Practical Software and Systems Measurement (PSM) Digital Engineering Measurement Framework

Version 1.0c
June 21, 2022

Developed and Published by Members of:

Practical Software & Systems Measurement

Systems Engineering Research Center

Aerospace Industries Association

National Defense Industrial Association

International Council on Systems Engineering

Department of Defense Research & Engineering

The Aerospace Corporation

PSM Product Number: PSM-2022-05-001

INCOSE Product Number: INCOSE-TP-2022-002-01

Copyright Notice:

CONTENTS

LIST OF FIGURES

List of Tables

EXECUTIVE SUMMARY

Stakeholders and subject matter experts from across industry, government, and academia have come together here to work collaboratively on this consensus measurement framework to 1) help enterprises transition from traditional document and artifact-based development to a digital model-based future, and 2) to better assess the measurable impacts and benefits they aspire to achieve.

A successful measurement program depends on establishing a clear context and operational definitions for the measures to be collected. The context is defined as a set of work activities and measurement concepts defined in Section 2. Common definitions are listed in Section 3. The Digital Engineering (DE) measurement framework was developed using an approach based on Practical Software and Systems Measurement (PSM)[1], detailing common information needs to derive an initial set of digital engineering measures. Sections 4 and 5 describe this process. The derivation of these measures is documented in an "Information Categories-Measurable Concepts-Measures" (ICM) Table, in Section 7. The information needs address goals and the project (or product) and enterprise perspectives (i.e., What do we want to know with respect to the goals?) to provide insight and drive decision-making. The ICM identifies an initial set of measures to address these information needs. For the highest priority measures, sample measurement specifications have been developed to describe these measures in detail along with guidance for their use. These are provided in Section 8.

This initial DE measurement framework proposed by our team of representative stakeholder experts is intended to help projects and enterprises establish an initial path toward a measurably effective transition and implementation of digital engineering processes, tools, methods and measures. This document represents the first of several steps along this path, which will be a long and challenging, but rewarding journey. Our industry will learn, iterate, and evolve as we go. It does not address, at this time, some of the broader aspects of DE transformation such as enterprise adoption, but will in a later version. A discussion of next steps is included in Section 6. We hope enterprises across a variety of application domains will find this initial measurement guidance useful to assess the effectiveness of their respective digital engineering transformation initiatives.

1. INTRODUCTION

1.1 BACKGROUND

Our industry is undergoing profound changes from traditional engineering requirements, design, development, integration, and verification methods based on documents and artifacts to a future based on digital models and cross-functional digital representations of system designs and end-to-end solutions. This document adopts a definition of digital engineering (DE) from the Defense Acquisition University (DAU):

> An integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life cycle activities from concept through disposal.[2]

This document also adopts a generalization of the digital and model-based aspects of engineering process from the initial release of ISO/IEC/IEEE DIS 24641:2021(E) standard: Systems and software engineering – Methods and tools for model-based systems and software engineering:[3]

> formalized applications of modeling to support systems and software engineering

Many of the measurable benefits of DE are associated with the use of both data and validated digital models as a "source of truth" across life cycle activities. Model-based systems and software engineering (MBSE) is an approach that uses models to drive all aspects of the product life cycle and that data and model elements are created once and reused by all upstream and data consumers.[4]

INCOSE is among many stakeholders that see digital MBSE as foundational to the future of our industry:

> The future of Systems Engineering is Model Based, leveraging next generation modeling, simulation and visualization environments powered by the global digital transformation, to specify, analyze, design, and verify systems.
>
> *INCOSE Systems Engineering Vision 2035 p.30*[5]

INCOSE defines Model-Based Systems Engineering (MBSE) as the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE has a particular value in DE as an approach to express and capture the relationships, interdependencies, and associated processes connecting systems level models and other disciplinary models as well as the life cycle process flow. MBSE supports system models that are useful for showing relationships among system functions, requirements, suppliers, acquirers, and users. Models allow relationships between constituent data elements to be established and both to be processed by software thus enabling efficiency-improving DE capabilities.

This framework recommends integrating MBSE and the terminology and practices of Model-Driven Development (MDD) from the software community into a single MBSE process framework. MBSE is a Systems Engineering approach centered on evolving models that serve as the "main / major source of knowledge" about the entity under consideration.

Thus, DE has three interrelated concerns: the transformation of engineering activities to fully digital infrastructure, artifacts, and processes; the use of authoritative sources of data and models to improve the efficiency and productivity of engineering practice; and the use of MBSE practice to fully integrate system data and models with engineering, program management, and other domains and disciplines.

As of this writing, our industry is still in the early stages of this digital transformation, and our processes, tools, methods, and measures must mature to fully achieve the apparent benefits of applying digital engineering methods and models across the product and system life cycle.

Organizations must be able to measure the effectiveness and business impact of their transformation efforts relative to traditional engineering methods. This need brought this DE Measurement Working Group together to define a proposed consensus measurement framework to help enable and assess effective digital engineering transformations.

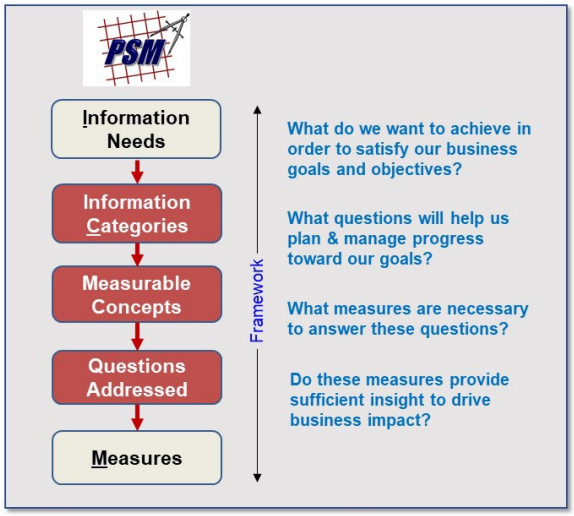


Figure 1.1-1: PSM Measurement Information Framework

Measures of effectiveness start with objectives. Accordingly, the stakeholder author team has chosen to build this measurement framework aligned with information needs (What do we want to know?) to define measures for decision making, following a process based on *ISO/IEC/IEEE 15939:2017 Systems and software engineering—Measurement process*[6] and Practical Software and Systems Measurement (PSM).[7] The PSM measurement information framework from Information Needs to Measures is summarized in Figure 1.1-1 and described in detail in Section 4.

1.2 MOTIVATION

Motivation for transformation towards a broad digital engineering initiative for model-based design, development and acquisition was sparked by the June 2018 release of the U.S. Department of Defense (DoD) Digital Engineering Strategy.[8] As illustrated in Figure 1.2-1, the strategy outlines five goals:

1. Formalize the development, integration and use of models to inform enterprise and program decision making.
2. Provide an enduring, authoritative source of truth.
3. Incorporate technological innovation to improve the engineering practice.
4. Establish a supporting infrastructure and environment to perform activities, collaborate, and communicate across stakeholders.
5. Transform the culture and workforce to adopt and support digital engineering across the life cycle.

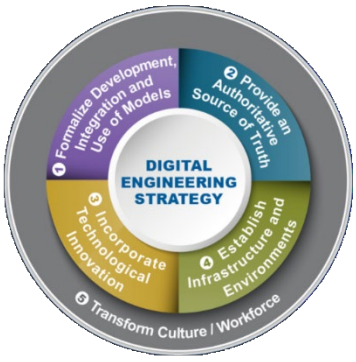


Figure 1.2-1: DoD Digital Engineering Strategy

The DoD Digital Engineering Strategy describes a foundation for enterprise stakeholders across government, industry, and academia to work on their respective digital transformation initiatives. These included partnerships with industry associations (INCOSE, NDIA Systems Engineering Division, AIA Engineering Management Committee, and others) on several collaborative initiatives such as:

- DoD Digital Engineering Working Group (DEWG)
- Digital Engineering Information Exchange Working Group (DEIXWG)[9]
- INCOSE Model-Based Capability Matrix (MBCM)[10]
- PSM User's Group Workshop for adapting Systems Engineering Leading Indicators for Digital Engineering,[11] as input to a planned revision of the PSM/INCOSE/MIT/SEA Systems Engineering Leading Indicators Guide.[12]

These sources were a basis for identifying some of the business information needs that are now articulated in the PSM DE measurement framework.

1.3 Stakeholder Collaboration in DE Measurement Working Group

A broad set of stakeholders across government, industry, and academia shared business imperatives to implement their digital engineering transformations and realize measurable benefits in performance, effectiveness, and product quality relative to traditional engineering methods. Defining a set of measures for digital engineering was identified by the DEWG as one of the government "pain points" (in their terminology) for enabling digital transformation.

In 2020, the AIA Engineering Management Committee (EMC) defined a strategic project plan to define a set of measures for digital engineering. Motivated by similar concerns, other industry associations (NDIA Systems Engineering Division, INCOSE, and member companies) offered to collaborate with AIA on this project, using a PSM measurement process applied successfully on a collaborative PSM/NDIA/INCOSE project to define a measurement framework for Continuous Iterative Development (CID).[13] Other stakeholders with related objectives subsequently joined this effort as listed in section 1.5, with the goal that the team could develop a digital engineering measurement framework with wide consensus for its use across industry.

The team started by gathering a set of information needs and objectives for digital engineering outcomes, which proved to be strongly aligned with research underway at the Systems Engineering Research Center (SERC) on DE benefits and measures described in section 1.4. This formed the basis for definition of the DE measurement framework described in the remainder of this document.

1.4 Derivation of the Measurement Framework

Several organizations have performed research studies on digital model-based engineering that have factored into this DE measurement framework. The SERC at Stevens Institute of Technology (supported by researchers at Virginia Tech) collaborated with INCOSE and the NDIA Systems Engineering Division on a survey to benchmark the maturity of Model-Based Systems Engineering (MBSE) practices across an enterprise. Survey questions were derived from the INCOSE Model-Based Enterprise Capability Matrix[14] and included questions on the maturity of DE/MBSE measurement. An additional study focused on deriving a DE Metrics framework from that survey and other literature provided an additional broad categorization of the DE/MBSE measurement landscape. These studies created a framework for describing the benefits of DE but also discovered that actual measurement in the community is still at its early stages. Analysis results are published in the following SERC reports:

- SERC-2020-SR-001, Benchmarking the Benefits and Current Maturity of Model-Based Systems Engineering across the Enterprise: Results of the MBSE Maturity Survey.[15]
- SERC-2020-SR-003, Summary Report on Digital Engineering Metrics[16]

The SERC survey analysis substantiated an industry early in its digital transformation progress with low maturity in measures of digital engineering effectiveness, with much room for future improvement but optimistic on the benefits and value to be achieved on DE. The SERC additionally conducted a literature review of digital engineering benefits and measures, whether perceived, observed, or measured.

Early discussion between the subject matter experts in the DE Measurement Working Group and members of the SERC research team settled on eight primary benefits of DE. These primary benefits (things an enterprise should do with data and models) were linked to secondary benefit measures and organizational adoption measures in a causal analysis.[17] This causal analysis and the expertise of the working group created the initial set of measurement concepts and constructs in this framework, which were used to define the initial version of the ICM Table presented in section 7. The initial set of constructs are intended to isolate those measurements that are most closely linked to the primary benefits of DE. This is not intended to replace any other measurement constructs that are associated with other disciplinary engineering processes.

The primary benefits are linked causally to the secondary benefits and measures specified in Section 8 as shown in Table 1.4-1. Note that this in not all the possible benefits and measures, only those currently defined in specifications in Section 8.

Table 1.4-1: Primary Benefits and Applicable Measurement Specifications from the Causal Analysis

Primary Benefits	Description	Applicable Measurement Specifications
Higher level support for automation	Use of tools and methods that automate previously manual tasks and decisions	8.6 Product Automation 8.7 Deployment Lead Time
Early Verification and Validation (V&V)	Moving tasks into earlier developmental phases that would have required effort in later phases	8.4 DE Anomalies 8.5 Adaptability and Rework 8.7 Deployment Lead Time

Primary Benefits	Description	Applicable Measurement Specifications
Reusability	Reusing existing data, models, and knowledge in new development	8.4 DE Anomalies 8.5 Adaptability and Rework 8.7 Deployment Lead Time
Increased Traceability	Formally linking requirements, design, test, etc. via models	8.7 Deployment Lead Time 8.8 Runtime Performance
Strengthened Testing	Using data and models to increase test coverage in any phase	8.1 Architecture Completeness and Volatility 8.2 Model Traceability 8.3 Product Size
Better Accessibility of Information (ASoT)	Leveraging an Authoritative Source of Truth (ASoT) to increase access to digital data and models to increase the involvement of stakeholders in program decisions	8.7 Deployment Lead Time 8.8 Runtime Performance
Higher Level of Support for Integration	Using data and models to support integration of information and to support system integration tasks	8.6 Product Automation 8.2 Model Traceability
Multiple Model Viewpoints	Presentation of data and models in the language and context of those that need access	8.1 Architecture Completeness and Volatility 8.7 Deployment Lead Time

1.5 CONTRIBUTORS

Table 1.5-1: PSM DE Measurement Framework Editors

Editors	Organization
Joseph M. Bradley	Main Sail, LLC Leading Change, LLC
Cheryl Jones	US Army DEVCOM AC Practical Software and Systems Measurement (PSM)
Geoff Draper	L3Harris Technologies AIA Engineering Management Committee NDIA Systems Engineering Division
Tom McDermott	Systems Engineering Research Center (SERC) Stevens Institute of Technology
Paul Janusz	US Army DEVCOM AC Practical Software and Systems Measurement (PSM)

Table 1.5-2: Additional Authors and their Organization

Authors	Organization
Lennis Bearden	L3Harris Technologies
Giacomo Gentile	Collins Aerospace
Richard Halliger	Volkswagen Group
Ray Madachy	Naval Postgraduate School
Drew Miller	The Boeing Company
Young Park	Collins Aerospace
Bob Scheurer	The Boeing Company NDIA Systems Engineering Division AIA Engineering Management Committee

Table 1.5-3: Core Team Contributors and their Organization

Core Team	Organization
Kaitlin Henderson	Virginia Tech
Al Hoheb	Aerospace Corporation
Bill Luk	BAE Systems AIA Engineering Management Committee
Lisa Murphy	Siemens Digital Industries Software
Jeff Nartatez	Office of the Undersecretary of Defense for Research and Engineering (OUSD R&E) [SAIC]
Garry Roedler	INCOSE NDIA Systems Engineering Division
Alejandro Salado	The University of Arizona
Frank Salvatore	Office of the Undersecretary of Defense for Research and Engineering (OUSD R&E) [SAIC]
Paul Segura	The Boeing Company AIA Engineering Management Committee
Natasha Shevchenko	Software Engineering Institute, Carnegie Mellon University
Marilee Wheaton	Aerospace Corporation INCOSE

Additional thanks go to the many additional colleagues who contributed to the development of the guide thorough participation in meetings, workshops and reviews.

Table 1.5-4: Additional Contributors

Additional Contributors	Organization
David Allsop	The Boeing Company
Sanjay Angadi	Ansys AIA Engineering Management Committee
David Cooper	BAE Systems
Mimi Davidson	Office of the Undersecretary of Defense for Research and Engineering (OUSD R&E) [SAIC]
John Dilger	BAE Systems
Paul Embry	L3Harris Technologies AIA Engineering Management Committee
Ann Hodge	Sandia National Laboratories
Mike McLendon	Retired
Robert Minnichelli	Aerospace Corporation
Gery Mras	Aerospace Industries Association (AIA)
Rosco Newsome	Ernst & Young (EY)
Bill Nichols	Software Engineering Institute, Carnegie Mellon University
Ryan Noguchi	Aerospace Corporation
Macaulay Osaisai	L3Harris Technologies
Steven Quinn	BAE Systems
Donna Rhodes	Massachusetts Institute of Technology (MIT)
Chris Schreiber	Lockheed Martin Corporation NDIA Systems Engineering Division
Paul Solomon	Retired
Brian Tenney	Lockheed Martin Corporation AIA Quality Assurance Committee
Sundar Thyagarajan	L3Harris Technologies
Steven Turek	United States Air Force
Timothy Walden	Lockheed Martin Corporation AIA Engineering Management Committee
John Wood	Naval Information Warfare Command

2. MAJOR CONCEPTS

This PSM DE measurement framework provides guidance on information needs and measures from two perspectives: project and enterprise. In many cases, the same base measures may be used, although aggregated to higher levels for enterprise needs. In other cases, different base measures may be used, or equivalent base measures used to answer different questions. The measurement specifications provide initial guidance on tailoring measures and indicators for these different perspectives and aggregation levels.

DE is generally implemented as a set of processes, tools, methods, and measures for the life cycle definition, development, and sustainment of complex engineered systems. DE creates not only the product itself, but also the digital data and models that define and then support the product over its life cycle. Because DE processes help to define the capabilities of the eventual system, DE measures can serve as useful leading indicators for other product related measures. DE can produce additional products in support of delivered data, hardware, and software products such as digital twins or other model- or simulation-based executable systems. For DE, stakeholders include expected users of the system and software, development teams, support teams, acquirers, operators, and managers. In an integrated DE environment, all relevant stakeholders, at all security and management levels, require secure and immediate access to the digital information they need to do their work. DE measures thus focus on the digital information products (data and models) developed and used across a product life cycle. Not all measures specified in Section 8 will be needed for a program and other measures may need to be specified. This framework is intended to be an initial list of potential measures.

A challenge with measures is both ensuring that they provide information needed to support decision making and that they are actually collected and used. A small set of measures should be tailored for each program and organization, focused on those needed for fact-based decision making. The measures should be regularly reviewed to ensure they are being used and that the decisions made using those measures are producing the intended outcomes. If not, other measures may be required, or additional training may be required for decision makers on how the measures can be utilized. For DE, the information is related to the primary benefits listed in Table 1-1. DE measures should inform the team, product managers, and/or the enterprise that they are achieving these benefits.

A successful measurement program depends on establishing a clear context and operational definitions for the measures to be collected. Definitions can sometimes vary depending on the references and how measures are applied. The diagrams and definitions that follow provide the terminology used in this DE measurement framework, in order to establish a common understanding, so that measures can be implemented and used consistently with community consensus.

2.1 DE WORK DECOMPOSITION

Decomposition of the DE work activities is generally associated with models, underlying data, and the digital infrastructure supporting them. All are important concepts in the measurement approach and have related specifications. Figure 2.1-1 shows a basic decomposition of the work associated with DE.

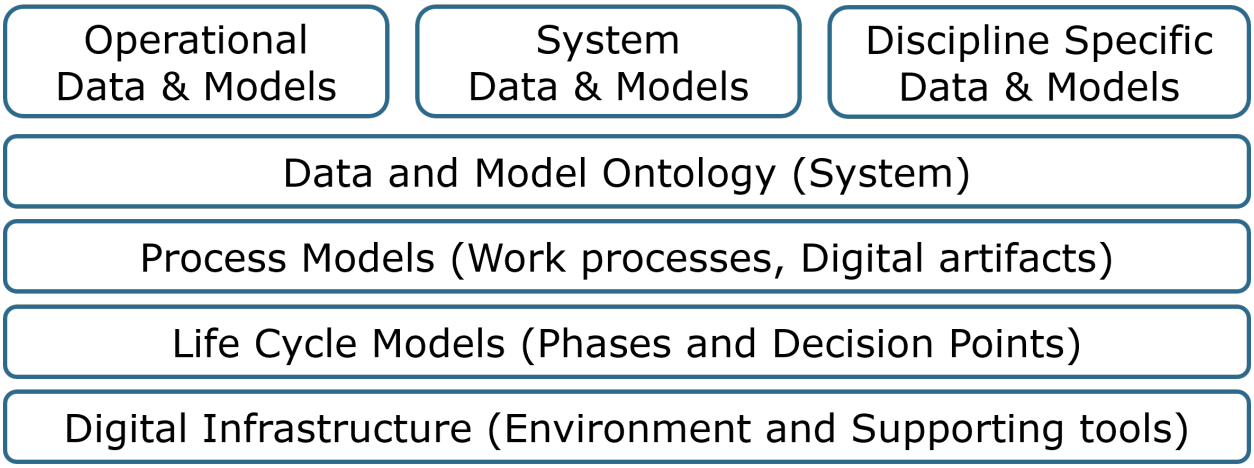


Figure 2.1-1: Decomposition of DE Work Activities

- Digital Infrastructure: The establishment of a set of processes, tools, methods, and measures that support the other DE work efficiently and productively, as well as the training and organizational capabilities to use these assets. The digital infrastructure may be program and domain-specific and will integrate tools from multiple disciplinary practices. Work requires an established and up-to-date digital infrastructure, which may be developed incrementally and evolve over time while maintaining the integrity of the digital content and its timeliness. The digital infrastructure must support the information needs and related measurement data for the organization.
- Life cycle Models: DE supports multiple practices and life cycle approaches. DE measures are generally associated with life cycle phases, decision points, and information needs. The measurement model implementation must be tailored to the specifics of the system/program life cycle(s).
- Process Models: Work is planned and implemented through a set of defined processes that evolve and produce a set of life cycle artifacts in digital form designed to integrate across the products, people, and processes involved in the project. A DE process model defines stakeholder roles, digital artifacts, when they are required, how they are used, how they are managed, and how data is produced and consumed by the stakeholders.
- Data & Model Ontology: DE artifacts are maintained in a digital repository, referred to as an Authoritative Source of Truth (ASoT). In a DE-based project, stakeholders work from the same data and models. This repository consists of sets of application specific data models, which define how data is stored and accessed, and a set of domain ontologies, which define more generic concepts and relationships in the domain that support sharing of data and knowledge. In order for work to proceed efficiently, all users of the repository must be able to work from a common taxonomy and underlying set of ontological relationships maintained by the DE toolsets. Work must include the development and maintenance of an appropriate data and model ontology.

- Operational, System, and Discipline Specific Models: DE is primarily concerned with the development and support of models and the data used by the models to support life cycle decisions. There is not a single model, but a set of models used to define the operational use of the system, analyze discipline specific concerns, and manage the relationships between individual models. In MBSE, the System Model is the result of a unique work activity that is used as the central repository for design decisions that span multiple engineering and business concerns; design decisions are captured as model elements in that System Model.[18] Modeling concerns include abstraction, correctness, completeness, accuracy, authority, accreditation, and validation. All of these affect the nature and amount of work necessary to develop and support models.

2.2 DE CONCEPTS

2.2.1 Authoritative Source of Truth

The concept of an Authoritative Source of Truth (ASoT) is central to the use of DE. Use of the ASoT requires a set of digital artifacts that is structured such that every artifact (data element or model element) is owned by a single entity and managed in only one place. In use, linkages to these artifacts are by reference only. Because all other DE activities refer back to the primary "source of truth" location, updates to the artifact in the primary location propagate to the entire system without the possibility loss or duplication.[19]

By definition, an authoritative source of truth is an entity such as a person, governing body, or system that applies expert judgement and rules to proclaim a digital artifact is valid and originates from a legitimate source.[20] The authoritative source of truth for a digital artifact serves as the primary means of ensuring the pedigree, credibility and coherence of the digital artifact that its creators share with a variety of stakeholders. It gives stakeholders from diverse organizations and distributed locations the authorization to access, analyze, and use valid digital artifacts from an authoritative source. The owners of digital environments or the community for digital engineering ecosystems provides stakeholders with an authoritative source of truth that assures confidence in the quality of the digital artifact across disciplines, domains, and life cycle phases.

In order to do so, a digital artifact's authoritative source of truth should meet four conditions. First, the digital artifact originates from a repository recognized by a governing entity as a System of Record (SoR). Second, the majority of experts accepts the credibility, accuracy, relevance, timeliness, and trustworthiness of a digital artifact because it meets their "criteria of truth". For example, in the MBSE domain, the digital artifact may meet the criteria of truth when most stakeholders agree that the preponderance of evidence upholds the validity of the digital artifact because it represents a commonly accepted perspective of reality. Third, a digital artifact's source is authoritative when most experts agree that the source is legitimate. Finally, the digital artifact originates from a technological system that maintains its integrity and reinforces the conditions. If the SoR satisfies the four conditions, then it is the Authoritative Source of Truth for its digital artifact.[20]

2.2.2 Model Element

The ISO/IEC/IEEE draft MBSE standard defines model element as atomic (elementary) items that represent individual components, actions, states, messages, properties, relationships, and other items that describe composition, characteristics, or behavior of a system.[3]

A model element is an abstraction drawn from the system being modeled, representing an elementary component of a model. The number and type of model elements in the ASoT will be determined by the development process. Delligatti states that if the system model is the central repository for design decisions, each design decision is captured as a model element or relationship between elements.[21] There is no predefined categorization of elements – they can be defined by the underlying ontology of the system model or of the tool used to create and manage the models.

The DE measurement approach and associated measures should recognize a defined concept of a model element such that 1) the relative size of the DE effort can be measured and compared to other efforts or plans, and 2) the quality of the DE design decisions (correctness and completeness) can be measured.

As one example, Sparx Systems defines the following Model Element Objects associated with SysML:[22]

> Model - Creates a Package containing a SysML Model.
>
> Model Library - Creates a Package containing a SysML Model Library.
>
> View - Creates a stereotyped Class that defines a SysML View of a system, from the perspective of a SysML Viewpoint.
>
> Viewpoint - Creates a stereotyped Class that defines a SysML Viewpoint, which specifies the rules and conventions for the construction and use of Views.
>
> Stakeholder - Creates a stereotyped Class that defines a SysML Stakeholder.
>
> Package - Groups model constructs in a single unit of containment.

As another example, IBM defines UML model elements into the following four categories:[23]

> Structural model elements - These elements model the static parts of a system. Some examples include classifiers such as actors, classes, components, information items, and nodes.
>
> Behavioral model elements - These elements model the dynamic parts of a system. Typically, you find behavioral model elements in state machine and interaction diagrams. Some examples include activities, decisions, messages, objects, and states.
>
> Organizational model elements - These elements group model elements into logical sets. A package is an example of an organizational model element.
>
> Annotational model elements - These elements provide comments and descriptions.

In order to extract measurement information from the ASoT, the project must determine the type of model elements it will measure. These will be constrained by the tools selected. Additional work is required to standardize on guidance for model elements that are most relevant to DE measurement.

2.2.3 Life cycle Phase

A life cycle is a set of phases and decision gates that capture the evolution of a system, product, service, or other human-made entity from conception through retirement.[24] Every developed product has a life cycle, even if it is not formally specified. The purpose of specifying a life cycle is to establish a framework for meeting stakeholder needs in an orderly and efficient manner, increasing the likelihood for optimizing the use of resources against the schedule. A life cycle consists of phases, with each life cycle phase having a purpose and an outcome. Life cycle phases and decision gates for transition between phases can be used to mature the product design by establishing specific checkpoints to ensure that acquirer and user needs are properly understood and met before committing time and resources too early. These checkpoints provide the development team, support team, management (internal and external), and other key stakeholders an incremental view of the progress being made with respect to planned expectations for that point in the life cycle, as well as related risks and issues. The checkpoints also provide opportunities for follow-on course correction to help ensure the project's successful mission delivery.

Each life cycle phase represents a team's work on the product leading to a release. A release is defined as some set of artifacts, and the work required to support, update, and then retire the product after a release. Each life cycle phase is an agreement between stakeholders in the project to create a product baseline and a decision point (called phase gates) that formally defines how the project should move forward. Each phase can have one or more gates. Each life cycle phase produces a set of artifacts that are used by the subsequent phases. The total set of these artifacts is termed the baseline. Often programs use a phase gate review process to determine artifact expectations or suitability for the next phase. Each gate has a target status; when the product has that status, the product can pass through the gate.[25]

In a DE-based project, all artifacts are managed in the ASoT. Configuration management of these artifacts from phase to phase and gate to gate must be assured to create consistency of artifacts across stakeholders. A primary benefit of DE is to improve the quality of the product as it moves from phase to phase. As many of these artifacts are not the actual product, it is important to maintain a formal process to assess their quality at each phase.

3. TERMS AND DEFINITIONS

Terms and definitions used in this document are derived from the following primary sources:

- SEVOCAB Vocabulary (Systems and Software Engineering Vocabulary) - ISO/IEC/IEEE terminology (www.computer.org/sevocab)
  - ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes
  - ISO/IEC/IEEE 24641:2000 (E) Systems and software engineering – Methods and tools for Model-based systems and software engineering
  - ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description
  - ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB database
- ISO online browsing platform for terminology (https://www.iso.org/obp/ui)
- Defense Acquisition University (DAU) as collected on the U.S. Undersecretary of Defense for Research and Engineering Digital Engineering Website (https://ac.cto.mil/digital_engineering/)
- Digital Information Exchange Working Group (DEIX) Topical Encyclopedia (https://www.omgwiki.org/MBSE/doku.php?id=mbse:topical_encyclopedia_for_digital_engineering_information_exchange_deixpedia)
- Model Based Engineering Forum website (https://modelbasedengineering.com)

3.1 DIGITAL ENGINEERING

Table 3-1 includes general terms and definitions associated with DE that are used throughout this measurement framework. The source of definition is included in italics after the definition.

Table 3.1-1: Digital Engineering Terms and Definitions

Term	Description
Digital Engineering	An integrated digital approach that uses authoritative sources of systems' data and models as a continuum across disciplines to support life cycle activities from concept through disposal. *(DAU Glossary)*
Digital Artifact	The artifacts produced within, or generated from, the digital engineering ecosystem. These artifacts provide data for alternative views to visualize, communicate, and deliver data, information, and knowledge to stakeholders. *(DAU Glossary)*
Digital Engineering Ecosystem	The interconnected infrastructure, environment, and methodology (process, methods, and tools) used to store, access, analyze, and visualize evolving systems' data and models to address the needs of the stakeholders. (*DAU Glossary*)
Digital Thread	An extensible, configurable, and component enterprise-level analytical framework that seamlessly expedites the controlled interplay of authoritative technical data, software, information, and knowledge in the enterprise data-information-knowledge systems, based on the Digital System Model template, to inform decision makers throughout a system's life cycle by providing the capability to access, integrate, and transform disparate data into actionable information. *(DAU Glossary)*

Digital Twin	An integrated multiphysics, multiscale, probabilistic simulation of an as-built system, enabled by Digital Thread, that uses the best available models, sensor information, and input data to mirror and predict activities/performance over the life of its corresponding physical twin. *(DAU Glossary)*
Model	A mathematical or physical representation (i.e., simulation) of system relationships for a process, device, or concept. *(IEEE Standards Dictionary, IEEE Std 1641)* Representation of a real world process, device, or concept. *(IEEE Standards Dictionary, IEEE Std 2413-2019)*
Digital System Model	A digital representation of a system, generated by all stakeholders, that integrates the authoritative technical data and associated artifacts, which defines all aspects of the system for the specific activities throughout the system life cycle. *(adapted from the DAU Glossary)*
Discipline Specific Model	Representation of a system, or system elements from the perspective of a discipline addressing domain specific concerns where the model elements come from a specific discipline. *(ISO/IEC/IEEE 24641:2000)*
Model-Based	Represented using a formalism which has a formal syntax and semantics, usually with a theoretical basis, and expressible in a symbolic language Note 1 to entry: Presentation of such models is often graphical but the definition mandates that the graphical representation be translatable into a symbolic language, thereby constraining interpretation of the graphical representation. Note 2 to entry: In order to satisfy specific stakeholder concerns, "model-based" is often used as a qualifier to characterize a kind of design, or practice, e.g. model-based system engineering, model-based design, model-based specification. *(ISO Online browsing platform)*
Model-Based Development	Development that uses models to describe the behaviour or properties of an element to be developed. *(ISO Online browsing platform)*
Model-Based Engineering	A software and systems development paradigm that emphasizes the application of modeling principles and best practices throughout the life cycle. *(ISO Online browsing platform)*
Model Configuration Item	A logical part of the model that is maintained in a controlled fashion, i.e., have a trackable revision history. *(ISO/IEC/IEEE 24641:2000)*
Model Element	Atomic (elementary) items that represent individual components, actions, states, messages, properties, relationships, and other items that describe composition, characteristics, or behavior of a system *(ISO/IEC/IEEE 24641:2000)*
Model Library	A group of model elements that are intended to be reused in other models. (*modelbasedengineering.com*)
System Model	An interconnected set of model elements that represent key system aspects including its structure, behaviour, parametric, and requirements. *(derived from discussion in ISO/IEC/IEEE 24641:2000)* A system model - is used to represent a system and its environment - may comprise multiple views of the system to support planning, requirements, architecture, design, analysis, verification, and validation - is a representation of a system with various degrees of formalism often expressed as a combination of descriptive and analytical models. The system model is an integrating framework for other models and development artifacts including text specifications, engineering analytical models, hardware and software design models, and verification models. In particular, the system model relates the text requirements to the design, provides the design information needed to support

	analysis, serves as a specification for the hardware and software design models, and provides the test cases and related information needed to support verification and validation. *(IEO/IEC/IEEE DIS 24641:2021)*

3.2 OTHER RELEVANT DEFINITIONS

Table 3-2 contains other relevant terms and definitions.

Table 3.2-1: Other Terms and Definitions

Term	Description
Capability	The ability to achieve a desired effect under specified standards and conditions through combinations of ways and means to perform a set of tasks. These are often represented as higher-level solutions spanning multiple product releases.
Product	Result of a process. There are four generic product categories: hardware (e.g., engine mechanical part); software (e.g., computer program); services (e.g., transport); and processed materials (e.g., lubricant). *(ISO/IEC/IEEE 15288: 2015)*
Requirement	Statement that translates or expresses a need and its associated constraints and conditions (*ISO/IEC/IEEE 29148:2011*)
Anomaly (synonym defect)	Anything observed in the documentation or operation of a system that deviates from expectations based on previously verified system, software, or hardware products or reference documents *(IEEE 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation, 3.1)* It is a condition in a product, that does not meet its requirements or end-user expectation, causes it to malfunction or to produce incorrect/unexpected results, causes it to behave in unintended ways, or leads to quality, cost, schedule, or performance shortfalls. Any digital artifact used to directly define, produce, or support the product should be included in the set of anomalies. Anomalies may be documented in a defect repository, or they may be added to the planned work for consideration in future iterations or life cycle phases. Escaped Anomalies are anomalies detected or resolved prior to release of the baseline artifact containing the anomaly. Anomalies are generally tracked separately for internal and external baselines. Contained Anomalies, also known as Saves, are anomalies detected and resolved within a phase or iteration before internal or external baseline deliveries of the artifact and version containing the anomaly. Once approved for implementation, a Change Request may be created, or the anomaly may be used to track implementation.
Change	Revision that adds, removes, or modifies any aspect of a digital artifact as managed in the ASoT.
Change Request	Requested change to the digital artifact. Some organizations may use Anomalies or Defects instead of separate Change Requests to track issues.
Stakeholder	Individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations (*ISO/IEC/IEEE 15288:2015 Systems and software engineering--System life cycle processes*)

4. Measurement Process Model and Information Model

The measurement process model in Figure 4-1 provides a framework for implementing measurement. The model is built around a typical "Plan-Do-Check-Act" management sequence.

The Plan Measurement activity encompasses the identification of management and technical information needs and the selection of appropriate measures to address these needs using the Measurement Information Model. The output of the Plan Measurement activity is a well-defined measurement approach that directly supports project and enterprise information needs.

Perform Measurement encompasses the collecting and processing of measurement data; the use of the data to analyze both individual information needs and how the information needs and associated issues inter-relate; and the generation of information products to present the analysis results, alternative courses of action, and recommendations to the project and enterprise decision makers.

The Evaluate Measurement activity assesses both the applied measures and the capability of the measurement process, and it helps identify associated improvement actions. The Establish and Sustain Commitment activity provides the resources and enterprise infrastructure required to implement a viable measurement program. More details on these activities are provided in Annex A.1.

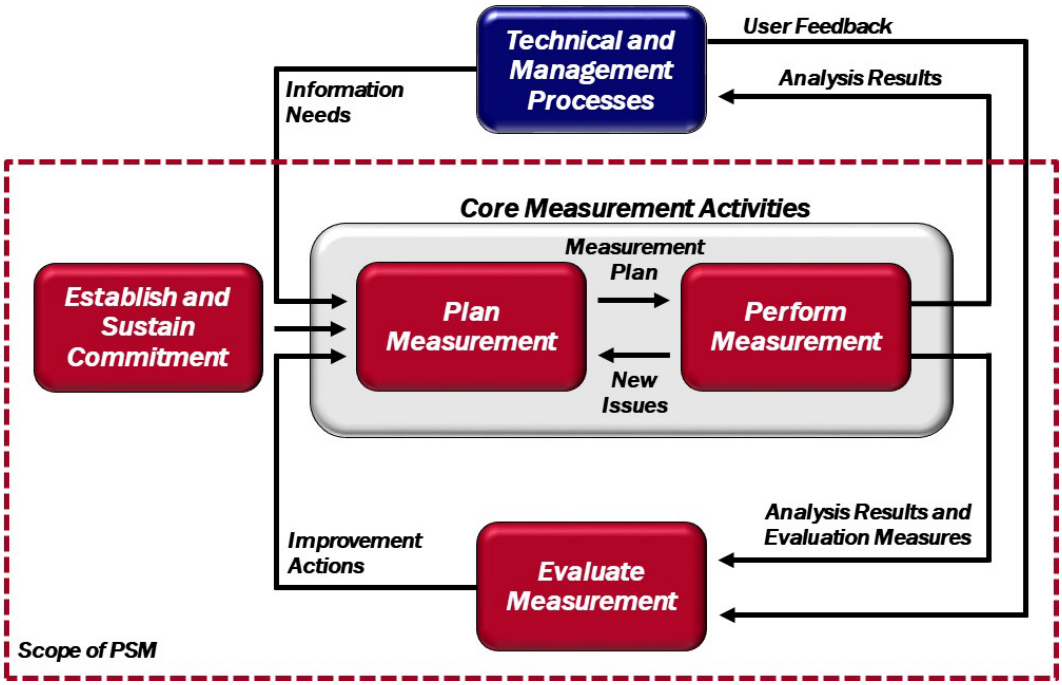


Figure 4-1: Measurement Process Model

In the PSM methodology, the information model links the data that can be measured to a specified information need, as illustrated in Figure 4-2.

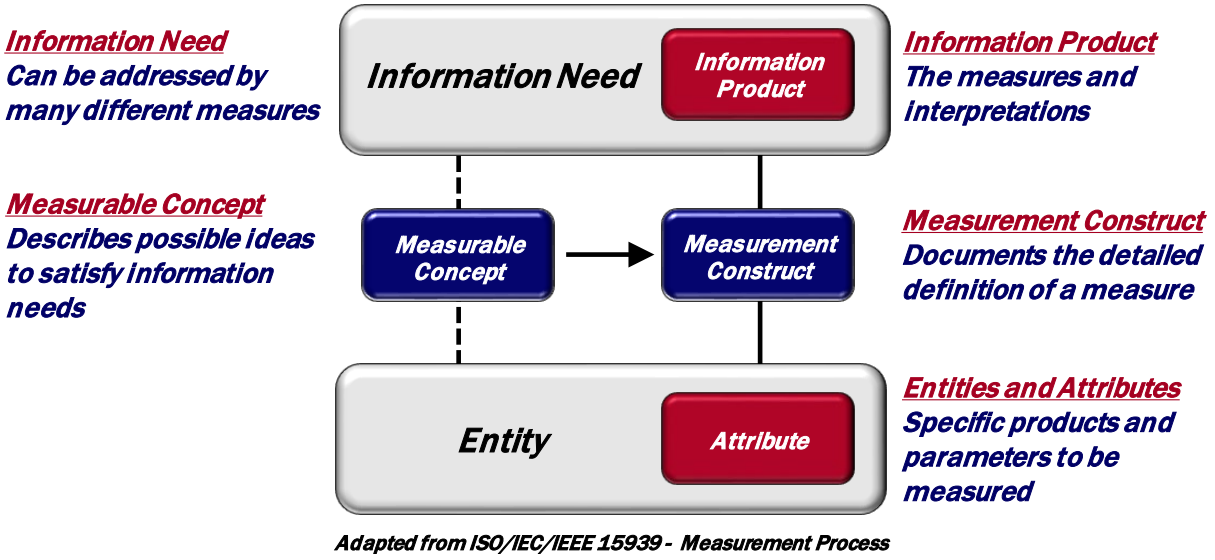


Figure 4-2: Information Model - High-Level View

The things that can actually be measured include specific attributes of the systems and software processes and products, such as size, effort, and number of defects. The measurement construct describes how the relevant attributes are quantified and converted to indicators that provide a basis for decision making. For each measurable concept, there may be multiple measurement constructs (measures) that address the identified measurable concept. A single measurement construct may involve three types, or levels, of measures: base measures, derived measures, and indicators. The measurement planner needs to specify the details of the measurement constructs to be used in the measurement plan, as well as the procedures for data collection, analysis, and reporting.

At each of the three levels of measures - base measures, derived measures, and indicators - additional information content is added in the form of rules, models, and decision criteria. Figure 4-3 illustrates the structure of a measurement construct in more detail.

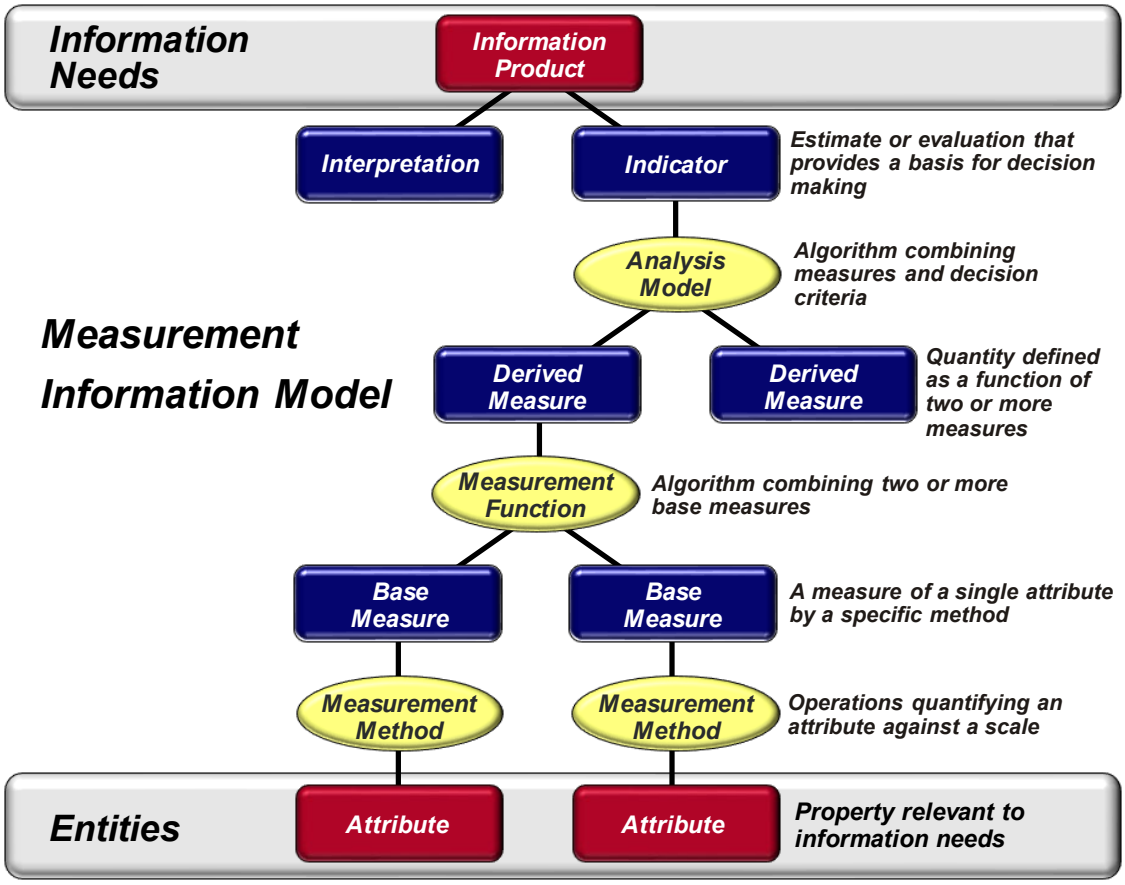


Figure 4-3: Measurement Information Model

This figure depicts how the base measures collected are dependent on the information needed by stakeholders. It also shows how the data is combined into an indicator and analysis model to form the information product provided to management.Figure 3.2-4 contains a specific example of this, for the DE Anomalies measure that is specified in Section 8.4. The measurement specification details the information needs, base measures, derived measures, and analysis models for each proposed measure.

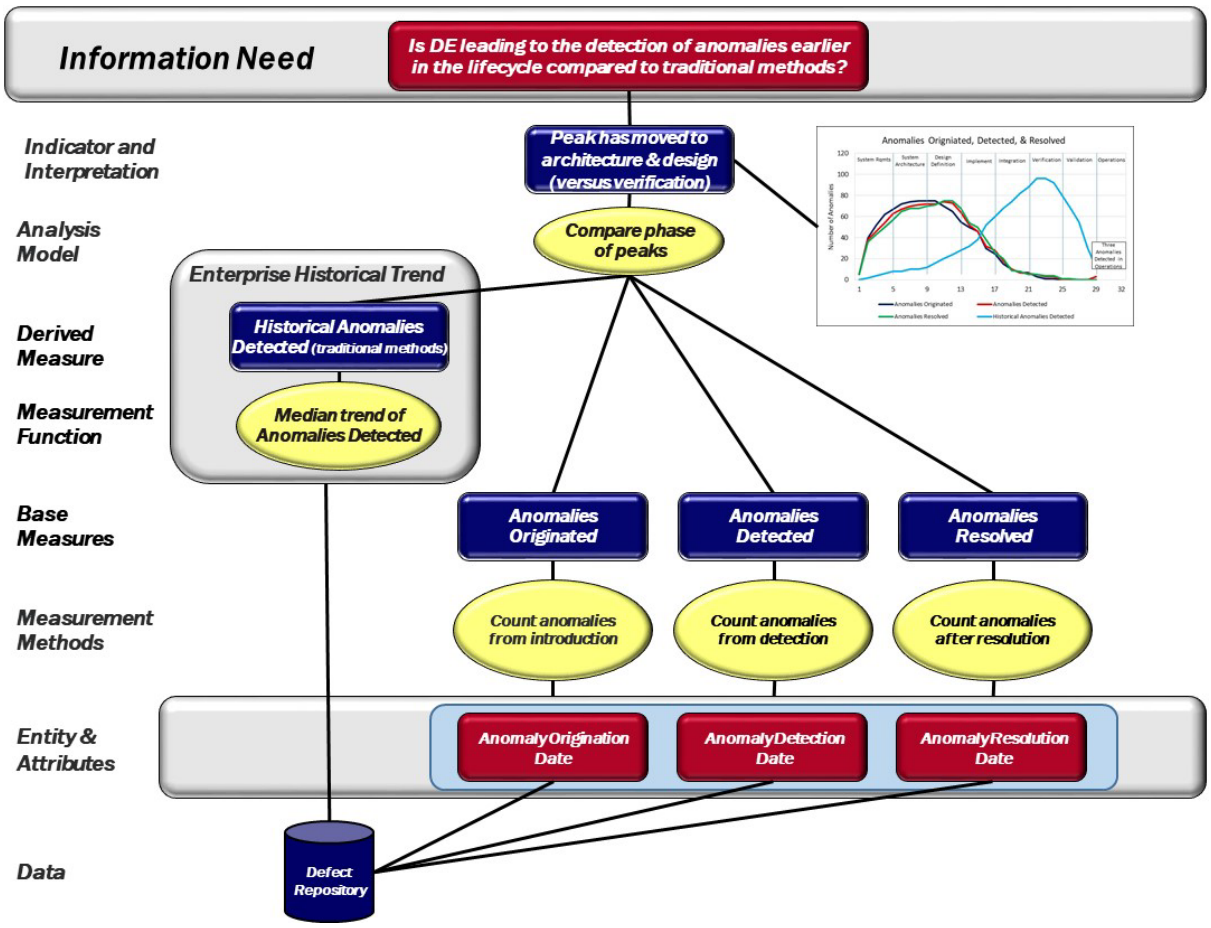


Figure 3.2-4: Mapping Data to Measures

In this example, the project has a quality management process with a defect repository (entity) that contains the data from which the measures will be calculated. Three base measures are defined: anomalies originated, anomalies detected, and anomalies resolved. An indicator is drawn that evaluates these measures over time, in order to evaluate whether the DE process addressed the question of whether the use of DE is leading to the detection of anomalies earlier in the lifecycle compared to traditional methods.

Definitions for the terminology in this section can be found in Annex A and more discussions on these topics are available in Practical Software and Systems Measurement (John McGarry (Author), 2001)[1]

5. MEASUREMENT PRINCIPLES

Some key principles for these information needs and measures include:

- What you measure is driven by the information needs of the enterprise and project.
- The existing technical and management infrastructures define what you are able to measure. The measures selected need to be integrated into the DE process and infrastructure of the enterprise and project. The collection of measures should be automated by utilizing the functionality of existing tools or by creating custom tools to the extent practical. These tools should be integrated with business workflows, development processes used, and with other DE practices adopted.
- The set of measures included in the ICM Table are sample measures identified through survey and subject matter expert (SME) review as being important in selected circumstances and at various levels. They are a starting point for consideration in an enterprise or project.
- As organizations stand up or start digital engineering efforts, it is valuable to create an initial set of measures. Otherwise, the organization is trying to create measures once the operation is already up. This is likely to delay the development of measures.
- Few programs can afford to model the entire system, so focus on models of high-risk areas of design that provide those most value. A minimum practical set of measures should be selected and tailored based on organizational and program circumstances, tools, and processes.
- The environment you work in drives how the measurement results are interpreted. Decision context is extremely important to understanding the data and developing recommendations.
- The selected measures should inform decisions, answer key programmatic questions, and drive actions.
- Action must be taken to realize any benefit from measurement. The goal is fact-based decision making. If the defined measures are not providing the expected benefit, or not being used, then the measures should be re-evaluated, and new measures or indicators should be developed.

6. NEXT STEPS

This version of the PSM DE measurement framework is an initial set of measures proposed by subject matter experts in the nascent field of digital engineering. Several of these have proven useful in practice; others are more exploratory, but we expect they will be useful based on the expertise of the participants. Additional measures will be considered and added in future releases, based on user feedback from adoption and use, comments received from SMEs, and working group recommendations.

Potential future additions include:

- Measures for enterprise DE
- Return on Investment measures
- Measure additional productivity indicators related to velocity and agility
- Measure additional indicators that isolate new value to the enterprise through DE, in areas such as quality and knowledge transfer
- Measure enterprise and personnel process adoption
- Analyze breadth of usability and user experience with digital tools, and measure issues with usability
- Supportability and Maintainability measures (impact assessment agility)
- Security related measures
- Identify typical digital artifacts
- Specify leading indicators

7. Information Categories, Measurable Concepts, Measures (ICM) Table

The "Information Categories-Measurable Concepts-Measures" (ICM) Table provides the PSM DE measurement framework detailing common information needs and measures. It is called the ICM table, after "Information", "Concept", and "Measure". The information needs address information needs and concerns from both the project and enterprise perspectives. The ICM Table identifies a set of measures that have been identified as being practical measures to address these information needs, based on user surveys, SMEs, and practical experience from the working group members.

Information needs are based on goals and objectives. Separate columns are provided for the project and enterprise information needs (columns C and D). These are mapped to the PSM Information Categories, to related Measurable Concepts, and then to Potential Measures. This mapping is described in Table 3.2-1, and the ICM table is provided in Table 7-2.

Table 3.2-1: ICM Descriptions and Table Structure

Column	Title	Description
Col A	Information Category	Common groups of information needs defined in PSM to provide structure for the Information Model. These categories facilitate the identification and prioritization of project or enterprise-specific information needs. In practice, measures related to similar information needs can often be addressed using similar measurable concepts, thus reducing the number of base measures and resources needed to implement a viable measurement program. PSM information categories include: • Schedule and Progress • Resources and Cost • Product Size and Stability • Product Quality • Process Performance • Technology Effectiveness • Customer Satisfaction Information categories are defined in A.3 and in Chapter 2 of the PSM book.[1]
Col B	Measurable Concept	An idea about the entity that should be measured in order to satisfy an information need. The attribute, entity, or characteristic that we are trying to measure and provide quantitative feedback on, organized into similar groupings.

Col C	Project Information Needs	The insight necessary to manage *project* specific objectives and issues. This includes: • Objectives - project goals or requirements • Issues - areas of concern that could impact the achievement of an objective, including: - Risks - concerns that may occur - Problems - concerns that have occurred - Lack of information - inadequate data Project-specific information needs provide a basis for objective communication and informed decision making with regard to Project Estimation & Planning, Project Performance Tracking, Project Tradeoff Analysis, and Resource Management.
Col D	Enterprise Information Needs	The insight necessary to manage *enterprise* objectives and issues, across a number of projects or throughout the enterprise. These are used to determine if the best practices/processes are being implemented consistently across a portfolio of projects. They support decision making with regard to meeting enterprise objectives in areas such as Performance Measurement, Normative Performance Baselines, Technical and Business Policy, Investment Decisions & Analysis, Process Improvement, and Project Planning Guidelines. In many cases, the same base or derived measures may be used to address both project and enterprise information, with a different aggregation level or indicator.
Col E	Potential Measures	Measures that can be used to address the identified project or enterprise information needs. In some cases, the measure is detailed in a sample measurement specification provided in section 8: these are denoted by "*" in the ICM table.
Col F	Notes	Summary details regarding implementation of the potential measure that amplify the information provided.

The ICM table structure provides guidance in selecting measures that address specific project or enterprise goals and objectives. The table and the Measurement Specifications also help in tailoring and specifying the data and implementation requirements for each measure. Many measures provide insight into more than one Information Area; however, the ICM table lists these measures under a single information category for simplicity.

The ICM table includes a list of the most critical information needs of digital engineering at both the project and enterprise levels, along with candidate measures to address them. The ICM table is not intended to represent an exhaustive or required set of project management measures. However, these measures are expected to provide benefit over a wide range of projects and represents the best engineering judgment for addressing information needs faced by project and enterprise managers. Users should augment and tailor the

list of measures based on their own experience and requirements. No project should implement all of the measures listed in the ICM table, but should instead concentrate on implementing a minimum, practical set of measures to aid in decision-making.

Table 3.2-2: Information Categories, Measurable Concepts, and Measures

Information Categories	Measurable Concepts	Project Information Needs	Enterprise Information Needs	Potential Measures	Notes (Guiding Objectives)
Schedule and Progress	Architectural Completeness	How complete is the architecture? Does the architecture account for all required functions? Is the architecture sufficiently complete to proceed with design at acceptable risk?	What is the amount of schedule and design risk for each project? What is the architecture progress across projects?	Architecture Completeness and Volatility *	
Schedule and Progress	Model Coverage	What is the extent of traceability across digital model elements? What traceability gaps exist? What is our progress in completing the digital model?	What is the extent of model traceability for a set of projects? What is the modeling coverage and progress of the digital engineering capability across projects? What is the current upper limit of the digital engineering capability?	Model Traceability * Model Coverage (e.g. modeled elements)	Measurement is against only the digital model elements. Model elements are created to fulfill the functions and interfaces allocated during the architecture and design phases.
Size and Stability	Functional Size and Stability	What is the size and scope for the DE project or product? How much work must be done? How many functions and interfaces have been identified in the system architecture or design? How much is that changing? How does DE product size	Is the current project similar in size and scope to historical projects? Is the work scope changing? Is the schedule and effort sufficient to address changes? How does DE product size relate to estimates and measures of cost,	Product Size * (e.g. Model Elements) Architecture Completeness and Volatility * Functions Identified Functional Change Requests	In development, product size can be determined by a count of model elements. Function Volatility includes the aspects of continuing to identify new functions and/or having the functional allocation continue to change. In maintenance, change

Information Categories	Measurable Concepts	Project Information Needs	Enterprise Information Needs	Potential Measures	Notes (Guiding Objectives)
		relate to estimates and measures of cost, schedule, productivity, or performance?	schedule, productivity, or performance?		requests are often used as a measure of work scope.
Product Quality	Functional Correctness	Are we finding and removing anomalies early in the life cycle using models and shared information? Is the quality of the product in question adequate for the product to be used in subsequent phases or activities?	How many anomalies were released (escaped) to operations? Is the use of DE leading to the detection of anomalies earlier in the lifecycle compared to traditional methods or projects)? Has the detection curve shifted to the left?	DE Anomalies *	For digital engineering focus on the defects for modeling and simulation (including drawings).
Product Quality	Functional Correctness	How much rework effort is spent maintaining planned or unplanned changes to DE work products across the life cycle?	How much is rework reduced through use of DE? Can changes to work products be implemented more efficiently and with less effort in a DE environment relative to traditional methods?	Adaptability and Rework * Acceptance of Completed Work Products (e.g. Model Elements, Artifacts) Rework or Rework Defects	Completion of work products requires defined acceptance criteria. Rework is required when the acceptance criteria are not met.
Product Quality	Functional Correctness	What traceability gaps or defects exist in the digital model? Does model traceability support change impact assessments (requirements, design, compliance)?	Is architectural traceability improved using digital engineering methods relative to traditional approaches?	Model Traceability * Traceability Anomalies	
Process Performance	Process Effectiveness	How many released, validated system definitions/analyzed elements	Is the organization learning how to reduce the	Model Element DE Anomalies	

Information Categories	Measurable Concepts	Project Information Needs	Enterprise Information Needs	Potential Measures	Notes (Guiding Objectives)
		were functionally correct, but returned for rework?	number of defects released to operations?		
Process Performance	Process Effectiveness	Are we containing defects in early phases using models and shared information?	Are we finding and removing defects earlier using digital engineering methods relative to traditional methods?	DE Anomalies * Rework Effort Reworked Model Elements	For digital engineering focus on the defects for modeling and simulation (including drawings). The focus is whether the process is improved using digital engineering, versus the raw numbers.
Process Performance	Process Efficiency - Automation	What percentage of artifacts are automatically generated from digital models? To what extent are artifacts facilitating program reviews?	What is the extent of automation across projects? How much is automation contributing to meeting performance and quality objectives? What is the return on investment for DE? How much can cycle time be reduced through automation of DE?	Product Automation * Cycle Time	
Process Performance	Process Efficiency - Speed	How long does it take to deploy an identified feature/capability? How long does it take to deploy a viable product for operational use after a request is received?	How long does it take to develop a DE model or product? Does the DE process performance meet business objectives?	Deployment Lead Time * Cycle Time	Proper analysis also requires an enterprise approach for quantifying size or complexity of work products.

Information Categories	Measurable Concepts	Project Information Needs	Enterprise Information Needs	Potential Measures	Notes (Guiding Objectives)
		Where is the deployment bottleneck; in planning/backlog, implementation, or deployment of the implemented capability?			
Process Performance	Process Efficiency	Is productivity improving over time (normalized model element/artifact delivered by effort)? How many model elements/artifacts are being produced per release? How many can be expected to be produced for the next release?	Is productivity improving over time (normalized model element/artifact delivered by effort)? Is our productivity sufficient to meet our customer's needs? How much is productivity increased through the use of digital engineering?	Productivity Model Elements/Release Artifacts/Release	
Technology Effectiveness	Technology Performance	What is the runtime performance of the capability or system? What is the likelihood that runtime performance will meet operational requirements (for each alternative solution)? Where are the runtime performance bottlenecks, and how can operational performance be optimized?	How much does runtime effect interoperability of the system? Where is redesign needed to solve compatibility issues?	Runtime Performance * Elapsed Time	

8. Measurement Specifications

This section provides detailed measurement specifications for a representative sample of the prioritized set of potential measures, based on user surveys and input from subject matter experts. This includes:

8.1 Functional Architecture Completeness and Volatility
8.2 Model Traceability
8.3 Product Size
8.4 DE Anomalies
8.5 Adaptability and Rework
8.6 Product Automation
8.7 Deployment Lead Time
8.8 Runtime Performance

The detailed measurement specifications in this section contain a number of information fields. Annex A provides definitions for each field of the measurement specification, along with a blank template to facilitate tailoring or development of project or enterprise specific measures. Each sample measurement specification is also provided in MS Word format on the PSM web site (www.psmsc.com) to facilitate tailoring of the specifications that are provided in this document.

Not all of the measures specified will be needed or applicable for a specific project or enterprise. Additional measures may be needed to address other identified information needs. The measures provided in this version of the Framework focus primarily on project information needs. Future versions will include additional measures for enterprise information needs.

The measures provided are examples only. While many originated in support of large software-intensive engineering projects, with appropriate tailoring, they can be applied to hardware development and systems engineering across many domains. The measures can be used throughout a product or system life cycle, during concept definition, system definition, system realization (development, engineering), test and evaluation, system deployment, operations, and sustainment. The user is expected to select and tailor the measures used based on their information needs, environment, and project and enterprise processes.

For each of the indicators provided, sample decision criteria are included. These are provided as examples only: projects should define decision criteria based on their project or enterprise experience, parametric tool values, or industry norms. As more experience is gained, actual project experience should be the source of decision criteria, and the expected values should be adjusted accordingly.

When planning a measurement process, it is important to identify and select measures that are *leading indicators*, i.e., measures that provide advance warning of issues and problems versus those that only indicate issues or problems after the fact, i.e., *lagging indicators*. Therefore, in selecting and tailoring measures for a particular project or enterprise, it is important to identify and select measures of upstream products and processes that can enable early detection and corrective action before a series of downstream products are affected. For example, a measure of requirements volatility can provide a leading indicator of schedule slips or effort increases that might occur later in a project.

This Framework provides guidance on measures that are specific to digital engineering. Many other useful measures, that are also useful to digital engineering, have been described elsewhere and are not included here. These include:

Table 3.2-1: Useful Publications for Additional Measures

Publication	Description
Practical Software and Systems Measurement: A Foundation for Objective Management[1]	General measurement guidance
Systems Engineering Leading Indicators Guide (SELI)[12]	Measures specific to systems engineering that provide early indications of issues and problems
Practical Software and Systems Measurement Continuous Iterative Development Measurement Framework[13]	Measures specific to agile or DevSecOps developments

Effective measures are derived from project activities. A digital engineering environment offers many opportunities to collect measures effectively from the automated tools and environment used. The measures should be tailored to take advantage of this available data, and to automate data collection, analysis, and reporting processes wherever possible. It may not be cost effective or practical to have models for everything: however the more models available, the more benefit from a digital engineering approach. From a measurement standpoint, automation enables consistent execution of processes and makes data collection more efficient. Using an ASoT ensures data is up to date. The use of automation to extract the data can remove the reliance on an individual, and further allows analysts to focus their valuable time on analysis, synthesis, and interpretation of indicators to enable timely decisions and actions in response to measures provided.

In tailoring a measure specification, the processes and tools used should be described in the data collection, analysis, and reporting fields of the measurement specification. In this document, only general data collection and analysis guidance is provided as the details will be very specific to the project or enterprise.

8.1 ARCHITECTURE COMPLETENESS AND VOLATILITY

Measure Introduction		
Description	This measure is used to evaluate progress toward completion of an architecture in a system or product development. An architecture is foundational for aligning the problem space with the solution space. Completeness and stability (i.e., absence of volatility) in the functions comprising the architecture provide a direct view into the maturity of a system development with DE. At the team level, architecture progress is measured based on declared functions and associated interfaces at each designated boundary and associated level(s) of design in the system. At the program or enterprise level, this measure can be used to monitor overall progress toward definition of a complete architecture that emerges from a system's functional requirements and containing all levels of a system's model elements (including design). It may also provide an indication as to the fidelity of a system definition as each level is iteratively decomposed into functions and interfaces across architectural elements and boundary partitions. It may further be used to augment measurement of product quality by indicating product readiness with respect to expected capability/performance, allocated functionality, or verified functional traceability to source requirements. This measurement specification can be utilized on different views of the architecture, including the functional, logical, physical, etc. The specific indicator used as an example in this specification discusses the functional architecture, but similar indicators can be developed for other architecture views. Similar measures may also be used to measure the completeness and volatility of other model elements (e.g. interfaces, hardware or software design elements). An important component of defining an architecture is specifying the boundary partition for the system and each system element, to delineate the scope. This is particularly important in a system-of-systems development, to understand the boundary for the system of interest, and to indicate what is in and out of scope.	
Relevant Terminology	Function	A task, action, or activity that must be accomplished to achieve a desired outcome. A function may originate from source functional requirements, use cases, or functional decomposition.
	Source Functional Requirement	Statement that identifies what results a product or process shall produce; a function that a system or system component shall perform.
	Source Functions	Functions identified directly from source functional requirements.
	Derived Functions	Functions that are not explicitly stated in source functional requirements but are inferred from contextual requirements or decomposition during analysis or model development.
	Allocated Functions	Functions that levy all or part of the performance and functionality of a higher-level requirement on a lower level model element.

Information Need and Measure Description	
Information Need	How complete is the architecture? Does the architecture account for all required functions? Is the architecture sufficiently complete to proceed with design at acceptable risk?
Base Measure 1	Source Functions - Number of source functions within defined boundary partition [integer]
Base Measure 2	Derived Functions - Number of derived functions within defined boundary partition [integer]
Base Measure 3	Allocated Functions - Number of source and derived functions identified, decomposed, and allocated to model elements with complete traceability within defined boundary partition [integer]

Base Measure 4	Milestone Date [date] T0 = $Date_{Start}$; T1 = $Date_{M1}$; T2 = $Date_{M2}$; T3 = $Date_{M3}$; T4 = $Date_{M4}$, ...Tx= $Date_{Mn}$ The specific milestones used on a project are based on the enterprise and project processes. In a traditional development, M1 to M4 have historically been SRR, SFR, PDR, and CDR. In a continuous iterative development, the acquirer and supplier jointly agree on the necessary review milestones.
Derived Measure 1	Total Functions = Source Functions + Derived Functions [integer]
Derived Measure 2	Percent of Functions Allocated = Allocated Functions / (Source Functions +Derived Functions) [real]
Derived Measure 3	Function Volatility = (Change in Number of Identified Functions) per Increment of Time [integer]
Derived Measure 4	Allocation Volatility = (Change in Number of Allocated Functions) per Increment of Time [integer]

Indicator Specification	
Indicator Description and Sample	In Figure 8.1-1, the graph shows the identification and allocation of functions over time in a system development program. On the X-axis, key program milestones are identified that have a direct correlation to understanding the functional maturation over time. The Y-axis identifies functional counts based on what is presented at the start of the program by stakeholders (i.e., source functions) and those identified by the supplier (i.e., derived functions) over the course of the system development.

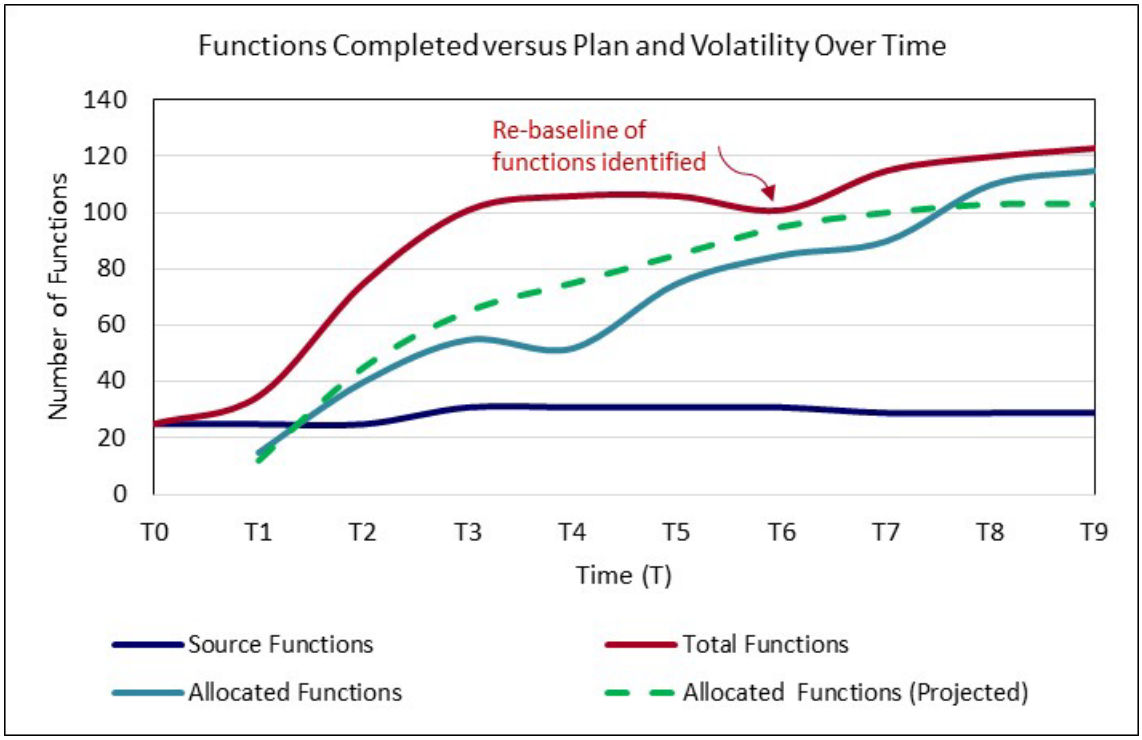


Figure 8.1-1: Functions Completed versus Plan and Volatility Over Time

The solid blue line represents the number of Source Functions for a system over time of the system's development. There are changes in this number at time T3 and time T7 due to a re-baselining of the functions identified. The dashed green line shows the Allocated Functions (Projected) prior to the start of the effort. The solid red curve shows the number of Total Functions, including the source and derived functions. There is also a drop in this curve at time T6, due to a re-baselining of the functions allocated. This curve evolves as the model elements are created and milestone reviews are completed.

As functions are identified and then allocated to model elements, changes are observed in allocation counts for Total Functions. The fluctuations in the red line show that functions may be identified or eliminated based on refinement of the model elements over time. The slope of the red line at any point represents the function volatility being experienced on the program. An increase in slope of the line represents greater volatility while a decrease in slope indicates less volatility. Negative slope indicates volatility associated with a net reduction in Allocated Functions. Complete function stability is indicated when the slope of the Allocated Functions line is zero (functional identification and allocation over time), thus signaling that all identified functions have been allocated to model elements in the system. The offset of the solid blue line from the red line represents a normal time lag associated with allocating functions via the architecture and design processes.

In iterative development, functions may evolve frequently for the iteration work scope defined by the prioritized backlog, and the time sequences are generally very short (e.g., weeks). Total functions are defined over time. Completeness and volatility of functions in an iterative context should be aligned with a testable definition of done, and generally will be measured for each iteration.

Data Point	Source Functions	Derived Functions	Total Functions	Change Per Time	Allocated Functions	Functions Allocated (Remaining)	Function Volatility (Allocated) Per Time	Percent Functions Allocated
T0	25		25					
T1	25	10	35	10	15	20	15	0.43
T2	25	50	75	40	40	35	25	0.53
T3	31	70	101	26	55	46	15	0.54
T4	31	75	106	5	52	54	-3	0.49
T5	31	75	106	0	75	31	23	0.71
T6	31	70	101	-5	85	16	10	0.84
T7	29	80	115	14	90	25	5	0.78
T8	29	80	120	5	110	10	20	0.92
T9	29	80	123	3	115	8	5	0.93

Figure 8.1-2: Functional Completeness & Volatility Analysis (Example Use Case)

It is useful to also look at the data that makes up this indicator, to more easily see the changes that are occuring, as shown in Figure 8.1-2. Late changes (Changes in Time or Function Volatility), even if they are small, have a disproportionate impact on project success.

Analysis Model

Functional completeness is achieved when Allocated Functions equals the Total Functions, for the iteration scope. Function stability is achieved (i.e., Function Volatility = 0) when the total number of added functions, changed functions, and deleted functions for a given unit of time is zero when compared to the previous unit of time of a measurement.

Architecture volatility is stable when the Allocated Functions = Total Functions, and both Allocation Volatility = 0 and Functional Volatility = 0 for the same unit of time.

Measures of Architecture Completeness and Volatility can be key indicators in determining when the architecture is sufficiently mature to justify proceeding with system design at acceptable risk. If the architecture is incomplete or continuing to undergo significant changes, this may indicate a risk of future rework.

Decision Criteria

Function Volatility should be 0 after milestone review 2 with 100% of derived functions identified within defined boundary partition, and 0 changes to source functions. Any changes after milestone review 2, will require rework, and additional review milestones.

Allocated Volatility should be 0 after milestone review 3 with 100% traceability from source and derived functions to model elements within defined boundary partition. Any changes after milestone review 3 will require rework, and additional verification and validation activities on the system.

Additional Information	
Additional Analysis Guidance	This model gives an understanding of key characteristics centered on functional capabilities and functional allocations to potential model elements involved in the system under development. Understanding of functional analysis and architecting in the system are needed to firmly employ this analysis model and achieve reliable indication of program health or signal risks. Completeness refers to the full establishment of functions allocated to model elements of a system, for the scope of work in an iteration. In iterative development, completeness of functions may vary across iterations. Volatility refers to the extent of change in total count(s) of involved elements. In this measure, those elements are functions, with volatility expressed as having a magnitude of positive, negative, or zero value.
Implementation Considerations	Stakeholder vague desires need to be converted to explicit and unambiguous function statements prior to addressing system functional volatility. The functional baseline needs to be solidified at milestone review 1 per stakeholder agreement. Delay in achieving the requirements baseline will potentially introduce additional functional volatility and allocation delays. In implementing this measure, the project or enterprise needs to define what it means to be complete. It may mean either supplier has completed work, work has been verified, identified defects have been resolved, or those defects have been verified.

Additional Specification Information	
Information Category	Schedule and Progress
Measurable Concept	Work Unit Progress
Relevant Entities	Requirements, Use Cases, Design Level or Defined Boundary Partition, System Under Design
Attributes	Functions Identified, Functions Allocated, Function Allocations Completed for each entity
Data Collection Procedure	At the team level, data is collected at the end of each increment of time by the team. Functions must be tested and satisfy "Done" criteria, with no orphan functions or functions with unterminated interfaces to be counted as completed. If a function does not satisfy "Done" criteria, then it is not considered "Complete" and it is not included in the Total Functions Allocated. For product measures, data is collected periodically (e.g., monthly, quarterly, end of each iteration or release). To maintain accuracy and consistency, it is preferable for Functional Volatility and Completeness measures to be collected in an automated fashion using digital modeling tools.
Data Analysis Procedure	Data is analyzed at the end of each derivation increment of time by the team during the derivation review and considered during the planning session for subsequent lower-level functional definitions. The data is also aggregated and analyzed at summary levels across derivation increment or releases to ensure that the program is completing its committed functional assignments. Function Volatility is Achieved when the total number of added functions, changed functions, and deleted functions for a given unit of time is zero when compared to the previous unit of time of a measurement.

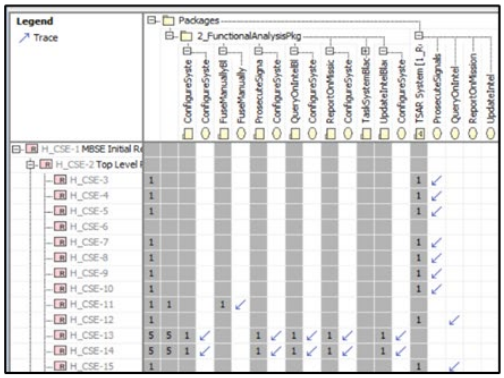
8.2 MODEL TRACEABILITY

Measure Introduction	
Description	The usefulness and quality of a digital model depends on the completeness and integrity of the relationships among model elements. Traceability between elements, such as requirements allocation and flow down to architectural, design, and implementation components, assures that the system solution is complete and consistent. Gaps in bi-directional traceability between the artifacts of two models or might indicate where further analysis or refinement are needed. This might further apply to traceability gaps within a single model, when there is no implicit traceability between artifacts of different design stages. The prerequisites of any traceability measurement are agreed-upon guidelines and definitions, e.g., what model elements and relationships shall be traced, that apply to the specific DE model of the system. *Note:* While traceability might be applied to any model elements of interest that shall be defined, functional architecture completeness always explicitly focuses on functions, requirements, and the associated hierarchy. Traceability reports and analyses might be facilitated by digital modeling tools. The traceability concepts and indicators in this specification are representative examples of more general traceability mappings and reports across the development life cycle, such as: • Traceability between stakeholder needs, system requirements, and allocated or derived requirements at each level of the system hierarchy • Traceability and flow down of requirements to the logical or physical solution domain (e.g., design, implementation, integration, verification, validation) • Allocation and traceability of performance measures or parameters, such as Measures of Effectiveness (MOEs) or Key Performance Parameters (KPPs) • Traceability of system interfaces
Relevant Terminology	Model Element: Modeling constructs used to capture the structure, behavior, and relationships among system model components (See 2.2.2 Model Element) Source Element: The base model elements defined per DE model from which other model elements shall be derived from or allocated to, e.g., a stakeholder needs. Destination Element: The model elements defined per DE model that shall be derived from or allocated to the Source Elements. Traceability Gap: One or more model elements defined per DE model that shall be traced, but that have not yet been derived or allocated to Source Elements. Note: For enhanced traceability concepts refer to the advanced topic discussion.

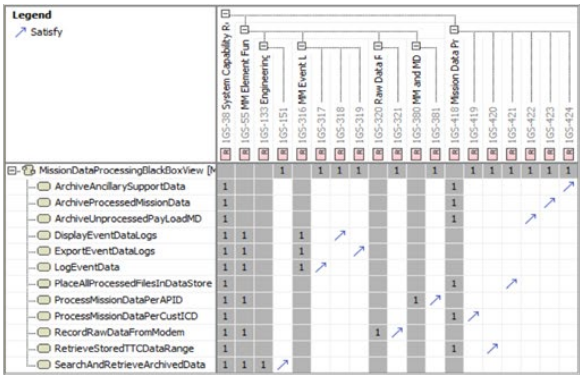
Information Need and Measure Description	
Information Need	What is the extent of achieved traceability coverage from Source Elements, e.g., requirements, down to the logical or physical solution domain? What is our progress in completing the digital model? What traceability gaps exist?
Base Measure 1	Model Elements Traced [integer] "Number of model elements in a 1 .. n source/destination element relationship(s) as defined in an agreed upon guideline.
Base Measure 2	Model Elements Not Traced [integer] Number of model elements not in any 1 .. n source/destination *element relationship as* defined in an agreed upon guideline.
Derived Measure 1	Total Model Elements = Model Elements Traced + Model Elements Not Traced [integer] Total number of model elements *Note:* As defined in an agreed upon guideline (See Base Measure 1 and Base Measure 2).

Derived Measure 2	Percent Traced = ((Model Elements Traced) / (Total Model Elements)) * 100 [percentage] Percent Not Traced = ((Model Elements Not Traced) / (Total Model Elements)) * 100 [percentage] Model traceability coverage Note: Once tools indicate that the model features the intended model traceability coverage, analysts shall review the model to check for any gaps or miscalculations not identifiable by tools.

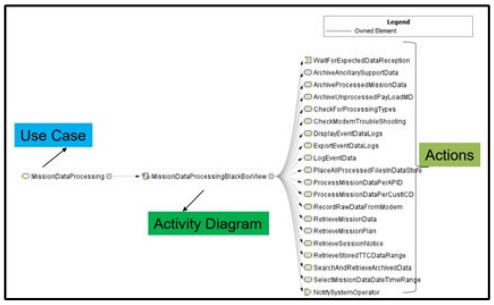
Indicator Specification	
Indicator Description and Sample	Model Traceability can be depicted using visual or tabular summaries of the relationships among model elements. The specific indicators may depend on the model elements for which traceability is being measured, and the built-in reports and analyses provided by the digital modeling tool. For example, traceability among model elements might be implemented by showing requirements derivation and model traceability coverage of stakeholder needs into system and component requirements. Representative example indicators used to assess traceability dependencies among selectable model elements (e.g., requirements, use cases, activities, logical architecture and design, physical design, interfaces, parameters, measures of performance) are depicted in Figure 8.2-1. Here, mostly 2-dimensional matrices containing model specific model elements of interest are utilized. Alternatively, the relationship between model elements might be depicted as flow down. With respect to Figure 8.2-1 (bottom left), a specific use case is linked to related actions via an activity diagram.



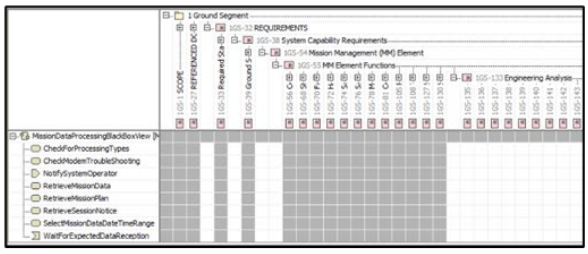
Traceability Between Model Elements (Dependency Matrix)



Relationships to Problem or Solution Domain (*«satisfy»* or *«refine»* Matrix)



Relation Map Diagrams (Model Traceability, Ownership)



Identifying Model Traceability Gaps (Orphans)

Excerpts from: 'MBSE and Requirements Analysis, Key to Successful System Engineering', M. Osaisai and F. Markham, 2019 MBSE Cyber Experience Symposium. Used with permission from Macaulay Osaisai. All other rights reserved.

Figure 8.2-1: Example Traceability and Dependency Diagrams

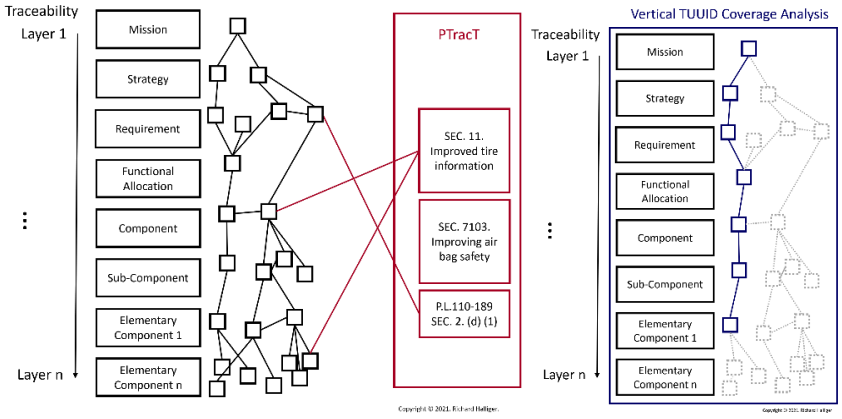
Traceability and model traceability coverage (or lack thereof) can be quickly visualized, and gaps or defects addressed, e.g., systems that do not satisfy requirements or any unsatisfied requirements might indicate incomplete work or systems without required functions.

For further visual, tabular, and reporting capabilities and information, refer to the advanced topic discussion section below.

Analysis Model	Projects and organizations shall define the objectives, constraints, and criteria for establishing traceability among applicable model elements. This is typically guided by a model schema, metamodel, or blueprint that constrains traceability to meet the model's purpose. Review and analyze traceability dependencies among model elements to assess the completeness, adequacy, quality, and integrity of the digital model. The analysis may vary according to the types of specific model elements selected, but general guidelines may include: • Each source (parent) model element (Model Element 1) should be traceable to one or more allocated or derived destination (child) model elements (Model Element 2). • Each destination (child) model element (Model Element 2) should be derived from, or refine, a parent requirement or model element (Model Element 1). • Determine if the set of linked dependencies are, in aggregate, sufficient to adequately implement the parent requirement or model element.
Decision Criteria	In case a desired model traceability coverage (Derived Measure 2), e.g., 70%, of model elements of interest has not been met, the team shall specifically address these gaps. To validate whether the system meets stakeholder needs, at minimum, the system requirements should be traceable to these stakeholder needs. Model elements that do not satisfy requirements, might be obsolete and shall be evaluated. Again, the prerequisites of any decision making are agreed-upon guidelines and definitions, e.g., what model elements and relationships shall be traced, that apply to the specific DE model of the system

Additional Information	
Additional Analysis Guidance	Traceability can be a useful indicator of model quality and modeling progress. Revisions to the model elements or relationships may be needed to close gaps. Derived measures of traceability for the selected model elements, such as Percent Traced and Percent Not Traced to assess the completeness and integrity of the digital model. Track progress in completing the traceability measures as the modeling effort matures.
Implementation Considerations	Traceability reports and analyses are typically available directly as built-in features of modeling tools. Traceability and analyses depend on the quality of relationships and dependencies established between modeling elements. Modeling conventions and guidelines should be established to assure the consistent use of model elements and relationships across the project. Failure to establish and enforce consistent modeling practices can impact model quality, and the integrity and usefulness of traceability measures. When stakeholders over-emphasize specific system requirements or physical implementations, then they should provide a rationale why these efforts are valid based on the needs-to-requirements flow down.

Additional Specification Information	
Information Category	Product Quality
Measurable Concept	Functional Correctness (Completeness)
Relevant Entities	Model components
Attributes	Level (e.g., system, requirements, design, component)
Data Collection Procedure	Counts of model elements and type are typically provided by modeling tools. Queries, scripts, or APIs may be available to automate the collection of model element count measures.
Data Analysis Procedure	Traceability reports (bi-directional linkages between selected parent and child modeling elements) are often generated directly from modeling tools. Review mappings between model elements for sufficient coverage. Generally, each parent must have one or more children, and every child must have at least one parent. Look for incorrect or disconnected traceability, such as orphans and barren requirements. Ensure adequate representations of associated modeling views and diagrams, e.g., use cases, sequence diagrams, activity diagrams.

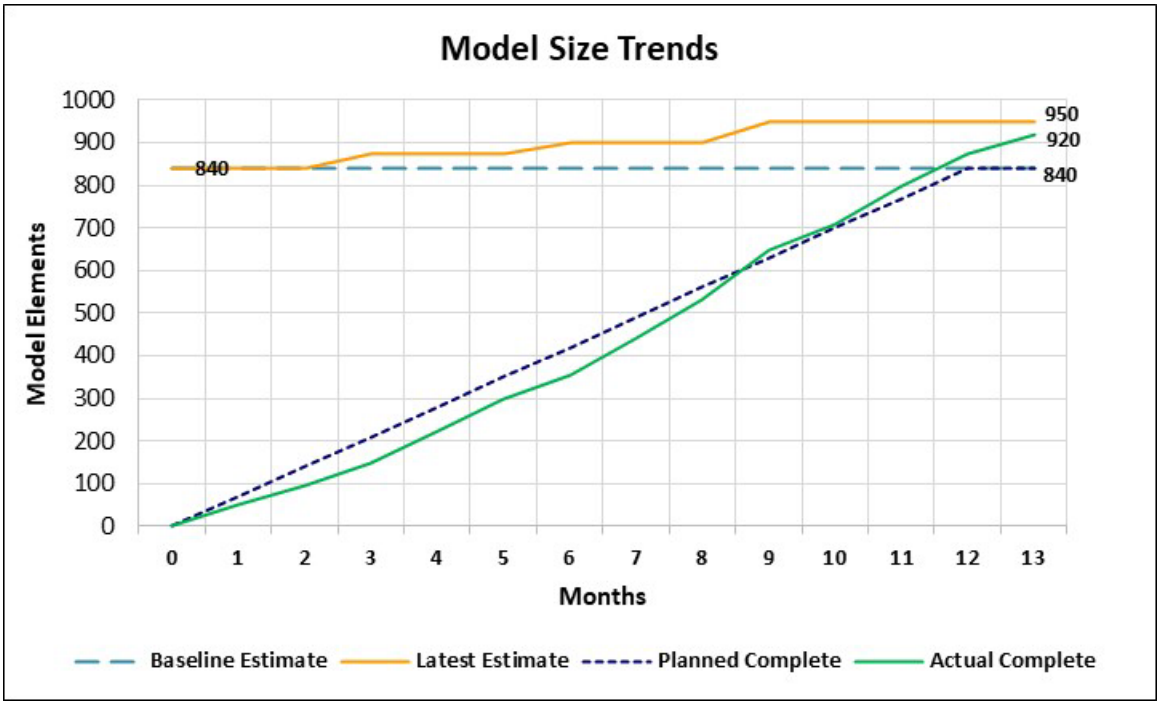
Advanced Topic - Traceability for Complex Systems and Missions	
Discussion	Beyond traceability of elements within a digital model as described above, traceability concepts can be scaled and expanded to consider higher level challenges, such as complex systems, compliance, and mission engineering. This way, traceability might also support other aspects, e.g., automated testing via executable model automation, impact assessments as a result of engineering change, and milestone reviews in acquisition pathways. Addressing these challenges on both the project and enterprise level is enabled by digital transformation and advancements in sophisticated toolsets. Complex systems can generally be decomposed into a hierarchical set of layers and a parallel set of legal frameworks to handle components and manage complexity. Distinct components and links between components within each layer are assigned Universally Unique Identifiers and attributes to enable 1 .. n relation traceability, inheritance, and assurance of data integrity across objects in the complex system. While the term Traceability Layer refers to general concept of the decomposing a system into vertical layers, an exemplary hierarchical set of Traceability Layers might include: • Mission Layer (mission needs, mission threads, mission effect chain) • Compliance and Strategy Traceability Layer (strategies, and legal or compliance constraints) • Requirements Traceability Layer (decomposition of requirements and link relationships) • Functional Allocation Traceability Layer (requirements allocation to domain functions) • Component Traceability Layers (hierarchical physical and cyberphysical component hierarchy linked to functions or requirements) • Sub-Component Traceability Layers (hierarchical physical and cyberphysical sub-component hierarchy linked to larger components) • Elementary Components Traceability Layers (allow further decomposition, as needed) Measures of traceability provide indicators of percentage coverage across layers (vertical, horizontal, requirements, mission needs, etc.).  **Figure 8.2-2: Traceability for Complex Systems and Missions[1]** Figure 8.2-2 illustrates of the concepts of decomposing complex systems into layers (left), the associated legal framework components run in parallel to these layers (middle), and traceability that runs through all layers to the top layer, e.g., Mission Layer (right). The red linkages illustrate the concept of horizontal traceability coverage. The second diagram illustrates traceability measures for vertical coverage. 1 Copyright 2021 by Richard Halliger. Reprinted with permission.
Description	The discussion above introduces high level concepts only. Further description and details are published in a white paper on the PSM website, https://www.psmsc.com/Prod_TechPapers.asp

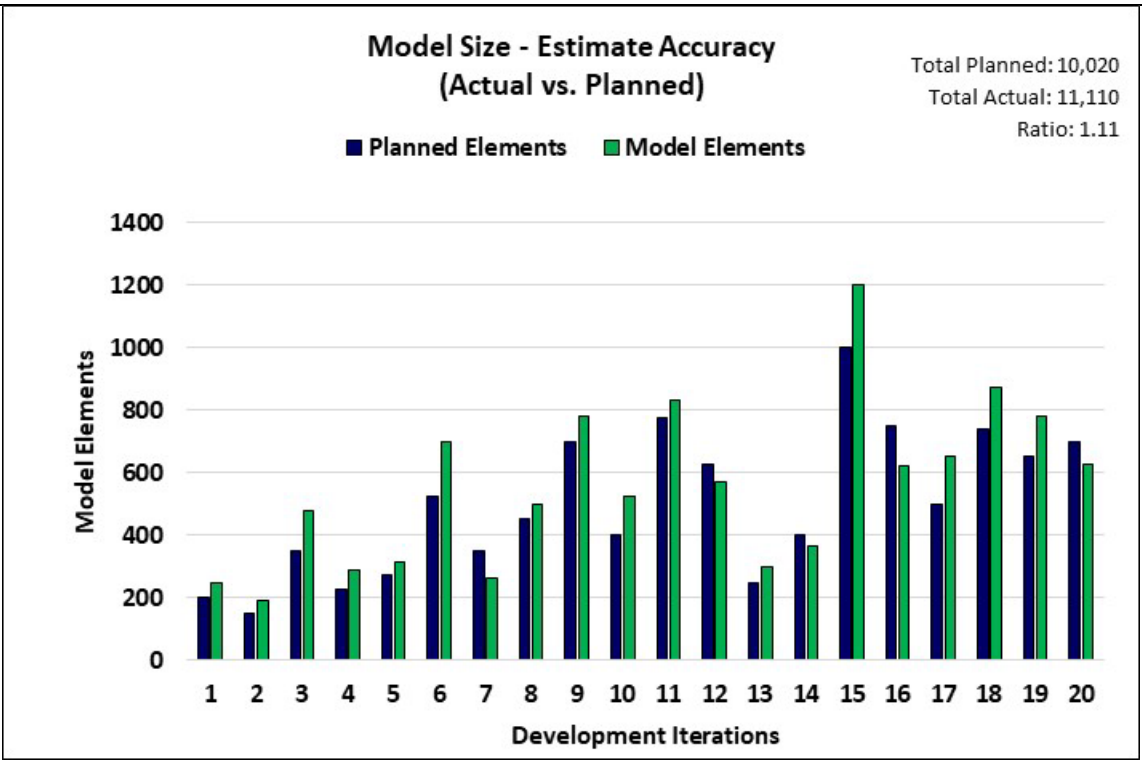
8.3 PRODUCT SIZE

Measure Introduction	
Description	Many process measures for estimating and managing engineering product development depend upon a meaningful characterization of the scope or quantity of work to be performed. Often product size is used as a proxy for determining measures such as effort (hours), schedule (months), productivity (size/hour), or capability performance (product delivered / months). Proxies for product size in this context are commonly used in many engineering disciplines, such as capabilities, requirements, use cases, unit or component counts, Lines of Code (LOC) or number of drawings. No such established proxies, conventions, or models for size or productivity are commonly used yet for digital engineering in practice. However some measure of product size is needed, at both the project and enterprise levels, to normalize historical performance data, characterize prior work, relate it to estimating future work, and to quantify business improvement trends. A measure of product size is also needed to support the definition or evaluation of digital engineering measures defined elsewhere in this document. This draft Product Size measure is offered to initiate this discussion across the industry and to advance a needed conversation toward industry consensus. It is more theoretical than practical, since limited experiential data exists. It is fully expected this definition will evolve over time, with the advantage that encapsulating a size measure here in this specification enables reference and reuse by other measurement specs with reduced impact on rework as digital engineering practices mature across the industry. The current proposed definition of digital engineering product size is based on the concept of **model elements** generated as an outcome of the modeling process, as described in section 2.2.2. Organizations may establish conventions for the model elements to be counted and analyzed for sizing, based on their applications, methodology, tools, or domain. Examples include structural (static) model elements, behavioral (dynamic) model elements, organizational model elements (packages, libraries), or annotational (descriptive) model elements. Refer to section 2.2.2 for additional details.
Relevant Terminology	Model Element See definition in section 3.

Information Need and Measure Description	
Information Need	What is the size and scope for the DE project or product? How much work must be done? How many functions and interfaces have been identified in the system architecture or design? How much is that changing? How does DE product size relate to estimates and measures of effort, cost, schedule, productivity, performance, or return-on-investment? Which model element category is contributing the most growth in the overall system model?
Base Measure 1	Model Elements: count of product size (planned and actual) [integer] *Organization or project-specific units and scope for what model elements are counted.*
Base Measure 2	Effort (Labor Hours): hours (planned and actual) [integer] *Estimated or actual effort in labor hours for the work to be designed or implemented.*
Base Measure 3	Duration (Calendar Months): months (planned and actual) [integer] *Length of the design or development effort (planned or actual) for the scope of work related to Product Size.*
Base Measure 4	Model Element Weight: weight [dimensionless] *A multiplier associated with an element representing relative effort due to reuse or complexity for the purpose of estimating cost, effort, or schedule.*
Base Measure 5	Reuse Savings: (Labor Hours): hours [integer] *Difference of effort between developing a system with reuse vs. developing it without reuse, or the difference of equivalent size as an effort proxy for a relative ROI analysis.*

Base Measure 6	Reuse Investment: (Labor Hours): hours [integer] *Difference of effort between developing a system for reuse across multiple systems vs. developing for a single system, or the difference of equivalent size as an effort proxy for a relative ROI analysis.*
Derived Measure 1	Productivity = (Model Elements) / (Effort) *Number of model elements generated per unit effort (e.g., model elements / labor hours)*
Derived Measure 2	Progress = (Model Elements$_{\text{actual to date}}$) / (Model Elements$_{\text{planned}}$) *Percent of planned model elements completed to date for characterizing progress and work remaining. Can also be used to characterize growth and stability (actual size vs. planned).*
Derived Measure 3	Throughput = (Model Elements) / (Duration) *Number of model elements completed per calendar period, e.g., model elements / calendar month. Can also be used to characterize a size vs. schedule relationship.*
Derived Measure 4	Equivalent Size = Σ (Model Element Weight * Model Elements) *Counts of each element type are weight-adjusted for relative effort as system size input to cost models. The weighted aggregate of size (model) elements account for effort due to degree of reuse (compared to new) and relative complexity of the size elements. Total equivalent size is used to estimate systems engineering effort, cost and schedule.*
Derived Measure 5	Return on Investment (ROI) = (Reuse Savings – Reuse Investment) / (Reuse Investment) *The reuse savings is the work difference between a non-reused and reused architecture across multiple systems which can be based on the equivalent size difference, and the investment is the additional work to make the system architecture reusable which can be reflected in the equivalent size difference.*

Indicator Specification	
Indicator Description and Sample	Indicators for Product Size will generally plot the size (number of model elements) over time and the relationship with effort and schedule, such as the example indicator concepts below.  **Figure 8.3-1: Model Size Trends** The indicator in Figure 8.3-1 plots planned vs. actual product size (number of model elements) over time. The original baseline estimate (light blue dashed line) was 840 model elements over a 12 month schedule (70 elements/month). Initial estimates early in the project were not yet well understood as requirements and

	scope were being defined, and the total estimate was increased (orange solid line) to 875 elements in month 3, and 900 elements in month 6. Change requests were received from the acquirer following a design review, and the scope was increased to an estimated 950 elements in month 9. Planned and actual modeling development progress by month (cumulative model elements completed) are plotted in the blue dashed line and green solid line, respectively. These indicate that modeling progress was behind plan for the first 8 months, but recovered in month 9 as the team identified reuse opportunities across separate modeling efforts and became more productive in leveraging capabilities of the modeling tools. However, the total count of modeling elements overall was under-estimated so the original completion date of month 12 was delayed a month to complete the additional modeling effort. Upon completion, the actual count of final modeling elements (920) was 80 more than the original plan (840), but 30 less than the last estimate (950).
Indicator Description and Sample (continued)	 **Figure 8.3-2: Model Size - Estimate Accuracy** The indicator in Figure 8.3-2 depicts the accuracy of model size estimates (planned vs. actual number of model elements) for a sequence of iterations. The tendency has been to under-estimate the quantity of model elements needed (by 11% overall), which can lead to challenges meeting budget or schedule.

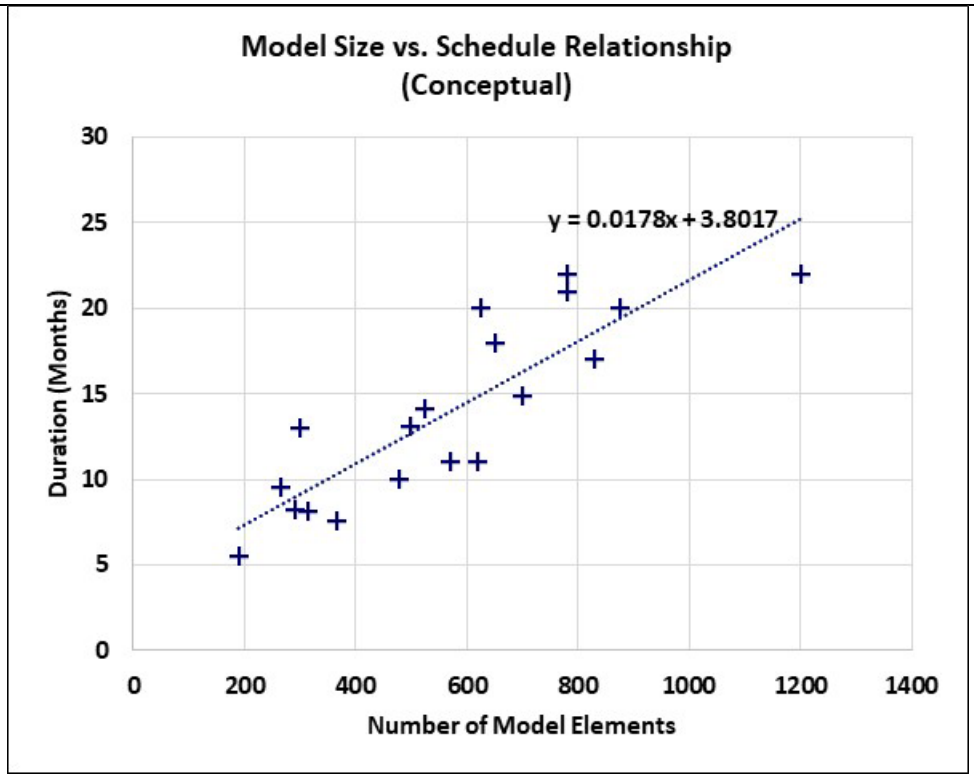


Figure 8.3-3: Model Size versus Schedule Relationship

The indicator in Figure 8.3-3 plots the size (number of model elements) of a set of products vs. their schedule duration. This example depicts a fairly consistent correlation between model size and schedule. A similar indicator might be plotted for size vs. effort (hours or person-months). This relationship might be used to validate schedule estimates for future development components based on their model size.

Indicator Description and Sample

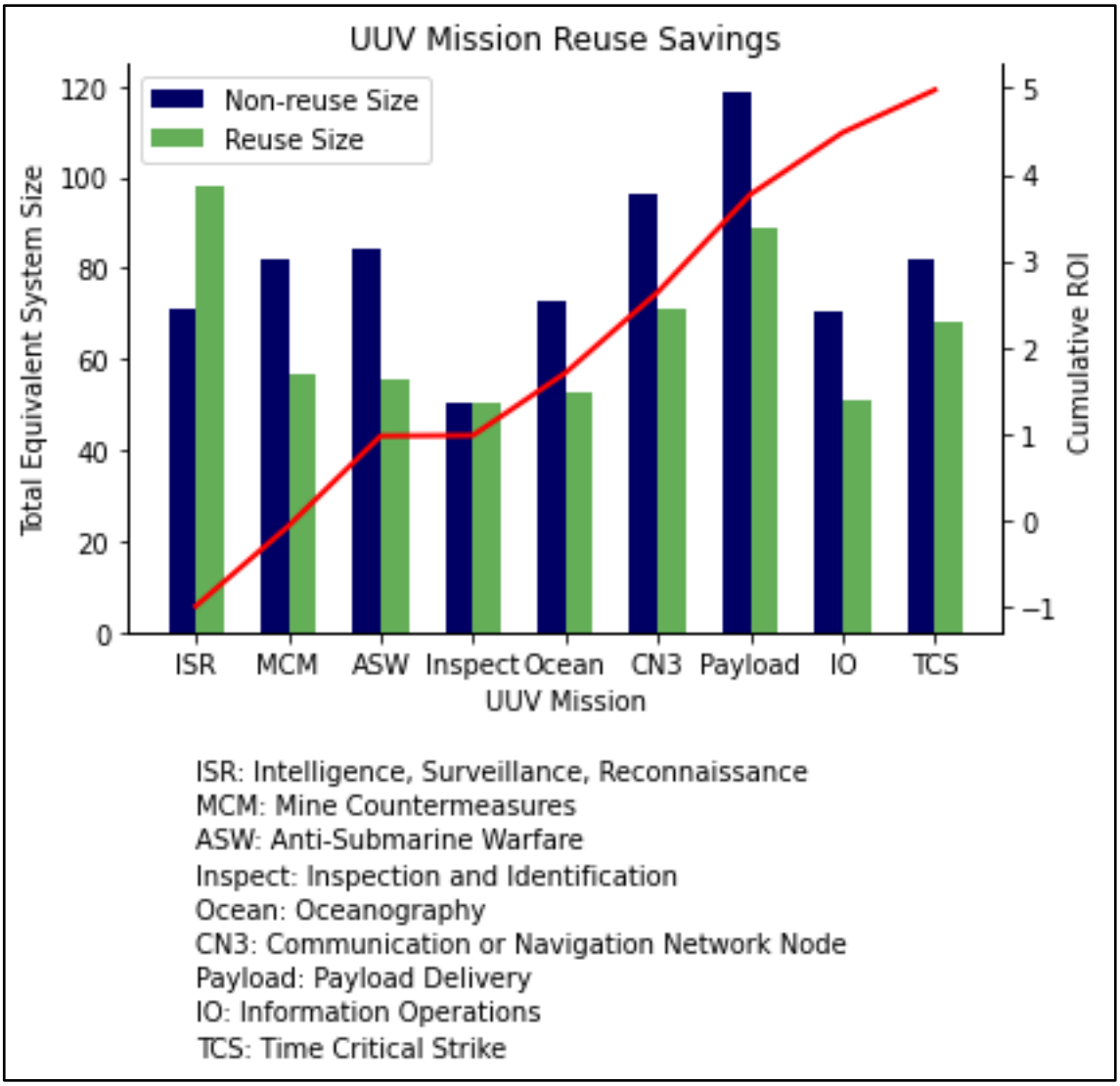


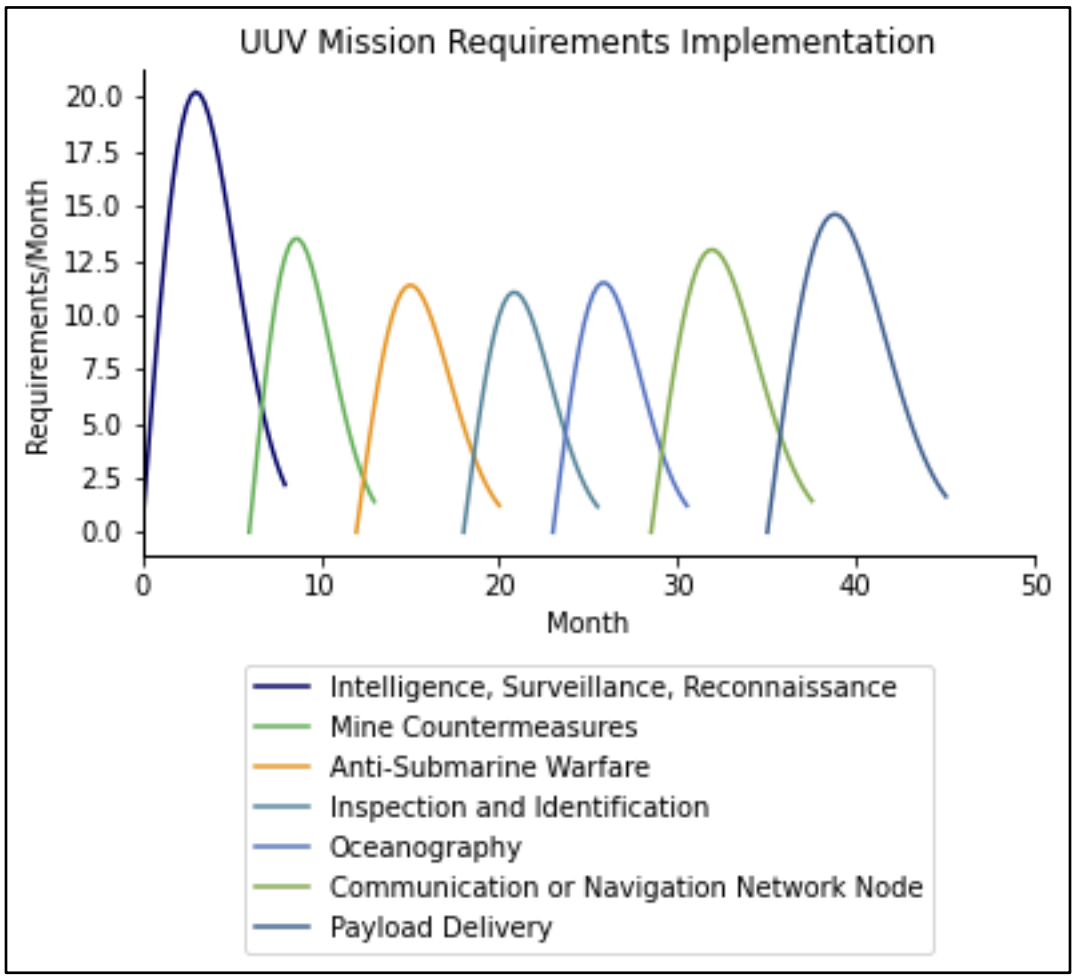
Figure 8.3-4: Reuse Savings and ROI

Figure 8.3-4 displays the total equivalent system sizes and resultant ROI of a product line approach for UUV systems with overlapping mission capabilities. The size is measured automatically from a product line requirements model set as an aggregate count of model requirements, interfaces, algorithms and scenarios weighted by equivalent size for reuse categories. Note the first mission ISR is the product line baseline designed for reuse which is the investment cost (reflected in additional equivalent size). The savings for subsequent missions are the differences between a traditional non-reuse approach and the product line reuse approach.

The cumulative ROI is the net savings over time divided by the investment cost based on the relative sizes. A positive value indicates a good investment and a value of unity indicates a net savings equal to the investment. The final cumulative ROI indicates a substantial 500% net savings compared to the investment and is the decision criteria to move forward with the UUV product line. The planned implementation of the requirements is shown in Figure 8.3-5.

The size is used as input to systems engineering cost models to quantify estimated costs. Current cost models, including the Constructive Systems Engineering Cost Model (COSYSMO) applied here, use model elements for system requirements, interfaces, algorithms and scenarios (use cases), which are outputs of the systems engineering process. The equivalent size difference represents a work savings, and

	added equivalent size represents the additional work investment to make the UUV baseline reusable. These explicit size measures are also aligned to viewpoints for tracking progress shown in Figure 8.3-5 next for the requirements.

Indicator Description and Sample	 **Figure 8.3-5: Planned Model Requirements Implementation** This indicator displays the planned systems engineering requirements implementation over time for selected UUV mission systems in Figure 8.3-5. These top-level requirements were measured automatically from the product line requirements model set which comprise a portion of the total equivalent size in the reuse savings in Figure 8.3-4. The phased implementation over time is derived from the COSYSMO effort and schedule model using requirements and other size elements directly measured in the model set as size inputs. A Rayleigh staffing curve and implementation rate is assumed for the plan with a specified degree of mission overlap. The missions are strategically planned in time by warfighter functionality needs and the phasing overlap is due to staffing dependencies. This indicator shows the planned implementation based on the decision to go forward with adequate ROI per Figure 8.3-4. It would be augmented with actuals to track progress during implementation. It only covers requirements and the planned implementation for interfaces, algorithms and use cases can tracked similarly.

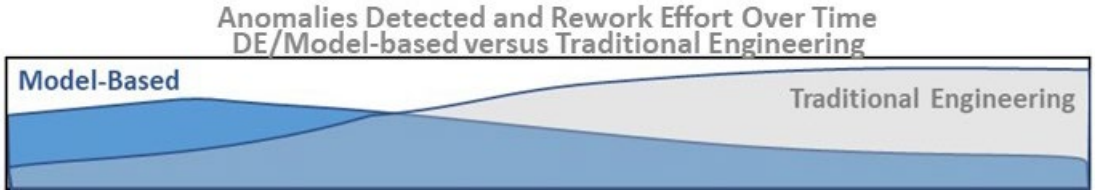
Analysis Model	Throughout the product development, compare the actual size of completed digital engineering products (count of model elements) versus plans and estimates, and consider the causes for deviations and if adjustments to plans are necessary. • Are model element counts for completed components consistent with engineering plans? • Are model components and elements being completed at a rate needed to meet progress, cost, and schedule? • What are the reasons for deviations in model element counts vs. the plan? Was work mis-estimated or misunderstood? Were there changes in the scope of work? Was work more complex than expected? • If actuals deviate significantly from estimates, do plans need to be adjusted for current or future work? Over time, a historical database of digital engineering modeling attributes (size, cost, effort, schedule) can be established across projects and used to inform estimates for future projects. For example: • Is the new project similar in size and scope as historical projects? • Can the actuals from those completed projects be used to develop or validate estimates on the current project? • Are projects becoming more productive? Are effort, cost, schedule, and efficiency improving? • How does technical debt contribute to product size? Is the technical debt decreasing over the life cycle? • How do size estimates change after a design is refactored or replaced? Size elements of the system are also used as normalization measurement quantities in Product Quality indicators (e.g. to normalize defect levels for comparison across systems).
Decision Criteria	Variances in model element counts exceeding defined thresholds (e.g., $\pm$ 10%) should dictate reconsidering the feasibility of current plans, estimates, or resources. An ROI greater than 1.0 typically justifies a go-ahead process decision.

Additional Information	
Additional Analysis Guidance	As modeling efforts complete, track actuals vs. historical performance in size estimates for potential use in predicting future performance. While size is often used to assess productivity, schedule, or effort, it is important to take care not to incentivize actions that create an unwieldy architecture (many more elements than needed) or push work on complex elements downstream to complete enough elements to appear "on schedule". The Product Size measures can be used with the Architectural Completeness and Volatility measures to assess increasing size and volatility of the model elements, and their subsequent impact on other project issues. Note that counts of model elements are likely to be specific to a given product or project and not directly comparable across projects due to varying modeling conventions, counting rules, estimating standards, or modeling tools. It is also likely that other attributes, such as domain, complexity, or model element type will factor into the relationships between product size and effort required to implement. Use of model size measures at the enterprise level is therefore likely problematic until modeling conventions, counting rules, and normalization standards are consistently applied across projects. Size is just a means to an end. Accurate size counts are not the primary objective of this measure. Size is a proxy for the amount of work to be performed, and enables accurate estimates and is useful for determining if feasible plans are in place including having adequate resources, effort, schedule, and cost. Product size is a basis for many other indicators, such as productivity, rework, cost and schedule estimating relationships, and other derived measures.

	If using size as a basis for other indicators, complexity of each model element is also a consideration. One way to evaluate complexity is use the complexity drivers from COSYSMO with DE additions. These drivers count the number of model elements, symbols, interfaces, interactions, diagrams, and requirements.
Implementation Considerations	A count of model elements can typically be obtained from project modeling tools. Product sizes may vary based on the tool and sizing methods used, therefore it is recommended that a project use the same set of tools to ensure consistent results. It is important to consider what will count as "Done", and only count completed model elements against this criteria.

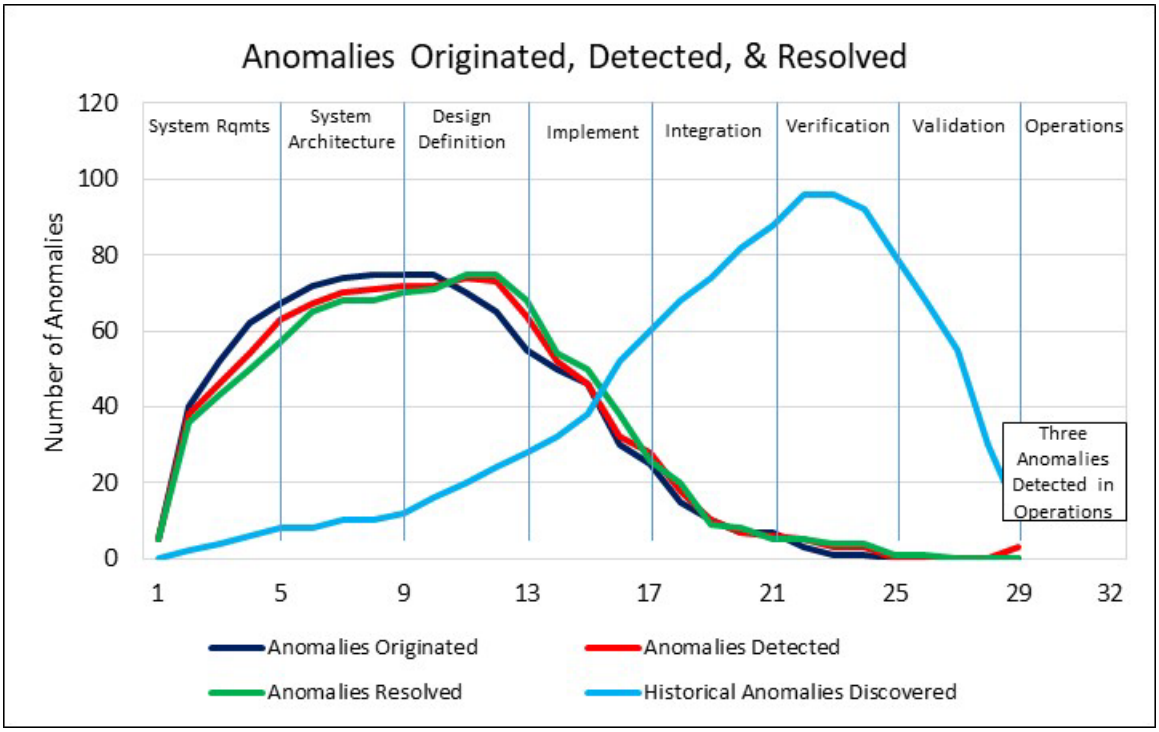
Additional Specification Information	
Information Category	Size and Stability
Measurable Concept	Functional Size and Stability
Relevant Entities	Model components
Attributes	Model element type (e.g., requirements, architecture, design) Model element category (e.g., structural, behavioral, organizational, annotational)
Data Collection Procedure	Counts of model elements and type are typically provided by modeling tools. Queries, scripts, or APIs may be available to automate the collection of model element count measures.
Data Analysis Procedure	Regularly analyze stability of model size and growth trends against plans and decision criteria (weekly, monthly), taking corrective action as needed to bring plans back into alignment.

8.4 DE ANOMALIES

Measure Introduction	
Description	One of the major benefits expected from digital engineering is improved system quality, and early detection of any defects, when they are less costly to correct. The terms used to discuss quality vary widely across enterprises and projects. For the purposes of this specification, we will use the term *anomaly* to discuss deviations from expectations. In practice, anomalies may be documented as action items, peer review comments, defect reports, problem reports, errors, warnings, exceptions, or other items. The level of formality of reporting depends on where a project is in the lifecycle, and the lifecycle model used. Early anomalies are often tracked informally (if at all), and may not be recorded in a defect repository. Generally, once the model element has been transition to a different team, then formal tracking of anomalies begins. Once an anomaly is investigated and determined to be a defect, then they are generally documented in a defect repository. A project may also separately track change requests (also called corrective actions or engineering change orders) to track changes required to address an anomaly. Anomalies are typically collected, analyzed, and monitored across life cycle process activities or boundaries, such as phases, stages, iterations, or releases. For DE, a key quality objective is to detect and resolve anomalies as early as possible, containing them to their source activity rather than allowing them escape to subsequent activities where they are more costly to correct. This is enabled by DE with its emphasis on early architecture and design activities, shifts toward earlier verification and validation (V&V), increased traceability between model elements, increased product automation, and strengthened testing leading to reductions in anomalies and rework. This may be accomplished through anomaly detection processes such as effective peer reviews, modeling, simulation, automated testing throughout development, and other verification and testing approaches. Figure 8.4-1 depicts the leftward shift of focus, in DE and model-based engineering, toward earlier detection and removal of defects. The figure, which is notional, depicts the number of anomalies detected over time, as well the amount of rework effort required over time. The concepts apply to any life cycle process model (waterfall, iterative, incremental, etc.).  **Figure 8.4-1 Anomalies Detected and Rework Effort Over Time**
Relevant Terminology	

Information Need and Measure Description	
Information Need	Is the quality of the product in question adequate for the product to be used in subsequent phases or activities? Are we finding and removing anomalies early in the life cycle using models and shared information? Is the use of DE leading to the detection of anomalies earlier in the lifecycle compared to traditional methods or projects)? Is the use of DE resulting in fewer overall anomalies? Are certain activities generating an extraordinarily high (or low) number of anomalies? How can DE and modeling efforts be improved to reduce the leading causes of anomalies?
Base Measure 1	Anomalies Originated [integer] *Number of anomalies introduced into the system or product at the point of origin, over time*
Base Measure 2	Anomalies Detected [integer]

	Number of anomalies detected and tracked over time
Base Measure 3	Anomalies Resolved [integer] *Number of anomalies that were resolved and tracked to closure over time*
Base Measure 4	*Anomalies by Category [integer]* *Number of anomalies assigned or filtered to selected categories* *Categories include groupings of anomalies meaningful to a project. See, for example, figure 8.4-5, or the attributes listed in Additional Specification Information.*
Derived Measure 1	Historical Anomalies Detected [integer] *Median trend of anomalies detected over time from enterprise projects*
Derived Measure 2	Anomalies Open [integer] *Anomalies Detected – Anomalies Resolved (includes anomalies deferred)*
Derived Measure 3	Escaped Operational Anomalies [integer] *Number of anomalies that are open at the time of delivery) (includes anomalies deferred)*
Derived Measure 4	Anomaly Distribution = (Anomalies by Category) / (Anomalies Detected) [percentage] *Percentage of total anomalies by category*

Indicator Specification	
Indicator Description and Sample	Many potential indicators can be used to manage defects and product quality. This specification illustrates a few that have been adapted for digital engineering and MBSE.  **Figure 8.4-2: Anomalies Originated, Detected and Resolved (non-cumulative)** In the indicator in Figure 8.4-2, anomalies are tracked from when they are introduced into the system (Anomalies Originated), to when they are detected (Anomalies Detected), and finally, to when they are resolved (Anomalies Resolved). These measures, collected from a project using DE, are plotted over time in Figure 8.4-2, along with the enterprise benchmark of Historical Anomalies Detected (using traditional engineering methods).

In this example, anomalies originated and peaked early in system architecture and design definition phases. With DE, most anomalies were detected and resolved soon after being introduced, as indicated by the closely tracking anomalies originated, detected, and resolved lines. In contrast, the historical pre-DE processes detected anomalies much later, with detection peaking during the Integration and Validation phases. This provides evidence that the DE process was successfully used to identify and resolve anomalies earlier in the development process.

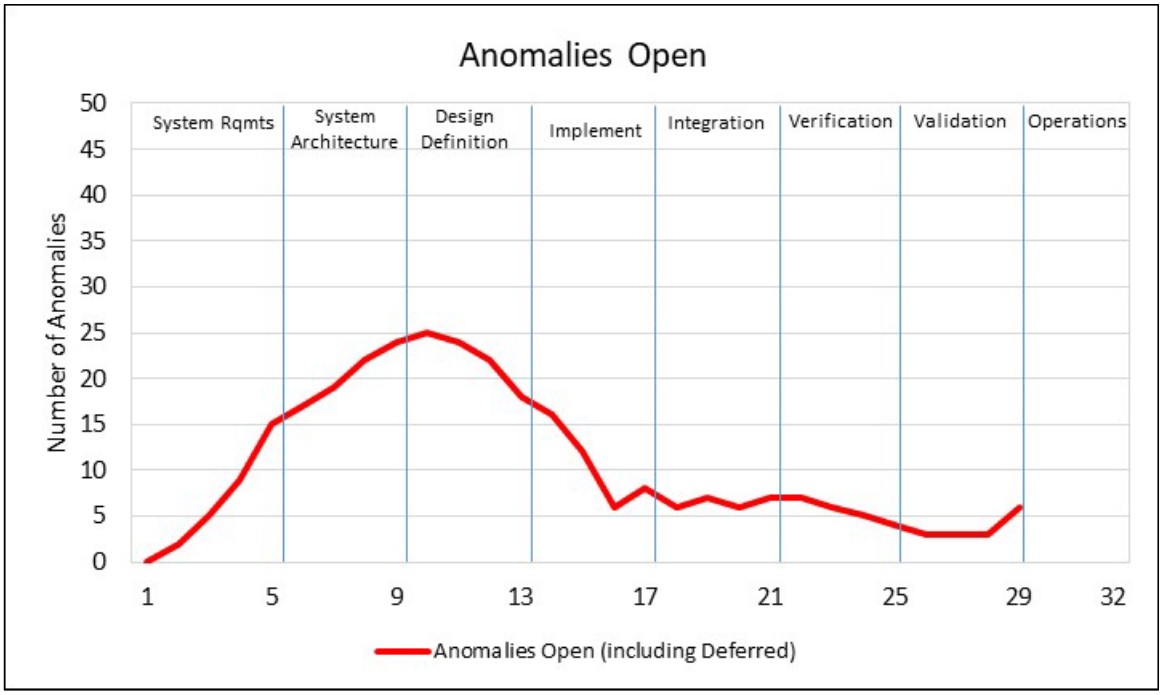


Figure 8.4-3: Anomalies Open

Figure 8.4-3 shows the number of anomalies open over time. The number of anomalies open peaked during Design Definition, and decreased during Implementation. This shows that most of the anomalies were resolved shortly after being detected, except for some interface anomalies that were outside of the program manager's control. These were deferred until the product's next iteration. These open anomalies resulted in three additional anomalies when the product entered Operations. Anomalies still open (deferred or not resolved) at the time of delivery to Operations represent Anomalies Escaped, and provides feedback on the technical debt and associated risk going forward. Another three anomalies were detected during operations.

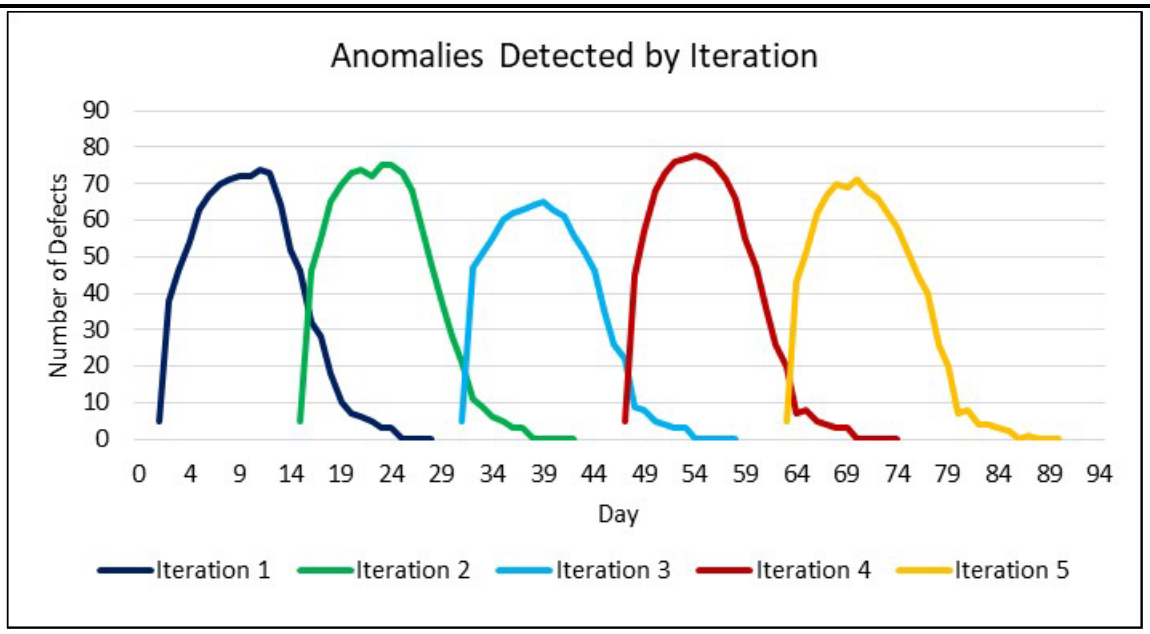


Figure 8.4-4: Anomalies Detected over Several Iterations

By repeating this process across multiple iterations or projects (Figure 8.4-4), the impact of the DE efforts to detect anomalies can be assessed. Trends that can be assessed are whether anomaly detection is shifting to earlier phases/activities in the lifecycle and where the number of anomalies are peaking. This indicator shows that anomaly detection has shifted slightly to earlier lifecycle phases over the course of 5 iterations.

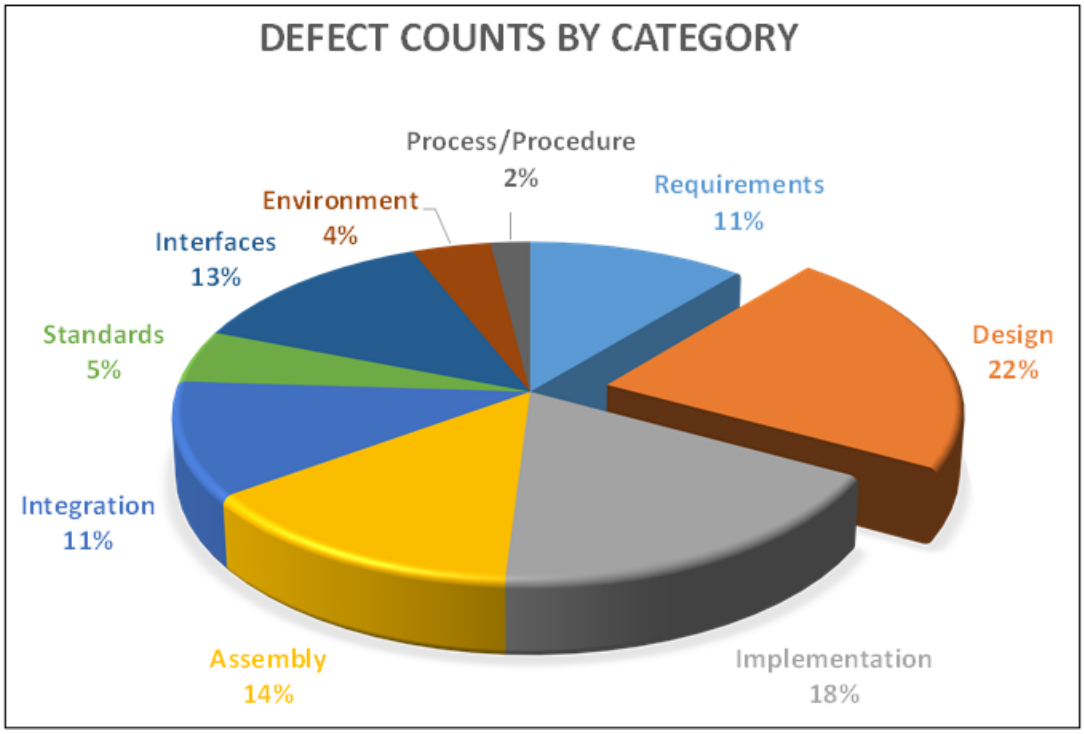


Figure 8.4-5 – Anomalies Detected by Category

In Figure 8.4-5, the project saw that the highest proportion of anomalies was associated with design. The project did a similar analysis of rework effort by anomaly category (not shown here), which substantiated that design anomalies required extensive effort to resolve and were having the greatest impact on the

	project. To address this issue, the project implemented improvements to its design modeling guidelines and provided training to designers on the effective use of their modeling tool. Peer reviews were conducted to help improve design quality and reduce future anomalies and rework in future iterations.
Analysis Model	Through use of DE, anomaly counts are expected to rise and peak in the earlier architecture and design phases, and should decrease over time. Increases or spikes in anomaly counts should be analyzed to determine the root causes. Evaluating the difference (gap) between when anomalies originate and when they are detected should provide information on the effectiveness of the DE processes. Once anomalies originate and are detected (Figure 8.4-2), analysis will typically evaluate whether they were resolved in an expeditious manner. The concept of resolving anomalies within a phase is a key measure for efficiently managing product quality. Ideally, an anomaly would be resolved in the same phase as it was detected to reduce the impact of problems in downstream activities and reduce rework. Anomalies that are not resolved or planned to be resolved after multiple iterations may represent a risk to the inherent quality of the product. Anomalies that are not detected shortly after they originate may represent an issue with effectiveness of the work product review and anomaly detection processes. Periodic analysis of anomalies by category can help isolate and prioritize where to best invest effort in corrective improvement actions. Anomaly counts and distribution by category are typically analyzed to determine which anomaly categories are most prevalent, and which cause the greatest rework impact on the project. Analyzing this data across a set of phases or projects can identify systemic trends. Analyzing the sufficiency of the DE review, modeling, simulation, and automated test processes can help determine the root causes of the underlying issues. Corrective actions and process improvement efforts can then be taken to detect and resolve anomalies earlier in the lifecycle, and even prevent anomalies from being introduced in the first place.
Decision Criteria	Are the anomaly detection and resolution rates acceptable, as defined by the project or enterprise? Are the rates improving over time? Does the anomaly resolution rate meet project quality objectives (e.g., are at least 90% of anomalies resolved in the iteration where they originated?)

Additional Information	
Additional Analysis Guidance	In order to facilitate resolution of the anomalies efficiently and effectively, and to determine which anomalies get resolved first, additional indicators of various anomaly attributes may be utilized. Efforts should focus on the high priority anomalies (as determined by the user/acquirer) and ensure they are being resolved in a timely manner. However, this needs to be balanced against the anomaly category, the complexity of the fix, and the level of effort required to resolve each anomaly. Focusing too much attention on anomaly resolution alone could erroneously lead to the easy anomalies being fixed first to make the closure rate look good, while deferring the more challenging anomalies to later phases, or until they can't be put off any longer. Indicators for anomaly aging would expose the tendency to push off difficult anomalies. Seldom does one indicator alone provide sufficient insight to analyze the root cause of anomalies or quality issues and determine where to take corrective or improvement actions. Many other anomaly analysis indicators (see Attributes) can be used in combination to obtain a full picture. Indicators can also be filtered by selected anomaly attributes to drill down to analysis details where needed. The root causes or reasons for anomalous defect trends should be understood what (or if) actions are necessary. Refer to the Affordability and Rework indicator for additional guidance on managing changes to digital modeling work products (corrective, adaptive, perfective). Anomaly resolution is an example of corrective action. Relative to traditional development, DE/MBSSE activities should result in a shift of anomaly detection and resolution to earlier iterations of development and particularly should reduce the number of unresolved anomalies in a release iteration. At this early point in DE measurement, programs have seen around a 20% reduction in release anomalies using DE processes from earlier experience. Programs

	should evaluate their use of data, models, and DE/MBSSE processes with respect to improvements over time in the quality (number of anomalies) of releases. DE/MBSSE should result in early verification and product specification completeness in earlier lifecycle iterations accomplished via models and digital system views. Programs may want to pay particular attention to the number of anomalies detected and resolved early when implementing model-based versus document-based reviews and approval processes. Additional measures related to anomalies detected, specifically in review processes, may be of interest in the movement from document to model-based reviews.

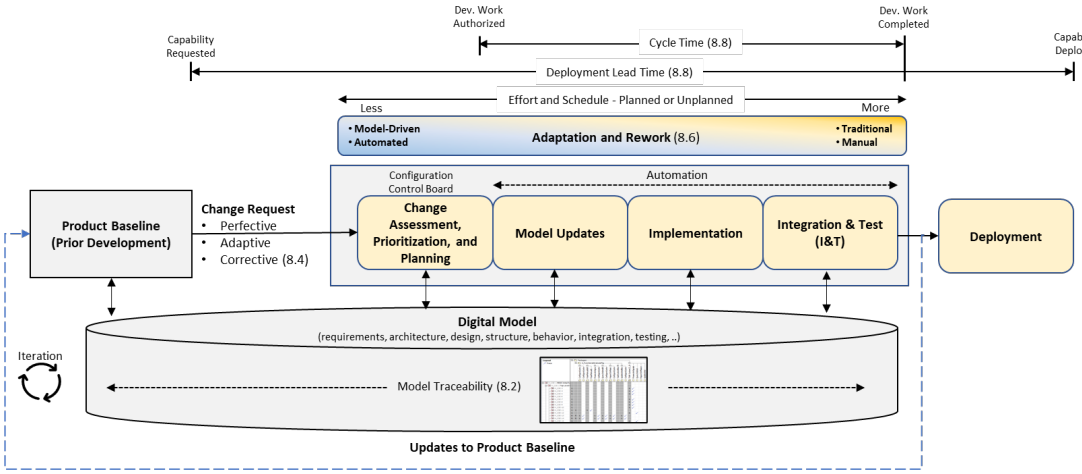
Implementation Considerations	Business systems, models, and tools must be configured and instrumented to collect the measures and attributes needed for anomaly data collection and analysis, as tailored from this specification.
	
	Anomalies recorded in the defect repository must be discernable whether they were detected before or after the in-process iteration or work activity. In addition, in a model-based process, anomalies should be assessed for the internal development team iterations. A parameter or a review of the dates could be used to determine if anomalies were contained or escaped.
	
	Counting methods and standards should be defined to determine:
	
	• What constitutes an anomaly
	  - e.g., peer review findings not considered internal anomalies
	  - e.g., an internal error that is sent back to the originating team and results in rework, may be considered an anomaly
	• When anomalies will be counted (e.g., upon hand-off to another team/3rd party)
	• Classification of internal vs. external anomalies (e.g., anomalies discovered by the supplier, by the acquirer in an operationally representative environment, or by the acquirer in operations)
	
	Resolution of detected anomalies should be prioritized and scheduled based on severity and the effective use of resources. Some projects may trade off quality and progress based on impact to the project and end user. In an iterative approach, certain lower priority anomalies may be placed in a backlog for resolution in a subsequent iteration. Refer to the PSM Measurement Framework for Continuous Iterative Development[13] for additional guidance.
	
	Proper analysis of anomaly detection and resolution measures must be performed by looking at the relationships between the measures. At some point, you have to ask people questions about the measures vs. relying on the numbers in isolation.

Additional Specification Information	
Information Category	Product Quality Process Performance
Measurable Concept	Functional Correctness Process Effectiveness
Relevant Entities	Anomalies
Attributes	There are many attributes associated with a defect repository, and any of these can be utilized for development of measures. Any attributes used should be adapted to a DE environment. The attributes will vary and should be defined according to the business domain, application, product, and quality objectives, but a robust quality management data set might include attributes such as the following:

Defect Data	**Example Attributes**
Anomaly dates	date detected or opened; date closed or resolved; anomaly state transition dates

<table>
<tr><td></td><td>
<table>
<tr><td>Anomaly state</td><td>submitted; open; analysis; in work; on hold; resolved; closed; deferred</td></tr>
<tr><td>Identification of work activities where anomalies occur
(originated, detected)
(assigned relative to the products being changed)</td><td>Stages: Concept; Development; Production; Utilization; Support; Retirement (ref. ISO/IEC/IEEE 24748-1).
Processes: such as those defined by ISO/IEC/IEEE 15288:2015 (e.g., Stakeholder needs and requirements definition; System requirements definition; Architecture definition; Design definition; Implementation; Integration)
Life Cycle Phases: System Requirements; System Architecture; Design Definition; Implementation; Integration; Verification; Validation
Iteration: build; iteration; release
Work Decomposition: (ref. Figure 2-1) Digital Infrastructure; Lifecycle Models; Process Models (gates); Data and Model Ontology; Operational Data & Models; System Data and Models; Discipline Specific Data & Models</td></tr>
<tr><td>Work product type</td><td>model; model element type; drawing; software; hardware; firmware; COTS; unit; component; assembly; chassis</td></tr>
<tr><td>Product identifier</td><td>version; increment; iteration; build; release; model number.; serial number.</td></tr>
<tr><td>Anomaly category</td><td>requirements; design; implementation; assembly; integration; standards; interfaces; environment; process/procedure</td></tr>
<tr><td>Anomaly severity</td><td>numeric scale (e.g., 1-5), or relative (low, medium, high, critical)</td></tr>
<tr><td>Customer impact</td><td>internal; external. Does the anomaly visibly affect the end user or deployed product?</td></tr>
<tr><td>Rework effort</td><td>hours, days, or established ranges (buckets) of rework labor (estimated, once impact assessment is complete; and actual, when rework is completed)</td></tr>
</table>
These are examples only, intended to be representative ideas but not complete sets. There is currently no consensus, convention, or standard for defining and classifying anomalies specific to DE or MBSSE. Enterprises and projects should tailor these potential attributes to their particular use case. Sufficient data should be collected in a defect repository to enable any analysis that needs to occur.
</td></tr>
<tr><td>Data Collection Procedure</td><td>Data is collected in a defect repository, as part of the quality management process. As an anomaly is identified and recorded, its specified attributes are included in the record. As anomalies are resolved, this information is added to the repository. The records should be tagged in a timely fashion with the selected attributes or relevant information.
Measurement begins with applying the measurement function to collect the base measures from the defect repository.</td></tr>
<tr><td>Data Analysis Procedure</td><td>Iterations in which anomalies are originated, detected, and resolved are discussed during the anomaly resolution meetings. Data is analyzed periodically (e.g., weekly, monthly; the analysis period should be stated) and at the end of each iteration or release according to the analysis model. Questions to be asked include, “Do we have the needed data to complete the analysis for these indicators? What observations or limitations regarding the data do we need to convey along with the indicators?”
Anomaly detection and resolution data is presented in the form of indicators with interpretation guidance. As needed, additional data may be supplied to support decision making. The resultant measures are used as a criterion for evaluating completeness at iteration gates, release readiness, iteration planning, and associated reviews.</td></tr>
</table>

8.5 ADAPTABILITY AND REWORK

Measure Introduction	
Description	Relative to traditional methods, model-driven approaches can enable greater resilience and adaptability to changes or maintenance of engineering work products (e.g., requirements, architecture, design, integration, testing) when they occur. Following an initial up-front investment, models and traceability to work products can be leveraged to reduce time and effort for implementation, maintenance, defect correction, and other modifications or rework. Change types can include: • **Corrective actions**: repair or mitigation of system anomalies or defects that risk or prevent the work product from meeting its intended planned purpose • **Perfective actions**: planned and scheduled enhancements to system products or services to implement improvements as a result of changes to requirements or user mission needs • **Adaptive actions**: adapting system configurations to support other applications, systems, environments, or iterative refinements. Traditionally, rework measures are focused on the effort to implement corrective actions for repair of defects. This is addressed in detail in 8.4, DE Anomalies. Here we envision the broader use of rework measures enabled through digital engineering to include change management, adaptability, and impact assessment contexts beyond simply the correction of defects. These are typically driven by change requests, under the governance of a Configuration Control Board (CCB) or equivalent authorizing changes. In a digital engineering environment products are model-driven, providing additional opportunities to cost-effectively incorporate changes to digital models that are directly traceable to the implemented and tested work products, some of which can be automatically generated. Digitally engineered work products can therefore be more resilient to changes of all types described above with reduced rework effort for work products and model elements, whether planned (intentional, perfective or adaptive changes) or unplanned (correction of defects). Rework is typically measured in terms of the effort or schedule needed to implement the change action. These concepts for efficient model-based adaptation and rework are illustrated in Figure 8.5-1 below. Other relevant digital engineering workflow-related measures are also depicted for overall context.  **Figure 8.5-1: Digital Engineering Rework** These concepts align with the SERC causal analysis described in Table 1-1, including measurable benefits such as ease to make changes, customize designs, and reduce rework effort. Although limited quantitative data of rework measures for model-based development currently exists, SERC research of industry literature cites reduction in defects, rework effort, rework cycles, percent rework, and technical debt as expected benefits.[16]

Relevant Terminology	Rework	Action taken to bring a defective or nonconforming component into compliance with requirements or specifications. *[IEEE SEVOCAB, PMBoK]* The effort or schedule needed to implement changes to digitally engineered work products, including corrective, perfective, and adaptive change actions. *[PSM Digital Engineering Measurement Framework]*

Information Need and Measure Description	
Information Need	How much rework effort is spent maintaining planned or unplanned changes to DE work products across the life cycle? Can changes to engineering work products be implemented more efficiently and with less effort in a DE environment relative to traditional methods?
Base Measure 1	Change Requests: Number of change requests to baselined work products, by change type [integer] *(see Attributes for other change request or defect characteristics useful for analysis or filtering)*
Base Measure 2	Model Elements Changed: quantity of model elements affected by the change request [integer] *(refer to Product Size measurement specification)*
Base Measure 3	Rework Effort: labor effort expended to implement a change request [integer]. Units: hours or equivalent.
Base Measure 4	Rework Cost: cost of rework expended to implement a change request (labor, material) [currency, e.g., dollars]
Derived Measure 1	Cumulative Changes: = Σ (Changes) [integer] *(total number of changes for the selected data set, filtered by change type and attribute)*
Derived Measure 2	Cumulative Rework Effort = Σ (Rework Effort) [integer] *Sum of rework effort for the selected change record data set.*
Derived Measure 3	Statistical analyses of rework correlated with selected change record attributes. *Examples: mean, median, variance, standard deviation, quartiles, correlations, outliers.*

Indicator Specification	
Indicator Description and Sample	Rework measures from traditional approaches (e.g., rework by stage or activity, percent of rework, Cost of Poor Quality) can be adapted and applied to digital engineering contexts and compared to legacy measures to assess measurable model-driven benefits. Rework analyses are often conducted across a set of many change requests, perhaps in affinity groupings or filters selected by product component, change request type, priority, or other parameters (see Attributes for additional examples). Indicators such as histograms, scatter diagrams, control charts, box charts or other indicator types can be used to collect and analyze a set of changes by attributes such as effort (e.g., hours), resources (e.g., full-time equivalent (FTE) staff allocated), cost ($), or schedule impact (hours, days, weeks). Such data for MBSE or DE rework is not yet consistently available in practice, so the indicators below are conceptual examples with artificial data for illustration only.

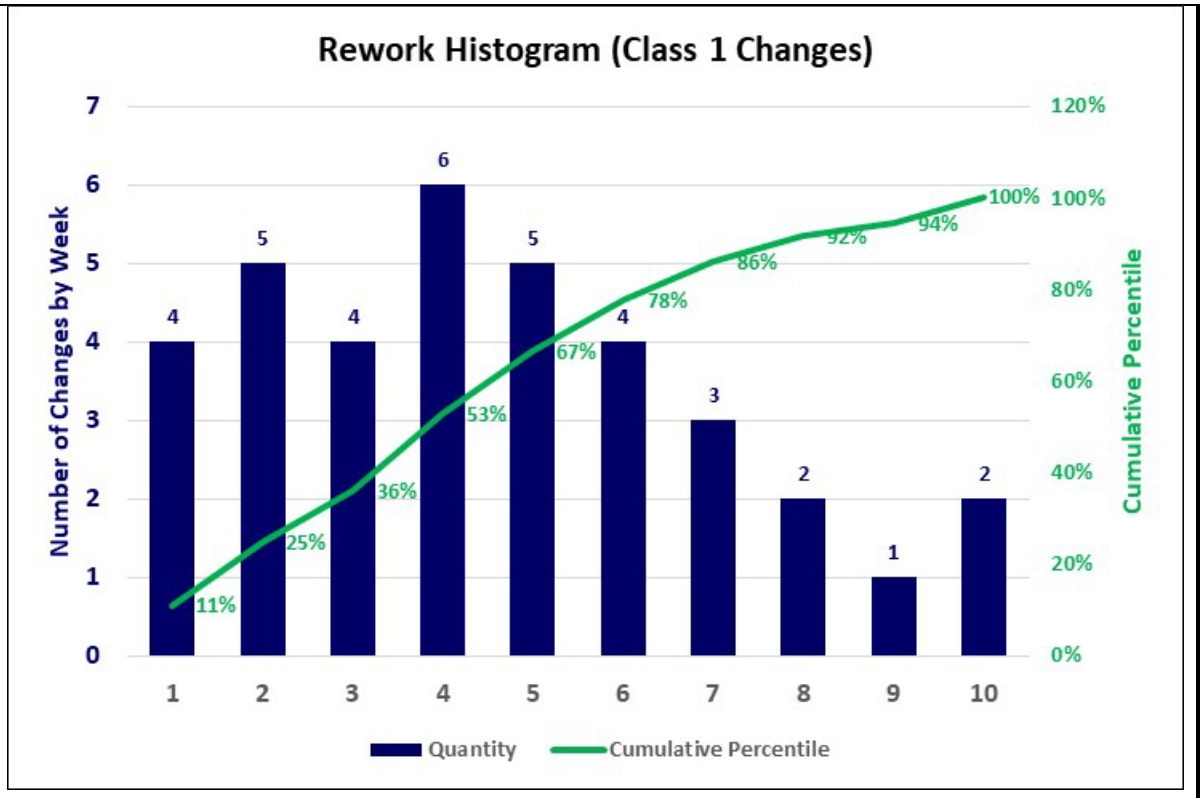


Figure 8.5-2: Rework

In this example in Figure 8.5-2, rework for Class 1 changes (planned modifications, or unplanned defects) is analyzed in a histogram by weeks of schedule duration to implement and test the change, including updates to associated models, work products, documentation, reviews, verification, and regression testing of changes. Twenty-five percent of Class 1 changes were completed within 2 weeks; over half within 6 weeks. This is a top level summary for a rework analysis; based on the distribution of rework and comparison against project plans/objectives, additional deep dive analyses may be needed to identify root causes and areas for further investigation or corrective action. The cumulative distribution series plotted on the right axis can be used to develop schedules for rework estimates or for establishing Service Level Agreements (SLAs), such as 75% of change requests completed within 6 weeks.

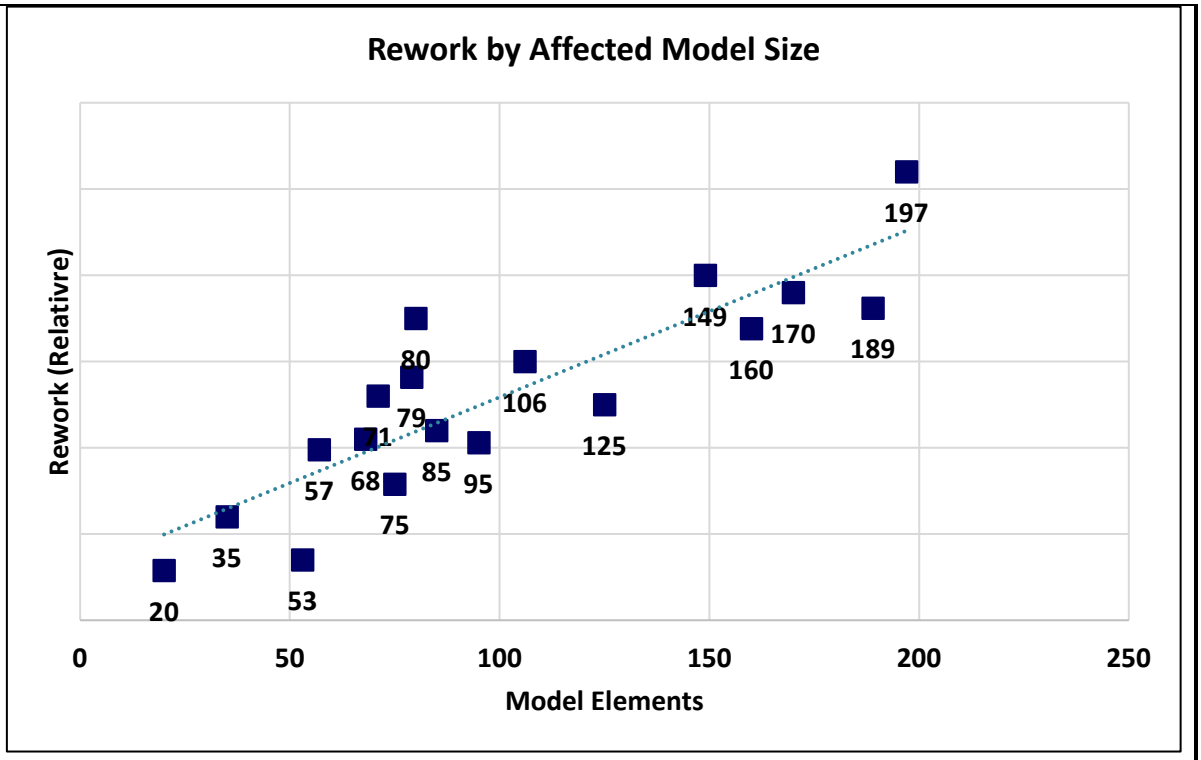


Figure 8.5-3: Rework by Affected Model Size

Individual changes can also be plotted and rework effort correlated with other factors or attributes. In Figure 8.5-3, a scatter diagram is used to plot rework effort vs. the size of the change, specified in terms of product size (affected model elements). The vertical axis for the scale of rework effort has been intentionally excluded in this conceptual example since no actual model size vs. rework data is available to the authors, and to depict an actual size vs. rework effort relationship would be inaccurate and misleading. Conceptually, this type of analysis indicator might be used with actual data in the future to determine estimating relationships for rework vs. change size, or to further investigate anomalies outside expectations.

Analysis Model	Analyze measured change effort (rework – planned or unplanned) for the selected data set and filtered attributes. Look for correlations, trends, and indicators that can be used to investigate rework anomalies, systemic issues, root causes for improvement actions, or develop models that can be used to manage future performance and rework. Example analyses include: • Rework trends: Is the amount of rework appropriate to the size of the change effort and change type? Is the normalized relative rework increasing or decreasing? Is rework within expected bounds? • Rework distribution: Plot the distribution of rework by work product, activity, or life cycle stage. Is the distribution of rework effort as expected or are there areas that need further analysis? Is the model-driven approach leading to less rework relative to traditional development, with rework effort shifting from later to earlier life cycle activities as anticipated?
Decision Criteria	Establish data set thresholds, performance targets, and tolerances for the range of expected rework based on change type and selected attributes. Measures of rework outside expected performance should trigger further investigation. Assess rework measures and trends against project plans (cost, schedule) and determine if adjustments to the plan are needed.

Additional Information	
Additional Analysis Guidance	Evaluate defect rework in conjunction with other defect measures. Other relevant and complementary measures from this PSM digital engineering framework include: • DE Anomalies • Functional Architecture Completeness and Volatility • Model Traceability • Product Automation Refer to these corresponding measurement specifications for additional analysis guidance. Projects or organizations with a robust collection of historical data from past projects may be able to analyze the measurable benefits of model-based development relative to prior traditional development projects. Being able to differentiate a new defect problem type from older defect problem types will enable better analysis of emergence or business changes, as well as assessing any corrective action effectiveness.
Implementation Considerations	Business systems, models, and tools must be configured and instrumented to collect the measures needed specific to the change effort, as tailored from this specification. Consider filtering the defects by some classification model, enabling separating defects in stories, model or code. This prevents conflating different types of defects too early in the specification hierarchy.

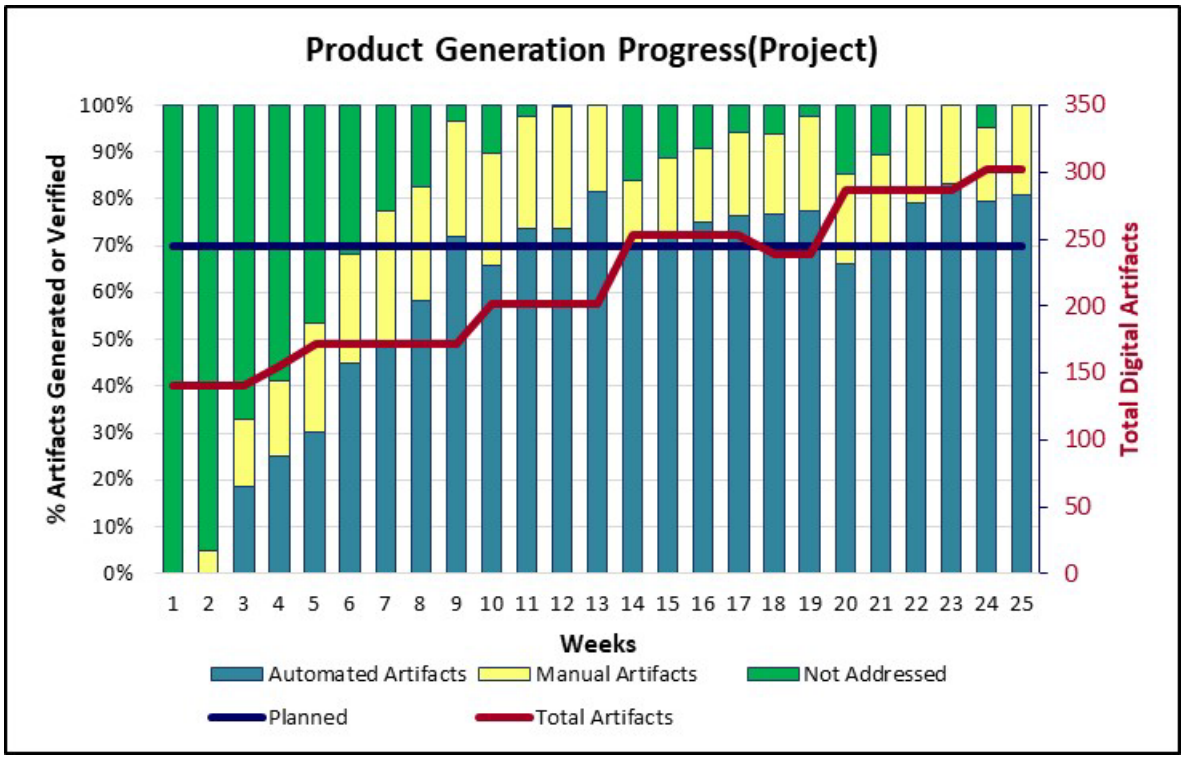
Additional Specification Information	
Information Category	Product Quality
Measurable Concept	Functional Correctness
Relevant Entities	Approved change requests, by change type
Attributes	Rework measures may vary according to the work product being modified and stage in the product life cycle. Example attributes that can be used to guide the rework analysis may include: • Product type (e.g., requirements, design, implemented work product, test) • Change type (corrective, perfective, adaptive) • Program activity or phase • Change reason (e.g., requirements change, defect) • Change request priority or severity See also the defect attributes described in 8.4.
Data Collection Procedure	Collect change requests approved for work from a baseline change management repository or tool. Collect defects and associated attributes from the project configuration management repository or defect management tool. Collect the size of the change effort (count of affected model elements) from the project modeling tool. Collect labor measures from the project time tracking system. Labor should be tagged, categorized, or otherwise retrievable specific to the scope of the change effort. Collect cost measures from the project financial accounting system. Cost should be tagged, categorized, or otherwise retrievable specific to the scope of the change effort.
Data Analysis Procedure	Analyze aggregate rework measures and trends at regular intervals, such as monthly or quarterly, or in response to observed performance anomalies.

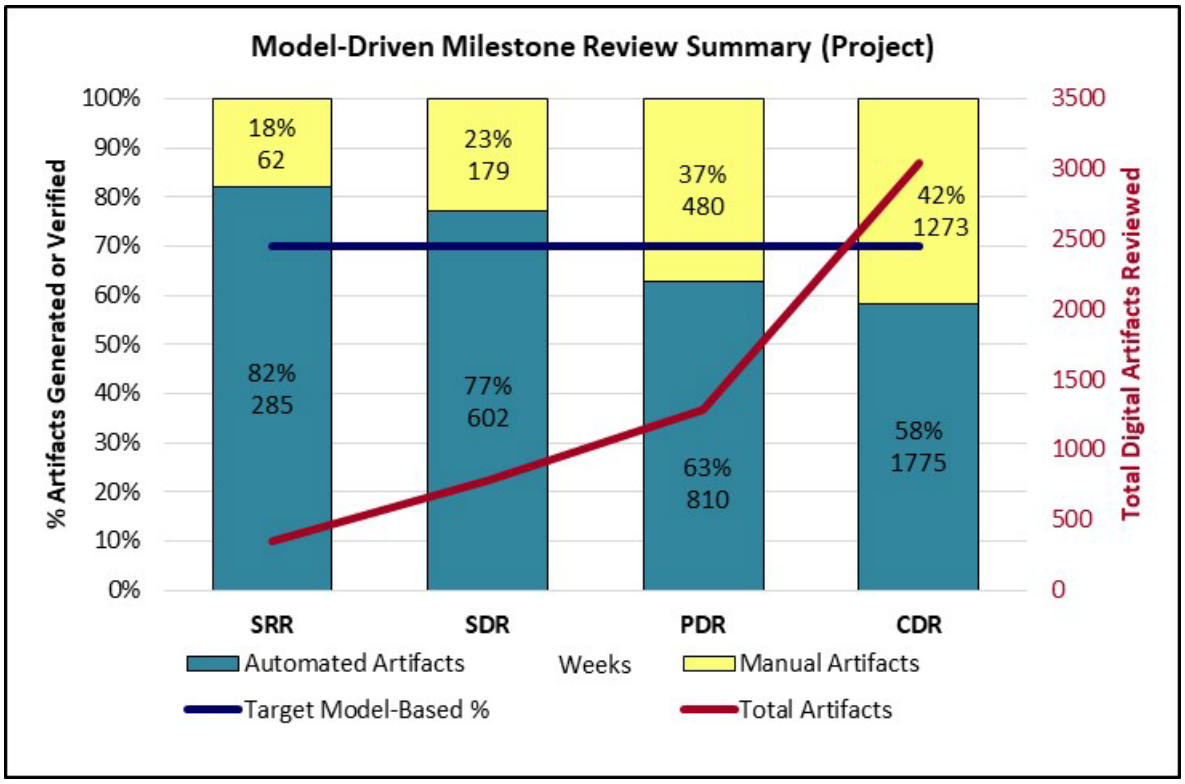
8.6 PRODUCT AUTOMATION

Measure Introduction	
Description	Model-driven development provides opportunities to automate engineering processes and generation of work products that have often been done manually in traditional approaches. Model-based work products such as requirements, architecture, design, use cases and other views or modeling artifacts can be automatically generated and published directly from modeling tools, at significant savings in effort relative to traditional documentation-centric approaches. Model-driven automation based on an Authoritative Source of Truth (ASoT) can lead to process efficiencies, labor reductions, shorter cycle times, less rework, and earlier verification and validation of solutions. Artifacts applicable for automation may vary based on many factors, including product, requirements, domain, availability of reference models, processes, resources, tools, and business constraints. It may not be practical for projects or enterprises to expect that all artifacts are model-generated. Projects or enterprises may set objectives for the quantity or percentage of engineering products that are automatically generated in a model-centric approach. Examples of potential model-driven measures of digital engineering product automation include: • percentage of digital model artifacts produced via automation • percentage of requirements verified through automation of digital model parameters and constraints • percentage of labor hours spent generating digital artifacts through automated vs. manual methods The industry sees automation of digital artifacts as one of the most significant expected benefits from a digital engineering implementation, so this specification is currently focused on measuring the actual artifacts only, with the goal of inspiring progress toward a widespread practice of model-based automated artifacts and reviews. As of this writing, the authors are not familiar with representative studies substantiating consistent savings in labor, cost, rework, or reviews realized through digital model-driven vs. documentation-driven approaches. It can be difficult to perform direct comparisons since systems vary widely, and it is likely some proportion of both approaches will continue be common on projects for some time. As the industry is still generally in the early stages of digital transformation with little historical data, it is also not clear that projects can accurately estimate the quantity of artifacts needed to compare plans vs. actuals in a digital model-driven environment. Other benefits of the modeling process and automation include: • reduce the potential for error by standardizing/reusing well-vetted model elements and software code • generate review artifacts automatically, thus facilitating the completeness, correctness, and consistency of the review process • provide the savings that contribute to a return on investment (ROI)
Relevant Terminology	Automated artifacts — Products or artifacts produced and reviewed directly from digital models without significant manual intervention or generation of separate documents for development and review. Artifacts (data elements or model elements) are defined based on program/enterprise conventions and tools. Examples: model-based views and diagrams for requirements, architecture, design.

Information Need and Measure Description	
Information Need	What percentage of artifacts are automatically generated from digital models? To what extent are artifacts facilitating program reviews? How much is automation contributing to meeting performance and quality objectives?
Base Measure 1	Total Artifacts *[integer > 0]* *Total count of artifacts generated using both automated and manual methods.*
Base Measure 2	Automated Artifacts *[integer ≥ 0]* *Count of artifacts generated from automated model-driven methods.*

Base Measure 3	Manual Artifacts *[integer ≥ 0]* *Count of artifacts generated using manual (non- model driven) methods, e.g., documentation generation.*
Base Measure 4	Known Artifacts Not Yet Addressed *[integer ≥ 0]* *Count of artifacts known to be necessary, but not yet generated using either automated or manual methods.*
Derived Measure 1	Percent of Automated Artifacts = ((Automated Artifacts) / (Total Artifacts)) * 100 *[percentage ≥ 0%]*
Derived Measure 2	Percent of Manual Artifacts = ((Manual Artifacts) / (Total Artifacts)) * 100 *[percentage ≥ 0%]*
Derived Measure 3	Percent of Known Artifacts Not Yet Addressed = ((Known Artifacts Not Yet Addressed) / (Total Artifacts)) * 100 *[percentage ≥ 0]*

Indicator Specification	
Indicator Description and Sample **(Product Model-Driven Artifact Automation)**	Figure 8.6-1 depicts the percentage of project artifacts that are generated or verified by automated vs. manual methods. In this example, the project set a planned objective for 70% automation, and ultimately met and exceeded that objective. Percentages are used rather than absolute values to facilitate comparisons across projects, as the absolute quantities of artifacts generated is likely to vary widely. The total number of artifacts changes over time as digital modeling matures across engineering requirements, design, or verification stages. The term "artifact" is used as a proxy for the quantity or size of work products or model elements plotted on the vertical axes for the system or discipline of interest, e.g., requirements, design, use cases, test cases. The total quantity of artifacts is plotted on the secondary axis for context to enable consideration of the scale and complexity of the development and automation effort. Tradeoff decisions can be made on the benefit of investing further program effort to develop new digital modeling automation tasks to increase coverage. This may include estimating the net impact on program throughput, quality, or cost.  **Figure 8.6-1: Automation Coverage (Project Level)** The project work scope may evolve iteratively (with additions, modifications, deletions) based on collaboration with acquirers and other stakeholders. The scope of the automation effort may also vary accordingly. By week 9, over 70% of the planned modeling artifacts are generated or verified using

	automated methods enabled by digital modeling tools. In week 18, as shown by the blue line, a modeling component was deleted from the work scope and the artifact count was reduced. Over time, additional automated artifacts are integrated into the digital model that reduce the dependence on manual development, documentation, and verification tasks. The project has generally met its objective of 70% of artifacts generated directly from digital models.
Indicator Description and Example **(Model-Driven Milestone Reviews)**	In traditional approaches extensive effort is often spent preparing products for milestone reviews with acquirers, such as exporting or packaging applicable products into presentation slides for review. In a model-based approach, many of the project products can be reviewed directly from modeling tools, saving effort and schedule from the presentation preparation and conduct, and using the Authoritative Source of Truth (ASOT) directly as a basis for reviews instead of using copies separate from the model itself.  **Figure 8.6-2: Model-Driven Design Reviews** Figure 8.6-2 depicts the relative percentage of artifacts reviewed with acquirers in a series of model-based milestone reviews. In this example, the project has established an objective of 70% of the artifacts reviewed in the milestone reviews being published directly from the digital models. This objective was met for systems engineering reviews early in the project lifecycle (System Requirements Review, System Design Review) based primarily on MBSE models. However, it proved difficult to meet this same level of model-based content for the Preliminary Design Review and Critical Design Review as the project engineering disciplines (software, mechanical, electrical) are still working toward their respective digital design transformation and integration of cross-discipline models. Further, both the supplier and acquirer base are still overcoming traditional and cultural obstacles within the acquisition system and workforce to become fully receptive to a model-based design review approach.
Analysis Model	Percentage of Automated Artifacts Generated or Verified: • What percentage of artifacts are automated from digital models? Is each requirement or design element fully covered by the automation, or are some aspects verified manually, or not yet verified? • Decisions must be made on the value obtained from investing in automation. Any artifacts not generated or verified through automation must be done manually, which can impact productivity, schedule, and resources. Apply decision tradeoffs for the cost vs. performance benefit of investing effort to expand the extent of automation.

	Automated model-driven verification is a primary enabler for achieving efficiency, quality, and cost savings at both the project and organizational levels. Organizations should monitor automated verification measures in relation to achievement of their desired performance objectives.
Decision Criteria	The impact of digital modeling automation is judged best not by the quantity of artifacts generated, but by the savings in effort and schedule relative to generating and maintaining similar artifacts using manual generation and documentation-driven methods. Automation alone is not an objective; it is the associated gains in accelerating performance and improving product quality at the project and organizational levels that make investments in automation worthwhile. Automation measures should be evaluated in the context of other performance measures, such as those defined elsewhere in the PSM DE measurement framework. Objectives for the extent of model-driven automated artifact generation may be specific to the product or domain. Automation in the range of 70%-80% is often beneficial in producing improved performance outcomes, but this may vary by domain or application.

Additional Information	
Additional Analysis Guidance	If automation measures are lower than planned, or if there are process effectiveness or product quality issues that are impacting objectives, consider root cause analysis and decision tradeoffs to assess the impact and determine if they can be improved by further investments in automation. Effort and cost measures can be correlated with digital product automation measures in order to determine the business savings and efficiencies gained.
Implementation Considerations	Relying solely on digital model artifact automation may not be wholly sufficient to exercise all functionality needed (e.g., user interfaces, quality attributes). It may be necessary to supplement automated artifact generation and verification with manual effort to adequately cover all required functionality. Some models provide more value than others, e.g. supporting validation of more requirements, and a larger percentage may not always aggregate to a higher value. It may not be cost effective or practical to have models for everything.

Additional Specification Information	
Information Category	Process Performance
Measurable Concept	Process Efficiency - Automation
Relevant Entities	Digital modeling artifacts
Attributes	Quantity of automated artifacts generated and verified (planned and actual)
Data Collection Procedure	Data is typically collected directly from digital engineering modeling tools. Results are recorded in team tracking tools. Summaries of automated artifact generation and verification results can often be collected automatically using scripts or collected on demand.
Data Analysis Procedure	Data is reviewed and analyzed to ensure adequate quality for each candidate product. Discrepancies in process effectiveness, product quality, or coverage not meeting threshold targets may indicate updates to code or test scripts are necessary.

8.7 DEPLOYMENT LEAD TIME

<table>
<tr><th colspan="2">Measure Introduction</th></tr>
<tr><td>Description</td><td>
Deployment Lead Time is a measure of how rapidly authorized requests for system capabilities and work products can be engineered, developed, and delivered for use in their intended operational environment. Deployments may be related to a single product, multiple iterations of that product, or across multiple comparable products or programs. By systematically measuring the duration of processes and workflow steps in product development over time, decision makers are enabled to analyze process performance efficiency and act on bottlenecks to reduce the deployment lead time for new capabilities. DE models are expected to both facilitate this analysis, decision making, and joint efforts. Attributes characterizing the relative work performed (e.g., product requirements, model elements, product size, complexity) can be used to normalize and synthesize comparable work performed under similar defined conditions.

Deployment Lead Time in aggregate generally consists of major workflow stages and milestones as depicted in Figure 8.7-1. Major workflow stages are Queued Time, Cycle Time, and Deploy Time. Major milestones are Work Identified, Work Started, Work Completed, and Work Deployed. Deployment Lead Time and its elements are used to evaluate efficiencies in deploying work products and as a predictor for estimating future product deployment times.

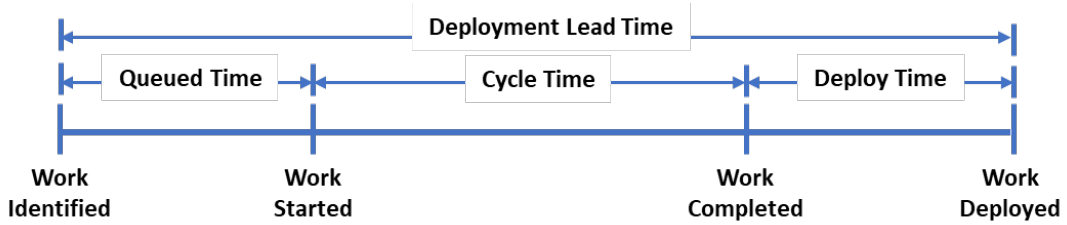


Figure 8.7-1: Stages and Elements of Deployment Lead Time

These general concepts are similar to those common in many manufacturing and development domains (e.g., software agile methods). For digital engineering, the overarching objective of this specification is to characterize the process efficiency for developing and deploying digitally engineered products relative to traditionally engineered products.
</td></tr>
<tr><td>Relevant Terminology</td><td>
<table>
<tr><td>Queued Time</td><td>The time a received and approved work request sits idle. Queued time includes the up-front effort needed to define and prepare the work to be implemented, such as backlog, prioritization, prototyping, planning, precursor or pre-existing DE modeling efforts, and authorization to start work.</td></tr>
<tr><td>Cycle Time</td><td>The elapsed time from when development work is started until the time development work has been completed and is ready for deployment. This time includes activities such as planning, requirements analysis, design, implementation, and testing. Cycle Time is typically targeted at measuring repeatability and predictability of team performance for well-scoped work so that results are comparable across multiple similar efforts.</td></tr>
<tr><td>Deploy Time</td><td>The elapsed time to deploy completed development work for operational use. Deploy Time includes the time needed to schedule and obtain access to the operational environment for deployment to commence. Deployed means available for use as part of mission operations. If the work is deployed to multiple sites, deploy time is the time that the work is deployed at the site(s).</td></tr>
<tr><td>Deployment Lead Time</td><td>The total time from when an approved request for a new capability is received until the capability is completed, deployed, and available for use in the operational environment. Note: The efforts included per components of Deployment Lead Time might differ from enterprise to enterprise.</td></tr>
</table>
</td></tr>
</table>

Information Need and Measure Description	
Information Needs	How long does it take to deploy an identified feature or capability? How long does it take to deploy a viable product for operational use after a request is received? Where is the deployment bottleneck; in planning/backlog, implementation, or deployment of the implemented capability? How long does it take to develop a DE product?
Base Measures 1	Process Timestamps: date - start and end dates bounding the duration of process workflow events • Identified Date: timestamp when a system requirement or capability request is received and validated. The work may be queued until it is prioritized and resources are available. • Started Date: timestamp when the system capability request is prioritized and authorized to begin development. • Completed Date: timestamp when authorized work completes development (design, implementation, integration, testing) and is authorized for deployment. • Deployed Date: timestamp when work is deployed for use in its operational environment. Timestamps and durations of workflow events are typically in days, but projects may use different scales as appropriate. *Note*: Depending on the chosen acquisition pathway, the DE modeling efforts that facilitate decision milestones might start prior to the Started Date. E.g., prior to the Acquisition Decision Memorandum in the Middle Tier of Acquisition pathway or prior to the development milestone in the Urgent Capability Acquisition pathway (See Queued Time).
Base Measure 2	Product Size: units may vary; refer to the Product Size measurement specification *Product Size is used to normalize the process workflow durations for the amount of work performed.*
Derived Measure 1	Queued Time = (Started Date - Identified Date) [integer: days]
Derived Measure 2	Cycle Time = (Completed Date - Started Date) [integer: days]
Derived Measure 3	Deploy Time = (Deployed Date - Completed Date) [integer: days]
Derived Measure 4	Deployment Lead Time = (Deployed Date - Identified Date) = (Queued Time + Cycle Time + Deploy Time) [integer: days]

Indicator Specification	
Indicator Description and Sample (Deployment Lead Time)	In Figure 8.7-2, notional data for Deployment Lead Time of deployed capabilities is depicted as a stacked column with Queued Time, Cycle Time and Deploy Time shown for each capability. The height of the stacked column is Deployment Lead Time.

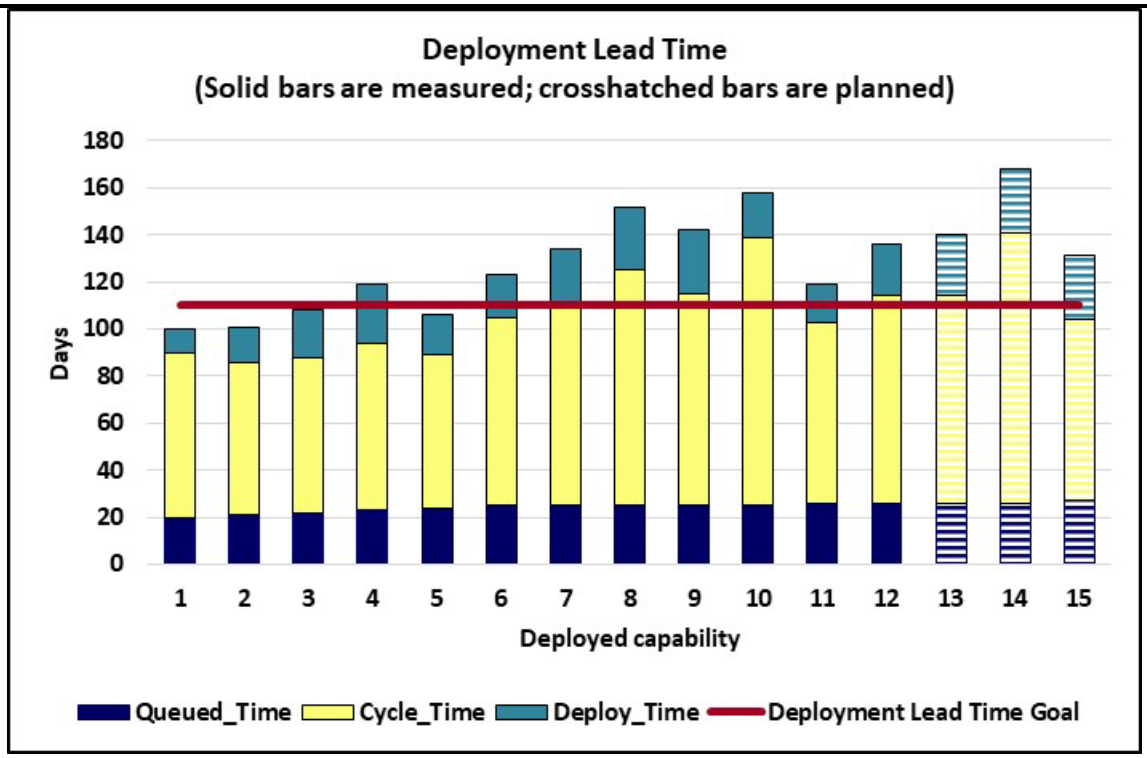


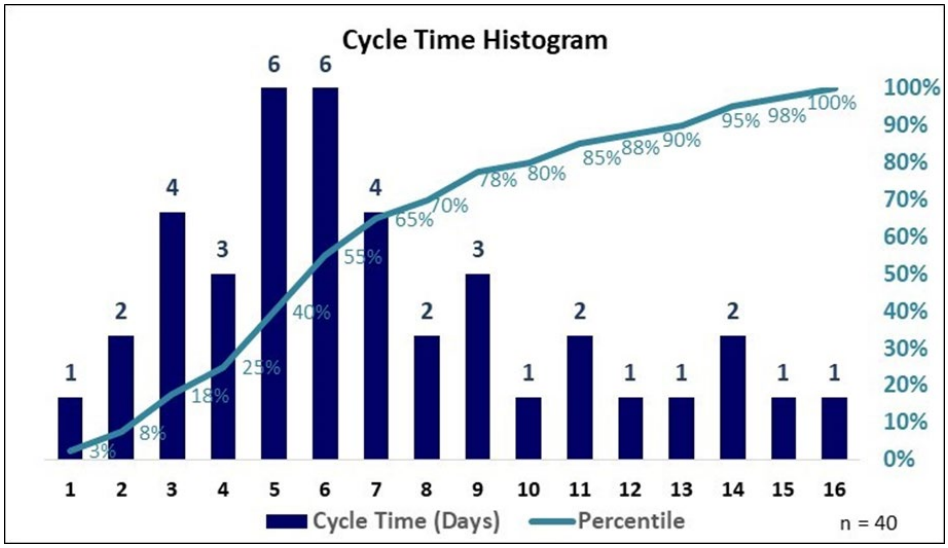
Figure 8.7-2: Deployment Lead Time for Operational Capabilities

This chart allows simple comparison of deployments of multiple work products and planned deployments. Table 8.7-1 below provides a sample of observations that may be drawn from this chart and potential actions associated with the observations.

Table 8.7-1: Sample Observations from the Deployment Lead Time Chart

Observation	Analysis and Actions
The Deployment Lead Time goal was not met for the last 7 completed deployments.	Note that early deployment met goals. Note that large variances in Deploy Time are mostly due to variances in cycle time. Analyze root causes of Cycle Time variance.
Cycle Time variance is the largest contributor to Deploy Time variance.	Analyze root causes of Cycle Time variance. Is it due to lower productivity, increased product size, or inaccurate estimates?
The planned deployments (13, 14 and 15) exceed the Deploy Lead Time goal.	Note that large Cycle Times are the main contributor. Consider ways to reduce Cycle Time, such as adding resources or deferring functionality.
Queued Time is slowly trending upward.	Analyze the root cause of increasing Queued Time. Is the increasing Queued Time indicative of an increasing backlog of approved requests?
Deploy Time has significant variation.	Determine the root cause of Deploy Time variations.*

*Oftentimes, deployment requires coordination with the acquirer or operational environment outside the supplier's control. From the supplier's perspective, potential delays in scheduling access to the operational environment can greatly affect overall Deployment Lead Time. For these reasons, measures based on Deploy Time can be interesting and useful to some extent but may be not as repeatable or actionable as Cycle Time which is more under direct project control.

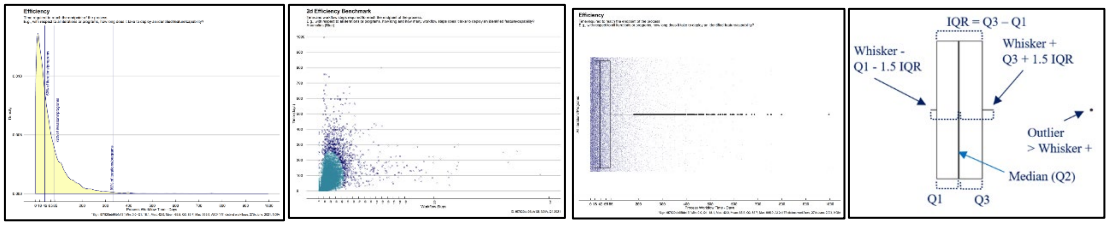
Indicator Description and Example (Cycle Time)	 **Figure 8.7-3: Cycle Time Analysis** Cycle time performance is frequently analyzed in histograms for a set of related products, as depicted in Figure 8.7-3. In this example, 90% of the project's digital engineering product releases are completed in 13 days or less, 65% within 7 days. Cycle time reflects the 'Voice of the Process' from actual results. When conducted for a set of products with similar attributes (domain, scope, product size, complexity, etc.), analyses such as this can be used to characterize process capability (what is achievable?) and the likelihood of meeting project objectives or acquirer expectations ('Voice of the Customer'). Note: In this indicator, the product is comprised of a set of capabilities. DevSecOps programs may track capabilities individually.
Analysis Model	Shorter deployment lead times and cycle times can indicate more efficient delivery/deployment flow and quicker response to business objectives or mission needs. Longer deployment lead times and cycle times are often correlated to the scope, product size, and complexity of work products. Product Size, as an example, may be used to filter or scale the source data set for analysis of root causes of process anomalies, or to obtain greater confidence in estimates or predictions for future work. Teams should implement improvements to bring capability and performance in alignment with the mission need. Deployment lead times can often be optimized by managing depth of the work backlog, improving timely access to the operational environment for deployment, or applying additional resources to perform more work concurrently. Cycle times can also be improved by adding resources or through other approaches, such as improvements to processes, automation, or tooling. Analysis of deployment lead time and cycle time can indicate process performance trends or potential indicators of issues for root cause analysis and performance improvement. Example analyses may include: • Process efficiency and stability (increase/decreasing deployment lead times or throughput) • Predictability for future performance (narrowing or widening standard deviation in deployment outcomes) The analyst may consider questions such as: • Are the deployment lead time and cycle time consistent across iterations? • Are durations increasing or decreasing? Why? • Does the process performance meet business objectives or the mission need? • How predictable are deployment lead time and cycle time? Can we reliably estimate future performance? • What are the process outliers? • What are the root causes for process variance? • What actions should be taken to bring performance in line with expectations?

Decision Criteria	Investigate root causes for variations. For example, review samples that are more than 10% from the average deployment time or work time. When Deployment Lead Time surpasses established objectives, then the delays affect the operational environment. This could lead to not fielding capabilities in the operational environment within schedule.

Additional Information	
Additional Analysis Guidance	As Deployment Lead Time is analyzed, it is important to analyze its components (Queued Time, Cycle Time and Deploy Time). Each of these times will likely have different drivers as described below: • Queued Time - may be driven by backlog, release cycle and priority • Cycle Time - may be driven by work complexity and product size • Deploy Time - may be driven by operational system constraints and procedures "Completed" is expected to be defined within the project's context and criteria, e.g., definition of done. Under consistent conditions, deployment lead time can be used as a measure of team capability and throughput that can be used in lieu of traditional size-based productivity measures. Reductions in deployment lead time measures indicate faster delivery to the acquirer, which yields additional potential business benefits such as: • Identification of innovation opportunities • Higher user satisfaction and employee satisfaction • Increased productivity
Implementation Considerations	The components of Deployment Lead Time can be automatically collected and analyzed by many common tool suites, or by other implementations. Data may reside in different repositories and may need to be combined for analysis.

Additional Specification Information	
Information Category	Process Performance - Process Effectiveness
Measurable Concept	Deployment Lead Time
Relevant Entities	Elapsed time duration
Attributes	Time stamps Unique Identifier for each deployed capability Identification of team and project for each work product
Data Collection Procedure	Measurement of milestone timestamps should be collected from project management and workflow tools, or from other implementations. Operational deployment milestones may be collected from the acquirer's business systems. Product size, if used to normalize the amount of work is collected as described in the Product Size measurement specification.
Data Analysis Procedure	Data is analyzed at the end of each deployment by the team and considered during planning sessions for the follow-on deployments. Performance trends may be analyzed at periodic intervals (e.g., quarterly) by the program to assess systemic issues and identify improvement actions to align performance with business and mission objectives.

Advanced Topic - Statistical Measures for Digital Engineering	Digital engineering process efficiency, effort, or time-based measures, such as deployment lead time or cycle time, are enablers to characterize and act upon current performance, predict future performance, or commit to schedules for estimating future work. Hence, projects and organizations will want a higher level of confidence in the integrity and representativeness of their data sets for decision making. That likely involves more detailed analysis such as:

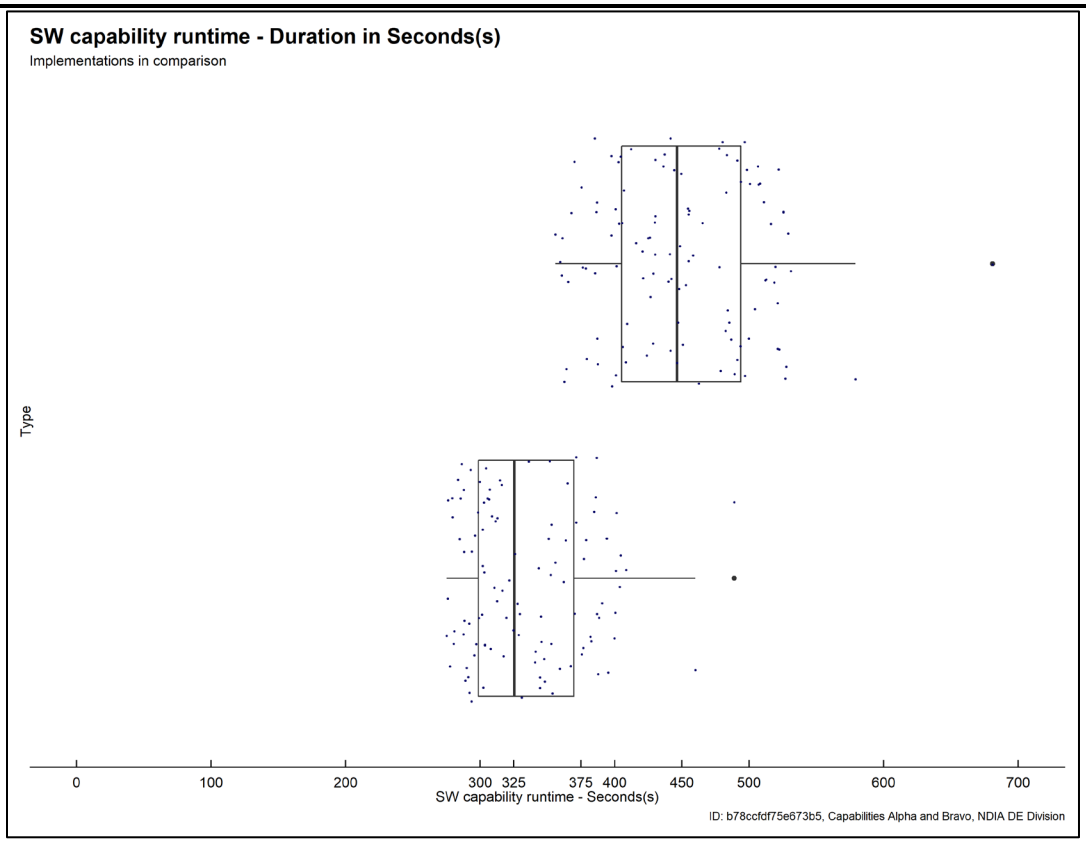
Efficiency and Prediction	• Applying statistical methods to gain a deeper understanding of process performance, capability, and predictability • Analyzing sources of variation (common causes, special causes), especially anomalies or outliers outside typical ranges of process performance • Decomposing process elements and dependencies to deep dive into key issues or bottlenecks Example methods might include: • Statistical measures: mean, median, standard deviation, inter-quartile ranges, outliers, etc. • Analyses: frequency distribution curves, scatter plots, box plots, etc.  **Figure 8.7-4: Plots and Advanced Analyses** In the context of this measurement specification, this could support more advanced analyses with greater insight, such as: • Plotting cycle time or deployment time vs. a separate measure of interest (e.g., workflow steps, defects) • Plotting absolute frequency of workflow steps vs. start/end boundaries, or durations • Identifying distinct workflow patterns • Monitoring trends in the cumulative flow of work across process states, stages, or milestones Armed with this insight, stakeholders and projects can make better informed decisions based on a detailed understanding of their process capability. This discussion introduces high level concepts, only, with some possibly plots and advanced analyses depicture in Figure 8.7-4. A more detailed description with measures and mathematical analyses is beyond the scope of this digital engineering measurement framework document. Further description and details are published in a white paper on the PSM website, https://www.psmsc.com/Prod_TechPapers.asp

8.8 RUNTIME PERFORMANCE

Measure Introduction	
Description	Ensuring the efficient performance of deployed operational systems is fundamental to meeting business requirements or satisfying a mission need. Performance analysis is critical to early requirements development, architecture, and design processes to ensure the ultimate target solution is feasible. With respect to agile pathways, incremental assessment of the capabilities' performance might be incorporated into the incremental software development process, for example, per sprint. This is generally done through sophisticated models, simulations, and prototypes to validate applicable algorithms or ranges of performance prior to final implementation and deployment in the operational environment. In a digital engineering environment nearly all artifacts are digital, and integration of the tech stack enables stakeholders to maintain and collaborate around an Authoritative Source of Truth (ASoT) for engineering design, review, and validation. The tech stack hosts models that form a digital twin. The runtime infrastructure and performance become crucial concerns in this environment, enabling applicable cross-functional elements to converge on trade-off analyses toward a feasible optimized solution. Runtime performance is a particular concern for models that tax the computing infrastructure, where data latency or sluggish infrastructure performance can have significant adverse effects on the digital design effort. Runtime performance is the amount of time, or duration, that it takes a software system to perform or execute one of its capabilities. By systematically measuring the modeled and implemented runtime performance of alternative solutions, suppliers are able to analyze the likelihood of best meeting operational performance requirements and respond early, as required. Additionally, during the design phase, analysts can plot performance analyses based on historical data to tailor future capabilities to their expected environments and workloads. This iterative data collection, analysis, insight generation, and capability optimization differentiate DE efforts from traditional engineering efforts. This specification introduces summary concepts for measuring runtime performance in a digital engineering environment. Details are beyond the scope of this specification but are described in a separate Digital Engineering Addendum white paper on the PSM website.
Relevant Terminology	Section 3.

Information Need and Measure Description	
Information Need	What is the runtime performance of the capability or system? What is the likelihood that runtime performance will meet operational requirements (for each alternative solution)? Where are the runtime performance bottlenecks, and how can operational performance be optimized?
Base Measure(s) 1	Runtime performance timestamps: date and time • Runtime Performance Start - start timestamp for a runtime performance [timestamp] • Runtime Performance End - end timestamp for a runtime performance [timestamp]
Base Measure(s) 2	Additional Technical Measures within the Runtime Ecosystem: definition and units vary. Often design decisions depend not only on the measured runtime performance but also on a relationship with one or more dependent measures, such as consumption of other computing resources (e.g., memory utilization and bandwidth). The combination of measures can be analyzed in trade off analyses to determine an optimized solution. *Note:* From a Decision Authority point of view, additional non-technical measures, e.g., achieved availability might be of interest.

Derived Measure 1	Elapsed Time = (Runtime Performance End) - (Runtime Performance Start) [duration] *Duration between start and end of a performance interval*
Derived Measures 2	Statistical analysis: measures of runtime performance across a set of measured time intervals. e.g., min; max; mean; median; standard deviation; percentiles; and outliers These common derived statistical measures and analyses are well defined in practice and are not detailed in this measurement specification. Example statistical graphs and indicators include box plots, scatter graphs, distribution profiles, histograms, etc. Refer to the separate Digital Engineering Efficiency white paper on the PSM website for further description and examples: https://www.psmsc.com/Prod_TechPapers.asp
Derived Measures 3	Runtime performance benchmarks: [duration] Time required to compute or perform a capability, process, subprocess, or activity. May include a set of iterations (1.. n) or weightings to create a linear combination.
Derived Measures 4	Multivariate analyses: varies by selected parameters in relationships with runtime performance.

Indicator Specification	
Indicator Description and Sample	 **Figure 8.8-1: Runtime Performance Plot** Figure 8.8-1 uses box plots to contrast the runtime performance results for two alternative implementations. Multiple runtime performance samples for each alternative are summarized in box plots, which include statistical depictions of the sample median, Interquartile Ranges (IQRs), dispersion of measured data points, and outliers. In this example, module Alpha (bottom) features faster median runtime performance, 325.4 seconds, than module Bravo (top), 446.4 seconds, due to the utilization of refactored code.

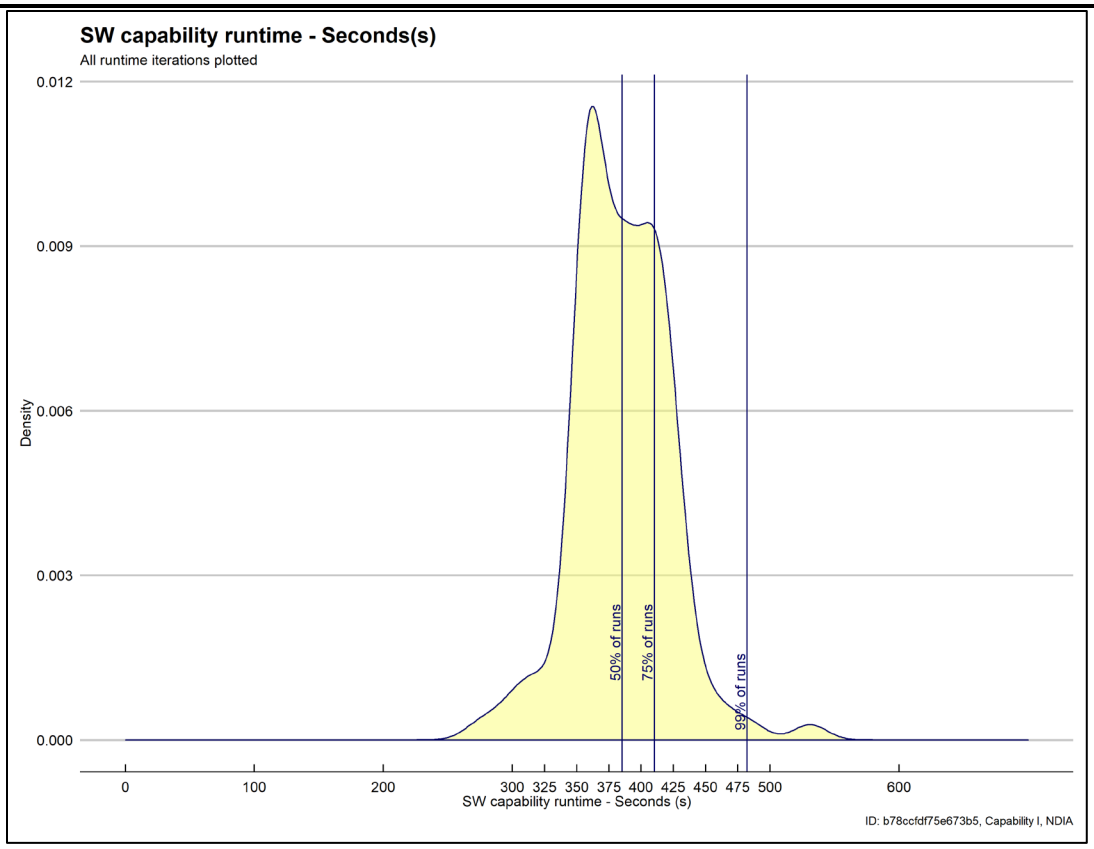


Figure 8.8-2: Runtime Performance Density Distribution

Figure 8.8-2 plots the density distribution of runtime performance samples, with specific percentiles, for the probabilities of performance within performance ranges. In this example, 50% of runtime performance samples are measured at ≤ 385.3 seconds, 75% of samples are measured at ≤ 410.3 seconds, and 99% of samples are measured at ≤ 482.1 seconds. The program team can use analyses such as these to consider the likelihood that alternative solutions meet operational performance objectives.

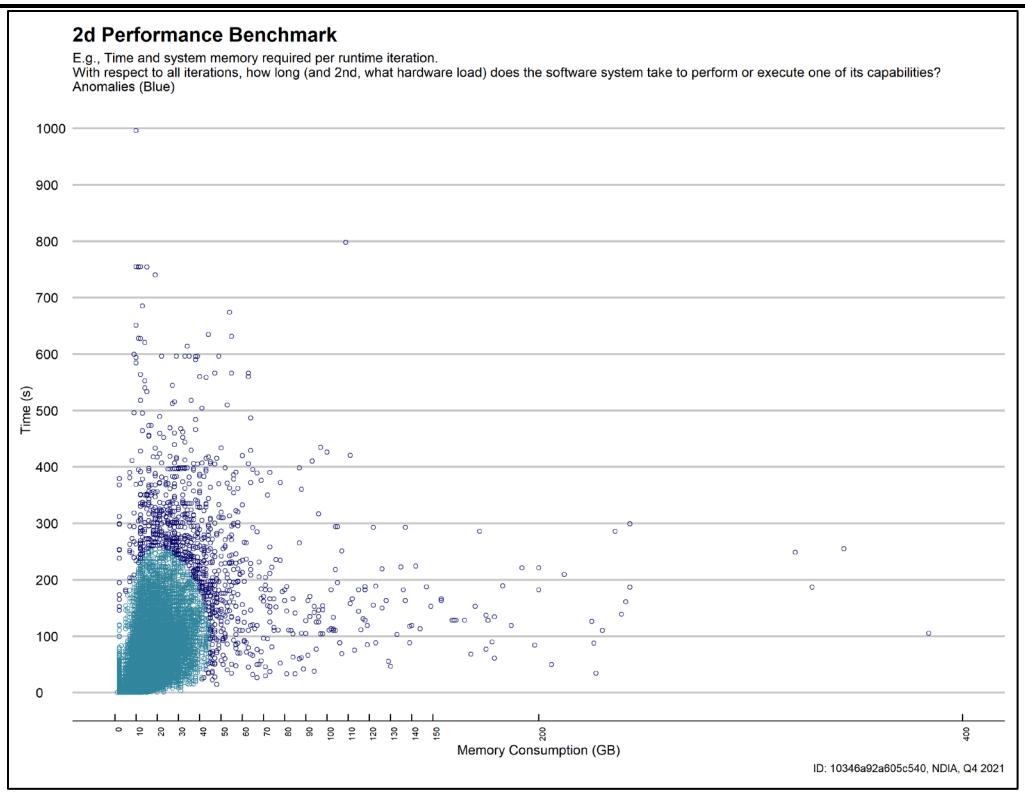


Figure 8.8-3: Anomaly Analysis

Figure 8.8-3 uses a two-dimensional scatter graph to depict the measured runtime performance against a second technical measure of interest within the runtime ecosystem. The analysis of both measures in combination can be used to determine an optimized solution. In this DE environment example, there seems to be a relationship between runtime performance and the memory consumption. Distinct runtime iteration anomalies are depicted via purple points. All other points seem to be clustered in the 0 - 200 seconds and the 1 - 50 GB memory consumption range.

Analysis Model

Figure 8.8-1 depicts the utilization of common, derived statistical measures that form runtime performance box plots. These allow the analyst to conduct a comprehensive, first-glance runtime assessment of the

- "best case" runtime,
- "worst" runtime,
- "center of gravity" of the runtimes,
- "spread", i.e., degree of dependability for the end user,
- "best" and "worst" quarters of runtimes, and
- runtimes the end user might expect over multiple iterations.

With respect to Figure 8.8-2, analysts might utilize customizable percentiles, e.g., 75%, to assess whether the DE capability meets the specific performance requirements of the acquirer. Additionally, extreme runtimes might be identified.

While specific tests for outlier detection (See Derived Measure 2) might be utilized at the beginning of an analysis, advanced algorithms enable the analysis of >10k runtimes in a reasonable amount of time (e.g., 5 to 60+ seconds). Figure 8.8-3 visualizes the results of an anomaly detection run. Analysts might assess the range and typical clusters of the two measures of interests at first glance. Additionally, this graph supports

	continuous monitoring, e.g., one might visually assess the results of interventions, e.g., hardware changes, by comparing a pre-change and post-change anomaly run visualization.
Decision Criteria	*Note*: This sections refers to scenarios that indicate a need to take action. **Figure 8.8-1:** Outliers, i.e., points outside the whiskers of the box graphs, or extensive whiskers exist: The runtime of the DE capability might not be reliable or specific cases lead to extensive resource and time consumptions. When one of the compared box graphs features one or more of the following possible observations, the decision maker shall favor this DE capability and the associated supplier, with: • a lower median score, • smaller IQRs, • less outliers, or • shorter whiskers. **Figure 8.8-2:** DE capability does not meet the specific performance requirements, for example 99% <450s: Each quarter or milestone, the acquirer might brief the DE capability supplier about the (objective) status quo of the capability and request more contextual data of these extreme runtime cases for further analysis. Moreover, the supplier might elaborate on these extreme cases. **Figure 8.8-3:** Anomalous runtime and the Additional Technical Measure (See Base Measure 2) combinations that feature over 100% higher runtime or Additional Technical Measure readings have been identified: The decision maker shall order the replacement of software modules and shall order in-depth analyses of the specific DE capability runs that feature these extensive resource and time consumptions.

Additional Information	
Additional Analysis Guidance	Customized measures, e.g., statistical tests for outlier detection might enable more sophisticated analysis. On the capability level, i.e., complex system, one might account for module interaction effects. These apply to complex systems that feature constrained hardware or network resources, e.g., due to undersized microelectronics or design decisions. Please refer to: https://www.psmsc.com/Prod_TechPapers.asp
Implementation Considerations	Suppliers/analysts might integrate specific modules or lines of code that benchmark specific parts of the call stack. For instance, one might calculate the time taken of a method that loads structured data into memory. Via monitoring efforts, e.g., via logging, suppliers gain an understanding of the runtime of their code.

Additional Specification Information	
Information Category	Technology Effectiveness
Measurable Concept	Technology Performance
Relevant Entities	Runtimes of the DE capability Runtime measurements and associated information (See Attributes)
Attributes	Time stamps (See Base Measures) *[mandatory]* ID *[optional]* Additional Technical Measures *[optional]* Operational environment or contextual information *[optional]* Additional information of analytical interest *[optional]*

Data Collection Procedure	Common elements of the collection process: Existing log files might serve as a starting point. However, to enable runtime monitoring and analysis, the data collection needs shall be defined by stakeholders and analysts. Collect duration timestamps using performance monitoring implementations or tools. A DE model likely facilitates iterative data collection.
Data Analysis Procedure	The analyst might assess specific capabilities, modules, or submodules by filtering. Aggregations enable further computations. The actual implementation of the analyses varies, e.g., some might utilize built-in capabilities of performance monitoring tools, additional source code, or statistical packages.

ANNEX A: PSM MEASUREMENT PROCESS AND TERMINOLOGY

This annex includes additional details on the measurement process, and definitions for the terminology utilized. Section A.1 includes details on the measurement process. Section A.2 includes definitions for each of the fields in the measurement specification. The annex also includes a blank measurement specification template for project or enterprise use (section A.3). Section A.4 includes definitions for the PSM information categories.

A.1 PSM MEASUREMENT PROCESS

The PSM Measurement Process in Figure 4-1 is built around a typical "Plan-Do-Check-Act" management sequence. It encompasses the activities of Plan Measurement, Perform Measurement, Evaluate Measurement, and Establish and Sustain Commitment.

The Plan Measurement activity encompasses the identification of management and technical information needs and the selection of appropriate measures to address these needs using the Measurement Information Model. Plan Measurement also includes tasks related to the definition of data collection, analysis, and reporting procedures; tasks related to planning for evaluating the measurement results in the form of various information products; and tasks for assessing the measurement process itself. Most significantly, the Plan Measurement activity provides for the integration of the measures into existing technical and management processes. Rather than force a project or enterprise to implement predefined measures, Practical Software and Systems Measurement, ensures that the selected measures will be effective within the context of the project or enterprise. The Plan Measurement activity also addresses the resources and technologies required to implement the measurement program. The output of the Plan Measurement activity is a well-defined measurement approach that directly supports project and enterprise information needs.

The Perform Measurement activity, along with Plan Measurement, is one of the core activities that directly address the requirements of the measurement user. Perform Measurement encompasses the collecting and processing of measurement data; the use of the data to analyze both individual information needs and how the information needs and associated issues inter-relate; and the generation of information products to present the analysis results, alternative courses of action, and recommendations to the project and enterprise decision makers. Perform Measurement implements the measurement plan and produces the information products necessary for effective measurement-based decision making.

The Evaluate Measurement activity applies measurement and analysis techniques to the measurement process itself. It assesses both the applied measures and the capability of the measurement process, and it helps identify associated improvement actions. The Evaluate Measurement activity ensures that the project or enterprise measurement approach is continually updated to address current information needs and promotes an increasing maturity of the measurement process.

The Establish and Sustain Commitment activity ensures that measurement is supported both at the project and enterprise levels. It provides the resources and enterprise infrastructure required to implement a viable measurement program.

A fifth activity, Technical and Management Processes, is also depicted in the Measurement Process Model. Although technically not a measurement- specific activity the technical and management processes interface directly with the measurement process. The decision makers operate within these processes, defining information needs and using the measurement information products to make decisions.

The Measurement Process Model is iterative by design. It is defined to be tailored to the characteristics and context of the project or enterprise and to be adaptable to changing information and decision requirements. Both the Measurement Information Model and the Measurement Process Model establish a measurement approach that captures the experience of and principles learned from previous measurement applications. Together they provide the basis for an effective measurement program.

A.2 MEASUREMENT SPECIFICATION DEFINITIONS

PSM measurement specifications in this framework are defined and documented using a common template, described in Table A.2-1.

Table A.2-1: Measurement Specification Definitions

Measure Introduction	
Description	An overview of the measure and the information it provides and how it is used.
Relevant Terminology	Definitions of terms that are specific to this measurement specification. Section 3 of this framework defines terms that apply to all measures.

Information Need and Measure Description	
Information Need	What the measurement user and stakeholder need to know in order to make informed decisions. These are generally written in the form of questions that are addressed by the measure, related to the goals and objectives of the project or enterprise.
Base Measure	Measure of a single ***attribute*** defined by a specified measurement method. Executing the method produces a value for the base measure. A base measure is functionally independent of all other measures and captures information about a single attribute. Example base measures include number of requirements traced to architecture elements, cost, and number of anomalies. Base measures are defined with the following characteristics: • **Measurement Method**: The logical sequence of operations used to quantify an attribute with respect to a specified scale. The operations may involve activities such as counting occurrences or measuring the passage of time. • **Type of Method**: Either (1) subjective, involving human judgement, or (2) objective, using only established rules to determine numerical values. • **Scale**: An ordered set of values or categories to which an attribute is mapped. It defines the range of possible values that can be produced, and often includes a unit of measurement. • **Type of Scale**: The type of the relationship between values on the scale, either: - *Nominal* - categorical data, as in defects by their type. - *Ordinal* - discrete rankings, as in assignment of defects to a severity level. - *Interval* - numeric data for which equal distances correspond to equal quantities of the attribute without the use of 0 values, such as the cyclomatic complexity value for each logic path in a software unit. - *Ratio* - numeric data for which equal increments correspond to equal quantities of the attribute, beginning at zero, such as size measurement in terms of number of requirements. • **Unit of Measurement**: A particular quantity, defined and adopted by convention, with which other quantities of the same kinds are compared in order to express their magnitude relative to that quantity. Only quantities expressed in the same unit of measure are directly comparable.

Derived Measure	A quantity defined as a function from two or more base and/or derived measures. A derived measure captures information about more than one attribute. An example of a derived measure is a calculated value of productivity derived by dividing the base measure of hours of effort by the base measure of product size (e.g., model elements). Derived measures are defined with the following characteristics: • **Measurement Function**: The formula or algorithm that is used to combine two or more values of base and/or derived measures.

Indicator Specification	
Indicator Description and Sample	**Indicator**: A measure that provides an estimate or evaluation of specified attributes derived from an analysis model with respect to defined information needs. Indicators are the basis for measurement analysis and decision making, and are generally what should be presented to measurement users. Measurement is often based on imperfect information, so quantifying the uncertainty, accuracy, or importance of indicators is an essential component of presenting the actual indicator. Indicators are often displayed as graphs or charts. Indicators are defined with the characteristics of an ***analysis model*** and ***decision criteria***. In the sample measurement specifications provided in this paper, the indicator section includes a description of the indicator, along with one or more sample graphs or charts. In addition, a description of how the sample indicator was interpreted and used is included.
Analysis Model	An algorithm or process that applies supporting information and ***decision criteria*** to a combination of base and/or derived measures to inform decisions. Analysis models produce estimates or evaluations relevant to defined information needs. An analysis model may consider or include the following: • other contextual information to put it into proper perspective, • thresholds of acceptable/not acceptable/borderline performance, • defined decision criteria and the need to take corrective action, and • associated issues, risks, and problems noted.
Decision Criteria	Numerical thresholds, targets, and limits used to determine the need for action or further investigation, or to help describe the level of confidence in a given result. Decision criteria help to interpret the measurement results. They may be based on a conceptual understanding of expected behavior, or calculated from data. They should be quantitative, versus qualitative, whenever possible. Decision criteria may be derived from historical data, plans, heuristics, or computed as statistical control limits or statistical confidence limits. Decision criteria should be established before collecting data to strengthen resolve to take action once a threshold or limit is breached, or a target is achieved.

Additional Information	
Additional Analysis Guidance	Any additional guidance on variations of this measure, or advanced analysis that may be performed, including comparisons to other measures.
Implementation Considerations	Any process or implementation requirements that are necessary for successful implementation of this measure on a project or within an enterprise. Lessons learned for implementation are provided.

Additional Specification Information	
Information Category	A logical grouping of information needs that are defined in PSM to provide structure for the Information Model. PSM information categories include Schedule and Progress, Resources and Cost, Product Size and Stability, Product Quality, Process Performance, Technology Effectiveness, and Customer Satisfaction. Information categories are defined in section A.3 and in Chapter 2 of the PSM book.[1]

Measurable Concept	An idea for satisfying the information need by defining the entities and their attributes to be measured. Measurable concepts are a subset of measures within an information category that utilize the same entities or attributes.
Relevant Entities	The object that is to be measured. Entities include processes, products, projects, and resources. An entity may have many attributes, only some of which may be suitable to be measured.
Attributes	Distinguishable property or characteristic of an entity.
Data Collection Procedure	Information on how data is collected and validated. In the measurement specifications in this document, general guidance is provided. When tailoring for a specific project, the information should include: • The frequency or time period when data is collected (e.g. daily, monthly, per release, etc.) • The person or role who is assigned to collect the data (e.g. measurement analyst, project manager) • The phase, iteration or activity when the data is collected. • Tools or methods used to collect the data. • V&V tests that will be run to ensure the data is complete and accurate. • Tools or repositories where data is stored after it is collected (e.g., database).
Data Analysis Procedure	Information on how data is analyzed, disseminated, and used for decision making. In the measurement specifications in this document, general guidance is provided. When tailoring for a specific project, the information should include: • How often data is reported (this may be less frequently than it is collected). • The person/role who is assigned to analyze data and report the results. • The phase or activity when the data is analyzed. • Sources of data for this analysis. • Tools used for analysis (e.g., statistical tools). • When results are reviewed and reported, along with the intended user of the results, and the expected decisions that are supported by this measure.

A.3 MEASUREMENT SPECIFICATION TEMPLATE

Table A.3-1 is a blank template that can be used to develop specific measures for a project or enterprise.

Table A.3-1: Blank Measurement Specification Template

Measure Introduction	
Description	
Relevant Terminology	

Information Need and Measure Description	
Information Need	
Base Measure 1	
Derived Measure 1	

Indicator Specification	
Indicator Description and Sample	
Analysis Model	
Decision Criteria	

Additional Information	
Additional Analysis Guidance	
Implementation Considerations	

Additional Specification Information	
Information Category	
Measurable Concept	
Relevant Entities	
Attributes	
Data Collection Procedure	
Data Analysis Procedure	

A.4 MEASUREMENT SPECIFICATION DEFINITIONS

Practical Software and Systems Measurement is an information-driven approach to measurement for project and enterprise management. PSM defines seven common information categories:

Table A.4-1: PSM Common Information Categories

Information Category	Project Level	Enterprise Level
Schedule and Progress	At the project level this information category addresses the achievement of project milestones and the completion of individual work units or tasks. A project that falls behind schedule can usually meet its delivery objectives only by eliminating functionality or sacrificing product quality.	At the enterprise level, this information category focuses on the completion of the set of projects, those needing management attention, and enterprise technical debt.
Resources and Cost	At the project level this information category relates to the balance between the work to be performed and resources assigned to the project. A project that exceeds the budgeted effort usually can recover only by reducing functionality or by sacrificing product quality or quantity.	At the enterprise level, this information category focuses on managing qualified personnel and other resources across the portfolio of projects.
Size and Stability	At the project level, this information category addresses the stability of the functionality or capability required of the software or system. It also relates to the volume of product delivered to provide the required capability. Stability includes changes in functional scope or quantity. An increase in quantity or scope usually requires increasing the applied resources or extending the project schedule.	At the enterprise level, this information category addresses the size of the portfolio.
Product Quality	At the project and enterprise levels, this information category addresses the ability of the delivered products to support the user's needs without failure. At the project level, correcting poor product quality will impact cost and schedule, and may affect technical performance. If the delivered product quality is poor, the burden of making it work usually falls on the assigned maintenance organization.	At the enterprise level, this information category addresses the amount of enterprise rework required, and the (expected) quality improvements from a DE environment.
Process Performance	At the project level, this information category relates to the capability of the project to meet its contractual needs. A poor development process, or low productivity, may have difficulty meeting aggressive project schedule and cost objectives.	At the enterprise level, this information category relates to the capability of the enterprise, increased efficiency due to digital engineering automation, and to systemic performance issues that affect multiple projects.
Technology Effectiveness	At the project level this information category addresses the viability of the proposed technical approach. It addresses engineering approaches such as software reuse, use of commercial software or system components, reliance on advanced processes, and implementation of common architectures. Cost increases and schedule delays may result if key elements of the proposed technical approach are not achieved.	At the enterprise level, this information category addresses technology maturity, improved runtime performance and forecasting, and tradeoffs.

BIBLIOGRAPHY

[1] John McGarry (Author), et al (2001). *Practical Software Measurement: Objective Information for Decision Makers.* Addison-Wesley Professional

[2] Office of the Deputy Assistant Secretary of Defense (Systems Engineering) [ODASD (SE), "DAU Glossary: Digital Engineering," Defense Acquisition University (DAU), 2017. [Online]. Available: https://www.dau.mil/glossary/pages/3626.aspx.

[3] ISO/IEC/IEEE DIS 24641:2021(E) standard: Systems and software engineering – Methods and tools for Model-based systems and software engineering (note this document version was released 9/2021 for comment to the working group and is not yet released for use).

[4] J. Lubell, K. Chen, S. Frechette, J. Horst and P. Huang, "NIST Technical Note 1753: Model Based Enterprise / Technical Data Package Summit Report," August 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.1753.pdf.

[5] INCOSE Systems Engineering Vision 2035, https://www.incose.org/about-systems-engineering/se-vision-2035

[6] ISO/IEC/IEEE 15939:2017 Systems and software engineering — Measurement process; https://www.iso.org/standard/71197.html

[7] Practical Software and Systems Measurement web site, https://www.psmsc.com/

[8] Department of Defense Digital Engineering Strategy June 2018; https://ac.cto.mil/wp-content/uploads/2019/06/2018-Digital-Engineering-Strategy_Approved_PrintVersion.pdf.

[9] OMG Standards Development Organization, MBSE Wiki, Digital Engineering Information Exchange Working Group (DEIX WG), https://www.omgwiki.org/MBSE/doku.php?id=mbse:deix

[10] INCOSE Model-Based Enterprise Capability Matrix and User's Guide, Version 1.0, January 2020; https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=MBCM.

[11] PSM Workshop, Adapting Systems Engineering Leading Indicators for Digital Engineering, 2019, http://www.psmsc.com/UG2019/Workshops/w02.zip

[12] Systems Engineering Leading Indicators Guide, https://connect.incose.org/Pages/Product-Details.aspx?ProductCode=TechGuideLeadInSoft, OR https://www.psmsc.com/Prod_TechPapers.asp

[13] PSM Continuous Iterative Development Measurement Framework, v2.1, Parts 1 through 3, April 15, 2021, http://www.psmsc.com/CIDMeasurement.asp

[14] INCOSE Model-Based Enterprise Capability Matrix and User's Guide, Version 1.0, January 2020

[15] SERC, Benchmarking the Benefits and Current Maturity of Model-Based Systems Engineering across the Enterprise, March 2020, https://sercuarc.org/wp-content/uploads/2020/03/SERC-SR-2020-001-Benchmarking-the-Benefits-and-Current-Maturity-of-MBSE-3-2020.pdf

[16] Systems Engineering Research Center. Summary Report, Task Order WRT-1001: Digital Engineering Metrics. SERC-2020-SR-003, June 2020, https://sercuarc.org/wp-content/uploads/2020/06/SERC-SR-2020-003-DE-Metrics-Summary-Report-6-2020.pdf

[17] Henderson, K., Salado, A., McDermott, T., and Van Aken, E. "Measurement Framework for MBSE," American Society for Engineering Management 2021 International Annual Conference and 42nd Annual Meeting, 27 - 30 October 2021.

[18] L. Delligati, SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley 2013.

[19] Wikipedia definition, https://en.wikipedia.org/wiki/Single_source_of_truth.

[20] OMG Standards Development Organization, MBSE Wiki, Definition of Authoritative Source of Truth, https://www.omgwiki.org/MBSE/doku.php?id=mbse:authoritative_source_of_truth.

[21] L. Delligati, SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley 2013.

[22]SPARXS Systems Model Elements, https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/sysml_model_elements.html

[23] IBM UML Model Elements, https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=models-uml-model-elements

[24] ISO/IEC/IEEE 15288:2015 Systems and software engineering - System life cycle processes

[25] IBM Lifecycles, phases, and gates, portions adapted from https://www.ibm.com/docs/en/urbancode-release/6.1.1?topic=lifecycles-phases-gates