MIT-SE-21-246

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Investigation of Leading Indicators for Systems Engineering Effectiveness in Model-Centric Programs

23 September 2021

Dr. Donna H. Rhodes

Massachusetts Institute of Technology

Abstract

This technical report summarizes the research conducted by Massachusetts Institute of Technology under contract award HQ0034-19-1-0002 during July 22, 2019 – August 31, 2021. Involved research team members include: Dr. Donna H. Rhodes, Principal Investigator; Dr. Eric Rebentisch, Research Associate; and Mr. Allen Moulton, Research Scientist.

Systems engineering practice is evolving under the digital engineering paradigm, including use of model-based systems engineering and newer approaches such as agile. This drives a need to re-examine the existing use of metrics and leading indicators. Early engineering metrics were primarily lagging measures, whereas more recent leading indicators draw on trend information to provide more predictive analysis of technical and programmatic performance of the engineering effort. The existing systems engineering leading indicators were developed under the assumption of paper-based (traditional) systems engineering practice.

This research investigates the model-based implications relevant to the existing leading indicators. It aims to support program leaders, transitioning to model-based engineering on their programs, in continued use of leading indicators. It provides guiding insights for how current leading indicators can be adapted for model-based engineering. The study elicited knowledge from subject matter experts and performed literature review in identifying these implications. An illustrative case was used to investigate how four leading indicators could be generated directly from a model-based toolset.

Several recommendations for future research are proposed extending from the study. A companion research study ("phase 2") under contract HQ0034-20-1-0008 provides insights for the art of the possible for future systems engineering leading indicators and their use in decision-making on model-centric programs. For completeness, selected background information and illustrative case are included in the technical reports in both studies. This research aims to provide insights for current practice within programs transforming to digital engineering, for continued use of systems engineering leading indicators. Several recommendations for future research are proposed extending from results of the study.

About the Author

Dr. Donna H. Rhodes – Donna H. Rhodes is a principal research scientist at the Massachusetts Institute of Technology, and director of the Systems Engineering Advancement Research Initiative (SEAri). Dr. Rhodes conducts research on innovative approaches and methods for architecting complex systems and enterprises, designing for uncertain futures, and human-model interaction. Previously, she held senior management positions at IBM, Lockheed Martin, and Lucent. Dr. Rhodes is a Past President and Fellow of the International Council on Systems Engineering (INCOSE), and INCOSE Founders Award recipient. She received her Ph.D. in Systems Science from T.J. Watson School of Engineering at Binghamton University.

Massachusetts Institute of Technology
77 Massachusetts Avenue, E17-361
Cambridge, MA 02139
Tel: (617)-324-0473
rhodes@mit.edu

THIS PAGE INTENTIONALLY LEFT BLANK

MIT-SE-21-246

ACQUISITION RESEARCH PROGRAM
SPONSORED REPORT SERIES

Investigation of Leading Indicators for Systems Engineering Effectiveness in Model-Centric Programs

23 September 2021

Dr. Donna H. Rhodes

Massachusetts Institute of Technology

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. HQ0034-19-1-0002. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

THIS PAGE LEFT INTENTIONALLY BLANK

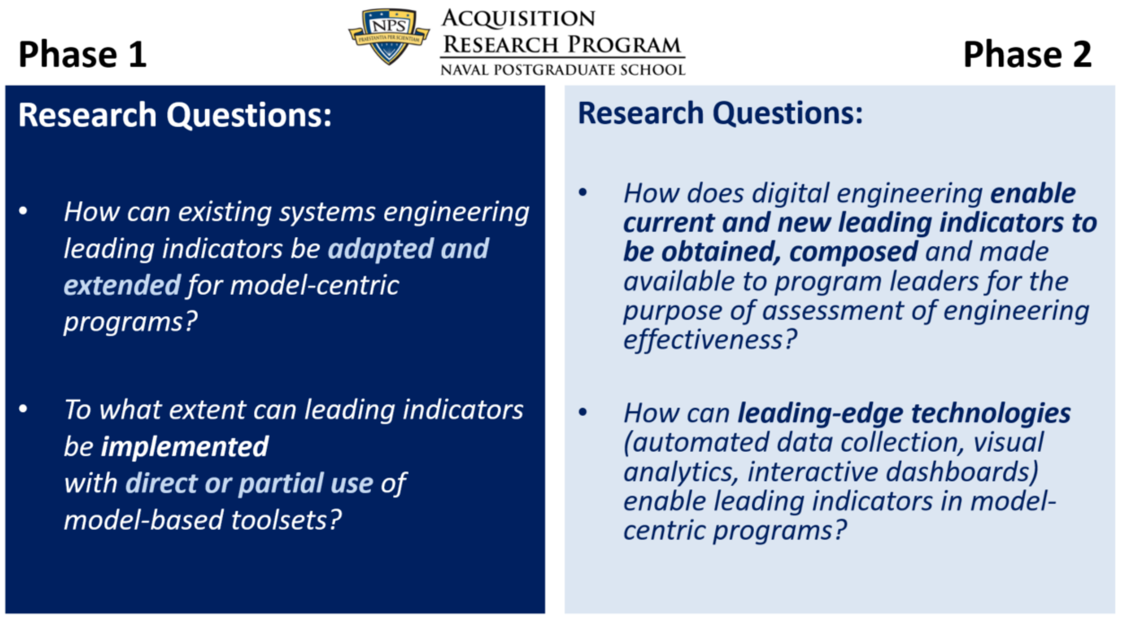
Table of Contents

Introduction

This report *HQ0034-19-1-0002 Investigation of Leading Indicators for Systems Engineering Effectiveness in Model-Centric Programs* discusses results of an empirical investigation related to systems engineering leading indicators. The focus of this research has been on investigating adaptation of existing systems engineering leading indicators for model-based engineering practice. Practical guidance on the model-based implications is generated through subject matter expert knowledge and literature investigation. An illustrative case is used to observe how four of the existing eighteen leading indicators could be generated using a model-based toolset. This research focuses on the present, investigating adaptation and extension of existing leading indicators for programs transitioning to model-based approaches.

This research was performed by Massachusetts Institute of Technology. Involved research team members include: Dr. Donna H. Rhodes, Principal Investigator; Dr. Eric Rebentisch, Research Associate; and Mr. Allen Moulton, Research Scientist.

A related research investigation under the NPS Acquisition Research Program, HQ0034-20-1-0008: *Phase 2: Investigation of Leading Indicators for Systems Engineering Effectiveness in Model-Centric Programs,* was initiated during the second year of this research and continued in parallel with this project. Figure 1 provides the two research questions explored in the two phases.


Phase 1

Research Questions:

- *How can existing systems engineering leading indicators be **adapted and extended** for model-centric programs?*
- *To what extent can leading indicators be **implemented** with **direct or partial use of** model-based toolsets?*

Phase 2

Research Questions:

- *How does digital engineering **enable current and new leading indicators to be obtained, composed** and made available to program leaders for the purpose of assessment of engineering effectiveness?*
- *How can **leading-edge technologies** (automated data collection, visual analytics, interactive dashboards) enable leading indicators in model-centric programs?*

Figure 1. Research questions in the each phase of research.

This second phase research took a future-oriented perspective, investigating opportunities afforded by use of model-based toolsets for composing leading indicators, as well as identifying leading-edge techniques to collect, compose and display measurement data for proactively assessing digital engineering effectiveness on model-centric programs. Selected background information on leading indicators and the illustrative case (used in both phases of the research) are included in each of the reports for completeness.

Background

Defense programs have long used engineering metrics to provide status and historical information, however implementation has been limited by the nature of the traditional, document-based engineering approach. Further, early systems engineering metrics were primarily lagging measures, providing information for the next program instead of the current one. Systems engineering leading indicators use an approach that draws on trend information to allow for more predictive insight (Rhodes, Valerdi, & Roedler, 2009).

A *systems engineering leading indicator* is a measure for evaluating the effectiveness of how a specific program activity impacts engineering effectiveness, which may affect the system performance objectives. As discussed in Zheng et al. (2019) both lagging and leading indicators are found to be useful in many fields (e.g., economic, health, social science). (While lagging measures (e.g., system defects) continue to provide useful information over time for an enterprise, they are insufficient for real-time decisions during a program.

Foundational work on systems engineering leading indicators was initiated in 2004, under the Lean Aerospace Initiative (LAI) at MIT. The systems engineering leading indicators were developed through a collaborative effort by government, industry and academia to allow for more timely predictive analysis of the technical and programmatic performance of the engineering effort on a program. Following publication of the initial guide with thirteen leading indicators, the collaborative team continued efforts resulting in publication of the *Systems Engineering Leading Indicators Guide, Version 2.0* (Figure 2). This second version of the guide (Roedler, G., Rhodes, D.H., Schimmoller, H. & Jones, C., 2010) included five additional leading indicators and several new appendices.

SYSTEMS ENGINEERING
LEADING INDICATORS
GUIDE

Version 2.0

January 29, 2010
Supersedes Initial Release, June 2007

Editors

Garry Roedler
Lockheed Martin Corporation
garry.j.roedler@lmco.com

Donna H. Rhodes
Massachusetts Institute of Technology
rhodes@mit.edu

Howard Schimmoller
Lockheed Martin Corporation
howard.l.schimmoller@lmco.com

Cheryl Jones
US Army
cheryl.jones5@us.army.mil

Developed and Published by Members of

INCOSE Technical Product Number: INCOSE-TP-2005-001-03

Copyright © 2010 by Massachusetts Institute of Technology, INCOSE, and PSM, subject to restrictions on page 2

Five additional indicators (18 total)

Applied Leading Indicator Implementation Guidance

- New Appendix A: NAVAIR's Systems Engineering Development & Implementation Center (SEDIC) use of SE leading indicators to develop advanced analysis techniques and toolkit for Navy programs
- New Appendix B: Human Systems Integration Considerations
- New Appendix C: Early Identification of Program Risks

Figure 2. Systems Engineering Leading Indicators Guide, version 2.0 (MIT, INCOSE, PSM)

Systems engineering leading indicators have continued to evolve through collaboration from organizations and individuals across the systems engineering community with over twenty organizations as contributors. Additional studies and papers have been published by various authors (including, Elm et al., 2008; Rhodes, et al., 2009; Montgomery & Carlson, 2010; Gerst & Rhodes, 2010; Knorr, 2012; Elm & Goldenson, 2013; Gilbert et al., 2014; Orlowski et al., 2015; Shirley, 2016; Orlowski, 2017; Zheng, et al., 2017; Zheng, et al., 2019).

Value of Leading Indicators

Effectiveness of systems engineering has been shown to be have positive relationship to the performance outcomes of projects and programs (Elm et al., 2008; Elm & Goldenson, 2013). A study by Orlowski (2017) shows the use of systems engineering measurement on a project as positively impacting the performance of the project; his findings are 59% of higher performance programs in his study had higher use of systems engineering leading indicators.

Leading indicators provide the most value when they give a proactive assessment that informs programmatic decisions and/or corrective actions. The Requirements Trend indicator, for instance, is used to evaluate trends in the growth, change, completeness and correctness of the definition of system requirements. Traditionally, this indicator provides insight into the rate of maturity of the system definition against the plan. Additionally, it characterizes stability and completeness of the system requirements that could potentially impact design, production, operational utility, or support.

One of the trend indicators, requirements volatility, has been used to drive milestone technical reviews. The graph in Figure 3 illustrates the rate of change of requirements over time.

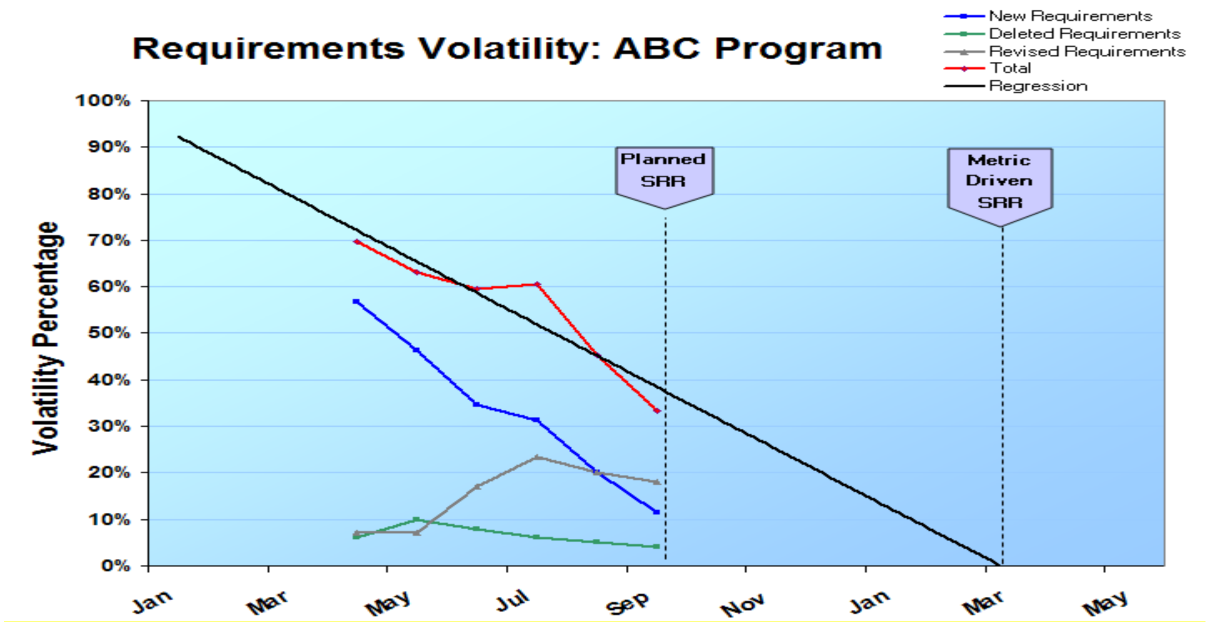


Figure 3. Illustrative Application of Leading Indicators on a Program (Rhodes, Valerdi, & Roedler, 2009)

The requirements volatility indicator also provides a profile of the types of change (new, deleted, or revised), allowing root-cause analysis of the change drivers. By monitoring the requirements volatility trend, the project team was able to predict the readiness for the System Requirements Review (SRR) milestone. In this example, the project team initially selected a calendar date to conduct the SRR, but in subsequent planning made the decision to have the SRR be event driven, resulting in a new date for the review wherein there could be a successful review outcome.

In traditional engineering practice, requirements are the central objects used for assessing maturity of system definition. In digital engineering, however, there are many other model constructs (e.g., activity diagrams) that are available and potentially composable to inform assessment of maturity for system definition. As such, it is important to understand model-based implications for leading indicators. One of the expected outcomes of digital model-based engineering is to move away from milestone design reviews to more continuous reviews using direct reviewer access to the maturing system model. Leading indicators can be very supportive of this goal (Orlowski et al., 2015). An open question is how the trend information regarding the full set of digital artifacts (e.g., SysML diagrams) could be used in a similar manner to predict when the model is in a state where a review activity is most useful.

Motivation and Research Approach

The broad motivation for the research, as well as the second phase study, is to enable more timely and informed decisions on systems engineering activities and resources. While use of systems engineering measures is standard part in traditional practice, its limitations are acknowledged. Systems engineering leading indicators overcome some of the limitations but until recently collecting the underlying data and performing analysis has been constrained by document-driven engineering practice. As the use of model-based approaches and tools in systems engineering increases, the increased ease of generating systems engineering leading indicators will make these more tractable for systems programs.

Model-based systems engineering (MBSE) is defined 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." (OMG). The transformation to digital engineering has prompted a need to re-examine the systems engineering leading indicators for this new context. The investigation of the two phases of this research aims to provide findings for model-centric programs seeking to use the leading indicators, as well as contribute recommendations to inform the larger effort of the systems engineering community to establish the next generation of digital engineering effectiveness measurement. This report focuses on the present use of leading indicators, whereas the second report takes a future orientation.

Model-based systems engineering (and other digital engineering tools) are recognized as a means to increase engineering efficiency (DoD, 2018, p.17; McDermott et al., 2020). Leading indicators are especially important to monitoring effectiveness of engineering on a continuous basis, ensuring that effectiveness is not compromised for sake of efficiency. The DoD Digital Engineering Strategy calls for leadership to “establish accountability to measure, foster, demonstrate, and improve tangible results across programs and the enterprise” (DoD, 2018, p. 22). Common enabling technologies used in digital environments to generate, analyze and display measurement data will encourage a common foundation for cross-program comparison and learning.

As discussed, existing leading indicators were developed under the "paper-based" or document-based engineering approach. The introduction of digital engineering, with model-based systems engineering practices, has potentially radical or disruptive impact on the processes, tools, and timelines of engineering programs. Research is necessary in order to understand and adapt existing systems engineering indicators for model-based/ digital engineering and management practice in model-centric programs.

The research sought to address these two questions:

1. How can existing leading indicators to adapted and extended for model-centric programs?
2. To what extent can leading indicators be implemented with direct or partial use of model-based toolsets?

The digital engineering environment and newer technologies open new possibilities for providing program leaders with leading insights into the effectiveness of systems engineering efforts. Since each of the indicators requires some additional considerations under model-based systems engineering, this investigation focused on identifying potential modifications and guidance (Rhodes, 2020).

This research used literature review and knowledge gathering from subject matter experts through technical exchanges and workshops. This included investigation of publications, studies, workshop reports and interim research findings from academic research groups, professional societies, industry associations, and cross-industry initiatives. Findings were used to generate insights regarding adaptation of existing leading indicators for use by programs transitioning to model-based practice. Model-based implications are identified for each of the existing eighteen systems engineering leading indicators. An illustrative case was used by the research team to explore how four leading indicators could be generated directly from a model-based toolset.

THIS PAGE LEFT INTENTIONALLY BLANK

Systems Engineering Leading Indicator Specification

The systems engineering leading indicators are described using an industry standard measurement specification. The current set of eighteen systems engineering leading indicators is listed below, citing leading insights that are provided to the program.

Requirements Trends: Rate of maturity of the system definition against the plan. Additionally, characterizes the stability and completeness of the system requirements that could potentially impact design, production, operational utility, or support.

System Definition Change Backlog Trends: Change request backlog which, when excessive, could have adverse impact on the technical, cost and schedule baselines.

Interface Trends: Interface specification closure against plan. Lack of timely closure could pose adverse impact to system architecture, design, implementation and/or V&V any of which could pose technical, cost and schedule impact.

Requirements Validation Trends: Progress against plan in assuring that the customer requirements are valid and properly understood. Adverse trends would pose impacts to system design activity with corresponding impacts to technical, cost & schedule baselines and customer satisfaction.

Requirements Verification Trends: Progress against plan in verifying that the design meets the specified requirements. Adverse trends would indicate inadequate design and rework that could impact technical, cost and schedule baselines. Also, potential adverse operational effectiveness of the system.

Work Product Approval Trends: Adequacy of internal processes for the work being performed and also the adequacy of the document review process, both internal and external to the organization. High reject count would suggest poor quality work or a poor document review process each of which could have adverse cost, schedule and customer satisfaction impact.

Review Action Closure Trends: Responsiveness of the organization in closing post-review actions. Adverse trends could forecast potential technical, cost and schedule baseline issues.

Technology Maturity Trends: Risk associated with incorporation of new technology or failure to refresh dated technology. Adoption of immature technology could introduce significant risk during development while failure to refresh dates technology could have operational effectiveness/customer satisfaction impact.

Risk Exposure Trends: Effectiveness of risk management process in managing / mitigating technical, cost& schedule risks. An effective risk handing process will lower risk exposure trends.

Risk Treatment Trends: Effectiveness of the systems engineering organization in implementing risk mitigation activities. If the systems engineering organization is not retiring risk in a timely manner, additional resources can be allocated before additional problems are created.

Systems Engineering Staffing & Skills Trends: Quantity and quality of systems engineering personnel assigned, the skill and seniority mix, and time phasing of their application throughout project lifecycle.

Process Compliance Trends: Quality and consistency of the project defined systems engineering process as documented in SEP/SEMP. Poor/inconsistent systems engineering processes and/or failure to adhere to SEP/SEMP, increase project risk.

Technical Measurement Trends: Progress towards meeting the Measures of Effectiveness (MOEs) / Performance (MOPs) / Key Performance Parameters (KPPs) and Technical Performance Measures (TPMs).

Systems Engineering Staffing & Skills Trends: Quantity and quality of SE personnel assigned, the skill and seniority mix, and the time phasing of their application throughout the project lifecycle.

Process Compliance Trends: Quality and consistency of the project defined SE process as documented in SEP/SEMP. Poor/inconsistent SE processes and/or failure to adhere to SEP/SEMP, increase project risk.

Facility and Equipment Availability Trends: Availability of non-personnel resources (infrastructure, capital assets, etc.) needed throughout the project lifecycle.

Defect/Error Trends: Progress towards the creation of a product or the delivery of a service that meets the quality expectations of its recipient. Understanding the proportion of defects being found and opportunities for finding defects at each stage of the development process of a product or the execution of a service.

System Affordability Trends: Progress towards a system that is affordable for the stakeholders. Understanding the balance between performance, cost, and schedule and the associated confidence or risk.

Architecture Trends: Maturity of an organization with regards to implementation and deployment of an architecture process that is based on an accept set of industry standards and guidelines.

Schedule and Cost Pressure: Impact of schedule and cost challenges on carrying out a project. Indicates whether the project performance can be adversely effected by efforts to meet customer expectations.

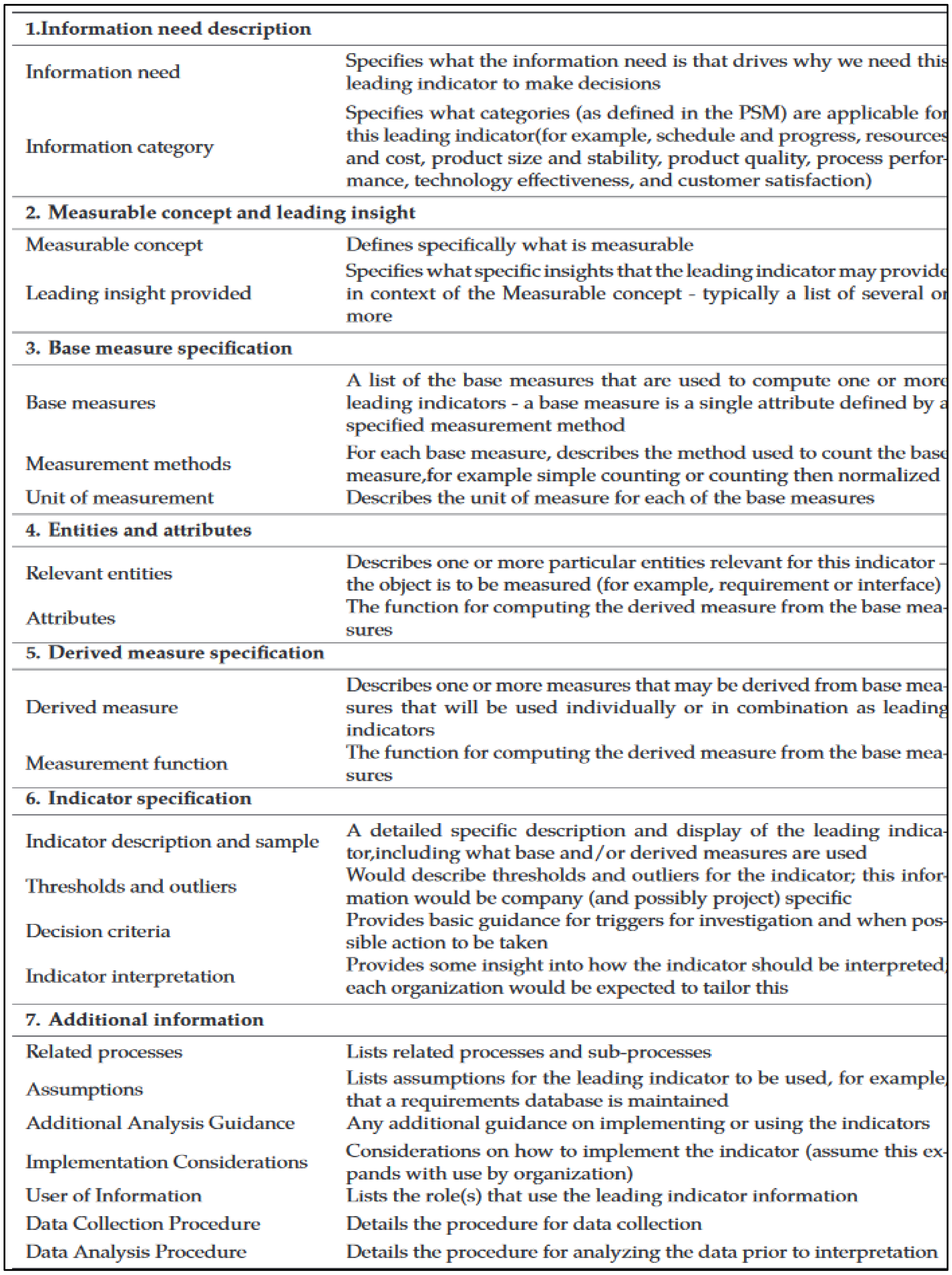
Each of these leading indicators is fully described using a standard measurement specification.

Measurement Specifications

The systems engineering community has been using measurement specifications for many years, based on foundational work of PSM in software and systems measurement (PSM, 2020). The systems engineering leading indicators initiative adopted the PSM measurement specification format. Accordingly, each of the eighteen systems engineering indicators is characterized using a measurement specification with detailed description, insights provided, interpretation guidance and usage guidance. Detailed contents of the measurement specifications for the eighteen leading indicators is described in Roedler et al. (2010).

Table 1 describes the content that is included in the specification. There are seven sections in the specification: (1) information need description; (2) measurable concept and leading insight; (3) base measure specification; (4) entities and attributes; (5) derived measure specification; (6) indictor specification; and (7) additional information. Each section has a set of information within it, providing a comprehensive description for the leading indicator.

Table 1. Systems Engineering Leading Indicator Specification Fields. Source: adapted by (Zheng L. et al., 2019) from (Roedler G. J., Rhodes, D.H., Schimmoler, H. & Jones, C. 2010).



1.Information need description	
Information need	Specifies what the information need is that drives why we need this leading indicator to make decisions
Information category	Specifies what categories (as defined in the PSM) are applicable fo this leading indicator(for example, schedule and progress, resources and cost, product size and stability, product quality, process perfor-mance, technology effectiveness, and customer satisfaction)
2. Measurable concept and leading insight	
Measurable concept	Defines specifically what is measurable
Leading insight provided	Specifies what specific insights that the leading indicator may provid in context of the Measurable concept - typically a list of several o more
3. Base measure specification	
Base measures	A list of the base measures that are used to compute one or mor leading indicators - a base measure is a single attribute defined by a specified measurement method
Measurement methods	For each base measure, describes the method used to count the bas measure,for example simple counting or counting then normalized
Unit of measurement	Describes the unit of measure for each of the base measures
4. Entities and attributes	
Relevant entities	Describes one or more particular entities relevant for this indicator - the object is to be measured (for example, requirement or interface)
Attributes	The function for computing the derived measure from the base mea-sures
5. Derived measure specification	
Derived measure	Describes one or more measures that may be derived from base mea-sures that will be used individually or in combination as leading indicators
Measurement function	The function for computing the derived measure from the base mea-sures
6. Indicator specification	
Indicator description and sample	A detailed specific description and display of the leading indica-tor,including what base and/or derived measures are used
Thresholds and outliers	Would describe thresholds and outliers for the indicator; this infor-mation would be company (and possibly project) specific
Decision criteria	Provides basic guidance for triggers for investigation and when pos-sible action to be taken
Indicator interpretation	Provides some insight into how the indicator should be interpreted, each organization would be expected to tailor this
7. Additional information	
Related processes	Lists related processes and sub-processes
Assumptions	Lists assumptions for the leading indicator to be used, for example, that a requirements database is maintained
Additional Analysis Guidance	Any additional guidance on implementing or using the indicators
Implementation Considerations	Considerations on how to implement the indicator (assume this ex-pands with use by organization)
User of Information	Lists the role(s) that use the leading indicator information
Data Collection Procedure	Details the procedure for data collection
Data Analysis Procedure	Details the procedure for analyzing the data prior to interpretation

Prior Research on Augmenting Measurement Specifications

Leading indicators for assessing the effectiveness of systems engineering on a program are expected to be more tractable and more useful in model-centric programs of the future. A necessary first step undertaken in this research was to re-examine the existing set of systems engineering leading indicators to understand impacts of digital engineering on the defined measurement specifications. The intended outcome of the longer term effort will be to extend and adapt these fields as needed, and in the process to identify candidates for new leading indicators. Prior research informs this work, showing the value of continuing to mature systems engineering leading indicators.

The approach of augmenting existing measurement specifications was also used in research that investigated enhancing consideration of human systems integration (HSI) in systems engineering. HSI is the integrated, comprehensive analysis, design and assessment of requirements, concepts, and resources for system Manpower; Personnel, Training, Environment, Safety, Occupational Heath, Habitability, Survivability and Human Factors Engineering. Accordingly, HSI is tightly coupled with the systems engineering process, particularly in large defense and government programs, making it challenging to determine if HSI is sufficiently considered to ensure a successful program. It is also challenging to isolate and identify HSI issues, particularly in early stages of acquisition programs. The objective in the HSI leading indicators research was to augment and extend the current systems engineering leading indicators, including interpretive guidance, to enhance the predictability of programmatic and technical performance on a program to include adequate HSI consideration. As a means to understand how the existing leading indicators may be augmented and the set of indicators extended to include additional useful indicators, the approach employed was to gather expert data through workshop discussions, surveys, and interviews. In this discovery process, the goal was to identify observations and “soft indicators” (that is, early insights and qualitative indicators) as a first step toward developing mature leading indicators drawing on quantitative information (Rhodes, Valerdi, & Roedler, 2009). The investigation of adapting the leading indicators confirmed that basic augmenting of information in the measurement specification could be beneficial as a first step (Rhodes, Valerdi, Gerst, & Ross, 2009). In addition to adapting existing indicators, the findings led to a new proposed leading indicator focused on involving end-users in design (Gerst & Rhodes, 2010). Accordingly, this approach of considering adaptation and extension of the measurement specifications was adopted for this research. The initial outcome was to identify model-based implications to inform future work related to the measurement specifications.

Model-Based Implications for Leading Indicators

The existing eighteen leading indicators, as investigated through literature, semi-structured interviews and technical exchange workshops, were shown in this research to have varying implications related to model-based systems engineering. Implementation of a leading indicators in context of digital engineering will be based on many factors, such as nature of the program, processes used by the enterprise, model-based toolset selection and implementation, engineering culture of the enterprise, and maturity of digital engineering in the enterprise, as well as external influences (e.g., customer preferences, etc.). Using research findings, the leading indicators are grouped into three subsets: (1) leading indicators most likely to be implemented with direct use of a model-based toolset; (2) leading indicators most likely to be partially implemented with use of a model-based toolset; and (3) leading indicators less likely to be implemented with use of a model-based toolset. First, two leading indicators are discussed in detail. Following this, the three groups of leading indicators are then summarized in Tables 2, 3 and 4 to highlight identified model-based implications, respective to come of the insight provided (see Roedler et al., 2010 for more information on leading insights). The cited implications are based on subject matter expert opinion, and should not be considered as complete, nor demonstrated in practice.

Discussion of Model-Based Implications for Two Leading Indicators

In this section, two of the leading indicators are discussed. First, the *requirements trends* leading indicator, one of the indicators that is most likely to be implemented with use of model-based toolsets, is discussed including perspectives on near term and longer term implications. Second, the *facility and equipment availability trends* leading indicator is discussed

Requirements Trends Leading Indicator

The *Requirements Trends* leading indicator is used to evaluate the stability and adequacy of the requirements to understand the risks to other activities towards providing required capability, on-time and within budget. This is done through an evaluation of trends in the growth, change, completeness and correctness of the system requirements definition, as well as the quality of and consensus around the system operations concept. This indicator provides insight into rate of maturity of the system definition against the plan, and whether the system definition is maturing as expected. Additionally, it characterizes the stability and completeness of system requirements that could potentially impact design, production, operational utility, or support. Requirements growth, changes, or impacts that exceed expectations or exhibit a lower closure rate of TBDs/TBRs than planned may indicate insufficient quality of architecture, design, implementation, verification, and validation efforts. This in turn could result in elevated schedule and cost risks, and/or a future need for different levels or types of resources/skills.

Near Term: The use of requirements management tools and databases is a mature practice in systems engineering. Tracking the growth trends and volatility of requirements is therefore a relatively straightforward matter of the compilation of data on the requirements within the database and the development of processes for regular review and action where implied. These functions could be incorporated into or added to existing requirements management tools within the MBSE environment to assist program decision makers in assessing progress during the system development.

Longer Term: MBSE tools and methods introduce a number of new ways to assess and understand the quality of requirements and the degree to which they are being met over the course of the system development lifecycle. A transition to primary use of an MBSE approach in system development could enable a broader range of analysis and model checking.

The expression of requirements as executable models as has been demonstrated to improve the quality of requirements and decrease errors relating to poorly-defined requirements (Micoun, P. et al., 2018). Model-based requirements provide the ability to validate that the system model is logically consistent, and the ability to answer questions such as the impact of a requirement or design change, or the assessment of how a failure could propagate through a system. Using this approach it is possible to verify design models using a simulation-based verification process in order to detect and remove design errors. Model-based requirements may be included in a curated database for reuse in other development efforts, with the potential for savings in time and resources.

Model-based requirements may be used in early system analysis to assess requirements completeness and correctness through the identification of gaps, conflicts, or redundancies in the existing requirements set, prior to the development of more detailed engineering models and analysis. MBSE analysis using model-based requirements could validate the requirements themselves and ensure that they do not contribute to undesirable emergent behaviors at the system level. A potential indicator of requirements quality in the MBSE environment might include the percentage of requirements that are formatted and expressed as models and the rate and total proportion of requirements validation through modelling and simulation at both the component and system level.

Model-based requirements may be archived and reused across multiple development projects. Any issues that are identified in the requirements for one project could potentially be traced to other projects that use the same models. The traceability inherent in using these archived requirements models enables enhanced root cause analysis and system refinement, triggering actions to correct and validate the originating requirement to prevent continuing propagation of errors. An indicator of requirements maturity in a MBSE environment might include the proportion of requirements models that include a validation pedigree. The presence of requirements models without a validation pedigree (at least to a specific standard defined by the enterprise) could indicate greater risk of potential future requirements changes and instability in the system baseline.

Facilities and Equipment Trends Leading Indicator

The *Facility and Equipment Availability Trends* leading indicator is used to determine the availability of critical facilities and equipment needed for systems engineering activities over the project lifecycle. The indicator is composed of two metrics, measuring facility availability and equipment availability. The intent of this indicator is to provide a view of facility and equipment availability on the project over time. Facilities and equipment are of different types and may provide key capabilities to the program. The Facility Availability measurement provides insight into the difference between the planned need for a facility type and the existing inventory of available facilities that meets the need for the desired capability. Insufficient facilities (e.g., labs, test ranges, floor space, etc.) of various types may cause a project to be unable to meet its customer needs, create costly overruns, and inability to meet schedule targets. Similarly, project requires various types of equipment that also may provide key capabilities for the program. Equipment availability measurement provides insight into the difference between the planned need for an equipment type and the existing inventory of available inventory that meets the need for the desired capability. Insufficient equipment (fabrication equipment, measurement equipment, cleanroom equipment, test equipment, software and systems applications, etc.) may cause a project to be unable to meet its customer needs, create costly overruns, and inability to meet schedule targets. Facility Availability and Equipment Availability as measurable concepts assess whether adequate facilities and equipment can be allocated to the project to meet lifecycle milestones. This reveals differences between systems engineering needs on the project and facilities and equipment projected to be available based on existing plans. The leading insights provided to the project are potential shortfalls of systems engineering related facilities and equipment, and potential problems with the project's ability to meet desired milestones. (Roedler et al., 2010).

Near Term: As an initial step in adapting the existing SE leading indicators, the measurement specification can be augmented by adding model-based systems engineering implications to the Implementation Considerations within the Additional Information section of the measurement specification. Model-based programs necessitate personnel have (or have access to) computing "equipment", to include desktop/laptop computers or workstations with adequate performance, access to networks and/or intranet, data and model repositories, model libraries, computer services support, data/cloud storage, etc. Facilities may include the individual engineer's workspace, as well as collaborative spaces. There is also a need to have access to the selected version of model-based toolset that is maintained. The facilities and equipment need to support any required upgrades of versions, which may have implications for the existing computing facilities. Another implication consideration is that facilities and equipment must accommodate any necessary collaboration with other internal groups and/or external organizations (e.g., a supplier or customer) as needed. The facilities and equipment must be adequate to support this. This includes necessary facilities and equipment to support tool interoperability, data/model exchange, version compatibility control, model sharing, model security, etc. Model-based programs need to have adequate budget allocated, as insufficient availability of the necessary facilities and equipment will have major impact on systems engineering effectiveness.

Longer Term: As we look to the future of digital engineering, the issue with using the existing Facilities and Equipment Availability leading indicator is that it takes a somewhat decoupled approach at these rather than as highly interconnected, as is the case for MBSE. In fact, with the transformation of traditional engineering to digital engineering, there is a need to look at this in context of the larger digital ecosystem. This includes interconnected digital environments that extend beyond the boundaries is the engineering organization. In the existing SE leading indicator guide published in 2010, the Facilities and Equipment Leading Indicator has relatively less substance than other indicators given it was not a major focus of the team. With digital engineering transformation, taking the perspective of the overall digital engineering ecosystems is necessary. The success of systems engineering on a program will be fully dependent upon the environment and infrastructure available to participate as part of the larger ecosystem. The supporting infrastructure required for digital engineering (Bone, et al., 2018) necessities a new leading indicator be developed respective to the importance it has to system success and the dimensions and complexity of that infrastructure.

Leading Indicators Most Likely to Be Implemented with Direct Use of a Model-Based Toolset

As discussed, implementing leader indicators with a model-based toolset will vary based on the ontology, selected toolset, and details of how the toolset is used. Based on knowledge gathered in this research, the eighteen indicators were organized in three subsets.

The first subset of leading indicators, as shown in Table 2, are those that are most likely to be implemented with the direct use of the program's MBSE toolset. In this case, the base measures as shown in the respective measurement specifications in the leading indicator guide (Roedler, et al., 2010) are likely to be obtained from the system model and composed into a leading indicator. Assuming an effective user interface and any required trend data, this could provide the ability to obtain real-time leading indicator information to better inform and accelerate decisions.

Table 2. Leading Indicators Most Likely to Be Implemented with Direct Use of Model-Based Toolset

Leading Indicators Most Likely to Be Implemented with Direct Use of Model-Based Toolset		
Leading Indicator	**Insight Provided (source: 2010 guide)**	**Model-Based Implications**
Requirements Trends	Rate of maturity of the system definition against the plan. Additionally, characterizes the stability and completeness of the system requirements that could potentially impact design, production, operational utility, or support.	See subsection 3.1.1 for a detailed discussion
System Definition Change Backlog Trend	Change request backlog which, when excessive, could have adverse impact on the technical, cost and schedule baselines.	Model-based tools will enable collection and analysis of change data MBSE enables fixing defects earlier in time, where less effort is typically required. Accordingly, historical trends will vary from model-centric programs
Interface Trends	Interface specification closure against plan. Lack of timely closure could pose adverse impact to system architecture, design, implementation and/or V&V any of which could pose technical, cost and schedule impact.	Similar to requirements trends Model-based implementation can enable useful information (e.g., risks, action items, rationale) to be directly associated with interfaces. Accordingly, this may have a positive impact on resolving issues given engineers will more easily access information
Requirements Validation Trends	Progress against plan in assuring customer requirements are valid and properly understood. Adverse trends would pose impacts to system design activity with corresponding impacts to technical, cost & schedule baselines and customer satisfaction.	Related to requirements trends; see subsection 3.1.1 Since model-based tools may accelerate the pace of validation, historical trend data may not be as useful Model-based implementation will enable real-time measurement of validation
Requirements Verification Trends	Progress against plan in verifying design meets the specified requirements. Adverse trends would indicate inadequate design and rework that could impact technical, cost and schedule baselines. Also, potential adverse operational effectiveness of the system.	Related to requirements trends; see subsection 3.1.1 Since model-based tools may accelerate the pace of verification, historical trend data may not be as useful Model-based implementation will enable real-time measurement of verification

Leading Indicators Most Likely to Be Partially Implemented with Use of a Model-Based Toolset

The second subset of leading indicators, as shown in Table 3 , are those that are most likely to be partially implemented with the use of the program's model-based toolset. For example, technical performance risk information might be associated with the system model, but there may be other programmatic risk information that is tracked elsewhere. The extent to which the five leading indicators in this table are able to be generated from a model is dependent on what types of models the program uses, and how model-based toolsets are customized and extended.

Table 3. Leading Indicators Most Likely to Be Partially Implemented with Use of Model-Based Toolset

Leading Indicators Most Likely to Be Partially Implemented with Use of Model-Based Toolset		
Leading Indicator	**Insight Provided (source: 2010 guide)**	**Model-Based Implications**
Risk Exposure Trends	Effectiveness of risk management process in managing / mitigating technical, cost & schedule risks. An effective risk handing process will lower risk exposure trends.	Model-based toolsets provide opportunity to associate risk with or directly include risk within models, providing engineers with timely and enhanced visibility
Risk Treatment Trends	Effectiveness of the SE organization in implementing risk mitigation activities. If SE is not retiring risk in a timely manner, additional resources can be allocated before additional problems are created.	Model-based toolsets provide opportunity to associate risk with or directly include risk within models, providing engineers with timely status on handling/treatment of risks Historical trend data may vary from traditional engineering programs
Technical Measurement Trends	Progress towards meeting the Measures of Effectiveness (MOEs) / Performance (MOPs) / Key Performance Parameters (KPPs) and Technical Performance Measures (TPMs). Lack of timely closure is an indicator of performance deficiencies in the product design and/or project team's performance.	Model-based approaches, methods and tools will enhance technical performance measurement through direct access to base measures Ability to project planned value and predict variances may be improved with model-based approach, including executable models and simulations Tolerance bands may vary from traditional engineering

Leading Indicators Most Likely to Be Partially Implemented with Use of Model-Based Toolset		
Leading Indicator	**Insight Provided (source: 2010 guide)**	**Model-Based Implications**
Defect/Error Trends	Progress towards the creation of a product or the delivery of a service that meets the quality expectations of its recipient. Understanding the proportion of defects being found and opportunities for finding defects at each stage of the development process of a product or the execution of a service.	With model-based approach errors and defects may be found earlier in time; software can automate finding and fixing some defects Since models are not measured in pages, defining an alternative to the 'defects per page' metric will be required Historical defect discovery profiles from traditional engineering will likely not be suitable; defects models and discovery profiles will need to be developed as experience in model-centric programs grows
Work Product Approval Trends	Adequacy of internal processes for the work being performed and also the adequacy of the document review process, both internal and external to the organization. High reject count would suggest poor quality work or a poor document review process each of which could have adverse cost, schedule and customer satisfaction impact.	Models are likely to become tracked work products in model-centric programs; criteria would need to be developed for this purpose Models may influence approval rate of system work products, as well as approach to how model-based product approval is performed; trend data would need to be developed for model-centric programs

Leading Indicators Less Likely to Be Implemented with Use of Model-Based Toolset

The third subset of leading indicators, as shown in Table 4 are those that are less likely to be implemented with the use of a program's model-based toolset. Presently, these leading indicators would likely be tracked in a separate technical management tool or tracking system. Model toolset experts view it as possible to extend model-based toolsets to include any programmatic and process models in a model-centric environment. While at present there are likely to be few programs that have implemented this, likelihood will increase over time as model-based environments evolve.

Table 4. Leading Indicators Less Likely to Be Implemented with Use of Model-Based Toolset

Leading Indicators Less Likely to Be Implemented with Use of Model-Based Toolset		
Leading Indicator	**Insight Provided (source: 2010 guide)**	**Model-Based Implications**
Technology Maturity Trends	Risk associated with incorporation of new technology or failure to refresh dated technology. Adoption of immature technology could introduce significant risk during development while failure to refresh dates technology could have operational effectiveness/ customer satisfaction impact.	Increased use of models is likely to enhance ability to measure potential impacts of immature technology Models may enable engineers to understand cascading impacts of technology maturity in a system, and accordingly could provide leading indicator information.
Review Action Closure Trends	Responsiveness of the organization in closing post-review actions. Adverse trends could forecast potential technical, cost and schedule baseline issues.	Selected information to support the tracking of action item closure may be generated from model-based toolset Technical-related action items may be directly linked to models Model-centric programs may have more continuous action item review than traditional programs
Systems Engineering Staffing & Skills Trends	Quantity and quality of SE personnel assigned, the skill and seniority mix, and the time phasing of their application throughout the project lifecycle.	Insufficient model-based staffing/skills have impact on cost, schedule and quality Model-based approaches, methods, and tools require additional staffing and skills, possibly at different points in program Attributes of model-based environments can be used to identify staffing and skill needs
Process Compliance Trends	Quality and consistency of the project defined SE process as documented in SEP/SEMP. Poor/inconsistent SE processes and/or failure to adhere to SEP/SEMP, increase project risk.	Model-based programs will be using newer processes and/or developing processes integrated with toolsets In the future SE processes and plans may be implemented in a model-based manner Compliance deviations and comments recorded within the

Leading Indicators Less Likely to Be Implemented with Use of Model-Based Toolset		
Leading Indicator	**Insight Provided (source: 2010 guide)**	**Model-Based Implications**
		model enable automated compliance measurement Process compliance measurement needs to accommodate modifications to process given learning on program and/or other programs
Facility and Equipment Availability Trends	Availability of non-personnel resources (infrastructure, capital assets, etc.) needed throughout the project lifecycle.	See subsection 3.1.2 for a detailed discussion
System Affordability Trends	Progress towards a system that is affordable for the stakeholders. Understanding the balance between performance, cost, and schedule and the associated confidence or risk.	Assessing affordability under the digital engineering paradigm is likely to require different approach than traditional engineering Lacking historical data, model-based programs need to develop approach and adapt measurement of affordability Trend data will likely vary from traditional programs
Architecture Trends	Maturity of an organization with regards to implementation and deployment of an architecture process that is based on an accepted set of industry standards and guidelines.	Model-based approaches/tools will have influence on assessing maturity of architecture process Model-based toolsets aim for alignment with industry standards Programs should tailor base measures as needed to reflect advantages of model-based approaches/tools
Schedule and Cost Pressure	Impact of schedule and cost challenges on carrying out a project	Minimal historical data available for digital engineering situation Setting notional values for thresholds may be challenging

Summary

In the near term, the existing measurements specifications can be augmented with model-based implications that can help inform model-centric programs. In this research exploring implications, three subsets emerged. In the future, modified and new measurement specifications are envisioned in a potential new release of the leading indicators guide.

As programs using systems engineering leading indicators transform to digital engineering, they will need to consider what specific leading indicators can be implemented through their chosen model-based toolset and the manner in which these will be used. The unique implementation of leading indicators through model-based toolsets will depend upon the program's chosen ontology, toolset, and specifics of how the toolset is used by the engineering team. Initially, the lack of historical trend information will initially necessitate use of expert judgement rather than data.

In the following section, we describe results of experimenting with a small illustrative case, using one of the currently available ontologies and an associated toolset. This example is intended to offer a glimpse of the thought process that a program will need to go through in adapting leading indicators under the digital engineering paradigm.

THIS PAGE LEFT INTENTIONALLY BLANK

Illustrative Case

An illustrative case is used to examine how digital engineering (DE) is expected to change systems engineering practice compared to traditional document-driven methods, as relevant to leading indicators. This case assisted in exploring how selected leading indicators would most likely be modified or adapted for implementation with direct use of model-based toolsets. It should be noted that this example is unique to the ontology and toolset used for this experimentation. This section discusses:

- An illustrative case of a self-driving autonomous vehicle development,
- An overview of the LML ontology used for knowledge representation,
- Explicit natural language requirements in traditional and digital model-based methods, and methods for identifying, representing and managing TBDs/TBRs in DE,
- Functional analysis in DE and using relationships to help identify strengths and weaknesses in the model,
- Physical system modeling in DE and how physical model conduits can be used to identify external and internal interfaces,
- Examples of validation and verification using integrated spider diagrams and discrete event simulation, and
- Deriving leading indicator base measures from the system model.

Self-Driving Autonomous Vehicle Development Illustrative Case

The small illustrative case involves systems engineering applied to new development of a self-driving fully autonomous vehicle meeting the Level 5 ("Full Driving Automation") SAE International Standards[1].

[1] For more in-depth case examples, beyond the small illustrative case presented here, see Tepper (2010) for a more complete application of MBSE to Naval Ship Design using SysML and Dam (2019) for an application of LML and MBSE to a hypothetical establishment of a permanent manned base on the moon.

For the purposes of this case, the cloud-based Innoslate® software tools were used to conduct a number of small-scale exercises[2]. Innoslate® is an integrated MBSE software package that implements the open source LML Ontology, which is compact but comprehensive (Dam, 2019, p.5). The LML Ontology provides a guiding structure for investigating how information needed for leading indicators is represented in MBSE. Dam (2019) states the ontology is a compact, but comprehensive, organized, structured and customizable terminology for systems engineering from the earliest concept stage throughout the lifecycle to final system disposal (Dam, 2019, p. 6, 10). Vaneman (2018) reports that LML has sufficient constructs to be able to represent knowledge expressed or expressible in other modeling languages, such as SysML and DODAF.

Innoslate enforces the important principle of concordance, which facilitates single source of truth by requiring that a given piece of information in the systems engineering knowledge base will have the same meaning when viewed through different language or visualization lenses.

Although the version of the Innoslate tools used in these experiments is implemented on a central cloud database, more complex configurations with virtual integration of data stored in multiple physical locations are also feasible. Current generation database and semantic web technology would also support such virtual integration provided that the semantics of the data can be made compatible.

LML Ontology used for Knowledge Representation in the Case

The LML ontology knowledge framework is built on an Entity-Relationship-Attribute (ERA) data model. ERA was first introduced by Peter Chen (Chen, 1976) and is widely used today for conceptual modeling including by UML (https://www.omg.org/spec/UML/) and other methods. Entities represent things of interest (analogous to nouns in natural language). Relationships represent connections across entities. Attributes represent information about an entity or relationship. Standard entity attributes are defined in the

[2] This tool set was selected based on its ready availability to the research team, and should not viewed as an endorsement or preference for any tool or tool vendor company.

LML Ontology along with standard relationships and the entity types they connect (LML Steering Committee, 2015).

The LML Ontology provides definitions of typical relationship types for each combination of entity types (see Vaneman, 2018). Relationships are directional and matched with complementary relationships that go in the opposite direction (e.g., "performed by" and "performs" are complements). The primary entity structure of the LML Ontology is shown in Figure 4.

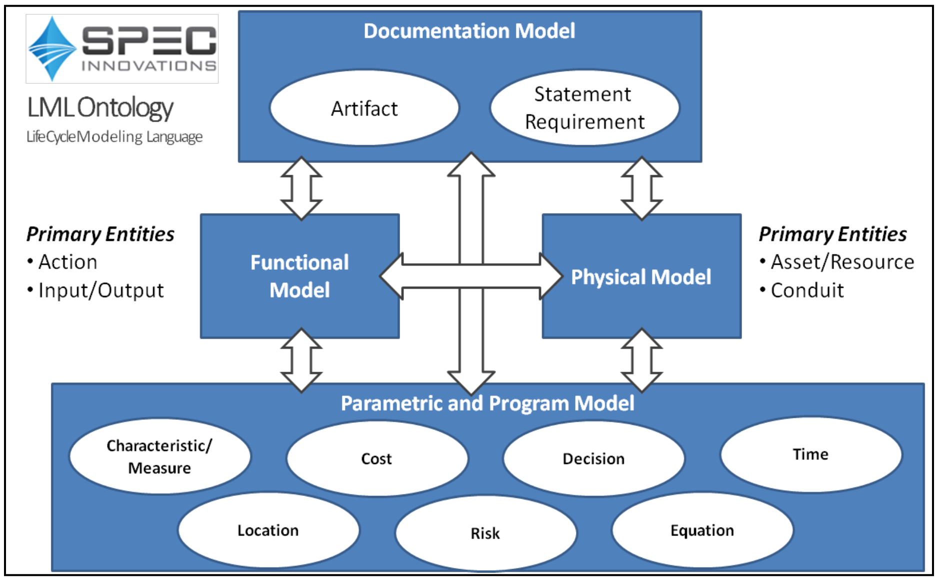


Figure 4. LML Ontology Primary Entity Classes (Dam, 2019)

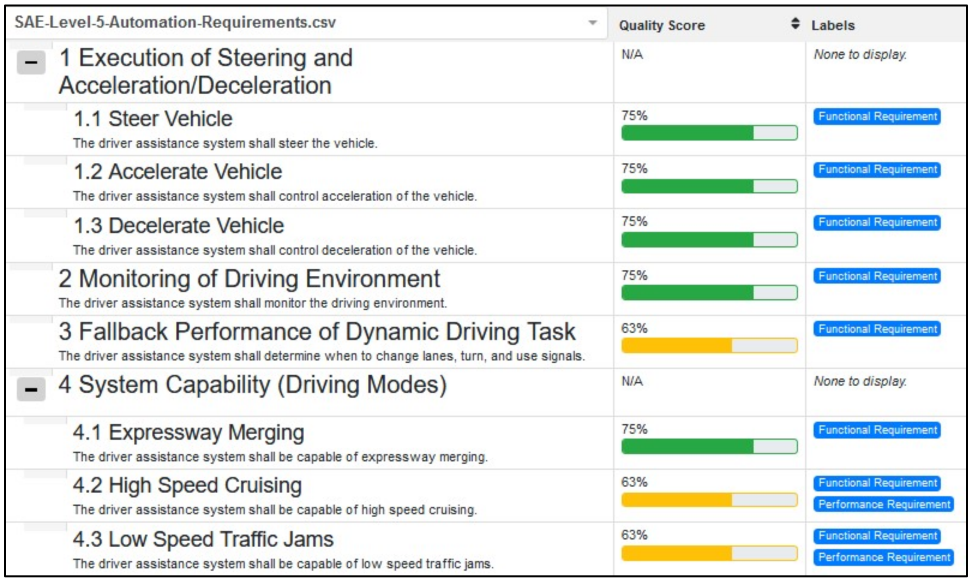
The Documentation Model section at the top of the ontology diagram in Figure 4 includes entity classes that are ancillary to the system model and intended for exchanging information with humans or other outside systems. The Artifact entity represents a document, spreadsheet, test plan, or other source of information that is referenced by, or generated into, the knowledgebase. A Statement entity specifies text that is usually drawn from an Artifact. A Requirement entity is a Statement that expresses a capability or characteristic of a system that must be present for the system to have value to users. The LML Requirement entity is similar to the Requirement block in SysML. Both are structures that encapsulate a textual requirement statement (e.g., "The system SHALL …"). All of the Documentation section entities can be loaded from the outside or generated for use by people or systems.

The Functional Model and Physical Model sections along with the Parametric and Program Model section have entity classes used for building and executing system models. The Action entity, on the left of the ontology diagram in Figure 4, is the primary building block for functional-behavioral models. The Asset entity, on the right, is the primary building block for physical models. Every entity also has a "type" property, which allows many variants of Actions and Assets to be represented. For example, Actions may be assigned a type of Activity, Capability, Event, Function, Process, or Task. Assets may be assigned a type of Component, Entity, Service, Sub-system, or System as needed in a particular modeling context. Assets may represent human actors as well as physical components and software systems.

Another key basic conceptual element in functional models is Input/Output (IO), which represents the flow of information or other resources in or out of an Action, including Item, Trigger, Information, Data, and Energy. The corresponding basic concept in a physical model is the Conduit. At the physical model level, data or other resources represented by a functional model IO entity are transferred from one physical model Asset to another via a Connection or Conduit, which might be implemented as a Data Bus, Interface, or Pipe.

Natural Language Requirements in Traditional and Digital-MBSE Approaches

Figure 5 shows an extracted sample LML requirements document for an SAE Category 5 autonomous self-driving vehicle. The artifact name appears at the top along with ten Statement entities below. Each statement is described by attributes (ID number, name, and description) of each entity along with quality score, and label attributes in the columns on the right side of the table. Eight of the ten Statements are also Requirement entities (all except 1 and 4) and have been labeled by the systems engineer as Functional Requirements.



SAE-Level-5-Automation-Requirements.csv	Quality Score	Labels
1 Execution of Steering and Acceleration/Deceleration	N/A	None to display.
1.1 Steer Vehicle The driver assistance system shall steer the vehicle.	75%	Functional Requirement
1.2 Accelerate Vehicle The driver assistance system shall control acceleration of the vehicle.	75%	Functional Requirement
1.3 Decelerate Vehicle The driver assistance system shall control deceleration of the vehicle.	75%	Functional Requirement
2 Monitoring of Driving Environment The driver assistance system shall monitor the driving environment.	75%	Functional Requirement
3 Fallback Performance of Dynamic Driving Task The driver assistance system shall determine when to change lanes, turn, and use signals.	63%	Functional Requirement
4 System Capability (Driving Modes)	N/A	None to display.
4.1 Expressway Merging The driver assistance system shall be capable of expressway merging.	75%	Functional Requirement
4.2 High Speed Cruising The driver assistance system shall be capable of high speed cruising.	63%	Functional Requirement Performance Requirement
4.3 Low Speed Traffic Jams The driver assistance system shall be capable of low speed traffic jams.	63%	Functional Requirement Performance Requirement

Figure 5. Autonomous Vehicle Originating Requirements Document

The Quality Score is calculated by evaluating the text of each individual requirement against six of the eight standard criteria for a well-formed requirement -- clear, complete, consistent, design, traceable, verifiable. The correctness and feasibility criteria cannot be determined by this means.

The lower Quality Score for Requirement 3 identifies a potential problem: the text expresses a conjunction of three subsidiary requirements (change lanes, turn, and use signals). During requirements analysis, Requirement 3 will need to be broken down into at least three separately testable sub-requirements. These sub-requirements will then need to be added to the model as requirements analysis proceeds. This is an example of how the number of requirements grows during analysis.

Requirements 4.2 and 4.3 are also marked as lower quality because each uses ambiguous language (high speed and low speed). The ambiguous values for high and low speed are examples of TBDs. Resolving the TBDs into specific values or ranges of values require reference to the CONOPS and context of the autonomous vehicle. One set of values would capture the operating speeds of a vehicle to operate on highways and city streets. Alternatively, if the vehicle is intended to operate as a golf car or delivery vehicle in a community where speeds are limited to 20 mph, different values would be appropriate. By introducing Decision entities to the model, the problem can be specified and the methods and rationale for determining the appropriate values recorded for later review.

While the table of requirements in Figure 5 look quite similar to the way they would appear in requirements tools used for document-based systems engineering, a closer look shows that each row is actually a model entity in model database. Since they are entities, relationships describe connections among them. Figure 6 shows the relationships captured automatically when the originating requirements are loaded into Innoslate.

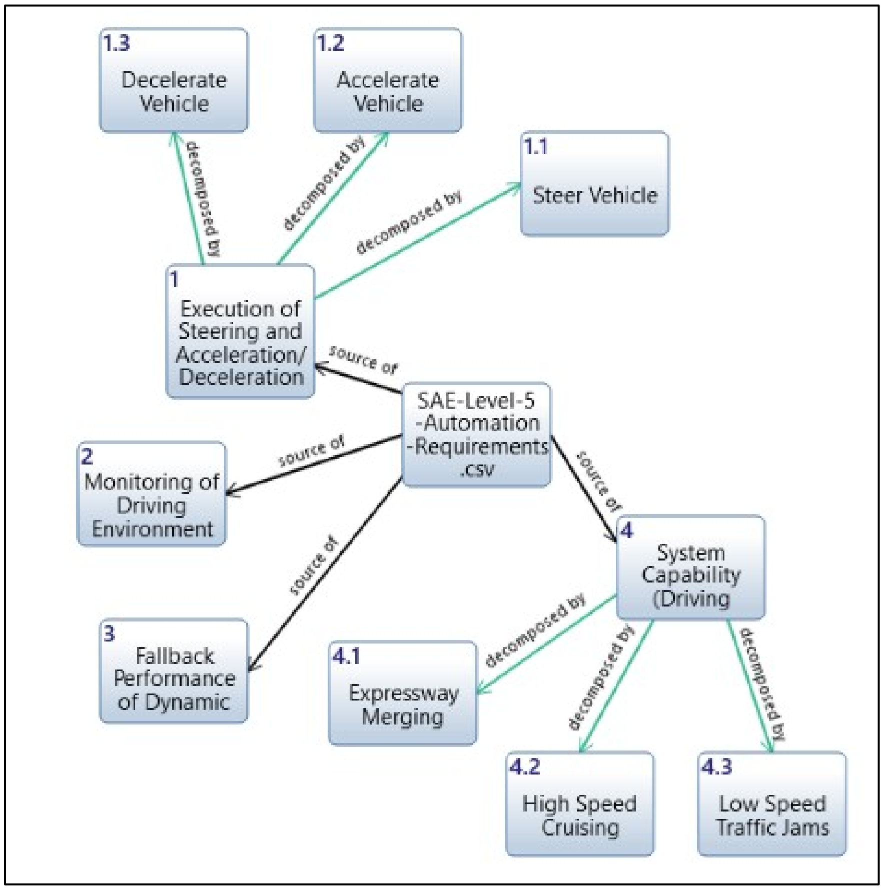


Figure 6. Relationships among Originating Requirements Entities

As mentioned above, there are many standard relationship types that apply to different combinations of entity types. In Figure 6, the four top level requirements are connected by the “source of” relationship to the original artifact. Each of the six subsidiary requirements, (1.1, 1.2, 1.3, 4.1, 4.2, and 4.3) appear as decompositions of the higher level requirement. These relationships mark the start of the development of a systems engineering model for the autonomous vehicle.

As the system model develops, additional relationships will connect model elements together in different ways. When an Asset is allocated to perform a functional model Action, a “performed by/performs” relationship is established. A functional model Input/Output entity may be allocated to a physical model Conduit via the “transferred by/transfers” relationship where the functional flow thereby becomes constrained by the properties of the physical device implementing the Conduit.

TBD and TBR quantities are an important aspect of leading indicators, but could be better integrated into the modeling. As system understanding develops, some information will be less refined than other information. For example, the value for a parameter in a requirement may be unknown (TBD) or estimated (TBR). LML does not have a standard TBD or TBR entity class, but the Decision entity definitions can be extended to create an TBD/TBR subclass that can be attached to the model to represent the both the uncertainty and the process for finding the missing information as well as defining assumptions. When the TBD/TBR is resolved, the updated TBD/TBR entities provide a record of how the value was obtained. By using a specialized TBD/TBR class, the number of TBD/TBR remaining can be readily obtained for use in leading indicators. A burn-down chart for progress on resolution of TBD/TBR entities would also be informative as a leading indicator.

Functional Modeling

Figure 7 shows a top level Action Diagram for the functional model in the autonomous vehicle example. The diagram depicts three parallel functional flows with actions A.1, A.2, and A.3 performed by the User, Autonomous Vehicle, and Environment physical assets respectively. The physical assets here are viewed functionally with physical properties captured elsewhere in the model as appropriate. Since assets User and Environment extend outside the system boundary to the system context rather than the system being engineered, the IO flows that they provide must cross external interfaces at the physical model level. IO flows that do not cross the system boundary capture internal interfaces.

Even though three assets are present, everything in the Action Diagram depicts functional requirements. The two IO entities describe the flow of Destination Location from A.1 to A.2 and Environmental Conditions from A.3 to A.2. As depicted, each of these IO flows is a trigger that enables the A.2 Drive Vehicle operation to start.

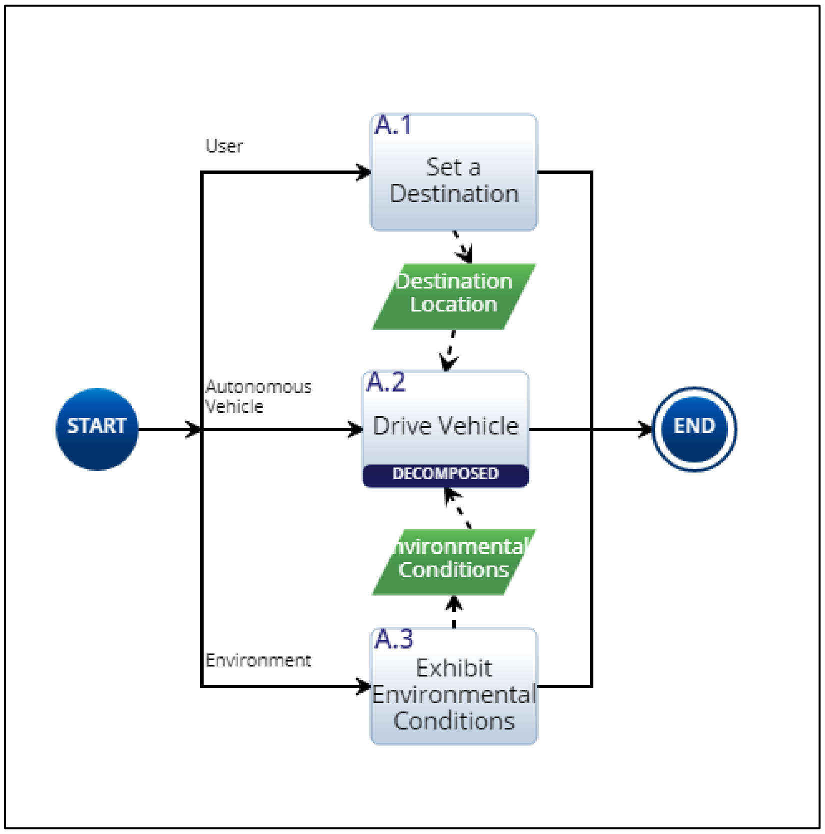


Figure 7. Top Level Action Diagram

The A.2 Drive Vehicle action is decomposed as shown in Figure 8 . The Drive Vehicle function continues in a loop until the destination is reached. During driving, three parallel functional control flows continue. The Sensors asset performs the A2.2 Monitor Environment action producing IOs for Camera Data and Lidar Data. These IOs then flow into the Control System asset, which performs A2.3 Calculate Waypoint and Obstacles and sends them out as IOs. The Drive System asset takes those IOs and performs A2.4 Navigate Vehicle. All these actions occur in parallel.

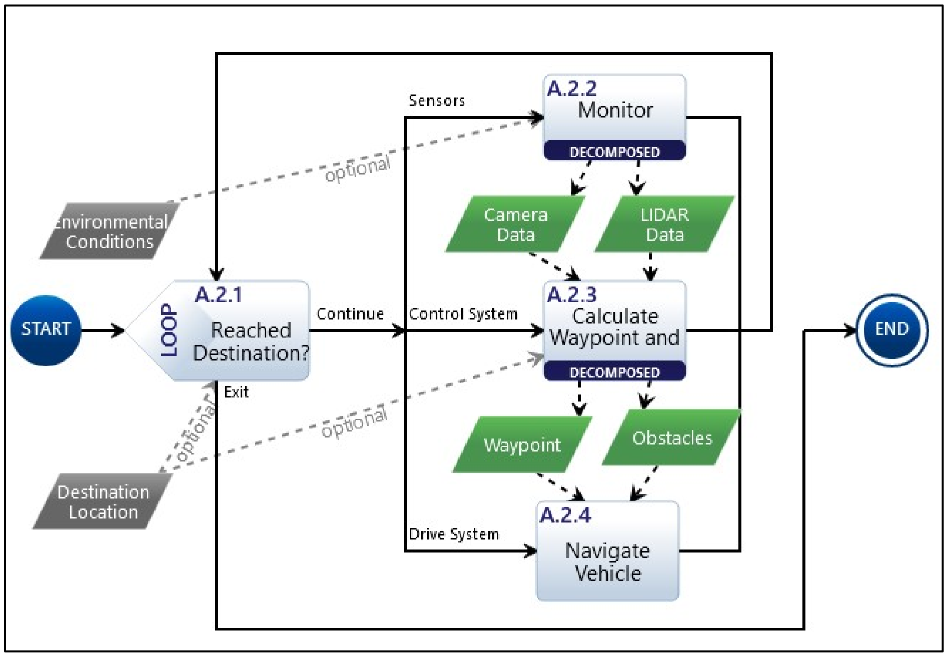


Figure 8. A.2 Drive Vehicle – Decomposition Action Diagram

These two action diagrams capture the first two layers of functional decomposition for the autonomous vehicle system. The third layer of decomposition is shown in the functional hierarchy diagram in Figure 9. Detailed action diagrams are not shown here for the third layer.

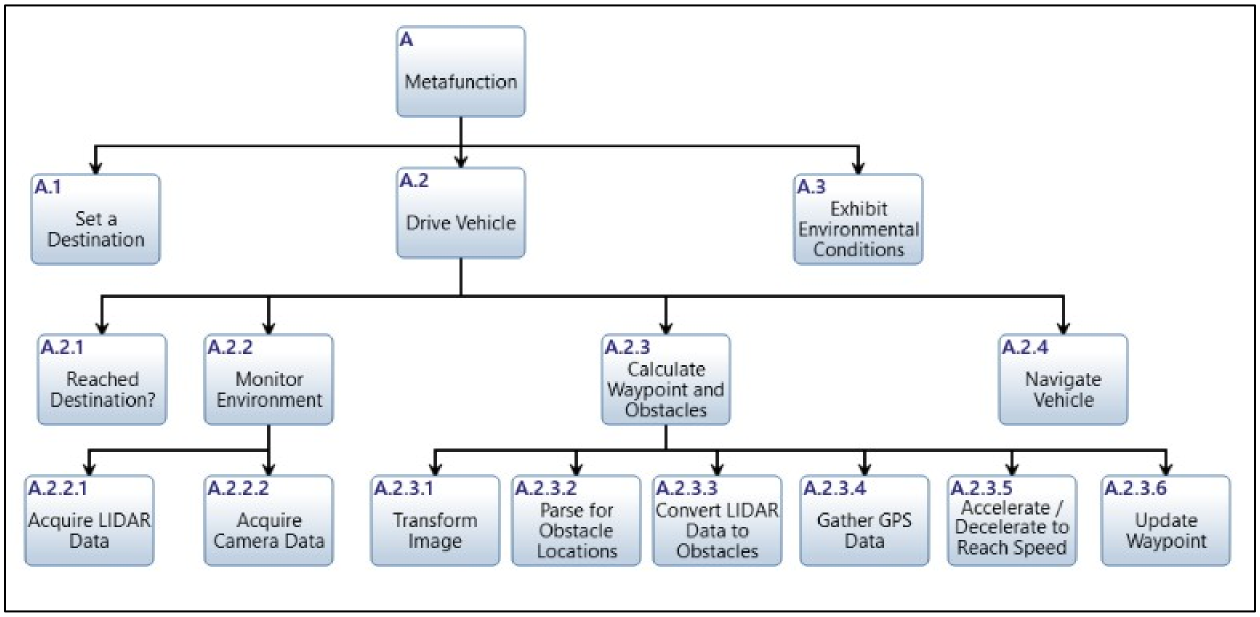
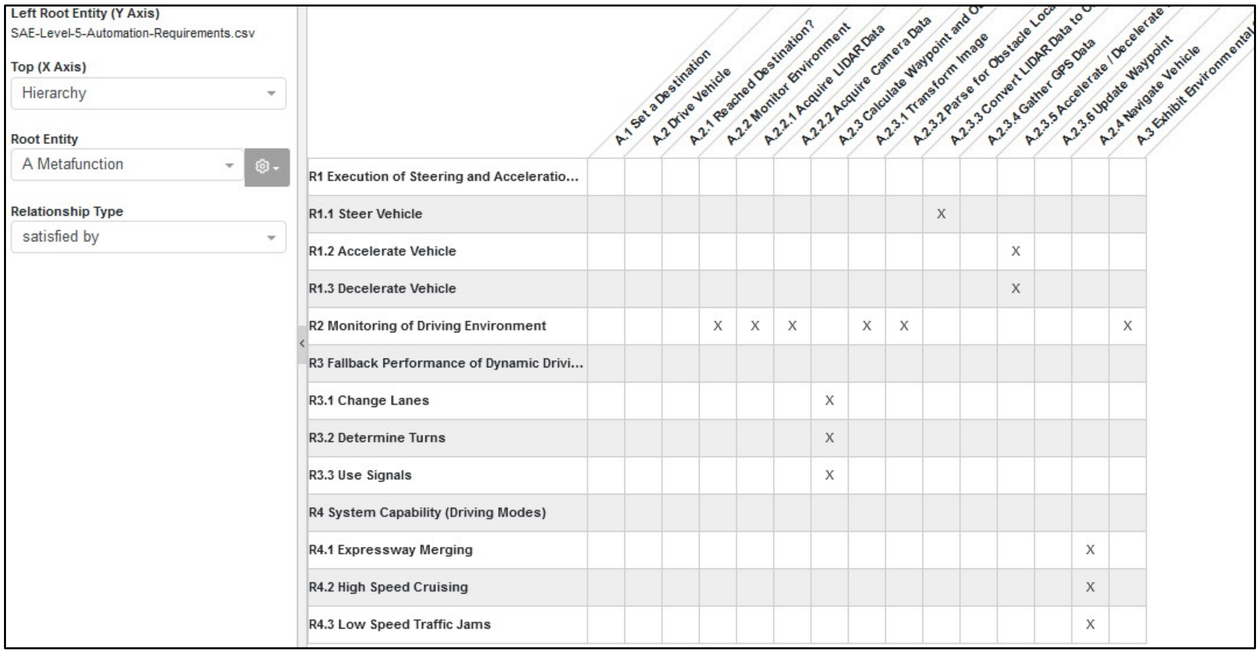


Figure 9. Functional Hierarchy Diagram

As requirements analysis proceeds, the model and the requirements will grow deeper and broader. In traditional practice, requirements are frozen in text and isolated from the models that engineers use for analysis. Whether explicit or implicit, a requirement in MBSE is linked by relationships to other elements of the model thereby giving greater context to understanding the meaning of a requirement. For example, by running simulations on executable models, the engineer can check whether a set of requirements has face validity or meets expectations. Spider charts and hierarchy charts can be used to visualize the structure of the model and the requirements.

Figure 10 shows a matrix of requirements (vertical) against functions (horizontal). Notice that at this point in the analysis, there are no requirements satisfied by functions A.1 (Set a Destination) or A2.1 (Reached Destination). In other words, the Destination does not appear in the Originating Requirements at all. Since it seems logical for a vehicle being driven to have a destination, more investigation and refinement will be needed. One possible explanation is that the actions related to the destination came from a concept of operations (CONOPS) that is not shown. A similar flaw in the model is the lack of a requirement for A2.3.6 (Update Waypoint). From a leading indicator point of view, functions that have no requirement are open issues that will need more work. Mismatch problems of this sort can be surfaced by automated analysis of relationships in the model.


Left Root Entity (Y Axis)
SAE-Level-5-Automation-Requirements.csv

Top (X Axis)
Hierarchy

Root Entity
A Metafunction

Relationship Type
satisfied by

	A.1 Set a Destination	A.2 Drive Vehicle	A.2.1 Reached Destination?	A.2.2 Monitor Environment	A.2.2.1 Acquire LIDAR Data	A.2.2.2 Acquire Camera Data	A.2.3 Calculate Waypoint and O	A.2.3.1 Transform Image	A.2.3.2 Parse for Obstacle Loca	A.2.3.3 Convert LIDAR Data to O	A.2.3.4 Gather GPS Data	A.2.3.5 Accelerate / Decelerate	A.2.3.6 Update Waypoint	A.2.4 Navigate Vehicle	A.3 Exhibit Environmental
R1 Execution of Steering and Acceleratio...															
R1.1 Steer Vehicle										X					
R1.2 Accelerate Vehicle												X			
R1.3 Decelerate Vehicle												X			
R2 Monitoring of Driving Environment				X	X	X		X	X						X
R3 Fallback Performance of Dynamic Drivi...															
R3.1 Change Lanes							X								
R3.2 Determine Turns							X								
R3.3 Use Signals							X								
R4 System Capability (Driving Modes)															
R4.1 Expressway Merging														X	
R4.2 High Speed Cruising														X	
R4.3 Low Speed Traffic Jams														X	

Figure 10. Matrix of Requirements (vertical) Satisfied by Functions (horizontal)

Physical Architecture Modeling

The functional model includes depictions of physical model Assets as performers of functions and connections across those Assets. An Asset entity specifies an object, person, or organization (such as a system, subsystem, component, or element) that is used to create value and perform Actions. Examples include: Infrared Sensor, Accounting Department, Internal Revenue Service. Standard “type” nomenclature for Assets also includes: Architecture, Assembly, Component, Context, Element, Environment, External System, Facility, Hardware, Human, HW Element, HWCI, Infrastructure, LRU, Materiel, Operational Element, Organization, Part, Performer, Personnel, Segment, Service, Software, Subassembly, Subsystem, System, System Instantiation, Test Equipment, Test Software, Unit. The list of “types” is extensible as needed by the firm or project.

Figure 11 shows the hierarchy of Assets mentioned in the functional model through four levels of decomposition.

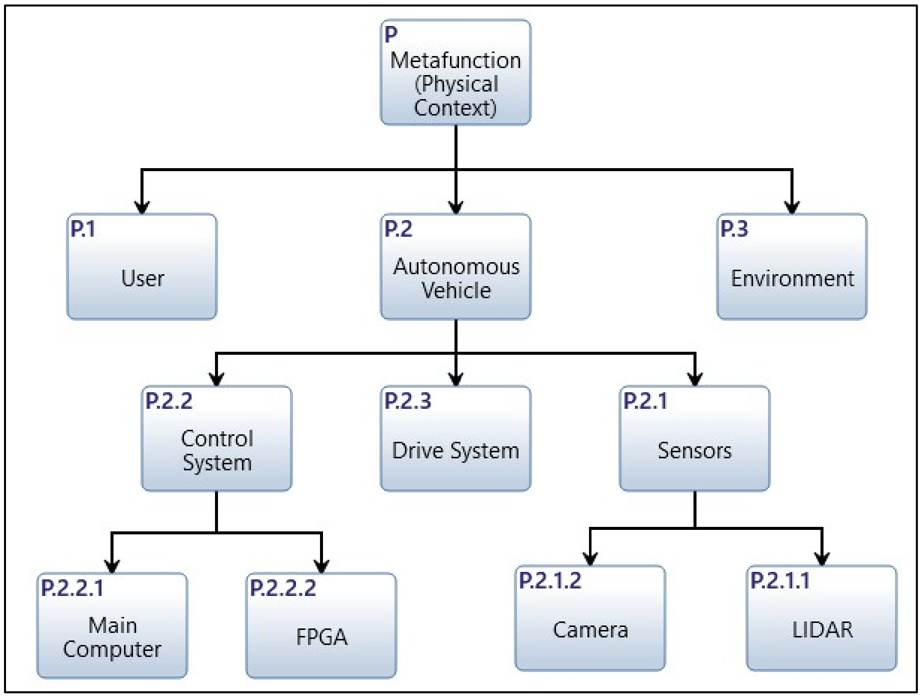


Figure 11. Physical Model Asset Hierarchy

At the lowest level, the P2.1 Sensors are broken down into P2.1.2 Camera and P2.1.1 LIDAR. Similarly, P2.2 Control System is decomposed into P2.2.1 Main Computer and P2.2.2 FPGA (field programmable gate array device). The Conduits that provide pathways for data to move between the physical model elements at level 4 are shown in Figure 12. LIDAR data and Camera data from sensors flow through Conduits into P2.2 Control System, which generates Waypoints and Obstacles to send through other Conduits to P2.3 Drive System where driving actions are decided.

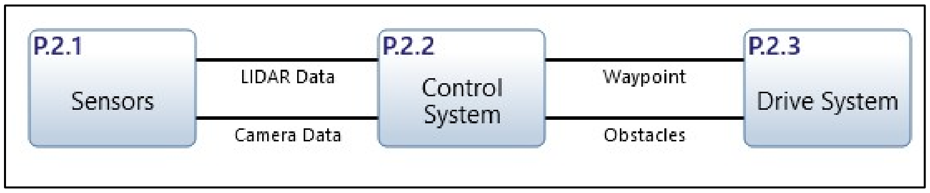


Figure 12. Example of IO flows through Conduits

Implicit requirements are inferred from the Functional and Physical Models developed by engineers during requirements analysis. Functional Requirements may be defined by Action entities and the flows, relationships, and properties that describe them.

Innoslate also has a tool that converts Actions in a functional model into implied Assets and Conduits in a Physical Model. Interface Requirements can be inferred from Conduit entities describe connections across Assets supporting transfer of IO entities in the physical model. The technical characteristics of the endpoint Assets and the Conduit combine to specify the interface requirements. Performance Requirements often come from data related to Asset entities and connections. External interfaces are be represented by Conduits that connect Assets within the system to Assets outside the system boundary in the system context.

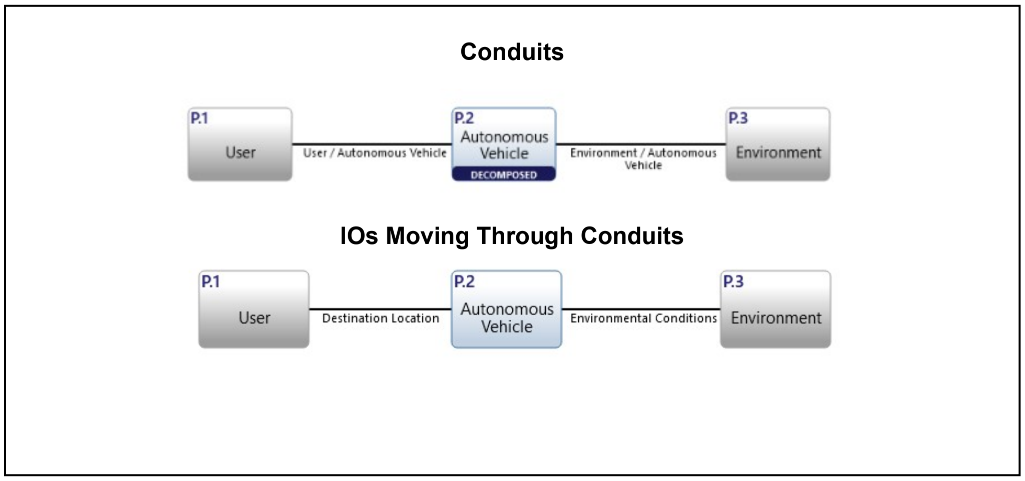


Figure 13. Conduits in First Level of Physical Decomposition

Figure 13 shows two views of the conduits connecting the User (P1) and the Environment (P3) to the Autonomous Vehicle (P2). Both the User and the Environment are shown with shading indicating that these Assets are outside of the system boundary. The top view in the figure shows conduits. The lower view shows the IO entities flowing asynchronously through each conduit. A connection between an Asset and a Conduit will be governed by an interface. These interfaces are external, since the Asset one side of the Conduit is outside the system boundary. The connections in Figure 12, on the other hand, are between Assets inside the system boundary and will be internal interfaces. The specifications of each interface will be derived from the properties of conduit itself and of the assets on either end.

Validation and Verification (V&V)

For V&V, the relationships among model elements are used to trace the derivation of the model. As mentioned above, the diagrams tell only part of the story with relationships filling in additional detail. Figure 14 is a “spider diagram” showing how some of the model elements relate to the Drive Vehicle function. Similar diagrams can be used to review other connections across elements of the model, which can be helpful in establishing model validity.

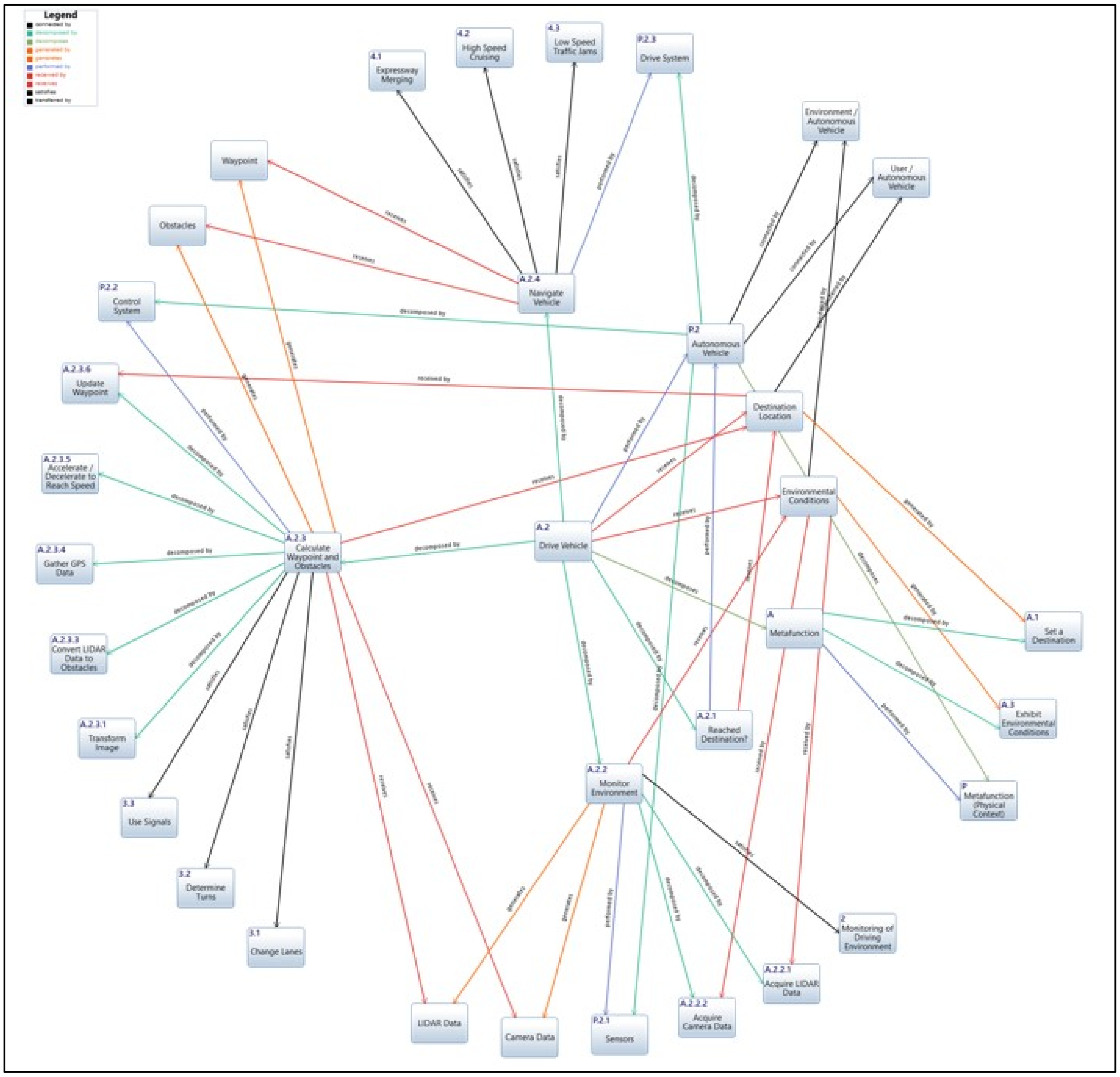
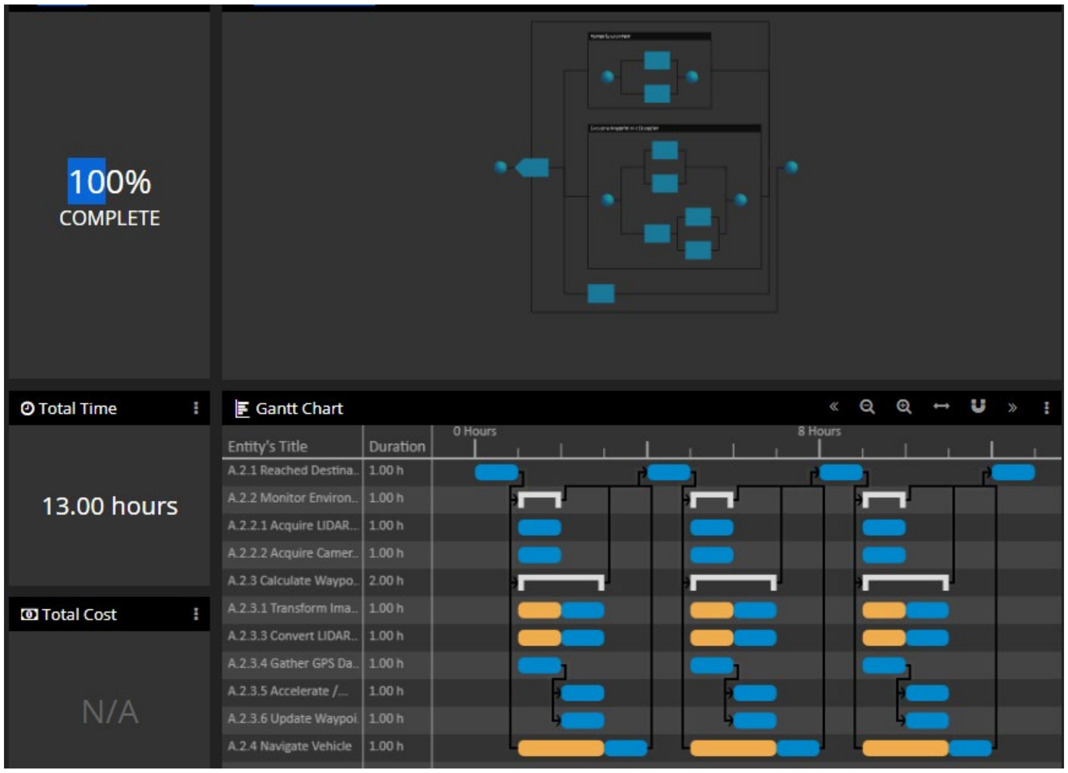


Figure 14. Functional and Physical Relationships for Drive Vehicle

Since these models are executable, another form of validation is to run discrete event and other simulations and inspect the results to see if expectations of system behavior appear to be met. Figure 15 shows the output of a discrete event simulation run of the sample model. As performance characteristics of model elements are refined, the timing results will change. As the fidelity of the model improves, additional Monte Carlo and other simulations can be used to explore optimization of models.

Model diagrams, as well as spider and hierarchy diagrams and model simulators can be incorporated into dashboards for interactive exploration of the model and its implications.



Entity's Title	Duration
A.2.1 Reached Destina...	1.00 h
A.2.2 Monitor Environ...	1.00 h
A.2.2.1 Acquire LIDAR...	1.00 h
A.2.2.2 Acquire Camer...	1.00 h
A.2.3 Calculate Waypo...	2.00 h
A.2.3.1 Transform Ima...	1.00 h
A.2.3.3 Convert LIDAR...	1.00 h
A.2.3.4 Gather GPS Da...	1.00 h
A.2.3.5 Accelerate /...	1.00 h
A.2.3.6 Update Waypoi.	1.00 h
A.2.4 Navigate Vehicle	1.00 h

Figure 15. Sample Validation with Discrete Event Simulation

Generating Leading Indicators from System Model

For this case, we examine four leading indicators that relate to aspects of requirements management. These four were found to be the subset of the eighteen leading indicators that were most likely to be implemented with direct use of model-based toolsets based on availability of the base measure information in the system model (Rhodes, et al. 2021). The following is a brief discussion on how these are generated from the system model.

In the current state of practice, requirements are typically collected and stored in a specialized requirements database, often using software (e.g., DOORS® or other similar packages suited to needs of the project/enterprise). These types of packages are generally interoperable with and/or loosely coupled to other systems engineering model-based toolsets. It is the assumption of this research team that the specific details of this will vary based on the chosen model-based tools used.

Tracking the trends needed for the leading indicators requires taking snapshots of metrics values at intervals over time as the program proceeds. If integrated into an LML meta-model, this program management data would be stored as baselines or other objects in the database to facilitate integrative analysis with other program data.

Requirements Trends and Interface Trends

The metrics required for *Requirements Trends* and *Interface Trends* can be composed by counting explicit and implicit requirements identified in the Innoslate database. Explicit requirements are found in Requirements entities that contain natural language statements, which are (1) sourced from documents loaded into the system; (2) entered directly into the database by engineers; or (3) generated from other data and stored in the database.

Implicit requirements are derived from the functional and physical models developed by engineers during requirements analysis. Functional Requirements may be defined by Action entities and the flows, relationships, and properties that describe them. Innoslate also has a tool that converts Actions in a functional model into implied Assets and Conduits in a physical model.

Interface Requirements can be inferred from Conduit entities used for transferring IO entities between Assets in the physical model. The technical characteristics of the endpoint Assets and the Conduit combine to specify the interface requirements. Performance Requirements often come from analysis of data related to Asset entities and connections. External interfaces would be represented by connecting a Conduit to an

Asset that is outside the system boundary, as the User and Environment assets as in Figure 13.

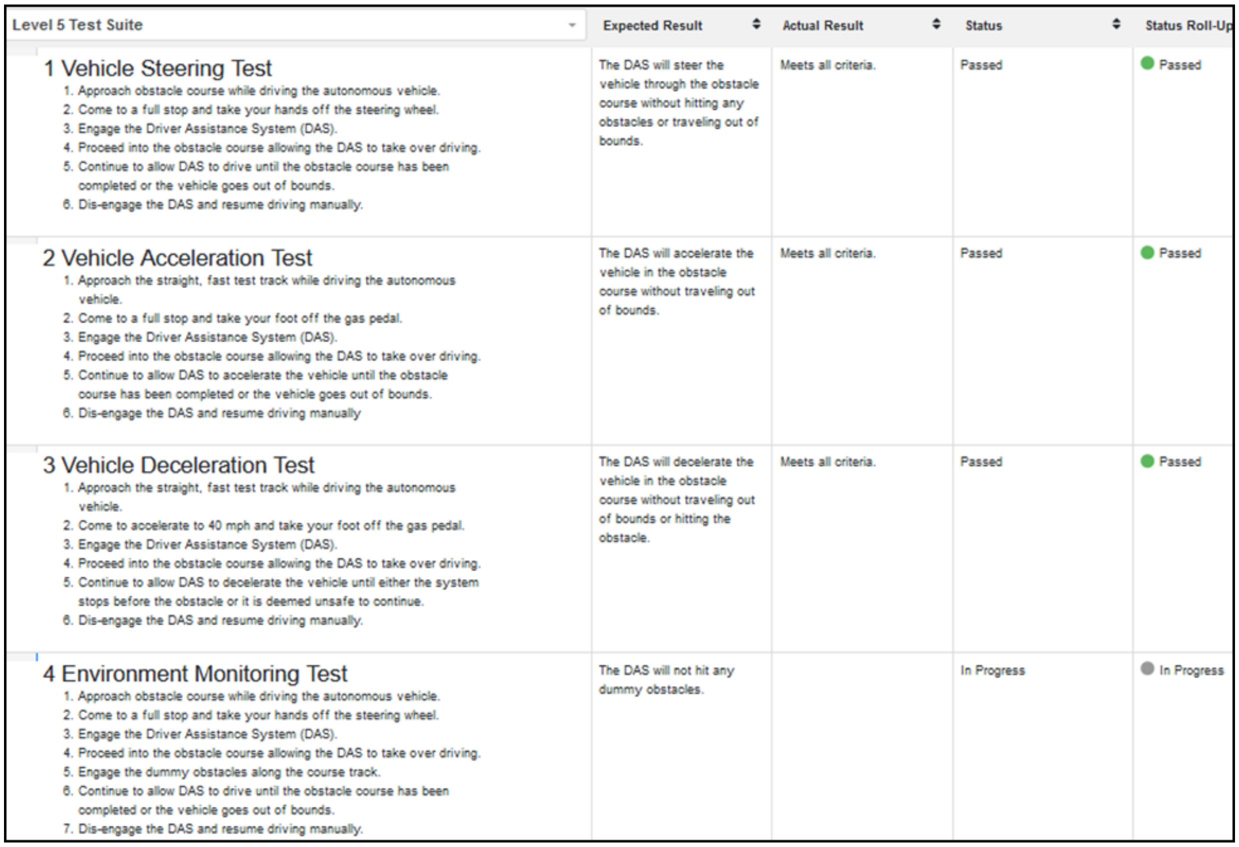
As requirements analysis progresses, the model and the requirements will grow deeper and broader. In traditional practice, requirements are frozen in text and isolated from the models that engineers use for analysis. Whether explicit or implicit, a requirement in MBSE is linked by relationships to other elements of the model giving greater context to understanding the meaning of a requirement. For example, by running simulations on executable models, the engineer can identify whether a set of requirements has face validity or meets expectations. Spider charts and hierarchy charts can be used to visualize the structure of the model and the requirements.

As systems understanding develops, some information will be less refined than other information. For example, the value for a parameter in a requirement may be unknown (TBD) or estimated (TBR). LML Decision entities can be attached to the model to represent the both the uncertainty and the process for finding the missing information as well as defining assumptions. When the TBD/TBR is resolved, the updated Decision entities provide a record of how the value was obtained. A burn-down chart for progress on resolution of Decisions would also be informative as a leading indicator.

Requirements Validation Trends and Requirements Verification Trends

Systems engineering best practice recommends initiating requirements validation and verification early in the project as requirements are found and entered into the database. At the early stage, Innoslate and some other toolsets offer a natural language tool for checking the quality of requirements statements against six of the eight standard criteria (clear, complete, consistent, design, traceable, verifiable but not correct and feasible). Another tool applies heuristics to evaluate models and requirements in more depth. A roll-up of these quality metrics could provide leaders with early insight on how well the requirements are progressing and whether problems are being left to later in the life cycle where they will be more difficult to resolve.

Innoslate also includes a Test Center where test plans and scenarios can be built for early or later use and VCRM Reports generated. Figure 16 shows a snapshot of part of a test plan and results with verification status. The leading indicators for requirements verification and requirements validation could be improved by adding measures for progress on developing test plans to complement the metric for successful completion of validation and verification testing. Product validation and verification also needs to be considered holistically as well as individually by requirement. The model can be used with simulation tools to predict the behavior of the whole system or subsystems.



Level 5 Test Suite	Expected Result	Actual Result	Status	Status Roll-Up
1 Vehicle Steering Test 1. Approach obstacle course while driving the autonomous vehicle. 2. Come to a full stop and take your hands off the steering wheel. 3. Engage the Driver Assistance System (DAS). 4. Proceed into the obstacle course allowing the DAS to take over driving. 5. Continue to allow DAS to drive until the obstacle course has been completed or the vehicle goes out of bounds. 6. Dis-engage the DAS and resume driving manually.	The DAS will steer the vehicle through the obstacle course without hitting any obstacles or traveling out of bounds.	Meets all criteria.	Passed	● Passed
2 Vehicle Acceleration Test 1. Approach the straight, fast test track while driving the autonomous vehicle. 2. Come to a full stop and take your foot off the gas pedal. 3. Engage the Driver Assistance System (DAS). 4. Proceed into the obstacle course allowing the DAS to take over driving. 5. Continue to allow DAS to accelerate the vehicle until the obstacle course has been completed or the vehicle goes out of bounds. 6. Dis-engage the DAS and resume driving manually	The DAS will accelerate the vehicle in the obstacle course without traveling out of bounds.	Meets all criteria.	Passed	● Passed
3 Vehicle Deceleration Test 1. Approach the straight, fast test track while driving the autonomous vehicle. 2. Come to accelerate to 40 mph and take your foot off the gas pedal. 3. Engage the Driver Assistance System (DAS). 4. Proceed into the obstacle course allowing the DAS to take over driving. 5. Continue to allow DAS to decelerate the vehicle until either the system stops before the obstacle or it is deemed unsafe to continue. 6. Dis-engage the DAS and resume driving manually.	The DAS will decelerate the vehicle in the obstacle course without traveling out of bounds or hitting the obstacle.	Meets all criteria.	Passed	● Passed
4 Environment Monitoring Test 1. Approach obstacle course while driving the autonomous vehicle. 2. Come to a full stop and take your hands off the steering wheel. 3. Engage the Driver Assistance System (DAS). 4. Proceed into the obstacle course allowing the DAS to take over driving. 5. Engage the dummy obstacles along the course track. 6. Continue to allow DAS to drive until the obstacle course has been completed or the vehicle goes out of bounds. 7. Dis-engage the DAS and resume driving manually.	The DAS will not hit any dummy obstacles.		In Progress	● In Progress

Figure 16. V&V Sample Level 5 Test Suite

Case Summary

The illustrative case provides some concrete examples of how model-based systems engineering differs from traditional document-based approaches. The examples are drawn from a small, but representative, case of new development of a self-driving autonomous vehicle complying with SAE standards. The case includes highlights of the LML systems engineering ontology. The cloud-based Innoslate toolset was used for developing examples. Innoslate implements the LML ontology and has methods for translating content to other languages, such as SysML and DODAF. The SAE Level 5 Automation Requirements are used as a representative example of natural language text-based requirements. While the requirements loaded into MBSE look similar to traditional requirements, it was found that the requirements statements are encapsulated in computer data entities, but still require a human to interpret the meaning. The only exception found is the automated connection of LML requirements entities to other elements of the model through a network of relationships.

Examples of TBD/TBD incomplete information are identified and approaches to extending the LML ontology for formally representing these important aspects of the systems engineering process, which are critical indicators of the state of maturity of the system definition. Some representative examples of functional decomposition illustrate a visual approach to functional requirements specification. The examples show how functional flows are captured. Relationship connections between model elements provide additional depth and texture to the model and can assist in evaluating the maturity of the model, which has implications for leading indicators. Some examples of physical system modeling of information flows (or other resource flows) provide a small window into system architecture and specification. Conduits used to model these flows provide the basis for identifying and specifying interfaces. Examples of the close relationship between functional models and physical system models are illustrated.

Some approaches for using the system model to generate information for Requirements Trends and Interface Trends leading indicators are explored. An example of discrete event simulation directly from the model shows how executable models can be used for evaluating system requirements against user expectations. Examples are shown of early generation of Test Plans for Requirements Validation and Requirements Verification with links back to model elements. Assuring that all requirements have an executable test plan can serve as a leading indicator of project success.

THIS PAGE LEFT INTENTIONALLY BLANK

Discussion

Further development of leading indicators would require a community effort extending from implications identified in this research, insights from practitioners, and results of other ongoing measurement investigations and initiatives. Building on this research, a possible organizing approach is to use categories related to level of effort required. Knowledge-based practice appears to be a very effective method to use to ensure that leading indicators provide information decision makers need for performing digital engineering. In addition to transitioning to model-based systems engineering, programs are also implementing new approaches such as Agile and DevOps. Any future work on systems engineering leading indicators needs to take a holistic view of approaches and technology.

Three Categories for Required Effort

Section 3 discusses the model-based implications for the existing leading indicators. Looking toward the possibility of developing a new version of the most recent Systems Engineering Leading Indicators Guide (Roedler et al., 2010) in context of digital engineering, we propose taking the perspective of three categories that will require different levels of effort (Table 5).

Table 5. Three categories for approaching the adaptation and extension of leading indicators.

Category	*Digital engineering impact on leading indicator*	*Effort required*
Cat 1	Digital engineering has minimal impact on the leading indicator measurement specification	*Additional Information* section of measurement specification augmented with descriptive information
Cat 2	Digital engineering results in significant changes/additions to the measurement specification	Modify and add information to all relevant areas of the measurement specification
Cat 3	Digital engineering results in identifying and characterizing novel leading indicators, requiring new measurement specifications	Generate new measurement specification and illustrative graphics of displayed information

Category 1 is defined as a leading indicator that requires minimal change under digital engineering. Accordingly, the *Additional Information* sections of the measurement specification could be augmented with an approach similar to what was done in the prior work on extending leading indicators for HSI (Gerst & Rhodes, 2010). An example of a category 1 leading indicator is Staff and Skill Trends Leading Indicator, where some additions to the additional information section of the specification can be made to describe the any new aspects of staffing and skills that would be required on a program. The attributes of staffing and skills will be impacted by digital engineering, however the measurement specification itself may not require extensive change. More extensive changes to this leading indicator specification could be necessary if digital engineering practice evolves to full use of model-based environments for planning and tracking, in which case this leading indicators would move to the second category.

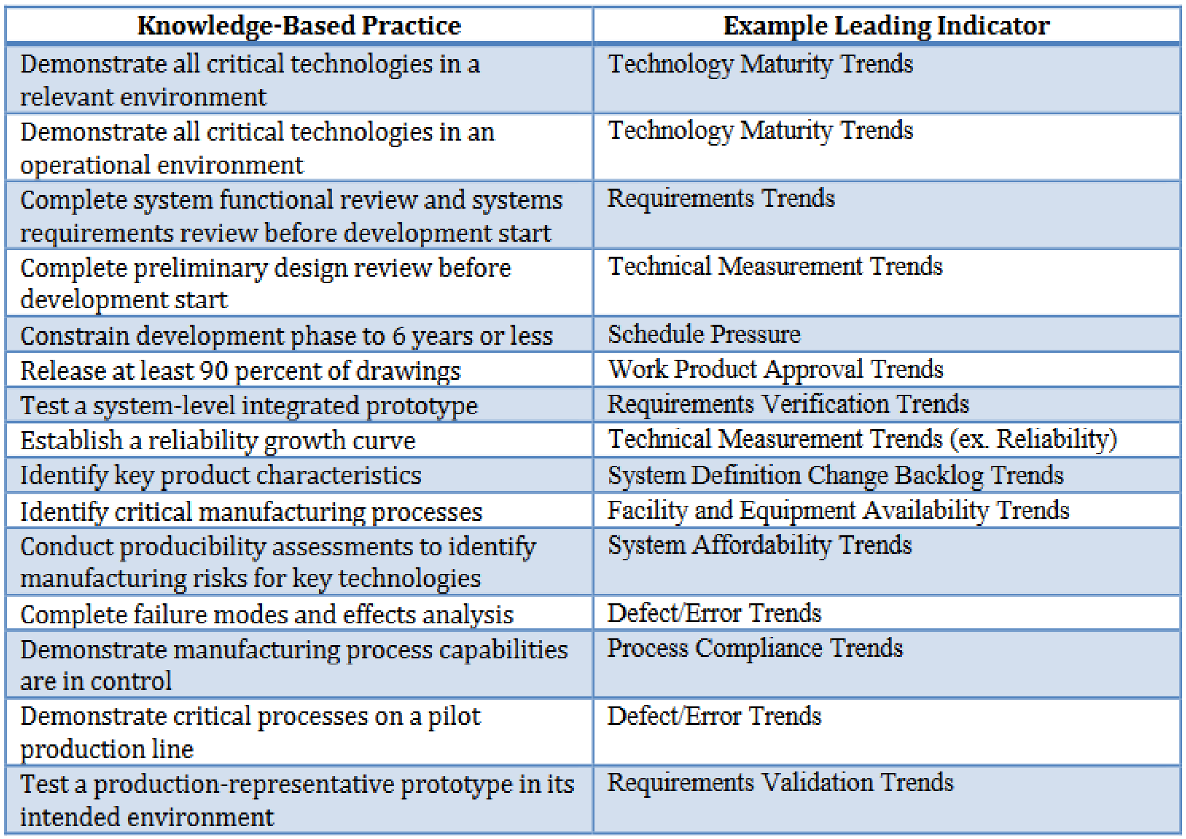
Category 2 is defined as the case where digital engineering necessitates significant modifications and/or additions to the leading indicator measurement specification content. Accordingly, there is a need to modify and/or add to numerous relevant sections in the specification. An example of a potential leading indicator is Work Product Approval Trends. It is expected that a model-centric program, will involve new interim digital work products, and approvals may happen on a different time frame than in traditional engineering programs. As a result, there would be multiple sections in the specification that are revised and augmented.

Category 3 is defined as novel leading indicators that do not currently exist that are made possible with model-based methods and tools, and newer approaches (e.g., Agile). While not the central focus of this current phase of the research, discussions with stakeholders have confirmed the need for some novel leading indicators. Some of the existing leading indicators are suggested to have model-based equivalents. Similar to existing leading indicators related to requirements trends, there may be indicators for model trends (e.g., requirements volatility and model volatility). Just as requirements volatility can be used to judge readiness based on an acceptable change level in requirements respective to proceeding to a next phase of system development (e.g., typically a decision made at a milestone review), model volatility would also enter into such a decision. Just like requirements change, model information will be added, modified and deleted; trend information would enable judging the acceptable change level in the model over time.

Knowledge-Based Approach

Recent work on use of knowledge-based practice is identified as a promising area to consider in further research. Orlowski (2017) proposes Knowledge-based Measurement as an effective method for selecting indictors on a project. A mapping of systems engineering leading indicators to knowledge-based practice is shown in Table 6. A similar approach could be used respective to digital engineering, with existing and new leading indicators mapped accordingly.

Table 6. Knowledge-based Leading Indicators from Orlowski, 2017, p.58.



Knowledge-Based Practice	Example Leading Indicator
Demonstrate all critical technologies in a relevant environment	Technology Maturity Trends
Demonstrate all critical technologies in an operational environment	Technology Maturity Trends
Complete system functional review and systems requirements review before development start	Requirements Trends
Complete preliminary design review before development start	Technical Measurement Trends
Constrain development phase to 6 years or less	Schedule Pressure
Release at least 90 percent of drawings	Work Product Approval Trends
Test a system-level integrated prototype	Requirements Verification Trends
Establish a reliability growth curve	Technical Measurement Trends (ex. Reliability)
Identify key product characteristics	System Definition Change Backlog Trends
Identify critical manufacturing processes	Facility and Equipment Availability Trends
Conduct producibility assessments to identify manufacturing risks for key technologies	System Affordability Trends
Complete failure modes and effects analysis	Defect/Error Trends
Demonstrate manufacturing process capabilities are in control	Process Compliance Trends
Demonstrate critical processes on a pilot production line	Defect/Error Trends
Test a production-representative prototype in its intended environment	Requirements Validation Trends

The methodology proposed by Zheng et al. (2019) aims to integrate systems engineering leading indicators with processes of the PMBoK knowledge areas in order to adapt these for project performance measurement. They propose a five step method to select, specify, identify, tailor, and apply as shown in Figure 17. This method could be similarly applied to leading indicators, given that knowledge areas of digital engineering are defined.

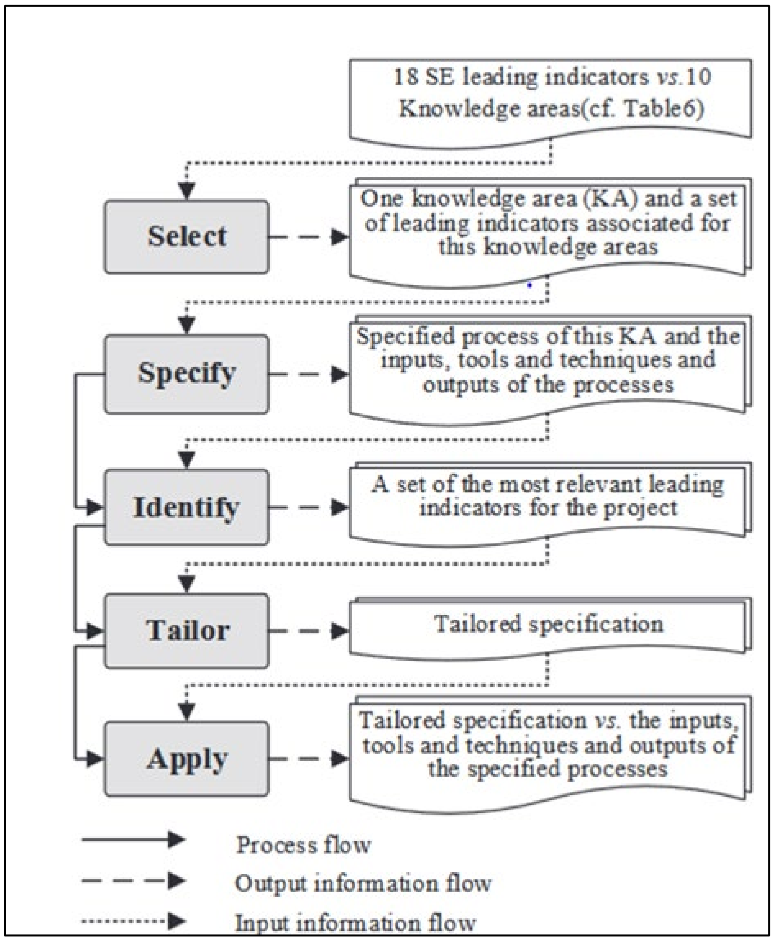

18 SE leading indicators vs. 10 Knowledge areas(cf. Table6)
Select
One knowledge area (KA) and a set of leading indicators associated for this knowledge areas
Specify
Specified process of this KA and the inputs, tools and techniques and outputs of the processes
Identify
A set of the most relevant leading indicators for the project
Tailor
Tailored specification
Apply
Tailored specification vs. the inputs, tools and techniques and outputs of the specified processes
Process flow
Output information flow
Input information flow

Figure 17. Proposed Methodology (Zheng et al., 2019)

Influence of Newer Development Approaches on Measurement

It is likely that many of the leading indicators will need to be re-examined as systems and software engineering adopt newer development approaches such as Agile, Scaled Agile (SAFe), DevOps, and DevSecOps. An ongoing collaboration of PSM, NDIA and INCOSE has developed a measurement framework for Continuous Iterative Development (CID). Jones et al. (2021) provide the following definition of CID: *A method of managing development, testing, and release of software, or systems, to continually, or iteratively, provide working functional systems of increasing capability to internal and external customer*. The current and ongoing work of this collaborative group includes the

PSM CID measurement framework and measurements specifications[3]. While the present CID work has more of a software focus, it is highly relevant for systems engineering. Further, it addresses team, product and enterprise level measures. Additionally, the CID collaboration team is defining and developing measurement specifications for novel measures. One of these is Team Velocity, defined as: *Velocity is a measure of team performance and the amount of work that is completed in an iteration, typically a count of completed story points or equivalent. Velocity calculations can be used to estimate the amount of work that can be accomplished by the team in future iterations and when planned deliveries will be completed*. (Jones et al., 2021).

Research Limitations

There are four key research limitations. First, the focus of the leading indicators effort has been defense acquisition programs; future investigation could benefit from expanding the focus to commercial programs. Second, the scope of this investigation allowed only limited experimentation with illustrative cases. Future efforts should include other cases, other ontologies and other toolsets in order to study variations to expand the model-based implications. As suggested in Vaneman (2018), a broad high-level ontology for systems engineering would provide a framework for locating research results, toolsets, and practices on a map of the field and identify gaps and overlaps in leading indicators. Third, the pandemic situation has had impact on the planned approach. Expert knowledge was gathered though available workshops and from prior leading indicator project participants in the early phases. The limitations imposed by the COVID-19 pandemic, especially on workshops and conference events other than virtual, resulted in reduced opportunities for engaging the community of interest. Planned group discussions were replaced with individual and small group interviews and discussions, which resulted in reduced interaction and feedback opportunities. Fourth, this research was proposed and initiated prior to several ongoing collaborative efforts on measurement in the systems and software community. Practical limitations allowed for limited discussions with these other teams, but collaborative investigation was not feasible.

[3] For the most recent work, see the PSM website https://www.psmsc.com/CIDMeasurement.asp

Future Research

Five areas for future research are recommended. First, as systems become increasingly complex and interconnected, leading indicators are essential to decision-making. The prior work on leading indicators in regard to information needs should be re-visited in context of digital engineering. An area of future research is to investigate specific approaches that have been identified that could be useful. The approach of using a knowledge-based approach has been proposed by recent researchers (Orlowski, 2017; Zheng, et al., 2019) and is a promising area for additional work in context of digital engineering leading indicators. Implementing direct and partial use of model-based toolsets to generate leading indicators is dependent on the detailed implementation on any individual program. Lacking specific research on this, recent work on how artifacts are produced from system models provides useful information. An excellent example of this is the work of Parrott and Weiland (2017) on using MBSE to provide artifacts for NASA project life-cycle and technical reviews[4].

Second, leading indicators require trend information; this has been a challenge over the past decade. Model-centric programs will greatly enhance the ease of collection and storage of the necessary data. Approaches for measurement data collection to better enable trend information requires investigation. Data science and visualization technologies will enhance the interaction with measurement information. Further, this information can be more effectively displayed for human consumption. Automation and augmented intelligence can be used in generating and interpreting leading indicator trend information.

The third area of recommended future research is to further elicit ideas from the systems community for program-level indicators and enterprise-level indicators. Desirable research is to conduct industry case studies to learn from digital engineering early adopters concerning what metrics and leading indicators they have implemented, as well as novel approaches that have been developed. Additionally, impacts of specific

[4] This paper prepared for the Space Forum sponsored by AIAA in 2017 has been issued as NASA/TM-2017-219575

implementation of new approaches such as Agile need to be investigated in regard to issues and opportunities.

Fourth, in this research some limited experimentation with extraction of metric data directly from MBSE toolsets was performed with a single systems engineering toolset (selection of toolset was based on ease of use and availability to research team). A recommended area of future research is to more fully investigate extraction of base measures and generation of leading indicators across the available model-based toolsets. For example, hooks for data collection can be built into management process meta-models integrated into and exploiting tools used for the system models.

Fifth, as new measurement efforts are ongoing within the systems and software communities (e.g., PSM CID, DoD Digital Engineering Metrics), any future effort needs to involve active engagement. In considering the future of the systems engineering leading indicators there is a need to harmonize or combine the work of the various collaborative efforts. A strategy needs to be determined; various options include developing a coordinated set of measurement guidance, developing an integration of all measurement guidance, or developing independent but harmonized measurement guidance.

THIS PAGE LEFT INTENTIONALLY BLANK

Conclusion

Existing systems engineering leading indicators were developed under the assumption of paper-based (traditional) systems engineering practice. This research investigates the model-based implications relevant to the existing leading indicators, aiming to support program leaders transitioning to model-based engineering on their programs that wish to continue to use these indicators. The study elicited knowledge from subject matter experts and performed literature review. Adaption of the leading indicators will vary based on the model-based approaches and toolsets used. An illustrative case was used to assess how four leading indicators could be generated directly from a model-based toolset. Findings have been synthesized as model-based implications for current leading indicators, informing how these can be adapted for under model-based engineering.

Digital engineering transformation continues to drive a need to re-examine how engineering effectiveness is measured and assessed, as well as how to leverage new technologies for generating, analyzing and displaying measurement information and for interpreting and making decisions. This technical report summarizes the work conducted by Massachusetts Institute of Technology under contract award HQ0034-19-1-0002 during the performance period July 22, 2019 – August 31, 2021. A companion research study ("phase 2") under contract HQ0034-20-1-0008 provides insights for the art of the possible for future systems engineering leading indicators and their use in decision-making on model-centric programs. This technical report aims to provide insights for current practice within programs transforming to digital engineering, for continued use of systems engineering leading indicators.

THIS PAGE LEFT INTENTIONALLY BLANK

References

Bone, M. A., Blackburn, M. R., Rhodes, D. H., Cohen, D. N., & Guerrero, J. A. (2019). Transforming systems engineering through digital engineering. *The Journal of Defense Modeling and Simulation*, 16(4), 339-355.

Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. *ACM transactions on database systems* (TODS), 1(1), 9-36.

DoD. (2018, June). *Digital Engineering Strategy,* Office of the Deputy Assistant Secretary of Defense for Systems Engineering.

Dam, S. (2019). *Real MBSE: Model-Based Systems Engineering (MBSE) using LML and Innoslate*.

Elm, J. & Goldenson, D. (2013). Quantifying the effectiveness of systems engineering. *IEEE Systems Conference (SysCon)* (pp. 6-13).

Elm, J., Goldenson, D., El Eman, K., Donatelli, N., & Neisa, A. (2008). *A survey of systems engineering effectiveness survey.* Carnegie Mellon University and National Defense Industrial Association.

Gerst, K. J. & Rhodes, D. H. (2010). Strenthening systems engineering leading indicators for human systems integration considerations - Insight from the Practitioner Community. *Proceedings of Conference on Systems Engineering Research.*

Gilbert, D., Yearworth, M., Oliver, L. (2014). Systems approach to the development and application of technical metrics to systems engineering projects, *Conference on Sysems Engineering Research, Procedia Computer Science*,Volume 28, Pages 71-80.

LML Steering Committee. (2015). Lifecycle Modeling Language (LML) specification, Retrieved April 1, 2021 from https://lifecyclemodeling.org/wp-content/uploads/2021/01/LML_Specification_1_1.pdf

Jones, C., Draper, G., Golaz, B., Janusz, P. (2021, April 15). Practical Software and Systems Measurement Continuous Iterative Development Measurement Framework, Part 1: Concepts, Definitions, Principles, and Measures. Version 2.1. PSM, NDIA and INCOSE.

McDermott, T.A., Hutchinson, N., Clifford, M., Van Aken, E., Slado, A., & Henderson, K. (2020). Benchmarking the benefits and current maturity of model-based systems engineering across the enterprise. *Systems Engineering Research Center (SERC)* Technical Report SERC-2020-SR-001

Micoun, P., Paper, P., Fabre, L., Razafimahefa, T., Becquet, R. (2018). Property model methodology: A landing gear operational use case. INCOSE International Symposium, hal-01829910

Montgomery, P., & Carlson, R. (2010). *Systems engineering applied leading indicators: enabling assessment of acquisition technical performance.* Naval Postgraduate School, Graduate School of Business and Public Policy, Monterey.

OMG, Object Management Group Standards Organization, https://www.omg.org/hot-topics/syse.htm, Retrieved July 2021.

Orlowski, C. T. (2017). *A framework for implementing systems engineering measures at technical reviews and audits*. Retrieved January 10, 2020, from https://pqdtopen.proquest.com/doc/1878241332.html?FMT=ABS

Orlowski, C., Blessner, P., Blackburn, T., Olson, B. A. (2015). A framework for implementing systems engineering leading indicators for technical reviews and audits. *Procedia Computer Science: Complex Adaptive Systems Conference, 61*, 293-300.

Parrot, E. & Weiland, K. (2017). Using model-based systems engineering to provide artifacts for nasa project life-cycle and technical reviews. *AIAA Space and Astronautics Forum and Exposition*, (p. 5299).

PSM. (2020). *Practical Software & Systems Measurement*. Retrieved from Practical Software & Systems Measurement: http://www.psmsc.com

Rhodes, D.H. (2021). Adapting Systems Engineering Leading Indicators to the Digital Engineering & Management Paradigm. *18th NPS Annual Acquistion Research Symposium.*

Rhodes, D.H. (2020). Investigation of leading indicators for systems engineering effectiveness in model-centric programs. *17th NPS Annual Acquistion Research Symposium. https://dair.nps.edu/bitstream/123456789/4209/1/SYM-AM-20-060.pdf*

Rhodes, D. H., Valerdi, R., & Roedler, G. J. (2009). Systems engineering leading indicators for assessing program and technical effectiveness. *Systems Engineering, 12*(1), 21-35.

Rhodes, D. H., Valerdi, R., Gerst, K. J., & Ross, A. M. (2009). Extending Leading indicators for human systems integration effectiveness. *7th Conference on Systems Engineering Research*

Roedler, G. J. & Rhodes, D. H. (2007). *Systems engineering leading indicators guide, Version 1.0.* MIT, INCOSE, PSM

Roedler, G. J., Rhodes, D. H., Schimmoller, H., & Jones, C. (2010). *Systems engineering leading indicators guide, Version 2.* MIT, INCOSE, PSM. INCOSE-TP-2005-001-03

Shirley, E. (2016). A*pplication of system engineering leading indicators to scrum agile projects*, Master's thesis, Air Force Institute of Technology.

SERC, Systems Engineering Research Center (SERC). (2020). *SERC Website*. Retrieved from http://sercuarc.org

Tepper, Nadia (2010). Exploring the use of Model-Based Systems Engineering (MBSE) to develop Systems Architectures in Naval Ship Design. MIT SM thesis. https://dspace.mit.edu/handle/1721.1/61910

Vaneman, W.K., Ph.D. (2018), Evolving Model-Based Systems Engineering Ontologies and Structures. INCOSE International Symposium, 28: 1027-1036. https://doi.org/10.1002/j.2334-5837.2018.00531.x

Zheng, L., Baron, C., Esteban, P., Xue, R., Zhang, Q., & Yang, S. (2017). Considering the systems engineering leading indicators to improve project performance measurement. IFAC-PapersOnLine, *50*(1), pp. 13970-13975.

Zheng, L., Baron, C., Esteban, P., Xue, R., Zhang, Q., & Yang, S. (2019). Using leading indicators to improve project performance measurement. *Journal of Systems Science and Systems Engineering, 28*(5), pp. 529-554.