TNO innovation for life

Vision and Outlook on Systems-of-Systems in the High-Tech Equipment Industry

TNO Public ⟩ TNO 2025 R11685

10 December 2025

ICT, Strategy & Policy
www.tno.nl
+31 88 866 50 00
info@tno.nl

TNO 2025 R11685 – 10 December 2025

Vision and Outlook on Systems-of-Systems in the High-Tech Equipment Industry

Author(s)	Johan Lukkien Jan Tretmans

Classification report	TNO Public
Title	TNO Public
Report text	TNO Public
Number of pages	24 (excl. front and back cover)
Number of appendices	0

Disclaimer
In the creation of this document, Microsoft 365 Copilot was used to improve the presentation in chapter 3. In all instances where Copilot was used, the authors thoroughly reviewed the results to ensure the end result is accurate and free from errors, thus allowing the authors to take responsibility for the content.

Acknowledgements
The research is supported by the Netherlands Organisation for Applied Scientific Research TNO.

Summary

This document outlines the TNO-ESI vision and strategic direction for research and development in the area of systems of systems, focused on the high-tech equipment industry.

A system-of-systems (SoS) is a large-scale, non-monolithic, distributed, heterogeneous system, built from multiple interacting constituent systems (CS), which are independently operating systems themselves. Generally, there is no single owner or responsible for the entire SoS, yet, by collaboration in an SoS, functionalities can be provided, and behavior may emerge that would not be possible in any of its CS's alone. The main characteristic of an SoS, as opposed to a classical system consisting of components, is the autonomy and independence of the CS's. [1]

SoS's enable new applications and open up opportunities, but they also generate new challenges. CS's are not developed together and are integrated in unforeseen ways. Emergent behavior, good or bad, cannot always be predicted because there is uncertainty about the global SoS state. This state involves the configuration of CS's, their status, and the quality and validity of the information processed by the CS's. This means that an SoS must deal with meta-information about the SoS and its CS's, and must reflect on its own operation. Development of an SoS is evolutionary, following the pace of CS's. Additionally, security, privacy, and confidentiality are complicated when each CS has its own policy with respect to data management.

There is not a single, established SoS approach or SoS Engineering (SoSE) methodology. Research on SoS and SoSE is highly context and application-dependent, and should be focused and directed towards a specific class of SoS, a particular domain, or even specific circumstances.

This report is about systems-of-systems challenges in the high-tech equipment (HTE) industry. It discusses the definition of SoS as given in the literature and identifies generic challenges. TNO-ESI and its partner organizations observe that HTE equipment is increasingly integrated in ways that are best understood by regarding the overall system as an SoS. Six SoS concerns for the HTE equipment industry were identified via interviews, and compared with three SoS roadmaps. First, the Systems Engineering Body of Knowledge (SEBoK[2]); second the Strategic Research and Innovation Agenda of the Electronics Components and Systems community[3]; and third, the roadmap of the Road2SoS european project[4].

Combing the HTE concerns with these roadmaps and the generic challenges, a vision is proposed on SoS from the perspective of the HTE industry. The vision entails that the HTE industry is transitioning from a monolithic, product-focused perspective, to seamless and intelligent integration of equipment at the system level, becoming part of an SoS. From this vision, ten research directions are identified that address technological advances as well as advances in methodology to make this happen.

Contents

1 Purpose and Scope of this Document

This document outlines the vision and strategic direction for research and development in the area of systems-of-systems (SoS), focused on the High-Tech Equipment industry. It addresses the evolving challenges and opportunities that arise when multiple independent systems must collaborate and function cohesively. The document serves as guidance for research on methodologies for designing, implementing, and managing SoS, in a way that maximizes value, enhances collaboration, and fosters innovation within the High-Tech Equipment industry ecosystem.

This document is based on the TNO-ESI vision on Systems Architecting (SA) and Systems Engineering (SE) as described in [5]. It can be regarded as a further analysis and detailing of the SoS description in that report. While traditional SE focuses on individual systems, SoS introduces additional complexity by requiring seamless integration and dynamic adaptability across multiple interconnected systems while still maintaining system autonomy. The principles and practices discussed here are meant to complement and extend SA/SE practices, enabling the industry to address the demands of a rapidly changing technological landscape.

The material in this document is based on interviews with companies that are active in the High-Tech Equipment industry discussing their priorities. This is combined with some of the extensive literature on the subject of systems-of-systems. The outcome is a vision on the subject along with a series of relevant questions and topics to work on.

This document is structured as follows. Section 2 introduces the concept of systems-of-systems. Generic challenges are described in Section 3. Section 4 introduces concerns of the High-Tech Equipment industry regarding systems-of-systems and compares these with directions in the Systems Engineering Body of Knowledge [2], the Electronic Components and Systems strategic research and innovation agenda [3], and the recommendations of the Road2SoS European project [4]. Section 5 then gives vision and outlook in the form of challenges and directions. It introduces architecture concepts and examines life cycle consequences. Section 6 gives recommended reading to get familiar with the subject. Section 7 concludes the report.

2 The Concept of System-of-Systems

2.1 Description

More and more systems are not operating in isolation but are connected to and interact with other systems. This leads to a collection of connected and interacting systems that can provide functionalities that the individual constituent systems cannot provide. Such a collection of systems is called a *system-of-systems*. This section introduces and highlights the concept of systems-of-systems.

A system-of-systems (SoS) is built from multiple interacting constituent systems (CS). The constituent systems are autonomous, and independently operating systems themselves. This is also the major characteristic. This differs from the traditional view of systems consisting of components that are fully dedicated to implement the required functionality of a single system. In an SoS there is no single owner or responsible entity for the entire system-of-systems but new functionality and behavior emerges in an SoS that would not be possible in any of the constituent systems by themselves. The key is collaboration.

Historically, developments in the area of systems-of-systems naturally followed the increase in system integration through integrating components. Increased connectivity among systems enabled collaboration, and led to new challenges and possibilities . Besides technological advances, challenges included dimensions of trust, management and governance because of subsystem autonomy. Also political and legal aspects became important. Methodologically, systems engineering methods needed to be extended [6], [7].

Current examples of systems-of-systems in the High-Tech Equipment industry are production lines where systems from different manufacturers cooperate; a hospital where equipment cooperates to provide the best patient care; a fleet of naval vessels that provide resources (sensors, weapon systems) to a single combat management systems while still preserving vessel autonomy. Further examples that are often mentioned include traffic management where various systems, including in-car systems, collaborate to optimize traffic flows and avoid accidents. Smart cities, smart buildings, public transport, surveillance, situation awareness – they all rely on the cooperation of independent systems [1]. These examples vary in how control is organized, ranging from centralized to fully distributed. This organization of control and subsequent governance is an important distinction between regular systems and SoS’s.

2.2 Definition

From ISO/IEC/IEEE 21839 (literally cited via [2]) the following definition of a system-of-systems is obtained.

> *“An SoS is a set of systems or system elements that interact to provide a unique capability that none of the constituent systems can accomplish on its own.*

Note: Systems elements can be necessary to facilitate the interaction of the constituent systems in the system of systems.

Constituent systems can be part of one or more SoS.
Note: Each constituent is a useful system by itself, having its own development, management goals and resources, but interacts within the SoS to provide the unique capability of the SoS.“

This definition sketches the context and idea of systems-of-systems. The property that the ‘unique capability’ cannot be achieved by one constituent system is true for any system built from components. However, the discriminating factor is that in an SoS the top-level components to be integrated are independent systems themselves. In addition to this definition, there are other defining properties aimed at identifying an SoS more sharply. In this way, SoS can be seen to pose new challenges and require new solution directions. The seminal paper by Maier[6] introduces two defining properties that a system must have to be called a system-of-systems. According to Maier (literal quote):

“A system-of-systems is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties:
1. *Independence of the Components: If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. That is, the components fulfill customer-operator purposes on their own.*
2. *Managerial Independence of the Components: The component systems not only can operate independently, they do operate independently. The component systems are separately acquired and integrated but maintain a continuing operational existence independent of the system-of-systems.”*

Since then, the term constituent system (CS) has become accepted instead of ‘component’. Besides these two defining properties, literature mentions properties that are typical for systems-of-systems.Table 1 gives the most common properties. Evolutionary development, as opposed to systematic design, refers to two aspects: first, the evolution of CS’s follows their own development according to their primary function. Second, the SoS evolves as result of the evolution of its CS’s. Geographic distribution is about the geographic extent of the SoS as, for example, in a power grid. The network as the linking pin emphasizes the logical nature (software defined) of the integration. Classifications or defining properties are relevant since they result in specific technical challenges and particular architectures. For example, geographic distribution poses restrictions on communication latency.

In most literature, these properties are regarded as binary properties – either present or not present. In practice, however, one may think about a range from 0 to 100% that expresses how much a considered system has this property. This is because these properties include a wide range of aspects that not all SoS’s have. The ‘score’ column in Table 1 suggests this: the ‘> 0’ indicates that the property is mandatory; empty cells indicate no restrictions.

As an example, a machine inside a production line might hand over part of its managerial scope to the owner of the production line. Similarly, a machine that serves two SoS’s has operational capacity for both which will be experienced by both SoS’s as operational independence. The important point is not whether an example SoS ticks all boxes but whether SoS concepts and techniques are applicable and useful.

Table 1: Defining properties and other common properties of Systems-of-Systems [6].

Property	***Short description***	***Score***
Managerial independence	Different stakeholders manage CS's, possibly with competing objectives	>0
Operational independence	A CS has a useful functionality independent from any SoS it is part of	>0
Evolutionary development	The SoS development is by continuous evolution with changes in functions and operation of CS's	
Geographical distribution	SoS's are assemblages of CS's distributed over a large geographic area	
Emergent behaviour	There are behaviour and results from the SoS that cannot be attributed to a single CS. This behaviour may be the goal but it may also be unintended and unexpected	>0
Heterogeneous	The SoS consists of CS's of all kinds and types	
Connected by a network	The network is the most important means of integration of CS's into an SoS	

The literature provides many dimensions for creating taxonomies[6], [7]. Classifications are useful when they limit the solution space and provide insights and technology to solve problems in this class. A common classification of SoS's is in the dimension of CS autonomy, which is also used in [5]:

- *directed SoS:* CS's are centrally managed and the SoS has a clear goal;
- *acknowledged SoS:* CS's contribute negotiated resources but retain individual governance;
- *collaborative SoS:* CS's interact and contribute resources voluntarily to the shared goal;
- *virtual SoS :* there is only emergence.

The production line example would be an acknowledged or directed SoS. An example of a collaborative SoS is a collection of collaborating vehicles avoiding a traffic accident.

3 Generic Challenges

The combination of absent central control, limited governance, and distributed responsibility in an SoS, together with the high autonomy and independent operation of the CS's, has significant implications for the design, validation, testing, deployment, and maintenance of SoS. These implications manifest at several interrelated dimensions as challenges. These challenges are discussed below, following the presentation in [1] closely.

Continuous Evolution and Runtime Reconfiguration
Challenge: Maintain service continuity while enabling online admission, reconfiguration, substitution, and removal of autonomous CS's without central coordination.
An SoS is inherently dynamic. Because CS's operate independently, each can be changed, started, stopped, removed, replaced, updated, or degraded without coordinating with a central authority. In typical application domains, continuous operation is mandated. The SoS must therefore adapt itself in two ways: discover and admit a substitute CS when a CS departs, and remove or disconnect a CS when its service quality falls below acceptable thresholds. This, even without central coordiantion.

Runtime Validation, Assurance, and Fault Handling
Challenge: Execute verification, validation, and diagnosis during live operation, minimizing interaction side effects and preserving service quality throughout reconfigurations.
Where reconfiguration occurs at runtime, validation and quality assurance must also be performed at runtime. This requires continuous monitoring of CS's and their quality of service, assessing new CS's for admission, and diagnosing faults as failures arise. Runtime testing introduces additional complexity: it can cause unintended side effects from interactions between the operational SoS and the system under test (e.g., testing an alarm should not inadvertently trigger a call for a security official). Therefore, runtime verification and validation must be engineered to preserve service continuity, manage interaction risks, and sustain the overall quality and reliability of the SoS, particularly during and immediately after reconfigurations.
Related to this, SoS evolution often proceeds in ways not anticipated by individual CS's. After a change, the affected CS's must be reassessed with respect to quality and interoperability with minimal disruption to normal operations. This demands that the runtime monitoring and testing techniques are capable of safeguarding performance while changes occur. In the event of failure, runtime diagnosis must localize and isolate faulty CS's swiftly to prevent error propagation and to enable timely recovery.

Emergent Behavior and Interoperability Constraints
Challenge: Achieve predictable behavior and seamless interoperability among heterogeneous CS's, despite unknown partners, syntactic/semantic differences, and evolving standards.
The collective behavior of CS's can yield emergent phenomena that are not predictable solely from the behaviors of individual CS's, thereby introducing uncertainty about the SoS' overall behavior. In addition, CS's are not explicitly designed to interoperate. When CS's must interact with systems not known in advance, syntactic and semantic differences frequently arise. Flexible interfaces, or specialized connectors/adapters, are required to bridge such differences. Designing CS's that can operate, adapt, and connect within the evolving context of an SoS is a sustained challenge. Standardization can mitigate compatibility risks; however,

standards themselves evolve, and different versions may coexist across CS's, complicating integration and compliance.

Global State Awareness and Meta-Reasoning

Challenge: Establish accurate, timely global state awareness and robust meta-information governance to manage uncertainty in configuration, system health, and data quality.

The absence of central control, dynamic reconfiguration and ongoing testing, make it difficult to determine the precise global state of the SoS at any given time. This includes knowing which CS's are present and available, which CS's are healthy and operating correctly, and whether the information processed by CS's is valid and reliable. Because unhealthy systems may produce unreliable outputs, the SoS must actively manage uncertainty arising from incomplete or evolving state knowledge.

To cope with the uncertainty about the global state, an SoS must use and reason with meta-information (information about data and configurations) alongside its primary operational data. This includes communicating, monitoring, and assessing: (i) its own configuration; (ii) the health of the SoS and of individual CS's; and (iii) the quality and reliability of information streams. In effect, the SoS must observe and reflect on its own operation [8]. For example, a road-traffic management SoS primarily monitors and controls traffic flow while detecting anomalies and unsafe conditions; in parallel, it monitors data quality, anomalies in data streams, and the state and configuration of constituent CS's and the SoS as a whole, so that quality issues and anomalies can be detected and addressed. These concerns intersect with the broader field of autonomous systems [8].

Security, Privacy, Confidentiality, and Trust

Challenge: Enforce consistent security and privacy across autonomous CS policies and resolve conflicts or inconsistencies in the data through principled trust and assessments based on history.

Security in SoS presents additional complexities. Autonomy implies that each CS enforces its own policies governing the sharing and protection of sensitive information and resources; no central authority uniformly regulates security. As a result, explicit policies and mechanisms are required to manage authorizations and to specify who may access which information under which conditions. Special care is necessary to avoid information leakage during reconfiguration and runtime testing.

Beyond access control, the distributed and heterogeneous nature of SoS raises fundamental questions of trust: which information sources are reliable, and under what circumstances? When multiple CS's produce inconsistent or contradictory information, the SoS must employ mechanisms to evaluate consistency, and credibility to determine which data can be trusted for decision-making.

Blurred Spatial and Temporal Boundaries for Engineering

Challenge: Manage lifecycle continuity and governance when system and environment boundaries and development phases are fluid.

Compared with traditional systems and development processes, SoS's exhibit softened boundaries both spatially and temporally. Spatially: the difference between a system and its environment becomes less clear because the membership and interfaces of the SoS evolve over time. Temporally: the conventional sequence of development phases—design, construction, validation, testing, deployment, operation, maintenance, and decommissioning—becomes fluid. After extended periods of continuous operation, while CS's are leaving, joining, and updating, a functionally similar yet structurally different SoS may have emerged, composed of new or significantly updated CS's but still performing the same mission.

In summary: The autonomy of CS's and the absence of centralized control in SoS's necessitate sustained attention to runtime reconfiguration, dynamic validation, interoperability and standards, global state awareness through meta-reasoning, and robust security and trust frameworks. These characteristics blur traditional boundaries and demand engineering approaches tailored to continuous evolution and uncertainty [1].

4 Systems-of-Systems in the High-Tech Equipment Industry

4.1 Trends at High-Tech Equipment Manufacturers

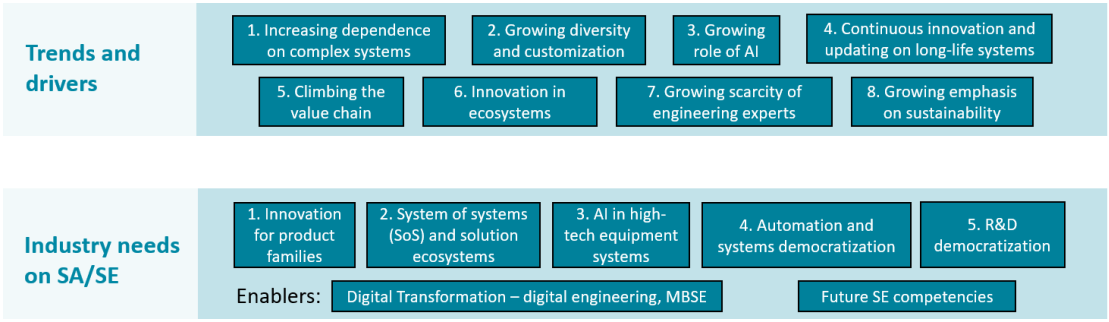

Figure 1: Overview of trends and drivers, and associated industry needs on System Architecture / Systems Engineering. Figure taken from [1].

TNO-ESI and partner organizations in the High-Tech Equipment (HTE) industry regularly discuss strategic business directions, opportunities, and challenges in the partner organizations as well as the key capabilities that are needed to address these challenges. This regular consultation of a broad range of leading industries in the Dutch high-tech equipment industry has brought a number of trends and challenges that this industry sector is facing, summarized for SA/SE (Systems Architecting / Systems Engineering) in Figure 1.

Industry need number 2 is systems-of-systems and solution ecosystems. A (technical) ecosystem is a network of interconnected, interdependent and complementary organizations that collaborate to achieve innovations. “Solution” refers to the capability to collectively solve challenges. In the HTE industry, the ecosystem builds systems that will be constituent systems when they become part of a system-of-systems (trend 6), for example, a production line or defense systems.

“Systems of systems” is therefore considered an important subject by HTE manufacturers although the meaning of the concept is not the same for everyone. High-Tech equipment includes products (i.e., machines like chips manufacturing machines, large printers, or MR machines, of, e.g., ASML, Canon or Philips) that used to be stand-alone and, more or less, dictate their environment. This is changing: the products must be integrated in the workflow of the customer and further be configurable to customer requirements (trend 2). This entails the behavior of the machine, largely determined by software, e.g., via software interfaces and supervisory control. It is also about parameter configuration, material flow and physical connections and dimensions. In this view, the focus lies with horizontal integration: the SoS in operation is the production line in which the individual, independent machines must fit as

constituent systems, next to each other, with no other hierarchy than implied by control. This renders this as an *acknowledged SoS*.

Following are SoS aspects and concerns mentioned by different HTE manufacturing organizations. They address parts of the challenges mentioned in Section 3, instantiated for the particular case of the HTE industry.

- **Integration in the context of the customer**. How to prepare these constituent systems for (horizontal) integration in an unknown environment? And the converse: how to prepare the CS's or the partial SoS for integrating third-party systems?
- **Customization, personalization**. Constituent systems must be configured according to customer rules, taking into account the (unknown) configurations of other CS's in the SoS, while companies do not want large numbers of different instantiations.
- **SoS optimization**. In an SoS like the one described above, it is challenging to optimize the workflow and the entire SoS, since the integration is late and the constituent systems are closed, in principle, except for their interfaces. These challenges extend to quality aspects, like dependability and timeliness, e.g., SoS performance issues, such as throughput problems, may emerge, if the throughput depends on timing aspects of multiple CS's that have been designed independently of each other.
- **Architecting for SoS**. During the architecting process of systems, the customization and integration (as constituent systems) in an SoS should be considered as a concern.
- **Life cycle aspects**. Testing, deployment, maintenance, upgrade in the field remain largely what they were for CS's; some preparation for functioning in an SoS context will have to be added into the design. For SoS applications, however, these life cycle aspects are entirely new. In particular, testing is a challenge, not only functionally but also to evaluate quality properties. Limited access to customer data, where now the 'customer' of one CS may include another unknown CS, further hinders this.
- **SoS as methodology for increased complexity**. Introducing the SoS perspective can be a systems engineering method to address complexity. Over the years, constituent systems incrementally increased their complexity (trend 4). This is simply the result of increasing functionality and quality requirements upon each innovation iteration. It can be beneficial to decompose a constituent system in separate systems and use SoS integration methods, on aspects where these methods are successful. In this way, high levels of decoupling and independence can be achieved for more regular systems.

In the next sections, these concerns of HTE SoS's, which typically fall in the classes of *directed* or a*cknowledged* SoS's, are compared with three relevant roadmap documents.

4.2 Systems Engineering Body of Knowledge

In the Systems Engineering Body of Knowledge (SEBoK[2]), the chapter on Systems-of-Systems formulates seven challenges. The concerns expressed by the High-Tech industry can be mapped as specific cases to these challenges, see Table 2. As follows from the discussion above, "leadership" is less a concern for the HTE SoS's. This is because SoS's in the HTE domains are mainly directed or acknowledged SoS's.

Table 2: Mapping of the HTE concerns to the challenges described in the guide to the SEBoK

SEBoK challenge	Short description	HTE challenge mapping
SoS Authorities.	Multiple authorities,agreement on common purpose, negotiate.	Life cycle aspects: access to data and internal operation by owner and manufacturer
Leadership.	In the absence of central control: influence, build incentives.	
Constituent Systems' Perspectives.	CS to operate in context it was not designed for, Data and service semantics in new context.	Integration in the context of the customer; customization, personalization; architecting for SoS.
Capabilities and Requirements.	Requirements of SoS may differ or need more than CS can give.	Integration in the context of the customer.
Autonomy, Interdependencies and Emergence.	Changes in CS because of autonomy and interdependencies lead to unpredictable emergent behaviour.	Life cycle aspects: decoupled DevOps of different CSs'; SoS optimization.
Testing, Validation, and Learning.	End-to-end testing (functionality and qualities) different from traditional and difficult;	Integration in the context of the customer; SoS optimization; Life cycle aspects: end-to-end V&V
SoS Principles.	Relatively recent subject for systems thinking, examples needed and methodology, steep learning curve and lack of sharing experience.	SoS as methodology for increased complexity.

Mapping HTE challenges to this roadmap shows that HTE challenges are recognized and part of current research in SoS. In addition, it makes the roadmap more concrete for HTE. Since there is an enormous body of literature on SoS at a high abstraction level, this more focused discussion is complementary. As expressed in [9]: “it is generally agreed that it is necessary to tailor SoSE (SoS Engineering) approaches to specific circumstances”.

4.3 ECS Strategic Research and Innovation Agenda

The SEBoK focus is on consequences of the SoS structure, and on questions that result from that. Another important document is the SRIA (Strategic Research and Innovation Agenda) of ECS (Electronics Components and Systems [3]). The SRIA is more focused on life cycle aspects, specific for SoS’s. The agenda names 5 major challenges on which HTE concerns can be mapped as illustrated in Table 3.

.

Table 3: Mapping of the HTE concerns to challenges described in the ECS SRIA.

ECS SRIA major challenge (innovation direction)	***Short description***	***HTE challenge mapping***
Open SoS architecture and infrastructure.	Encompassing the multidimensional, multi-stakeholder, multi-technology and evolutionary nature of large SoS's with key aspects along the full life cycle regarding e.g. safety, security, scalability etc.	Architecting for SoS
SoS interoperability.	Enabling instant and seamless understanding of information exchanged within and between distributed SoS's.	Integration in the context of the customer
SoS evolvability.	To control the uncertainty of emergent properties, functionalities and behaviours generated from the evolution and composition of constituent sytstems in a SoS.	Life cycle aspects: SoS testing, CS updates, maintain qualities over evolution
SoS integration.	Integration along the life cycle with integration and engineering methodologies, tools, toolchains, and tool interoperability for the implementation of SoS solutions using SoS architectures and platforms, with full lifecycle support.	Integration in the context of the customer
SoS monitoring and management.	Address SoS management from a number of perspectives, e.g. functional, security, safety, maintenance, SW updates, real-time behavior, and evolution.	SoS optimization; integration in the context of the customer; life cycle aspects: maintain qualities over evolution

4.4 EU FP7 project Road2SoS

EU FP7 project Road2SoS[4] was established to develop a roadmap and recommendations for future SoS deployment. The project identifies two trends: increasing numbers of IT systems ranging from the very small to the very large, partly dispersed in our environment, and increased connectivity of such systems; together these lead to Systems of Systems applications. Required technology advances for SoS's are discussed as a scheme and applied to four use cases: integrated multi-site industrial production; distributed energy generation and smart grids; multi-modal traffic control; and emergency and crisis management. In terms of this report, HTE could be another use case. The report contains comparisons between classic systems and SoS's and between classic Systems Engineering and SoS Engineering. The recommendations and mapping to HTE priorities are found in Table 4.

Table 4: Recommendations from the Road2SoS project regarding a roadmap for strategic actions for future SoS deployment

Road2SoS recommendations	***HTE challenge mapping***	***Rationale***
Establishment of an SoS Engineering discipline, offering generic methods and tools (i.e. domain-unspecific but generally applicable) along the entire life-cycle of an SoS	Architecting for SoS, Life cycle aspects	SoSE (SoS engineering) must develop from SE (Systems Engineering) specifically for the HTE domain, including tools
Establishment of SoS demonstrators	Use cases taken from HTE	Speaks for itself
Support standardization / harmonization of existing standards	Customization, personalization, Life cycle aspects: platforms, qualities	Standardization of interfaces and platforms are key to establish required interoperability to support customization
Development of SoS-enabled business models	SoS as methodology for increased complexity	This SoS methodology point of view is assumed to support and improve the business model by addressing the complexity
Advancement of technologies for real-time, low-latency communication	Architecting for SoS; Life cycle aspects: interfaces	This is somewhat specific, but in general, architectures have to support quality aspects like timing and resource use
Research on interoperability mechanisms	Life cycle aspects: interfaces, maintain qualities over evolution	Predictable interoperability, including qualities are a key aspect for HTE
Advancement of smart sensor technologies, sensor data fusion approaches, and efficient handling of big data	Integration in the context of the customer; life cycle aspects: data-based diagnostics	Data, and data management must be adapted to the situation that data is not avaliable to the manufacturer, while performance data is required for analysis and diagnosis
Advancement of autonomous decision-making capabilities	SoS optimization; integration in the context of the customer	This is less an explicit concern formulated by HTE, mainly because these SOS's are directed or acknowledged
Understanding of emergence	Life cycle aspects: verification, diagnostics; integration in the context of the customer	Correct behavior emerges as a result of integration in a larger, un-known whole. Understanding this is important in the life cycle

5 Vision and Outlook

5.1 Introduction

This section introduces a vision on SoS from the perspective of the HTE industry. From the vision and Section 3, challenging aspects are identified that must be addressed in order to realize the vision. A short-term agenda in the form of research questions and engineering goals is derived from this, and is presented as a prioritized list.

5.2 Vision

The HTE industry moves from a monolithic, product-focused perspective, to seamless and intelligent integration of equipment at the system level, becoming part of an SoS. Systems are modular and operate in new environments and are used in ways unknown at development time. Systems will be able to learn (from) their environment, and adapt and optimize their behavior to operate in the larger context of the SoS. Qualities of the SoS like safety, robustness and timeliness can be predicted and guaranteed. Manual work in this integration steadily decreases.

Zooming out, the HTE industry is at the heart of global innovations, as enabler of break-throughs. Key technologies and continuously expanding requirements drive future development and innovations in HTE. Key technologies include, for example, technological developments in AI, data and semantics, cloud, cyber security, trust, MBSE, and Digital Twins. These, mainly digital, and software intensive technologies are already intertwined among themselves. In addition, powerful systems engineering methodologies that harness resulting complexity are developed and adopted, such that the SoS system integration concepts can be applied recursively.

5.3 Challenging technological aspects

In the following, challenging aspects are introduced that derive from the particular properties and requirements of SoS's for the HTE industry as expressed in the above vision and Section 3. Research on these aspects is required.

Architectures and effective strategies for SoS development and evolution.
In an SoS, there are always two levels to consider: the CS level and the SoS level. CS-level architecting focuses on architecting for integration and collaboration with other systems. This means that concerns of future integration into an SoS shall be addressed in the architecture design of the CS (that is, of newly developed CS's), even when that SoS context is not known. For CS's that were not designed for arbitrary integration, proper connection technology needs to be developed leading to extra "system elements" (Section 2.2), that facilitate integration. The challenge amounts to developing architectures, architectural principles [6] and technology that support this very flexible system integration, that goes beyond simple interoperability by providing guarantees for the resulting SoS.

SoS-level architecting is about designing the application on top of CS's. For an acknowledged or directed SoS, this entails the logic to manage the entire SoS-level application(s). Challenges derive from the question on how to define and deploy SoS

applications, including aspects of resource management, behavior specification, safe and secure operations and predictability.

SoS life cycle concerns: verification, testing, deployment, and upgrading.
Activities in the life cycle are addressed differently by SoS's than in regular systems. This is rooted in SoS-defining properties, like partial ownership, dynamic configuration and very late binding, even after deployment. Eventually, SoS life cycle aspects are projected on the CS's. Quite a few challenges derive from the perspective of (integration) testing since integration can be at run-time and the tester of a CS has no control over the other CS's. In addition, these other CS's are not available at development time. Reflective technologies (self-* properties, [8]), in which CS's observe themselves are required.

For deployment of SoS applications, the logical and physical conditions of operating several CS's in an SoS context have to be satisfied, via the CS's and/or in adaptation and coordination components. Evolving (upgrading) the SoS application is a challenge as it typically requires upgrading CS's and support components. This upgrading is evolutionary, it follows the life cycle of the CS's and not of the SoS. (Re)deploying or updating a CS might lead to different (emergent) behavior of the SoS and this effect of a CS on the SoS needs to be carefully examined. Practical problems include obtaining access to deployed CS's and becoming authorized to realize the change. A further challenge is to bound the impact and control the result of partial upgrades, since there is no guarantee that changes that require multiple CS's to update are indeed established together.

Integration technology, focusing on interoperability, interfaces, interface management, and data and its semantics.
Physical interfaces for communication and, in many cases, material flow, are the basis for connecting and integrating CS's and for basic interoperability. Physical aspects with physical impact are under direct control of the individual CS's. In fact, such physical impact is part of the original purpose of the CS. This can be seen in the examples of directed SoS's like a single vessel that is part of a fleet, or a machine that is part of a production line: physical interactions and operations are implemented by the CS's. Other forms of interaction between CS's, e.g., power connections and EMC (electro-magnetic compatibility), may also occur and be part of the "physical fitting" aspect of integration. Challenges include specifying the nature of the physical impact and interactions, how physical behaviors get integrated and what the effect is on SoS behavior. There is a strong methodological component required that is dependent on the SoS domain at hand.

Interoperability further entails the entire stack from physical elements to communication hardware and subsequent data exchange, to the meaning of such data. Naturally, the behavior of the SoS is determined by such communication through supervisory or other forms of control. Information about the behavior of a system at interfaces can be processed digitally, but then a challenge is to maintain consistency, to automate management, and to generate test cases and simulators. Interface Control Documents (ICD) are a vehicle for this. ICDs specify and model the behavior at interfaces precisely and as completely as necessary for successful interaction between CS's. They are maintained for the most important interfaces of a system and can therefore play an important role in SoS integration[10]. However, ICDs and their methodology have not yet been extended to support automated integration into an SoS.

Interfaces are crucial in SoS integration. Interface models should be rich, including aspects like physical dimensions and timing properties. Integration should also be interpreted broadly. For example, "fit in a physical space'' can be an aspect of integration. The integration might

not be in the context of the CS developer. The challenge remains how to specify interfaces such that integration can be largely automated and how to deal with the particular boundary conditions of SoS.

Emergence, autonomy, management, governance
Again, two views are relevant: the view of CS's and the SoS view. Deployment of single systems, as CS's are, follows their design and development phases, possibly taking integration into an SoS into account. Also upgrading and redeployment follow regular system engineering workflows.

Composing CS's into an SoS leads to emergent behavior that is sometimes aimed for but can also be an unwanted side effect. Emergent properties include, for example, collective functional properties, and quality properties like robustness, timeliness, and security. A significant challenge is that the aimed-for emergent behavior needs to be designed and absence of unwanted emergence needs to be proven. For example, guaranteeing aimed-for emergence implies careful management of resources, bringing some of these resources under governance of the SoS application. Partial solutions to this sharing may exist (like reservation schemes) but this is not solved. This "competition on control" or "competition on governance" is a typical consequence of the relation CS-SoS.

HTE Systems will generate a lot of data which can be used for a broad range of goals, including performance optimization, diagnosis, and training AI models. Management and governance of such data is a concern, for example when HTE manufacturers cannot access data in the equipment of their customers. Proper access to data can enhance the autonomy and agentic behavior that CS's already have. Again, reflective technologies in which systems observe themselves are applicable.

Methodology: Systems of Systems Engineering
For successful and systematic introduction of SoS's, Systems Engineering methodologies must be expanded, including associated tools. Literature introduces the concept of SoS Engineering (SoSE). Nielsen et al. [11] give an overview of SoSE approaches and results. They state that SoSE "deals with planning, analyzing, organizing and integrating the capabilities of a mix of existing and new systems into an SoS capability greater than the sum of the capabilities of the constituent parts". SoSE therefore entails engineering of the construction, optimization, and maintenance of systems-of-systems. The scope of SoSE is the life cycle of the SoS and possibly of CS's. SoSE must address the challenges as described above. The very nature of SoS's include a great deal of uncertainty, and SoSE should deal with this.

Leveraging MBSE (Model-Based Systems Engineering) to specify structure, behavior, and qualities at interfaces is a natural step in SoSE, e.g., using contract-based design [12]. Proper tools working at the right abstraction level and new methodologies are required for this.

5.4 Outlook and Research Directions

Since the concept of SoS's represents a renewed focus on integration and interoperability, it is important that fundamental and practical knowledge and expertise about integration and interoperability are developed, including methodology. Research should be exploratory and have a case-driven and scenario-central approach, like in the prioritized list below. Scenarios

aim at addressing the points in this list by research and prototyping, and by advancing SoSE in a particular field or domain.
Besides these technical challenges there are the social-technical aspects of SoS. These aspects concern, for example, users of the SoS and its CS's, the engineers that have to envisage alternate life cycles and the governance of the SoS. Social-technical aspects should be considered in all the technical challenges.

Following is a prioritized list of exploratory research activities relevant to HTE.

1. Develop SoSE methodology for effective integration in the specific context of HTE and develop management policies. This is an overarching point.
2. Take as a concern, while designing CS's and interfaces, how to enable access by third parties to resources and functions in the CS, how to facilitate predictions and guarantees of quality properties, and how to optimize integrated CS behavior in customer SoS context.
3. Investigate the utility of different architecture styles and interaction styles for CS's, support components and SoS integration. Architectural choices must not block integration.
4. Leveraging MBSE: MBSE is an important instrument for developing future SoS's since calculations, simulation and reasoning will be model based. Use AI for deriving such models, and ensure consistency among models. This ties in to the second point.
5. For integrating legacy systems, develop adapter technology and management strategies that particularly address SoS issues, like problems of governance, resource use and predictability.
6. Enable evolution of an SoS in the form of adaptivity and upgrades, while not breaking operations, functional results, or certification.
7. Identify security requirements and how they are addressed and interact among CS's.
8. Increase dependability by self-monitoring, such as responding to anomalies where actual behavior and expected behavior do not match.
9. Generate the data and representations required to achieve goals in this list. Use AI to support SoS operations and model building; reduce human intervention. Or use AI to fill in missing information due to limited access to CS's.
10. Create and use standards, reference architectures, interface definitions, for applying and maintaining developed insights.

6 Some Further Reading

The seminal paper by Maier [6] sets the stage by introducing the subject as well as terminology, some taxonomy and common concerns. The paper introduces architectural principles and gives examples on how to use these principles. The background of the author is in aerospace though the examples are not limited to that.

Cook and Pratt [7] are concerned with SoSE for the Australian defense force ADoD. The paper presents SoSE characteristics and classifications from the literature and a mapping from ADoD enterprise characteristics to implications for SoSE. It gives categories of success factors of SoSE. The 2021 paper [9] gives a state-of-the-art of SoSE with many examples from literature.

Ingram, Payne and Fitzgerald [13] discuss architectural patterns (like service oriented, publish and subscribe, and blackboard) in relation to SoS characteristics. They explain how properties of these patterns address concerns of SoSE.

Dahmann has published widely on SoSE, including introductory texts (with Henshaw)[14] and contributions to the SEBoK, looking at consequences of SoS on design and engineering [15] (with Baldwin).

Jamshidi [16] gives a control perspective on SoSE, considered to be one of the most challenging aspects. Jamshidi published a book [17] on the subject of SoS and several other works on the topic.

A more formal approach to modelling and analysis of SoS's is by DeLaurentis and others. The book [18] is an introduction.

Sauser and Boardman have several influential publications on SoS and SoSE. In [19], a comparison between systems and SoS is given along five lines: Autonomy, Belonging, Connectivity, Diversity and Emergence. The work of Sauser has often a logistics and supply chain perspective [20].

Adams and Keating [21] give requirements for SoSE methodology from a holistic perspective, like transportable (reusable in a different context), adaptable and rigorous a.o. They give seven perspectives that represent steps to define and apply an SoSE method. Keating and Katina [22] give an extensive view on the development of the field of SoSE.

Nielsen et al. [11] give a survey of formal modelling and model-based engineering in SoS. The paper gives an overview of challenges in architecture, simulation, testing and verification from several dimensions: autonomy, independence, distribution, evolution, dynamic reconfiguration, emergence of behavior, interdependence and interoperability. It indicates how these dimensions can be strengthened. It has an extensive list of references and can serve as a starting point for studying SoS.

Manzano et al. [23] describe models and a simulation-based method to predict whether an SoS that is dynamically formed can sustain its operations.

7 Conclusion

Systems of Systems as a concept has become widely accepted as a means to describe a particular type of system integration. From this, new engineering concepts were developed, collectively referred to as SoS Engineering (SoSE). Both researchers and practitioners start to recognize that an SoS viewpoint is relevant and helpful, although adoption is sometimes found to be slow [24].

There is not a single, established SoS approach or SoSE methodology. Research on SoS and SoSE is highly context- and application-dependent, therefore generic research is too broad and abstract, potentially leading to results that are not directly applicable. SoS research should be focused and directed towards a specific class of SoS's and a particular domain. This can be done, for example, by following the taxonomy introduced in Section 2.2, references [5],[3],[1] or another SoS classification [9]. But no matter which classification, the core of SoS is about integration and interoperability, so research on these topics is needed. Other research topics are given in the (ordered) list in Section 5.4 on Research Directions.

Because of their interdependence and complex governance, SoS's have socio-technical concerns throughout their life cycles. While in systems engineering several socio-technical systems have to collaborate to realize the result [25] (e.g., the context system, the realization system, the deployed system), for SoS's this socio-technical impact is even more the case.

SoS's rely heavily on digital technologies. Advances in these technologies may solve some problems but will also enable new opportunities. For example, advances in AI can help in learning interfaces, in developing trust and in adapting to situations. But they also enable automated testing, self-monitoring and agency.

The analysis in Section 4 showed that HTE problems are consistent with research directions in influential research agendas. The adoption of SoS principles in the High-Tech Equipment industry is crucial for managing complexity, enhancing innovation, and ensuring long-term adaptability. It is the basis for a well-functioning solution ecosystem of companies that act as and deliver constituent systems.

8 References

[1] P. Van de Laar, J. Tretmans, and M. Borth, *Situation awareness with systems of systems*. Springer Science & Business Media, 2013. Accessed: Jun. 23, 2025. [Online]. Available: https://doi.org/10.1007/978-1-4614-6230-9

[2] “Systems of Systems (SoS) - SEBoK.” Accessed: Nov. 28, 2024. [Online]. Available: https://sebokwiki.org/wiki/Systems_of_Systems_(SoS)

[3] “ECS SRIA 2025 | ECS SRIA.” Accessed: Jun. 23, 2025. [Online]. Available: https://ecssria.eu/2025

[4] C. Albrecht *et al.*, *Roadmaps and Recommendations for Strategic Action in the field of Systems of Systems in Europe*. Steinbeis-Edition, 2015. Accessed: Feb. 24, 2026. [Online]. Available: https://www.steinbeis-edition.de/media/60/22/58/1725883338/179651_blick.pdf

[5] S. Acur and T. Hendriks, “Vision and Outlook for Systems Architecting and Systems Engineering in the High-Tech Equipment Industry,” TNO, TNO 2024 R10542, 2024.

[6] M. W. Maier, “Architecting principles for systems-of-systems,” *Syst. Eng.*, vol. 1, no. 4, pp. 267–284, 1998, doi: https://doi.org/10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.

[7] S. C. Cook and J. M. Pratt, “Towards designing innovative SoSE approaches for the Australian defence force,” in *2014 9th International Conference on System of Systems Engineering (SOSE)*, Glenelg, SA, Australia: IEEE, Jun. 2014, pp. 295–300. doi: 10.1109/SYSOSE.2014.6892504.

[8] A. Berns and S. Ghosh, “Dissecting Self-* Properties,” in *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, San Francisco, California, USA: IEEE, Sep. 2009, pp. 10–19. doi: 10.1109/SASO.2009.25.

[9] S. C. Cook and J. M. Pratt, “Advances in systems of systems engineering foundations and methodologies,” *Aust. J. Multi-Discip. Eng.*, vol. 17, no. 1, pp. 9–22, Jan. 2021, doi: 10.1080/14488388.2020.1809845.

[10] H. Cadavid, V. Andrikopoulos, and P. Avgeriou, “Documentation-as-Code for Interface Control Document Management in Systems of Systems: A Technical Action Research Study,” in *Software Architecture*, vol. 13444, I. Gerostathopoulos, G. Lewis, T. Batista, and T. Bureš, Eds., in Lecture Notes in Computer Science, vol. 13444. , Cham: Springer International Publishing, 2022, pp. 19–37. doi: 10.1007/978-3-031-16697-6_2.

[11] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, “Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions,” *ACM Comput. Surv.*, vol. 48, no. 2, pp. 1–41, Nov. 2015, doi: 10.1145/2794381.

[12] I. Stierand, “Contract-Based Design in Model-Based Systems Engineering,” in *7th Workshop on Software Engineering for Cyber-Physical Production Systems*, 2024.

[13] I. Claire, P. Richard, and F. John, “Architectural modelling patterns for systems of systems,” *INSIGHT*, vol. 19, no. 3, pp. 63–66, Oct. 2016, doi: 10.1002/inst.12112.

[14] J. Dahmann and M. Henshaw, “Introduction to systems of systems engineering,” *INSIGHT*, vol. 19, no. 3, pp. 12–16, Oct. 2016, doi: 10.1002/inst.12100.

[15] J. Dahmann and K. Baldwin, “Implications of systems of systems on system design and engineering,” in *2011 6th International Conference on System of Systems Engineering*, Albuquerque, NM, USA: IEEE, Jun. 2011, pp. 131–136. doi: 10.1109/SYSOSE.2011.5966586.

[16] M. Jamshidi, “From Large-Scale Systems to System of Systems – Control Challenges for the 21st Century1,” *IFAC Proc. Vol.*, vol. 43, no. 8, pp. 13–19, 2010, doi: 10.3182/20100712-3-FR-2020.00004.

[17] M. Jamshidi, *Systems of systems engineering: principles and applications*. CRC press, 2017. Accessed: Jan. 20, 2025. [Online]. Available: DOI https://doi.org/10.1201/9781420065893
[18] D. A. DeLaurentis, K. Moolchandani, and C. Guariniello, *System of systems modeling and analysis*. CRC Press, 2022. Accessed: Jun. 01, 2025. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.1201/9781003231011/system-systems-modeling-analysis-daniel-delaurentis-kushal-moolchandani-cesare-guariniello
[19] J. Boardman and B. Sauser, “System of Systems - the meaning of of,” in *2006 IEEE/SMC International Conference on System of Systems Engineering*, Los Angeles, California, USA: IEEE, 2006, pp. 118–123. doi: 10.1109/SYSOSE.2006.1652284.
[20] C. G. Kochan, D. R. Nowicki, B. Sauser, and W. S. Randall, “Impact of cloud-based information sharing on hospital supply chain performance: A system dynamics framework,” *Int. J. Prod. Econ.*, vol. 195, pp. 168–185, 2018.
[21] K. MacG. Adams and C. B. Keating, “Overview of the systems of systems engineering methodology,” *Int. J. Syst. Syst. Eng.*, vol. 2, no. 2/3, p. 112, 2011, doi: 10.1504/IJSSE.2011.040549.
[22] C. B. Keating and P. F. Katina, “Systems of systems engineering: prospects and challenges for the emerging field,” *Int. J. Syst. Syst. Eng.*, vol. 2, no. 2/3, p. 234, 2011, doi: 10.1504/IJSSE.2011.040556.
[23] W. Manzano, V. V. G. Neto, and E. Y. Nakagawa, “Simulation of systems-of-systems dynamic architectures,” in *Congresso Brasileiro de Software: Teoria e Prática (CBSoft)*, SBC, 2020, pp. 245–254. Accessed: Jul. 10, 2025. [Online]. Available: https://sol.sbc.org.br/index.php/cbsoft_estendido/article/view/14632
[24] H. Cadavid, V. Andrikopoulos, P. Avgeriou, and J. Klein, “A Survey on the Interplay between Software Engineering and Systems Engineering during SoS Architecting,” in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Bari Italy: ACM, Oct. 2020, pp. 1–11. doi: 10.1145/3382494.3410671.
[25] J. N. Martin, “3.1.2 The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems,” *INCOSE Int. Symp.*, vol. 14, no. 1, pp. 459–470, Jun. 2004, doi: 10.1002/j.2334-5837.2004.tb00509.x.

ICT, Strategy & Policy

High Tech Campus 25
5656 AE Eindhoven
www.tno.nl

TNO innovation for life