Human Factors in Software and Systems Engineering, Vol. 140, 2024, 59–66
https://doi.org/10.54941/ahfe1005038

A Definition of Inclusive Systems Engineering

Sandor Dalecke

AG Cyber-Physical Systems, RPTU Kaiserslautern, Kaiserslautern, 67663, Germany

ABSTRACT

This paper provides the first definition of inclusive systems engineering, as well as eight criteria we deem important for tools to be considered supporting inclusive systems engineering. Inclusive systems engineering focuses on domain experts by making systems engineering more approachable, reducing the need for communication between systems engineers and domain experts. To achieve this, we incorporate techniques of model-based systems engineering and agile software engineering approaches.

Keywords: MBSE, Inclusive systems engineering, Systems engineering, Systems modeling language

INTRODUCTION

With the need for more comprehensive and complex systems, which still need to be able to adapt to new constraints and technological advances, systems engineering has moved away from document driven development towards more interactive or model based development. Both of these approaches focus on minimizing the development of useless systems due to miscommunication (Madni, 2018; Karban, 2011). Agile approaches focus on continuous communication with stakeholders, whereas model based approaches focus on developing comprehensive models first, against which the developed system can be checked throughout the development. In order to include stakeholders and domain experts even more, we propose inclusive systems engineering, focusing on making the systems engineering process more approachable for domain experts. To do so we expand upon the current systems engineering definition given by INCOSE, as well as providing criteria on which a systems engineering tool can be checked against to identify if it can be considered to support inclusive systems engineering and in which areas it might be lacking.

This paper gives a very brief overview of systems engineering, followed by a proposed definition for inclusive systems engineering. Afterwards, a list of nine criteria is given, as well as, a brief explanation why we consider each criteria to be important for inclusive systems engineering and the anticipated benefit.

State of Systems Engineering

INCOSE defines systems engineering as *"[...] a transdisciplinary and integrative approach to enable the successful realization, use, and retirement* *of engineered systems, using systems principles and concepts, and scientific, technological and management methods"* (Incose, 2017).

Systems engineering has changed a lot in the last twenty years. Dealing with ever more complex systems, the need to change functionality during development, adapt to new developments, policies or needs, as well as focusing more on continuous service systems results in new challenges for software development and systems engineering. Communication between stakeholders is often a bottleneck in these instances (Madni, 2018). This is especially true for more task specific, often highly complex, systems, tailored towards specific domains with unique requirements.

These requirements lead to incorporating agile approaches known from software development, as well as shifting the focus towards model-based systems engineering.

The following section will briefly discuss the intent of agile approaches in software development, as well as, model-based systems engineering.

Agile Approach

In software development, the agile approach is nowadays an umbrella-term describing a number of different practices moving away from a project based development towards an iterative development method (Fowler, 2001), focusing on small finished and deployed functionality leading to larger more complex systems in smaller steps, instead of developing one, large system at once. This approach has been incorporated by more and more companies and scaled up to work for larger projects and development teams despite being firstly intended for smaller groups (Uludağ, 2022). One key aspect of agile software development is focusing on continuous communication between teams and stakeholders to make sure the tasks are well understood and technologically feasible. Additionally, this enables easier incorporation of new or changed functionality. However, this makes it difficult to maintain up-to-date documentation and systems models, which are especially important for larger systems which include a multitude of hardware and software components (Kasauli, 2021).

Model-Based Systems Engineering

Model-based systems engineering(MBSE) on the other hand focuses especially on the use of models during the systems engineering process. According to INCOSE MBSE is the formalized application of modeling to support systems requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases (INCOSE SE Vision, 2007).

MBSE also intends to reduce wasted time and money by implementing wrong or useless features (Madni, 2018). Instead of focusing on an iterative design to move away from the document based approach, which is especially hard to manage with the ever increasing scale and complexity of systems, the use of a comprehensive model is suggested. The model-based approach should create clear, connected models which can be shared among shareholders, achieving better communication and reducing the risk of useless features due to miscommunication. This model needs to be updated in case of major changes, but is intended to be checked against throughout the whole engineering process.

However, large-scale models are hard to create perfectly, needing to be revisioned, which is time-consuming and can have large repercussions (Bucchiarone, 2020; Broy, 2019). In addition, policy changes or technological advances can affect the validity of a model throughout the development.

Furthermore, MBSE is still a rather new approach, lacking a wide-spread standardized methodology. Currently the OMG works to resolve this issue by developing the Kernal Modeling Language (KerML) to provide a very basic, easily extendable language definition as well as a more sophisticated language, with many needed extensions already provided with the Systems Modeling Language version 2 (OMG, 2017).

Inclusive Systems Engineering

In order to merge the MBSE and agile approach we propose inclusive systems engineering (ISE) by extending the systems engineering definition by INCOSE.

Inclusive *Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological and management methods* ***with the specific focus on domain experts. User guidance techniques, intuitive syntax and interactive approaches are used to make the systems engineering process as approachable as possible for non-systems engineering experts.***

While not directly referring to the use of models in the definition, it is highly recommended, being easier to communicate and understand for domain experts, making models an obvious, but not necessarily exclusive, starting point for inclusive systems engineering.

The INCOSE definition of Systems Engineering considers a transdisciplinary approach to systems engineering. This is obviously true, as systems for countless domains are defined and built. However, we propose to broaden this transdisciplinary aspect to make systems engineering more accessible to domain experts, shifting the focus from transdisciplinary systems towards transdisciplinary systems engineers. We consider this approach to be inclusive systems engineering, not to be confused with systems engineering of inclusive systems.

ISE is supposed to be an extension of systems engineering, providing guidelines and suggestions to make systems engineering a more inclusive, more approachable process. Additionally, tools following the ISE approach can be used as a means of guiding domain experts towards classical systems engineering tools which provide additional functionality, or make tools more approachable, allowing the users to choose when to use more complex features themselves.

Metric to Determine Inclusive Systems Engineering

In order to evaluate existing and newly developed software engineering tools if they can be considered to support ISE we propose eight criteria to be checked. Table 1 lists these broad criteria and includes a suggestion how to achieve the criteria and a list of anticipated benefits. Following Table 1 is a brief discussion of each criteria.

Table 1. Criteria list for inclusive systems engineering.

Criteria	Suggestion	Benefit
Simple Syntax	Limited number of keywords	Reduce learning time Low entry hurdle
Tool Independence/ Compatibility	Standardised SE basis	Communication and Cooperation across domains and tools
Knowledge Basis	Existing Knowledge Basis	Enable User Guidance Techniques Automatic Code Generation
Version Management	Version Control / Commit System	Sharing of Models Support Agile Methods
Code Generation	Language Model for code generation	Enable domain experts to create code Create complex systems from informal description
Extendibility	Basic solution Provide means for extension	Extensions specifically designed Common basis
User Guidance	Recommender systems Nudging Feedback	Lower entry Domain expert focus
Easy Documentation	Markdown documentation Enable picture documentation	More comprehensive documentation Easier knowledge transfer

Please note, that these criteria are research topics in their own right, rendering the following descriptions necessarily superficial.

Simple Syntax

Systems engineering burrows from highly specific concepts well known from programming languages. However, it's not safe to assume domain experts from other domains are familiar with these concepts. A simple syntax is important to prevent confusion or overwhelming complexity.

Suggested approach: Burrowing from natural language in order to specify common relationships lowers the threshold for domain experts while still providing tools to define many aspects of systems. More intricate relationships can be used after domain experts become familiar with the systems engineering process. Limiting the number of keywords follows the same reasoning, making sure new systems engineers are not hindered by an extensive number of keywords which can be confused.

Anticipated benefits: A reduction in training time for a new tool can be anticipated, as well as functioning as a low entry point, offering more complex features at the users own pace.

Tool Independence / Compatibility

Systems Engineering isn't a tool specific process in of itself. It can be facilitated greatly by using tools, however, which specific tool to use should be ea choice left to the user. Different domain experts will likely be familiar with different tools and will prefer similar tools to work with to build a system. Keeping the systems engineering process tool independent is important to allow for different tools most suited for a specific domain. Preferably, an inclusive systems engineering process would be compatible with many different tools and providing the ability to switch between multiple tools, as long as no sole tool is suitable.

Suggested approach: A standardised systems engineering process or specified language as proposed with SysMLv2 for MBSE (OMG, 2017).

Anticipated benefit: A common standard is important to create compatible tools and systems in a wide variety of domains, facilitating cooperation and communication between domains.

Knowledge Base

A suitable knowledge base can store similar systems or parts thereof to be reused or be used as a reference. Additionally, knowledge bases can be used to communicate knowledge between experts of different domains. Furthermore, depending on the tool the systems themselves can be translated into knowledge bases for future reference.

Suggested approach: Depending on the size and complexity of systems different knowledge bases can be chosen, for large and highly complex systems the web ontology language (Guo, 2004; Wawrzik, 2022) can be explored.

Anticipated benefits: The most important benefit of a knowledge base is facilitating other criteria of this list. Some user guidance techniques make use of knowledge basis, enabling recommender systems and functioning as example systems to faster and easier systems engineering. Furthermore, knowledge bases can be used to be included in parts, reducing the need to redefine common system parts.

Version Management

Version management is important when features are developed by distributed groups and is important when sharing or iterating upon existing systems or models.

Suggested approach: Many possible version management systems exist, however, depending on the tool and intended use a version management system can be integrated into a tool, making it easier to use.

Anticipated benefits: Version management is important to enable incomplete model and iterative development. Being able to communicate different versions, integrating them and checking change history is important when several groups work towards larger systems, integrating new parts iteratively.

Code Generation

Automated code generation has made impressive progress in the last couple of years. By using informal description formalized code can be generated, representing the specified system as an implementation that can be checked for inconsistencies and errors. This is especially helpful for domain experts with no prior knowledge in programming that are tasked to implement a system formally.

Suggested approach: Recently, machine learning has been used to create large models to automate complex programming tasks (Dehaerne, 2022). These models use natural language descriptions to create code effectively.

Anticipated benefits: Using an effective automated code generation enables domain experts with no prior programming knowledge to create complex code. This gives domain experts tools a tool to better communicate or even implement the systems they require.

Extendibility

As systems engineering is becoming more important in a multitude of domains it's impossible to provide a solution suited for everyone. Therefore, it is important to provide a solution that can easily be extended to fit special needs in specific domains.

Suggested approach: Providing a simple solution to be a basis with the stated intension of becoming extended to suit different needs and provide tools to create these needed extensions.

Anticipated benefits: An underlying standard can be extended, facilitating the creation of specified extensions which are still operating on a common basis to be able to communicate and cooperate across extensions.

User Guidance

User guidance can encompass many aspects. During the modeling process the user can be assisted, using techniques such as nudging, persuasive systems design and recommender systems and automatic code generation. These techniques facilitate the work with the tool. User guidance can also be implemented before and after systems definitions, providing helpful feedback, such as error messages with suggestions how to fix the error, comprehensive tutorials and documentation.

Suggested approach: Nudging, Persuasive systems design and recommender systems can all be incorporated into a systems engineering tool, making it more approachable for domain experts and new users in general (Dalecke, 2023).

Anticipated benefits: User guidance is used to make a systems engineering tool itself more approachable and guide users through the engineering process, providing feedback, suggested next steps and guide towards more comprehensive systems and documentation

Easy Documentation

The benefits of comprehensive documentation have been shown multiple times, however, it's still often something done insufficiently. Providing an easy method to document is therefore crucial. Preferably this can be paired with user guidance to motivate providing comprehensive documentation (Forward, 2002; Plösch, 2014).

Suggested approach: The use of markdown language is widespread and easy to use, enabling also the inclusion of images as documentation.

Anticipated benefits: Clear documentation can be used to communicate systems parts to stakeholders and to enable users to find existing system parts to include in their own, reducing the time spent to define common parts, which is time that can be spent upon defining unique parts of a system more comprehensively. Additionally, with the use of machine learning tools documentation, in combination with a knowledge base could be used to extract formal systems definitions.

CONCLUSION

The increasing complexity of systems, as well as the need to respond to new developments and policy has changed the systems engineering process significantly. Multiple methods aim to reduce the time wasted developing useless features by focusing on increased communication and involvement of stakeholders and domain experts. Our approach further focuses on involving domain experts potential as systems engineers by making the systems engineering process more approachable. We have provided a definition for inclusive systems engineering, as well as a list of criteria against which systems engineering tools can be checked to identify their inclusive systems engineering potential and with which lacking can be detected. Furthermore, we have given a brief discussion on each of these specific criteria, providing suggestions and expected benefits for the user, be it domain or systems engineering experts.

FUTURE WORK

Currently, this work is based on anecdotal evidence from personal discourse with industrial partners, as well as a small questionnaire conducted on students, giving some insight on the preferred criteria, resulting in the proposed inclusive systems engineering criteria. The work on SysMD as an inclusive systems engineering tool will be used to further evaluate the inclusive systems engineering approach. Furthermore, a more representative questionnaire is planned to be conducted with systems engineers in the automotive domain.

Additionally, the proposed metrics will be used to evaluate not only SysMD but also other commonly used systems engineering tools.

REFERENCES

Broy, M., 2006, May. Challenges in automotive software engineering. In *Proceedings of the 28th international conference on Software engineering* (pp. 33–42).

Bucchiarone, A., Cabot, J., Paige, R. F. and Pierantonio, A., 2020. Grand challenges in model-driven engineering: An analysis of the state of the research. *Software and Systems Modeling*, *19*, pp. 5–13.

Dalecke, S. 2023. 'A review of user guidance techniques to enable 'inclusive' systems engineering for domain experts. *AHFE international*. doi: https://doi.org/10.54941/ahfe1004318.

Dehaerne, E., Dey, B., Halder, S., De Gendt, S. and Meert, W., 2022. Code generation using machine learning: A systematic review. *Ieee Access*, *10*, pp. 82434–82455.

Fowler, M. and Highsmith, J., 2001. The agile manifesto. *Software development*, *9*(8), pp. 28–35.

Forward, A. and Lethbridge, T. C., 2002, November. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM symposium on Document engineering* (pp. 26–33).

Guo, Y., Pan, Z. and Heflin, J., 2004, November. An evaluation of knowledge base systems for large OWL datasets. In *International semantic web conference* (pp. 274–288). Berlin, Heidelberg: Springer Berlin Heidelberg.

Incose.org. (2017). Systems Engineering. [online] Available at: https://www.incose.org/systems-engineering.

INCOSE, S., 2007. Vision 2020 (INCOSE-TP-2004-004-02).

Karban, R., Weilkiens, T., Hauber, R., Zamparelli, M., Diekmann, R. and Hein, A., 2011. Cookbook for MBSE with SysML. *MBSE Initiative–SE2 Challenge Team*.

Kasauli, R., Knauss, E., Horkoff, J., Liebel, G. and de Oliveira Neto, F. G., 2021. Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software*, *172*, p. 110851.

Madni, A. M. and Sievers, M., 2018. Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering*, *21*(3), pp. 172–190.

Plösch, R., Dautovic, A. and Saft, M., 2014, October. The value of software documentation quality. In *2014 14th International Conference on Quality Software* (pp. 333–342). IEEE.

OMG.org (2017). SysMLv2 [online] Available at: https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2.

Uludağ, Ö., Philipp, P., Putta, A., Paasivaara, M., Lassenius, C. and Matthes, F., 2022. Revealing the state of the art of large-scale agile development research: A systematic mapping study. *Journal of Systems and Software*, *194*, p. 111473.

Wawrzik, F., 2022. *Knowledge Representation in Engineering 4.0* (Doctoral dissertation, TU Kaiserslautern).