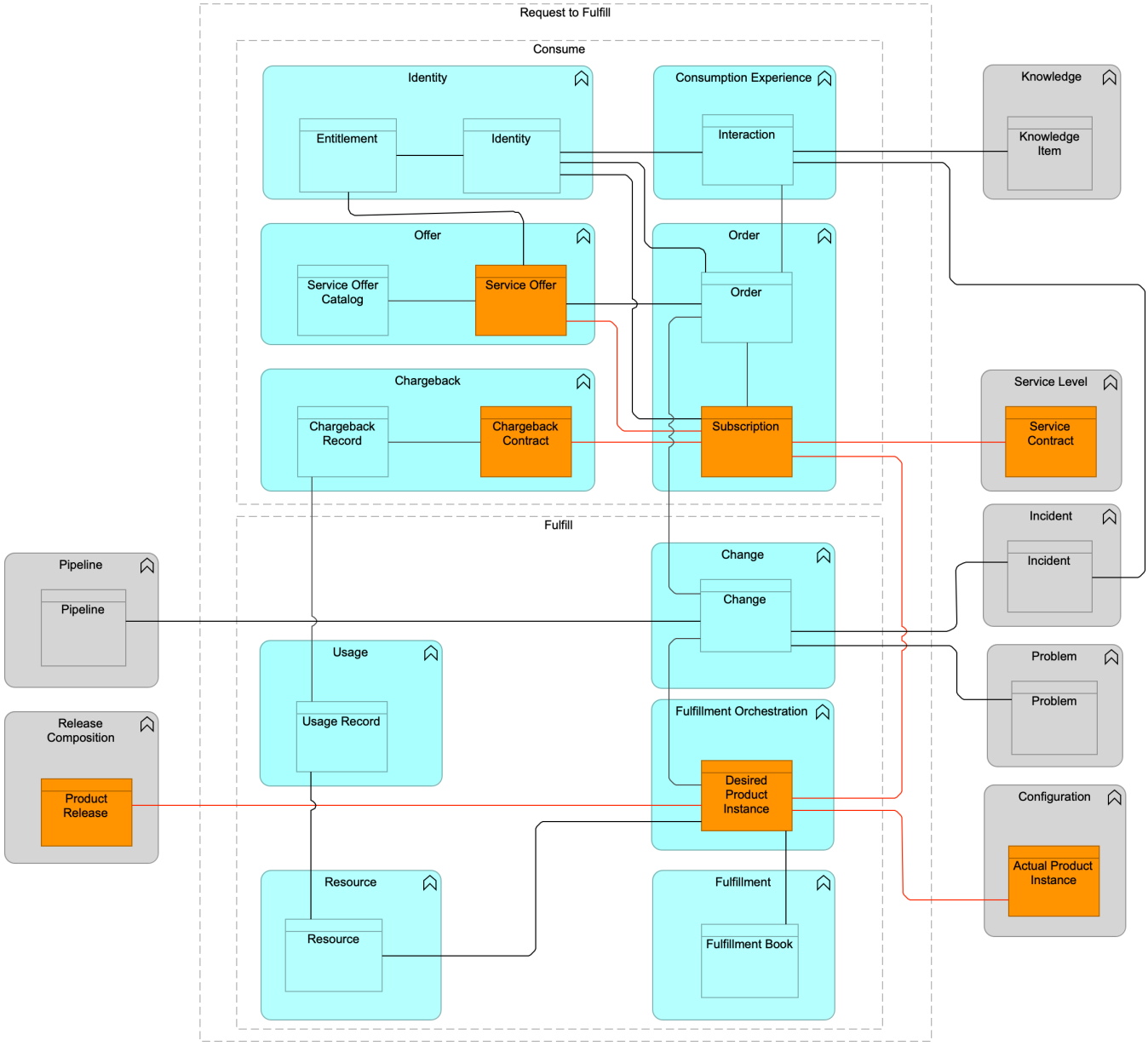

Figure 8-1. Request to Fulfill Functions Model

Description

The Request to Fulfill functions, informally referred to as "Deliver", represent a modern, consumption-driven engagement model that goes beyond the traditional service request management. It is a framework for connecting various consumers (business users, technology practitioners, or end customers) with the goods and services that they need to drive productivity and innovation. It fosters service consumption and fulfillment, service costing, knowledge sharing, self-service support, and collaboration between communities of interest to improve the overall engagement experience with IT. Many organizations use multiple technology services and/or service catalogs to address the needs of their consumers.

The Request to Fulfill functions bring these different catalogs and consumer personas into a single consumption experience, thereby eliminating complexity and confusion for consumers in browsing through multiple service catalogs to choose what services they need.

The Request to Fulfill functions contain the following functional components:

- Consume functionality:
  - Consumption Experience component
  - Identity Management component
  - Offer Management component
  - Order component
  - Chargeback component
- Fulfill functionality:
  - Change component
  - Fulfillment Orchestration component
  - Resource component
  - Fulfillment component
  - Usage component

One of the main objectives of the Request to Fulfill functions is to drive the system of engagement by facilitating a unified engagement framework between consumers and the other related IT4IT Functional Components. Below are some of the system of engagement goals:

- Drive the consumption through the unified aggregated 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

Related Value Streams

The following value streams use one or more functional components from the Request to Fulfill functions:

- Release
- Consume
- Deploy

Business Benefits

The Request to Fulfill functions place 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. The Request to Fulfill functions enable the aggregation of catalogs and Service Offers from multiple service providers into a single consumption experience. Therefore, while there is complexity on the delivery side in managing the various catalogs and Service Offers, it is not exposed to the consumer and the ordering experience is seamless and inviting. This also enables effective chargeback and service costing mechanisms, an important requirement in a multi-sourcing environment.

The main benefits of using the Request to Fulfill functions 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 Service Offer Catalog and Identity Management and Consumption Experience functional component to reduce complexity in the consumer experience
- Provides an architectural foundation for moving from traditional request management to service brokerage that increases both business and effectiveness
- Increased fulfillment efficiency and consistency through standard change deployment and automation
- Provides holistic visibility and traceability across the service subscription, usage, and chargeback to improve Financial Management
- Enables increased cost optimization; for example, by canceling expired Subscriptions and reclaiming resources, Subscriptions, and/or licenses that are unused

8.1. Consume Function

The Consume functionality is centered around making it easy and efficient to consume digital services.

8.1.1. Consumption Experience Functional Component

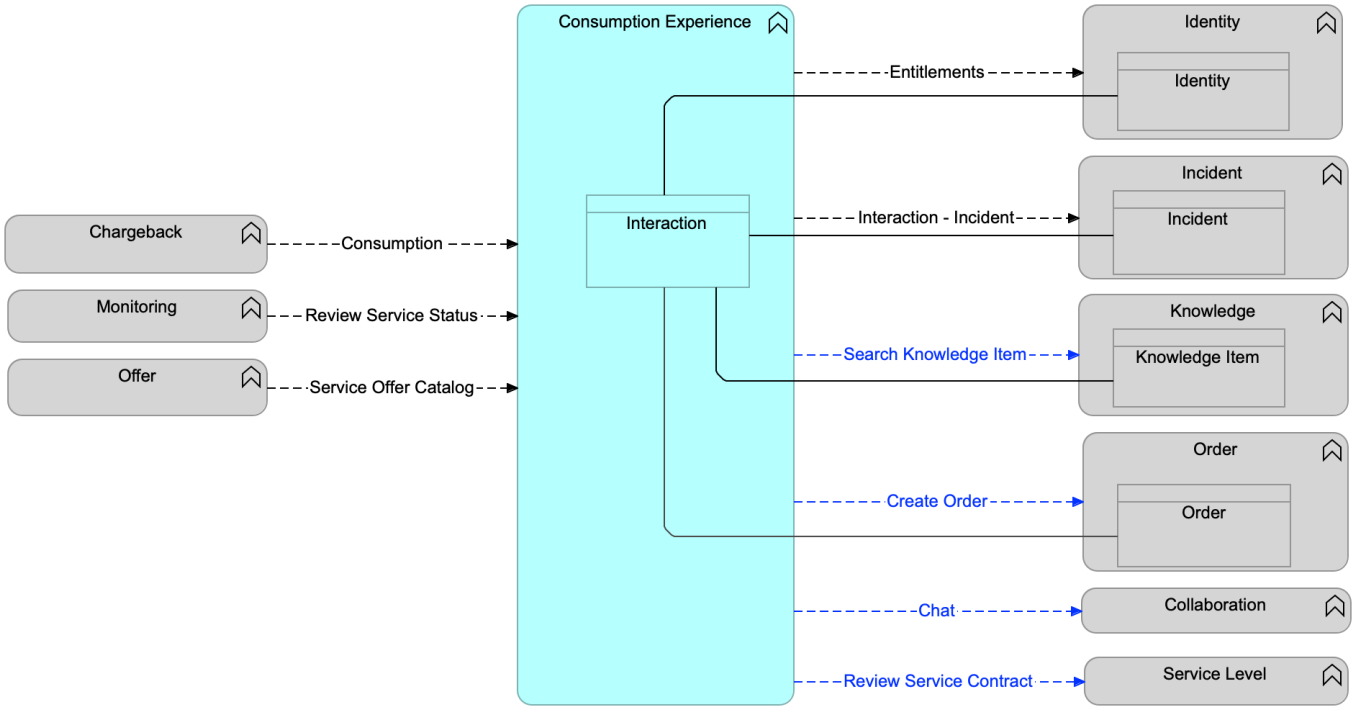

Figure 8-2. Consumption Experience Functional Component Model

Purpose

The Consumption Experience functional component is based on a system of engagement integration design pattern where consumers access different functional components through a common user experience. This enables a centralized location for all kinds of communication channels for consumers (such as chatbots, web applications, mobile applications) to provide a one-stop portal that facilitates the following features:

- Request a new service
- Request service support
- Request service modification
- Request service termination

This facilitates service consumption by connecting any potential consumer with the right information, goods, services, or capability at the right time through a single and intuitive experience, supporting “click, call, face” principles to get access to the desired support. Self-service concepts are used to limit the interaction flow to people and provide direct remediation, thereby enhancing customer experience.

The Consumption Experience functional component also embraces Customer Journey Mapping; typically the path followed by a consumer when they interact with the Consumption Experience functional component. It includes every touchpoint the consumer might run into throughout their interactions with IT, which helps in creating the overall self-service functionality in the Consumer Experience functional component that drives improved customer experience.

This provides an interface supported across multiple devices (e.g., smartphones, tablets) and also a 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.

The Consumption Experience functional component supports the value streams:

- Consume
- Operate

Functional Criteria

The Consumption Experience functional component:

- Shall be available to all users that desire to consume digital services
- Shall expose functionality based on user profile and entitlement
- Shall expose relevant aspects of the IT4IT functions and capabilities in a single place, unifying the experience – these functions and capabilities may be exposed in many forms; for instance, a form similar to smartphone apps, a traditional web application, or just as an API
- Shall determine each interaction, whether it is a Request for Information (RFI), a new Order, a Change, or an Incident, and route them to the appropriate functional component
- Shall route requests to queues for assignment or assign them to fulfillers automatically
- Shall provide the interface for consumers to search and read Knowledge Item data objects of all types and sources

  Knowledge Item data objects may include but are not limited to technology or supplier-created technical briefs, training videos, and user-created content.

- Shall reduce the load on the support organization by enabling and promoting self-help and self-healing behavior through the use of, for example, community assistance, knowledge sharing, content
- 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
- Shall provide the consumer a view and status update of planned and executed changes
- Can route users to access the Knowledge Item data objects before a new support ticket is created

- Can engage Collaboration & Communication secondary functional components to provide the user front end (such as a chat capability) for various purposes including service desk interaction, order routing, or information/knowledge gathering
- Can provide service consumers with a way to address more of their Digital Product and service-related issues, as well as receive information regarding their existing records without necessarily engaging underpinning providers

8.1.1.1. Interaction Data Object

Purpose

The Interaction data object manages a Consumption Experience for consumers that represents a request for assistance. The Interaction record can be created either through a virtual agent conversation (chatbot), a traditional self-service interface, or service desk agents. The Interaction determines whether the engagement is a new service order; for example, a Change, an Incident, or a simple RFI such as knowledge, contract, chargeback information

Key Attributes

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

- **Id:** unique identifier of the Interaction record
- **Description:** description of the consumer needs
- **Status:** current stage of the Interaction
- **Opened For:** consumer who initiated the Interaction
- **Assigned To:** name of the fulfiller: live agent name or virtual agent
- **Opened:** date and time when the Interaction started
- **Updated:** date and time when the Interaction record was last updated

Key Data Object Relationships

The Interaction data object shall maintain the following relationships:

- Interaction to Identity (n:1): connection to Identity for obtaining the consumer profile and contact information
- Interaction to Order (n:1): initiates the Order as per the information provided in the Interaction
- Interaction to Incident (1: n): provides the interface for consumers to submit their own Incidents or break-fix requests
- Interaction to Knowledge Item (1:n): provides the interface for consumers to search for information about products and services

8.1.2. Identity Functional Component

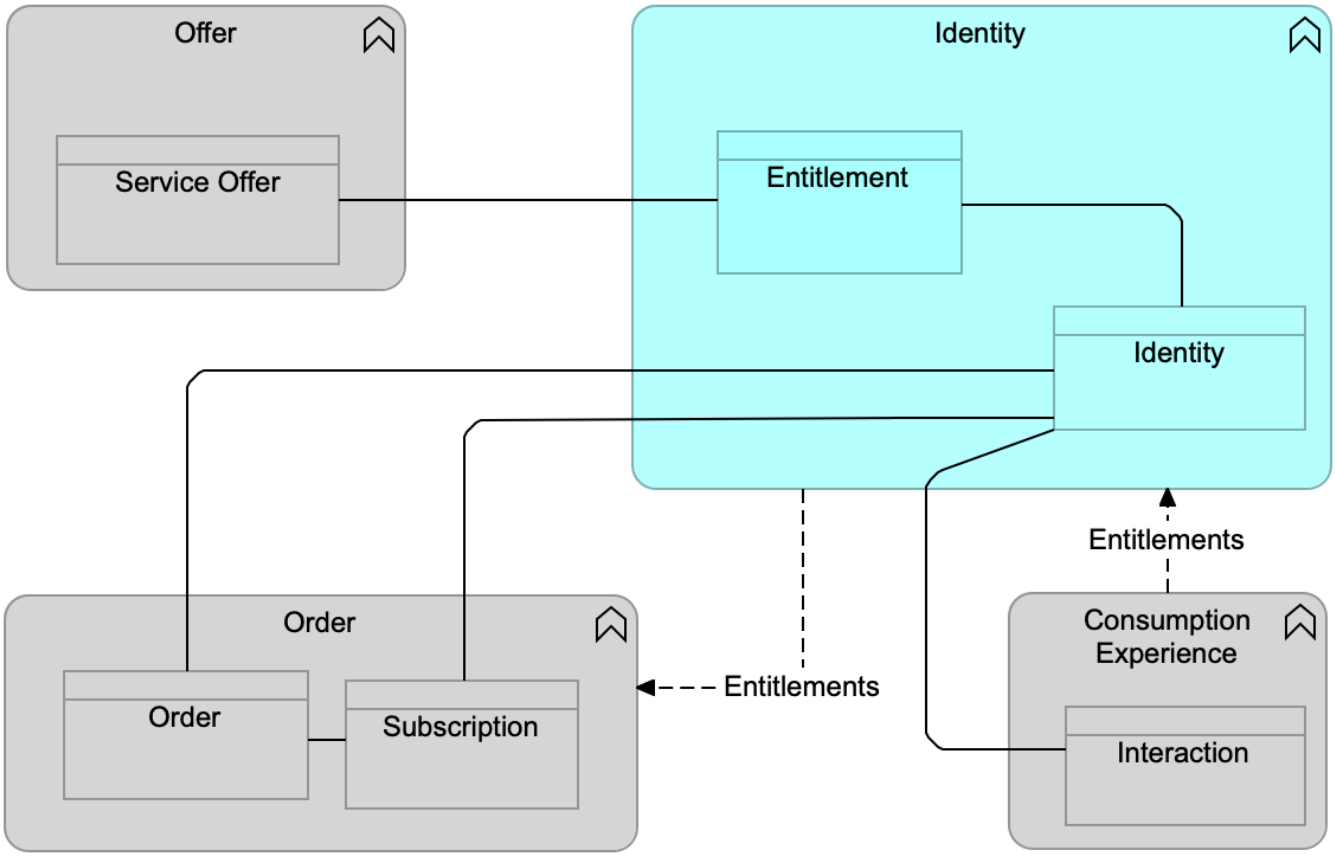

Figure 8-3. Identity Functional Component Model

Purpose

The Identity functional component controls the information about the IT4IT service consumers of the services provided by the Digital Products managed by the IT4IT system. IT4IT service consumers can either be humans or machines who are required to access the Consumption Experience functional component to initiate an RFI, a new Order, a Change, or an Incident. It provides access to the catalog view based on user Entitlement.

The Identity functional component is for managing the IT4IT service consumer identity, and tracking access rights to consume the IT4IT functionality. This facilitates the creation and administration of data used to identify a consumer, authorize, and define the attributes of IT4IT service consumers. This also facilitates the service with extensive information about a user, including address books, preferences, entitlements, and contact information. This information is subject to privacy and/or confidentiality requirements. The Identity functional component provides the required access control mechanism.

Users (Identities) get one or more Entitlements based upon their role and job profile, through which they are authorized to access specific resources. Accordingly, Orders for these services are automatically approved.

The Identity functional component supports the Consume value stream.

Functional Criteria

The Identity functional component:

- Shall maintain the confidentiality of information by controlling access
- Shall ensure that all consumers (humans or machines) and their activities in the Consumption Experience functional component are uniquely identifiable
- Shall confirm that consumer access to the catalog items data is in line with defined and documented business needs, and that Entitlements are attached to user Identities
- Shall provide the organization with the ability to meet compliance requirements and to verify adherence to regulations with respect to access to information
- Shall be able to list all consumers who have access to the Consumption Experience functional component, and identify the Entitlement level they have each been granted

8.1.2.1. Identity Data Object

Purpose

The Identity data object identifies, authenticates, and authorizes individuals (machines and users) who are required to access the Consumption Experience engagement portal.

Key Attributes

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

- **Id:** unique identifier value for the Identity
- **Name:** name of the Identity
- **Discriminator:** indicates the Identity type (i.e., user, agent, group, or role) of the Identity
- **Location:** location of an Identity assuming it is a person
- **Creation Date:** creation date of the Identity
- **Expiry Date:** expiry date of the Identity
- **Organization Id:** organization (company) to which the Identity belongs

Key Data Object Relationships

The Identity data object shall maintain the following relationships:

- Identity to Interaction (n:1): validates the Identity of the consumer and manages their access to the Consumption Experience functional component
- Identity to Order (1:n): validates access, Identity, and Entitlement for the consumer request that directly came to order

- Identity to Entitlement (1:n): validates the consumer's Entitlement to the Service Offer as per the attached Entitlement policy
- Subscription to Identity (n:1): to make an Identity an owner of one or more Subscriptions to manage the Subscription(s)

8.1.2.2. Entitlement Data Object

Purpose

The Entitlement data object is a set of attributes that grants or denies access to Service Offers, Digital Product Instances, and associated resources as per the defined Entitlements, such as Entitlement by organization, department, location, group, employee band, user profile, and country.

Key Attributes

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

- **Id:** unique identifier for every Entitlement type
- **Type:** whether the Entitlement can only be viewed or ordered

Key Data Object Relationships

The Entitlement data object shall maintain the following relationships:

- Entitlement to Service Offer (n:1): provides the Entitlement policy that needs to be attached to the Service Offer for validation
- Entitlement to Identity (n:m): indicates what Identities have the given Entitlement

8.1.3. Offer Functional Component

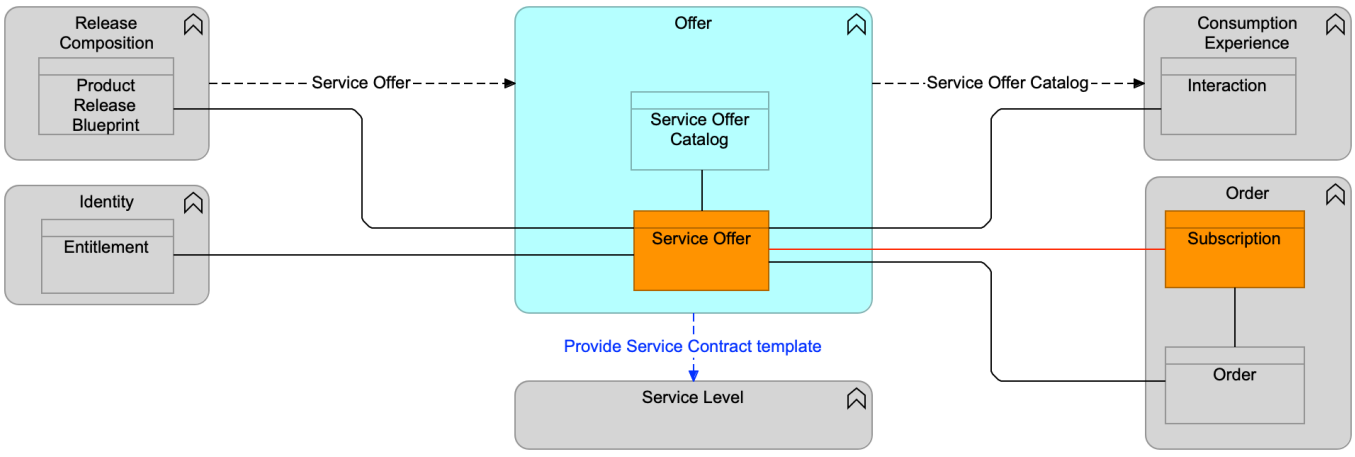

Figure 8-4. Offer Functional Component Model

Purpose

The Offer functional component aggregates all published services, delivered both internally and also from external supplier catalogs, into consumable Service Offers that the user can order through the Consumption Experience functional component. It builds and publishes the various offerings into Service Offer Catalogs for populations to consume, and to determine prices and valid options that consumers can select. It enables Service Offers to be grouped into a single Service Offer Catalog to expose them as a collection of consumable items for a given group of consumers. It ensures all required information is captured for the fulfillment (deployment/delivery) of the service from the Offer template of the Release Composition functional component. It fulfills each Service Offer through numerous underlying Offer templates as determined by this functional component. The offerings typically include the approval mechanisms needed for their fulfillment.

Based on the risk profile of the service offer, it also defines whether the fulfillment is needed to be routed through the Change Management process or directly goes for the fulfillment.

For example, Service Offers such as, for example, User ID creation, access rights, laptop requests, may lead to creating service Request Orders and such order requests shall be directed for fulfillment without the Change Management process. However, Service Offers (such as server migration, application releases) may lead to creating Change Orders that go through the Change Management process.

The Offer functional component supports the value streams:

- Release
- Consume

Functional Criteria

The Offer functional component:

- Shall contain all Offers available to consumers and provide this information to the Consumption Experience functional component
- May create the Service Contract template and provide information to the Service Level functional component
- Shall aggregate and provide a unified service catalog through the Consumption Experience functional component
- Shall drive consumption through consumer-specific views of the Service Offer Catalog and help identify prices and options for selection
- Shall enable the grouping of different Service Offers into a single consumption experience
- Shall define whether the Service Offer be fulfilled as Request Order or Change Order

8.1.3.1. Service Offer Catalog Data Object

Purpose

The Service Offer Catalog data object represents a set or collection of Service Offers that are grouped together as something that can be consumed by certain consumers or consumer groups.

Key Attributes

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

- **Id:** unique identifier for the Service Offer Catalog
- **Name:** Service Offer Catalog name used for consumers

Key Data Object Relationships

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

- Service Offer Catalog to Service Offer (n:m): represents the collection of Offers that comprise each Service Offer Catalog

8.1.3.2. Service Offer Data Object

Purpose

The Service Offer data object defines how a Service Offer template will be instantiated and under what terms and conditions; for example, price, deployment, approval, workflow, service level (contract).

Key Attributes

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

- **Id:** unique identifier for every Service Offer in the catalog
- **Catalog Id:** identify in which catalog this Service Offer is available (from the Service Offer Catalog)
- **Name:** description of the Service Offer for consumers to identify/search Offers
- **Start Date:** date/time on which the Service Offer may be consumed
- **Expiry Date:** date/time on which the Service Offer is no longer available
- **Status:** indicates if the Service Offer is ready for consumption (e.g., draft, published, retired)
- **Price:** if applicable, the pricing information on the service, including the type of Subscription
- **Required Value:** 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

Key Data Object Relationships

The Offer data object shall maintain the following relationships:

- Service Offer to Product Release Blueprint (n:1): each Service Offer is based upon the definition of one Product Release Blueprint
- Service Offer to Order (1:n): provides Service Offer details that will help to manage Orders
- Service Offer to Subscription (1:n): Service Offer contains contract-related terms and conditions (e.g., pricing, service levels) for the Subscription
- Service Offer to Entitlement (1:n): determines what a consumer (human or machine) is entitled to order
- Service Offer to Interaction (n:1): makes Service Offers available to consumers through the Consumer Experience functional component
- Service Offer to Service Offer (n:m): Service Offers can be built and defined as a collection of more fine-grained service components or service actions

8.1.4. Order Functional Component

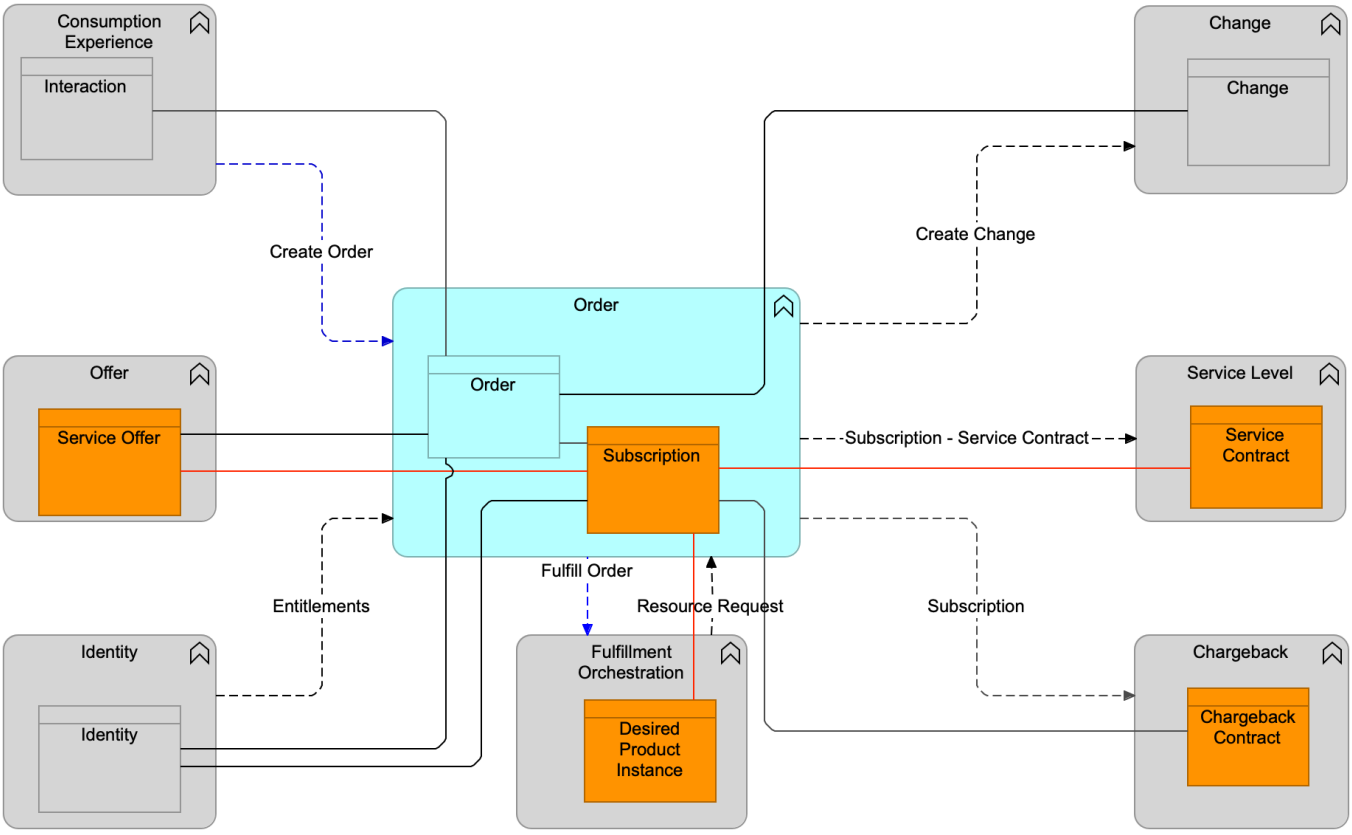

Figure 8-5. Order Functional Component Model

Purpose

The Order functional component rationalizes, breaks down, and routes “clean order” requests (ready for fulfillment) to the Fulfillment Orchestration functional component in order to deliver services to consumers. This may involve breaking down a single order/request into multiple Fulfillment Orders/requests. It ensures appropriate fulfillment-related Subscription information is kept up-to-date, such as approval/rejections, modifications, cancelations, and so on. It enables the recording of patterns of service consumption that can be used to shape demand for new and/or improved services. The fulfillment status is tracked and completion notifications from the fulfillment channel(s) are received. Consumers will be able to receive status updates at the Subscription level. The Order functional component can receive the order from three different channels:

- **Consumer-initiated** – the consumer may initiate a new Order using consumption experience communication channels such as chatbot, web application, or mobile applications
- **Machine-initiated** – a machine (such as an API) can directly trigger an Order request to the Order functional component; for instance, provisioning of a new Virtual Machine (VM), provisioning for, for example, a user Id, allocating a software license
- **Pipeline-initiated** – in this case, the Pipeline functional component will directly engage the Order functional component to order resources for the upkeep and running of the pipeline; for example, deploying a test instance infrastructure, deploying a test code

The Order functional component supports the value streams:

- Integrate
- Consume

Functional Criteria

The Order 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 confirm that consumer access to the catalog items data are in line with defined entitlements and attached to user identities
- Shall provide information on Order delivery times for SLA measurements
- Shall break down the composite Order into the individual orders/requests that need to be fulfilled
- Shall send the bound Offer template information from the Product Release package to the Change functional component in order to create the Change needed to deliver the Order fulfillment
- Shall rationalize, break down, and route “clean order” requests (ready for fulfillment) to appropriate fulfillment orchestrators or providers in order to deliver services to consumers
- Shall ensure the fulfillment-related subscription data is updated
- Shall identify service consumption trends for effective demand analysis

- Shall break the service order down into the appropriate fulfillment change request(s) and provide these to the Fulfillment Orchestration functional component via the Change functional component and create the Subscriptions for these services upon their successful fulfillment
- Shall create a traceability for fulfillments through the fulfillment channel(s)
- Should send the instances of the Service Contracts to the Service Level functional component if a Service Level functional component exists

8.1.4.1. Order Data Object

Purpose

The Order data object represents the formal request for the creation of, modifications to, or deletion of user consumption of a Service Offer.

Key Attributes

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

- **Id:** unique identifier for the Order (request)
- **Status:** controls the status of the Fulfillment functional component
- **Date:** date/time the Order was received
- **Latest Fulfill Date:** maximum date/time by which the Order needs to be fulfilled
- **Actual Fulfill Date:** date/time on which the Order is fulfilled
- **Required Value:** based on the Offer, the user might need to provide values/options for a successful fulfillment (from Offer)

Key Data Object Relationships

The Order data object shall maintain the following relationships:

- Order to Interaction (n:m): an Order can be a result of an Interaction and the relationship ensures that the status can be updated back to the requesting Identity using the same engagement
- Order to Identity (n:1): obtains information from Identity which can be used to validate and manage the approval of the Order
- Order to Subscription (n:m): enables traceability between the Order and the resulting Subscription; this helps to better understand how the current Subscription state was realized
- Order to Service Offer (n:1): obtains Service Offer details to initiate recursive Order requests (sub-orders) that may be required to fulfill an Order
- Order to Change (1:n): used for tracking the Order fulfillment through a Change Management system

8.1.4.2. Subscription Data Object

Purpose

The Subscription data object represents the rights to access a service that has been provided to a consumer.

Key Attributes

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

- **Id:** unique identifier for the Subscription
- **Status:** provides a status update to consumers on the Order status at the Subscription level; e.g., such as Work in Process/Progress (WIP), RFI, completed, closed
- **Start Date:** the date on which the Subscription was created
- **Expiry Date:** the date on which the Subscription will end

Key Data Object Relationships

The Subscription data object shall maintain the following relationships:

- Subscription to Service Offer (n:1): provides traceability between the Subscription and the Service Contract (via the Service 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 Product Instance (n:1): enables traceability between the consumer, their Subscription, and the Desired Product Instances; there can be multiple Subscriptions linked to a single Desired Product Instance
- Subscription to Identity (n:1): associates an Identity as an owner of one or more Subscriptions to manage the Subscriptions
- Subscription to Service Contract (1:1): sends the instances of Service Contracts to the Service Level functional component

8.1.5. Chargeback Functional Component

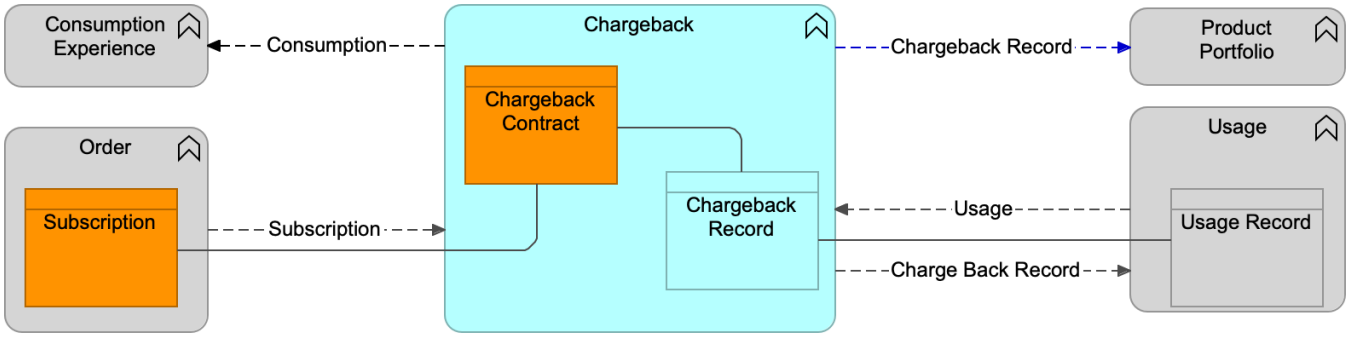

Figure 8-6. Chargeback Functional Component Model

Purpose

The Chargeback functional component provides chargeback (or showback) for internal and external services based on Subscription, Service Contract, and/or Usage information.

The Chargeback functional component supports the Consume value stream.

Functional Criteria

The Chargeback functional component:

- Shall provide cost consumption information to consumers through the Consumption Experience engagement portal
- Shall calculate the chargeback (or showback) of consuming/subscribing a service to a subscriber/consumer
- Can take actual Usage into consideration when calculating the charge of consuming a service
- Shall consolidate the charges from all subscribed services once Usage is collected for the given billing period
- Shall send the subscribed service charges to the Product Portfolio functional component for an Actual Product Instance if a Product Portfolio functional component exists
- Should send a Chargeback Record for approval and an internal reconciliation request to the Finance function if a Finance function (external to IT) exists

8.1.5.1. Chargeback Contract Data Object

Purpose

The Chargeback Contract data object details the contract for financial obligations between the service consumer and provider(s) as defined at the time of Subscription. The Chargeback Contract typically defines the billing rule and billing frequency used to price a given service and is often tightly linked to the Subscription.

Key Attributes

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

- **Id:** unique identifier for the Chargeback Contract
- **Status:** status of the contract (e.g., active, inactive)
- **Rule:** pricing rule captured at the time of Subscription describes the method/formula for translating the usage into chargeback; this might be as simple as a one-time fee for a product or as complex as an algorithm depending on the kind of service and granularity of chargeback/usage required to be captured
- **Frequency:** charging frequency to the subscriber (e.g., daily, weekly, monthly, quarterly)

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)
- Chargeback Contract to Chargeback Record (1:n): multiple billing records can be generated for a single Chargeback Contract as the Chargeback Record will be generated for each billing period

8.1.5.2. Chargeback Record Data Object

Purpose

The Chargeback Record data object represents the actual charge or showback amount to the subscriber based on the usage of subscribed services in a given time period. It is computed by the Chargeback functional component using the rules stored in the associated Chargeback Contract(s), with the input being the Usage Records collected for the period.

Key Attributes

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

- **Id:** unique identifier of the related Chargeback Record
- **Status:** status of the Chargeback Record (e.g., initiated, recorded)
- **From Date:** start date of the Chargeback Record
- **To Date:** end date of the Chargeback Record
- **Amount:** amount to be charged for the given period

Key Data Object Relationships

The Chargeback Record data object shall maintain the following relationships:

- Chargeback Record to Chargeback Contract (1:n): indicates to which contract (and thereby to which Subscription) the Chargeback Record relates
- Chargeback Record to Usage Record (1:n): each Chargeback Record is calculated based on all the Usage Records with which it is associated

8.2. Fulfill Function

The Fulfill function is centered around efficiently fulfilling requests.

8.2.1. Change Functional Component

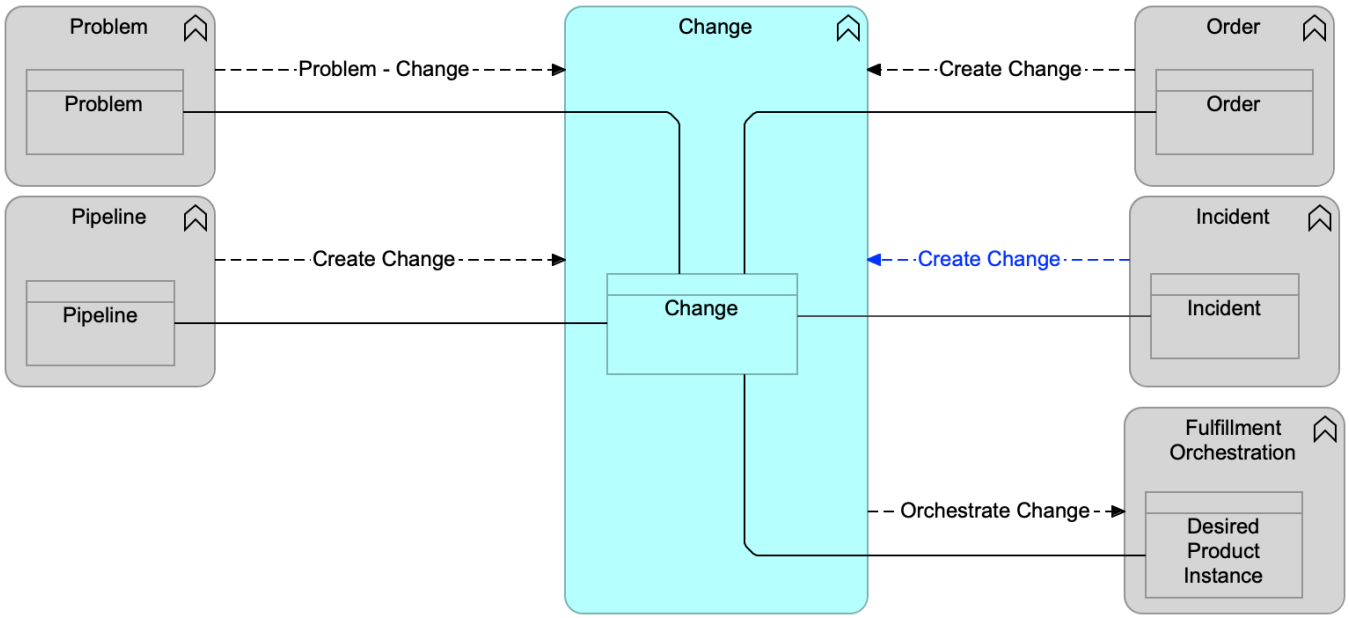

Figure 8-7. Change Functional Component Model

Purpose

The Change functional component is responsible for controlling the lifecycle of Changes in the technology environment to make sure that Changes are captured, analyzed, assessed, and implemented in a standardized and auditable way so that the business risk is minimized through the following means:

- Facilitate communication with stakeholders
- Assess the risk of proposed Changes (Change Orders) and their implementation; support the impact and risk assessments to minimize the risk of production disruptions involved when rolling out Changes

- Enable notification of all affected Change stakeholders and facilitate their collaboration on Change execution
- The Change authority reviews the Change Orders and authorizes the valid Changes
- The Change type, standard or normal, is determined based on the Change risk profile
- Support the automation of Changes so that human participation is reserved for the highest value-added and most complex Change activity

  For example, the Event functional component or Incident functional component may use a manual or automated Runbook to resolve well-understood issues without an active Change. These classes of pre-approved Changes may vary by company and by the criticality of service. For these pre-approved changes, it is assumed that the Change is recorded and that the Change Management process has access to that information. A typical example would be a Runbook automation script that fixes the issue. The relationship to the Actual Product Instance is maintained to allow Configuration Management to have access to Change information.

- Changes initiated by the Pipeline functional component (for CI/CD) making it completely automated, with a reduced change size (iterative) and to an acceptable risk level; therefore, only log the Change details in the Change record
- Change Management may automate the creation of the Change record, the risk assessment, and its approval process using data that is collected via integration to existing CI/CD pipelines to expedite the overall product deployment process while maintaining compliance
- Change functional component integration with the Fulfillment Orchestration functional component to orchestrate the deployments

  If a Change is low risk, it will be automatically approved, and deployed instantly. If a Change is high risk, the deployment will be delayed until the Change record is manually approved or stopped completely if a Change record is rejected.

- Certain types of changes (e.g., end-user request items) are initiated from the Order functional component and may directly trigger the Fulfillment Orchestration functional component for the fulfillment of the order
- Enable Change Management against a Change calendar to avoid Change collisions
- Check and steer conflict resolutions between parallel planned deployments

The Change functional component supports the value streams:

- Deploy
- Consume
- Operate

Functional Criteria

The Change functional component:

- Shall act as an authoritative system of record for all change requests
- Shall manage the state and lifecycle of the Change
- Shall facilitate communication with stakeholders
- Shall assess the risk of proposed Changes and their implementation
- Shall support the impact and risk assessments to minimize the risk of production disruptions involved when rolling out Changes
- Shall identify affected Change stakeholders and send them notifications throughout the Change lifecycle
- May support the automation of Change execution as much as possible
- Shall maintain the relationship of the Change to the actual service to allow Configuration Management to have access to Change information
- Shall facilitate with a Change calendar to avoid Change collisions
- Shall manage conflicting resolutions for collateral deployments
- Can receive a Change directly from the Pipeline functional component to fulfill the order initiated by the Pipeline

  These Changes, initiated by the Pipeline functional component (for CI/CD), are often completely automated, with a decreased Change size (iterative) and to an acceptable risk level

- Can provide Change data to the Event and/or Monitoring functional components to facilitate a root-cause analysis process in the context of the impact Changes may have
- Shall associate a Fulfillment Order request with a Change record
- Shall associate Change(s) to Desired Product Instance(s)
- Shall classify the Change type (standard, emergency, or normal) based on the Change risk profile
- Shall associate Changes with Incidents in response to Incidents that are resolved by implementing an emergency change
- Shall associate Changes to the Problem in order to implement a fix to the issue that is documented by the Problem if a Problem functional component exists

8.2.1.1. Change Data Object

Purpose

The Change data object includes details of a proposed Change to one or more CIs.

Key Attributes

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

- **Id:** unique identifier for the Change record
- **Title:** title of the Change
- **Description:** description of the Change
- **Category:** classify and define the type (e.g., emergency, standard, normal) of Change requested; each category has its own lifecycle
- **Approval Status:** current status of the Change approval
- **Risk:** the probability that the Change could cause harm or loss, or affect the ability to achieve objectives
- **Planned Start Time:** date/time when the Change implementation is planned to start
- **Planned End Time:** date/time that Change implementation is planned to end
- **Assigned To:** actor that is assigned to implement the Change

Key Data Object Relationships

The Change data object shall maintain the following relationships:

- Change to Order (1:n): receives Change request and Change request status
- Change to Desired Product Instance (1:n): acquires relevant information about the Desired System(s)
- Change to Pipeline (n:1): the Pipeline supplies the Change Orders with information needed to instantiate the service
- Change to Change (n:m): a Change can depend on other Changes that have been delivered, or it can be decomposed into a number of more fine-grained Changes
- Incident to Change (n:1): tracks the Incident(s) caused by the Change
- Change to Incident(1:n): tracks the Incidents that needs emergency Changes to fix the issues
- Problem to Change(1:n): to implement a fix to the issue that is documented by the Problem

Change data between the Order, Desired System, and the Change must manage the Change lifecycle state.

8.2.2. Fulfillment Orchestration Functional Component

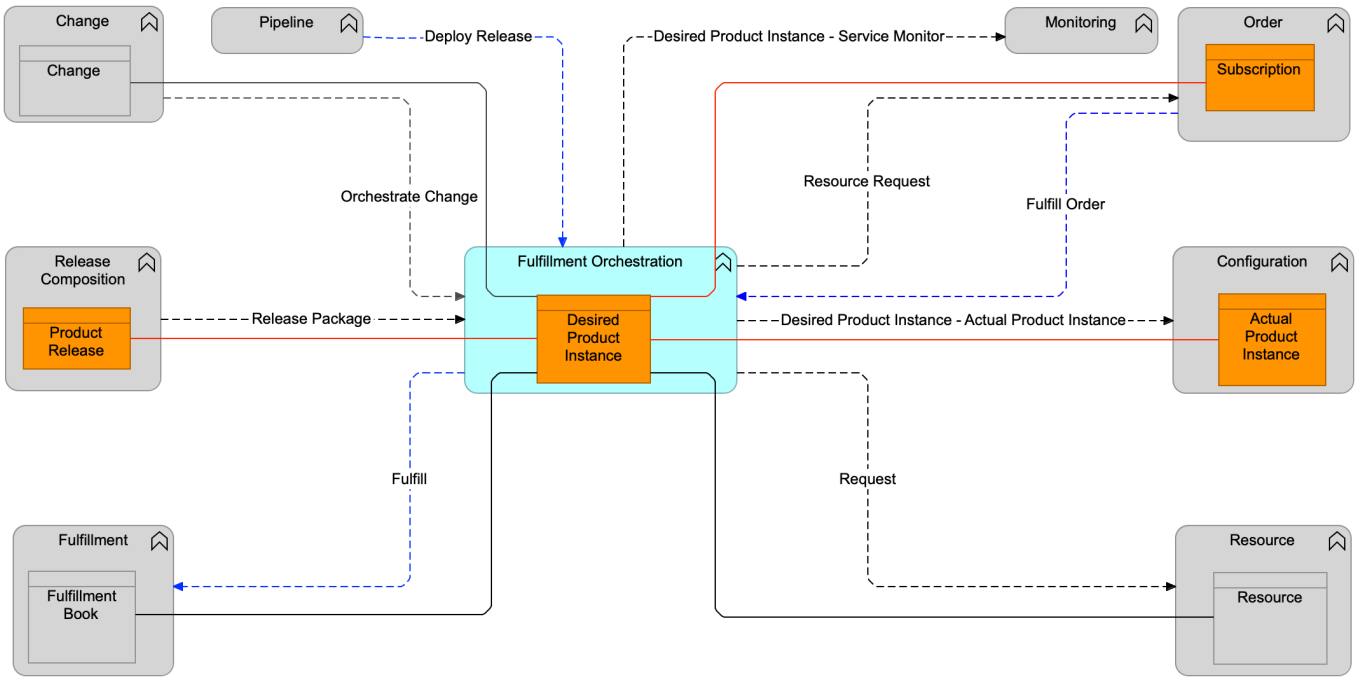

Figure 8-8. Fulfillment Orchestration Functional Component Model

Purpose

The Fulfillment Orchestration functional component orchestrates the delivery of various Orders across one or more Fulfillment functional components in order to fulfill the service orders (such as the provision of VMs, modification of resource allocation, addition/removal of capacity, patches installation, modification of access rights) that are triggered by:

- Release Composition to deploy new Product Releases received from the Integrate value stream
- Order (e.g., end-user request from the portal, application access request, password reset)
- Change (e.g., provision of the test environment, deployment of bug-fix, deployment of the patch)
- Pipeline (for continuous deployment)

The Fulfillment Orchestration may orchestrate through multiple internal and/or external Fulfillment functional components to automatically execute a fulfillment process workflow. To be able to engage the fulfillers, the Fulfillment Orchestration 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 released Offer template deployment recipe from Release Composition and update the change request (for fulfillment) and also the Desired Product Instance data object that represents the service model in its preconfigured state
- Get the resources allocated that are required to fulfill the service order

- Update the resource pool inventory as resources are provisioned
- Based on the new or updated Desired Product Instance and the result of fulfilling the Change (Change data object), inform the Configuration functional component (if needed)

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

- Consumer-driven

  In this paradigm, a consumer request results in a bound catalog item that is broken down into the necessary Change Orders needed to fulfill the originating request; typically, this is for delivering the consumption of service from an existing product/service system that has previously been deployed.

- Pipeline-driven

  In this paradigm, the Fulfillment Orchestration functional component is directly engaged by the Pipeline functional component; this paradigm is used when the Development team is ready to transition a product into the production environment and wants to deploy a service system for the product.

The Fulfillment Orchestration functional component supports the value streams:

- Integrate
- Deploy
- Consume

Functional Criteria

The Fulfillment Orchestration functional component:

- Shall orchestrate the implementation of the various Change Orders across the fulfillment systems, or track and manage manual change task implementation to fulfill the Orders
- Update the resource inventory as resources are ordered
- May manage a registry of the available fulfillers to include what each fulfiller does (capabilities) and how to engage each fulfiller (where they are located and how to invoke them)
- May take the bound Offer template from the Change functional component and generate both the relevant Fulfillment Orders to realize/fulfill the originating consumer request and the Desired Product Instance data object, which represents the system in its preconfigured state
- Shall get the technology resources (e.g., license, IP, VMs) allocated to fulfill the service order from the Resource functional component
- Shall select the appropriate fulfillment mechanism
- Shall coordinate if multiple fulfillment mechanisms are needed, and manage the dependencies required to fulfill the digital service orders

- Shall create a Desired Product Instance based on a template in the Product Release package if the Order is a request for a new product system
- Should modify the associated Desired Product Instance for all consumer Subscriptions to the service in order to track service delivery
- Shall provide the Subscription status to the Order functional component
- Shall create the Actual Product Instance as a copy of the Desired Product Instance within the Configuration functional component
- May update the resource pool inventory as resources are provisioned
- May trigger the Order functional component to order the appropriate dependent services, to fulfill a service order
- Can create a new Service Monitor or modify an existing one for the service provided in the Order as part of fulfillment
- Can create/route an Order to an external service provider to fulfill a part or all of the service
- Can create an Order if a given Digital Product is to be delivered as a service from another product system (enabling product)
- Can trigger fulfillment automation systems to enable fulfillment of (parts of) the service

8.2.2.1. Desired Product Instance Data Object

Purpose

The Desired Product Instance data object specifies the deployment of a product so that it meets the deployment requirements specified in the Order or Change records. It contains the relevant fulfillment parameters that determine how to instantiate the product.

Key Attributes

The Desired Product Instance data object shall have the following key data attributes:

- **Id:** unique identifier of the Desired Product Instance
- **Name:** name of the Desired Product Instance
- **Type:** type of the Desired Product Instance
- **Configuration Items:** model of the configuration of the Product Instance as a set of interconnected CIs
- **Create Time:** date/time the Desired Product Instance was created
- **Last Modified Time:** date/time the entry was last modified

Key Data Object Relationships

The Desired Product Instance data object shall maintain the following relationships:

- Desired Product Instance to Subscription (1:n): creates the traceability from service to Subscription
- Desired Product Instance to Product Release (n:1): acquires all the necessary service information for fulfillment
- Desired Product Instance to Actual Product Instance (1:1): creates traceability and enables verification of correct deployment/fulfillment
- Desired Product Instance to Fulfillment Book (m:n): creates Fulfillment Book(s) for each Desired Product Instance(s)
- Desired Product Instance to Resource (1:n): requests for resources needed for Order fulfillment demand
- Desired Product Instance to Change (1:n): obtains orchestration details for Order fulfillment
- Desired Product Instance to Desired Product Instance (n:m): the Desired Product can depend on services being delivered by other products

8.2.3. Resource Functional Component

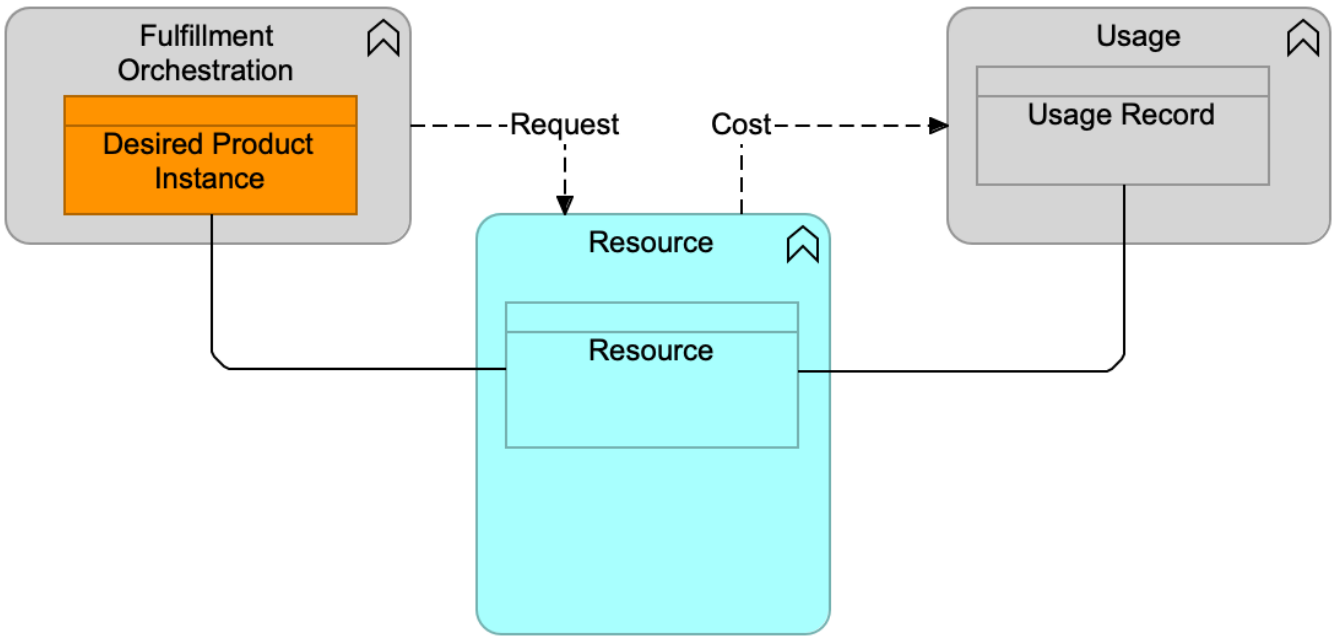

Figure 8-9. Resource Functional Component Model

Purpose

The Resource functional component manages the pool of resources (a logical abstraction of digital assets) that support Desired Product Instances delivering digital services, such as infrastructure, applications, and third-party services. This also ensures that the resource requirements for the services and their underlying resources are well understood and provisioned efficiently.

The Resource functional component ensures the pooling of resources that may be virtual or physical (e.g., server, storage, networking, software licenses, IP address) are made available for the Order fulfillment. It maintains the soft limits or hard limits of resource pooling for digital resources(e.g., cloud resources) and acts appropriately when there is a breach of the limits. It also provides resource utilization details for Usage functional components to compute costs.

The Resource functional component supports the value streams:

- Deploy
- Consume

Functional Criteria

The Resource functional component:

- Shall manage the pool of Resources that support services, such as infrastructure, applications, and third-party services
- Shall ensure that the digital Resource requirements against the services are ascertained
- Shall ensure the availability of physical assets/virtual Resource pools for the Order fulfillment
- Shall maintain, for example, the soft limits, hard limits, quotas of Resource pools for digital resources (e.g., cloud resources), and act appropriately when there is a breach of the limits
- May provide capacity Resource utilization details to the Usage functional component for cost computation

8.2.3.1. Resource Data Object

Purpose

The Resource data object represents an asset or other entity that is limited in nature. It can be allocated to a Digital Product or it can be unallocated.

Key Attributes

- **Id:** unique identifier for each Resource
- **Type:** describes the type of each Resource pool
- **Status:** record if the Resource is available

Key Data Object Relationships

The Resource data object shall maintain the following relationships:

- Resource to Desired Product Instance (n:1): indicates if a Resource is allocated to a given Product Instance
- Resource to Usage Record (1:n): provides Resource usage details for cost computation

8.2.4. Fulfillment Functional Component

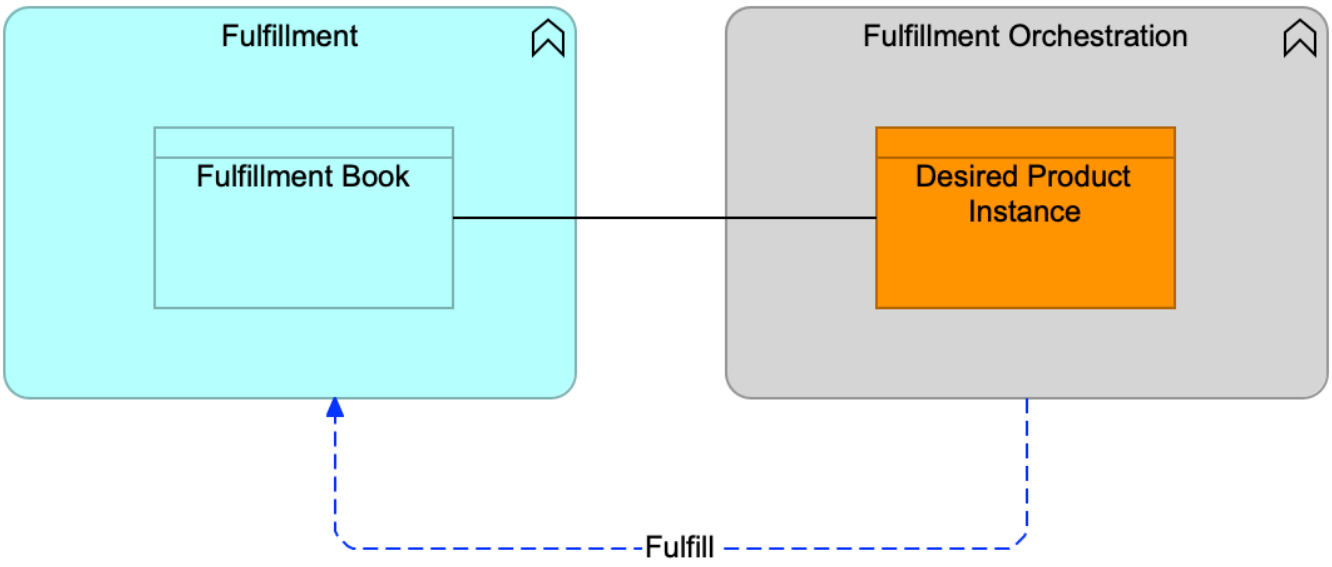

Figure 8-10. Fulfillment Functional Component Model

Purpose

The Fulfillment functional component ensures that fulfillment activities are documented and automated for the resources that support the Digital Product Instance. Such activity items may include provisioning, deploying, modifying, actions (i.e., start, stop), decommissioning, and so on. This manages the routine deployment and configuration activities into automated/semi-automated Fulfillment Book. The automated/semi-automated fulfillment scripts are often included in the release package. The Fulfillment functional component may receive these deployment scripts from Release Composition functional component for the product deployment.

The Fulfillment functional component supports the value streams:

- Integrate
- Deploy
- Release
- Consume

Functional Criteria

The Fulfillment functional component:

- Shall ensure the fulfillment activities are standardized, automated, or, if manually executed, well-documented for the resources that support the Digital Product Instance
- Shall ensure that the automated Runbooks are updated with routine deployment and configuration activities including but not limited to building and deploying resources, configuring VMs, decommissioning VMs, configuring platforms

- Shall monitor that the Fulfillment Books are being executed as per the designed specification
- Can allow the Fulfillment Orchestration functional component to trigger automated Fulfillment Books for deploying or configuring resources underpinning a digital service in order to fulfill a service order

8.2.4.1. Fulfillment Book Data Object

Purpose

The Fulfillment Book data object is a Runbook for the fulfillment of individual Resources as part of the Fulfillment Orchestration functional component. The plan can either be an automated template or/and manual process description.

Key Attributes

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

- **Id:** unique identifier for the Fulfillment Book record
- **Category:** helps in associating the Fulfillment Book to the Desired Product Instance
- **Description:** description of the Fulfillment Book
- **Execution Time:** date/time when the Fulfillment Book was created

Key Data Object Relationships

The Fulfillment Book data object shall maintain the following relationship:

- Desired Product Instance to Fulfillment Book (m:n): every Desired Product Instance will have one or more fulfillment plans

8.2.5. Usage Functional Component

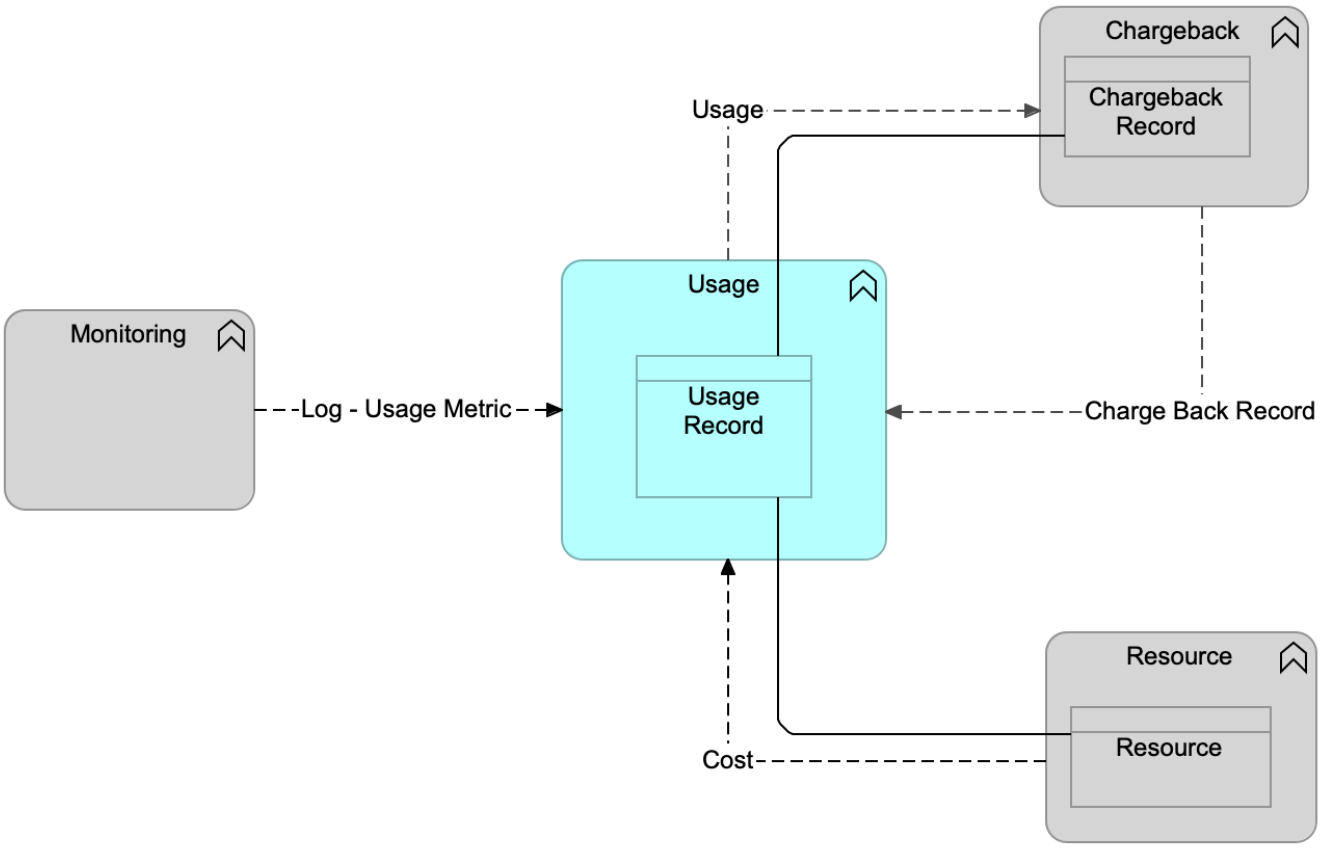

Figure 8-11. Usage Functional Component Model

Purpose

The Usage functional component tracks and manages the actual usage of subscribed digital services and their associated costs.

The Usage functional component supports the Consume value stream.

Functional Criteria

The Usage functional component:

- May track actual usage of subscribed digital services by gathering service usage metrics, activity, and history for both internal and external sourced services associated with an aspect of the Desired Product Instance
- Usage Records can have any unit:
  - Including Central Processing Unit (CPU) usage, storage consumption, transaction numbers
  - It can also be a cost number from the Resource functional component
  - It can be the price a sub-supplier is reporting – a price is considered as the cost of running that component

- May process and break down usage information, for example, for each Subscription, its consumers (singular, group), provider
- May collect service usage metrics from the Monitoring functional component
- Shall encrypt sensitive usage information or set appropriate access controls
- Can generate service usage history and activity reports
- May provide usage information to the Chargeback functional component, enabling usage-based chargeback (or showback)
- Shall collect costs associated with sub-services if a service is further decomposed; this will be the cost reported as Chargeback Records on the sub-services and will be reported as usage back up to the next level in the service composition
- Shall collect usage information from vendor invoices that represent Resources used by the service
- Shall collect cost of capacity partaking in delivery of the service

8.2.5.1. Usage Record Data Object

Purpose

The Usage Record data object represents a measurement of consumption of a particular service or service component. An example Usage Record can be composed of (internal) hours, system usage (e.g., capacity, CPUs), or external supplier usage.

Key Attributes

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

- **Id:** unique identifier for the service usage
- **Usage Date From:** date from which the service usage is captured (linked to billing frequency in the Chargeback Contract)
- **Usage Date To:** date up to which the service usage is captured (linked to billing frequency in the Chargeback Contract)
- **Units:** transaction or consumption units
- **Unit Type:** type of units used (e.g., CPU seconds, disk space, web transactions)

Key Data Object Relationships

The Usage Record data object shall maintain the following relationships:

- Usage Record to Chargeback Contract (n:1): every Usage Record for a Subscription is associated with a Chargeback Contract; the Chargeback Contract defines the billing rule and frequency for a Subscription
- Usage Record to Resource (n:1): if a Resource is measuring the consumption of a dedicated Resource for the Actual Product Instance

NOTE In case of fixed-price Chargeback, no Usage Record may exist for a given Chargeback Record.

Chapter 9. Detect to Correct Functions

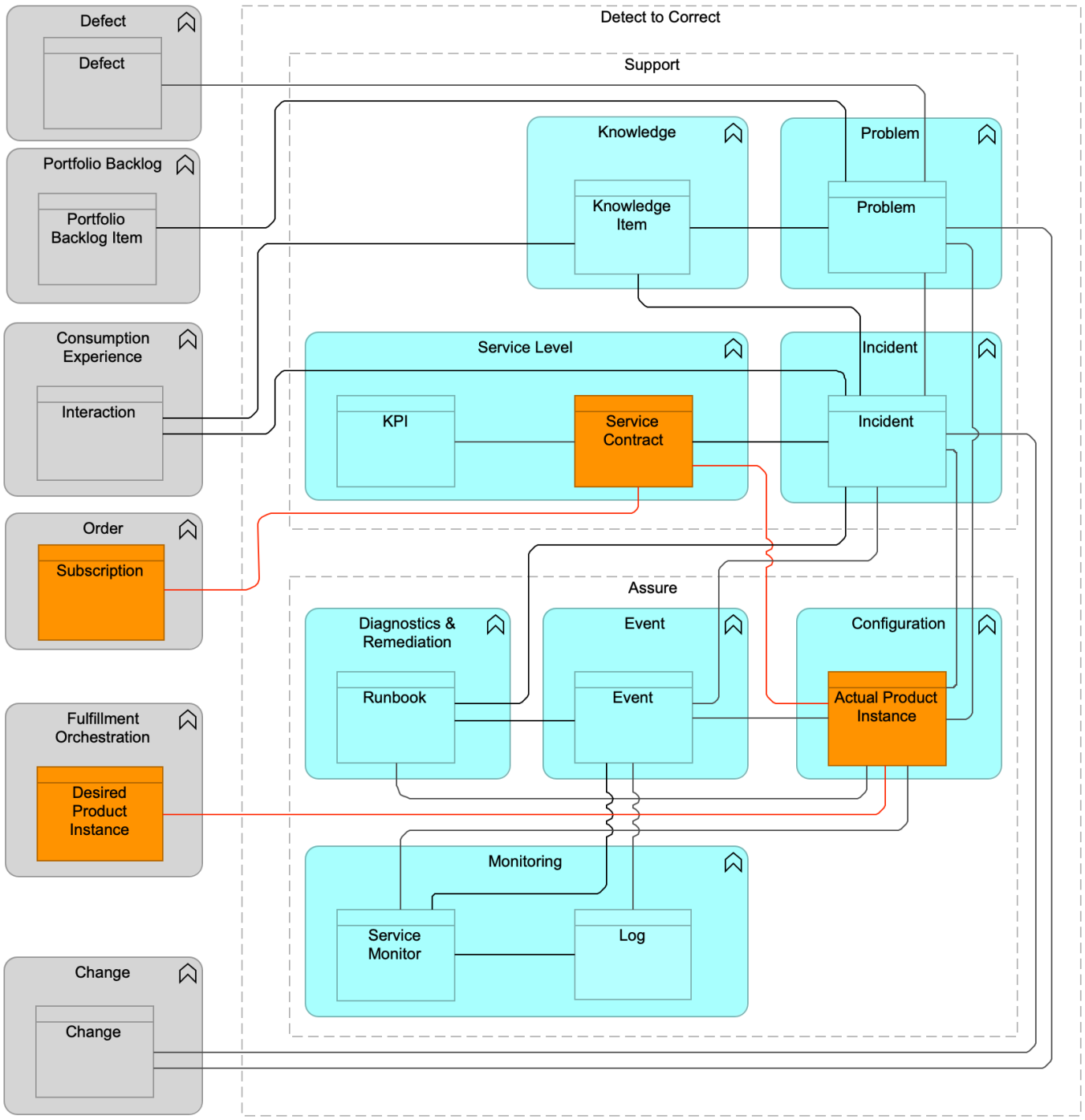

Figure 9-1. Detect to Correct Functions Model

Description

The Detect to Correct functions, informally referred to as “Run”, provide a framework for the operation of Digital Products, supporting the running services and systems while assuring that all running services are operating within stated boundaries and in a secure manner. The Detect to Correct functions also provide a comprehensive overview of the business of digital operations and the services delivered by an Operations team, including security operations. This viewpoint provides an

understanding of the inter-relationships among its many domains, and responsiveness to business requests and requirements.

The Detect to Correct functions bring operations functions together to enhance services and efficiencies – thus reducing risk.

The Detect to Correct functions contain the following functional components:

- Support functions:
  - Incident component
  - Problem component
  - Service Level component
  - Knowledge component
- Assure functions:
  - Monitoring component
  - Event component
  - Configuration component
  - Diagnostics & Remediation component

The Detect to Correct functions accommodate the technical inter-relationships and inter-dependencies required to fix operational issues and improve the ability to support business objectives by providing agility, increased uptime, and lower per-service cost.

Related Value Streams

The following value streams use one or more functional components from the Detect to Correct functions:

- Evaluate
- Consume
- Deploy

Business Benefits

The Detect to Correct functions enable 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.

The main benefits of using the Detect to Correct functions are:

- Increase efficiency and reduce cost by:
  - Focusing responses based on causal factor, priority, and business impact

  - Increasing the sharing of information and the reduction of multiple entries of the same data
  - Creating a prescriptive data flow between Event, Incident, Problem, and Change
  - Centralized Event Management for faster analysis
  - Automation between and across business functions
  - Knowledge management and self-service linkage
  - Driving Service Monitoring configuration and predefined Knowledge linked to the Deliver functions
  - Improving the speed at which issues with an Actual Product Instance are identified
  - Driving operating/service level targets
  - Improving the speed at which issues with an Actual Product Instance are proactively identified before the service impact is severe
- Reduce risk by:
  - Sharing consistent data and configuration information between operational silos
  - Prescriptive data flow and data objects
  - Defining business impact
  - Reducing the need for best-guess routing and clannish knowledge
  - Implementing network security to minimize intrusions that cause DoS, viruses, and theft or corruption of data, and to minimize risk exposure
  - Identifying attack signatures that can disrupt operations and affect compliance
  - Clearly defined ownership
  - Increased uptime by reduced Mean Time To Repair (MTTR)
  - Creating a consistent way of managing service level (SLM) definitions, measurements, KPI calculations, and reporting back to the proper Product Manager or Consumer
  - Performing Threat and Vulnerability Assessments (TVAs)
  - Providing an audit trail
- Continuous service improvement:
  - Defined data objects to be shared with Problem Management
  - Using this accumulated Knowledge as input into the Strategy to Portfolio functions
  - Improved management information and decision-making

The Detect to Correct functions provide the ability to efficiently manage operations by monitoring key services, correlating and appropriately escalating Events, sharing knowledge, managing (resolving) Incidents and Problems, tracking the Actual Product Instance and its interdependencies, and doing all of that in an automated way. It ensures that functional components used by groups can work together efficiently, through well-defined control points and data objects, to govern and run operations.

9.1. Support Function

The Support function centers around Incident, Problem, and Knowledge Management, as well as tracking service objectives.

9.1.1. Service Level Functional Component

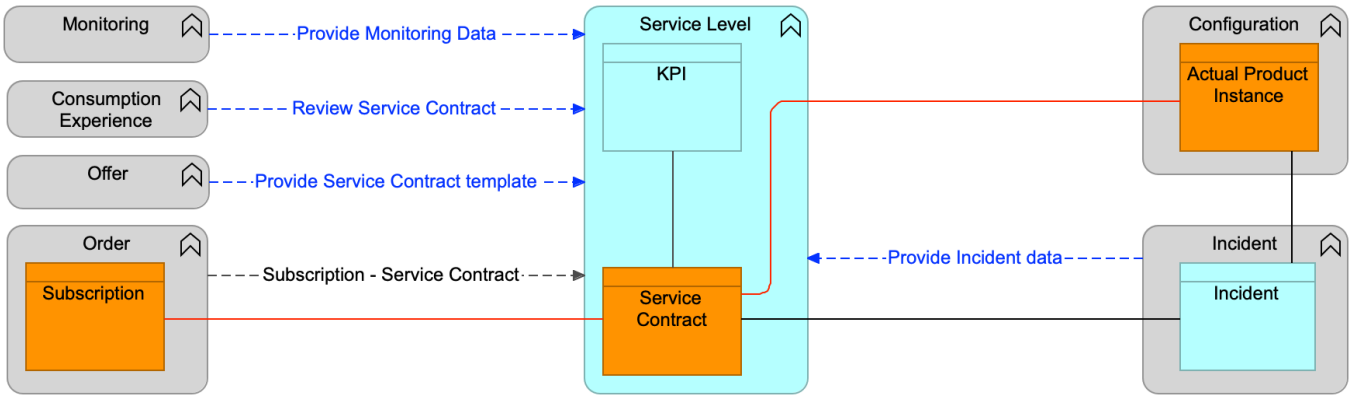

Figure 9-2. Service Level Functional Component Model

Purpose

The Service Level functional component enables the design and creation of Service Contracts. 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. It is responsible for collecting the relevant information in order to monitor compliance with the terms specified in the Service Contract and exposing data that reflects actual performance against the defined SLAs, SLOs, Operational-Level Agreements (OLAs), and/or XLAs.

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 of the Evaluate, Explore, and Integrate value streams as the main input for the demand and requirements definition stages.

The Service Level functional component supports the value streams:

- Deploy
- Release
- Consume
- Operate

Functional Criteria

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 manage the relations between the Service Contract and the KPI throughout their lifecycles
- Shall create reports on the Service Contracts to show the QoS per SLO
- Shall create a Service Contract (instance) and start measuring it once a Subscription is instantiated if the Order functional component exists
- May receive business/IT measurements from the Monitoring functional component if a Monitoring functional component exists
- May 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
- May instantiate a Service Contract from a Service Contract (template) originating from the Offer functional component
- May receive Incident business measurements from the Incident component if an Incident functional component exists
- May send reporting data on the service level status to the Consumption Experience functional component

9.1.1.1. Service Contract Data Object

Purpose

The 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 the nesting of 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 components, because there are multiple providers involved, or to cover different areas of service levels.

Key Attributes

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

- **Id:** 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
- **Start Date:** start date/time of the Service Contract
- **End Date:** end date/time of the Service Contract
- **Support Calendar:** contracted support hours of the service
- **Adherence Calculation Periodicity:** service measurement calculation period
- **Maintenance Window:** service maintenance timeframes/blackout periods

Key Data Object Relationships

The Service Contract data object shall maintain the following relationships:

- Incident to Service Contract (n:m): the service level which applies to an Incident is dependent on the related Service Contract(s)
- Actual Product Instance to Service Contract (1:n): relationships to Actual Product Instance(s) are maintained and updated to ensure 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

9.1.1.2. KPI Data Object

Purpose

The KPI data object is 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 KPI data object shall have the following key data attributes:

- **Id:** unique identifier of a KPI
- **Name:** name of the KPI

- **Timestamp:** time the KPI was measured
- **Metric:** the definition of the data collected as input for the metric calculation
- **Algorithm:** the exact algorithm for calculating the KPI

Key Data Object Relationships

The KPI data object shall maintain the following relationships:

- KPI to Service Contract (1:n): associates the measure with a particular contract

9.1.2. Incident Functional Component

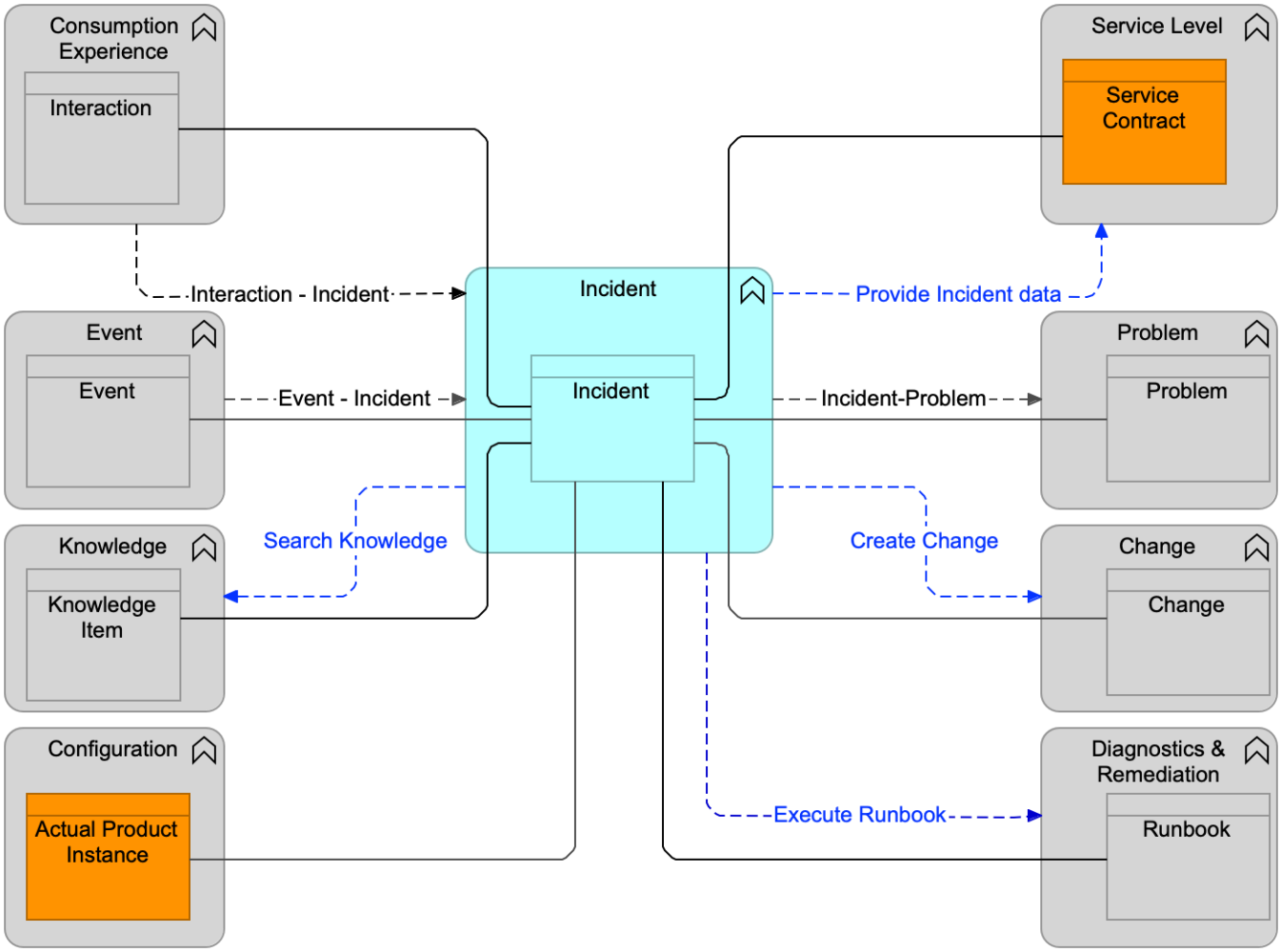

Figure 9-3. Incident Functional Component Model

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 Monitoring functional component, filter end-user Interactions and determine which ones should be associated with Incidents
- Detect Incidents, investigate the impacts across all domains (e.g., server, network, security), 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 a service or a reduction in the quality of a service as defined within the Service Contract related to the Actual Product Instance and underlying systems. 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.

The Incident functional component supports the value streams:

- Consume
- Operate

Functional Criteria

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 a Problem record when the Incident is severe, requires further deep investigation, or is repeating
- May trigger the execution of a Runbook (either automated or manual) to provide diagnostics information or remediation steps
- May trigger the creation of an emergency Change in order to implement a fix to the Incident
- May provide business measurements of Incident data to the Service Level functional component
- May receive knowledge from the Knowledge functional component to help diagnose or resolve an Incident

9.1.2.1. Incident Data Object

Purpose

The Incident data object hosts and manages Incident data and the lifecycle of the Incident.

Key Attributes

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

- **Id:** unique identifier of the Incident
- **Title:** title of the Incident
- **Category:** aids in determining assignment and prioritization
- **Subcategory:** second level of categorization, following the Category attribute
- **Status:** current stage in the lifecycle of an Incident
- **Status Time:** time stamp for the Status attribute
- **Outage Start Time:** time stamp for the start of service downtime
- **Outage End Time:** time stamp for the end of service downtime
- **Severity:** severity of the Incident
- **Priority:** priority of fixing the Incident
- **Description:** description of the Incident
- **Assigned To:** group or person that is assigned to fix the Incident

Key Data Object Relationships

The Incident data object shall maintain the following relationships:

- Interaction to Incident (1: n): provides the interface for consumers to submit their own Incidents or break-fix requests
- Incident to Problem (n:m): connection between Incidents that are converted to Problems to permanently address severe/repeating Incidents
- Incident to Knowledge Item (n:m): connection between Incidents and the Knowledge Items used for their resolution
- Incident to Runbook (n:m): an Incident is related to one or more Runbooks used to resolve the Incident
- Incident to Change (1:n): connecting emergency Changes to Incidents for remediation
- Incident to Actual Product Instance (n:m): Actual Product Instance to which the Incident is associated, and of which it is usually the main subject
- Incident to Service Contract (n:m): the service level which applies to an Incident is dependent on the related Service Contract(s)
- Event to Incident (n:m): enables the connection between Incidents and Events and supports the integration lifecycle between them
- Incident to Incident (n:m): several reported Incidents can be related to the same issue and will then be linked

9.1.3. Problem Functional Component

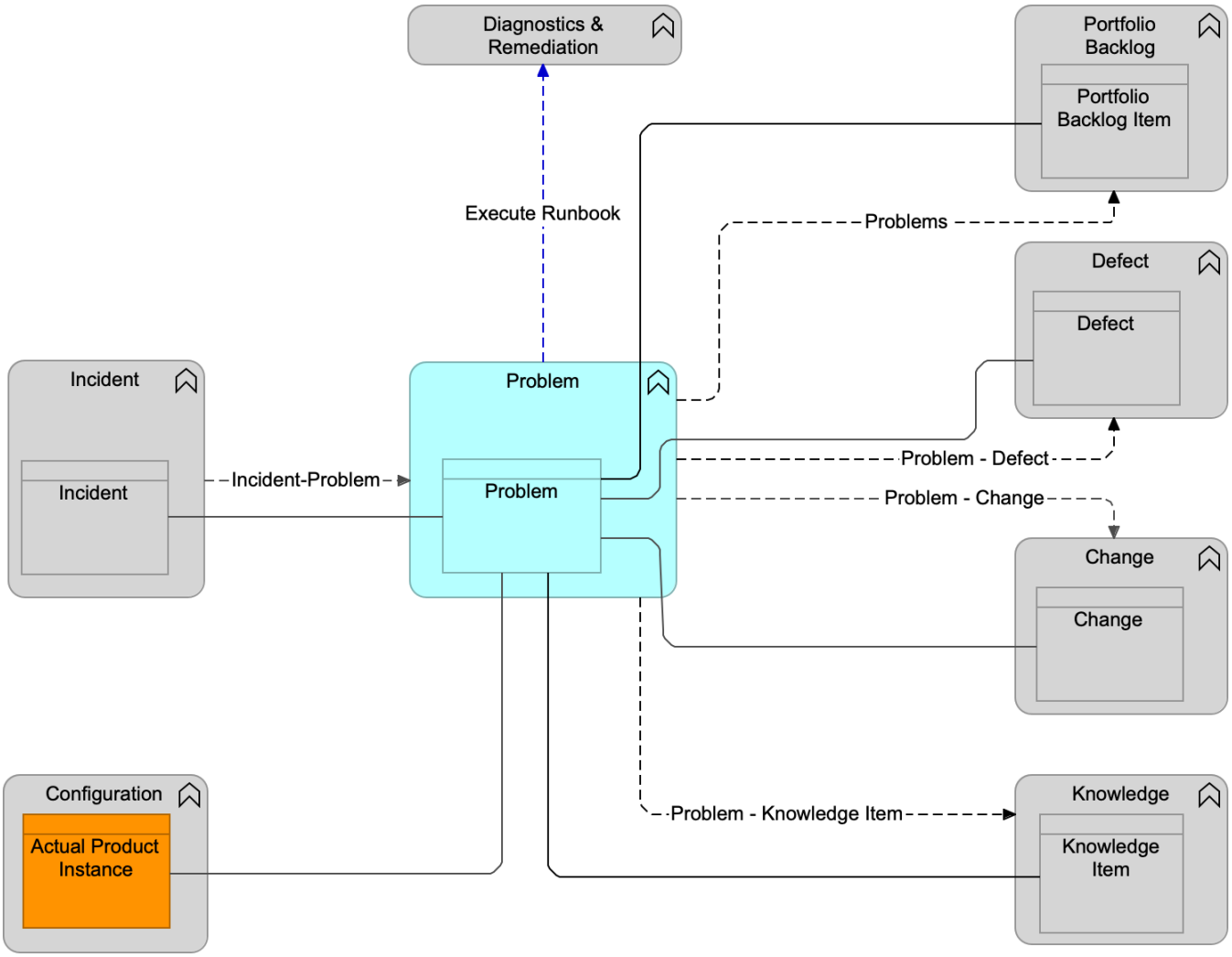

Figure 9-4. Problem Functional Component Model

Purpose

The Problem functional component is responsible for managing the lifecycle of all Problems. The objectives are to solve severe/repeating Incidents, prevent Incidents from happening, and minimize the impact of Incidents that cannot be prevented. The cause of the Problem is not usually known at the time of the Problem data object instance creation, and the Problem functional component is responsible for investigating. It also serves as the main exit point from the Operate value stream for the feedback information to the Evaluate, Explore, and Integrate value streams in the form of Portfolio Backlog Items.

The Problem functional component supports the value streams:

- Integrate
- Deploy
- Operate

Functional Criteria

The Problem functional component:

- Shall be the system of record for all Problem records
- Shall manage the state and lifecycle of the Problem
- Shall receive Incident information to create a Problem from the Incident functional component when additional diagnostics and root cause need to be determined
- Shall send Change information to the Change functional component associated to a Problem in order to implement a fix to the issue that is documented
- Shall push Problem data requiring emergency/specific development to the Defect functional component
- May send Problems to the Portfolio Backlog functional component to initiate corrective actions
- May send known error (Knowledge Item) information to the Knowledge functional component
- May push Problem data to the Diagnostics & Remediation functional component to trigger the execution of a Runbook data object (either automated or manual) to provide diagnostics information or remediation steps

9.1.3.1. Problem Data Object

Purpose

The Problem data object defines a Problem and manages the Problem lifecycle.

Key Attributes

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

- **Id:** unique identifier of the Problem
- **Title:** title of the Problem
- **Category:** aids in determining assignment and prioritization
- **Subcategory:** second level of categorization, following the Category attribute
- **Description:** description of the Problem
- **Status:** current stage in the lifecycle of a Problem
- **Status Time:** time stamp for the Status attribute
- **Assigned To:** group or person that is assigned to fix the Problem

Key Data Object Relationships

The Problem data object shall maintain the following relationships:

- Problem to Change (1:n): enables the relation of a Change record that is created when Problem resolution requires a Change
- Problem to Portfolio Backlog Item (1:1): ensures a Portfolio Backlog Item is created for Problems requiring a future fundamental/big fix/enhancement to the Digital Product
- Problem to Defect (1:n): enables the creation of Defects when emergency/specific fixes require development
- Problem to Knowledge Item (n:m): enables the creation of known error(s) (Knowledge Items) when the root cause of a Problem is identified
- Problem to Actual Product Instance (n:m): Problem records are mapped to the affected Actual Product Instance(s)
- Incident to Problem (n:m): connection between Incidents that are converted to Problems to permanently address severe/repeating Incidents

9.1.4. Knowledge Functional Component

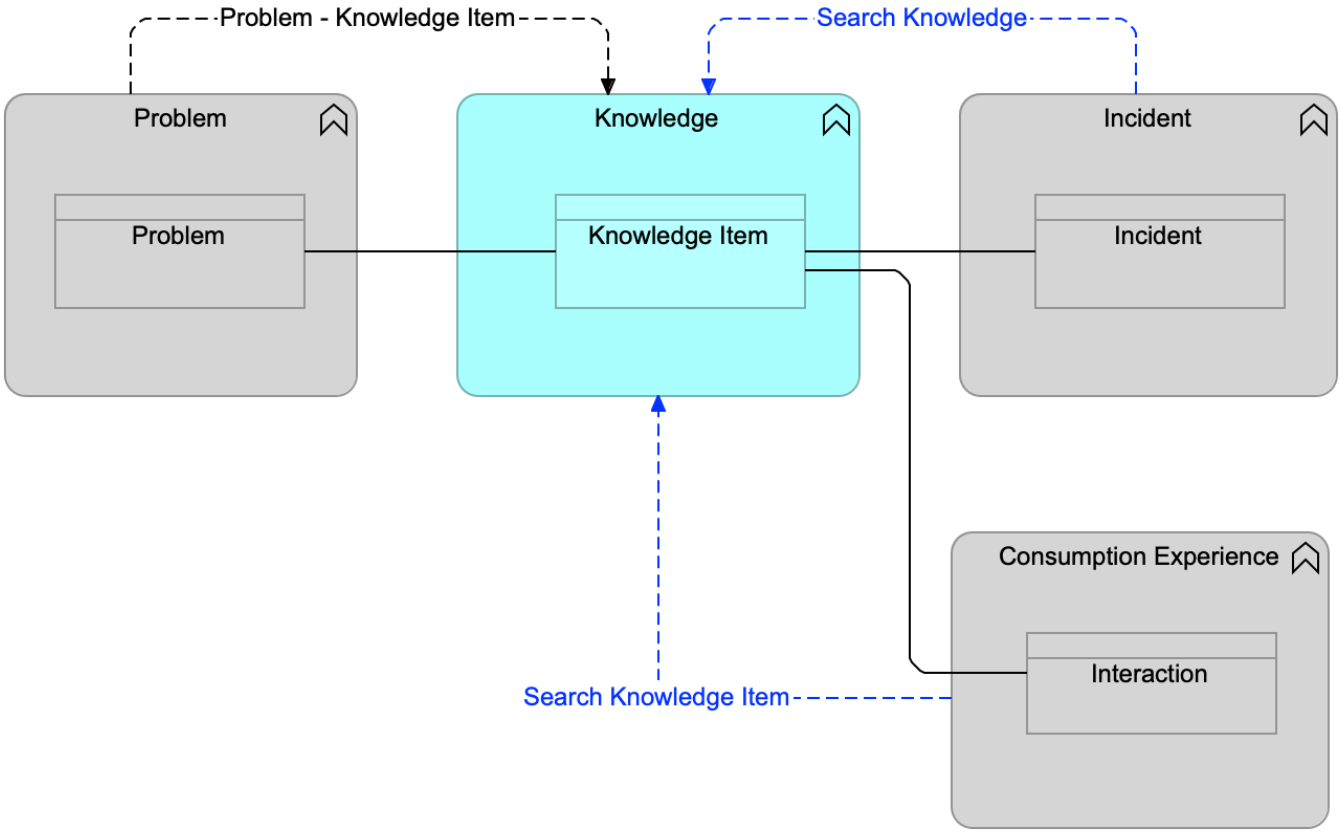

Figure 9-5. Knowledge Functional Component Model

Purpose

The Knowledge functional component includes searchable content which can take on multiple formats; for example, general knowledge documents including how to, technical guidance, frequently asked questions, known errors, webinars, videos, training materials, or other information to be managed. The Knowledge functional component contains known errors which describe a known condition/issue which exists for a Product Instance. This functional component increases the contribution to knowledge by providing all users with the ability to generate new content. It improves the accessibility of knowledge in the organization by:

- Supporting keyword search capabilities
- Providing filter capabilities based on various attributes of the Knowledge functional component, such as subject category, time range, source types (internal *versus* external)
- 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
- Reducing the number of requests for information/knowledge that arrive at the service desk; service consumers and staff consume third-party knowledge through the same experience as the formal and informal forms of knowledge the company provides

The Knowledge functional component supports the value streams:

- Deploy
- Release
- Consume
- Operate

Functional Criteria

The Knowledge functional component:

- Shall be the system of record for all Knowledge Item records
- Shall manage the state and lifecycle of the Knowledge Item
- Shall receive known error information from the Problem to create a Knowledge Item
- Shall provide functionality to enable the service consumers and staff to rank Knowledge Items, thus improving future knowledge consumption
- Shall provide knowledge in the form of content that helps to address the needs of both service consumers and digital operations
- Shall increase the contribution to knowledge by providing all users with the ability to generate new content

- Shall provide knowledge to digital operations as part of the diagnostics and remediation activities
- Shall reduce the number of requests for information/knowledge that arrive at the service desk through self-service
- May aggregate multiple (internal and external) Knowledge Item sources
- May include searchable content which can be structured IT/supplier-produced articles
- May share knowledge with consumers via the Consumption Experience engagement portal
- May improve accessibility of knowledge in the organization by:
  - Supporting keyword search capabilities
  - Providing filter capabilities based on various attributes of the Knowledge functional component, such as subject category, time range, source types (internal *versus* external)
  - 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
- May allow service consumers and staff to consume third-party knowledge through the same experience as the formal and informal forms of knowledge the company provides

9.1.4.1. Knowledge Item Data Object

Purpose

The Knowledge Item data object contains information to be used by both operators and the consuming business as structured searchable content managed and maintained.

Key Attributes

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

- **Id:** unique identifier of the Knowledge Item
- **Title:** title of the Knowledge Item
- **Status:** for example, draft, revise, published, review, archived
- **Author Id:** unique identifier of the responsible author
- **Publish Date:** date/time on which the item is available for publication
- **Expiry Date:** date/time on which the item is no longer visible
- **Body:** body text, video, or any other content that is published

Key Data Object Relationships

The Knowledge Item data object shall maintain the following relationships:

- Knowledge Item to Problem (n:m): if a link to a Problem exists, the Knowledge Item will need changing when a Problem is (partly) resolved
- Knowledge Item to Interaction (n:m): one or more Knowledge Items can be related to an Interaction when a consumer searches for knowledge during a self service experience
- Knowledge Item to Incident (n:m): one or more Knowledge Items can be related to an Incident as part of the Incident resolution
- Knowledge Item to Knowledge Item (n:m): knowledge can be linked if related

9.2. Assure Function

The Assure functionality centers around monitoring, configuration tracking, event correlation and diagnostics.

9.2.1. Configuration Functional Component

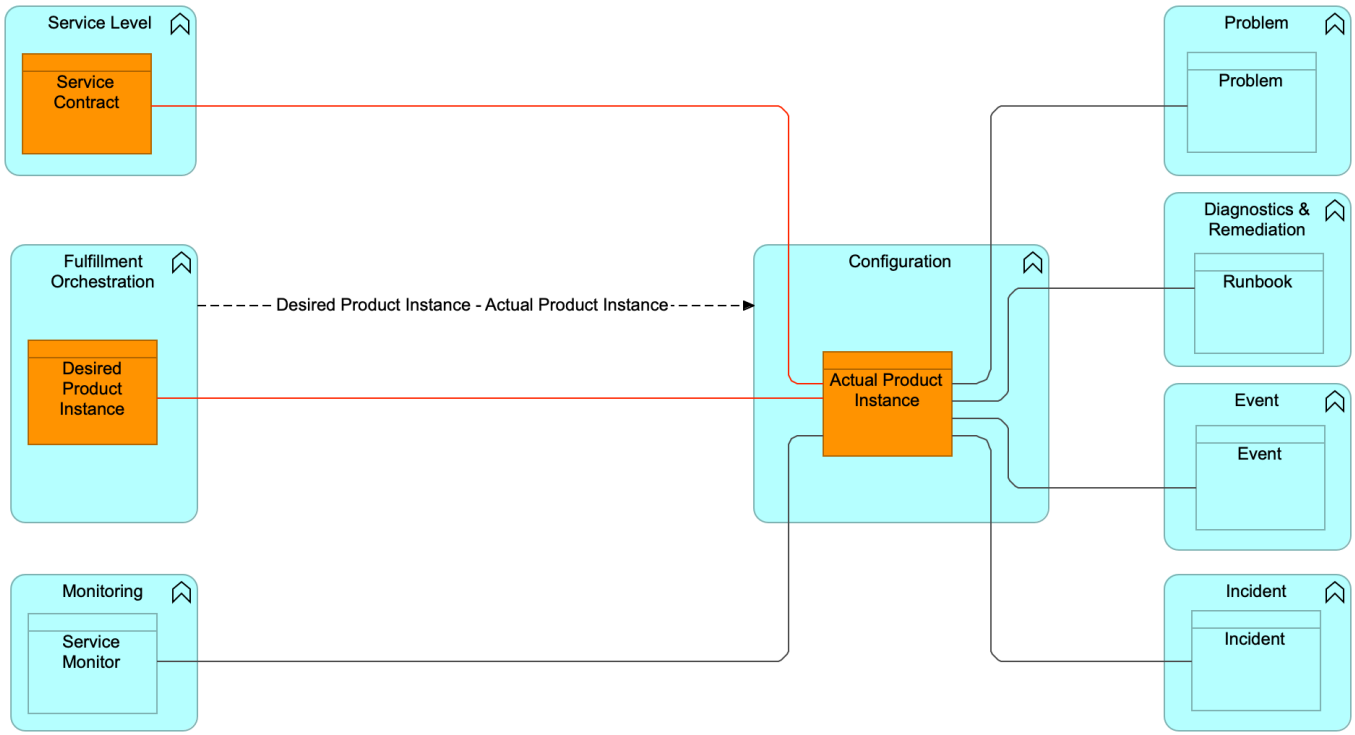

Figure 9-6. Configuration Functional Component Model

Purpose

The Configuration functional component is focused on tracking the inventories of Actual Product Instances and the associated relationships of the underlying systems for the specific Digital Product. The Actual Product Instance represents the entire stack of resources and CIs including integrations and

dependencies. In addition, this functional component tracks the relationship of an Actual Product Instance to other Digital Products – the Actual Product Instances needed to function, and the relationships between the underlying systems. Its purpose is to identify, control, record, report, audit, and verify service items; including versions, constituent components, their attributes, and relationships.

The Configuration functional component supports the value streams:

- Deploy
- Consume
- Operate

Functional Criteria

The Configuration functional component:

- Shall be the system of record for all Actual Product Instances and their associated relationships
- Shall manage the lifecycle of the Actual Product Instance
- Shall create Actual Product Instance(s) and underlying system(s) based on the Desired Product Instance in the Fulfillment Orchestration functional component
- Shall serve as the data store for the realization of the service in the production environment
- Shall calculate and provide impact analysis on proposed changes to the current enterprise landscape
- Shall calculate and provide the business impact of the Incident to help in the prioritization process
- Shall calculate and provide the business impact of the Event to help in the prioritization process
- May be populated by service discovery

9.2.1.1. Actual Product Instance Data Object

Purpose

The Actual Product Instance data object represents the realized deployment of a specific Desired Product Instance. It includes CIs that represent the implemented Actual Product Instance components.

Key Attributes

The Actual Product Instance data object shall have the following key data attributes:

- **Id:** unique identifier of the Actual Product Instance
- **Name:** name of the Actual Product Instance
- **Type:** type of the Actual Product Instance (e.g., infrastructure service, customer-facing service, enabling service, front office)

- **Configuration Items:** model of the configuration of the Product Instance as a set of interconnected CIs
- **Create Time:** date/time the Actual Product Instance was created
- **Last Modified Time:** date/time the entry was last modified in a significant way
- **Location:** location of the Actual Product Instance – this can vary from high-level country to city or low-level, such as a building or a room

Key Data Object Relationships

The Actual Product Instance data object shall maintain the following relationships:

- Desired Product Instance to Actual Product Instance (1:1): each CI should be traceable to the planned configuration described in the Desired Product Instance
- Problem to Actual Product Instance (n:m): Problem records are mapped to the affected Actual Product Instance(s)
- Runbook to Actual Product Instance (n:m): Runbook records are mapped to the associated Actual Product Instance(s)
- Incident to Actual Product Instance (n:m): Actual Product Instance(s) to which the Incident is associated
- Actual Product Instance to Event (n:m): Actual Product Instance(s) associated with the Event
- Actual Product Instance to Service Contract (1:n): connection between the Actual Product Instance and the Service Contract in which it is measured
- Service Monitor to Actual Product Instance (1:n): Actual Product Instance being monitored
- Actual Product Instance to Actual Product Instance (n:m): an Actual Product Instance can depend on services delivered by other products

9.2.2. Monitoring Functional Component

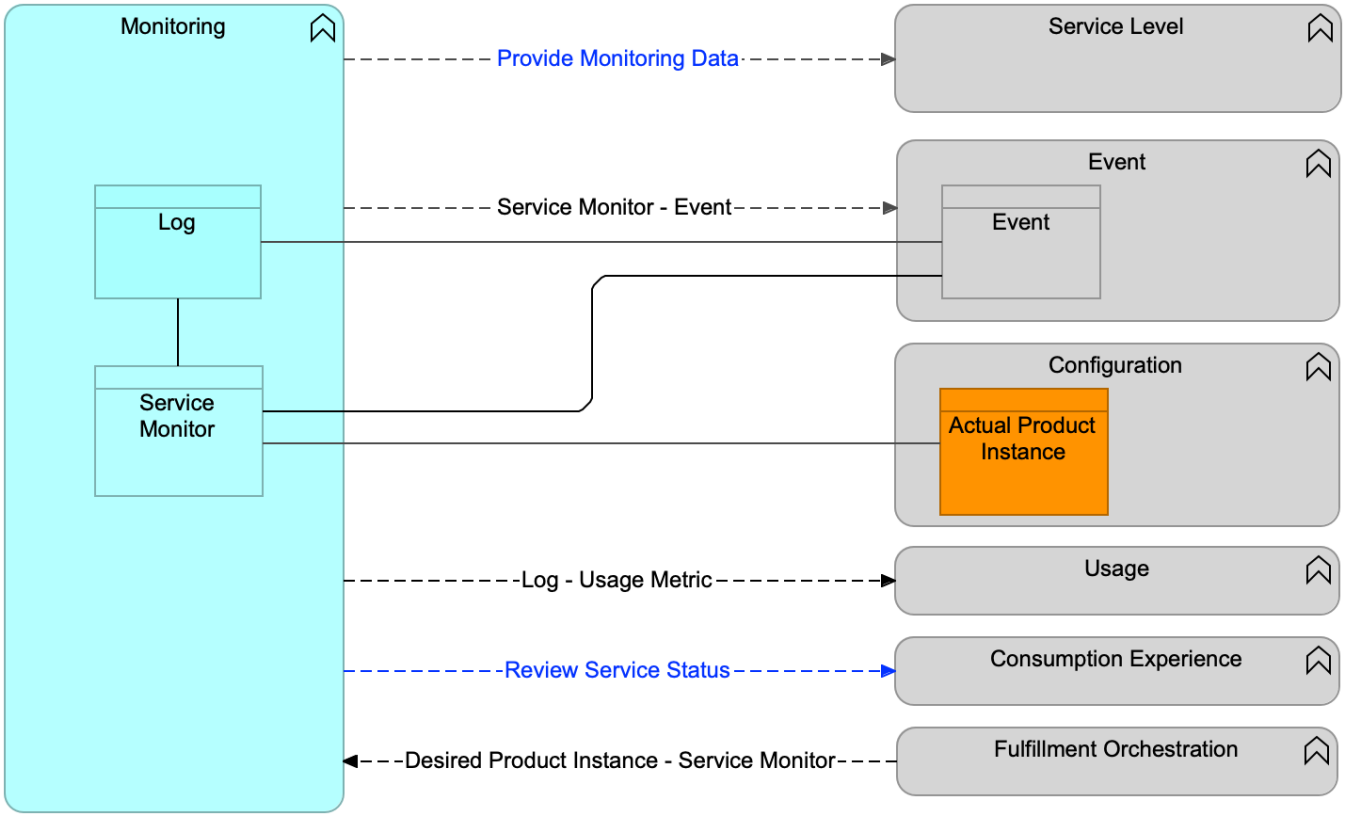

Figure 9-7. Monitoring Functional Component Model

Purpose

The 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 used to monitor aspects of service usage and Service Contracts. The results of monitoring are often captured in logs which can be referenced by other components. The collected data may trigger an Event. Monitoring can perform anomaly detection or detect certain patterns to predictively detect issues.

It is in charge of storing all measurement results and calculating compound measurements. ML and AI can be used to collect operational data used to predict certain service conditions or service degradation. The creation of the monitor definitions are done earlier in the product lifecycle in the Integrate value stream and are delivered to the Monitoring functional component by the Fulfillment Orchestration functional component. The Monitoring functional component also provides feedback on all aspects of the Digital Product and passes that information to the Evaluate, Explore, and Integrate value streams.

The Monitoring functional component supports the value streams:

- Integrate
- Deploy
- Operate

Functional Criteria

The Monitoring functional component:

- Shall be the system of record for all Service Monitors
- Shall monitor all aspects of an Actual Product Instance
- Shall store all of the results of the measurements being made on the Actual Product Instance
- Shall calculate the results of compound Service Monitors from one or more simple measurements
- Shall manage the lifecycle of the Service Monitor
- Shall create, run, and manage monitors that measure all aspects/layers of an Actual Product Instance, including:
  - Shall monitor infrastructure (system and network)
  - Shall monitor applications
  - Shall monitor security
  - Shall monitor Product Instances
  - May monitor customer experience
  - May monitor service-level attainment or breach for the Service Level functional component
  - May monitor usage for the Usage functional component
  - May monitor service status for the Consumption Experience functional component
- May receive Service Monitor definitions from the Fulfillment Orchestration functional component

9.2.2.1. Service Monitor Data Object

Purpose

The Service Monitor data object defined monitoring policies for CIs; that is, for the Actual Product Instance, or its underlying systems. The purpose is to understand the current status of the running service. The monitoring policies can for example specify:

- Monitoring of the performance and availability (of the Digital Products and its components and dependencies)
- Monitoring of user experience
- Monitoring of security and scans for potential vulnerabilities

- Monitoring of availability of the actual Digital Product and its components (and dependencies)
- Monitoring of performance and capacity (e.g., utilization of resources)

The Service Monitor definition is created in the Integrate value stream as part of the release package, and activated from the Fulfillment Orchestration functional component.

Key Attributes

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

- **Id:** unique identifier for the Service Monitor
- **Name:** name of the Service Monitor
- **Description:** description of the Service Monitor
- **Type:** type of the Service Monitor (e.g., system, application, network, security)
- **Measurement Definitions:** definitions of the measurements that the Service Monitor is collecting about the monitored entity (i.e., CI)
- **Last Run Time:** date/time that the Service Monitor was last run
- **Last Run Status:** the success, or not, of the last run of the Service Monitor

Key Data Object Relationships

The Service Monitor data object shall maintain the following relationships:

- Service Monitor to Log (1:n): enables the traceability from the log entries that are created to the Service Monitor that defined the collection of data
- Service Monitor to Event (1:n): an event can be created/updated by a Service Monitor
- Service Monitor to Actual Product Instance (1:n): the Actual Product Instance is the CI being monitored

9.2.2.2. Log Data Object

Purpose

A Log data object captures information related to CIs in order to better understand performance, health, and/or usage of an Actual Product Instance.

Key Attributes

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

- **Id:** unique identifier of a Log record
- **Timestamp:** time of the Log event
- **Data:** collected Log text

Key Data Object Relationships

The Log data object shall maintain the following relationships:

- Log to Event (n:m): according to monitoring policies, some Log events are forwarded as service events
- Log to Service Monitor (n:1): associated Log to the Service Monitor definition that enables the collection of the Log record

9.2.3. Event Functional Component

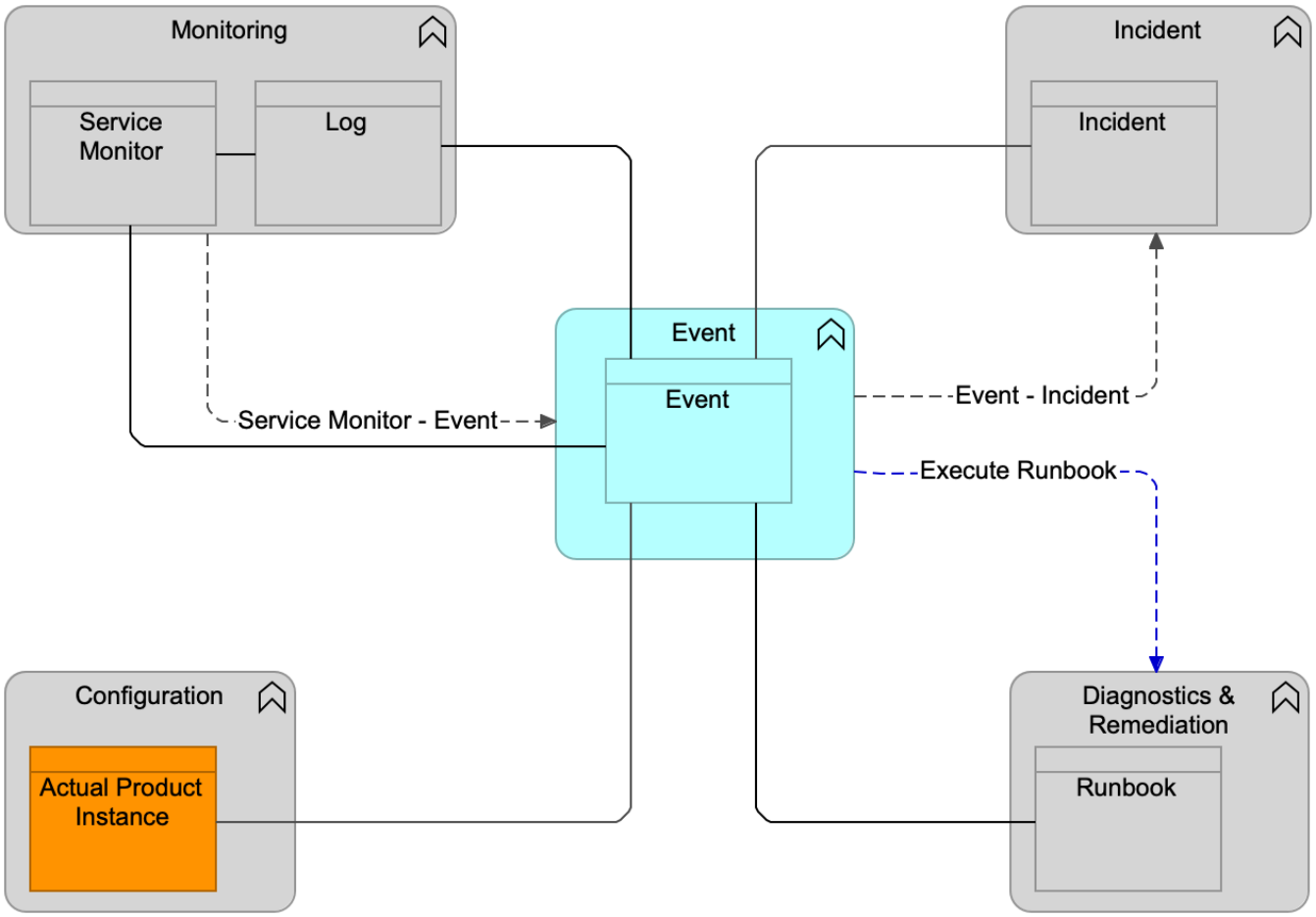

Figure 9-8. Event Functional Component Model

Purpose

The Event functional component manages Events through the Event lifecycle for Events that occur on any digital 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 alerts and/or Incidents, and for diagnostics and remediation activities.

The Event functional component supports the value streams:

- Deploy
- Operate

Functional Criteria

The Event functional component:

- Shall be the system of record for all Events
- Shall manage the state and lifecycle of Events
- Shall manage the correlation between Events
- Shall categorize Event data
- Shall receive Event information from Service Monitors from the Monitoring functional component
- Shall forward Events categorized as Incidents to the Incident functional component
- May initiate a Change based on Event data in the Change functional component (e.g., trigger for capacity increase)
- May send automated remediation (Runbook) to the Diagnostics & Remediation functional component

9.2.3.1. Event Data Object

Purpose

The Event data object represents an alert/notification signifying a change of state of a monitored CI or Actual Product Instance.

Key Attributes

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

- **Id:** unique identifier of an Event
- **Name:** name of an Event
- **Category:** category of an Event (e.g., info, warning, error); aids in determining assignment and prioritization
- **Type:** categorizing the Event to causal or symptom type
- **Status:** current stage in the lifecycle of an Event
- **Status Time:** time stamp for the Status attribute
- **Severity:** severity of the Event
- **Threshold Definitions:** definitions of the thresholds by which the Event severity is determined
- **Assigned To:** group or person that is assigned to handle the Event

- **Is Correlated:** is the Event correlated to other Events?

Key Data Object Relationships

The Event data object shall maintain the following relationships:

- Event to Incident (n:m): enables the connection between Incidents and Events and supports the integration lifecycle between them
- Event to Change (1:n): associated Event is available for Change processing
- Event to Actual Product Instance (n:m): Actual Product Instance(s) associated with the Event(s)
- Event to Log (n:m): enables the traceability from the Events that are created to the Log object from which they originated
- Event to Service Monitor (n:1): an event can be created/updated by a Service Monitor.
- Event to Event (n:m): several events can correlate to the same root cause, and thus they relate to a primary event

9.2.4. Diagnostics & Remediation Functional Component

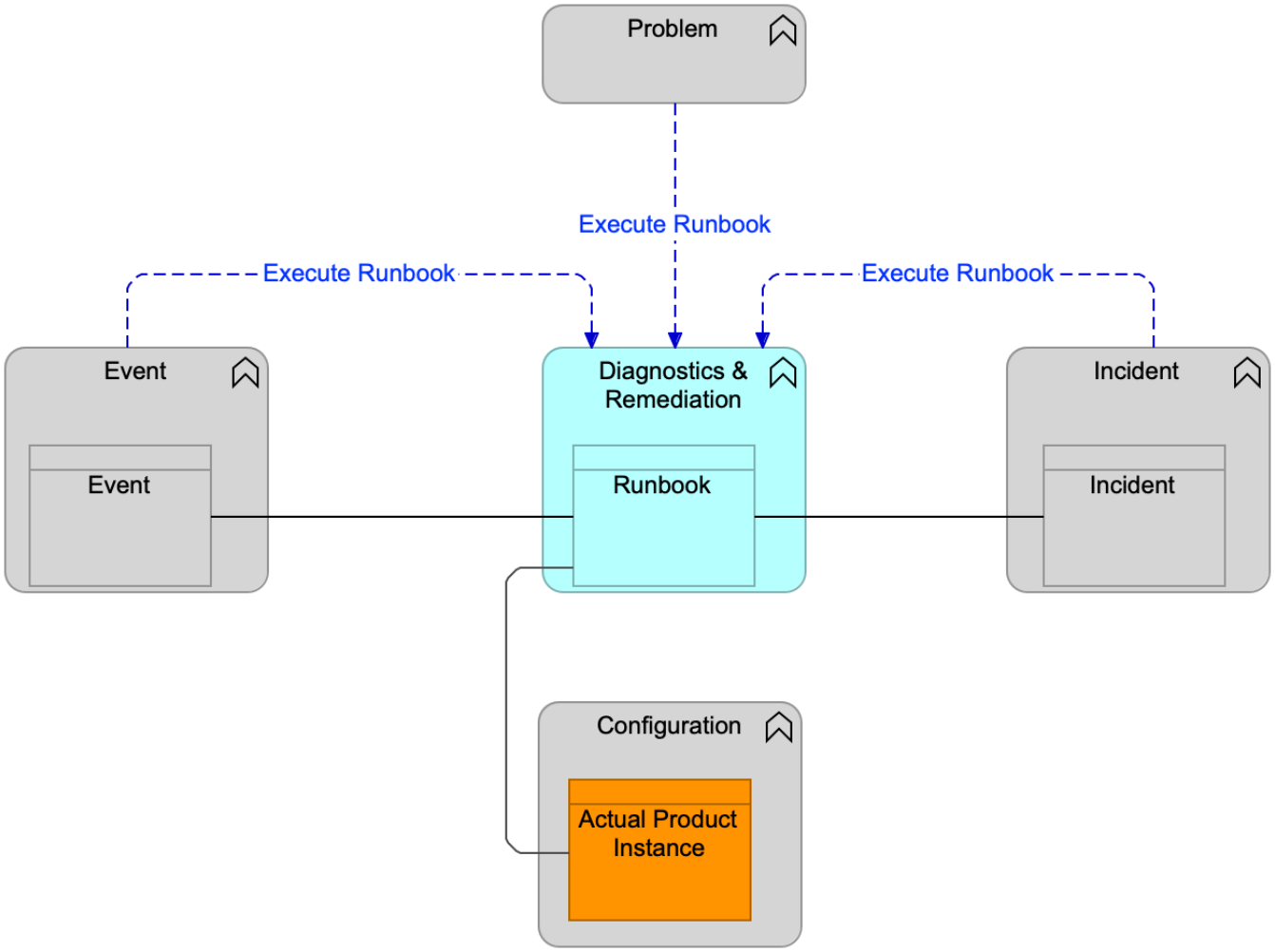

Figure 9-9. Diagnostics & Remediation Functional Component Model

Purpose

Through the use of manual and automated Runbooks, the Diagnostics & Remediation functional component provides diagnostics information and/or remediation steps to shorten the MTTR. During the Event lifecycle, some categories of Events can serve as initiators of alerts and/or Incidents and also for diagnostics and remediation activities. Machine Learning (ML) and Artificial Intelligence (AI) can be utilized in this function to automate the diagnosis and remediation of Events. Runbooks help to streamline diagnosis and remediation for service functions by applying knowledge solutions to service anomalies.

The Diagnostics & Remediation functional component supports the value streams:

- Deploy
- Operate

Functional Criteria

The Diagnostics & Remediation functional component:

- Shall be the system of record for all Runbooks
- Shall manage the Runbook lifecycle
- May allow an Event to trigger a Runbook for diagnostics or remediation purposes
- May allow an Incident to trigger a Runbook for diagnostics or remediation purposes
- May allow a Problem to trigger a Runbook for diagnostics or remediation purposes

9.2.4.1. Runbook Data Object

Purpose

The Runbook data object is a routine compilation of the procedures and operations which the administrator or operator of the system carries out. A Runbook can be either a manual process or an automated script.

Key Attributes

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

- **Id:** unique identifier of the Runbook
- **Description:** description of the Runbook
- **Category:** aids in determining assignment and prioritization
- **Execution Time:** date/time the Runbook was last executed

Key Data Object Relationships

The Runbook data object shall maintain the following relationships:

- Actual Product Instance to Runbook (n:m): track Runbooks and the Actual Product Instance(s) with which they are associated
- Event to Runbook (n:m): an Event is related to one or more Runbooks used to resolve the Event
- Incident to Runbook (n:m): an Incident is related to one or more Runbooks used to resolve the Incident

Chapter 10. Supporting Functions

Figure 10-1. Supporting Functions Model

The Supporting Functions, shown in Figure 10-1 and defined as part of the IT4IT Reference Architecture, underpin the Digital Value Network and often support multiple value streams and functional components. They are enablers for managing Digital Products. These Functions are aligned to the Digital Product lifecycle to support the effective delivery of common outcomes and goals. They are typically related to, or are part of the enterprise Supporting Functions such as Finance, HR Management, and Procurement. However, these shared enterprise functions often need specific extensions or even dedicated tools (and integrations with other functions within the Digital Value Network) to manage Digital Products.

For example:

- Calculating the service cost of Digital Products based upon actual consumption and usage (which requires integration with the Digital Product Backbone)
- Allocating and managing budgets associated with the delivery of Digital Products (e.g., allocating budgets for Digital Products)
- Managing relationships with service providers and vendors involved in the Digital Value Network (such as SaaS and IaaS vendors)
- Managing contracts and purchase orders linked to the Digital Product Portfolio, Orders, Service Contracts, and Actual Product Instances
- Managing risk and compliance related to the Digital Products
- Providing data analytics and reporting in the context of Digital Product delivery
- Providing common communication and collaboration functionality (which need to be integrated into the digital delivery model)

The Supporting Functions consist of functional components and data objects that are essential for delivering and managing Digital Products. They are necessary to optimize value while managing risks, compliance, and costs. Although they play an important role in delivering Digital Products as a shared function, they are not defined in the IT4IT Standard as normative. The Supporting Functions will be defined in more detail in a future release of the IT4IT Reference Architecture.

Each of the Supporting Functions is described in the context of managing the Digital Product lifecycle and highlighting the interaction between the functional components.

10.1. Financial Management Function

The Financial Management function manages the financial aspects of the Digital Product lifecycle, including:

- Funding of Digital Products (and related Scope Agreements) and managing budgets
- Cost accounting for Digital Products (and all associated activities)
- Maintaining Cost Models and calculating product costs (e.g., defined Cost Models, calculate costs based upon actual consumption)
- Define pricing models for Digital Products
- Provide billing (or cost allocation) to consumers

The Financial Management function is part of/integrated with the enterprise-wide Financial Management function. This functionality supports the planning and tracking of the costs of managing the Digital Products throughout the entire lifecycle. The IT4IT Reference Architecture defines the flow of information for Digital Product delivery and offers end-to-end traceability for the Financial Management of Digital Products.

The Financial Management function intersects with many functional components, including:

- Proposal functional component: for allocated and monitoring budgets related to Scope Agreements
- Product Portfolio functional component: for managing the lifecycle costs associated with the delivery of Digital Products
- Monitoring and Usage functional components: to collect consumption and usage data
- Chargeback functional component, to provide chargeback/showback of services consumed by customers

10.1.1. Cost Modeling Functional Component

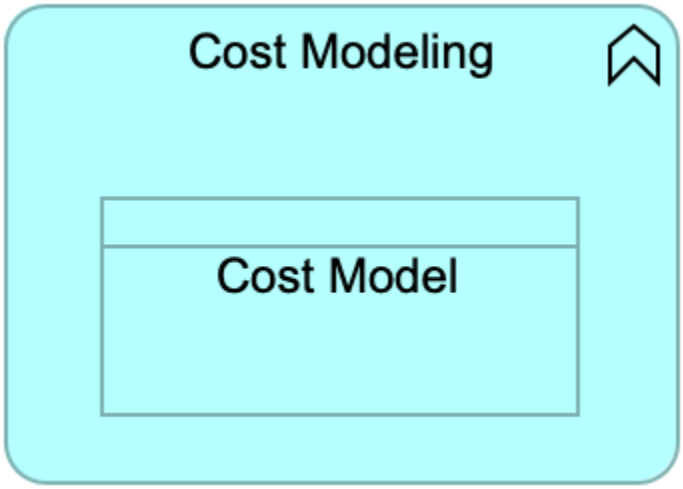

Figure 10-2. Cost Modeling Functional Component Model

Purpose

The main purpose of the Cost Modeling functional component is to associate cost elements (internal or external) with the resources that bear these costs (hardware, software, or people). The Cost Model serves as the cost reference to feed the Chargeback functional component with elementary costs (costs per unit of work). The Chargeback functional component will multiply these elementary costs by a Usage factor to compute the cost of a service.

The model should also be used to evaluate the total cost of IT and how they are spread over all resources. A simulation, or scenario-based rendering, may also be supported by the Cost Model.

The Cost Modeling functional component supports the value streams:

- Release
- Consume

Functional Criteria

The Cost Modeling functional component:

- Shall be the reference for cost allocations on the corresponding resource
- May be structured by categories of resource (e.g., type of resources, typically grouped by tower, sub-tower)
- May include a simulation capability to evaluate the distribution of all IT costs

10.1.1.1. Cost Model Data Object

Purpose

The Cost Model data object defines the way a Digital Product accounts for its cost pertaining to delivered services.

Key Attributes

The Cost Model data object shall have the following key data attributes:

- **Id:** unique identifier of the Cost Model
- **Name:** descriptive name for the Cost Model
- **Description:** text description detailing the purpose and principles of construction of the Cost Model
- **Status:** indicates if the Cost Model is in use, depreciated, or experimental
- **Version:** current version of the model (models evolve over time)
- **Calculation:** formula for calculating cost
- **Resource Type:** type of resource considered (hardware, software, HR, or external service – IaaS, PaaS, SaaS)
- **Resource Id:** identification of the Resource that will incur cost charges
- **Resource Unit of Work:** the most granular element for which a cost is charged; e.g., hours, GBytes, CPUs)
- **Cost per Unit of Work:** monetary amount per unit of work (e.g., *x* euros per gigabyte of storage)
- **Time Period:** time span for which the cost per unit of work has been calculated

Key Data Object Relationships

The Cost Model data object shall maintain the following relationships:

- Cost Model to Chargeback Record (1:n): a “live” Cost Model informs the Chargeback Record of the cost per unit of work; for a basic service (i.e., not a composite of other services), the Chargeback Record will multiply the cost per unit of work of the Resource underlying the service by the Usage data from its associated Usage Record
- Cost Model to Resource: (1:n): the Cost Model must be structured by the various Resource types used in IT

10.1.2. Investment Functional Component

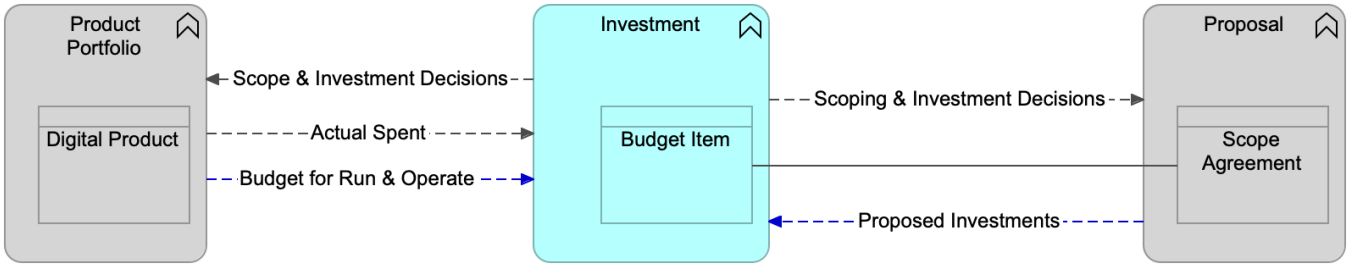

Figure 10-3. Investment Functional Component Model

Purpose

The main purpose of the Investment functional component is to manage the portfolio of all authorized investments. It supports the budget cycle (from budget creation, arbitration, and tracking to multi-year Investment planning, fed from the business and technology strategy and structured by the Enterprise Architecture Roadmaps).

The Investment functional component supports the Evaluate value stream.

Functional Criteria

The Investment functional component:

- Shall be the authoritative source for all investments requested over a given time period
- Shall manage the entire investment lifecycle
- Shall receive investment requests for development from the Proposal functional component
- May receive proposed investments for run and maintain and non-service investments from investment owners
- Shall assess proposal feasibility (for example, for cost, value) and obtain the required approval from Finance
- Shall communicate the status of the final scoping and investment decisions back to the respective stakeholders

10.1.2.1. Budget Item Data Object

Purpose

The Budget Item data object is an approved investment pertaining to one or more Digital Products. All Budget Items taken together in a specific financial period comprise the approved budget for the entire Digital Product Portfolio over that time period.

Key Attributes

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

- **Id:** unique identifier of the service or investment (e.g., overhead) for which the Budget Item is recorded
- **Financial Period:** financial period for which the Budget Item is recorded
- **Investment Type:** nature of investment (e.g., run, development, overhead)
- **Approved Budget:** approved budget
- **Spend:** actual spend
- **Forecast:** estimated spend at the end of the financial period

Key Data Object Relationships

The Budget Item data object shall maintain the following relationships:

- Budget Item to Digital Product (n:1): one Budget Item shall hold budget for a single Digital Product and budget for one Digital Product may be spread across multiple Budget Items
- Budget Item to Scope Agreement (1:n): helps track how much budget is allocated to which Scope Agreement(s)

10.2. Governance, Risk, & Compliance Function

Managing risk and compliance is an integral part of the development and delivery of Digital Products. The Governance, Risk, & Compliance function ensures that security, risk, and compliance are fully embedded into the digital delivery network, including all involved parties and vendors. The ability to identify and manage risk in an increasingly complex digital environment has become a priority in organizations. Organizations need to understand the variables that affect their business, so describing, classifying, managing, and mitigating risk factors is very important. Important concepts in this context are “Secure by Design” and “Compliant by Design” (including “Privacy by Design”). This Supporting Function needs to be integrated with the enterprise-wide Risk Management function. It includes the relationships with functions like internal audit, compliance, risk, legal, and finance.

The Governance, Risk, & Compliance function ensures the transparency and traceability of the risks and compliance information related to the Digital Product delivery. It is tightly linked to the Policy functional component where the Policies and related control requirements are maintained.

Examples of activities within this function are:

- Define risk assessment methods
- Maintain a risk register (and related mitigation plans and requirements)
- Perform risk assessments to identify potential risks associated with Digital Products (and their releases and deployments)

- Assess the risks of vendors in the ecosystem (vendor Risk Management)
- Define and ensure mitigation plans and actions are implemented to reduce risks
- Monitor and ensure products are developed according to defined policies and security requirements
- Continuous monitoring and evaluation of compliance and managing deviations/exceptions
- Audit management – perform audits to identify issues, exceptions, and determine improvement opportunities

Examples of information managed by this function are:

- Risk assessments, such as those related to BIA, TVA, and DPA
- Risks (associated with the Digital Products, Product Release, or Actual Product Instances)
- Vendor risks
- IT audits and audit findings
- Compliance evidence and compliance-related exceptions/issues
- Risk issues (actual occurrence of a risk item)

Examples of key relationships with other functional components are, for example:

- Policy: to understand the Policies and control Requirements
- Product Portfolio: to perform the risk assessments against the Digital Products (and associated Product Releases)
- Requirements and Product Design: to ensure products are developed Secure by Design and Compliant by Design (e.g., Privacy by Design)
- Portfolio Backlog and Product Backlog: to add backlog items needed to resolve risk-related issues, audit findings, and non-compliance records
- Incident and Problem: to ensure risk issues/findings or compliance exceptions are handled by Incident and/or Problem Management
- Test: to ensure that risk and compliance is part of the test execution
- Sourcing & Vendor Management function: to manage the risk and compliance of vendors in the digital ecosystem
- Workforce Management function: to ensure all employees are aware of the security, risk, and compliance requirements
- Intelligence & Reporting function: reporting and data analytics related to risk and compliance

10.3. Workforce Management Function

The Workforce Management function (for digital) supports the HR Management aspects within the digital delivery model. The Workforce Management function is part of the overall Workforce Management function within an organization.

In a digital ecosystem it is important to support employees in their career, providing coaching and training, as well ensuring the resource portfolio, skills, and competencies are aligned with the strategic direction. This function defines the functional components and data objects which need to be integrated with the digital delivery model.

This includes, for example:

- Onboarding and off-boarding employees
- Maintaining team administration (defining the teams and related Digital Products)
- Supporting Product Portfolio Managers and product owners with resource planning, and allocating resources to teams
- Monitoring and evaluating employee satisfaction
- Maintaining a competency and skills framework
- Defining job profiles and their associated roles and responsibilities within the Digital Value Network (such as the Product Manager)
- Supporting employees with HR-related questions and issues
- Supporting employees in their career, providing coaching and training
- Identifying and planning training associated with the introduction of new (or modified) Digital Products

Examples of the information part of this function are:

- Employee (including external contractors) linked to an Identity in the Identity functional component
- Job profile and associated roles (linked to the employee and the Identity functional component)
- Team (e.g., DevOps or support teams) related to the Digital Products (and linked to the Identity functional component) for access management
- Resource allocation of employees to teams

10.4. Sourcing & Vendor Management Function

The Digital Value Network consists of an increasing number of vendors participating in the development and operation of Digital Products. It is becoming an essential function for an organization to be able to leverage ecosystems to create business value. This includes creating and maintaining a dynamic partner network of vendors involved in the end-to-end digital delivery model; for example, to

enable the effective and efficient onboarding and off-boarding of vendors, as well as to manage the relationships with vendors. Vendors in the digital ecosystem are not just providers of services, software, or hardware, but are essential partners in co-creating Digital Products and delivering value and outcome.

The Sourcing & Vendor Management function is typically part of an enterprise-wide shared function; however, it requires specific experience, skills, and integrations in the context of Digital Product delivery.

Examples of sub-functions within the Sourcing & Vendor Management function:

- Contract Management (for Digital Products and services)
- Procurement
- Vendor Management

This function enables researching and market intelligence, sourcing vendors, obtaining quotes with pricing, negotiating contracts, managing relationships (and primary stakeholders), managing the contracts, purchase orders, monitoring and evaluating performance, managing issues and disputes, benchmarking, and ensuring payments are made according to the contracts and usage of their services.

Examples of information managed within the Sourcing & Vendor Management function are, for example:

- Vendor: managing the system of record of all vendors involved in the digital ecosystem
- Contract: managing the lifecycle of contracts associated with the Digital Products and other data objects within the Digital Product Backbone (such as license records in the asset administration)
- Purchase Order: related to the Digital Product Backbone and other activities such as Scope Agreements or Portfolio epics

10.5. Intelligence & Reporting Function

The Intelligence & Reporting function provides data management, analytics, and reporting to support the various functional components within the Digital Value Network.

A data-driven approach is needed to manage the new digital ecosystem, providing insight, transparency, and traceability for continuous improvement. This function provides a consolidated view of the data provided by the different functional components in the Digital Value Network. The context of the data is provided by the Digital Product Backbone.

The Intelligence & Reporting function typically leverages the company-wide data analytics and reporting functionality (such as data lake or reporting tools). However, for managing Digital Products, specific Intelligence & Reporting solutions need to be implemented and configured. The Intelligence & Reporting function for example the following sub-functions or features:

- Define and manage key metrics and KPIs (such as those related to value, risk, cost, compliance, and user experience)
- Publish a standard data catalog and associated definitions (and master data sources)
- Define data governance and data ownership
- Collect and consolidate data from the various functional components (e.g., into a data lake or operational data store)
- Provide data analytics and reporting functionality
- Support the various functional components in their reporting and analytics needs (e.g., service level reporting for the Service Contracts)

Examples of information managed by the Intelligence & Reporting function are:

- Metrics and KPI
- Reports (e.g., service level reports)

10.6. Collaboration & Communication Function

The Collaboration & Communication function supports many other IT4IT Functional Components by enabling optimal collaboration and communication between the various stakeholders and parties within the ecosystem. This includes internal and external collaboration with all parties in the digital ecosystem.

In a modern digital work environment it is essential that all people and teams work together in a transparent way, with a high visibility of work and maximum support from the right collaboration tools. Most of these collaboration tools are part of enterprise-wide collaboration tools (including email, chat, collaboration and communication platforms).

The Collaboration & Communication function contains various functional components to support for example:

- Online collaboration (e.g., Slack™, Cisco WebEx®, Microsoft Teams™)
- Email and chat
- Alerting and notification systems (information stakeholders in cases of critical issues)

This function maintains the different communication channels, involved stakeholders, and their communication preferences. This function needs to be tightly integrated with functional components within the digital enterprise, such as:

- Publish status updates of new builds, test results, and releases to stakeholders
- Provide alerting/notification in case of Incidents and support major Incident collaboration
- Event notification or alerting (for stakeholders to subscribe to relevant Events)
- Support collaboration between teams to resolve Incidents and Problems

Chapter 11. IT4IT Concepts and Metamodel

This section formally defines all the concepts of the IT4IT Standard, which includes the IT4IT Metamodel.

The IT4IT Standard uses the ArchiMate modeling language, and thus the IT4IT Metamodel is defined as an IT4IT solution pattern in the ArchiMate language. This consists of the IT4IT Reference Architecture as a model specified in the ArchiMate language, which can be exchanged using the *ArchiMate® Model Exchange File Format for the ArchiMate Modeling Language, Version 3.1* [C19C].

The IT4IT Standard is complemented with a simplified notation that makes it easier to communicate the high-level concepts of the IT4IT Standard to people not familiar with the ArchiMate language. Note that the simplified notation has a direct translation into the ArchiMate representation, and thus is as formal as the ArchiMate representation.

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.

Note that the following definitions are ordered to reflect a hierarchy of refinement level as introduced in IT4IT Abstractions.

Refer to Acronyms and Abbreviations for an alphabetic list of acronyms and abbreviations used in this document.

11.1. IT4IT Metamodel

The IT4IT Standard introduces ten normative concepts as well as a number of complementary non-normative concepts. Their names and relations are pictured in Figure 11-1.

The number associated with each concept indicates at which abstraction level of the IT4IT Standard the concept is introduced. Some concepts are introduced at a high abstraction level but only formally defined further down; this is indicated by one or two numbers in the metamodel objects; see IT4IT Abstractions for details.

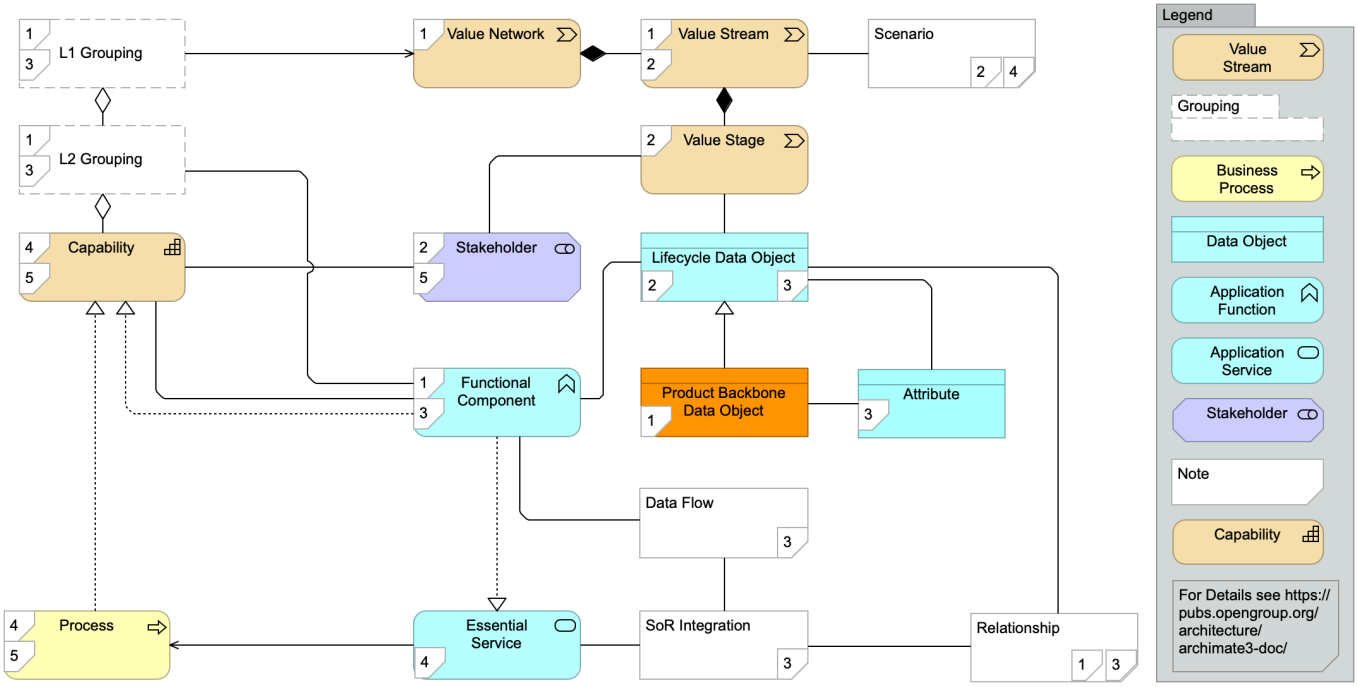

Figure 11-1. IT4IT Metamodel

11.2. IT4IT Abstractions

The IT4IT Reference Architecture is communicated using multiple levels of abstraction, which is inspired by a similar approach employed by the eTOM® frameworks from the TM Forum. Each abstraction level expands on the previous level to expose more details and prescriptive guidance. The upper levels are vendor and implementation independent and provide generic views that are suitable for strategy and planning purposes as well as for creating roadmaps for implementing the IT4IT Reference Architecture. The lower levels provide 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 introduces five abstraction levels and standardizes the first three, as shown in Figure 11-2.

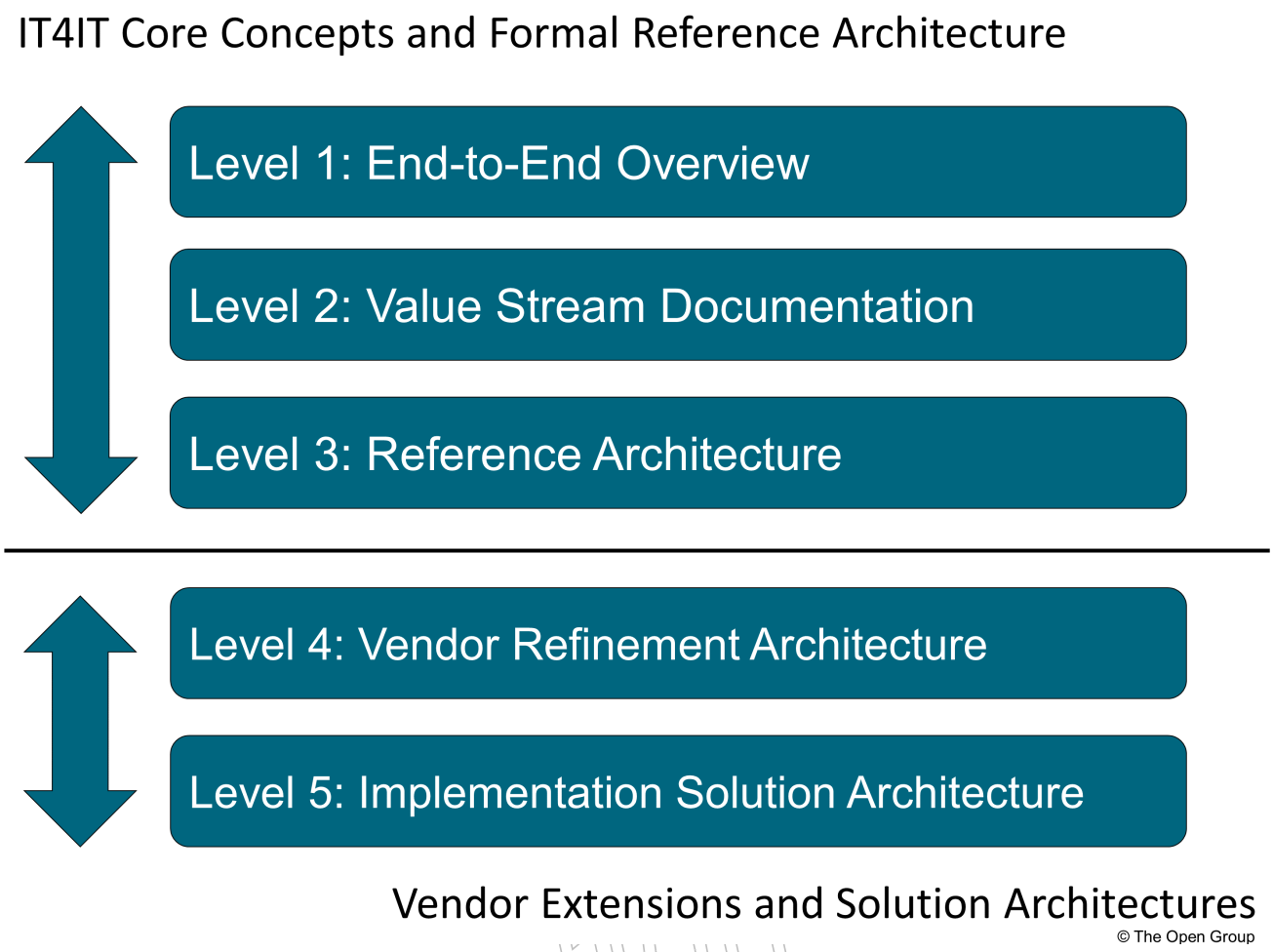

Figure 11-2. IT4IT Abstraction Levels

11.3. Level 1

Level 1 introduces the core concepts that make up the IT4IT Reference Architecture. These include:

- The seven IT4IT Value Streams
- The four high-level functional groups (plus the Supporting Functions) needed to manage the world of Digital
- The notion of a Digital Product
- The IT4IT concept of key data objects that control the lifecycle of Digital Products
- The underpinning functional components that control the key data objects

This is documented in Digital Management.

Following from this introduction, the concept of a Digital Product is defined in detail. Level 1 also introduces the concepts of the Product and Service Offer Backbone. This is documented in Digital Product.

11.4. Level 2

At this level, the seven essential IT4IT Value Streams are defined as part of the normative standard. This is documented in IT4IT Value Streams.

This chapter also introduces the concepts of:

- Value stream stages, as a refinement of value streams
- Stakeholders

11.5. Level 3

At this level, the IT4IT Functional Components and data flow across them, the IT4IT Data Objects and their relations are defined as part of the normative standard.

This is structured into five chapters, one for each high-level functional group:

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

11.6. Formal Reference Architecture Model

Levels 2 and 3 of the IT4IT Standard constitute a formal reference architecture. In addition to being documented in the chapters referenced above, it is maintained as an ArchiMate® Extensible Markup Language (XML) model in the ArchiMate model exchange file format.

The metamodel and associated definition of IT4IT terms are documented in IT4IT Concepts and Metamodel.

11.7. 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 foundational controls an IT organization needs in place to standardize, automate, and generally manage the IT4IT Value Streams.

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

- Value Network
- IT4IT Value Streams
- Management capabilities
- Functional components
- Data objects
  - Digital Product Backbone data objects
  - Service Offer Backbone data objects
- Data relationships

11.7.1. Value Network

A Value Network is a construct that defines the relationships across the value-creating functions that deliver value.

Within the IT4IT framework it is used to describe the model of the digital business function. It includes primary activities such as planning, production, consumption, fulfillment, and support. It also includes supporting activities such as Finance, HR, Governance, and Supplier Management.

NOTE The Value Network concept is an evolution of the Value Chain concept from Michael Porter's book *Competitive Advantage: Creating and Sustaining Superior Performance* [Porter 1998].

The Value Network concept breaks with the waterfall-oriented aspect of the original Value Chain construct and makes each value-creating activity both consuming and value-producing to potentially many other activities.

11.7.2. Value Stream

A value stream describes the value-creating activities for discrete areas within the Value Network where some unit of net value is created or added to the Digital Product as it progresses through its lifecycle. The IT4IT Reference Architecture defines seven essential value streams for digital management. These are referred to as the IT4IT Value Streams; see IT4IT Value Streams Model.

Notation and Naming

The IT4IT Standard has both a simplified notation and a notation aligned with the ArchiMate language, as pictured in Figure 11-3.

As a general rule, the IT4IT Value Streams are named using:

- A verb that describes the essence of what the value stream is doing

Note that occasionally the verb can be used in connection with the essential (backbone) data object that the value stream works on. This is only in the description, and not the normative name. Also note that this document breaks with the tradition of using a x2y notation like “Detect to Correct”. This brings it more in line with how similar value streams are named in some Agile frameworks. Finally note that the value streams are not prepended with the word “continuous”, which is often found in Agile text books. The prepending of "continuous" is considered a maturity construct and should not be part of the name.

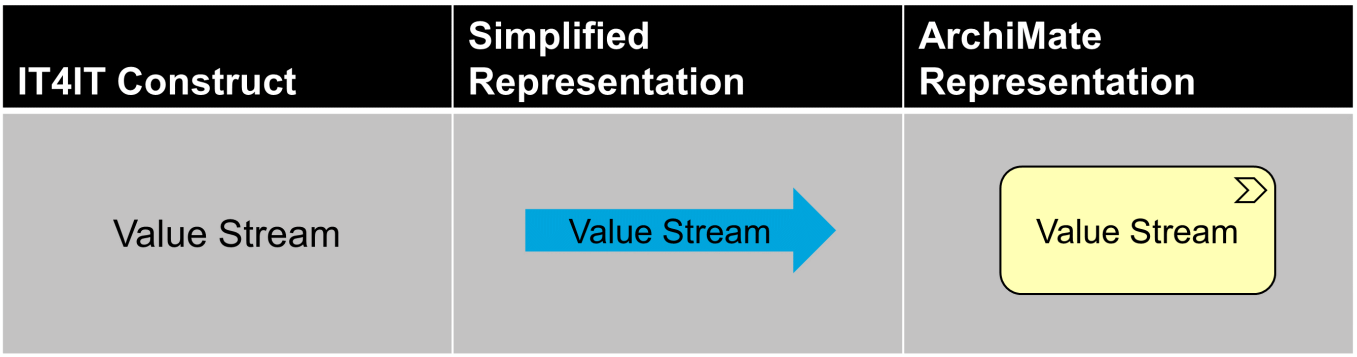

Figure 11-3. Value Stream Notation

11.7.3. Functional Groups

The IT4IT Reference Architecture uses the ArchiMate grouping construct as a way of grouping the functional components and data objects together to provide context for the functionality that organizations must deliver.

This can be used as a basis for capability planning in an organization but the standard does not prescribe what capabilities a Digital Organizations should build.

The normative IT4IT Standard defines a grouping at two levels:

- Level 1: five high level groups of functionalities
- Level 2: further decomposition into 14 groups

At Level 1, the IT4IT Standard defines the following functional groups:

- Strategy to Portfolio (Plan)
- Requirement to Deploy (Build)
- Request to Fulfill (Deliver)
- Detect to Correct (Run)

As well as the Support functions that are referenced but are not normative definitions in the IT4IT Standard.

Each one of these is then further decomposed at Level 2 to provide more structure to the reference architecture.

Note that the groupings are simply a way of grouping related functional components together to be able to talk about functionality in a more abstract way, such as Support functions, Assure functions, or simply Run functions. When the IT4IT Reference Architecture is used to implement digital management in a real organization it is necessary to also do capability mapping, as it remains an important activity for organizations. Detailed capabilities that would include process and people aspects are 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. Figure 11-4 gives an example of a possible decomposition of the IT4IT normative Level 1 and Level 2 groups into detailed capabilities.

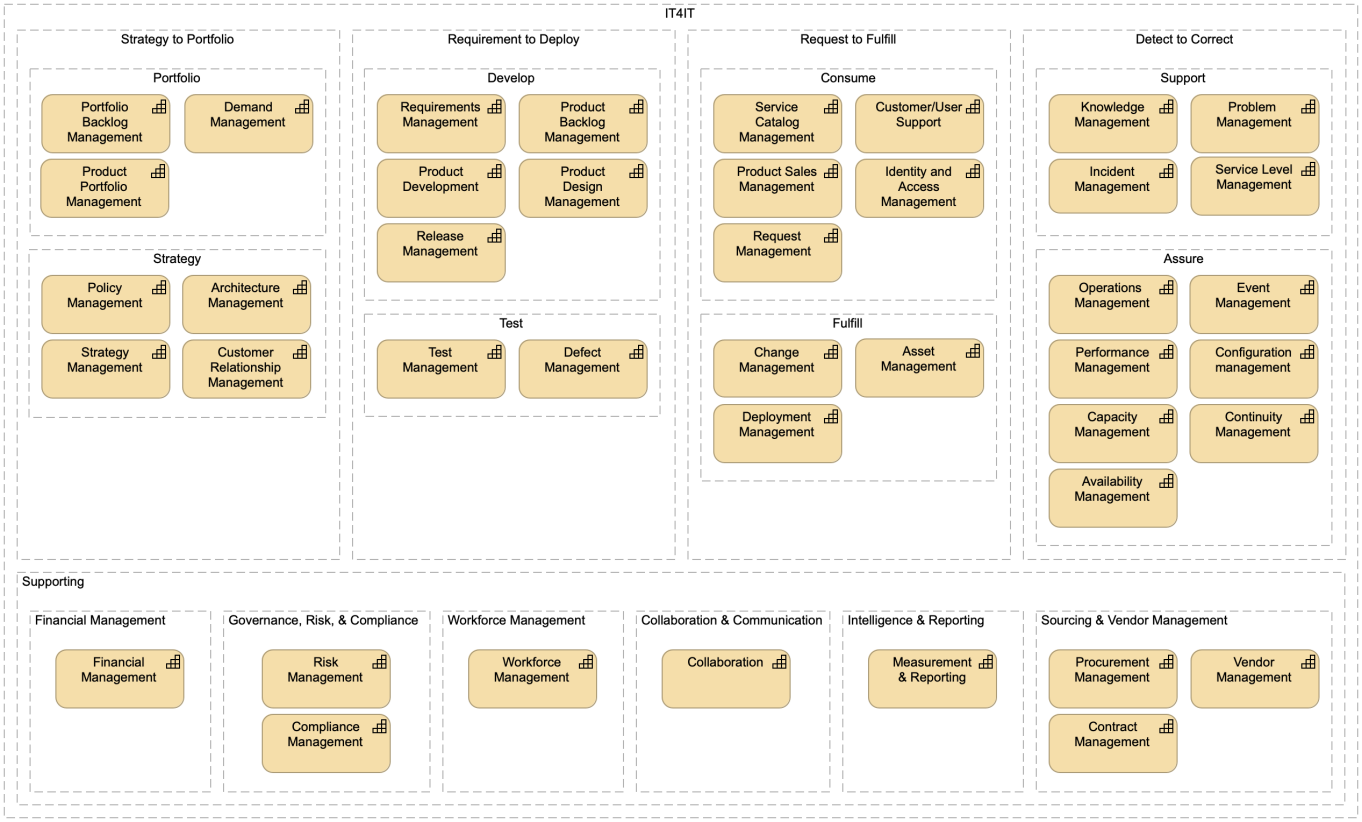

Figure 11-4. Example of Level 1, Level 2 Groups with a Possible Capability Decomposition

There is also Supporting Functionality within the Value Network model, such as Supplier & Vendor Management, Workforce Management, and Financial Management. As with detailed capabilities, these do not define the Supporting Functions in details, but do define some of the essential data objects and functional components that are expected from the Supporting Functions in order for the primary functionality and value streams to work.

Notation and Naming

The IT4IT Standard uses the ArchiMate grouping types and a containment notation to indicate decomposition and to illustrate which functional components support a given functional group.

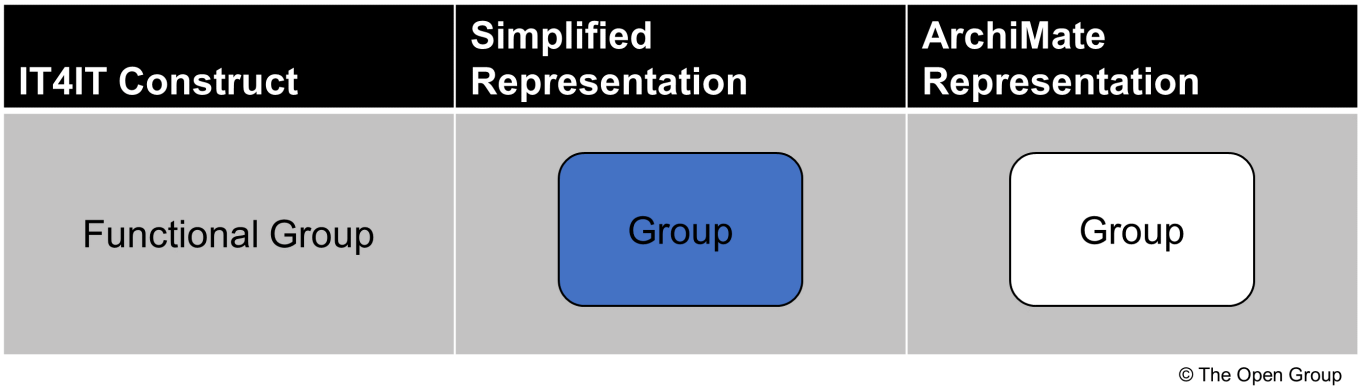

Figure 11-5. Functional Group Notation

NOTE Figure 11-4 illustrates a number of capabilities. This is for illustration, and not normative. The IT4IT Standard only defines a general grouping of functionalities and is not defining a formal capability model.

As a general rule, the IT4IT Functional Groups are named using:

- A verb that describes the essence of what the group of functions is performing

11.7.4. Functional Component

A functional component is the smallest unit of technology that can stand on its own and be useful as a whole to a Digital Practitioner (or IT organization). It has defined input(s) and output(s) that are data objects and has an impact on a key data object that it controls. Typically, a functional component controls and/or manages a single key data object, but this is not dictated by the architecture; it can control several, especially if they are intrinsically connected.

Examples of functional components include Event, Product Backlog, Defect. 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 used by IT organizations in the normal course of business, but that are not considered part of the prescriptive IT4IT Reference Architecture. Examples of these types of components could include Corporate Finance systems, HR applications, and Contract Management systems. These will be supplied by the Supporting Capabilities. These are not normative but in some cases the reference architecture will make reference to those.

Notation and Naming

The IT4IT Reference Architecture uses both a simplified and an ArchiMate notation style. At Level 1, the informal notation renders primary functional components as blue rectangles (see Figure 11-6). Secondary functional components are rendered as gray rectangles. Functional components are represented formally in the ArchiMate language using the “Application Function” type.

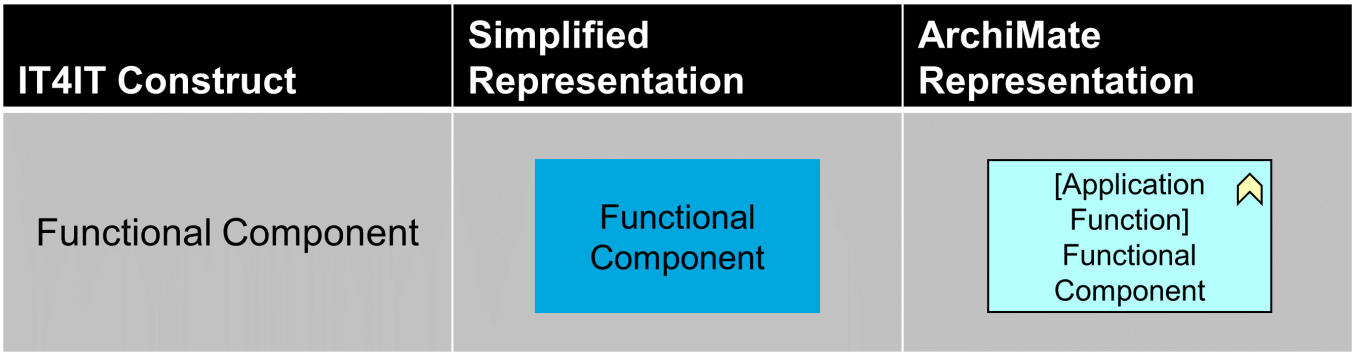

Figure 11-6. Functional Component Notation

As a general rule, the IT4IT Functional Components are named using:

- A noun or a noun with a qualifier that describes the essence of what the functional component is managing

Occasionally the name is appended with the word “component” as a shorthand for “functional component”.

11.7.5. Key Data Object

A Key Data Object represents data (records, information, and so on) that annotate or model an aspect of a Digital Product being planned, developed, offered, or consumed. Data objects can take a digital or physical form and can be comprised of structured, semi-structured, or unstructured data. Our definition of Key Data Object is aligned contextually with the Object Management Group® (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.” [UML]

Examples of data objects include incident records, training videos, requirements documents. These types of data objects contribute in some way to define and control a Digital Product over the course of its lifecycle. There are also “special” data objects that represent an abstraction or view of a specific product; for example, a system diagram, release package, or a binary executable file; this provides a “view” of a product to various personas at different stages of the product lifecycle. These special data objects and their relationships form a “backbone” that binds together all of the information needed to offer and manage a Digital Product, thus they are referred to as Digital Product Backbone data objects.

The central focus of the IT4IT Forum has been to identify the data objects that play a “key” role in the product lifecycle. In other words, the reference architecture focuses exclusively on the data objects that are mandatory for ensuring end-to-end traceability of the product lifecycle operationally and/or financially. There are other data objects used across the digital landscape 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

The IT4IT Reference Architecture utilizes both a simplified and an ArchiMate notation for data objects so that the work can be easily understood by both architects and non-architects. In the Level 1 model, a simplified notation that depicts data objects as circles is used. Color is used to specify whether a data object is key data object (black) or a backbone data object (orange). Data objects are modeled formally in the ArchiMate language using the "Data Object" type, as shown in Figure 11-7.

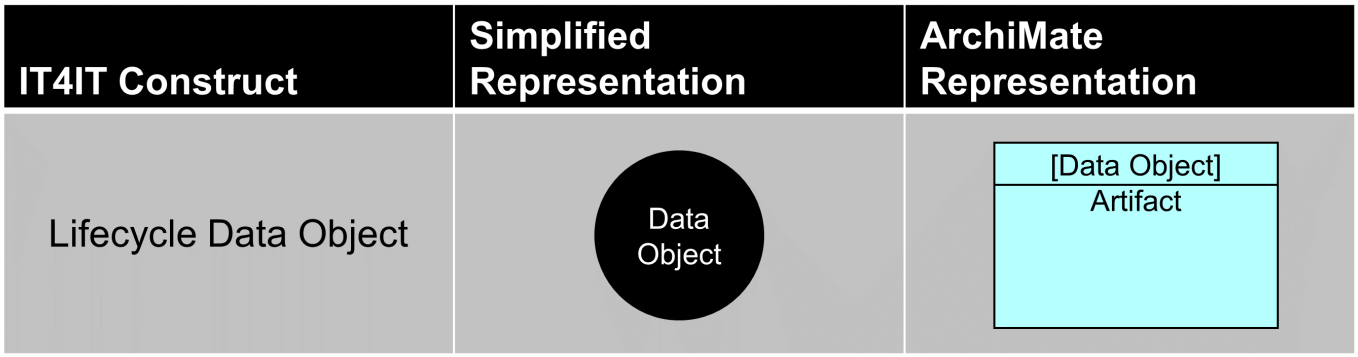

Figure 11-7. Key Data Object Notation

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.

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, a notation that embeds the "controlled" data object in the functional component is used; see Figure 11-8.

Diagrams can be complex and require clear "visual syntax" to help the reader interpret the elements and their relationships. For that reason, the IT4IT Reference Architecture at Level 1 is depicted without data flows and a reduced set of data relationships; see The IT4IT Level 1 Functional Diagram.

As a general rule, the IT4IT Data Objects are named using:

- A noun that describes the essence of what the data object is controlling

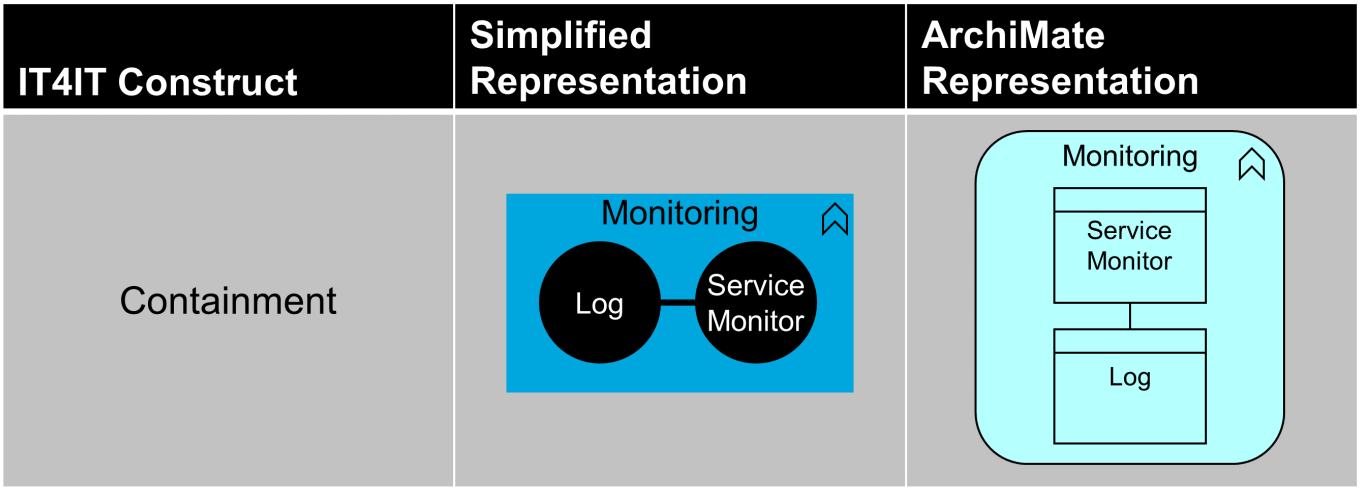

Figure 11-8. Functional Component Data Object Notation

11.7.6. System of Record

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

The key data objects represent the authoritative source data and/or master data for managing digital. In IT, “system of record” is a term that is commonly used as a synonym for an authoritative source system.

11.7.7. Relationships

An important aspect of the IT4IT Reference Architecture is to define 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 digital management (see Figure 11-9 and Figure 11-10).

Note that in this document there are no defined “reflective” relationships, where a data object can relate to other data objects of its own type; for instance, the Digital Product data object relates to all the Digital Products on which it depends, the Service Offers can represent a hierarchy of offers, or Events can relate to Events to represent correlation. A future version of the IT4IT Standard will include these relations.

Notation and Naming

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

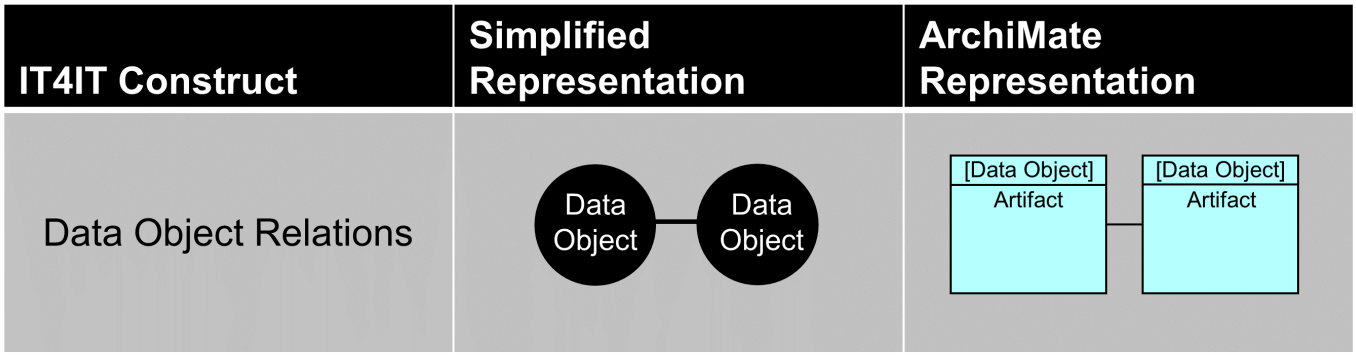

Figure 11-9. Relationship Notation

Compliance with the prescribed system of record relationship mapping will ensure end-to-end traceability operationally and financially, and ensure Digital Product 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.

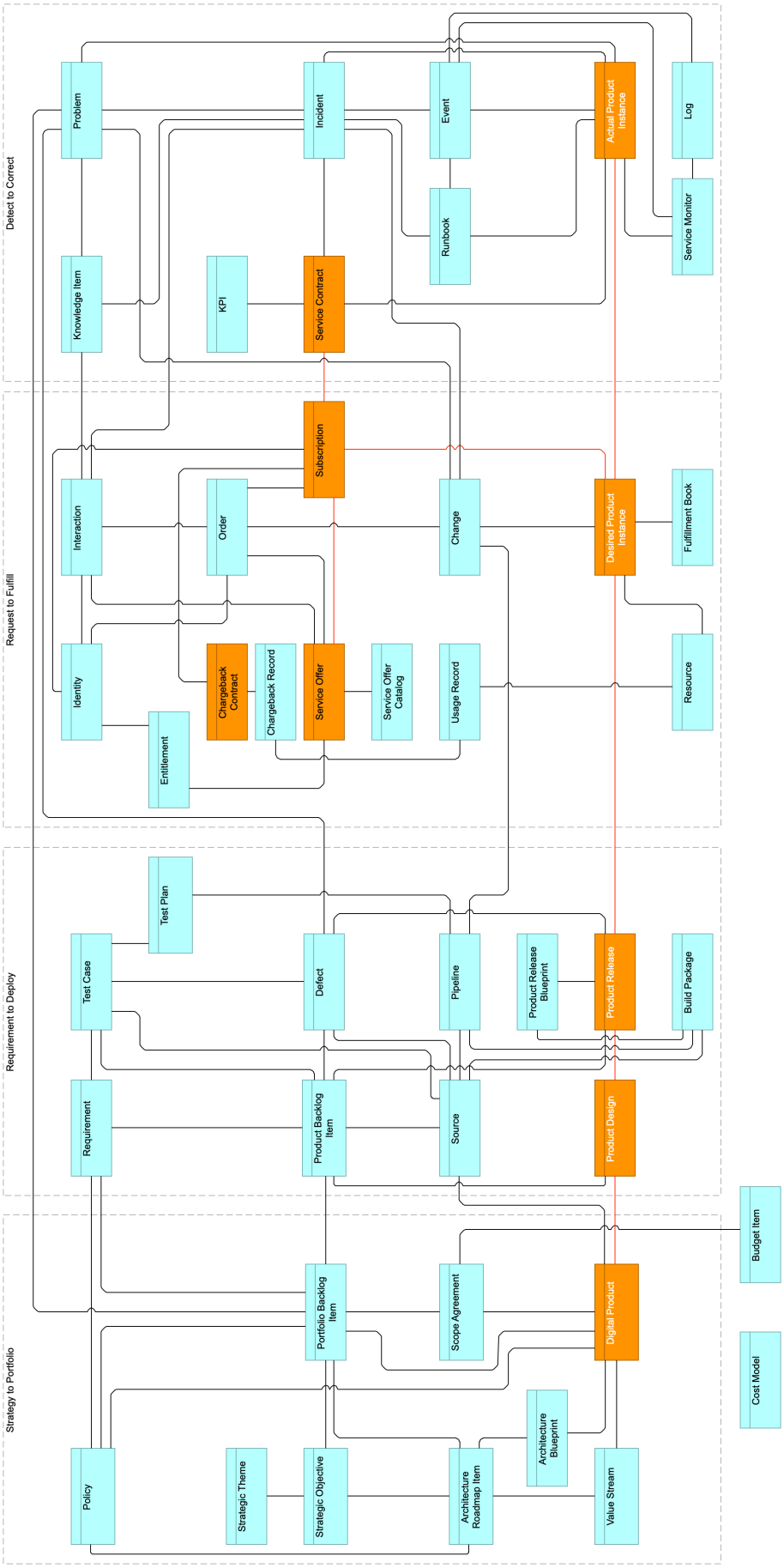

Figure 11-10. IT4IT End-to-End Traceability (System of Record Fabric)

11.7.8. Digital Product Backbone Data Objects

The Digital Product Backbone data objects are key data objects in the IT4IT Reference Architecture that annotates an aspect of the Digital Product model in its planning, development, consumable, or running state. These data objects and their relationships form the Digital Product Backbone, which provides a holistic view of a Digital Product and includes the Digital Product, Product Design, Product Release, Desired Product Instance, and Actual Product Instance.

The IT4IT Standard collectively refers to all Digital Product Backbone data objects as the “Digital Product Backbone”.

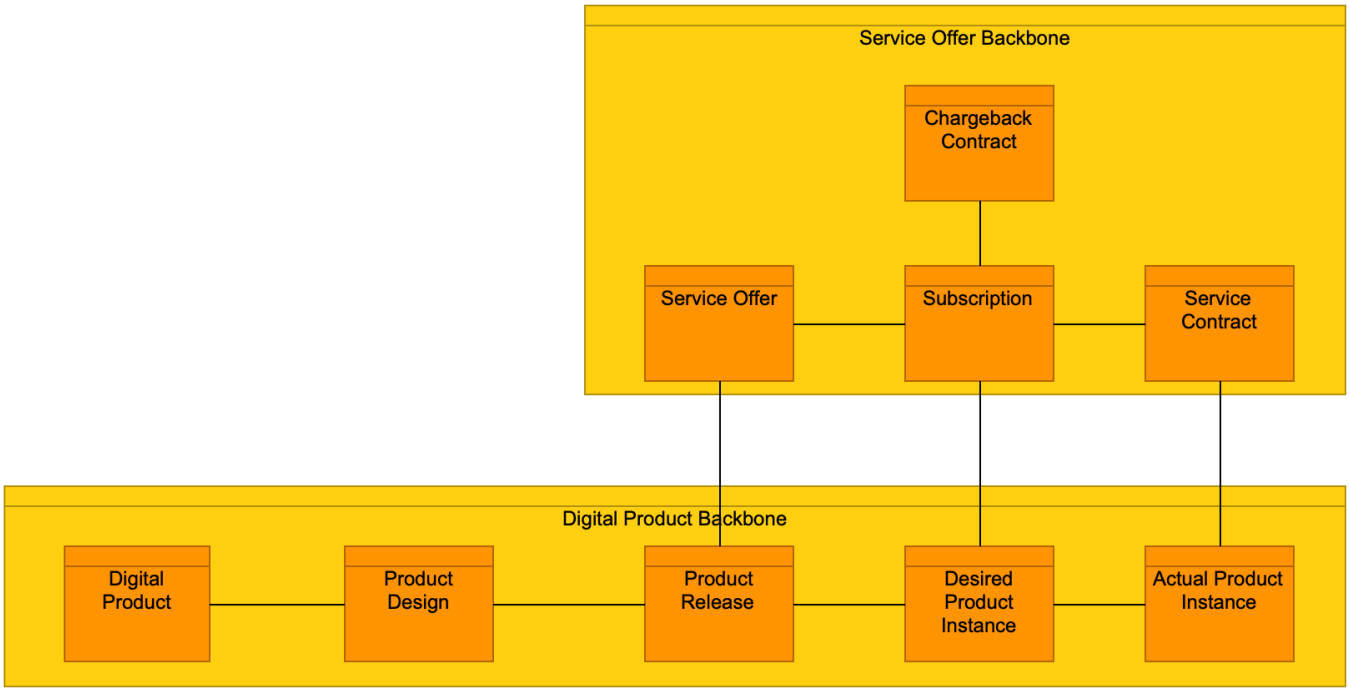

Figure 11-11. IT4IT Digital Product Backbone and Service Offer Backbone Data Objects

11.7.9. Service Offer Backbone Data Objects

The Service Offer Backbone data objects are key data objects in the IT4IT Reference Architecture that control and define the consumption of a Digital Product. These data objects and their relationships form the Service Offer Backbone and include the Offer, Subscription, Chargeback Contract, and Service Contract data objects.

The IT4IT Standard collectively refers to to all the Service Offer Backbone objects as the “Service Offer Backbone”; see Figure 11-11. Note that Figure 11-11 is an abstraction and the relation between Service Offer and Product Release is via the Product Release Blueprint.

11.7.10. Level 1 ArchiMate Model

With the above definitions of the simplified functional component, data object, and ArchiMate notation, IT4IT Level 1 can be visualized in Figure 11-12, which can be compared to the simplified notation in The IT4IT Level 1 Functional Diagram.

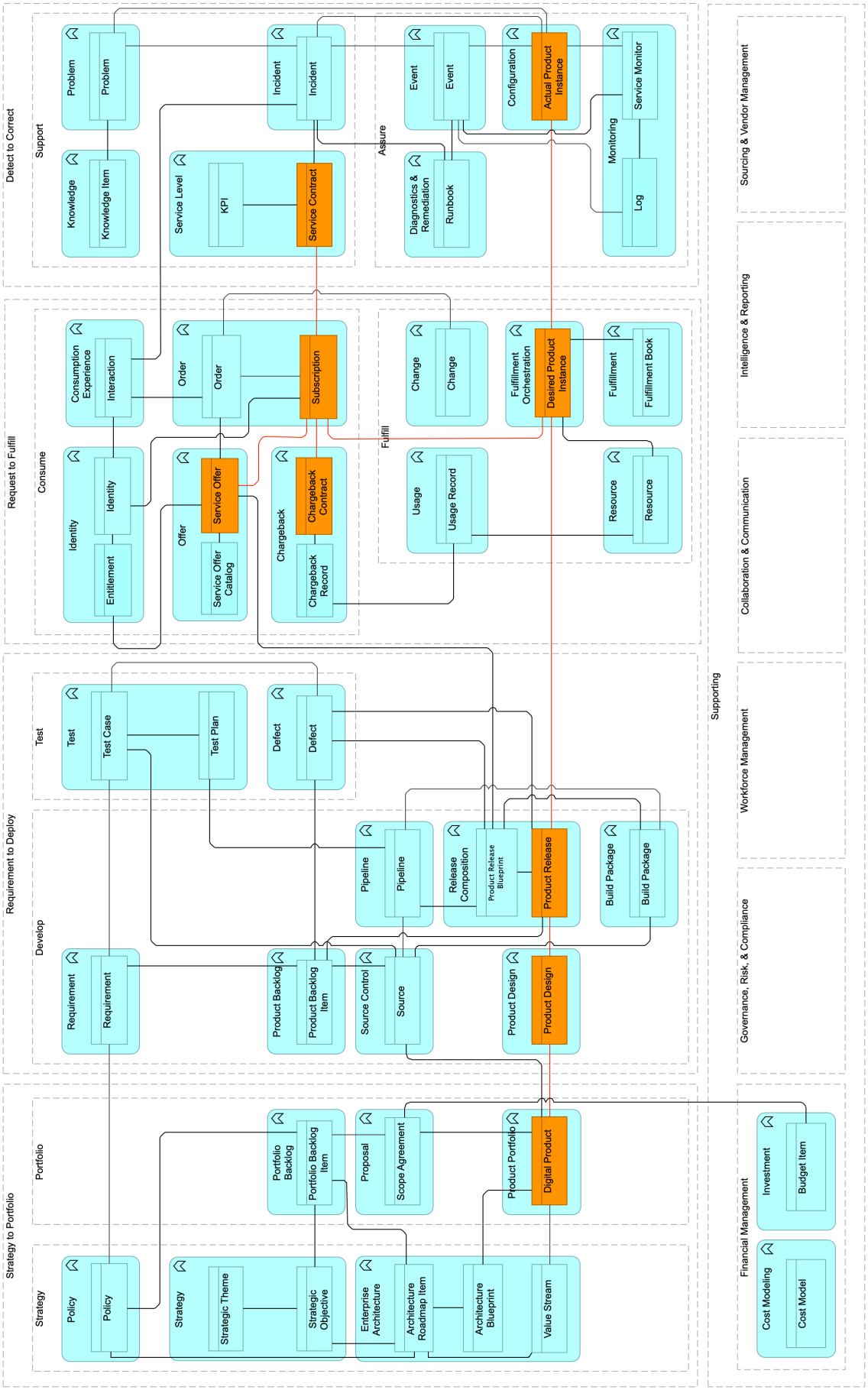

Figure 11-12. IT4IT Level 1 in ArchiMate Notation

Note that the Level 1 figures are abstractions, created for easy readability. This implies that not all the relations of the formal model are shown. For all the IT4IT data object relations, see the diagrams in the definition of the functional groups: Strategy to Portfolio Functions Model, Requirement to Deploy Functions Model, Request to Fulfill Functions Model, Detect to Correct Functions Model.

11.8. Concepts at Level 2: Value Stream Documentation

The value stream concept is introduced at Level 1, but is defined in detail at Level 2. In order to define value streams, the following concepts are introduced:

- Scenarios
- Value stream stages
- Stakeholders

Further, the normative definition of the IT4IT Value Streams refers to data objects, which are introduced and defined at Level 1.

11.8.1. Value Stream

In the normative definition of value streams, scenarios are defined for which the value stream is executed. This is complemented by defining the activities that are expected to happen as part of implementing the value stream; documented as value stream stages.

Note that the value stream stages of an IT4IT Value Stream are not considered to be a sequence of activities; value streams are abstract, and not a process description.

11.8.2. Scenario

A scenario defines a situation in which a given value stream should be triggered to add value to the overall Value Network.

The ArchiMate solution pattern for this is an “Outcome” motivational element.

11.8.3. Value Stream Stage

A value stream stage is a refinement of the value stream, and uses the same ArchiMate solution pattern: a Value Stream data object.

A value stream stage is defined in terms of

- Entry and exit criteria
- Main activities
- Examples of participating stakeholders
- Participating data objects and containing functions

11.8.4. Stakeholder

A stakeholder is a persona or role that an organization is expected to engage in a given value stream stage.

NOTE The IT4IT Standard does not define the needed stakeholders, but merely proposes possible stakeholders to engage. Stakeholders must be refined when the reference architecture is used to derive a concrete implementation architecture at Level 5.

The ArchiMate solution pattern for this is a “Business Stakeholder” motivational element.

This leads to the ArchiMate structure of the seven IT4IT Value Streams in Figure 11-13. For more detail, see the definitions in IT4IT Value Streams.

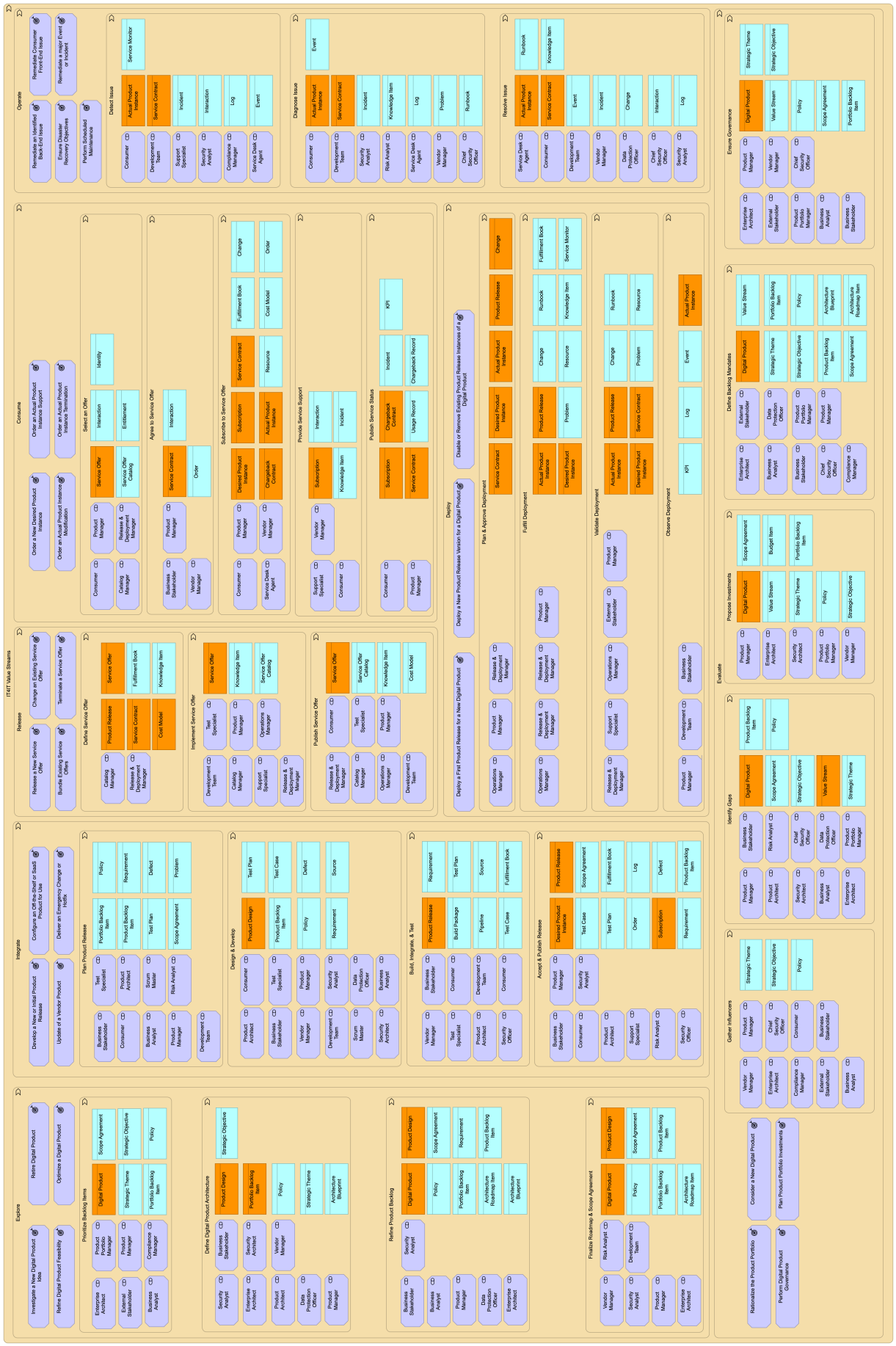

Figure 11-13. IT4IT Value Streams, Scenarios, Stages, Stakeholders, and Data Objects

11.9. Concepts at Level 3: Vendor-Independent Architecture

The normative definition of the IT4IT Value Streams at the Level 2 abstraction is complemented by the normative definition at Level 3 of:

- Functional components
- Data objects
- Object relations

All of which have already been introduced at Level 1.

The normative definition of the functional components and data objects introduces a few more concepts:

- Data objects including key attributes
- Relationships between data objects are updated with cardinality information (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 grouped into record integrations and engagement integrations

11.9.1. Key Attributes

As described in Levels 1 and 2, data objects represent the data exchanged between functional components. At the Level 3 abstraction 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 required to maintain the integrity of the system of record fabric is defined. These attributes 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 should be included is defined.

Note that the specification of a key attribute for a data object only implies that the reference architecture insists that an implementation must support the attributes. It does not specify that every instance of the data object type must have a value associated with the attribute. For example, an approval data attribute will only be filled out if the approval has happened.

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

Notation and Naming

The ArchiMate language does not have a means to model the concept of attributes. The IT4IT Standard documents the attributes in the description of the individual key data objects. Attributes are documented as a list of name/description parts.

The naming of attributes is without prepended or appended data to the object name. So, in the Incident

example above, the IT4IT Standard does *not* define the identifier attribute for the Incident object to be “IncidentId” or the name attribute to be “IncidentName”. It simply has attribute names such as “Id” and “Name”. If there is a need to specify the Identity (Id) of a particular data object, then the usual dot notation common in database systems is used:

- The “Id” of the Incident data object can be referred to as “Incident.Id”

The IT4IT Standard aims to reuse attribute names that denote the same thing across data object definitions to facilitate the ease of implanting systems using inheritance and to ease the management of different objects in a coherent way.

11.9.2. Cardinality

Cardinality (sometimes referred to as “multiplicity”) describes the number of allowable relations a given element can have with elements of a given other type.

For example, there is a one-to-n (1:n) relationship between “Order” and “Change” data objects. This indicates that a single order can result in multiple change requests. In the instance of a “laptop” Order, a change request might require a request for the laptop to be allocated and configured and a further change for a corporate user account to be security configured for mobile work. Therefore, one Order generated multiple Change objects, and relations must be maintained to each.

The IT4IT Reference Architecture only defines the key relationships – those that contribute to the advancement of the Digital Product 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. Further, relationships may be maintained simply to optimize the implementation; for example, the Actual Product Instance might have a relationship back to the Digital Product data object, of which it is an instance. This relationship could be derived through the relationships across the Digital Product Backbone. The normative IT4IT Standard does not make recommendations for such optimizations.

In UML, the concept of multiplicity extends the definition/understanding/concept of cardinality by including more information on participation; i.e., what relationships must exist, or not. The IT4IT Standard does not specify multiplicity.

The IT4IT Standard only differentiates between three types of cardinality, as shown in Table 11-1.

Table 11-1. Three Types of Cardinality in the IT4IT Standard

IT4IT Cardinality	Representation	Notes
One-to-no more than one (also called one-to-one)	1:1	Means “can” have a relationship. “Must” have a relationship is not specified. In UML this would be a 0..1 : 0..1 relation.
One-to-many	1:n or n:1	As above, there might not be a relationship or there might be a relationship to only one. In UML this would be a 0..1 : * relation.
Many-to-many	n:m	Both objects in a relationship “can” have relations to several other objects. In UML this would be a * .. * relation.

Notation and Naming

The ArchiMate language does not allow the specification of cardinality in relationships, which means that cardinality is not represented in diagrams in the normative IT4IT Standard. To resolve this, data object relationships are documented in the description of the relationship in a section named “Key Data Object Relationships”.

It lists all the relations and for each in the following form:

- <DataObject> to <DataObject> (x:y): <description of why the relation is created>

This gives the four possible notations, illustrated with four examples from the standard:

- Problem to Defect (1:1): indicates that an identified Problem can be the source of one Defect, and similarly a Defect can be related to a Problem (but not multiple Problems)
- Incident to Change (1:n): indicates that a single Incident can relate to several Change objects (indicating that potentially more than one Change was done in order to remediate the Incident)

  On the other hand, a given Change object can only relate to a single Incident; e.g., you can track which Incident was the reason for the Change to be created

- Change to Incident (n:1): the opposite of the above when documenting the Change data object as opposed to documenting the Incident data object
- Incident to Event (n:m): indicates that any given Event can relate to several Incidents (indicating that several reported Incidents are really all related to the same operational Event), and each Incident can relate to several Events (to indicate that multiple Events are occurring as a consequence of the same Incident)

11.9.3. 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 3 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.

NOTE Data flows do not represent process. Data flows represent core integrations between components that ensure data between modules is synchronized to ensure the information model is maintained.

The data flow arrow originates from the functional component that would normally trigger the exchange of information. This does not imply that information is only exchanged in the direction of the arrow.

An example of a data flow can be seen in Figure 11-14.

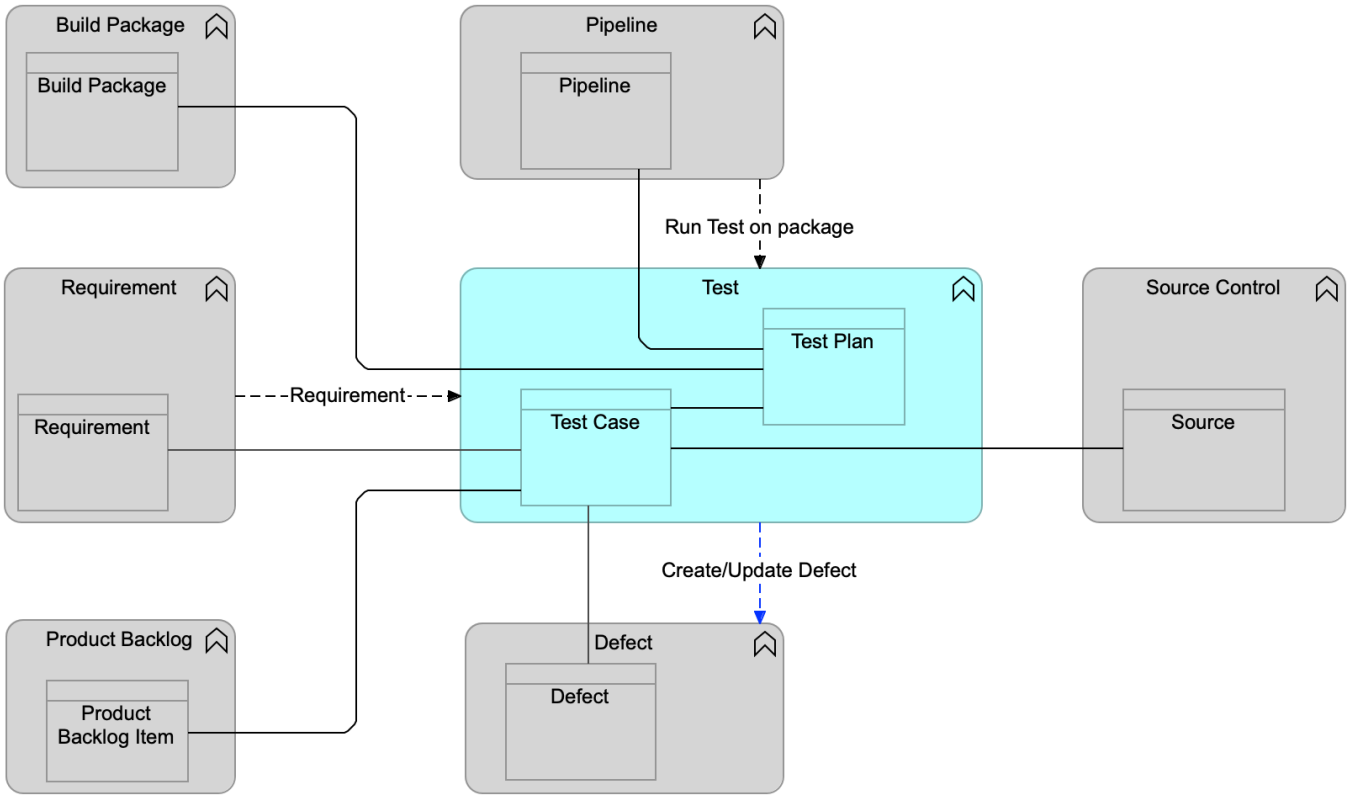

Figure 11-14. Data Flow Notation

Figure 11-14 indicates a data flow of requirement data from the Requirement functional component to the Test functional component. There is also a data flow of test result data from the Test functional component to the Defect functional component. These two flows are used to maintain the relationships between the corresponding data objects.

Finally, there is a data flow from the Pipeline functional component to the Test functional component indicating that the Pipeline functional component engages with the Test functional component. The

data flow indicates that a specific test needs to be triggered for execution.

NOTE In diagrams for a specific functional component, like Test in Figure 11-14, we only show data flow in and out of that component. For instance, there is a data flow between Pipeline and Build Package, which is not shown in this diagram but can be seen in Build Package Functional Component Model and Pipeline Functional Component Model where Build Package and Pipeline is documented.

As is seen in Figure 11-14, an arrow is used for representing data flows. As indicated, there are two types of data flow:

- System of record data flows: these establish system of record integrations and are essential for maintaining a consistent and traceable information model for digital management

  Typically indicating that the data object in the respective components are interlocked

- System of engagement data flows: these represent integrations that are useful for efficiently implementing the value streams that orchestrate the use of many of the IT4IT components

  In the above example the data flow from Test to Defect is a system of engagement data flow.

System of engagement data flows are marked in blue, as opposed to system of record data flows that are black.

11.9.4. System of Record Integration

In Level 1, key data objects and their relationships are introduced. In Level 2, how data objects are consumed or created/updated by the value streams is described. In Level 3, all the necessary key data objects and relationships are defined to form a consistent and traceable information model for managing Digital Products.

In order for the system of record relationships to be sustained over the course of the product lifecycle, integrations between functional components controlling the key data objects must be well defined. This requires that some data flows need to be “refined” into system of record integration specifications. These specifications become part of the normative IT4IT Standard.

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; 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 11-15. This figure provides an illustrative example that describes the state dependency between the “Event” and “Incident” data objects. Here, Events generate Incidents, which results in a relationship and dependency between these two data objects.

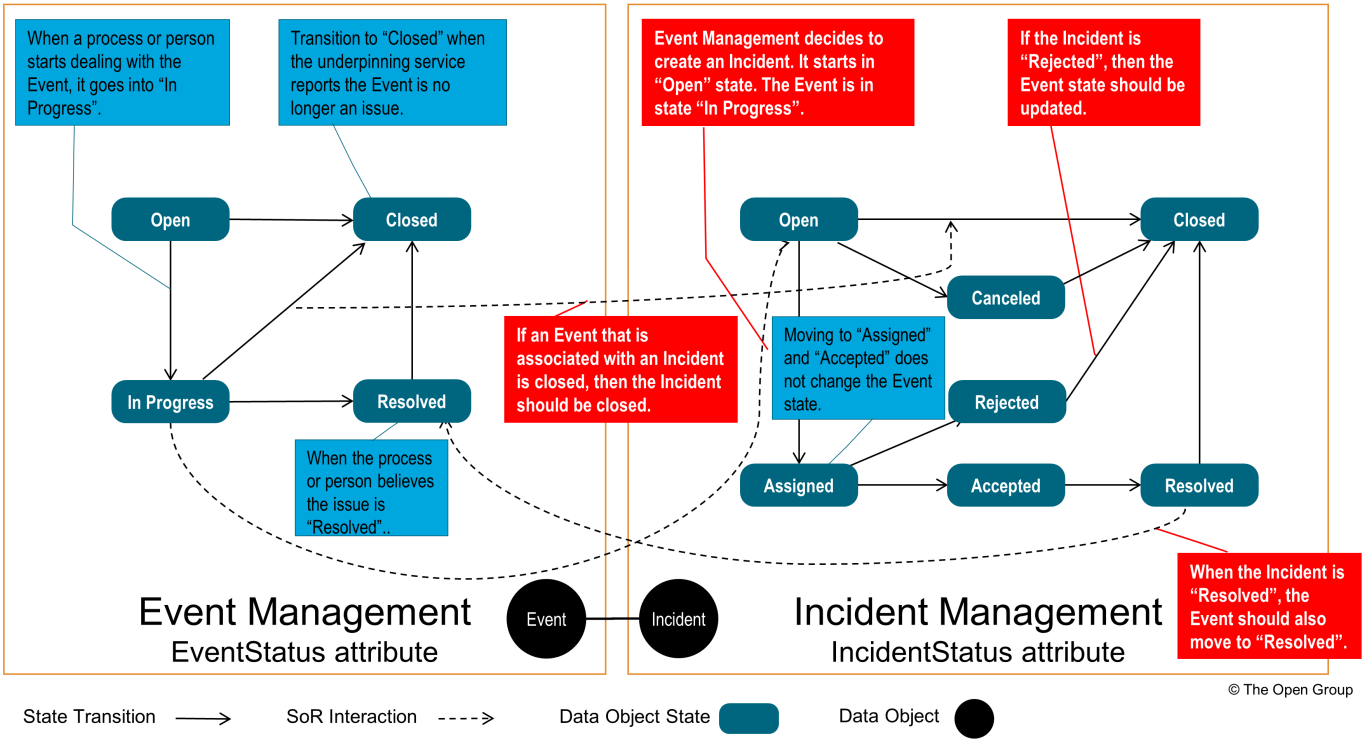

Figure 11-15. Data Object State Model Dependency Illustration

11.9.5. System of Engagement 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) and are derived from value stream use-cases and user stories. They are offered as guidance rather than prescriptive practices; the intent being to demonstrate to the reader how data object relationships stimulate and/or are affected by human-to-machine and machine-to-machine interactions.

For example, the Incident to Knowledge system of engagement integration “Search Knowledge” (see Incident Functional Component Model) indicates that a common use-case for managing an incident is to look up relevant Knowledge Items based on the Incident information and potentially create a link to the most relevant ones. But unlike a system of record integration we do not specify that there is a long-lived maintained relationship, such as when an incident is updated then the Knowledge Item needs to be updated, or that updates to Knowledge Items should result in updates to the related Incidents.

Notation and Naming

At Level 3, the normative notation depicts system of record integrations using a black dotted line with a 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 use blue dotted lines with a solid end-arrow, and a name that describes the type of engagement or process action responsible for triggering the integration/data flow.

11.10. Concepts at Level 4 and Level 5

Level 4 and Level 5 of an IT4IT architecture are not part of the normative standard.

Level 4 is used to represent reference architectures built by vendors and system integrators as a refinement of the normative IT4IT Standard.

Level 5 is used to refine Level 4 (or Level 3) reference architectures into a concrete implementation architecture.

Levels 4 and 5 are not owned and controlled by The Open Group but by the consumers of the IT4IT Standard.

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 *add to*, but not change the prescriptive model defined in Levels 1 to 3.

NOTE In The Open Group IT4IT Forum, some examples of Level 4 and Level 5 architectures have been developed as guiding material.

11.10.1. Level 4: Vendor and System Integrator Extensions

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 key 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.

11.10.2. Capabilities

A capability is the ability of an organization to produce an outcome of value through the utilization of people, processes, and technology. A list of capabilities with a high level of granularity is being compiled. These come from ITIL, COBIT, SAFe, PMBOK, and other industry sources. It is not the focus or intent of this document to redefine or modify existing work that has been done around such IT capabilities, but, as a non-normative work, a comprehensive list is being compiled that defines the relationships to functional components.

Capabilities can be viewed as a refinement of the Level 2 normative functional groups of the IT4IT Reference Architecture. In the solution pattern/metamodel, the functional components belong to a Level 2 group but will relate to one or several capabilities.

Note that in the IT4IT Standard, Version 2.1, and other previous IT4IT Standard versions, capabilities were named “Capability Disciplines”.

Notation and Naming

While capabilities are not part of the normative IT4IT Reference Architecture, we recommend to use the ArchiMate notation to describe them for consistency across the architecture.

11.10.3. 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., Create, Read, Update, Delete (CRUD). Essential services are typically implemented as an API, web service, or microservice 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, for example, the attributes, parameters.

These essential services will be used to implement the data flows and the processes that are deemed necessary to deliver optimized value streams.

Notation and Naming

Essential services can be modeled in the ArchiMate language using the “Application Service” construct. In the informal notation at Level 4 they are depicted as an oval inside a functional component element, as in the scenario example given in Figure 11-16.

11.10.4. Scenarios and Processes

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 “epic” 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 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**: a high-level description of the scenario to be described along with the goal to be achieved
- **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 of the process flow in the organization; this example explains how an Incident is handled using existing knowledge

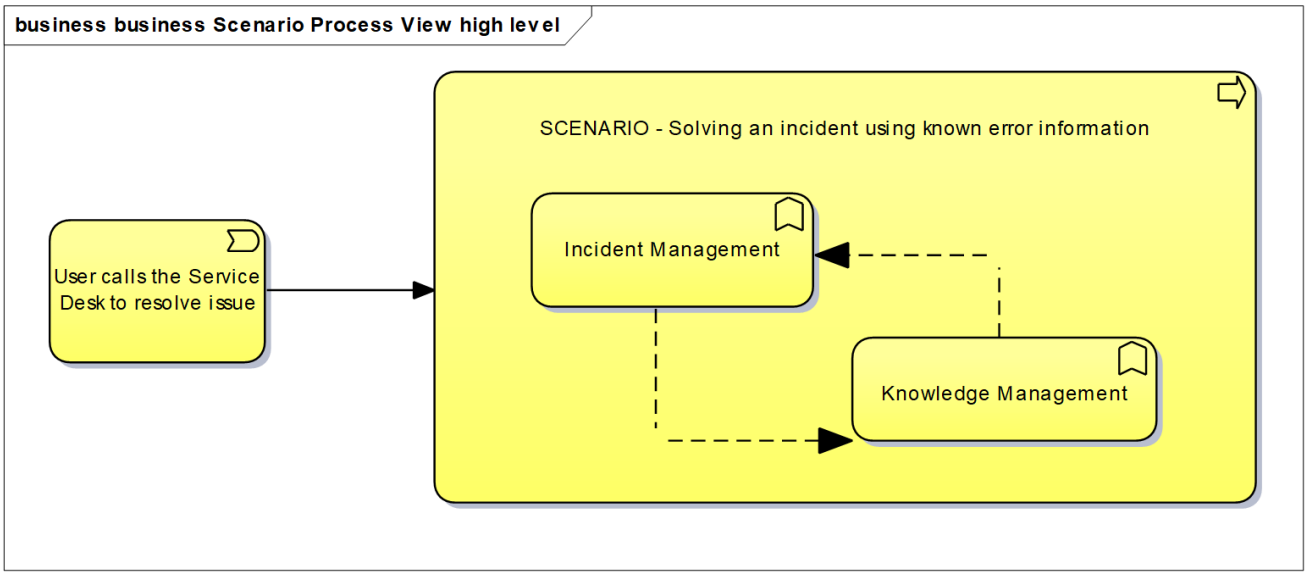

Figure 11-16. Scenario Process Flow Example Utilizing Two Essential Services

- **Automation specification using the reference architecture**: explains how the process is to be supported by the reference architecture

In the example shown in Figure 11-17, 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 Figure 11-16
- **Data objects and key attributes**: describes the data objects that are involved and the attributes of

the data objects that are needed or impacted

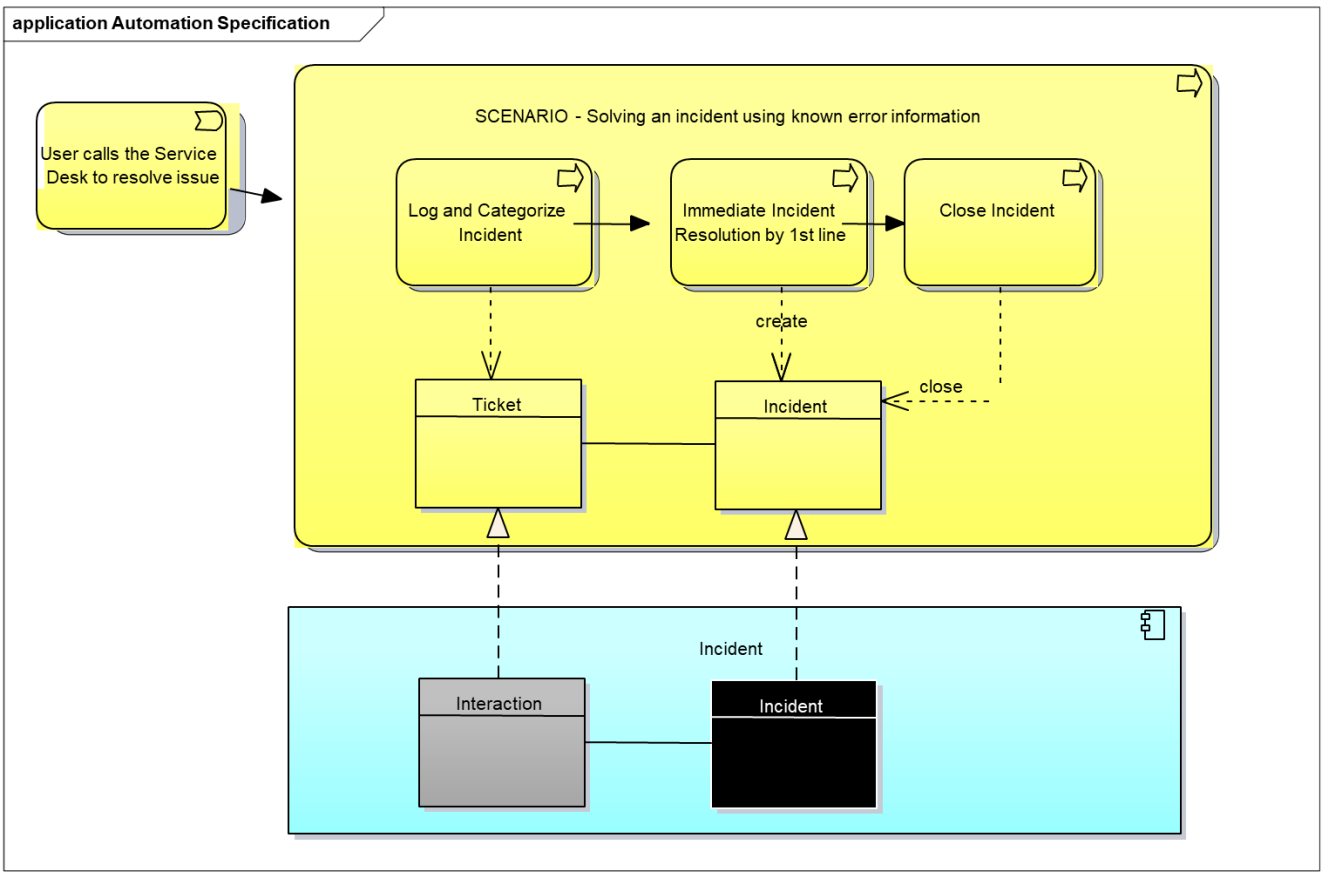

Figure 11-17. Essential Service Diagram Example

11.10.5. Level 5: Implementation Architecture

The full power of the ArchiMate language can be used to specify an actual current or future state implementation of the IT4IT Reference Architecture.

Appendix A: Value Stream – Functional Component – Data Object Tables

The following tables list Functional Components, Data Objects, and Value Streams alphabetically and with cross references.

A.1. Functional Components

Table A-1. Functional Components to Data Objects and Value Streams

Functional Component	Data Object	Value Streams
Build Package	Build Package	Integrate
Change	Change	Deploy, Consume, Operate
Chargeback	Chargeback Contract, Chargeback Record	Consume
Configuration	Actual Product Instance	Deploy, Consume, Operate
Consumption Experience	Interaction	Consume, Operate
Cost Modeling	Cost Model	Release, Consume
Defect	Defect	Integrate
Diagnostics & Remediation	Runbook	Deploy, Operate
Enterprise Architecture	Architecture Roadmap Item, Architecture Blueprint, Value Stream	Evaluate, Explore
Event	Event	Deploy, Operate
Fulfillment	Fulfillment Book	Integrate, Deploy, Release, Consume
Fulfillment Orchestration	Desired Product Instance	Integrate, Deploy, Consume
Identity	Identity, Entitlement	Consume
Incident	Incident	Consume, Operate
Investment	Budget Item	Evaluate
Knowledge	Knowledge Item	Deploy, Release, Consume, Operate
Monitoring	Service Monitor, Log	Integrate, Deploy, Operate
Offer	Service Offer Catalog, Service Offer	Release, Consume
Order	Order, Subscription	Integrate, Consume

Functional Component	Data Object	Value Streams
Pipeline	Pipeline	Integrate
Policy	Policy	Evaluate, Explore, Integrate
Portfolio Backlog	Portfolio Backlog Item	Evaluate, Explore, Integrate
Problem	Problem	Integrate, Deploy, Operate
Product Backlog	Product Backlog Item	Evaluate, Explore, Integrate
Product Design	Product Design	Explore, Integrate
Product Portfolio	Digital Product	Evaluate, Explore
Proposal	Scope Agreement	Evaluate, Explore, Integrate
Release Composition	Product Release, Product Release Blueprint	Integrate, Deploy, Release
Requirement	Requirement	Explore, Integrate
Resource	Resource	Deploy, Consume
Service Level	Service Contract, KPI	Deploy, Release, Consume, Operate
Source Control	Source	Integrate
Strategy	Strategic Theme, Strategic Objective	Evaluate, Explore
Test	Test Case, Test Plan	Integrate
Usage	Usage Record	Consume

A.2. Data Objects

Table A-2. Data Objects

Data Object	Functional Component	Value Streams
Actual Product Instance	Configuration	Deploy, Consume, Operate
Architecture Blueprint	Enterprise Architecture	Evaluate, Explore
Architecture Roadmap Item	Enterprise Architecture	Evaluate, Explore
Budget Item	Investment	Evaluate
Build Package	Build Package	Integrate
Change	Change	Deploy, Consume, Operate
Chargeback Contract	Chargeback	Consume
Chargeback Record	Chargeback	Consume
Cost Model	Cost Modeling	Release, Consume

Data Object	Functional Component	Value Streams
Defect	Defect	Integrate
Desired Product Instance	Fulfillment Orchestration	Integrate, Deploy, Consume
Digital Product	Product Portfolio	Evaluate, Explore
Entitlement	Identity	Consume
Event	Event	Deploy, Operate
Fulfillment Book	Fulfillment	Integrate, Deploy, Release, Consume
Identity	Identity	Consume
Incident	Incident	Consume, Operate
Interaction	Consumption Experience	Consume, Operate
KPI	Service Level	Deploy, Consume
Knowledge Item	Knowledge	Deploy, Release, Consume, Operate
Log	Monitoring	Integrate, Deploy, Operate
Order	Order	Integrate, Consume
Pipeline	Pipeline	Integrate
Policy	Policy	Evaluate, Explore, Integrate
Portfolio Backlog Item	Portfolio Backlog	Evaluate, Explore, Integrate
Problem	Problem	Integrate, Deploy, Operate
Product Backlog Item	Product Backlog	Evaluate, Explore, Integrate
Product Design	Product Design	Explore, Integrate
Product Release	Release Composition	Integrate, Deploy, Release
Product Release Blueprint	Release Composition	
Requirement	Requirement	Explore, Integrate
Resource	Resource	Deploy, Consume
Runbook	Diagnostics & Remediation	Deploy, Operate
Scope Agreement	Proposal	Evaluate, Explore, Integrate
Service Contract	Service Level	Deploy, Release, Consume, Operate
Service Monitor	Monitoring	Deploy, Operate
Service Offer	Offer	Release, Consume
Service Offer Catalog	Offer	Release, Consume

Data Object	Functional Component	Value Streams
Source	Source Control	Integrate
Strategic Objective	Strategy	Evaluate, Explore
Strategic Theme	Strategy	Evaluate, Explore
Subscription	Order	Integrate, Consume
Test Case	Test	Integrate
Test Plan	Test	Integrate
Usage Record	Usage	Consume
Value Stream	Enterprise Architecture	Evaluate

A.3. Value Streams

Table A-3. Value Streams

Value Stream	Functional Component	Data Object
Consume	Consumption Experience, Identity, Offer, Order, Chargeback, Change, Fulfillment Orchestration, Resource, Fulfillment, Usage, Service Level, Incident, Knowledge, Configuration, Cost Modeling	Actual Product Instance, Change, Chargeback Contract, Chargeback Record, Cost Model, Desired Product Instance, Entitlement, Fulfillment Book, Identity, Incident, Interaction, KPI, Knowledge Item, Order, Resource, Service Contract, Service Offer, Service Offer Catalog, Subscription, Usage Record
Deploy	Release Composition, Change, Fulfillment Orchestration, Resource, Fulfillment, Service Level, Problem, Knowledge, Configuration, Monitoring, Event, Diagnostics & Remediation	Actual Product Instance, Change, Desired Product Instance, Event, Fulfillment Book, KPI, Knowledge Item, Log, Problem, Product Release, Resource, Runbook, Service Contract, Service Monitor
Evaluate	Policy, Strategy, Enterprise Architecture, Portfolio Backlog, Proposal, Product Portfolio, Product Backlog, Investment	Architecture Blueprint, Architecture Roadmap Item, Budget Item, Digital Product, Policy, Portfolio Backlog Item, Product Backlog Item, Scope Agreement, Strategic Objective, Strategic Theme, Value Stream
Explore	Policy, Strategy, Enterprise Architecture, Portfolio Backlog, Proposal, Product Portfolio, Product Backlog, Requirement, Product Design	Architecture Blueprint, Architecture Roadmap Item, Digital Product, Policy, Portfolio Backlog Item, Product Backlog Item, Product Design, Requirement, Scope Agreement, Strategic Objective, Strategic Theme

Value Stream	Functional Component	Data Object
Integrate	Policy, Portfolio Backlog, Proposal, Product Backlog, Requirement, Product Design, Source Control, Pipeline, Build Package, Release Composition, Test, Defect, Order, Fulfillment Orchestration, Fulfillment, Problem, Monitoring	Build Package, Defect, Desired Product Instance, Fulfillment Book, Log, Order, Pipeline, Policy, Portfolio Backlog Item, Problem, Product Backlog Item, Product Design, Product Release, Requirement, Scope Agreement, Source, Subscription, Test Case, Test Plan
Operate	Consumption Experience, Change, Service Level, Incident, Problem, Knowledge, Configuration, Monitoring, Event, Diagnostics & Remediation	Actual Product Instance, Change, Event, Incident, Interaction, Knowledge Item, Log, Problem, Runbook, Service Contract, Service Monitor
Release	Release Composition, Offer, Fulfillment, Service Level, Knowledge, Cost Modeling	Cost Model, Fulfillment Book, Knowledge Item, Product Release, Service Contract, Service Offer, Service Offer Catalog

A.4. Functional Component Map

Table A-4. Functional Component to Value Streams via Data Objects

Functional Component	Evaluate	Explore	Integrate	Deploy	Release	Consume	Operate
Policy	Policy	Policy	Policy				
Strategy	Strategic Objective, Strategic Theme	Strategic Objective, Strategic Theme					
Enterprise Architecture	Architecture Blueprint, Architecture Roadmap Item, Value Stream	Architecture Blueprint, Architecture Roadmap Item					
Portfolio Backlog	Portfolio Backlog Item	Portfolio Backlog Item	Portfolio Backlog Item				
Proposal	Scope Agreement	Scope Agreement	Scope Agreement				

Functional Component	Evaluate	Explore	Integrate	Deploy	Release	Consume	Operate
Product Portfolio	Digital Product	Digital Product					
Product Backlog	Product Backlog Item	Product Backlog Item	Product Backlog Item				
Requirement		Requirement	Requirement				
Product Design		Product Design	Product Design				
Source Control			Source				
Pipeline			Pipeline				
Build Package			Build Package				
Release Composition			Product Release	Product Release	Product Release		
Test			Test Case, Test Plan				
Defect			Defect				
Consumption Experience						Interaction	Interaction
Identity						Entitlement, Identity	
Offer					Service Offer, Service Offer Catalog	Service Offer, Service Offer Catalog	
Order			Order, Subscription			Order, Subscription	
Chargeback						Chargeback Contract, Chargeback Record	
Change				Change		Change	Change

Functional Component	Evaluate	Explore	Integrate	Deploy	Release	Consume	Operate
Fulfillment Orchestration			Desired Product Instance	Desired Product Instance		Desired Product Instance	
Resource				Resource		Resource	
Fulfillment			Fulfillment Book	Fulfillment Book	Fulfillment Book	Fulfillment Book	
Usage						Usage Record	
Service Level				KPI, Service Contract	Service Contract	KPI, Service Contract	Service Contract
Incident						Incident	Incident
Problem			Problem	Problem			Problem
Knowledge				Knowledge Item	Knowledge Item	Knowledge Item	Knowledge Item
Configuration				Actual Product Instance		Actual Product Instance	Actual Product Instance
Monitoring			Log	Log, Service Monitor			Log, Service Monitor
Event				Event			Event
Diagnostics & Remediation				Runbook			Runbook
Cost Modeling					Cost Model	Cost Model	
Investment	Budget Item						

Appendix B: Acronyms and Abbreviations

ABB

Architecture Building Block

AI

Artificial Intelligence

API

Application Program Interface

BIA

Business Impact Assessment

CI

Configuration Item

CI/CD

Continuous Integration/Continuous Delivery or Deployment

CIO

Chief Information Officer

CMDB

Configuration Management Database

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

DAST

Dynamic Application Security Testing

DevOps

Combined Development and Operations practice

DevSecOps

Combined Development, Security, and Operations practice

DoS

Denial of Service

DPIA

Data Privacy Impact Assessment

ERP

Enterprise Resource Planning

eTOM

Business Process Framework (TM Forum)

GPS

Global Positioning System

GtB

Grow the Business

HR

Human Resources

IaaS

Infrastructure as a Service

Id

Identity

IDE

Integrated Development Environment

IoT

Internet of Things

ISACA

Information Systems Audit and Control Association

ISO

International Standards Organization

IT

Information Technology

ITFM

IT Financial Management

ITIL

Information Technology Infrastructure Library

ITSM

IT Service Management

KPI

Key Performance Indicator

MAO

Maximum Acceptable Outage

ML

Machine Learning

MMP

Minimum Marketable Product

MTTR

Mean Time To Repair

MVP

Minimum Viable Product

OLA

Operational-Level Agreement

OMG

Object Management Group

OT

Operational Technology

PaaS

Platform as a Service

PMBOK

Project Management Body of Knowledge

QoS

Quality of Service

RFI

Request for Information

RFP

Request for Proposal

ROI

Return On Investment

RPO

Recovery Point Objective

RtB

Run the Business

RTO

Recovery Time Objective

SaaS

Software as a Service

SAFe

Scaled Agile Framework

SAST

Static Application Security Testing

SBB

Solution Building Block

SCA

Software Composition Analysis

SIA

Security Impact Assessment

SLA

Service-Level Agreement

SLM

Service-Level Management

SLO

Service-Level Objective

SOA

Service-Oriented Architecture

SPOF

Single Point Of Failure

TCO

Total Cost of Ownership

TDD

Test-Driven Development

TOSCA

Topology and Orchestration Specification for Cloud Applications (OASIS)

TVA

Threat and Vulnerability Assessment

UAT

User Acceptance Testing

UI

User Interface

UML

Unified Modeling Language

UX

User Experience

VM

Virtual Machine

WIP

Work in Process/Progress

XLA

Experience-Level Agreement

XML

Extensible Markup Language

Index