American Journal of Information Science and Technology
2025, Vol. 9, No. 2, pp. 87-93
https://doi.org/10.11648/j.ajist.20250902.13

Research Article

The Path from Software Engineering to System Engineering: Gamification Based S2S-G Framework

Wei Ren* , **Bo Qin**

China Academy Electronics and Information Technology, Beijing, China

Abstract

System engineering is a multidisciplinary, structured approach designed to manage the lifecycle of complex systems, ensuring their effective design, integration, and retirement. Benefits of system engineering include reduced risks, better stakeholder participation, adaptable systems, and improved documentation. However, as systems become more complex, traditional methodologies are often insufficient, leading to the emergence of Model-Based System Engineering (MBSE). MBSE, using Systems Modeling Language (SysML), offering a feasible pathway for software engineers transitioning to systems engineering through focused training. While software engineering shares similarities with systems engineering, particularly in process and goal alignment, the two disciplines differ significantly in scope and focus. The challenge lies in bridging the knowledge and mindset gaps between the two fields, as software engineers often struggle to transition to systems engineering due to differences in methodologies and focus areas. Gamification, the integration of game design elements into non-game contexts, has gained attention as a tool to facilitate this transition. This study compares software engineering and systems engineering, this work highlights their similarities and differences and proposes the S2S-G Framework, a gamification based framework, as a structured, effective tool to bridge the gap between the two disciplines.

Keywords

System Engineering, Software Engineering, Gamification, S2S-G Framework, MBSE, SysML

1. Introduction

System engineering is a structured, multidisciplinary approach aimed at designing, managing, and retiring systems effectively throughout their lifecycle [1]. Its primary function is to guide the engineering of complex systems through processes like defining stakeholder needs, system requirements, conceptual design, development, integration, testing, and validation [1, 4]. This approach emphasizes iterative refinement to ensure systems meet functional, performance, and reliability standards within budget and schedule constraints. Benefits include reduced risks of cost and schedule overruns, improved stakeholder participation, adaptable systems, better documentation, and fewer defects [6].

System engineering ensures technical integrity by guiding design processes, assessing concept options, managing technical information, addressing stakeholder issues, and resolving challenges throughout the system's lifecycle [1]. As systems grow increasingly complex and interdisciplinary, traditional methodologies may prove insufficient. Consequently, Model-Based System Engineering (MBSE) has emerged as a new approach to enhance information quality and optimize

*Corresponding author: renwei10@cetc.com.cn (Wei Ren)

Received: 24 March 2025; **Accepted:** 28 April 2025; **Published:** 22 May 2025

communication between teams, thereby simplifying the management of complex projects [2]. MBSE's core technology, Systems Modeling Language (SysML), evolved from Unified Modeling Language (UML), a standard tool in software engineering. The transition from UML to SysML is feasible, and software engineers are more easily transitioning to systems engineering through training [24].

Software engineering applies engineering principles to design, develop, maintain, test, and evaluate software, integrating computer science with engineering practices to deliver quality software on time and within budget [23]. It involves phases like requirements gathering, design, implementation, testing, and deployment [3]. Although software engineering and system engineering has similarity, but it still a challenge convert software engineer to system engineer due to knowledge gaps, mindset shifts, or lack of formal training. Software engineering and systems engineering are related but differ in scope and focus. For example, software engineering focuses on software components, systems engineering ensures their cohesive integration into larger systems [17].

Gamification is a way to help transfer from software engineering to system engineering, which is defined as incorporating game design elements into non-game settings, and gamification is gaining attention across various fields, including software engineering and systems engineering [8-12]. This study develops a gamification framework: S2S-G Framework. S2S-G Framework help software engineering convert to system engineering, which include four parts.

This work contributes to the field by providing a comprehensive comparison between software engineering and systems engineering, emphasizing their distinctive yet complementary roles in the development of complex projects. While both disciplines share foundational principles of engineering design, their scope, focus, and methodology differ significantly, influencing how they approach problem-solving, project management, and system integration. Through this comparison, we illuminate the unique challenges faced by engineers transitioning between these domains, particularly in multidisciplinary environments where collaboration is key.

A core contribution of this study is the introduction of the S2S-G Framework, a novel gamification-based approach designed to facilitate the transition from software engineering to systems engineering. By leveraging gamification techniques, the S2S-G Framework encourages engineers to approach systems engineering problems with a mindset that fosters innovation, adaptability, and a holistic view of project challenges. Moreover, the S2S-G Framework is designed to be adaptable, allowing it to be customized for different project contexts, whether in academia, industry, or large-scale, cross-functional teams. This work not only advances the theoretical understanding of the relationship between software and systems engineering but also offers a solution— the S2S-G Framework—that directly addresses the need for improved interdisciplinary collaboration and knowledge transfer in the face of increasingly complex and integrated engineering projects.

2. Related Work and Hypothesis Development

Systems engineering is a methodical, multidisciplinary approach used for designing, developing, managing, operating, and retiring complex systems [1]. Its primary function is to guide the engineering process of intricate systems by ensuring a structured process that begins with defining stakeholder needs and system requirements, followed by conceptual design, detailed design, development, integration, testing, and validation [1, 4]. This iterative process ensures that systems meet functional, performance, and reliability standards while adhering to budget and schedule constraints.

The benefits of systems engineering include reducing the risk of schedule and cost overruns, improving stakeholder participation, shortening project cycles, enhancing adaptability and resilience of systems, verifying functionality, reducing defects, and producing better documentation [6]. Systems engineering's responsibilities cover maintaining technical integrity throughout the system's lifecycle, guiding end-to-end engineering design, assessing concept options, properly archiving technical information, resolving stakeholder issues, and addressing questions as they arise [1].

In recent years, systems have become increasingly complex and interdisciplinary [2]. Traditional methodologies are often insufficient to address these complexities. Therefore, a new approach, Model-Based Systems Engineering (MBSE), has emerged as a promising solution. MBSE helps improve information quality within a project and simplifies stress management by enhancing communication between diverse teams [2].

The core technology of MBSE is the Systems Modeling Language (SysML), which is an extension of the Unified Modeling Language (UML), a foundational tool in software engineering. As a specialization of UML, SysML introduces advanced diagram types, such as Requirement Diagrams and Parametric Diagrams, which enable comprehensive modeling of system structures, behaviors, requirements, and constraints [25]. Unlike UML, which focuses primarily on software visualization through diagrams like Class Diagrams and Sequence Diagrams, SysML offers broader capabilities suitable for systems engineering. This technological continuity makes it easier for software engineers to transition to systems engineering, with 68% of those trained in UML successfully making the shift through training programs averaging 4.2 months [24].

Software engineering, on the other hand, is the disciplined application of engineering principles to the design, development, maintenance, testing, and evaluation of software systems [7]. It combines computer science knowledge with engineering practices to produce high-quality software that meets user requirements and is delivered within budget and on

time.

The software engineering process generally includes phases such as requirements gathering, design, implementation, testing, and deployment [3]. Its benefits include improved software quality and reliability, reduced risks through better management, increased efficiency in development cycles, enhanced user satisfaction, better maintainability through documentation, and improved collaboration among stakeholders [7].

The responsibilities of software engineers include designing and maintaining software systems, evaluating and testing new software programs, optimizing software for speed and scalability, and consulting with various stakeholders such as clients, engineers, and security specialists [5]. Thus, as Table 1 shows, the progress, benefits, and benefits of software engineering and system engineering are very similar.

Table 1. *Similarity of Software Engineering and System Engineering.*

Aspects	Software Engineering	System Engineering	Similarity
Progress	Design	Design	Yes
	Implementation	Implementation	Yes
	Testing	Testing	Yes
	Operation	Operation	Yes
	Maintaining Software	Maintaining System	Yes
Responsibilities	Guiding Design	Guiding Design	Yes
	Evaluating Testing	/	/
	Optimizing Software	Ensuring info archived	Yes
	Stakeholders	Stakeholders	Yes
Benefits	Risk Management	Reducing Risk	Yes
	User Satisfaction	Meet User's Needs	Yes
	Efficiency	Shorter Project Cycles	Yes
	Scalability	Adaptable and Resilient	Yes
	Improved Quality	Verified Functionality	Yes
	Maintainability	Better Documentation	Yes
	Collaboration	Stakeholder Participation	Yes

Software engineering and systems engineering are two closely related disciplines that share common goals, such as delivering high-quality solutions through structured processes, but they differ significantly in scope, focus, and application. Software engineering is a specialized field that concentrates on the design, development, testing, and maintenance of software systems. It employs methodologies such as agile development, DevOps, and iterative prototyping to ensure that software products are functional, reliable, and scalable. Tools like version control systems, integrated development environments (IDEs), and automated testing frameworks are central to software engineering practices. The primary focus is on writing efficient code, optimizing algorithms, and ensuring a seamless user experience, often within the constraints of time and budget [5, 15].

In contrast, systems engineering takes a broader, interdisciplinary approach, addressing the design, integration, and management of complex systems that may include hardware, software, processes, and human elements. Systems engineers focus on the entire system lifecycle, from conceptual design to deployment and maintenance, ensuring that all components work together cohesively to meet stakeholder requirements. Techniques such as systems modeling, trade-off analysis, and risk management are integral to systems engineering. Tools like SysML (Systems Modeling Language) and MBSE (Model-Based Systems Engineering) are often used to analyze and optimize system interactions [1, 16]. The discipline emphasizes a holistic perspective, balancing performance, cost, schedule, and risk across the system.

While software engineering can be viewed as a subset of systems engineering, their methods and tools differ significantly. For example, in a spacecraft project, software engineers develop flight control software, while systems engineers ensure seamless integration of that software with hardware,

sensors, and communication systems to achieve mission objectives [17]. This distinction highlights the complementary nature of both fields, where collaboration between software and systems engineers is essential for delivering complex, integrated solutions. The difference can be seen in Table 2.

Table 2. *Difference of System Engineering and Software Engineering.*

Aspects	Software Engineering	System Engineering	Similarity
Key Concern	Software Development	Meet Stakeholder Requirement	No
Methodology	Agile Development	Integration Components	No
Tools	UML	SysML	Yes
	IDEs	MBSE	No
	Code	Management Complex System	No
Primary Focus	Algorithms	Balancing Performance	Yes
	User Experience	Cost/Risk/…	Yes

Given their similarities, it is natural to consider whether software engineers could successfully transition to systems engineering. However, several challenges hinder this transition, including knowledge gaps, mindset shifts, and a lack of formal training in systems engineering.

Gamification has emerged as a potential solution to facilitate this transition. Widely applied across various fields such as education, healthcare, tourism, and business [8-11], gamification refers to the integration of game design elements into non-game contexts to enhance user engagement, motivation, and performance [12]. In software engineering, gamification has been used to improve code quality and enhance development practices [13, 14].

Research on the application of gamification in systems engineering is still limited. Existing studies focus primarily on enhancing academic performance by setting goals and increasing engagement and learning in systems engineering projects [18, 19]. Despite these efforts, the application of gamification strategies to train systems engineers remains largely unexplored, even though gamification has proven effective in promoting skill development in software engineering [9].

Since software engineering and systems engineering share many similarities, it raises a critical research question: Can gamification help software engineers transition to systems engineering? To address this question, we propose developing a gamification-based framework that provides a detailed approach to applying gamification principles during the transition from software engineering to systems engineering.

3. S2S-G Model

Several frameworks exist within the realm of gamification, including 6D, MDA, and Octalysis [20, 21]. Additionally, [22] introduces a gamification framework tailored for the software engineering process. This paper aims to develop a Software-to-System Engineering Gamification Model (S2S-G Model) to facilitate the transition from software engineering to system engineering while providing detailed guidance for implementation. The S2S-G Model comprises four main components: Preparation, Platform, Gamification Design, and Development (See Figure 1).

The Preparation phase involves a series of four steps, culminating in the transition to the Platform stage. The first step requires identifying the key differences between system engineering and software engineering, such as variations in methodologies and processes. During this stage, operators also determine the appropriate gamification level based on the complexity of the task. The second step involves evaluating whether a given task is suitable for gamification. If the task is deemed unsuitable, the gamification process may be bypassed during the conversion. However, if the task is appropriate for gamification, the fourth step requires operators to define specific requirements. For instance, this may involve redirecting software engineers' focus from agile development practices to enhancing interactions between systems and engineers.

The Platform phase is centered on user analysis, metric selection, and persona creation. The first aspect of this phase involves classifying engineers into various levels and identifying their motivation types, such as encouraging better interactions or promoting the transition from using UML to SysML. Based on this user analysis, operators proceed to select suitable gamification metrics that align with both engineers' progress and the requirements of the task. The chosen metrics are then assigned to engineers at different levels, ensuring appropriate alignment with their skill sets and motivations. Subsequently, operators create personas to represent

various engineer profiles within the gamification environment, aiding in the personalization of the gamified experience.

The Gamification Design phase focuses on constructing a gamified environment and ensuring its effectiveness. Operators first gather suitable gamification features and theories, including points, leaderboards, guides, and rewards, derived from previous stages. These features are then integrated into a coherent storyline with clear objectives and guidance, providing engineers with a structured path to follow throughout their tasks. The storyline serves as a guiding framework to influence and direct engineers' behavior. Once the design is established, operators proceed to develop a prototype to test the effectiveness of the gamification environment. If the prototype meets the desired outcomes, full-scale platform development begins. If not, operators return to the design phase to refine the prototype and address any shortcomings.

The Development phase focuses on building the gamification platform and evaluating its overall effectiveness. Operators select appropriate development tools and incorporate the relevant metrics of transfer into the platform. Gamification features are embedded within the platform, such as awarding points for modeling real-world scenarios using SysML, to motivate and guide user behavior. Once the platform is operational, user performance is assessed through data analysis or survey feedback. If the gamification effect does not meet expectations, operators may revisit the prototype design phase for further improvements. Otherwise, the process is considered complete.

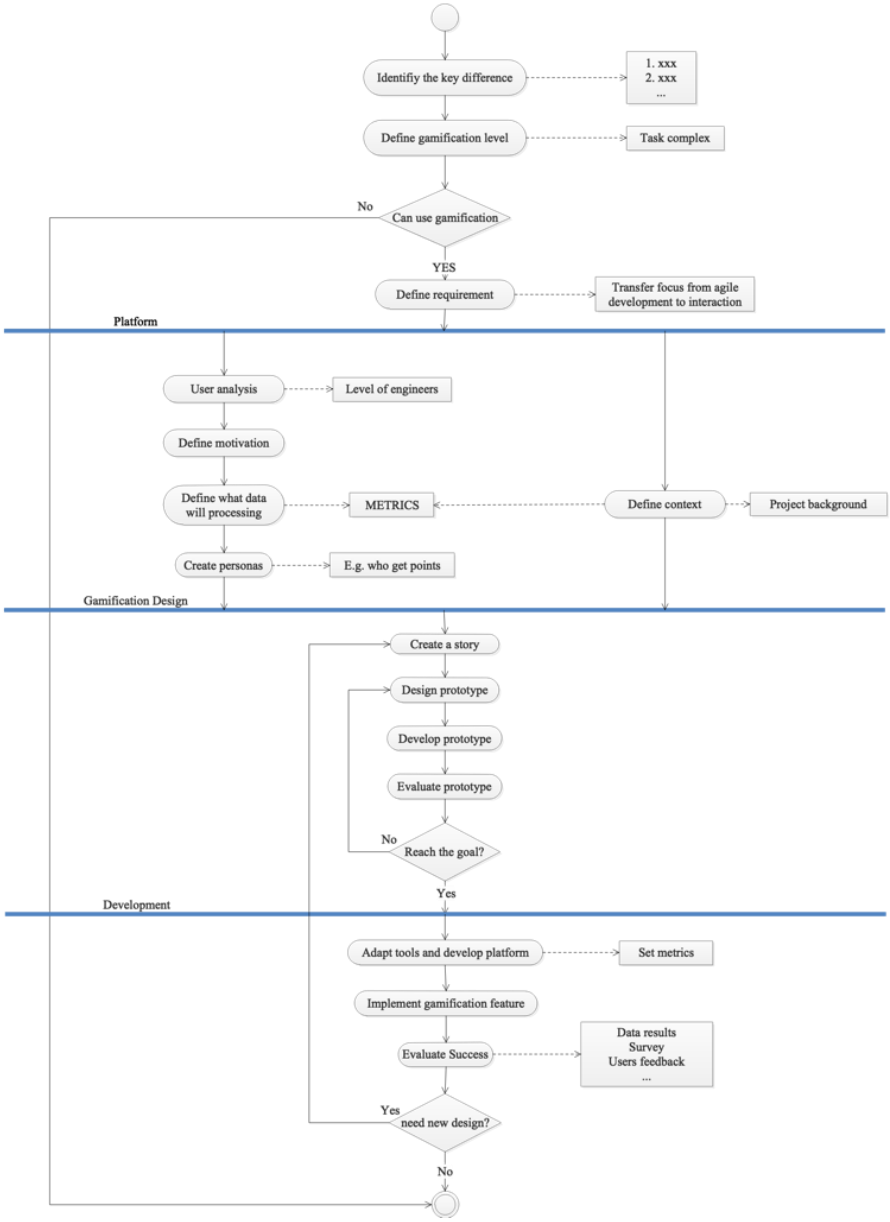

Figure 1. *S2S-G Framework.*

4. Conclusion

In conclusion, both software engineering and systems engineering are essential for creating complex solutions, though they differ in scope and focus. Software engineering focuses on the development, testing, and maintenance of software systems, while systems engineering takes a broader approach, integrating hardware, software, and processes throughout the system lifecycle. Despite these differences, the two disciplines share common goals of delivering high-quality, reliable solutions. Gamification, which incorporates game design elements into non-game contexts, has shown promise in improving engagement and performance in both fields. To bridge the gap between software and systems engineering, we propose the S2S-G Framework, a gamification-based approach designed to help transition software engineering practices into systems engineering. This framework identify how to apply gamification during the transition.

5. Future Work

Future work will involve conducting experiments to apply the S2S-G Framework in real-world settings, using empirical studies to evaluate its effectiveness in transitioning software engineering practices to systems engineering. The goal is to gather data on the gamification's impact on engagement, collaboration, and performance in system engineering projects. A key challenge will be defining the appropriate dependent and independent variables to assess the gamification strategies' success. Dependent variables could include measures of system integration, project completion time, and quality, while independent variables might encompass the gamification elements, team dynamics, and training methods employed. By conducting rigorous experiments, I aim to refine the framework and provide evidence of its practical benefits, offering a scalable solution for professionals in both fields. This research will contribute to understanding how gamification can bridge the gap between software and systems engineering, ultimately enhancing the effectiveness of engineering teams in complex projects.

Abbreviations

MBSE	Model-Based Systems Engineering
SysML	Systems Modeling Language
UML	Unified Modeling Language
S2S-G Framework	Software Engineering to System Engineering Gamification Framework

Author Contributions

Wei Ren: Conceptualization, Methodology, Investigation
Qin Bo: Validation, Project administration

Funding

This work is supported by CETC.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Hirshorn, S. R., Voss, L. D., & Bromley, L. K. (2017). NASA systems engineering handbook (No. HQ-E-DAA-TN38707).

[2] Riva, M., Zanutta, A., Genoni, M., Scalera, M. A., & Balestra, A. (2024, August). MBSE or no MBSE: is MBSE the final answer to system engineering?. In Modeling, Systems Engineering, and Project Management for Astronomy XI (Vol. 13099, pp. 115-120). SPIE.

[3] Aggarwal, K. K. (2005). Software engineering. New Age International.

[4] Kossiakoff, A., Sweet, W. N., Seymour, S. J., & Biemer, S. M. (2011). Systems engineering principles and practice (Vol. 83). John Wiley & Sons.

[5] Sommerville, I. (2011). Software engineering (ed.). America: Pearson Education Inc.

[6] Blanchard, B. S. (2004). System engineering management. John Wiley & Sons.

[7] Stuart Hallifax, Audrey Serna, Jean-Charles Marty, and Elise Lavou ́e. Adaptive gamification in education: A literature review of current trends and developments. In European conference on technology enhanced learning, pages 294–307. Springer, 2019.

[8] Juho Hamari, Jonna Koivisto, and Harri Sarsa. Does gamification work?–aliterature review of empirical studies on gamification. In 2014 47th Hawaii international conference on system sciences, pages 3025–3034. Ieee, 2014.

[9] Manuel Trinidad, Mercedes Ruiz, and Alejandro Calder ́on. A bibliometric analysis of gamification research. IEEE Access, 9: 46505–46544, 2021.

[10] Kai Huotari and Juho Hamari. A definition for gamification: anchoring gamification in the service marketing literature. Electronic Markets, 27(1): 21–31, 2017.

[11] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gaming contexts. In CHI'11 extended abstracts on human factors in computing systems, pages 2425–2428. 2011.

[12] Catalin-Virgil Briciu and Ioan Filip. Applying gamification for mindset changing in automotive software project management. Procedia-Social and Behavioral Sciences, 238: 267–276, 2018.

[13] Ren, W. (2023, December). Gamification in test-driven development practice. In Proceedings of the 2nd International Workshop on Gamification in Software Development, Verification, and Validation (pp. 38-46).

[14] IEEE Computer Society. (2020). *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Retrieved from https://www.computer.org

[15] INCOSE. (2023). *What is Systems Engineering?* Retrieved from https://www.incose.org/systems-engineering

[16] Blanchard, B. S., & Fabrycky, W. J. (2017). *Systems Engineering and Analysis* (5th Edition). Pearson Education.

[17] Huaman ́ı G. T., Rodriguez, L. A., & Alca, C. (2020, April). Agile method and implementation of gamification in an engineering course. In 2020 IEEE Global Engineering Education Conference (EDUCON) (pp. 1815-1818). IEEE.

[18] Tonkal, M., Wu, A., & Rogers, C. (2024, October). Exploring the Impact of Systems Engineering Projects on STEM Engagement and Learning. In 2024 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.

[19] Klock, A., & da Cunha, L. (2015). Gamification in e-learning systems: A conceptual model to engage students and its application in an adaptive e-learning system. In Learning and collaboration technologies (Vol. 9192, pp. 595–607). Berlin: Springer.

[20] Chou, Y.-K. (2015). Actionable gamification: Beyond points, badges, and leaderboards. Fremont: Octalysis Media.

[21] Ren, W., Barrett, S., & Das, S. (2020, January). Toward gamification to software engineering and contribution of software engineer. In Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences (pp. 1-5).

[22] Ankobiah, W. A. (2022). Integrating Model-Based Systems Engineering Industry Transformations for Workforce Development (Master's thesis, The University of Texas Rio Grande Valley).

[23] Friedenthal, S., Moore, A., & Steiner, R. (2014). A practical guide to SysML: the systems modeling language. Morgan Kaufmann.

[24] Fowler, M. (2018). UML distilled: a brief guide to the standard object modeling language. Addison-Wesley Professional.

[25] Rumpe, B. (2016). Modeling with UML (Vol. 98). Ber-lin/Heidelberg, Germany: Springer.