EasyChair Preprint

№ 11333

An Empirical Survey on the Prevalence of Technical Debt in Systems Engineering

Howard Kleinwaks, Ann Batchelor and Thomas Bradley

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

November 18, 2023

33rd Annual INCOSE international symposium
hybrid event
Honolulu, HI, USA
July 15 - 20, 2023

An Empirical Survey on the Prevalence of Technical Debt in Systems Engineering

Howard Kleinwaks
Colorado State University
Modern Technology Solutions, Inc.
Engineering Building 202
6029 Campus Delivery
Fort Collins, CO 80523-6029
howard.kleinwaks@colostate.edu

Ann Batchelor
Colorado State University
Engineering Building 202
6029 Campus Delivery
Fort Collins, CO 80523-6029
ann.batchelor@colostate.edu

Dr. Thomas Bradley
Colorado State University
Engineering Building 202
6029 Campus Delivery
Fort Collins, CO 80523-6029
thomas.bradley@colostate.edu

Copyright © 2023 by Howard Kleinwaks, Ann Batchelor, and Thomas Bradley. Permission granted to INCOSE to publish and use.

Abstract. The technical debt metaphor is used within software engineering to describe technical concessions that produce a short-term benefit but result in long-term consequences. Systems engineering is subject to these concessions, yet there is a limited amount of research associating technical debt with systems engineering. This paper provides the results of an empirical survey investigating the prevalence of technical debt in systems engineering, including the occurrence of technical debt, the use of the metaphor, and the distribution of technical debt within the systems engineering lifecycle. The results of the survey show that while technical debt is common in systems engineering and occurs throughout the lifecycle, the metaphor and terminology of technical debt is not consistently applied. These results emphasize the need to enrich the usage of the technical debt metaphor within systems engineering to enable the management of technical debt and to reduce the risk of technical bankruptcy.

Introduction and Background

Modern technology and the digital engineering transformation are increasing the emphasis on delivering flexible systems more rapidly (Dunlap & Chesebrough 2021). Agile systems engineering methods (Douglass 2016) and iterative and incremental development strategies (Mooz & Forsberg 2004) are used to increase flexibility and to limit the cost and schedule increases traditionally associated with requirements changes (Schmidt, Weiss & Paetzold 2018). While Agile processes can be mapped to the system development lifecycle (Darrin & Devereux 2017), the increased

emphasis on shorter times to market can result in "system sponsors and stakeholders… encourag[ing] developers to take shortcuts early in the development process in order to get system capabilities deployed quickly" (Lane, Koolmanojwong & Boehm 2013, p.1386).

If not carefully managed, the decisions made during the planning and execution of iterations can have far reaching consequences on the future state of the system. The work in each iteration places design constraints on future iterations (Forsberg & Mooz 1995) which can result in more expensive changes later in the development cycle (Walden, et al. 2015) or the failure to meet performance objectives. Projects may start work prior to fully understanding the problem in order to deliver a working system faster and then rely on user feedback to improve the system to better meet the users' needs. However, the initial decisions made to produce early value may result in severe inefficiencies in the implemented system, such as lower usability and increased rework later in the development schedule (Ciolkowski, Lenarduzzi & Martini 2021). This phenomenon is known as technical debt.

The technical debt metaphor was introduced as a method to communicate the need to refactor software code to remove short-cuts that were put in place to meet a goal, such as a scheduled release, before those short-cuts could add up to larger problems within the system (Cunningham 1992). Much like financial debt, technical debt accrues interest, which manifests as increased development timelines, increased project cost, and/or rework later in the development cycle. Unmanaged technical debt may lead to technical bankruptcy – the state where system development cannot continue without first repaying back the technical debt (Li, Avgeriou & Liang 2015).

Since 2008, published research on technical debt in the field of software engineering has steadily increased (Besker, Martini & Bosch 2017). Technical debt has been classified into multiple types (Rosser & Ouzzif 2021, Lenarduzzi, et al. 2021), different causes have been identified (Martini, Bosch & Chaudron 2014, McConnell 2008), and multiple measurement techniques have been suggested (Brown, et al. 2010, Nord, et al. 2012, Seaman & Guo 2011, Abad & Ruhe 2015). This research, however, has been primarily constrained to the field of software engineering (Kleinwaks, Batchelor & Bradley 2023). Despite the fact that systems engineering has borrowed many concepts from software engineering, including lifecycle models and development approaches (Darrin & Devereux 2017), there is not a substantial amount of published research on technical debt with systems engineering (Kleinwaks, Batchelor & Bradley 2023).

The concepts behind technical debt are not new to systems engineering. Terminology such as 'rework' has been used to define similar problems. Guenov and Barker (2005) applied axiomatic design theory and design structure matrices to identify design conflicts that result in delays due to unplanning iterations and rework. Boehm, Valerdi, and Honour (2008) discuss the reduction in rework that can be achieved by applying systems engineering to software-intensive systems. Broniatowski and Moses (2016) define a "rework potential" to measure the rework associated with design choices. Raman and D'Souza (2019) developed a decision learning framework that, in part, addresses the uncertainty of architectural design decisions, including those that may lead to more effort than an optimal solution. Shallcross et al. (2020) discuss the use of set based design to limit premature design decisions which may result in expensive rework. Siyam, Wynn, and Clarkson (2015) identify a need to evaluate how changes in processes affect the value of a system later in the lifecycle. Bahill (2012) developed a process to deal with unintended consequences. These research papers all define similar problems to technical debt - minimizing the amount of effort required to correct a technical issue through early detection and mitigation. However, none of the

cited works include the term “technical debt.” Instead, each paper uses their own terminology to describe the problem.

Recognizing that the lack of a common ontology prevents a common understanding of the problem (Uschold & Jasper 1999), Kleinwaks, Batchelor & Bradley (2023) conducted a systematic literature review to determine the prevalence of the technical debt metaphor within published systems engineering research. They concluded that the technical debt metaphor is not prevalent in published papers on systems engineering, that there is not a consensus definition for technical debt within systems engineering, that there is little empirical evidence on the impact of technical debt within systems engineering, and that a common ontology for technical debt in systems engineering has not been established.

Kleinwaks, Batchelor, and Bradley (2023) recommend gathering empirical data to understand the use of the technical debt metaphor by practicing systems engineers to supplement the research in the literature review. This paper provides the results of an empirical survey following this recommendation. The survey was constructed to answer the following research questions:

- RQ1: Does technical debt occur within systems engineering, and if so, what is its impact?
- RQ2: What are the causes of technical debt within systems engineering?
- RQ3: How prevalent is the use of the technical debt metaphor among systems engineering practitioners?
- RQ4: Where does technical debt occur within the systems engineering lifecycle?

The rest of this paper is structured in four sections. First, the research methodology is presented. Next, the primary findings of the survey are presented. Then, the findings are discussed in the context of the research questions. Finally, the paper is concluded and concepts for future work are presented.

Research Methodology

Study Method. This research was conducted using an online survey tool to collect responses to a series of questions designed to assess the respondents’ familiarity with situations that can be classified as technical debt, their familiarity with the metaphor of technical debt, and the stages in the systems engineering lifecycle where technical debt occurs.

Participants in the study were recruited through email solicitations and social media postings sent to specific groups of systems engineers, including a corporate systems engineering community of practice, a graduate university systems engineering department, the local INCOSE chapter, and the authors’ LinkedIn networks. These participant groups were selected based on experience with systems engineering as well as the ability of the authors to contact the group members.

The survey was conducted anonymously, however, some basic demographic questions, such as current position and years of experience, were asked in order to inform the data analysis process. The survey was first released on July 14, 2022 and was closed on August 31, 2022. 50 respondents replied to at least one question in the survey.

Data Analysis. The collected data reports were generated with an anonymous respondent identifier that was matched to the responses for each question. Respondents were not required to answer every question and therefore percentages are reported based on the number of respondents who answered the question and not on the total number of participants in the survey. Several questions allowed the respondent to select multiple responses; in these cases, the percentages are reported as the number of respondents who selected that answer and therefore the percentages may add up to be greater than 100%.

Threats to Validity. The internal validity of a study is the measure of how well the collected data corresponds to the research questions (Crawford 2019). The internal validity is assessed by examining the potential biases that may arise within the study formulation, including the development of the research questions and the survey questions. To limit biases in the development of the research questions, gaps in the current state of academic research on technical debt in systems engineering (Kleinwaks, Batchelor & Bradley 2023) formed the basis of the research questions. Multiple researchers reviewed and developed the survey questions to confirm that they mapped to the research questions. Upon completion, a professional systems engineer evaluated the survey questions and the authors refined the questions based upon the engineer's feedback. Terminology was carefully selected in Question Group 2 to avoid the use of the term "technical debt" in the questions to minimize previous familiarity (or lack thereof) with the term from biasing the answers prior to the introduction of the technical debt metaphor within the survey. The data was collected using an online survey tool that allowed for anonymous responses to prevent biases in reviewing and analyzing the data.

The external validity of the study is the measure of how well the research findings can be extended from the sample group to the general population of interest (Crawford 2019). In this study, the sample group was recruited through email and social media postings. The general population of interest is the set of professional systems engineers, across all disciplines. The majority of respondents indicated background in similar industries, especially the defense industry. This factor has the potential to bias the results towards the defense industry, and therefore the results of the survey may be more generalizable to that subset of professional systems engineers. Another concern prior to the execution of the survey was that a potential bias may arise if software engineers responded to the survey, due to the familiarity of the technical debt metaphor within systems engineering. This concern is addressed in later in this paper.

Given the lack of published research on, and common definitions of, technical debt within systems engineering, it is possible that the respondents do not represent a valid source of knowledge for providing responses regarding the occurrence of technical debt within the systems engineering lifecycle. This threat to the study validity is mitigated by providing the survey participants with a common definition of technical debt prior to asking questions in Question Group 3 and Question Group 4.

Survey Questions. Table 1 lists the questions included in the survey, along with a mapping to the research questions. The starred question numbers indicate questions that allowed multiple answers.

Table 1: Survey Questions

#	Question	RQ
1.1	What is your current position?	N/A
1.2	How many years of professional experience do you have?	N/A
1.3	How many years of experience do you have as a systems engineer?	N/A
1.4	What industry do you currently work in?	N/A
2.1	Have you ever worked on a system where a less than ideal short-term solution to a problem created negative long-term impacts on the system? Negative impacts may include issues such as difficulty meeting requirements, decreased ease of use, and increased system maintenance.	RQ1
2.2*	What negative long-term impacts have you experienced from less-than-ideal short-term solutions? Select all that apply.	RQ1
2.3	Do negative long-term impacts arise primarily from decisions to implement less than ideal short-term solutions (e.g., as way to reach a project completion milestone) or from the accumulation of unintentional decisions (e.g., as the by-product of poor requirements)?	RQ2
2.4	When making the decision to implement the less-than-ideal short-term solution, were there any considerations of the potential for negative long-term impacts?	RQ2
2.5*	Which reasons explain why a systems engineer would implement a less than ideal solution that has benefits in the short-term but negative long-term impacts? Select all that apply.	RQ2
2.6	Have you ever had to correct system issues that were due to less-than-ideal short-term solutions that had negative long-term impacts?	RQ1
2.7	If you have had to correct negative long-term impacts of a decision, how did the effort to correct the negative long-term impacts compare to the effort that would have been required to implement the ideal original solution?	RQ1
3.1	Prior to this survey, how familiar were you with the term technical debt?	RQ3
3.2	How frequently do you use the term technical debt in your daily work?	RQ3
3.3	How familiar are your co-workers with the term technical debt?	RQ3
3.4*	In what engineering contexts have you used or heard the term technical debt? Select all that apply.	RQ3
4.1*	In which stage(s) of the systems engineering lifecycle is technical debt most likely to be created (the decision is made to implement then less than ideal solution)? Select all that apply.	RQ4
4.2*	In which stage(s) of the systems engineering lifecycle is the impact of the technical debt (additional work due to the less-than-ideal solution) most likely to be observed? Select all that apply.	RQ4
4.3*	In what stages of the lifecycle is creating technical debt (deciding to implement the less-than-ideal solution) acceptable? Select all that apply.	RQ4
4.4*	In what stages of the lifecycle is creating technical debt (deciding to implement the less-than-ideal solution) unacceptable? Select all that apply.	RQ4

Question Group 1 (QG1) included basic demographic questions to identify the professional background and experience of the participants. Question Group 2 (QG2) contained questions that were designed to identify if survey participants had experience with technical debt without using the

term “technical debt.” Instead, these questions used the terms “less than ideal short-term solution” and “negative long-term impacts.” This terminology was specifically chosen to convey the concepts behind the technical debt metaphor, without relying on the metaphor to convey the meaning. In this way, it is possible to assess the participants experience with the conditions that give rise to technical debt without biasing the answers towards familiarity with the metaphor.

After completing QG2, the respondents were provided with the following definition of technical debt: “Technical debt is a metaphor reflecting technical compromises that can yield short-term benefits but may hurt the long-term health of a system” (Kleinwaks, Batchelor & Bradley 2023). If a respondent indicated that they were not familiar with technical debt, they were provided a short example.

Question Group 3 (QG3) assessed the respondents’ familiarity with the technical debt metaphor, introducing the terminology into the questions. These questions were designed to assess the frequency with which the terminology is used in professional situations. Question Group 4 (QG4) assessed the respondents’ view of the impact of technical debt in the following phases of the systems engineering lifecycle: needs analysis, requirements definition, preliminary design, critical design, integration, verification and validation, and operations. These phases were chosen since they occur in all system development, regardless of the development method used. Agile and iterative development cycles include the same phases; however, the phases are repeated more frequently. The questions in QG4 asked respondents to consider the lifecycle phases where technical debt is likely to be created and observed, and in which lifecycle phases it is acceptable and unacceptable to create technical debt. These questions were designed to identify the lifecycle stages where technical debt identification and management is the most important in preventing technical bankruptcy.

Research Findings

This section presents the main findings of the survey.

Participant Demographics. QG1 asked the respondents to provide information about themselves and their background as systems engineers. The results are shown in Figure 1. The left chart shows the breakdown of the participants by their current position. The middle chart shows the breakdown of the participants by their current industry. The right chart shows the participants’ total professional experience (Total) and their experience as a systems engineer (SE). The chart is colored based on the participant’s current position. For example, the chart shows that 10% of the participants classified themselves as management with 5-10 years of experience as a systems engineer.

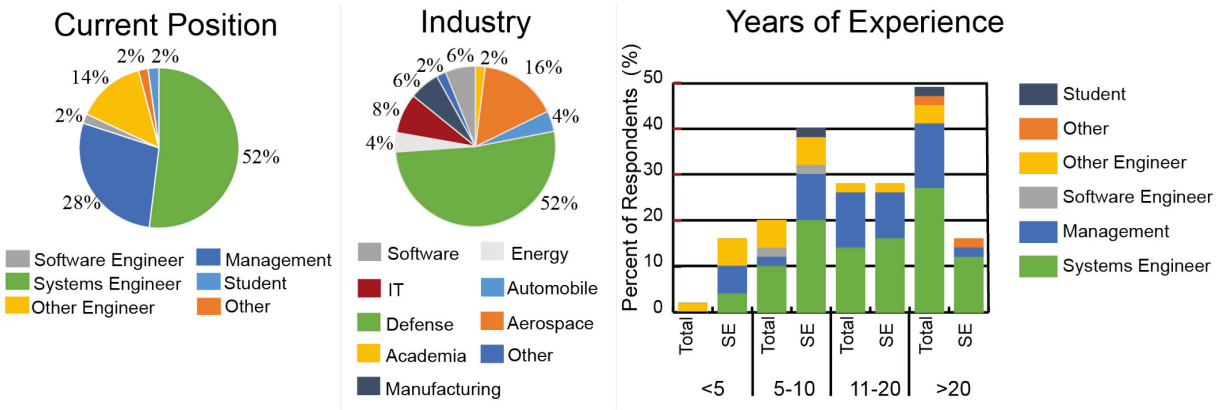


Figure 1. Demographics of Survey Respondents

The majority (52%) of the survey respondents listed systems engineer as their current position (shown in blue in Figure 1). 84% of the respondents reported more than 5 years of experience as a systems engineer, indicating that the respondents have substantial backgrounds in the field, even if they are not currently serving as a systems engineer. These results show that the survey reached the targeted audience of experienced and professional systems engineers. Of note is that only 2% of the respondents listed themselves as a software engineer. One potential concern with the survey was familiarity with the technical debt term due to experience with software engineering. While later results will show that there is likely carryover of terminology from the software engineering field, the limited number of participants who identified as software engineers reduces the concern that the results are biased based on a large number of responses from software engineers.

The majority of respondents (68%) work in the Aerospace and Defense industries. The large section of respondents with similar backgrounds has the potential to bias the results towards those industries.

Technical debt is common in systems engineering. Question 2.1 asked if the respondents experienced the conditions that are defined as technical debt, without using the metaphor. 100% of participants responded that they had worked on such as system. Question 2.6 asked if participants had to correct issues associated with technical debt. 86% percent of the respondents stated that they have had to correct issues caused by less-than-ideal short-term solution.

The answers to question 2.1 and 2.6 clearly indicate that technical debt is a common occurrence within systems engineering. Every respondent experienced negative long-term effects due to short-term decisions, and the large majority of the respondents have corrected issues associated with these decisions. In other words, the respondents have repaid technical debt.

Technical debt accrues interest. Technical debt is typically measured in terms of principal and interest. The principal represents the amount of effort that would have been required to implement the ideal solution (Ampatzoglou, et al. 2015) and the interest refers to additional effort to implement that same solution at a later time, due to the presence of the less-than-ideal solution (Ampatzoglou, et al. 2020). Question 2.7 addressed the presence of technical debt interest in systems engineering. If it is more difficult to correct the problems with a less-than-ideal solution than it would have been to initially implement the ideal solution, then it can be inferred that the technical debt has accrued interest. 79% of the respondents to question 2.7 stated that it was either more effort (36%) or significantly more effort (43%) to correct the issues after the less-than-ideal

solution was implemented, as shown in Figure 2. These data indicate that technical debt accrues interest within systems engineering.

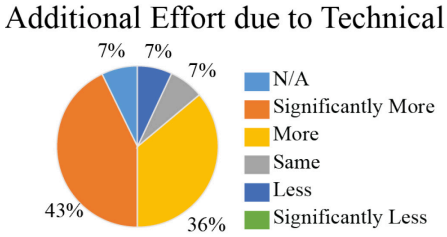


Figure 2. Additional Effort Required to Correct Technical Debt Compared to the Effort to Implement the Ideal Solution Originally

Six survey respondents answered "no" to question 2.6, indicating that they never had to correct issues associated with technical debt. Of those six respondents, three answered that question 2.7 was not applicable to them (N/A in Figure 2), two did not answer question 2.7, and one respondent answered that correcting the issue required less effort. These answers are deemed to have no impact on the overall conclusions from this question, namely that technical debt does accrue interest.

Technical debt has multiple long-term impacts. Question 2.2 asked the participants to specify what negative long term impacts they had observed from implementing less than ideal short-term solutions. Participants were able to select more than one answer and the results are shown in Figure 3. "Failure to meet performance objectives" and "Substantial rework of an earlier part of the system" were the most common responses. Only 4% of the participants selected "Other", indicating that the answer choices well covered the negative impacts due to technical debt.

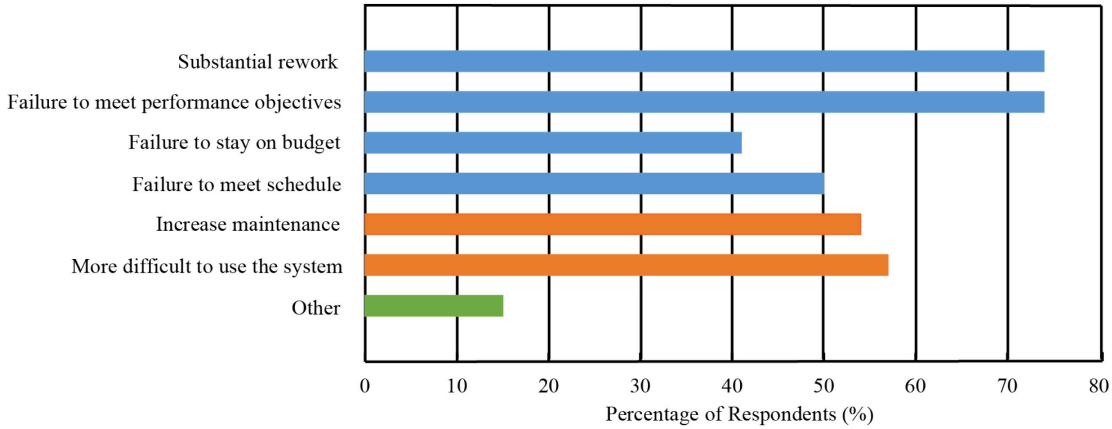


Figure 3. Negative Long-term Impacts of Technical Debt

These data indicate that there is not a single impact of technical debt on a system but rather that the impact is felt in multiple areas. The answer choices cover two areas: those that occur during system development, shown in blue in Figure 3, and those that occur after the system is deployed, shown in orange in Figure 3. Over 50% of respondents indicated that negative long-term impacts occur in both of these areas. From these data, it can be concluded that technical debt is something that will need to be managed throughout the system lifecycle.

Technical debt is driven by schedule and cost pressures and both intentional and unintentional decisions. Questions 2.3, 2.4, and 2.5 addressed the reasons why a project would take on

technical debt. 79% of the respondents indicated that potential long-term consequences were considered when making short-term decisions. These long-term consequences were determined to arise from both intentional and unintentional decisions, as shown in the left side of Figure 4. The right side of Figure 4 shows the reasons for accruing technical debt. Over 80% of the respondents stated that schedule pressure contributes to the decisions to introduce technical debt into the system. Over 60% of the respondents stated that cost pressure contributes to the introduction of technical debt. Technical compromise was selected by 36% of the respondents. These results indicate that cost and schedule are the primary factors that drive a system to make technical compromises and therefore incur technical debt.

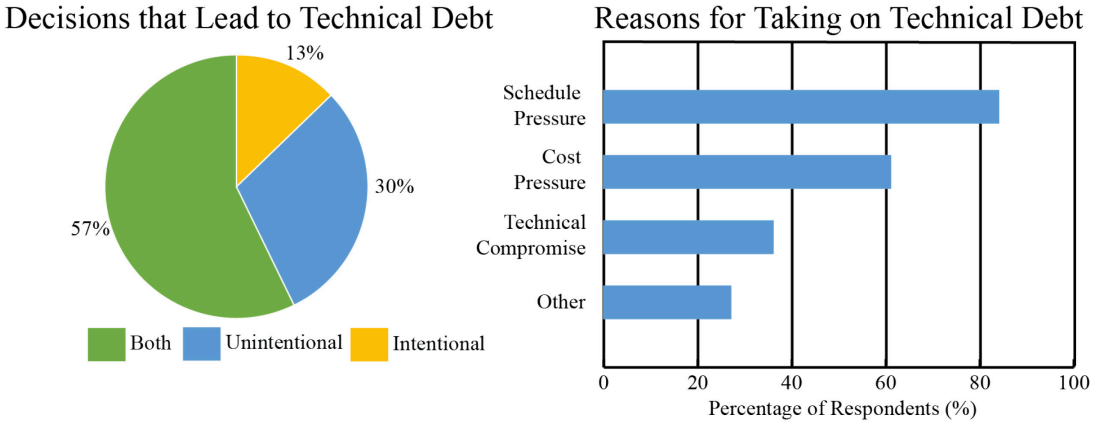


Figure 4. Rationale for Accruing Technical Debt

Participants were allowed to select multiple answers for question 2.5, including identifying other reasons for taking on technical debt. Other responses included acceptance of a prototype, political pressure from management and other external sources, the lack of consideration of long-term goals and impacts in the daily decisions, and the inability to react to previous instances of technical debt. Failure to react to previous instances of technical debt is an indicator that a system may be on a path to technical bankruptcy.

The technical debt metaphor is not common terminology in systems engineering. QG3 assessed the participants' familiarity with and usage of the technical debt metaphor after providing all participants with a common definition of technical debt. The left side of Figure 5 shows the self-assessed familiarity with the metaphor, broken out by the participant's years of experience as a systems engineer. 47% of respondents stated that they were very or extremely familiar with the metaphor and 30% of respondents stated that were either slightly familiar or not at all familiar with the metaphor.

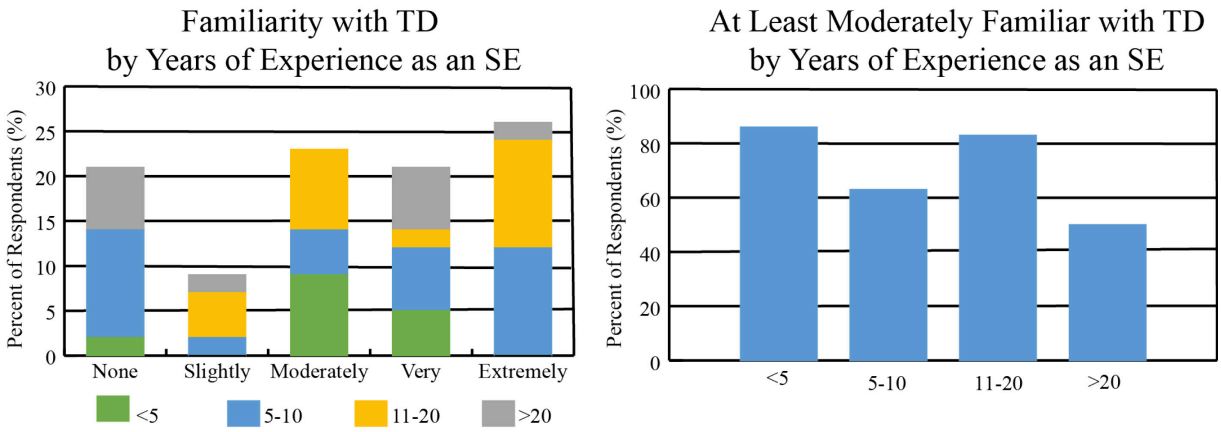


Figure 5. Participant Familiarity with the Technical Debt Metaphor

When examined through the lens of years of experience as a systems engineer, some interesting trends appear. The right side of Figure 5 shows the percentage of respondents who are either moderately familiar, very familiar, or extremely familiar with technical debt based on the respondents' years of experience as a systems engineer. The percentages are based on the total number of respondents with the stated years of experience. For example, seven respondents had less than five years of experience as a systems engineer. Of those respondents, six stated that they were at least moderately familiar with technical debt, resulting in a value of 86%.

While there is not enough data to make conclusive arguments, it can be seen that the less experienced (< 20 years of experience) systems engineers tend to be more familiar with technical debt than the very experienced systems engineers (> 20 years of experience). This could be a result of the small sample size (only eight respondents had > 20 years of experience); however, it could also indicate that the technical debt terminology is better known to less experienced systems engineers due to the relative newness of the terminology. Technical debt research in software engineering accelerated around 2008 (Li, Avgeriou & Liang 2015). Systems have become more software intensive (Boehm 2006) and familiarity with software engineering is now part of recommended systems engineering graduate school curriculum (Pyster, et al. 2012). It is possible that these trends contribute to a greater familiarity with the metaphor among less experienced systems engineers.

Figure 5 shows that overall, there is familiarity with the technical debt metaphor among systems engineers. However, familiarity with a term is not enough to establish that the term is a common part of the lexicon. Therefore, participants were asked to identify how frequently they use the technical debt metaphor and in which technical contexts it is used. These results are shown in Figure 6. The left side of Figure 6 shows the usage of the technical debt metaphor. Only 26% of the participants reported using the technical debt metaphor frequently (gray) or very frequently (yellow), and 56% of the participants reported not frequently using the metaphor (blue). These results indicate that the metaphor, while it may be familiar to systems engineers, is not commonly used.

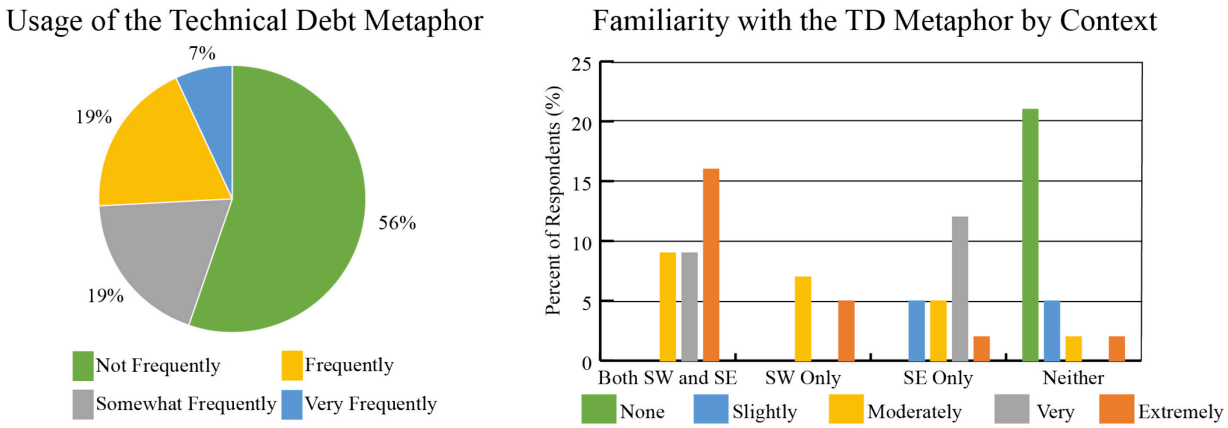


Figure 6. Usage of and Familiarity with the Technical Debt Metaphor in Various Contexts

The right side of Figure 6 shows the answers to question 3.4, which assessed the contexts in which participants have used or heard the technical debt metaphor. 35% of the respondents reported that they have used or heard technical debt in both systems engineering (SE) and software engineering (SW) context. Only 23% of the respondents said they have only used or heard the term in just the SE context and 12% of the respondents stated that they have used or heard the term in just the SW context. A likely interpretation is that the familiarity with the technical debt metaphor from software engineering produces carryover usage in the field of systems engineering. Of note is that the respondents who indicated usage in both the SE and SW context also indicated higher levels of familiarity with the technical debt metaphor. From these results, it can be concluded that the technical debt metaphor is present in the systems engineering lexicon, however, it is not a frequently used component of that lexicon.

Technical debt occurs throughout the system lifecycle. QG4 focused on technical debt in the system lifecycle. The participants' responses, shown in Figure 7, demonstrate that technical debt occurs throughout the system lifecycle, both in terms of its creation and its impact. The left chart in Figure 7 shows the design phases where technical debt is most likely to be created and most likely to be observed, according survey responses. These data show that technical debt is more likely to be created during the design phases of the system and that the impact is more likely to be observed during the integration, verification and validation, and operations phases. These results show why technical debt is dangerous to a program – it is created based on decisions made in early phases, but the impacts are not felt until later phases, when it is more difficult to correct the issues.

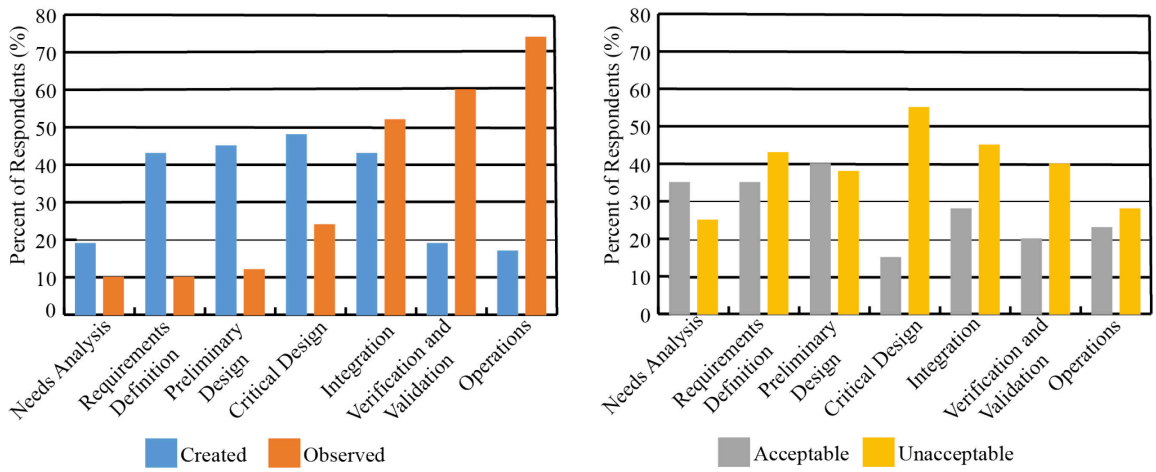


Figure 7. Technical Debt in the System Lifecycle

Questions 4.3 and 4.4 asked if there are specific phases within the system engineering lifecycle where it is more or less acceptable to create technical debt. The right side of Figure 7 shows the responses to these questions. The results show that participants generally viewed technical debt created in the early phases of the program to be more acceptable, with technical debt created during critical design to be the most unacceptable. However, the critical design phase is the phase that was indicated as the most likely place for technical debt to be created. These data support the premise that unmanaged technical debt is dangerous to a system. Technical debt is created where it is deemed unacceptable to do so, since creation in those phases is likely to drive to poor outcomes for the system. Therefore, it is critical to manage the creation of technical debt to prevent later impacts.

Technical debt can be beneficial. The phrasing of the technical debt metaphor implies that the consequences of technical debt are always negative. If true, then it would be expected that the survey respondents would never have indicated that creating technical debt was acceptable. However, as shown in the right side of Figure 7, over 30% of respondents identified that technical debt is acceptable to create in the early stages of the system lifecycle. Why would technical debt creation be acceptable?

While the survey did not ask this question, a reasonable answer is that technical debt creation is acceptable if it provides a benefit to the development of as system. The initial technical debt metaphor highlighted this aspect of technical debt stating "A little debt speeds development so long as it is paid back promptly with a rewrite" (Cunningham 1992). Taking on technical debt can enable a system to achieve critical results, such as delivering on schedule, even if compromises are made in the design. However, without a plan to repay the debt, it may spiral out of control and result in technical bankruptcy.

Discussion

This survey provides an empirical basis for understanding the prevalence of the technical debt metaphor in the field of systems engineering. The results can be used to draw several conclusions based on the research questions.

RQ1: Impact and occurrence of technical debt in systems engineering. The survey results clearly indicate that technical debt commonly occurs within systems engineering. The impacts of technical debt, such as increased effort and increased rework, were clearly identified by survey participants. Participants identified factors that can lead to technical bankruptcy, such as failure to meet cost and schedule, as impacts of technical debt. Participants identified that technical debt creation during early system development phases can be acceptable, indicating that there can be benefits to taking on technical debt.

The confirmation of technical debt as a contributor to project success and failure means that it needs to managed within the systems lifecycle. Tools need to be created to identify, manage, and monitor technical debt to minimize its impact. If a system developer waits until the impact of technical debt is seen in the system, it may be too late or too expensive to correct the issues. The survey results show that technical debt is more likely to be observed later in the system lifecycle, when it is more expensive to correct problems (Walden, et al. 2015). Therefore, technical debt needs to be monitored from the start of the system and should be repaid soon as possible.

RQ2: Causes of technical debt within systems engineering. Multiple factors contribute to technical debt; however, schedule pressure was cited as the top cause by the survey respondents. Schedule pressure is a significant concern in iterative development programs. As systems embrace Agile development strategies, they are often faced with fixed-duration development periods (sprints). Each sprint is intended to deliver a potentially releasable product (Cohn 2010). This combination naturally exerts pressure on the developer to release a working system and can result in the developer taking shortcuts, intentionally or unintentionally, in order to make the delivery timeline. Proper planning involves sequencing tasks based on both the functional value delivered to stakeholders and on the temporal value delivered to the system. Understanding both the functional and temporal dependencies in the system development is critical for avoiding the need to incur technical debt. Supporting requirements, such as quality requirements (maintainability, reliability, etc.), must be given proper weight such that future iterations can begin with all the required infrastructure in place, even if they are not perceived as high-value to the stakeholder. Otherwise, the future iterations are likely to need to take shortcuts, and thereby take on technical debt, to account for the missing components.

Another major driver of technical debt is cost pressure. The system may reach budget limits that require compromise in one area or another. For example, insufficient funding for testing may result in insufficient tests being performed on the system. The lack of testing may then result in an underperforming system. Budget allocations must be sufficient to enable proper system development, or else the system risks accruing technical debt.

While technical compromise was not cited by as many respondents as cost and schedule pressure, it was still cited as a cause of technical debt by over 30% of the respondents. Technical compromise means that the system developer makes technical concessions in one area to enable satisfaction of technical goals in another area, such as reducing the size of a satellite antenna to satisfy the mass constraints. If the full impacts are not assessed, the technical concessions can result in a system that cannot meet its overall performance goals.

RQ3: Use of the technical debt metaphor among systems engineering practitioners. The respondents to the survey stated that they had a broad range of familiarity with the metaphor of technical debt and that they were more familiar with it than their coworkers. However, they also

responded that they do not frequently use the metaphor. These results indicate that the metaphor is not prevalent among systems engineering practitioners. Yet, the responses to QG2 indicate that the impacts of technical debt were observed by all survey respondents. This apparent disconnect highlights that the impacts associated with technical debt are real, but that it is not part of the lexicon of systems engineering. Instead, systems engineers use terms such as rework (Broniatowski & Moses 2016) and unintended consequences (Bahill 2012), however a detailed examination of the terminology in current use was outside the scope of this survey.

The survey results show that systems engineers understand some aspects of technical debt, such as the implications of short-term decisions on the long-term health of the system. However, the lack of general usage of the metaphor implies that the full richness of the technical debt metaphor is not used or understood. Simply delaying work does not result in technical debt and identifying the potential for rework does not quantify the impact on the future state of the system.

The use of inconsistent vocabulary creates barriers to effective communications even amongst practitioners in the same field (Uschold & Jasper 1999). The technical debt metaphor, through its use of concept such as principal, interest amount, and interest probability, can create a consistent vocabulary to allow systems engineers to quantify the impact of decisions. The quantified impact can then be used to support the decision-making processes during system development. Technical debt ontologies have been proposed within software engineering (Alves, et al. 2014); however, even the definition of technical debt is not agreed upon within systems engineering (Rosser & Norton 2021). The results of this survey indicate that the technical debt terminology is not widespread within the systems engineering field, and this may be due, in part, to the lack of a consistent ontology. Development of such a ontology, specific to systems engineering applications, will aid in furthering the understanding of the impacts of technical debt and developing strategies for managing technical debt when it occurs.

RQ4: Occurrence of technical debt within the systems engineering lifecycle. The survey results show that technical debt is more likely to be created early in the systems engineering lifecycle and also more likely to be observed late in the systems engineering lifecycle. This combination results in an accumulation of interest on the technical debt and is what makes technical debt expensive to the systems developer.

Of particular interest is the combination of the most respondents stating that technical debt is likely created during critical design and the most respondents stating that it was unacceptable to create technical debt during critical design. These data indicate that systems engineers may “know what they are doing is wrong” during the critical design phase, and yet they do it anyway – intentionally creating technical debt to get the design completed. If there is no plan to manage and pay back this technical debt, then it can be harmful to the system. This technical debt will then likely appear in the integration and/or operations phases. These results also indicate how technical debt can arise – schedule pressures and other outside influences can force the system developer to take those short cuts to complete the design by a set time. These data reinforce the need to manage and monitor technical debt. It is when the most critical elements of the development occur that taking on technical debt is most likely, and also the most unacceptable.

Conclusion and Future Work

Kleinwaks, Batchelor, and Bradley (2023) proposed a research agenda to develop a systems engineering-centric view of technical debt. This agenda includes:

- Gathering empirical data to baseline the usage of the technical debt metaphor and the impacts of technical debt within systems engineering applications;
- Developing an ontology of technical debt for the field of systems engineering, developing methods and techniques to identify causes and occurrences of technical debt within systems development, developing processes and methods to measure technical debt; and,
- Verifying and validating the processes developed through application to systems engineering problems.

This survey represents the first step in the above research agenda and its results form the basis from which the above research agenda can be continued. The survey provides an empirical basis for the usage of technical debt within the systems engineering field and future work will continue to develop this usage. The survey results will guide the development of the ontology of technical debt by providing area of emphasis where common language is required. For example, the prevalence of the impact of technical debt is clear from the survey results, but respondents do not use the same terminology. Additional surveys can be conducted to determine the terminology that is used in practice, which will further inform the development of the ontology.

This survey has provided a substantial amount of empirical evidence leading to the following key conclusions:

- Technical debt is common in systems engineering applications, but the associated terminology is not frequently used.
- Technical debt results in problems with system performance, cost, and schedule and bears interest – it requires more effort to correct the problem then it would have taken to do it correctly in the first place
- Cost and schedule pressure are the primary drivers of technical debt
- Technical debt is created early in the system lifecycle and observed late in the system lifecycle

The impacts of technical debt on a system are real and substantial. By enriching the usage of the technical debt metaphor within systems engineering, a common language can be used to manage and reduce those impacts. This research will continue to fulfill the above research agenda to provide a mechanism for managing technical debt to reduce the risk of technical bankruptcy.

References

Abad, ZSH and G Ruhe 2015, 'Using Real Options to Manage Technical Debt in Requirements Engineering', *2015 IEEE 23rd International Requirements Engineering Conference (RE),* IEEE, pp. 230-235.

Alves, NSR, LF Ribeiro, V Caires, TS Mendes, and RO Spinola 2014, 'Towards an Ontology of Terms on Technical Debt', *2014 Sixth International Workshop on Managing Technical Debt*, IEEE, pp. 1-7.

Ampatzoglou, A, A Ampatzoglou, A Chatzigeorgiou, and P Avgeriou 2015, 'The Financial Aspect of Managing Technical Debt: A Systematic Literature Review', *Information and Software Technology,* vol. 64, pp. 52-73.

Ampatzoglou, A, N Mittas, AA Tsintzira, A Ampatzoglou, EM Arvanitou, A Chatzigeorgiou, P Avgeriou, and L Angelis 2020, 'Exploring the Relation between Technical Debt Principal and Interest: An Empirical Approach' *Information and Software Technology,* vol. 128.

Bahill, AT 2012, 'Diogenes, a Process for Identifying Unintended Consequences', *Systems Engineering,* vol. 15, no. 3, pp. 287-306.

Besker, T, A Martini, and J Bosch 2017, 'Managing Architectural Technical Debt: A Unified Model and Systematic Literature Review', *The Journal of Systems and Software,* vol. 135, pp. 1-16.

Boehm, BW 2006, 'Some Future Trends and Implications for Systems and Software Engineering Processes', *Systems Engineering,* vol. 9., no. 1, pp. 1-19.

Boehm, BW, R Valerdi, and E Honour 2008, 'The ROI of Systems Engineering: Some Quantitative Results for Software-intensive Systems', *Systems Engineering,* vol. 11, no. 3, pp. 221-234.

Broniatowski, DA, and J Moses 2016, 'Measuring Flexibility, Descriptive Complexity, and Rework Potential in Generic System Architectures' *Systems Engineering*, vol. 19, no 3, pp. 207-221.

Brown, N, Y Cai, Y Guo, R Kazman, M Kim, P Kruchten, E Lim, et al. 2010, 'Managing Technical Debt in Software-Reliant Systems', *Proceedings of the FSE/SDP workshop on Future of software engineering research,* pp. 47-52.

Ciolkowski, M, V Lenarduzzi, and A Martini 2021, '10 Years of Technical Debt Research and Practice: Past, Present, and Future', *IEEE Software,* vol. 38, no. 6, pp. 24-29.

Cohn, M 2010, *Succeeding with Agile: Software Development Using Scrum,* Addison-Wesley, Upper Saddle River, NJ (US).

Crawford, LM 2019, 'Qualitative Research Designs', in GJ Burkholder, KA Cox, LM Crawford, and JH Hitchcock (eds) *Research Design and Methods: An Applied Guide for the Scholar-Practitioner*, SAGE Publications, United States, pp. 81-98.

Cunningham, W 1992, 'The WyCash Portfolio Management System', 26 March 1992, viewed 29 January 2022, <http://c2.com/doc/oopsla92.html>.

Darrin, MAG and WS Devereux 2017, 'The Agile Manifesto, Design Thinking, and Systems Engineering', *2017 Annual IEEE International Systems Conference (SysCon)*, pp. 1-5.

Douglass, BP 2016, *Agile Systems Engineering*, Morgan Kaufmann, Waltham (US).

Dunlap, H and D Chesebrough 2021, 'Transforming Our Systems Engineering Approach Using Digital Technology', National Defense Industrial Association, viewed 10 April 2022, <https://www.nationaldefensemagazine.org/articles/2021/10/4/transforming-our-systems-engineering-approach-using-digital-technology>.

Forsberg, K and H Mooz 1995, 'Application of the 'Vee' to Incremental and Evolutionary Development', *INCOSE International Symposium,* vol. 5, no. 1, pp. 848-855.

Guenov, MD and SG Barker 2005, 'Application of Axiomatic Design and Design Structure Matrix to the Decomposition of Engineering Systems', *Systems Engineering,* vol. 8, no. 1, pp. 29-40.

Kleinwaks, H, A Batchelor, and TH Bradley 2023, 'Technical Debt in Systems Engineering', *Systems Engineering*, 10 April 2023, pp 1-13.

Lane, JA, S Koolmanojwong, and BW Boehm 2013, '4.6.3 Affordable Systems: Balancing the Capability, Schedule, Flexibility, and Technical Debt Tradespace' *INCOSE International Symposium,* vol. 23, pp. 1385-1399.

Lenarduzzi, V, T Besker, D Taibi, and A Martini 2021, 'A Systematic Literature Review on Technical Debt Prioritization: Strategies, Processes, Factors, and Tools', *The Journal of Systems and Software*, vol. 171.

Li, Z, P Avgeriou, and P Liang 2015, 'A Systematic Mapping Study on Technical Debt and its Management' *The Journal of Systems and Software,* vol. 101, pp. 193-220.

Martini, A, J Bosch, and M Chaudron 2014, 'Architecture Technical Debt: Understanding Causes and a Qualitative Model', *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 85-92.

McConnell, S 2008, 'Managing Technical Debt', Construx Software Builders, < http://www.construx.com/uploadedfiles/resources/whitepapers/Managing%20Technical%20Debt.pdf>

Mooz, H and K Forsberg 2004, 'Clearing the Confusion About Spiral/Evolutionary Development', *INCOSE International Symposium,* vol. 14, no. 1, pp. 1675-1688.

Nord, RL, I Ozkaya, P Kruchten, and M Gonzalez-Rojas 2012, 'In Search of a Metric for Managing Architectural Technical Debt', *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, pp. 91-100

Pyster, A, DH Olwell, TLJ Ferris, N Hutchison, S Enck, JF Anthony, D Henry, and A Squires (eds) 2012, 'Graduate Reference Curriculum for Systems Engineering (GRCSE™) V1.0', The Trustees of Stevens Institute of Technology.

Raman, R and M D'Souza 2019, 'Decision Learning Framework for Architecture Design Decisions of Complex Systems and System-of-Systems', *Systems Engineering*, vol. 22, no. 6, pp. 538-560.

Rosser, LA and JH Norton 2021, 'A Systems Perspective on Technical Debt', *2021 IEEE Aerospace Conference (50100)*, IEEE, pp. 1-10.

Rosser, LA and Z Ouzzif 2021, 'Technical Debt in Hardware Systems and Elements', *2021 IEEE Aerospace Conference (50100).* IEEE, pp. 1-10.

Schmidt, TS, S Weiss, and K Paetzold 2018, 'Expected vs. Real Effects of Agile Development of Physical Products: Apportioning the Hype', *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, pp. 2121-2132

Seaman, C and Y Guo 2011, 'Measuring and Monitoring Technical Debt', *Advances in Computers,* vol. 82, pp. 25-46.

Shallcross, N, GS Parnell, E Pohl, and E Specking 2020, 'Set-based Design: The State-of-Practice and Research Opportunities', *Systems Engineering*, vol. 23, no. 5, pp. 557-578.

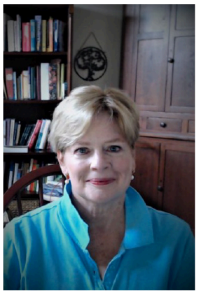
Siyam, GI, DC Wynn, and PJ Clarkson 2015, 'Review of Value and Lean in Complex Product Development', *Systems Engineering,* vol. 18, no. 2, pp. 192-207.

Uschold, M and R Jasper 1999, 'A Framework for Understanding and Classifying Ontology Applications', *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*.

Walden, DD, GJ Roedler, KJ Forsberg, RD Hamelin, and TM Shortell 2015, *Systems Engineering Handbook*. 4th Edition, John Wiley and Sons, Hoboken (US).

Biography

Howard Kleinwaks is currently a candidate for the degree of Doctor of Engineering in Systems Engineering at Colorado State University in Fort Collins, CO. He is researching the management of technical debt in iterative systems development. He received both a Bachelor of Science and a Master of Science in Aerospace Engineering from MIT. He is the Chief Engineer of the Strategic Space Business Unit at Modern Technology Solutions, Inc. (MTSI) and has over 15 years of experience in the aerospace and systems engineering fields. He is a Project Management Professional (PMP), Associate Systems Engineer (ASEP), and Professional Scrum Master.



Ann Batchelor is a professor of Systems Engineering at Colorado State University, teaching Engineering Project and Program Management, Systems Requirements Engineering, and Engineering Risk Assessment. She has 20+ years of industrial experience in technical management, systems engineering, production, manufacturing, lean engineering, life cycle management, test and analysis, proposal and project management and technical writing. She has held roles including chief scientist, systems engineer, Director of Engineering, and Director of Program Management. She is a past certified program management professional (PMP), a Military Sensing Fellow (DOD Informational and Analysis Center for Military Sensing), and former President-elect of the INCOSE Atlanta Chapter.

Dr. Thomas H. Bradley, Ph.D. serves as the Woodward Foundation Professor and Department Head for the Department of Systems Engineering at Colorado State University. He conducts research and teaches a variety of courses in system engineering, multidisciplinary optimization, and design. Dr. Bradley's research interests are focused on applications in Automotive and Aerospace System Design, Energy System Management, and Lifecycle Assessment. Bradley earned the BS and BS in Mechanical Engineering at the University of California - Davis, and the PhD in Mechanical Engineering at Georgia Institute of Technology. He is a member of INCOSE, SAE, ASME, IEEE, and AIAA.