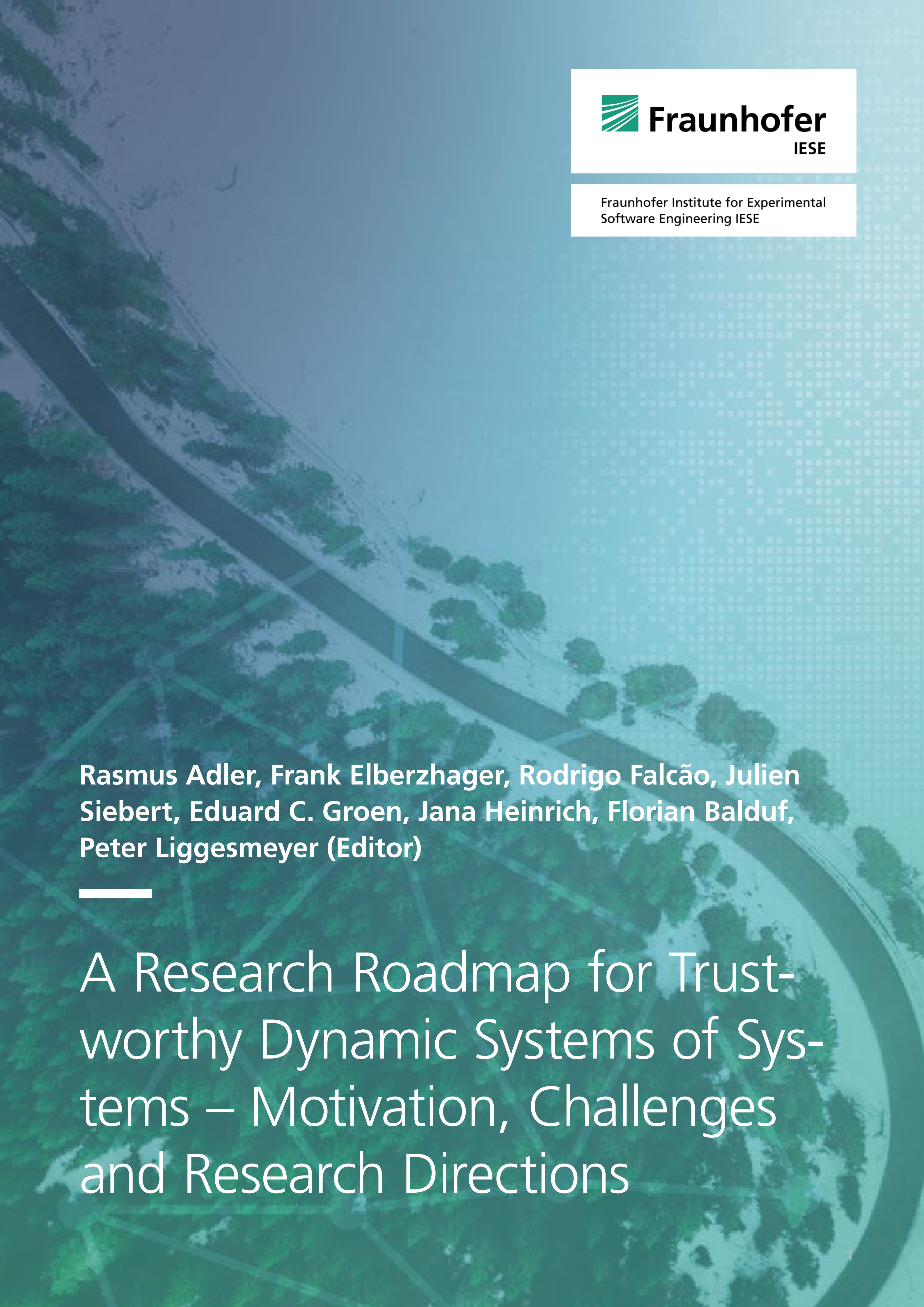


Fraunhofer Institute for Experimental Software Engineering IESE

Rasmus Adler, Frank Elberzhager, Rodrigo Falcão, Julien Siebert, Eduard C. Groen, Jana Heinrich, Florian Balduf, Peter Liggesmeyer (Editor)

A Research Roadmap for Trustworthy Dynamic Systems of Systems – Motivation, Challenges and Research Directions

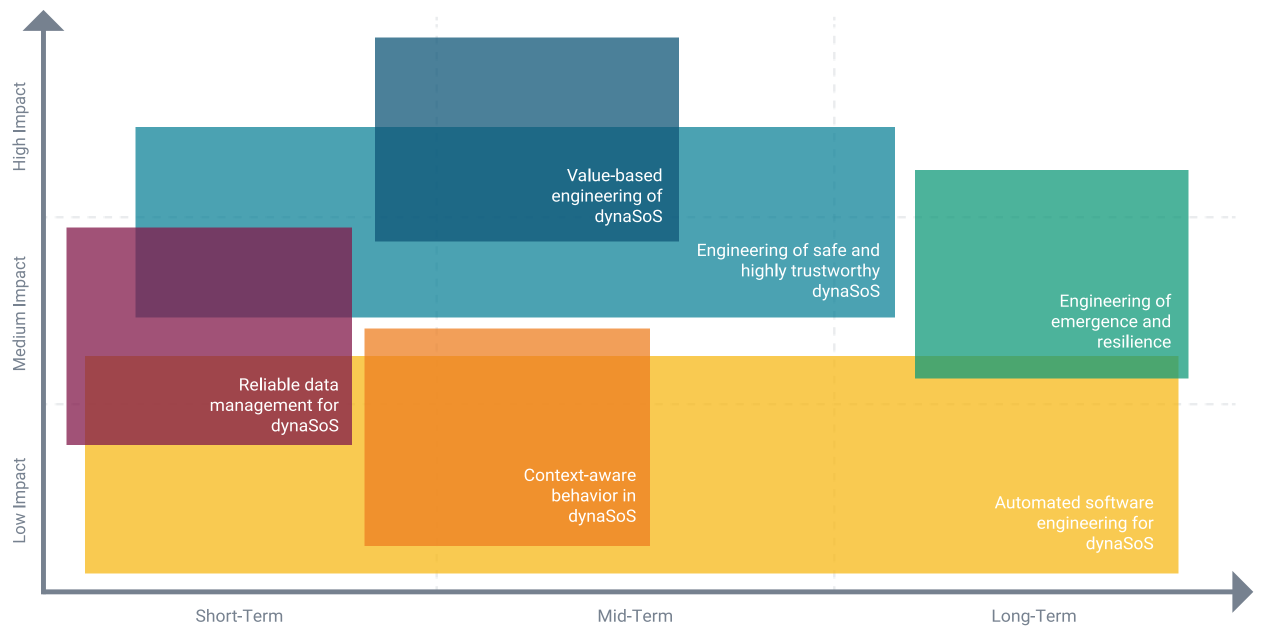
Executive Summary

This report summarizes the results of DynaSoS, a research project funded by the German Federal Ministry of Education and Research (grant number: 01IS21104). The main outcome is a research roadmap for the software engineering of trustworthy dynamic systems of systems (dynaSoS). The research roadmap for engineering dynaSoS was derived from interviews, expert workshops, and a systematic literature study, performed along the four phases of the project: Conceptualization, Characterization, Classification of challenges, and Recommendations.

In phase 1, use cases and example dynaSoS in several smart scenarios were identified. Different scenarios highlight different aspects of dynaSoS. Visions such as smart mobility, smart farming, or smart energy refer to dynamic systems of systems that dynamically adapt their behavior to the current situation in order to minimize required resources, reduce costs, improve delivered services, or provide novel services. We also considered cross-domain dynaSoS like smart cities that bring together sector-specific dynaSoS in certain area. Next, in phase 2, the use cases and example systems, together with the literature review, served as input for the identification of the core characteristics of dynaSoS.

In phase 3, related research challenges were collected and structured along three dimensions. The first dimension describes how large the dynaSoS is, whether it comprises other dynaSoS, and whether it belongs to a specific sector. The second dimension refers to established characteristics of systems of systems, such as the operational and managerial independence of the constituent systems in a system of

Figure 1: High-Level view of the roadmap towards trustworthy dynaSoS



systems. We added further characteristics to describe the role of Big Data, AI, and Autonomy in dynaSoS. The third dimension refers to engineering aspects such as activities, processes, or skills.

Finally, in phase 4, recommendations for further research were derived and structured along two dimensions. One dimension refers to the potential impact of the research direction. The other dimension indicates the time required to solve the challenge.

As shown in Figure 1, the roadmap proposes six main research topics that were derived during the course of the project:

1. **Reliable data management:** DynaSoS are data-intensive systems, where different organizations may need to share information and where value is created using data from various sources. Reliable data management is a prerequisite, which is hard to achieve in practice. »Data management« is an overarching term that covers multiple data-related aspects, such as data architecture, data acquisition, data storing, data quality, data integration, and data governance.
2. **Automated software engineering:** DynaSoS comprise technical systems that are continuously developed by different organizations. Unexpected evolution of some software systems is common in this setting. The evolution of the many non-technical systems is often also hard to predict. Handling these unexpected evolutions of systems before unwanted emergent phenomena occur increases the demand for automated software engineering. This demand is already very high and was highlighted by many system engineers from industry.
3. **Context-aware behavior:** DynaSoS behave in a situation-specific way, which requires some form of context awareness. Context awareness is the main challenge for many autonomous systems, such as autonomous vehicles. It requires the system to understand the current situation so that it can anticipate future scenarios. The dynaSoS has to understand the current situation so that it can anticipate phenomena that emerge from the behavior of its systems.
4. **Engineering of safe and highly trustworthy dynaSoS:** A dynaSoS typically provides essential services whose failure is highly critical. Assuring that these failures will not occur is challenging because the behavior emerges from complex interactions of evolving systems. Many novel safety approaches can contribute to dealing with this issue, but they are not sufficient, even if harmonized and integrated.
5. **Value-based engineering of dynaSoS:** A dynaSoS shall provide services so that it is in line with the current values of society. This requires that regulatory constraints and the economy support these values or even enforce them by prescribing high-level targets or principles. Value-based engineering refers to the challenge of getting from principles to practice. It is related to safety engineering because safety is a value. The difference is that it includes many values that are harder to grasp, such as sustainability, but where less rigor in assurance is demanded.
6. **Complexity, emergent phenomena, and resilience:** A dynaSoS is a complex system that can generate emergent phenomena. Resilience refers to the property that required emergent phenomena are provided in spite of disturbances and that disturbances will not lead to unwanted emergent phenomena or a collapse of the dynaSoS. Complexity science provides many theories and tools that need to be enhanced and integrated into software engineering for dynaSoS.

This list presents main software engineering research topics for dynaSoS. We expect these results to be used to direct investments and further research in the field of dynaSoS and to connect the multidisciplinary scientific community around the topic.

Table of contents

List of abbreviations

ADS	Agricultural Data Space
AI	Artificial Intelligence
CS	Constituent system
dynaSoS	Dynamic Systems of Systems (d lowercase)
DynaSoS	The DynaSoS project (D capitalized
IoT	Internet of Things
ML-OPS	Machine Learning Operations
SDG	Sustainability Development Goal
SE	Software Engineering
SoS	System of Systems
SoSE	System of Systems Engineering
S&SE	Systems and Software Engineering
VUCA	Volatility, uncertainty, complexit and ambiguity
UN	United Nations

Introduction

This chapter opens the report motivating the existence of dynamic systems of systems – dynaSoS - and provides an overview of the relationship between its characteristics, research challenges, and research recommendations.

The digital transformation is advancing across industries, enabling products, processes, and business models that radically change the way we communicate, interact, and live together. Systems, players, and markets that were once isolated are being integrated by digital ecosystems[1]. In this evolution, software-based systems are the central anchor through which value is added and desired system properties are realized.

Existing digital platforms and ecosystems already consist of complex networks of interconnected and globally distributed applications. Rapid advances in miniaturization, storage capacity, intelligent information exchange, and processing enable the digitalization of more and more diverse objects. All visions of the future of technology – be it production as a service in Industry 4.0, autonomous transport systems in intelligent mobility, or cyber-physical systems in digital health – point to the fact that the future of software engineering will have to deal with systems that are increasingly large and diverse, complex in scale and dynamics, and involve more and more actors from different organizations. In other words, the focus of software engineering will increasingly be on the design and development of **dyna**mic, semi-autonomous **S**ystems **o**f **S**ystems (**dynaSoS**).

Today, systems with characteristics similar to dynaSoS already pose technical challenges, particularly in terms of managing their quality and complexity, and these challenges will intensify with the technical advances of and the growing demands on these systems. Moreover, because of their scale (national, transnational, or even global), these socio-technical systems directly affect societies (think, for example, of the propagation of information – true or false – on social networks, or the management of a continental energy network) and the environment (such as the considerable impact cloud computing has on greenhouse gas emissions (Gröger, et al., 2021)). While there are challenges, there are also opportunities: These dynaSoS harbor the potential to solve the tensions between pressing ecological, social, and economic challenges; not least because of their ability to integrate information from various sources at large scales.

DynaSoS require an evolution of existing software and systems engineering approaches to ensure their reliable and secure operation despite their high complexity. These approaches must not only be able to handle the complex interactions of technical subsystems, but also the interactions between technology, people, and the environment. The publicly

1 https://s.fhg.de/studie-digitale-oekosysteme

funded project »DynaSoS« examined how the current state of software engineering must evolve to support a digital transformation in the area of systems-of-systems development. The goal of the project was to collect, consolidate, and present the current research challenges regarding the development of dynaSoS in order to create a research roadmap containing recommendations and research directions.

The dimensions of dynaSoS, the research challenges, and the recommendations, together with their corresponding research directions, are the key elements in the project. Figure 2 provides an overview of how they are related to each other. The dimensions are used to cluster the research challenges, which in turn can be traced to recommendations as motivating challenges. In a similar vein, each recommendation is associated with a certain number of research directions.

Report structure

This report presents the results of the DynaSoS project. It is structured as follows: Chapter 2 presents an overview of the project phases. Chapter 3 highlights the different use cases and example systems that illustrate the kinds of systems we understand by dynaSoS. Chapter 4 introduces the dimensions we use to frame dynaSoS and their corresponding challenges. Chapter 5 presents the research challenges, clustered according to the dimensions of dynaSoS. In Chapter 6, we provide six recommendations with corresponding research directions and discuss the research directions in the context of a bidimensional research roadmap. Finally, Chapter 7 concludes the report by presenting its limitations and providing an outlook on the future of dynaSoS.

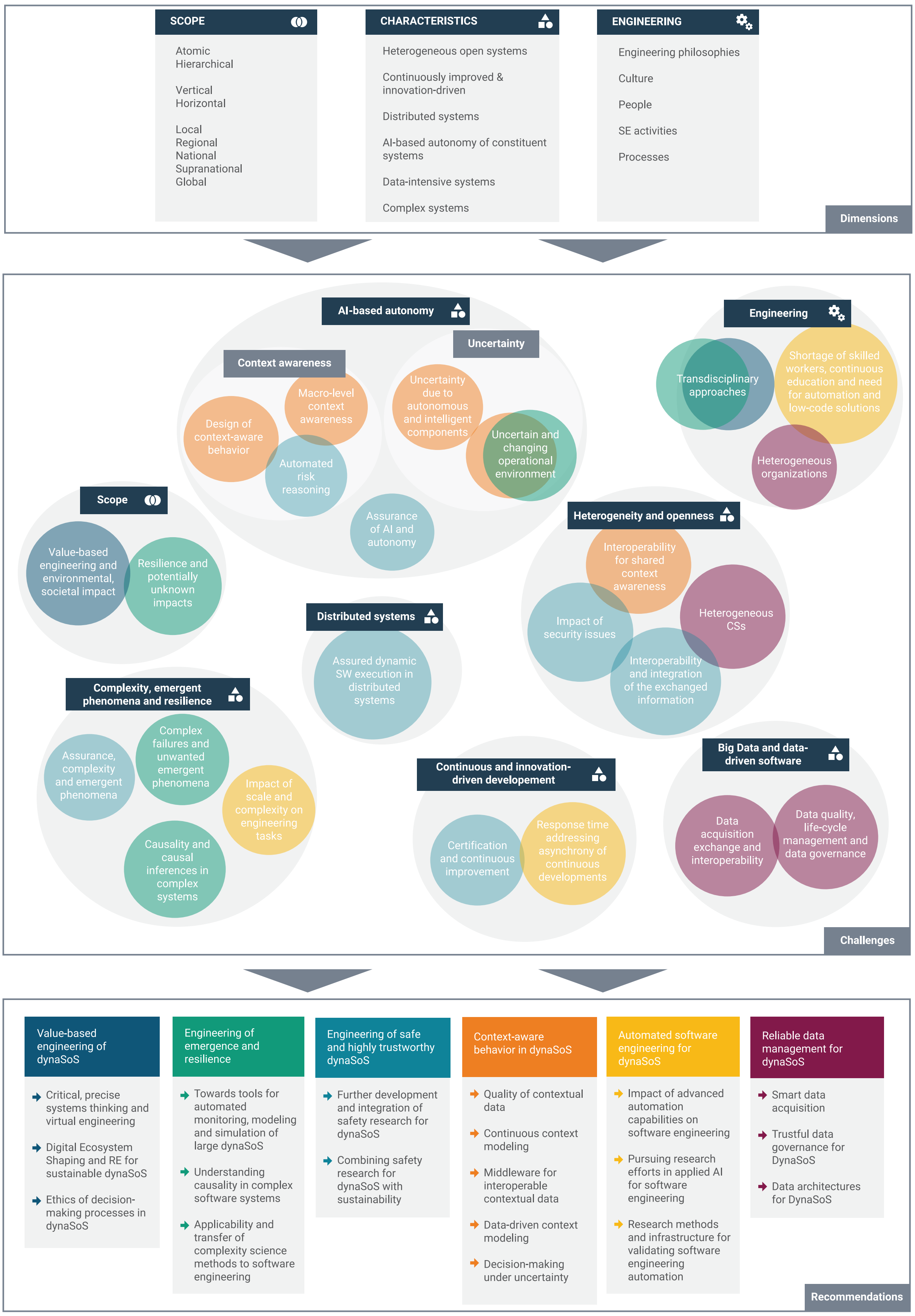

SCOPE
Atomic
Hierarchical
Vertical
Horizontal
Local
Regional
National
Supranational
Global
CHARACTERISTICS
Heterogeneous open systems
Continuously improved & innovation-driven
Distributed systems
AI-based autonomy of constituent systems
Data-intensive systems
Complex systems
ENGINEERING
Engineering philosophies
Culture
People
SE activities
Processes
Dimensions
AI-based autonomy
Context awareness
Design of context-aware behavior
Macro-level context awareness
Automated risk reasoning
Uncertainty
Uncertainty due to autonomous and intelligent components
Uncertain and changing operational environment
Assurance of AI and autonomy
Engineering
Transdisciplinary approaches
Shortage of skilled workers, continuous education and need for automation and low-code solutions
Heterogeneous organizations
Scope
Value-based engineering and environmental, societal impact
Resilience and potentially unknown impacts
Heterogeneity and openness
Interoperability for shared context awareness
Impact of security issues
Heterogeneous CSs
Interoperability and integration of the exchanged information
Distributed systems
Assured dynamic SW execution in distributed systems
Complexity, emergent phenomena and resilience
Assurance, complexity and emergent phenomena
Complex failures and unwanted emergent phenomena
Impact of scale and complexity on engineering tasks
Causality and causal inferences in complex systems
Continuous and innovation-driven developement
Certification and continuous improvement
Response time addressing asynchrony of continuous developments
Big Data and data-driven software
Data acquisition exchange and interoperability
Data quality, life-cycle management and data governance
Challenges
Value-based engineering of dynaSoS
Critical, precise systems thinking and virtual engineering
Digital Ecosystem Shaping and RE for sustainable dynaSoS
Ethics of decision-making processes in dynaSoS
Engineering of emergence and resilience
Towards tools for automated monitoring, modeling and simulation of large dynaSoS
Understanding causality in complex software systems
Applicability and transfer of complexity science methods to software engineering
Engineering of safe and highly trustworthy dynaSoS
Further development and integration of safety research for dynaSoS
Combining safety research for dynaSoS with sustainability
Context-aware behavior in dynaSoS
Quality of contextual data
Continuous context modeling
Middleware for interoperable contextual data
Data-driven context modeling
Decision-making under uncertainty
Automated software engineering for dynaSoS
Impact of advanced automation capabilities on software engineering
Pursuing research efforts in applied AI for software engineering
Research methods and infrastructure for validating software engineering automation
Reliable data management for dynaSoS
Smart data acquisition
Trustful data governance for DynaSoS
Data architectures for DynaSoS
Recommendations

Figure 2. Traceability of challenge clusters, recommendations, and research directions.

Project Overview

In this chapter, we provide an overview of the four project phases and explain how the output of each phase contributes to the subsequent ones.

The DynaSoS project was organized into four phases: (1) Conceptualization, (2) Characterization, (3) Classification of challenges, and (4) Recommendations. Figure 3 illustrates the phases, high-level activities, inputs, and outputs.

Phase 1: Conceptualization

In the first phase, we investigated scenarios for dynaSoS in six application domains (see Chapter 3). For each domain, the project appointed a team of domain experts from Fraunhofer IESE and the Technical University of Kaiserslautern, Germany. In the activity »Identification of relevant use cases«, potential use cases were elicited by (a) reviewing relevant literature, such as roadmaps describing known problems, (b) drawing inspiration from implementation examples and scenarios, and (c) performing interviews with experts from industry and academia. Figures 4, 5, 6, and 7 show the demographics of the participants – there were 97 individuals from 83 organizations.

The focus was on capturing the technological, societal, and economic implications of dynaSoS and the role of systems and software engineering regarding the use cases. The use cases provided a basis for identifying crucial aspects such as key stakeholders, possible business models, and variation points. All these aspects must be considered in the design and development of such systems. For example, a dynaSoS that is not economically viable or is not accepted by society will not be successful. These use cases supported the next activity, the description of the example systems that would be necessary to support the implementation of the use cases. These systems were selected based on their ability to demonstrate typical properties and challenges a dynaSoS will have in the respective domain. Each system described an innovative dynaSoS that is at least several years beyond the status quo, although existing implementations might help to demonstrate its feasibility and realistic expectations. Finally, the elaborated concepts were improved and expanded upon at a more granular level, including visualizations and models. The application areas were free to choose the appropriate format of these artifacts, which differed due to differences in the level of abstraction: While the Smart Mobility and Smart City & Region scenarios take a bird's eye view of a metropolitan setting, Smart Farming and Smart Manufacturing consider a specific application or production site with selected machinery.

Phase 2: Characterization

Both use cases and example system descriptions were used as input for the second phase, the characterization of dynaSoS. In this phase, we searched for the domain-independent characteristics that distinguish dynaSoS from other systems, that is, a set of characteristics that are distinctive and therefore could help differentiate dynaSoS from other types of systems, mainly from traditional systems of systems (SoS). Furthermore, we want these characteristics to underline the dynamic properties that are found in these systems while keeping any overlap between these characteristics at a minimum.

For this purpose, we performed a series of internal workshops in which we iteratively explored and challenged properties that characterize dynaSoS. In addition to the results of the first project phase, we also took the literature into consideration, in particular the classical properties of SoS, to determine (1) the extent to which these support the description of dynaSoS, (2) what aspects regarding dynamism have not been defined yet, and (3) what other aspects are potentially missing. This phase produced a set of characteristics of dynaSoS (see Section »Dimension related to the dynaSoS characteristics«).

Phase 3: Classification of challenges

In the third phase, we investigated research challenges associated with dynaSoS and classified them according to a classification framework, which we refer to as a research architecture. The characteristics of dynaSoS identified in phase 2, together with categorizations concerning the scope and the engineering of dynaSoS, provided the three dimensions of the framework. They aimed at classifying challenges related to the engineering of dynaSoS. The framework was developed and debated through internal workshops at Fraunhofer IESE and external workshops with the participation of expert academics in the field from different research institutions and universities in Germany (see demographics of the participants in Figures 8 and 9).

With respect to the concrete research challenges, we used a two-fold search strategy. On the one hand, we reviewed the literature covering the research challenges for dynaSoS-related systems such as cyber-physical systems, IoT-based systems, adaptive systems, complex systems, and autonomous systems. We extracted more than 240 challenges from 91 papers (see visual depiction of the core aspects of the literature review in Figure 10). On the other hand, we elicited further challenges in the aforementioned internal and external workshops. After identifying, prioritizing, and clustering them, we organized them according to the dimensions of the research architecture.

Phase 4: Recommendations

Finally, we derived research recommendations to address the prominent challenges to engineering dynaSoS. Each recommendation relates to some challenges previously identified and is accompanied by a set of potential research directions that can be pursued to fulfill the recommendation.

Summary

The project DynaSoS was organized into four phases: Conceptualization, Characterization, Classification of challenges, and Recommendations, which were briefly described in this chapter. In the next chapter, we will present the outcomes of the first phase: the use cases and example systems we defined to help identify the characteristics of dynaSoS.

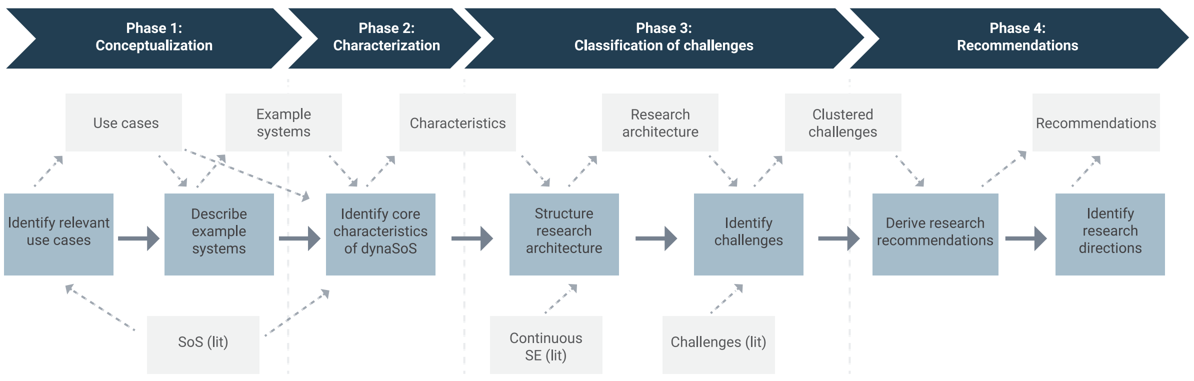

Phase 1: Conceptualization
Phase 2: Characterization
Phase 3: Classification of challenges
Phase 4: Recommendations
Use cases
Example systems
Characteristics
Research architecture
Clustered challenges
Recommendations
Identify relevant use cases
Describe example systems
Identify core characteristics of dynaSoS
Structure research architecture
Identify challenges
Derive research recommendations
Identify research directions
SoS (lit)
Continuous SE (lit)
Challenges (lit)

Figure 3. Project phases, high-level activities, input, and output.

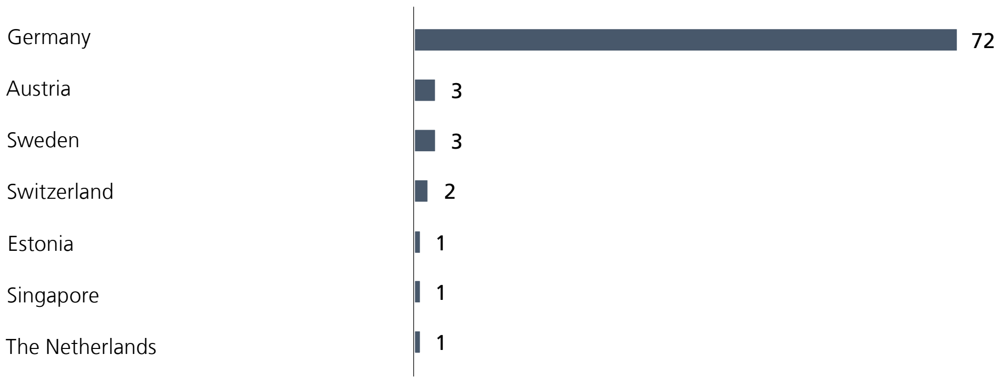


Figure 4. Frequency of organizations per country (N=83 organizations).

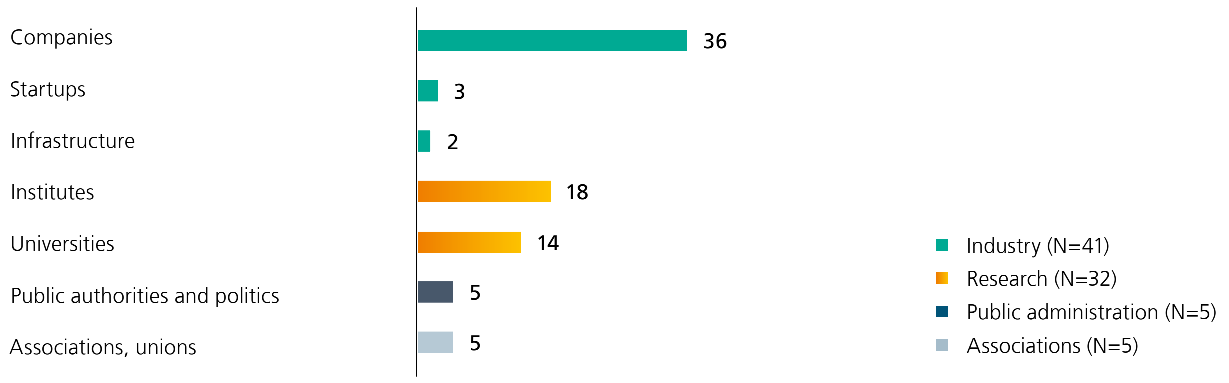


Figure 5. Frequency of organizations per type (N=83 organizations).

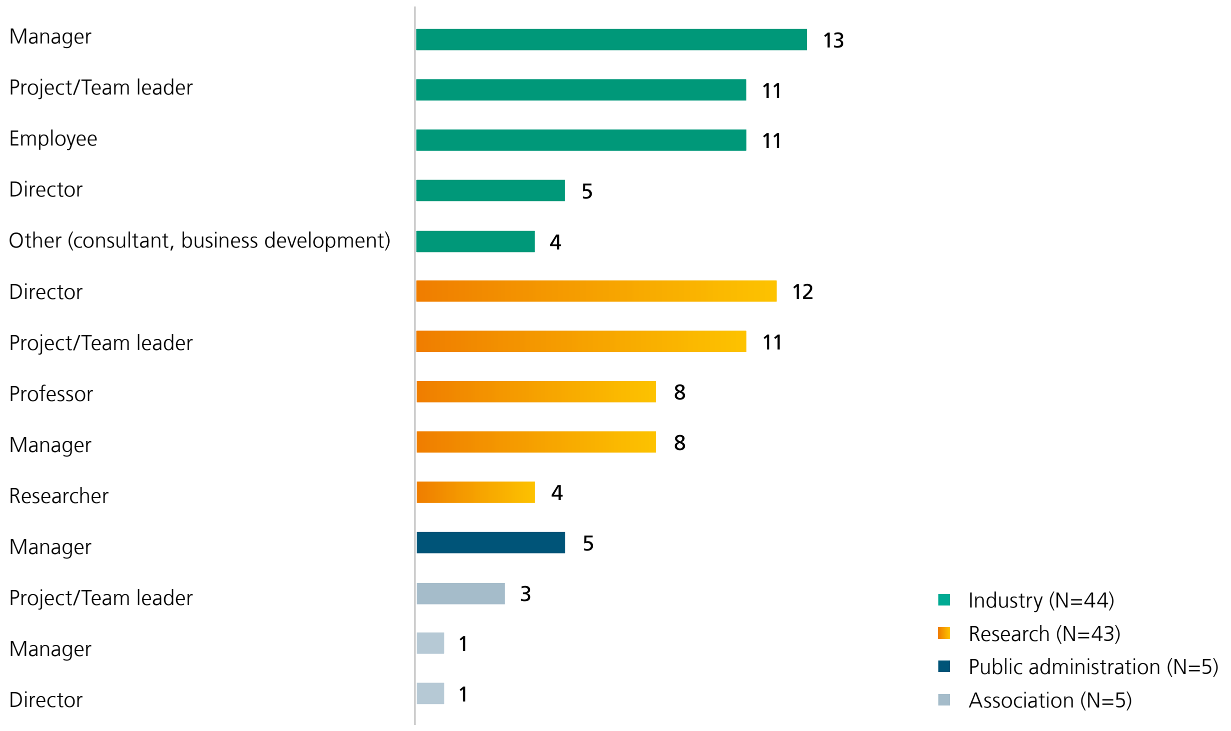


Figure 6. Frequency of role per organization type (N=97 individuals).

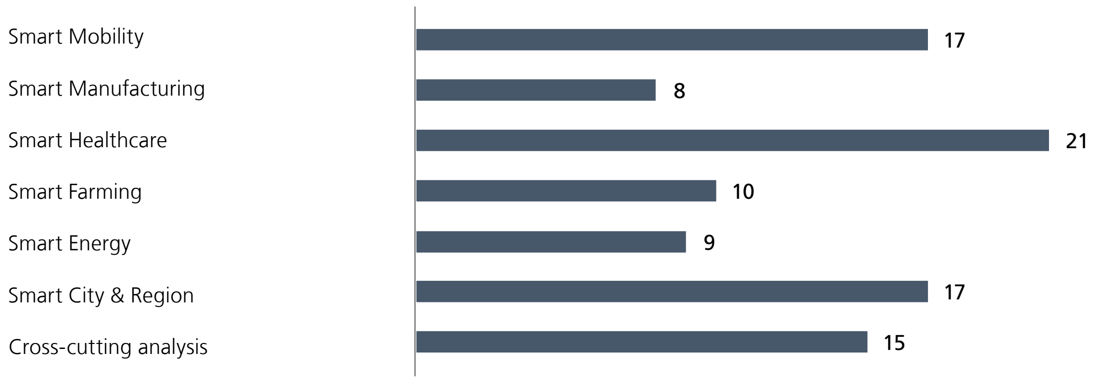


Figure 7. Frequency of participants per application domain (N=97 individuals).

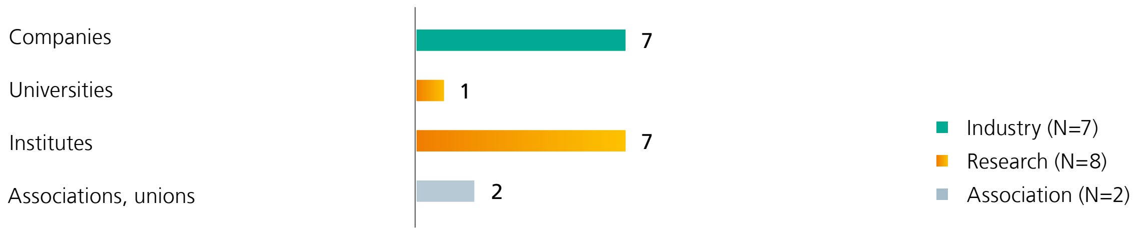


Figure 8. Participants in the external workshops per organization type (N=17).

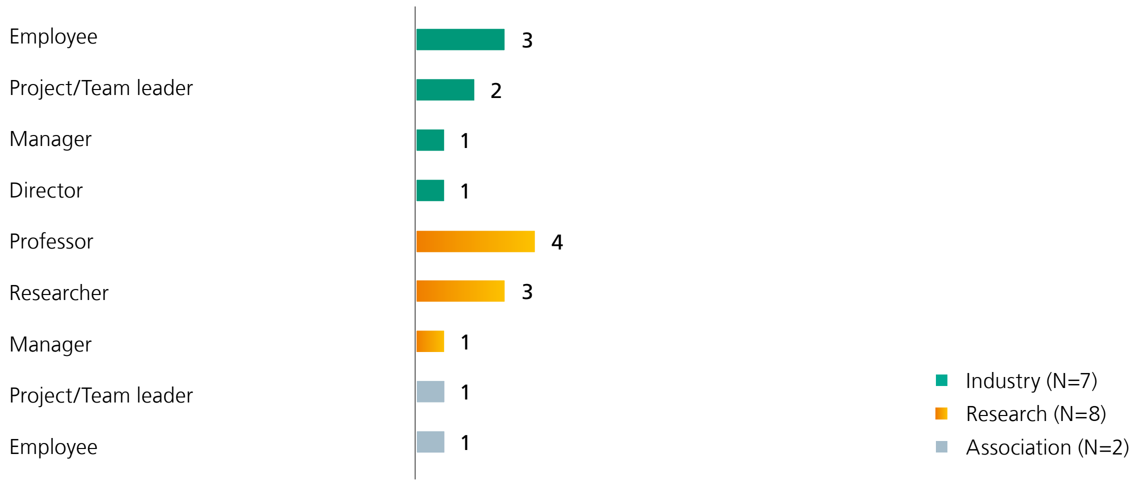


Figure 9. Participants of the external workshops per role (N=17).

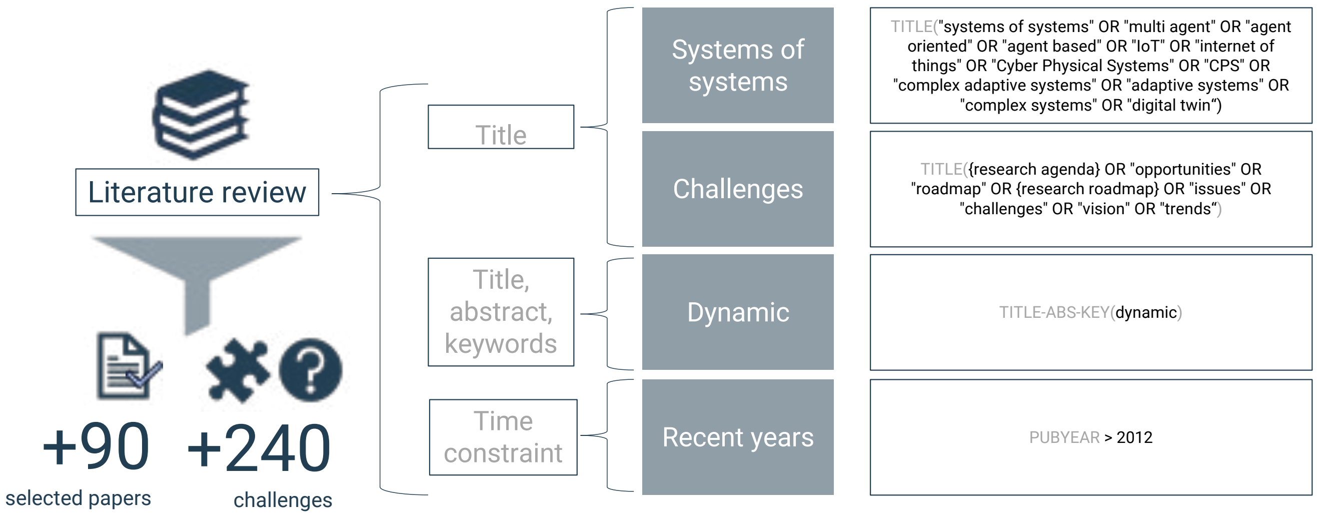

Literature review
+90
selected papers
+240
challenges
Title
Title, abstract, keywords
Time constraint
Systems of systems
Challenges
Dynamic
Recent years
TITLE("systems of systems" OR "multi agent" OR "agent oriented" OR "agent based" OR "IoT" OR "internet of things" OR "Cyber Physical Systems" OR "CPS" OR "complex adaptive systems" OR "adaptive systems" OR "complex systems" OR "digital twin")
TITLE({research agenda} OR "opportunities" OR "roadmap" OR {research roadmap} OR "issues" OR "challenges" OR "vision" OR "trends")
TITLE-ABS-KEY(dynamic)
PUBYEAR > 2012

Figure 10. Visual depiction of the literature review regarding challenges in dynamic systems of systems.

Use cases and example systems

In this chapter, we present the use cases that we created for six application areas of dynaSoS and their corresponding example systems. Furthermore, we also provide a transversal analysis of dynaSoS across the different domains.

Motivation for conceptualization work

Envisioning systems that are not yet in place – such as dynaSoS – requires people to imagine an innovative future before research challenges and recommendations can be proposed for them. Because the vision of dynaSoS is years beyond the status quo, abstract concepts such as autonomous, dynamic, reliable, and domain-agnostic systems are difficult to grasp. Individuals tasked with designing this vision need tangible examples to be able to understand the subject matter in depth, justify the benefits of dynaSoS, and determine their feasibility. Without referring to specific instances, it is also difficult to communicate these aspects to the public.

As a result, a significant portion of the project was dedicated to identifying relevant use cases and describing representative example systems. These activities made the concept of dynaSoS tangible. To increase the chance that the findings of DynaSoS can be extrapolated to other domains, we considered six application areas where dynaSoS are expected to play a crucial role in the foreseeable future: Smart Farming, Smart Manufacturing, Smart Mobility, Smart Healthcare, Smart Energy, and Smart City & Region. This provided the necessary depth to identify and explicitly consider important domain-specific properties. The six use cases we defined were published as a whitepaper (Groen, et al., 2022), and four example systems were presented in a blog entry on the DynaSoS project website[2].

2 https://dynasos.de/news/

In the conceptualization phase, we essentially translated the abstract visions into realistic scenarios in a given application context. This way, the scenarios help practitioners and researchers understand what the contributions of different dynaSoS (use cases) might reasonably be, and what those systems could look like from a technical perspective (example systems). Furthermore, the scenarios provided an indispensable discussion basis to envision those systems when composing and deriving the requirements, research architectures, research challenges, and recommendations regarding dynaSoS. A better understanding of the concept helped us to estimate the degree of automation, the ways these systems are interconnected, and the potential for innovative technologies.

While the use cases focused on the challenges that needed to be solved, the example systems illustrated how dynaSoS could practically solve or mitigate some of the urgent problems. Moreover, the use cases allowed us to investigate their economic advantage in highly regulated domains as well as their effect on increased compliance with the standards and regulations that cannot be attained by the state of the art.

Overview of application areas

The application areas provided the following use cases and example systems:

Smart Farming

Among the many shifts towards Digital Farming, crop protection is an important building block for food production. The lack of technical interoperability between interfaces and dynamic coupling of systems currently prevents the vision of treating every plant on the field individually from being realized. The vision requires the Farm Management Information System, sensors, autonomous agricultural vehicles, weather information systems, and other systems to exchange data. A dynaSoS can contribute to many things, including actively monitoring the application of crop protection products, preventing excess drift of pesticides to surrounding areas, detecting pests before they are perceptible by the human eye, and reacting accordingly. This requires high data quality in dimensions such as reliability, granularity, and timeliness. This vision was described in the use case »Reducing goal conflicts in the sustainability triangle in crop protection through consistent digitalization«.

The example system involves drones with sensors and satellites to determine the condition of the soil and the plants. Parts of the concept can also be transferred to other agricultural processes like fertilization. The description of the example system was »Dynamic and connected: To what extent will future crop protection be digital?«[3]

3 https://dynasos.de/2022/09/30/dynamisch-und-vernetzt-wie-digital-wird-der-pflanzenschutz-in-zukunft-sein/

Smart Manufacturing

Production lines are typically configured manually in order to be statically assigned to one specific production asset. By introducing technology that allows a production line to be dynamically reconfigured to a different production asset, costly shutdowns of the production line for reconfiguration purposes can be avoided. This vision was described in the use case »Dynamically reconfigurable production through virtual production lines«[4].

The example system, named »Dynamically reconfigurable production using dynamic systems-of-systems demonstrated with an example«, demonstrates how a Manufacturing Execution System (MES) performs production planning using virtual production lines. This planning not only improves the static assignment of production assets, but can dynamically share production plans and assets with other digital production lines to maximize efficiency. Part of the dynamic adaptivity of SoS relies on systems to describe their capabilities and skills to other systems. This enables the MES to orchestrate the systems independent of a device's manufacturer or type by comparing these systems, conducting feasibility checks, and assigning production tasks. Flexibility of the shop floor can, for example, be promoted through flexible transportation systems such as Automated Guided Vehicles, Autonomous Mobile Robots, or autonomous forklifts.

Smart Mobility

Urban mobility is associated with pollution, stress, and costs, which can be reduced by improved traffic flow. Parcel delivery greatly benefits from curbside management solutions, such as dynamic assignment of loading zones. A dynaSoS for mobility relies on high interdependence and dynamic coupling to operate within the continuously changing traffic flow. If this is achieved, it can contribute to reduced emissions and improved quality of life in the city. This vision was described in the use case »Dynamic delivery zones for optimized inner-city goods and delivery traffic«.

The example system »DynaZone«, described in the article »Smart delivery zones as a dynamic system-of-systems«[5], proposes monitoring smart parking zones using sensors and assigning priority tokens to reserve a zone. The system can assign zones for the predefined route of a parcel delivery vehicle and dynamically adapt to changes, such as delays or a DynaZone getting reassigned to an emergency response vehicle that has higher priority, while violators are fined to ascertain the continued availability of DynaZones.

Smart Healthcare

ATMPs are concentrated therapeutic agents such as CAR-T cells, microbiota, or mRNA. They can be used for personal medicine in which a single patient receives an individually produced ATMP (e.g., for cancer treatment), and for vaccines. The manufacturing process of ATMPs is highly manual, resulting in time-consuming and cost-intensive production, and only a limited number of patients that can be treated with ATMPs. To achieve a flexible, scalable, and automatable production chain, the systems currently in use need to be able to interoperate properly, not just for orchestrating the production steps, but also for strict quality control, including measurements, sterility, bioreactive processes, and the associated automated data analysis. This vision was described in the use »Smart Production of Advanced Therapy Medicinal Products (ATMP)«. Because the concepts for automating ATMP production follow the principles of Industry 4.0, this use case is considered a special instance of the example system of Smart Manufacturing.

Smart Energy

Organizing the energy grid into small and local energy cells supports the increased demand for flexibility while the grid is becoming more complex due to the growing number of energy consumers, generators (some of which are regenerative and volatile), and storage options. Using AI, these cells can autonomously balance their production and consumption. In essence, the dynaSoS performs »energy cell management« – compensating for energy surpluses and deficits at higher-level cells and converting the energy into different voltages. This vision was described in the use case »A connected cellular energy system for complexity control at a granular level«. Because the energy grid is a part of the infrastructure, most visibly that of urban areas, this use case is considered part of the example system of Smart City & Region.

4 https://dynasos.de/2022/10/14/dynamisch-rekonfigurierbare-produktion-mittels-dynamischer-systems-of-systems-vorgestellt-anhand-eines-beispiels/

5 https://dynasos.de/2022/10/07/smarte-lieferzonen-als-dynamisches-system-of-systems/

Smart City & Region

Factors such as urbanization, demographic and climatic changes, and people's habits influence the demand for simultaneous use of space, constraints on the use of resources (water, energy, money, skills), and the possibilities for implementing climate protection and adaptation measures in urban environments. One of the current challenges facing experts in the field is the ability to acquire real-time information of sufficient quality to support decision-making regarding the organization and development of urban areas. More futuristic approaches explore the possibility of multi-functional systems (e.g., connected and adaptive blue-green infrastructure) that adapt to the needs and constraints of cities (e.g., rainwater harvesting umbrellas, vertical gardens and living moss walls, temporary and mobile forests[6]). This vision was described in the use case »Self-driving trees: Blue-green infrastructure that adapts to a city's needs«. The example system described in the article »Urban sustainability through dynamic systems of systems«[7] illustrates how a dynaSoS approach could potentially help by making blue-green infrastructure more adaptable and connected.

Analysis of cross-cutting aspects

In order to identify dependencies between the application areas, we iteratively performed a cross-cutting analysis. This analysis aimed to steer and align the selection of use cases to fulfill three objectives:

1. Aligning the use cases with the notions of SoS that exist in S&SE;
2. ensuring that different kinds of SoS are considered; for instance, we wanted to make sure that not all use cases deal with swarms of robots because large-scale dynaSoS like intelligent traffic systems or smart grids are likely to face different kinds of challenges;
3. making sure that domain experts have S&SE challenges in mind, meaning that they should provide illustrative example use cases of their application domain that are sufficiently profound to address these challenges.

To perform the analysis, a cross-sectional team with expertise in SoSE conducted workshops with experts from the respective domains. An important activity involved the consolidation and uniformization of definitions within the project, and the achievement of a common understanding of dynaSoS and SoS.

Next, we discussed key aspects of dynaSoS, such as the importance of AI, the manifestation of dynamic coupling between the constituent systems, and the ways in which virtualization of the systems adds value. These aspects were found to manifest themselves differently in the different application domains. The cross-cutting analysis helped to sharpen the common scope of the domains. For example, virtualization has played a major role in Smart Manufacturing ever since the rise of Industry 4.0. For Smart Farming, on the other hand, it is a new challenge, as not only agricultural machines but also natural systems may be transformed into Digital Twins within a dynaSoS. Aspects such as safety are particularly relevant in areas where an increased risk of physical injury is to be expected.

The cross-cutting analysis demonstrated that further development of systems engineering (SoSE) must be considered as a major research aspect. SoSE builds on the domain of traditional SE. It helps to cope with the high complexity of system networks and to understand dependencies. So far, in the domains considered, only a few SoSE methods have been applied, such as (Mennenga, et al., 2020). To close this research gap, we investigated the current challenges in SoSE through an interview study resulting in 36 clusters of statements. These were then prioritized by importance and the potential for research to contribute to a solution. The investigation resulted in 13 challenges for the realization of dynaSoS, which were consolidated in the final results (Balduf, et al., 2022).

Summary

We elaborated the use cases and example systems in detail to make sure we attained the highest possible quality in painting a meaningful and robust picture of dynaSoS for different application areas. In turn, the use cases and example systems support the identification and derivation of typical characteristics of dynaSoS. The characterization of dynaSoS is the focus of the next chapter.

6 https://arcadia.frl/de/projecten/bosk/

7 https://dynasos.de/2022/11/29/nachhaltigkeit-in-urbanen-raumen-durch-dynamische-systeme-der-systeme/

Dimensions of dynaSoS

In this chapter, we introduce the research architecture of dynaSoS, a classification framework we used to characterize these systems.

Motivation

The first phase of the project involved gathering evidence on examples of forward-looking software systems. The second phase of the project aimed to consolidate this material and develop a framework to help define the characteristics of dynaSoS, reflect on the evolution of current software systems, and systematically organize research challenges related to dynaSoS. This framework is called a research architecture and consists of three core aspects: scope, characteristics, and engineering, each of which has different dimensions. Each aspect and its dimensions are described in more detail in the following sections. An overview of the research architecture is given in Figure 11.

Dimension related to scope of dynaSoS

DynaSoS can have completely different scales. Consider, for instance, a few collaborative robots in a factory versus a complete smart city or a global supply network. The scale has an impact on the engineering challenges and is illustrated in Figure 12. On the left of Figure 12, one can see some collaborating robots and drones that together form an SoS. The robots and the drones are the constituent systems (CS) of the SoS. If none of the constituent systems in an SoS is an SoS itself, we call the SoS **atomic**. Otherwise, we call it **hierarchical** and refer to hierarchy levels for the depth of its nested structure.

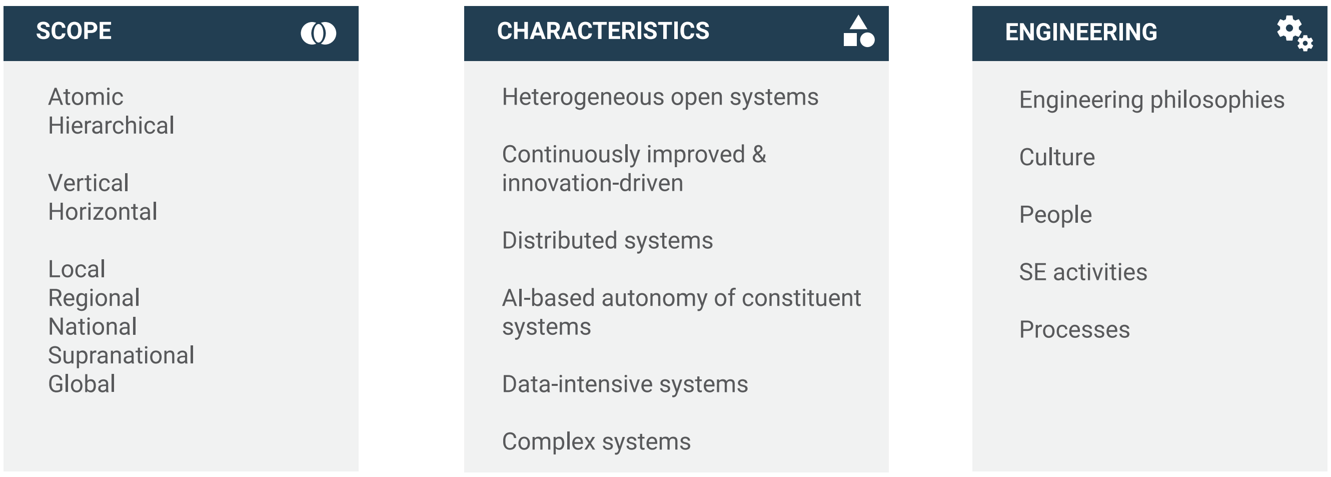


Figure 11. Overview of the three dimensions in the research architecture

A second aspect is whether an SoS is limited to a single domain or not. If an SoS belongs to a single application domain such as mobility or energy, we call it **vertical**. Otherwise, we call it **horizontal**.

A third aspect concerns the geographic distribution. To describe this aspect, we use terms such as **local**, **regional**, **national**, **supranational**, and **global**.

The three aspects are loosely related to each other. Atomic SoS tend to be local and vertical. Hierarchical SoS can be local and vertical, but the higher the hierarchy level, the more likely the SoS is horizontal and the more likely it has a wider geographic distribution. According to these loose dependencies, we roughly distinguish between four kinds of SoS, which are illustrated in Figure 12 and described in the following.

Small vertical atomic dynaSoS: Small, atomic, vertical dynaSoS are some (autonomous) robots, drones, or other machines that collaborate to implement a domain-specific task. In doing so, they can dynamically adapt to various context conditions. This dynamism is often enabled by AI and not considered in conventional SoS.

For instance, a swarm of field robots may collaborate when weeding. The CS of these SoS might be developed from scratch because they did not previously exist. For instance, a farmer may have only conventional agricultural machinery but no field robots that are able to collaborate. A manufacturer of agricultural machinery will likely develop collaborative field robots independently from existing conventional agricultural machinery because the latter have no collaboration capabilities. It is thus rather a development from scratch than evolutionary development. Furthermore, the manufacturer might do this without considering collaboration with field robots from other manufacturers. This means that the SoS characteristic »managerial independence« might not be fulfilled.

Vertical hierarchical dynaSoS: A vertical hierarchical dynaSoS involves various systems (including SoS) from the same vertical. It collects information from its CS and dynamically influences their behavior.

For instance, an Agricultural Data Space (ADS) may collect information to decide when it is time for weeding or irrigation. Based on this information, it tells weeding robots and irrigation robots how to do their job. The ADS and all the systems that are connected to it form a vertical hierarchical dynaSoS. Similar data spaces or platforms are envisioned in other verticals. For instance, the vision of shared multi-modal mobility requires all means of transportation to be connected to platforms that organize the matching between mobility demands and offers. A smart grid requires a platform that organizes the matching between flexibility demands and offers.

The basis for these dynaSoS is more and/or better information and communication technology (ICT). Existing SoS in various verticals are transformed by introducing ICT and by using the information to dynamically control and adapt processes in these verticals. Often, a lot of information needs to be processed in a very short timeframe. This rapid processing typically goes beyond human skills and motivates the application of technology, including AI-based control or AI-based recommendations. Thus, ICT and AI are enablers for the transformation from SoS to dynaSoS. ICT and AI were already known when Maier introduced the SoS characteristics in 1998, but the technological possibilities have grown tremendously since then. Moreover, there is increasing awareness and political pressure

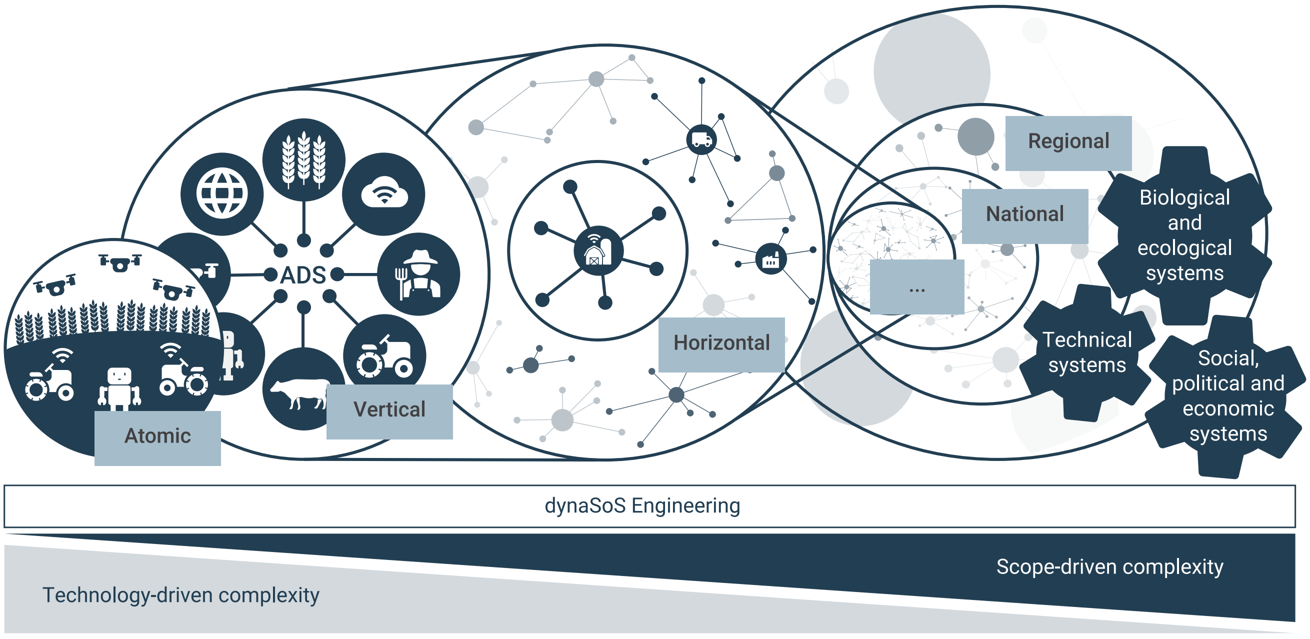


Figure 12. Illustration of different scopes of dynaSoS

that existing SoS in domains such as energy, mobility, and agriculture, to name but a few, have to be transformed in order to achieve sustainability objectives (Niestroy, et al., 2020).

Horizontal hierarchical dynaSoS: A horizontal hierarchical dynaSoS involves various systems (including SoS) from different verticals. It collects information from its CS and dynamically influences their behavior.

A prominent example is a smart city that integrates a smart energy dynaSoS and a smart mobility dynaSoS. Another example is a food supply chain that dynamically adapts how goods are transported depending on the current prices for possible transportation means. It involves systems from smart farming, smart manufacturing, and smart logistics.

Large holistic dynaSoS: The attributes large and holistic refer to sustainability. Following popular definitions of sustainability, we consider it something as holistic if it includes at least ecological, social, and economic dimensions. Furthermore, we consider it as something that refers to planetary boundaries (Rockström, et al., 2009). Consequently, engineering sustainability requires considering the entire planet, define goals at this level, and break them down to so that they can be allocated to something, that is, to large-scale dynaSoS.

A prominent example of engineering sustainability at a global scale is given by the 17 UN Sustainability Development Goals (SDG). These global objectives can be broken down into supranational objectives. For instance, the European Commission addresses the SDGs with the European Green Deal and other concepts (EU Commission, 2020). These objectives are further broken down to nations and verticals such as energy, transportation, production, and agriculture. Establishing reasonable global goals and breaking them down in a reasonable manner requires analyzing the interaction between biological/ecological systems, social/economic/political systems, and technical systems. These systems are loosely coupled, geographically distributed, evolve permanently, and generate emergent behavior. Furthermore, the interaction between the systems is of a dynamic kind, as described in (Scoones, et al., 2007): »'Dynamics' refers to the patterns of complexity, interaction (and associated pathways) observed in the behavior over time of social, technological and environmental systems«. The right part of Figure 12 illustrates the interaction between biological/ecological systems, social/economic/political systems, and the »mediating role of technology in altering and being altered by natural and social-political systems« (Scoones, et al., 2007).

Dimension related to the dynaSoS characteristics

We base our definition of the characteristics of dynaSoS on previous work in the domain of SoS. The literature on SoS usually defines five major characteristics: managerial independence, operational independence, evolutionary development, geographic distribution, and emergent behavior (Maier, 1998; Gorod, et al., 2008). We extended and reorganized these initial five SoS characteristics into dynaSoS characteristics. The motivation comes from the increased usage and application of AI, autonomous systems, Big Data, and data science. In total, six categories were chosen:

- Heterogeneous Open Systems (Managerial Independence & Operational Independence)
- Continuously Improved & Innovation-Driven (Evolutionary Development)
- Complex Systems (Emergent Behavior)
- Distributed Systems (Geographic Distribution)
- Data-Intensive Systems
- AI-based Autonomy of Constituent Systems

These categories were chosen because they give rise to challenges that do not necessarily overlap with each other but cover the full range of dynaSoS issues.

Heterogeneous Open Systems (Managerial Independence & Operational Independence): DynaSoS are composed of systems that operate independently and are often managed by different organizations. This typically leads to heterogeneity on multiple levels. On the technological level, different hardware and software technologies may need to interact. On the organizational level, different parties may have to agree on common goals and to synchronize their processes while pursuing their particulars goals, which may not always converge on the overarching goal of the system of systems to which they belong. An important property is **operational independence**. This property implies that any constituent system that is part of an SoS can be operated independent of the others and would still work if the SoS was disassembled. Operational independence contributes dynamics because a constituent system is not bound to an SoS. It is loosely coupled and can also contribute to other SoS with its capabilities. It also implies that the constituent systems have an (open) interface for collaboration. **Managerial independence** is a related characteristic of an SoS. It means that constituent systems in an SoS are managed independently and their owners may be evolving the systems to meet their own needs. Managerial independence is not a technical characteristic but a major cause of many systemic non-technical issues that hinder interoperability and lead to heterogeneous systems. Furthermore, the interfaces of constituent systems from different verticals are likely not interoperable. Together, these characteristics relate to the software-related research around open and heterogeneous systems as well as related interoperability issues.

Continuously Improved & Innovation-Driven Systems (Evolutionary Development): DynaSoS are not only developed and managed by different organizations, but also continuously enhanced. Novel features are constantly shipped into production, and novel systems need to be integrated and tested, whereas older systems may be retired. In that sense, the development of dynaSoS can be seen as a continuous and evolutionary process. Consequently, a dynaSoS will evolve incrementally rather than being 'delivered', as normally envisioned in a single system development or acquisition. Considering continuous software engineering, the timeframe between incremental steps can be very short.

Distributed Systems (Geographic Distribution): Like all SoS, dynaSoS are inherently distributed systems. They are not only developed by different entities, but the constituent subsystems may also be physically located in different places and interconnected through communication networks. In many cases, the location of the cyber-physical constituent systems with their execution platforms may change as they are mobile. Another aspect is the dynamic allocation of software functions to execution platforms. The characteristic »geographic distribution« can lead to various regulatory challenges. However, related research challenges rather address regulatory science than software engineering/research. Geographic distribution implies that there is potential for dynamic allocation of software functions to execution platforms, and opens up new opportunities for efficiency. This and other topics of distributed systems are more relevant from a software engineering/research perspective. Research on distributed systems has a long tradition and forms a research field on its own.

AI-based Autonomy: DynaSoS include constituent systems that act autonomously. This means that they are not controlled or operated by a human operator. Therefore, they have to sense and understand the environment. Based on this context awareness, which can also include anticipation of future scenarios, they make decisions and act accordingly. Depending on the degree of autonomy, there is no human supervisor, or the human is only partially in the loop. In contrast to conventional automation, autonomy comes with the notion of a programmed-self because dynaSoS act in a very situation-specific way, and even designers can often not explain upfront how the systems will behave in particular situations. A typical reason for the latter black-box behavior is the use of AI. Context awareness is often the key capability needed in order to react properly to operational situations. AI-based autonomy is one aspect that characterizes the difference between SoS and dynaSoS.

Data-Intensive Systems: Another aspect is that dynaSoS are data-intensive systems. Kleppmann defines an application as data-intensive »if data is its primary challenge – the quantity of data, the complexity of data, or the speed at which it is changing – as opposed to compute-intensive, where CPU cycles are the bottleneck« (Kleppman, 2017). This is equivalent to the concept of Big Data. In simple terms, Big Data is a situation that occurs when conventional approaches to data processing (i.e., moving data from storage to the main computer memory for processing, and then moving the results to storage) become virtually unfeasible or too expensive. Big Data can occur because the amount of data is simply too large to be processed by the computer at hand. It can also occur because the speed at which the data must be processed is too demanding (Laney, 2001) (Das, 2020).

Complex Systems (Emergent Behavior): Emergence occurs when a system is observed to have properties that its constituent parts do not have on their own. Such emergent properties or behaviors arise from interactions between constituent systems. In addition, an emergent phenomenon affects its constituents: There is a feedback loop between the whole and its parts (Siegenfeld, et al., 2020; Parrend, et al., 2022). At least two levels of abstraction are needed to see, understand, and control emergent phenomena: the micro level, which describes the components, and the macro level, which describes the system as a whole.

A traffic jam (e.g., on a motorway) is a simple example of emergent phenomena. The cars are the micro-level components. The traffic jam is the emergent phenomenon that happens at the macro level. A traffic jam can occur when cars interact and slow down, e.g., because of merging lanes or an accident. A traffic jam is a phenomenon with its own dynamics. It flows in the opposite direction of the cars and can spread in space (some can grow to several kilometers in length) and in time (sometimes even when the source of the slowdown is no longer present). Incoming cars are also affected by the traffic jam: They also have to slow down.

Some emergent phenomena are desirable, while others are unwanted. For instance, an intelligent traffic management system may generate emergent behavior that optimizes traffic flow, but other phenomena may lead to a traffic jam.

Dimension related to the engineering of dynaSoS

Over the years, software engineering tasks, processes, and methods have co-evolved with the type and scale of the software systems being engineered (Booch, 2018). For example, the recent increase in the use of AI-based software systems has raised new challenges for software engineering, and both technical (e.g., data version control, feature stores) and non-technical (ML-Ops, data and AI governance strategies) solutions are currently emerging (Martinez-Fernández, et al., 2022). The life cycle of software systems and the corresponding software engineering activities will continue to evolve in response to changes in the scale and complexity of software systems, as well as in response to the societal, economic, and environmental constraints (like moving toward a circular economy) that will apply to these systems.

Engineering Philosophies, Culture, and People: As digitalization and software continue to affect all domains, the demand for software engineering competencies will continue to grow. Furthermore, the scale and complexity of the systems being designed is also changing the set of skills software engineers need to acquire (think of data privacy, algorithmic fairness). For instance, the last two decades have seen the emergence and adoption of philosophies such as agile, chaos engineering, Site Reliability Engineering, or DevOps, which have been fueled by existing software engineering challenges and ideas from manufacturing practices or complex systems sciences, and are now spreading to other domains (e.g., agile administration). We foresee that the evolution toward cross-functional, transdisciplinary, and more diverse teams will continue, including more socio-technical and environmental competencies. This is because dynaSoS imply even more collaboration and communication between organizations having different, sometimes even conflicting goals, strategies, cultures, and processes.

Software engineering lifecycle, activities, and processes: As mentioned above, software engineering co-evolves with the software systems being developed. Although it is difficult to imagine what novel software engineering activities will happen in the future, we can foresee that systems such as dynaSoS will require novel software engineering tasks. The current state-of-the-art descriptions of software engineering activities, such as the SWEBOK[8] categories and continuous software engineering (Bosch, 2014; Farley, 2022; Antonino, et al., 2022; Humble, et al., 2010; Klotins, et al., 2022; Fitzgerald, et al., 2017), provide an overview of software activities as of today. As systems evolve toward dynaSoS, there will be the need to adapt and reinvent some software activities (see, for instance, the specific changes brought on by the use of AI for requirements engineering (Scharinger, et al., 2022) and for testing (Zhang, et al., 2022), or the recent advances in code generation through large deep learning models such as OpenAI Codex (Puryear, et al., 2022)).

Summary

The research architecture with its three core dimensions Scope, Characteristics, and Engineering, together with their associated categories, provides a framework for navigating the universe of dynaSoS. This framework formed the basis for the classification of the research challenges presented in the next chapter.

8 http://swebokwiki.org/Main_Page

Research Challenges

In this chapter, we list 24 research challenges for dynaSoS. These challenges were clustered according to the dimensions of the research architecture. While the dimensions »Scope« and »Engineering« provided one challenge cluster each, the dimension »Characteristics« contributed six clusters – one for each characteristic.

Humanity currently faces environmental, sanitary, and geopolitical crises that compel us to consider the implications of software systems and their potential to sustainably solve (or help solve) these crises. In a less technical sense, they also force us to reconsider the way we think and act. Indeed, the evolution of software systems by scale is accompanied by changes in how we manage projects (e.g., by introducing more agility, more cross-functional teams; see DevOps and VUCA approaches), and distribute and evaluate software (e.g., by using more open-source solutions, automating testing and deployment, or evaluating the fairness of automated decisions). On the one hand, dynaSoS can be one of the keys to implementing solutions to present and future crises. On the other hand, engineering dynaSoS is challenging, and failures may have catastrophic consequences on society and the environment. In the following, we present the challenges related to dynaSoS that we derived from the literature and several workshops. These challenges were classified according to the dimensions presented in the previous chapter.

Scope

The first dimension distinguishes between different types of dynaSoS ranging from small, atomic dynaSoS like a swarm of robots to large-scale dynaSoS connecting various technical, social, ecological, political, and other kinds of systems around the globe (Peter, et al., 2014). The challenges depend a lot on the type of dynaSoS and the related engineering scopes. The scope of engineering for many dynaSoS is given by regulatory constraints and business cases. It is very complex to develop these laws in such as way that they will lead to a sustainable world (i.e., living within the planetary boundaries) (Niestroy, et al., 2020). The modeling and analysis of large-scale dynaSoS can help to cope with this complexity and to solve this interdisciplinary challenge involving regulatory science, complexity science, sustainability science, and ethics. This may lead to a novel transdisciplinary role of S&SE that goes beyond the traditional transdisciplinary focus on mechanics, electronics, and software.

Defining government structures and drafting regulations is paramount in steering the evolution of large-scale SoS. However, regulations can only pinpoint a rough direction because it is neither possible nor reasonable to regulate everything in detail. This is especially true for the detailed behavior of constituent systems and the related emergent behavior of a dynaSoS. As the behavior of technical systems is implemented by means of software, software engineers can significantly influence the direction of the digital transformation. This leads to the challenges described in the following sections. Developing laws to amend regulatory constraints and economic conditions for business models goes beyond the scope of the DynaSoS project.

Value-based-engineering and environmental, societal impacts
It is challenging to bridge the gap from values of societies to practice in SE. Software-related design decisions can have a huge environmental and social impact (Schneider, et al., 2022). The growing trend toward connected objects, cloud computing, and increased digitalization opens up many opportunities for collecting relevant information for shaping the digital transformation of SoS and to engineer dynaSoS for a better world. It is necessary and challenging to understand and control the impact of technical systems on ecological systems, societies, and individuals. Safety science and safety engineering provide many approaches for dealing with safety risks, but these approaches are not sufficient when it comes to autonomous behavior of constituent systems, emergent behavior of dynaSoS, or other values like fairness or sustainability. Value-based engineering as described in IEEE 7000 (IEEE Computer Society, 2021) already addresses some of these issues, but these aspects are still not a mainstream component in the SE curriculum (Casañ, et al., 2020).

Resilience and potentially unknown impacts
History has shown that many technological risks were unknown until scaling up the technology led to severe consequences. Many dynaSoS are large-scale systems where scaling effects can be achieved very fast. Software updates can fundamentally change the behavior of constituent systems in dynaSoS. On the one hand, this makes it possible to instantly react to identified risks that were not foreseeable. On the other hand, this way of dealing with unknown unknowns affects the overall dynamics of the dynaSoS and may steer it towards tipping points. Performing risk management is only possible if risks are known. Instead, common approaches from security engineering to address unforeseen attacks may be more applicable. However, this kind of bug-fixing with respect to malicious faults is fundamentally different from controlling the interaction of technical constituent systems with other constituent systems in a dynaSoS. The technical constituent systems have to behave in a way that assures that the overall dynaSoS will remain in a state that is at a safe distance from any critical (tipping) points or conditions. For instance, flexibility management software in a smart grid has to protect the overall dynaSoS from power network overloads that would cause power outages.

Complexity, emergent phenomena and resilience

DynaSoS are complex systems that may be confronted with emergent phenomena. The main challenge with emergent phenomena is that they are relatively difficult to predict. This is due to the fact that the number of possible interactions grows exponentially with the number of interconnected systems. It is therefore simply impossible to monitor all possible interactions. In addition, the effect of interactions is often non-linear. This means that small changes can have large consequences (butterfly effect). The effect of an intervention may trigger a response at a different scale. It may take a long time for the effects to be noticed, or the effects may occur in a different system. Therefore, the effects of decisions made at the software engineering level (during development or operation of dynaSoS) might not be seen and understood in time by those responsible.

Complex failures and unwanted emergent behaviors
Because constituent systems may interact in ways that are sometimes impossible to foresee, it is difficult for software engineers to find root causes and handle failures in complex systems. In his seminal book »The logic of failure« (Dörner, 2003), Dietrich Dörner illustrates that, in order to make decisions about complex systems, people need the right mindset. In the domain of software engineering, complexity and emergent phenomena have been acknowledged, and shifts in the way systems and software are engineered and operated are already taking place (Jamshidi, et al., 2018). The main point is that even if novel processes or development approaches are in place, failure management in complex systems remains a major challenge (Snafucatchers consortium, 2017). One of the main problems remaining is how to engineer emergent behavior that controls unwanted behaviors (such as safety or security risks) and how to monitor and react to emergent behaviors before they introduce inacceptable risks.

Causality and causal inference in complex systems
In complex systems, isolating one element to study its behavior independent of the rest of the system and its interactions with other components is not effective. This is especially true when components adapt their behavior to their context. The major issue is that factors influencing the behavior of components might not be directly measurable. Assessing causal effects and finding root causes of problems in software systems is a long-standing problem, and many techniques and processes such as immutability, using a staging environment to test systems in conditions as close as possible to the real ones, A/B testing, and others have been developed to assess and improve software quality. In parallel, identifying causes and assessing causal effects in complex systems is a problem that research fields such as causal inference and causal discovery seek to address. These fields provide methods for identifying and evaluating causal effects even when the underlying data is not from randomized experiments or when confounding factors may not be measured. (Glymour, et al., 2019; MacKenzie, et al., 2018). While some causal inference methods have found their way into software engineering (Siebert 2022), determining the root cause of quality in complex software systems remains a challenging aspect.

Assurance, complexity and emergent phenomena
Coping with complexity and engineering emergent behavior is already challenging if risks are moderate. In case of high risks, high confidence is needed that some very critical emergent phenomena will not occur, or that required behavior will always occur. This is challenging and probably not feasible with the traditional assurance approach according to ISO/IEC/IEEE 15026. This approach assumes that we can collect enough evidences for building a strong argument for the claims to be assured. It is heavily applied in the context of safety in various domains and dates back to 1965 (cf. Figure 1 in (Rinehart, et al., 2017)). It is challenging to apply this approach to dynaSoS because the evidence that can be collected is generally not sufficient to build a strong argument. Many evidences can only be collected during operation. Furthermore, it is challenging to measure the strength of an argument. This is necessary in the context of certification because certification authorities and other stakeholders need to know under which conditions an argument is strong enough.

Impact of scale and complexity on engineering tasks
Building complex software systems, such as digital ecosystems like Facebook, Twitter, etc., has forced software engineering practitioners to find solutions for dealing with complexity at both technical and organizational levels (Jamshidi, et al., 2018). A key lesson is that some organizations, such as Spotify (Smite, et al., 2019), Github (Burton, et al., 2017), or Valve (Möller, et al., 2021), have adopted a less hierarchical type of structure to manage complexity. These have been shown theoretically (Barabási, et al., 2016) to improve information flow and are thought to be one of the reasons why such organizations are better at engineering complex software. However, the question of how the complexity of the system being built affects organizations, their processes, and their missions is far from being resolved (Kuusisto, 2017).

AI-Based autonomy

The AI-based autonomy of constituent systems in a dynaSoS is challenged by the different types of uncertainty that surround their runtime operation, be it related to how the autonomy is engineered or to the environment where they (inter)act. The usage of AI techniques in safety-critical systems calls for risk assessment and assurance mechanisms. In addition, as a large number of contextual elements are expected to be available in smart scenarios, leveraging opportunities for the design of context-aware behavior is not trivial. It requires a proper understanding of the context at the constituent system level and at the dynaSoS level.

Uncertainty due to autonomous and intelligent components
Uncertainty is a challenge with several facets in dynaSoS. One is the nature of the solutions that implement the autonomy of each constituent system. Autonomous systems that use data-driven software components can make their decisions based on an internal representation derived from data. Such systems have an implicit uncertainty about their functionalities (Kläs, et al., 2019). This uncertainty can stem from the type of decision model being used. Indeed, very complex decision models (such as deep neural networks) can, on the one hand, solve more complicated problems; on the other hand, they are sensitive to the problem of adversarial attacks, where a small change in their inputs can result in a drastic change in the output decision. The uncertainty can also stem from the data that was used to develop (or train) the internal decision model. When the data is not representative of the problem at hand, the learned decision model might not work well on unseen data points. Finally, the uncertainty can stem from the application scope of such systems, especially when the context changes or the internal goals are changed. An SoS that consists of multiple autonomous constituent systems must deal with the propagation of the uncertainty of the individual constituent systems to the entire SoS.

Uncertainty and changing operational environment
Uncertainty in dynaSoS also stems from the environments into which the constituent systems are inserted. These environments are open, which brings uncertainty that can range from technical issues (e.g., defects in physical devices) to organizational issues (e.g., unexpected demand fluctuations in the processes these system support), not to mention scenarios where an actor is trying to disrupt the system behavior on purpose (e.g., security attacks) (Adedeji, et al., 2020; Tavčar, et al., 2018). The environment challenges the adaptation capabilities of systems because it is hard to anticipate all adaptation scenarios at design time (Daun, et al., 2015), which makes not only the design of adaptation strategies a difficult task, but also the assurance of adaptation (Schmerl, et al., 2017), which is required in safety systems.

At the macro level, questions arise from possible unintended consequences caused by variations in the behavior of the individual constituent systems – for example, a particular constituent system that has been modified to improve its resilience (Uday, et al., 2015) – or even the modification of the current set of constituent systems in a dynaSoS – e.g., a new component joining the system can lead to unpredictable or undesirable behaviors (Tavčar, et al., 2018).

Design of context-aware behavior
The potential for implementing context-aware functionalities has increased as new and better sources are available. Sensors

are a typical example: They can provide systems with useful information about the context, including, for example, location, weather, presence of an obstacle, distances, temperature, battery level. Contextual information is not limited to sensor outputs, though. A widely accepted definition of context comes from Dey (Dey, 2001): According to him, »[c]ontext is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves«. Put like this, context can potentially be anything. Therefore, contextual information can be obtained not only from sensors, but also from any source that may provide access to the situation of entities that are relevant for a usage scenario. An essential step is to have access to good-quality contextual data, which is not always the case. In the Internet of Things (IoT), for example, high-precision indoor positioning is challenging (Kim, 2018), as is the accumulation of adequate data, context factor discovery, and filtering of contextual information (Ahlawat, et al., 2021).

Once systems have access to the context sources, they can know the context. Then the question is how context can be used to implement context-aware functionalities, such as adaptations or recommendations. The design of context-aware behavior is challenging because in smart scenarios, there are so many contextual factors available to be taken into consideration, that it is difficult to identify which of them are relevant for a certain task of interest, or how the individual contextual factors can be combined to describe a context-aware behavior (Falcão, et al., 2021).

Automated risk reasoning
A special aspect of context-aware behavior is the consideration of risks or possible harm scenarios (Feth, 2020). Risk is the combination of the likelihood of harm and the severity of that harm. It is challenging to formalize risks and measure them because of its subjective nature and the need for intersubjectivity in algorithmic decision-making. Automated risk reasoning has to be based on moral standards and raises questions related to the ethics of risk (Geisslinger, et al., 2021). Many approaches focus on the risks of a single autonomous system, like the collision risks of an autonomous vehicle. These approaches are not sufficient for dealing with risks due to emergent phenomena of dynaSoS.

Macro-level context awareness
While the design of context-aware behaviors aims at the autonomy of the constituent systems, contextual factors can also arise from the very dynamics of these systems due to emergent phenomena. This means that certain relevant context sources cannot always be provided by a single primary context source, but are rather provided by a combination of contextual factors from different sources. Strictly speaking, this issue is not limited to dynaSoS; however, the situation becomes more challenging in dynaSoS because of its openness: Constituent systems can join and leave without compromising the working of the dynaSoS, and these systems are heterogeneous. Consequently, the context sources provided by each constituent system may not be known in advance by the others. Nevertheless, to fully exploit the potential of context awareness, constituent systems should be able to understand the context that can only be described on the macro level of the dynaSoS. This raises questions such as who (if anyone) may be responsible for managing macro-level context in a dynaSoS, and how the macro-level context may affect the tasks being supported by the dynaSoS as a whole and by each of its constituent systems.

Assurance of AI and autonomy
Assurance mechanisms are used to check whether a system satisfies certain qualities of interest, such as safety and correctness. Constituent systems in a dynaSoS are autonomous due to the usage of AI and Big Data. These can, among other things, support the context awareness of these systems and subsequent self-adaptation. Implementing the assurance of self-adaptation has been considered an overarching challenge (Schmerl, et al., 2017), and there have been demands for the development of runtime assurance mechanisms (Schmerl, et al., 2017; Tavčar, et al., 2018). The design of runtime assurance processes for self-adaptive systems has not been sufficiently investigated though. Furthermore, under the assumption that we are able to perform runtime assurance, self-adaptation poses a question about the validity of the assurance cases when the context changes – in other words, reassurance must take place at runtime. The question is, which parts of an assurance case must be re-evaluated when there is a change in the context – be it the environment or the state of the system itself.

Heterogeneity and openness

Heterogeneity appears at different levels in dynaSoS. On a technical level, operational independence implies that the constituent systems might not rely upon the same technologies. On an organizational level, managerial independence implies that the organizations themselves might not be interoperable (they might use different engineering processes, have different legal frameworks, and follow different business strategies). Finding common languages and ways to interoperate is part of the digitalization journey, and we already see interoperability artifacts such as norms, standards, or reference architectures (like AUTOSAR[9], RAMI 4.0[10], or the Smart Grids Architecture Model (SGAM)[11]) that help overcome heterogeneity issues. A challenge for dynaSoS is how to transfer this existing work either to new domains that have not yet had to deal with heterogeneity issues, or how to build on this existing work to deal with future heterogeneity issues that will arise when previously separate domains, organizations, or systems need to interact.

Openness appears in dynaSoS because the constituent systems can join and leave a dynaSoS. A dynaSoS has an open door for constituent systems that support the overall mission of the dynaSoS. Furthermore, constituent systems communicate with each other to support the overall mission of the dynaSoS.

Interoperability for shared context awareness
Interoperability is a general challenge in software engineering that has different facets in different types of systems. In dynaSoS, one of these facets is how to make contextual data interoperable and provide constituent systems with shared context awareness, that is, with a common understanding of the context of a certain scenario. While constituent systems in a dynaSoS pursue their own individual goals, they also interact with each other to achieve the high-level goals of the dynaSoS. Such an interaction may take place directly – through the interfaces these systems expose to each other – or indirectly – through the ability of these systems to sense and recognize the other systems and their current state in an interaction scenario. Context plays a major role in the adaptation capabilities of the constituent systems, and as constituent systems are developed and operated independently, their abilities to sense and model the context into which they are inserted vary as well. As a consequence, different constituent systems that participate in a certain smart scenario may have different understandings of the surrounding context, which in turn may lead to poor or at least suboptimal collaboration and decision-making, in particular with respect to the high-level goals of the dynaSoS.

Heterogeneous constituent systems
Constituent systems in a dynaSoS are potentially heterogeneous in any technical dimension, including, for example, hardware, software, networks, and protocols (Singh, et al., 2018; Younan, et al., 2020; Li, et al., 2019). One thing cuts across virtually everything else though: data. Given the technical heterogeneity of constituent systems, the continuous exchange and integration of data with disparate quality are challenging. It is not trivial to propose, for example, mechanisms that can query data across different formats and structures (Diène, et al., 2020). Ideas toward modeling and meta-modeling raise questions about the adequate level of abstraction of these models – in order to prevent both oversimplification and overengineering – and bring with them computational challenges related to efficiency when we talk about large models (Uday, et al., 2015). As the details of each system are not or cannot be known in advance by the other systems, black-box integration via interface specifications is needed – which, in turn, poses a challenge for achieving interoperability (Liu, et al., 2020).

Another technical consequence of the heterogeneity of constituent systems is the problem of monitoring the high-level behavior of the SoS, which is, for example, observed in the IoT world due to the velocity, volume, and variety of IoT (Atlam, et al., 2017).

Impact of security issues
Security is frequently mentioned in the literature as major concern for SoS (Tlili, et al., 2022; Ahmed, et al., 2019; Schranz, et al., 2021; Goli, et al., 2021; Chen, et al., 2016; Ali, et al., 2021; Nazish, et al., 2018; Badr, et al., 2021; Li, et al., 2019). There are different notions of security, so we will first clarify what we mean by security. The taxonomy of dependable and secure computing by Laprie (Avizienis, et al., 2004) proposes considering security as a combination of confidentiality, integrity, and availability. Furthermore, it introduces the term »malicious faults« for faults introduced by a human with the malicious objective of causing harm to the system. We here refer to the challenge of dealing with malicious faults.

An attacker typically has many opportunities to attack a dynaSoS because many constituent systems can join a dynaSoS, and each of them can be subject to an attack. The constituent systems often communicate by exchanging information and their software is continuously being updated. These and other aspects lead to many opportunities for attacking a single constituent system. Dealing with all these possible attacks is challenging. Many attacks might be seen as not so severe from the local perspective of an operator of a constituent system, but they may have a severe impact from the global dynaSoS perspective, possibly generating emergent phenomena leading to severe losses. If the severity is really high, then we consider it as a safety issue. This means that we do not limit safety to loss of life, injury to humans, damage to property, etc.

9 https://www.autosar.org/

10 https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html

11 https://energy.ec.europa.eu/smart-grid-reference-architecture_en

Interoperability and integrity of exchanged information
In a dynaSoS, the constituent systems typically collaborate by exchanging information. This information is often truthful, that is, it may either be true or not. For instance, the information that the speed of a vehicle is not higher than a certain value X is truthful. If the information is used for making critical decisions, then high confidence is needed that the information is actually true. The level of required confidence depends on the criticality of the decision. This means that the senders of information have to somehow communicate the level of confidence so that the receivers can take this into account when making decisions or when deriving other pieces of information and related confidence information. Engineers also have to deal with this issue when they design a system where one component sends critical information to another component. The difference is that the components are known, and the engineers can reason on whether the required level of confidence is provided or in which situations it is provided. In the safety-critical domain, this communication is supported by the concept of integrity levels. This is why we refer to the integrity of exchanged information and not to confidence. There are domain-specific versions of integrity levels, like ASIL for automotive and AgPLr for agricultural machinery. This leads to interoperability challenges when components from different domains are put together. The IEEE P2851[12]»Standard for functional safety data format for interoperability within the dependability lifecycle« is working on this issue. This is also an issue for horizontal dynaSoS, but the main issues are that the engineers of a constituent system do not know how the information provided is used and that the usage defines which assumptions for generating the information are relevant for the receiver.

Distributed systems

Research on distributed systems or distributed computing deals with issues such as the tradeoff between running time and number of required computers, synchronization, byzantine errors, consensus problems, self-stabilization, or deadlocks. Distributed computing is also used in some constituent systems, even though a single computer would be possible, because it can be faster, more reliable, and/or more cost-efficient. Considering a dynaSoS and the geographic distribution of its constituent systems, distributed computing is essential for using the flexibility to execute software on available hardware resources. This flexibility can be used to minimize the required hardware resources, energy consumption, error handling, and many other criteria. Real-time constraints limit this flexibility, but 5G and other technologies push these limits. Safety is also a limiting factor because it is hard to perform all necessary safety analyses and provide related safety guarantees if the execution of software is not straightforward but highly dynamic and optimized according to many criteria.

Assured dynamic software execution in distributed systems
Safety standards for single systems such as road vehicles or agricultural machinery provide guidance for dealing with failures of execution platforms for application software. The safety requirements for the execution platform depend on the criticality of the application software. Moreover, it needs to be assured that application software of lower criticality does not affect application software of higher criticality via shared resources for their execution. Fulfilling such requirements is challenging if approaches are applied that are common in distributed computing and cloud computing. For instance, containerization enables the deployment of multiple applications using the same operating system on a single virtual machine or server. If the applications have different levels of criticality, then it is challenging to assure freedom from interference requirements. On the other hand, these approaches enable the implementation of dynamic safety mechanisms. For instance, containerization can be used to run several copies of a safety-critical application and implement fault tolerance based on voting strategies. The number of replicas can be changed at runtime depending on the current criticality of the application. This is a promising approach, but common cause failures need to be considered. The general research questions are as follows: Which conventional safety measures can be applied in the context of distributed computing and cloud computing? Which additional safety measures can be applied? How to demonstrate that all applied safety measures are at least equally effective as the conventional ones?

Continuous and innovation-driven development

Continuous and innovation-driven development is common in cloud-native and digital organizations (Ebert, et al., 2022). These approaches have been shown to improve the speed and quality of software systems. However, they are not silver bullets that can be applied overnight (Fitzgerald, et al., 2017; Ebert, et al., 2022). They are not common for safety-critical software embedded in vehicles or other physical products (Fayollas, et al., 2020). In that case, the market introduction of a product and changes after market introduction play an important role in terms of safety and liability. Laws and regulations define health and safety requirements that must be fulfilled prior to market introduction. The producer's liability obligation is excluded if the state of scientific and technical knowledge at the time the producer puts the product into circulation was not such as to enable the defect to be discovered

12 https://sagroups.ieee.org/2851/

(Bundesgesetzblatt, 1989). This may become problematic if the behavior of constituent systems is very situation-specific and the environment evolves. For instance, imagine that it was not reasonably foreseeable that people would wear masks when an autonomous system was put into circulation. This may affect the system behavior so that it becomes unsafe. Consequently, the behavior has to evolve with its environment. Continuous development of safety-critical products has this additional motivation but is particularly challenging because it is in conflict with established regulatory and certification frameworks. Besides, the independent development lifecycles of constituent systems pose another challenge for the evolution of these systems.

Certification and continuous improvement
Certification is well established for assuring critical properties such as safety. Regulations and safety standards define safety requirements. Product manufacturers implement these safety requirements and certification authorities check their fulfillment. If we consider dynaSoS, then several questions arise: Do we consider each technical system in the dynaSoS as a product, all technical systems together, or the complete dynaSoS including all non-technical elements? How to deal with the changing constituent systems that belong to a dynaSoS? How to deal with the evolution of non-technical systems? How to deal with the continuous improvement of technical systems? Only the last question is directly related to continuous engineering, but a holistic solution is required that answers all these questions together. This solution should support modularity so that manufacturers of technical systems can certify their system without knowing all possible dynaSoS to which their system may contribute in the future. Furthermore, this solution should support composability so that a certificate for the dynaSoS can be derived from the modular certificates. Certification authorities should be able to assess this mechanism for composition. Such visions are known as runtime certification (Rushby, 2008), conditional safety certificates (Schneider, 2014), or dynamic risk management (Schneider, et al., 2018). One challenge that makes it hard to realize them is how to align them with a regulatory framework or adapt the regulatory framework. For instance, the European Machinery Directive defines that machinery can mean assemblies of machinery. This makes it possible to consider a fleet of robots as one product that can be certified. Further investigation is required to identify the limits for interpreting the generic definition of machinery. Can a complete factory or even several factories be considered as machinery? How flexible is the definition with respect to changing numbers and types of machinery? What changes with the new machinery regulation? Other challenges relate to the technical implementation. So far, there are generic concepts and related implementations. How to tailor them to a particular domain for a small dynaSoS that is built from scratch? How to introduce this concept incrementally into existing systems of systems?

Response time addressing asynchrony of continuous developments
One of the challenges of continuous development in dynaSoS is managing the development and integration of multiple components with different lifecycles and development speeds. For example, different subsystems may be developed by different teams, and changes in one subsystem may affect the others. As systems continue to evolve, new requirements and changes may be introduced at different times, leading to asynchrony in the development process (Bauer, et al., 2019; Theobald, et al., 2018). This can result in significant delays and inefficiencies, especially in safety-critical systems where a delay in response time can have severe consequences.

One significant challenge is ensuring that the development teams responsible for different components of the system can work together effectively. Collaboration is critical to ensure that changes are integrated smoothly and that there are no conflicts between different components. This requires a clear understanding of the system's architecture and an effective communication strategy that enables teams to work together seamlessly (Tisi, et al., 2021; McDermott, et al., 2020).

Another challenge is ensuring that changes to the system do not compromise its safety or performance. Asynchrony in development can lead to unforeseen consequences, and changes to one component can affect the behavior of the entire system. Therefore, it is essential to have an effective change management strategy that enables changes to be integrated into the system while ensuring that safety and performance are not compromised.

Engineering

These challenges are mostly related to the dimension »Engineering«", but the changes are rooted in the evolution toward larger-scale and more complex interconnected systems (i.e., horizontal, global dynaSoS).

Digitalization and the digital economy are leading to a number of organizational and business changes. The way products are developed, sold, and used today is different from what was possible before the digital transformation. For instance, new versions of a software product can be shipped into production several time a day, or randomized experiments (so-called A/B testing) can be run in order to determine what features are most desirable or profitable.

In terms of culture and organization, companies that are at the forefront of digitalization are facing novel challenges, and they are proposing novel solutions, such as "T-shaped" profiles[13], cross-functional teams, lightweight and agile processes, flat hierarchies, everything as a service, etc. Many have also published handbooks promoting their values, vision, and culture[14]. Their experience (good or bad) is going to be evaluated and adapted by other domains as they advance on the path of digitalization.

Furthermore, the current Covid-19 crisis has forced many organizations to switch to remote work and make use of digital platforms. Although we can only see the very first effects of such crises on the state of digitalization (see, for example, a case study from Chemnitz University (Skulmowski, et al., 2020)), it is believed to have accelerated digitalization and will have an impact on the way people work and what users expect in terms of digital services (LaBerge, et al., 2020).

Shifts in culture, mindset, and organization appear to co-evolve with societal trends and regulations. The climate change crisis and related political engagement are pushing organizations to move toward a more circular economy and to take rebound effects into account (Sundberg, 2022).

For example, open source and open data are already an integral part of sustainable circular design[15] as both ease the reuse of digital assets such as software, data, or designs. However, it is less clear how the measures that will be adopted by organizations in response to the climate change crisis will in turn affect software engineering.

Shortage of skilled workers, continuous education and the need for automation and low-code solutions
In the interviews with systems and software engineering experts as well as in the interviews with domain experts (see, for example, the results of our interviews for Smart City (Brandt, et al., 2022)), both groups pointed out that skills shortage is a major challenge when it comes to pushing toward larger and more complex software systems such as dynaSoS. This problem also occurs in many other domains (Strietska-Ilina, 2008; Brunello, et al., 2019; George, et al., 2019; Amade, et al., 2021) and has been shown to be pronounced for activities related to digitalization (including IT, software engineering, and data science) (Janssen, 2022).

The Future of Jobs Report 2020 from the World Economic Forum (Zahidi, et al., 2020) estimates that half of all employees worldwide would need reskilling by 2025. The technological trends such as AI, Big Data, Data Science, or IoT, are seen as disruptive and will lead to radical changes, not only in the way we work, but also in the skills required (Li, 2022). Mitigating the risks related to skills shortage is a complex task, and several ways are being and will be explored to address this problem. For example, continuing education (and the integration of continuing education into the corporate culture) is one relevant aspect, increased automation and the application of low-code[16] applications is another one[17].

Transdisciplinary approach, fairness, diversity and inclusion
Not only are systems becoming more complex and their impact broader (see, for example, issues raised by automated decision-making (Zweig, 2019)), but requirements for the accountability of software engineers are increasing proportionally to the societal impact of the software being developed. Thus, many tech industries are investing in diversity and inclusion programs[18], partly to ensure talent acquisition and retention but also because diversity and inclusion have been shown to improve productivity and software quality (Rodríguez-Pérez, et al., 2021). A transdisciplinary approach is already at the heart of research into complex systems and is also recognized by researchers in software engineering (Méndez Fernández, et al., 2019). However, in practice, very little is known about this topic.

Heterogeneous organizations
Constituent systems in a dynaSoS are designed, managed, and operated independent of each other, notwithstanding the fact that the organizations behind them should collaborate in order to foster synergies and optimize the fulfillment of the overarching goals of the dynaSoS. Achieving such synergy is challenging though. Different organizations may use different processes, techniques, and tools to develop and maintain their constituent systems (Liu, et al., 2020). Whenever coordination is required, issues are expected to emerge from conflicts between the particular goals pursued by each party (which may

13 T-shaped persons are people who are capable of being a specialist in one domain, but have enough generalist knowledge to enable them to collaborate with other experts across disciplines (Johnston, 1978). See also https://www.incose-cc.org/blog/the-t-shaped-engineer

14 Several examples are discussed by practitioners; for one example among many, see: https://techbeacon.com/app-dev-testing/lessons-7-highly-successful-software-engineering-cultures

15 See https://opencircularity.info or https://oscedays.org/

16 Low-code (sometimes called no-code) is a software development approach that aims at minimizing the amount of coding by using a collection of ready-to-go UI components, boilerplate scripts, or visual workflow automation tools (such as the data analytics platform KNIME https://www.knime.com/blog/low-code-data-science-is-the-future or Google blockly https://developers.google.com/blockly).

17 https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023

18 https://corporate.zalando.com/de/dobetter-diversitaets-inklusionsbericht-2022 be reflected, for example, in how they prioritize their activities and which technology they use) and the overarching goals of the dynaSoS. In addition to that, the organizational heterogeneity is technically reflected in the different data formats and standards as well as the quality adopted and required by each system. From an organizational point of view, the question is how strategies to improve the overall SoS can be implemented when the distribution of costs and benefits among participants is not clear (Uday, et al., 2015).

Data acquisition, exchange, and interoperability
Different data sources may provide information about the same semantic concepts, but in different formats and resolutions (a simple example would be temperature in °C/day versus temperature in °F/week). It is also possible that the data sources provide their information in different modalities (e.g., text, images, videos, audio, …). Merging different data sources is known as data fusion (Bleiholder, et al., 2009). Recent advances in deep learning have improved multimodal data fusion techniques, but this research and its applications are still in a preliminary stage (Gao, et al., 2020).

Data-driven decision systems, such as ML-based systems, almost always require some form of data pre-processing that is context- and application-dependent. Part of this pre-processing is called feature engineering. A typical example from natural language processing would be the conversion of words into vectors using methods such as Word2Vec. These vectors (called features) can be used to compute the similarity between pieces of text. The challenge here is that the features created depend on the goal of the system that created them and on the data used to create them. Tools such as feature stores and pre-trained models are already being used in practice, but very little is known about the usability of these features in other contexts. Furthermore, there are also privacy issues, and it is not yet fully understood to what extent protected attributes can be reconstructed using features developed by another constituent system.

Data quality, lifecycle management and data governance
Challenges related to data-intensive systems include aspects of data protection, data usage control, and data sovereignty: who has what kind of data, who has access for what purpose, and how is the data being used and exploited? The issue about who owns the data that is collected and exchanged has not be solved, and there is a legal vacuum about the topic (Singh, et al., 2018). Beyond the legal implications, there are also ethical concerns, especially when personal data is involved. These need to be tackled by the different organizations taking parts in the dynaSoS and require the definition of a data governance strategy on the dynaSoS level and the alignment of strategies and processes in place.

Another key challenge related to data-intensive systems is data quality. Data preparation is claimed to consume much of the time[19] of any data-driven project. Data quality is a broad topic, and several standards have been devised to define it (e.g., ISO 8000, ISO/IEC 25012, or ISO/IEC 5259). There are basically two main sources of problems that can cause poor data quality. The first one is related to the way data is acquired, transformed, and stored. For example, the resolution of the initial measurements might be insufficient, raw data might be aggregated and transformed losing some information on the way, and storage capacity may impose limitations on how much data can be kept and how good its resolution can be. This first set of challenges is common to every software system, whether data-intensive or not, but increases with the scale and complexity of the software system.

The second aspect relates to the intended use of the data and is particularly relevant to data-driven decision systems. Data captures only part of reality. This means that some relevant information may be missing. More importantly, a part of reality that is irrelevant to the problem at hand may be captured by the data. Some of the latest AI applications have recently been accused of not being fair and of perpetuating stereotypes. One example is the Word2Vec method (cited above), which transforms words into vectors. When trained on available data from news articles, it has been shown to reproduce gender stereotypes (Bolukbasi, et al., 2016). Research on algorithmic fairness, accountability, and transparency has grown in recent years[20], providing initial tools to avoid this kind of undesirable behavior in data-driven systems and also helping to identify and address related data quality issues, which have in turn been implemented in major cloud providers[21]. However, these problems are far from being solved and will be relevant for dynaSoS.

Summary

We organized 24 research challenges for dynaSoS into eight clusters based on the research architecture. In the next chapter, we will detail the six research recommendations we derived from these challenges.

19 Here, the numbers vary between 50% and 80% depending on the sources.

20 See, for example, work done and published at these conferences https://facctconference.org/

21 See, for instance, https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-fairness-and-explainability.html or https://learn.microsoft.com/en-us/azure/machine-learning/concept-fairness-ml

Recommendations, research directions and roadmap

In this chapter, we detail our six research recommendations for dynaSoS. Each recommendation is related to some of the research challenges introduced above. The recommendations are followed by potential research directions to be pursued. In total, we indicate 19 research directions and position them in a roadmap.

Recommendation: Value-based engineering of dynaSoS

Economic rules and regulatory constraints define which atomic dynaSoS will be developed from scratch and how existing SoS will be transformed into dynaSoS. They have to comply with the values of the affected societies or individuals and establish the right incentives for a reasonable change. Even if this open issue seems more related to sustainability science, ethics, and regulatory science than to computer science, systems and software engineers are increasingly confronted with deep societal and environmental questions (Zweig, 2019; O'Neil, 2016; Sundberg, 2022) and software engineering has a role to play in this regard. Although ethics and sustainability have long been part of the engineering curriculum, they are still optional courses and training is insufficient for practitioners (Casañ, et al., 2020; Seyff, et al., 2020). The standard series IEEE 7000 already addresses this issue, but there is a lack of concrete methods and tools for implementing its requirements. In particular, methods and tools that allow us to investigate the potential sustainability effects of software systems are still in their infancy. Accordingly, we recommend accelerating research and the development of applications at the intersection of systems and software engineering, ethics, and sustainability.

Motivating challenges

1. **Environmental and social impact:** The growing trends of connected objects, cloud computing, and ever-growing digitalization open up many opportunities for collecting relevant information for shaping the digital transformation of SoS and for engineering dynaSoS for a better world. It is necessary to understand and control their impact on ecological systems, societies, and individuals. Software engineering plays a key role not only in implementing the digital transformation but also in finding and assessing reasonable directions.
2. **Transdisciplinary approaches, fairness, diversity, and inclusion:** Regarding the technical part, software engineering is increasingly dealing with how people organize themselves in order to solve (complex) problems in an interdisciplinary manner. Systems thinking, critical and precise thinking, and T-shaped engineers[22] are thus becoming increasingly important.

22 https://www.incose-cc.org/blog/the-t-shaped-engineer

Research directions

The following research directions are envisioned to address this recommendation:

Digital ecosystem shaping and requirements engineering for sustainable dynaSoS
As already stated above, software engineering plays a key role in understanding how to evolve SoS toward a sustainable world. In particular, requirements engineering is essential in this regard but still in its infancy when it comes to large heterogenous groups of stakeholders with conflicting interests. We recommend increasing efforts regarding the research, education, and development of approaches dealing with this issue. A promising research direction in this regard is given by digital ecosystem shaping. Digital ecosystem shaping helps to understand possible value streams and shape the business models of the organizations involved and related use cases of technical systems. A recommended future research direction is to enhance this approach with respect to damage streams and impact on ecological and social systems.

Ethics of decision-making processes in dynaSoS
Digital ecosystem shaping abstracts from the concrete behavior of technical systems and their decision.making. Algorithmic decision-making systems (Hauer, et al., 2021; Castelluccia, et al., 2019) (such as credit scoring, news filter and ranking algorithms, etc.) have brought many ethical questions to the public space and have triggered novel research involving ethics, regulatory science, and software-related science including AI. Even though many novel results have been achieved, the challenges brought by dynaSoS concerning this topic are far from being solved. The ongoing work will foreseeably continue in the near future and requires further support to address dynaSoS due to complex interdependencies between the decision-making processes of different technical systems and humans.

Critical precise systems thinking and virtual engineering
Understanding and engineering complex decision-making processes is hardly possible without simulating them in a virtual world. In the context of virtual engineering, many methodologies and tools have been developed, but applying them in the context of dynaSoS requires many systems engineering skills, such as critical and precise systems thinking (Arnold, et al., 2015). It is essential to understand dynaSoS as a whole and to reflect on possible evolutions. Critical and precise thinking is essential for avoiding fallacies in reasoning, including those related to ethics such as the naturalistic fallacy. The problems engineers are facing in terms of the sustainability of current SoS today cannot be solved at the same level of thinking they applied when they created them.

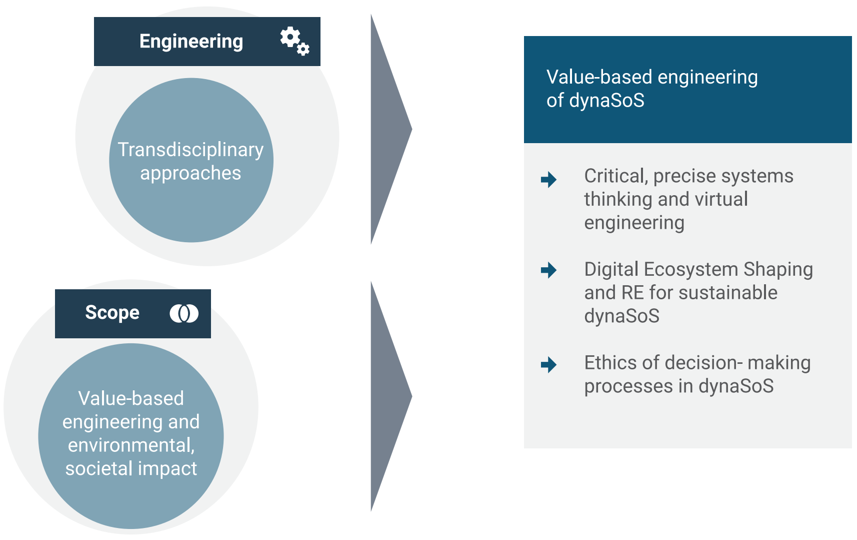

Engineering
Transdisciplinary approaches
Scope
Value-based engineering and environmental, societal impact
Value-based engineering of dynaSoS
Critical, precise systems thinking and virtual engineering
Digital Ecosystem Shaping and RE for sustainable dynaSoS
Ethics of decision- making processes in dynaSoS

Figure 13. Clustering of challenges for value-based engineering of dynaSoS and related research directions

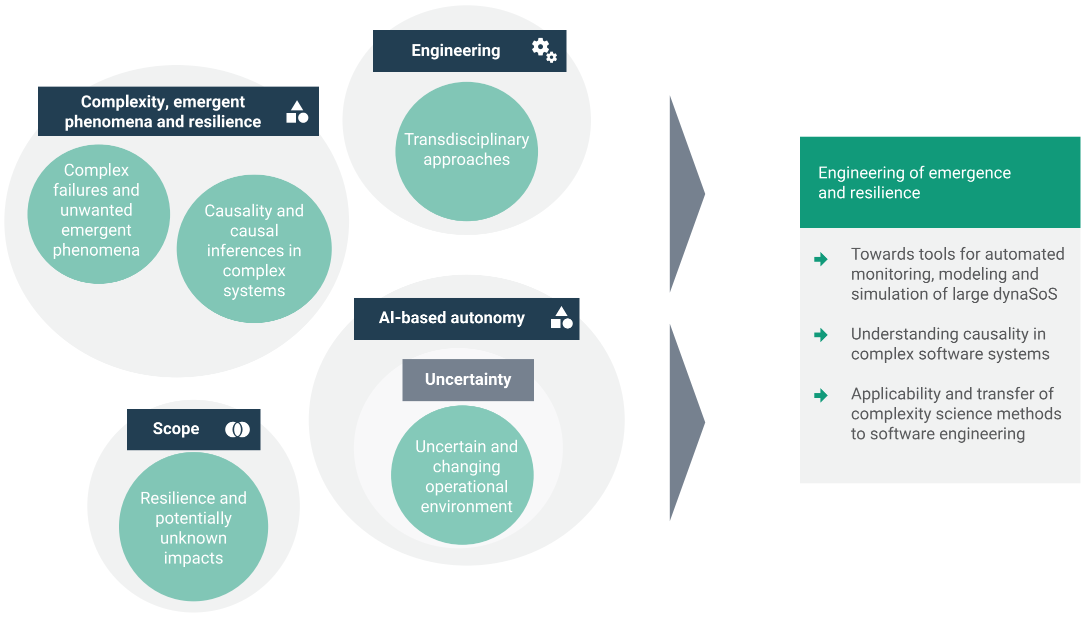

Engineering
Transdisciplinary approaches
Complexity, emergent phenomena and resilience
Complex failures and unwanted emergent phenomena
Causality and causal inferences in complex systems
AI-based autonomy
Uncertainty
Uncertain and changing operational environment
Scope
Resilience and potentially unknown impacts
Engineering of emergence and resilience
Towards tools for automated monitoring, modeling and simulation of large dynaSoS
Understanding causality in complex software systems
Applicability and transfer of complexity science methods to software engineering

Figure 14. Clustering of challenges for engineering emergence and resilience, and related research directions.

Recommendation: Engineering of emergence and resilience

DynaSoS are intrinsically complex, and their behaviors depend on complex non-linear interactions at multiple scales. Engineering such complex systems requires an appropriate mindset and suitable methods and tools. The most recent advances in complexity and complex network sciences usually find their application in software engineering. For example, understanding how information flows in complex networks such as small worlds has led some companies to change their organizations to flatter structures in order to improve communication and value flows (see, for example, the keynote by James Lewis about complexity and teams topologies[23]). However, it is very unclear how ideas from complexity science percolate into software engineering. Apart from different languages, the challenge is that complexity science studies emergent phenomena through the lens of models describing complex systems as a whole, taking a macro-level view of the problem. Software engineering, on the other hand, is concerned with the design of components that are part of complex systems, and has to deal with situations where the overall view of the whole is only partially available. Our recommendation is to continue supporting transdisciplinary projects in order to transfer methods from basic research on complex systems (such as non-linear physics, network science, or social sciences) to the field of software engineering.

Motivating challenges

The main motivating challenges for this recommendation come from the fact that dynaSoS are complex systems: They are by definition composed of many, potentially heterogeneous interacting systems, each developed by a different organization.

1. **Complex failures and unwanted emergent phenomena:** The behavior of dynaSoS is governed by intricate interactions that take place at different scales. Therefore, it is almost impossible to anticipate all the interactions between all the parts, and complex failures will inevitably happen.
2. **Causality and causal inference in complex systems:** A related challenge concerns the discovery, understanding, and analysis of causal effects in complex systems, where causal effects have different time scales and might be highly non-linear.

23 https://www.youtube.com/watch?v=_mYlSMepTGw

3. **Transdisciplinary approaches, fairness, diversity, and inclusion:** It is unclear how state-of-the-art techniques from complex systems sciences can infuse software engineering practice.
4. **Uncertain and changing operational environment:** Furthermore, dynaSoS are operating in open and dynamic environments, which makes it harder to anticipate all operating conditions.
5. **Resilience and potentially unknown impacts:** Finally, dynaSoS are large-scale systems, whose impact on their environment (be it societal or environmental) and feedback from their environment are largely unknown and difficult to assess.

Research directions

The following research directions are envisioned to address this recommendation:

Applicability and transfer of complexity sciences methods to software engineering
Dealing with large-scale complex systems is not a problem faced by software engineering alone. Indeed, over the years, other research domains have developed methods and tools to model, understand, and control emergent phenomena. Complexity sciences encompass fields of science such as network science, sociophysics, or econophysics[24]. Many different methods and tools have been developed, such as network analysis (including novel deep learning approaches for graphs), synergetics and non-linear dynamics, or agent-based modeling and simulations. However, very few of these techniques have found broad application in software engineering practice (see, for instance, (Mascardi, et al., 2019) or (Snafucatchers consortium, 2017)) and are not part of the classical software engineering curriculum yet.

One research direction to pursue is to understand where the bottleneck between complexity sciences and software engineering practice comes from and to develop methods, curricula, and interdisciplinary projects to assess the potential and impact of complexity sciences on software engineering practice.

Understanding causality in complex software systems
The intricate networks of interactions in complex systems and the increased involvement of cyber-physical systems make it difficult to perform randomized experiments (see, for example, (Mattos, et al., 2022) in the automotive domain) and increasingly requires reliance on observational data to understand causality and on the application of the relatively new techniques from statistical causal inference (Siebert, 2022). Classical statistical analysis (including machine learning) have been applied in software engineering for a long time. However, these have a major drawback, as they cannot distinguish between causal effect and spurious correlations. The data science, machine learning, and AI communities have recently recognized that these limitations undermine the properties of the developed systems such as fairness and explainability but also reliability and sustainability (Pearl, 2019; Schölkopf, 2019). Work is under way in the Data Science community (from both the academic[25] and the industry side[26]). One of the research directions to follow is to promote a shift in practice from correlational analysis to more causal inference methods as well as to promote the transfer of basic research (such as that Department of Empirical Inference of the Max Planck Institute for Intelligent Systems, see https://ei.is.mpg.de/) to applied research and software engineering practice.

Toward tools for automated monitoring, modeling, and simulation of large dynaSoS
The previously mentioned research directions focus more on changing the culture and the methods. It is clear that these changes will not be effective if the right tools are not developed in parallel. First, in order to observe emergent phenomena, it is necessary to monitor the system at different scales and to be able to link these observations to each other in order to understand how an event in a given component at a given time might impact the entire system later (Schlossnagle, 2017). Large complex software systems, even if properly monitored, are often not able to observe emergent phenomena simply because it is difficult to link events that are often observed at different scales by different systems or organizations. Second, modeling complex software systems requires coupling different models and simulation tools (Siebert, et al., 2010). In this case, co-simulation methods can help to reuse existing models and simulation software (Paris, et al., 2019). However, modeling and analyzing emergent phenomena still requires a level of expertise not mastered by all software engineering practitioners. One research direction to follow is to improve the ease of analyzing and modeling complex systems in order to understand and control emergent phenomena. A starting point could be the development of semi-automated monitoring and/or modeling tools for large complex software systems.

24 See an overview at https://www.art-sciencefactory.com/complexity-map_feb09.html

25 CauSES – Approaches to causation in the social and natural sciences and their implications for theory building in sustainability science https://causes.seslink.org/

26 See for instance https://www.causalscience.org/meeting/programme/programme-2022/ or https://fairness.causalai.net/

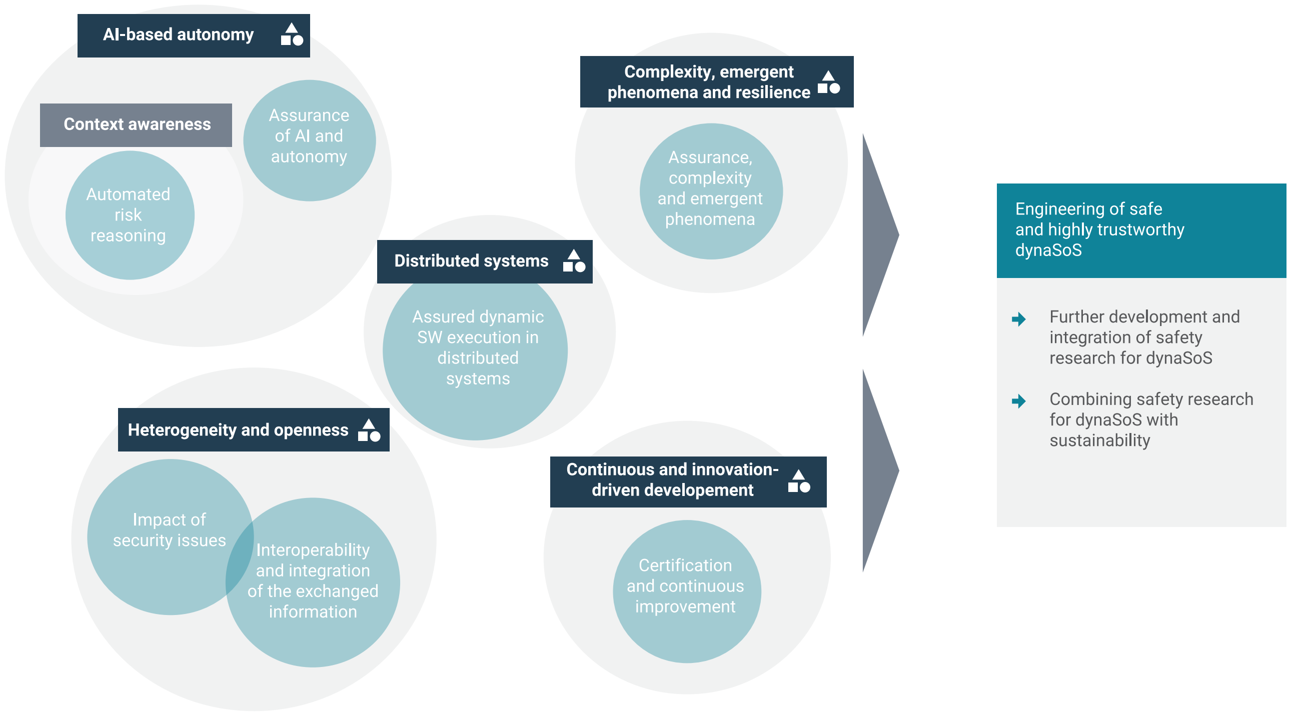

AI-based autonomy
Context awareness
Automated risk reasoning
Assurance of AI and autonomy
Complexity, emergent phenomena and resilience
Assurance, complexity and emergent phenomena
Distributed systems
Assured dynamic SW execution in distributed systems
Heterogeneity and openness
Impact of security issues
Interoperability and integration of the exchanged information
Continuous and innovation-driven developement
Certification and continuous improvement
Engineering of safe and highly trustworthy dynaSoS
Further development and integration of safety research for dynaSoS
Combining safety research for dynaSoS with sustainability

Figure 15. Clustering of challenges for engineering safe and highly trustworthy dynaSoS and related research directions.

Recommendation: Engineering of safe and highly trustworthy dynaSoS

On the one hand, the digital transformation of existing SoS into dynaSoS is indispensable to address ecological risks such as climate change. On the other hand, it introduces complexity and other properties that are in conflict with famous principles for avoiding and mitigating risks, such as »Keep it simple, stupid!« (KISS principle). Safety research already provides some approaches for dealing with this conflict. We recommend supporting the further development of these approaches, their integration, and their application with respect to other trustworthiness aspects. This is particularly relevant for small atomic dynaSoS. To address larger and holistic dynaSoS, we recommend enhancing the scope of safety research and integrating it with research from sustainability and complexity science.

Motivating challenges

1. **Automated risk reasoning:** Automated reasoning about trustworthiness is needed for two reasons. Technical systems in a dynaSoS make decisions that have an impact on safety and other trustworthiness aspects. Accordingly, algorithmic decision-making has to take different kinds of risks into account. This is challenging because it is hard to formalize risks and related risk reasoning.
2. **Interoperability and integrity of the exchanged information:** In a dynaSoS, the constituent systems typically collaborate to control risks. They exchange information that is relevant for estimating, assessing, and controlling risks. The level of confidence in the correctness of information may vary strongly for various reasons. In order to take this into account, approaches are required to model and analyze the integrity of information.
3. **Impact of security issues:** Constituent systems in a dynaSoS have to be open in order to collaborate. This leads to a large attack surface. Furthermore, complex interdependencies between malicious faults and other kinds of faults may exist because the constituent systems interact with each other in a complex way. Incorrect behavior of constituent systems may lead to emergent phenomena that have an impact on safety, availability, reliability, and other trustworthiness aspects.
4. **Certification and continuous improvement:** The evolution of a dynaSoS or its operating context can introduce new risks or increase existing ones. For this reason, dynaSoS have to be improved continuously, but current safety regulations and certification are limited to non-evolving systems with a clearly defined usage context. In many cases, traditional re-certification would unacceptably slow down the required evolution a dynaSoS.
5. **Assured dynamic software execution in distributed systems:** In a dynaSoS, the execution of application

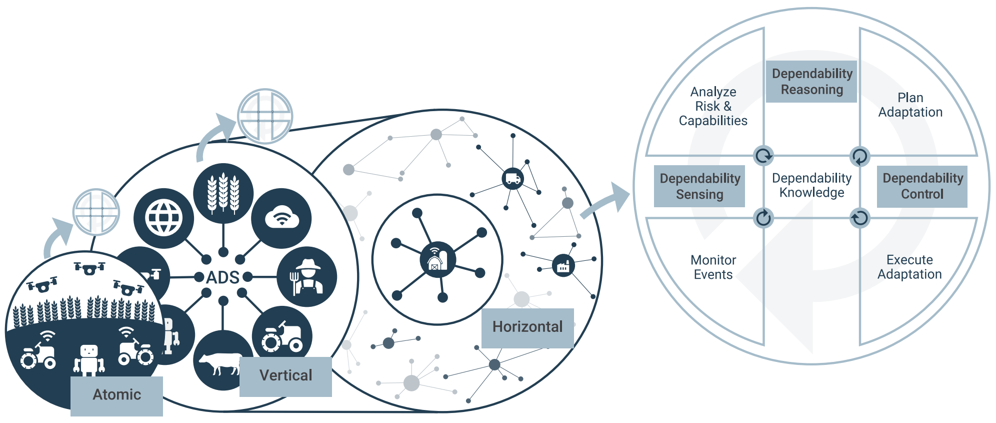


Figure 16. Extending the scope of dynamic risk management and automated risk reasoning.

software is often dynamic. The software is dynamically allocated to the execution platforms of its constituent systems and is dynamically scheduled after this allocation. This dynamism increases efficiency but makes it hard to provide good assurances. Optimization of the average case leads to large worst-case execution times. The failure rates of an application due to random hardware failures can hardly be determined. Thus, this dynamism is limited to non-critical applications. The transformation of SoS into dynaSoS often requires critical applications to also be executed dynamically (Adler, et al., 2022).

6. **Assurance of AI and autonomy:** The transformation of SoS into dynaSoS is based on Big Data and AI, which introduces autonomy of the constituent systems. On the one hand, related technical means and properties open up new opportunities for dealing with risks. On the other hand, they make it challenging to assure that the intended risk reduction is actually achieved by these means.
7. **Assurance, complexity, and emergent phenomena:** Coping with complexity and engineering emergent behavior is already challenging if the risks are not so high. In the case of high risks, high confidence is needed that some very critical emergent phenomena will not occur or that required emergent behavior will always occur.

Research directions

The following research directions are envisioned to address this recommendation:

Further development and integration of safety research for dynaSoS
Safety research already provides some approaches for dealing with these challenges. We recommend supporting the further development of these approaches, their integration, and their application with respect to other trustworthiness aspects.

Automated reasoning (cf. motivating challenge 1) about trustworthiness aspects is supported by the research around dynamic risk management (Feth, 2020). This includes the implementation of a »runtime risk manager«, as it is called in VDE-AR-E 2842-61. As illustrated in the right part of Figure 16, dynamic risk management can happen at different scopes. Standards such as ISO 21815 focus on machinery and collision risks. The application rule VDE-AR-E 2842-61 already considers shared perception and cooperative risk management in small atomic dynaSoS. We propose considering dynamic risk management also in larger vertical or horizontal dynaSoS.

Enlarging the scope of risk reasoning and the transformation of SoS into dynaSoS requires critical applications to also be executed dynamically (cf. motivating challenge 5). As illustrated in Figure 17, virtualization allows abstracting from the execution platform. This abstraction supports resource-efficient execution of applications but is challenging from an assurance perspective because dynamism complicates common cause failures analysis and other safety analyses. Some solutions have been proposed (Adler, et al., 2022), but further research is needed to bridge the gap from abstract concepts to concrete solutions. The objective of this research is for the platform to use different kinds and amounts of resources depending on the required level of safety integrity. As illustrated in Figure 17,

27 https://www.iese.fraunhofer.de/blog/zertifizierung-4-0/
28 https://www.din.de/de/ueber-normen-und-standards/smart-standards

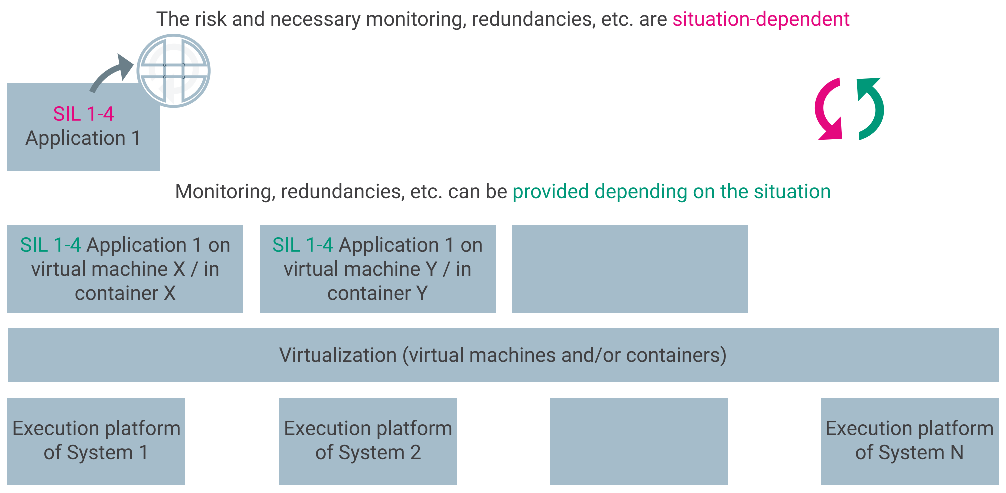

The risk and necessary monitoring, redundancies, etc. are situation-dependent
SIL 1-4 Application 1
Monitoring, redundancies, etc. can be provided depending on the situation
SIL 1-4 Application 1 on virtual machine X / in container X
SIL 1-4 Application 1 on virtual machine Y / in container Y
Virtualization (virtual machines and/or containers)
Execution platform of System 1
Execution platform of System 2
Execution platform of System N

Figure 17. Combing dynamic software execution (»SIL 4 cloud«) with dynamic risk management.

such a solution can be combined with risk reasoning on the application level. If the application identifies a change of risk in the current situation, the virtualization layer can adapt the mapping to the required resources.

A modular approach is required to flexibly combine the interaction among different applications, among different platforms, and among applications and platforms. Furthermore, the complete set of causes and effects need to be considered together. Causes should include conventional software and hardware faults but also malicious faults and faults of data-driven models. Effects should include the classical combination of reliability, availability, maintainability, and safety (RAMS), but also other trustworthiness aspects such as fairness. An example of a modular approach are (Executable) Digital Dependability Identities (Koorosh, et al., 2022). In order to deal with the evolution of dynaSoS and enable continuous trustworthiness engineering, this approach can be combined with the concept of DevOps. An example is SafeOps (Fayollas, et al., 2020).

These approaches need to be combined with solutions for handling the risks of autonomous constituent systems and related AI-based perception. This includes, for instance, solutions for measuring and handling uncertainty (Groß, et al., 2022). Furthermore, this includes assurance cases for AI and autonomous systems such as SACE, AMLAS from the Assuring Autonomy International Programme, or work to measure the strength of assurance arguments (Bloomfield, et al., 2022).

All this research toward **certification 4.0**[27] goes hand in hand with the digital transformation of the testing-inspection-certification (TIC) industry. The TIC industry pushes the digitalization of standards in order to automate TIC processes. Related standards are also referred to as »smart« standards[28] – standards that are machine-applicable-readable-transferable. Smart standards need to become part of digital twins and DevOps for trustworthiness.

Combining safety research for dynaSoS with sustainability research and complexity science

The safety research mentioned above focuses on technical aspects and relatively small dynaSoS. In order to support the necessary transformation of existing SoS into dynaSoS, a larger engineering scope is needed because of the complex interdependencies between different SoS. In 2019, the international collaboration community Engineering X launched a Safer Complex Systems initiative to address the issue of system interdependencies and related chain reactions if one system collapses. In this case, safety science already considers a very large geographic scope, but the scope is limited to the safety of the current population. In contrast to that, sustainability also takes into account the safety of future generations. In this regard, sustainability enhances safety. Accordingly, it makes sense to investigate whether the engineering approach for safety can be extended to include sustainability. Some initiatives and proposals already hint at this idea, such as the working group »Safety of the Environment« of the Safety-Critical Systems Club[29] or the paper »Global warming and system safety« (Jones, 2022). While this a reasonable research direction from a safety perspective, the primary challenge belongs to the field of sustainability science, and complexity science probably offers more solutions for coping with this complex challenge than safety science. However, sustainability science can benefit from safety science by using its lessons learned in regulation and certification. Accordingly, we recommend bringing these three research fields together.

29 https://scsc.uk/ge

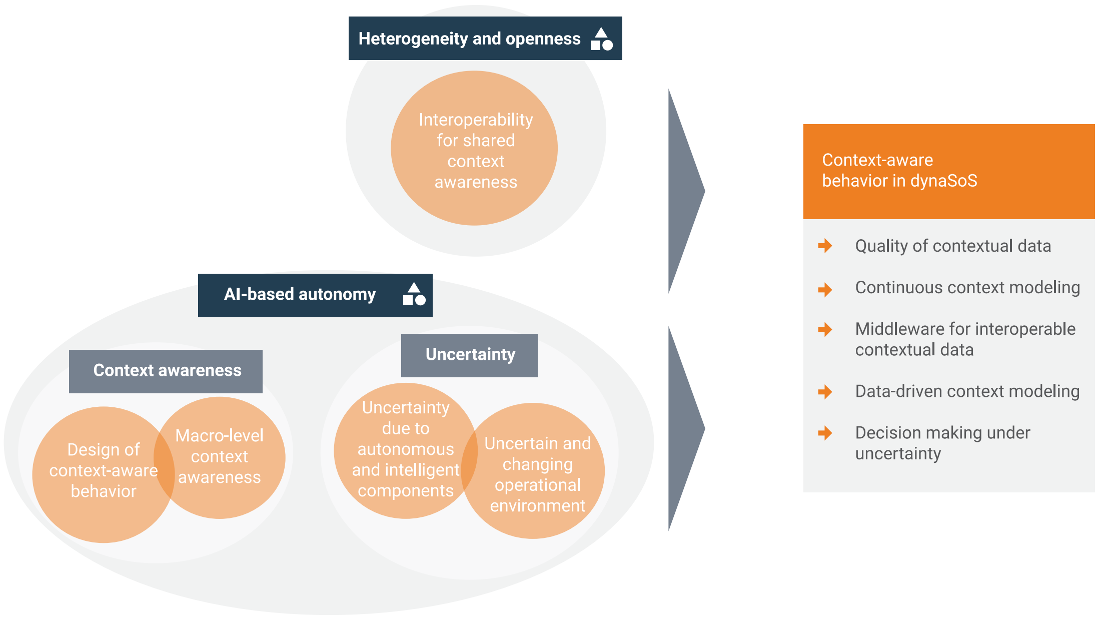

Heterogeneity and openness
Interoperability for shared context awareness
AI-based autonomy
Context awareness
Design of context-aware behavior
Macro-level context awareness
Uncertainty
Uncertainty due to autonomous and intelligent components
Uncertain and changing operational environment
Context-aware behavior in dynaSoS
Quality of contextual data
Continuous context modeling
Middleware for interoperable contextual data
Data-driven context modeling
Decision making under uncertainty

Figure 18. Clustering of challenges for context-aware behavior and related research directions.

Recommendation: Context-aware behavior in dynaSoS

Usage scenarios for dynamic systems of systems are often pictured as smart scenarios, including, for example, smart farming, smart health, smart mobility, and smart city, as illustrated in our first report (Groen, et al., 2022). The »smartness« of these scenarios is perceived by no one but humans, and context awareness is often the thing behind such smart behaviors (Pinheiro, et al., 2018). Therefore, the realization of the systems of the future requires investment to overcome current engineering challenges related to context-aware systems, which are amplified even more considering the characteristics of dynaSoS. We recommend further research in the field of software engineering of context-aware systems, covering both design time and runtime aspects.

Motivating challenges

1. **Uncertainty and changing operational environment:** The constituent systems of a dynaSoS are deployed in open and dynamic environments, which makes it harder to anticipate all relevant contexts that may influence their operation and their system-supported user tasks.
2. **Uncertainty due to autonomous and intelligent components:** Autonomous systems that use data-driven software components make their decisions based on an internal representation derived from data. Such systems have an implicit uncertainty regarding their functionalities.
3. **Design of context-aware behavior:** Even considering our increasing ability to sense the context (i.e., to identify WHAT context is), it is still rather difficult to derive context-aware functionalities (i.e., to identify HOW context influences or can influence tasks).
4. **Macro-level context awareness:** Context may not influence only the constituent systems but also the system of systems as a whole. It is therefore important to understand the role of context at the macro level of a system of systems.
5. **Interoperability for shared context awareness:** When constituent systems cooperate to achieve a high-level (SoS) goal, their partial understanding of the context hinders efficient collaboration. Shared context understanding calls for improved interoperability.

Research directions

The following research directions are envisioned to follow this recommendation:

Data-driven context modeling
The elicitation of context-aware functionalities can be improved through data-driven context modeling. Context modeling

refers to the activities involved in creating conceptual models of the context that express the relationships between the context and system-supported tasks of interest, aiming at providing engineers with better support to devise context-aware functionalities. The implementation of these data-driven context modeling approaches requires research on algorithms to support the analysis of context and its influence on tasks. Figure 19 illustrates the contrast between two companies, where only one of them uses data-driven context modeling. Early initiatives in this direction can be found in (Rodrigues, et al., 2019; Knauss, et al., 2016; Falcão, et al., 2022). It is necessary to further investigate, develop, and test different algorithms that specifically analyze contextual data and derive supporting knowledge to help requirements engineers identify novel context-aware functionalities.

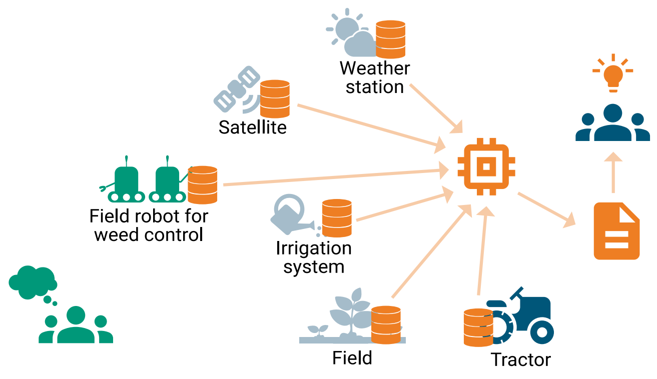


Figure 19. While engineers in the »Green Company« try to figure out by themselves how context can be used to improve their field robots, »Blue Company« engineers are supported by a data-driven context-modeling approach.

In addition to the development of these algorithms, it is also necessary to carry out research on how to represent the knowledge gained through them. In other words, it is necessary to develop and evaluate context model representations (i.e., context meta-models) to support the elicitation of context-aware functionalities. Such representations must be described formally in order to fit automated context-modeling approaches, and must be expressive enough to cover a broad range of systems, independent of their domains. Initial steps in this direction can be found, for example, in (Falcão, et al., 2022). However, there is a need for more research on the foundations of context model representations to further develop their expressiveness, as well as empirical research to validate them. Moreover, it is necessary to fill the gap between the output of the algorithms and the context model representations by providing tool support to software engineers to generate the concrete context models (initial results in this area can be found in (Falcão, et al., 2022).

Continuous context modeling
Data collection is required to enable any data-driven approach. Once the data processing algorithms and modeling tools are in place, most of the effort is shifted to the data collection step. Therefore, investing in automated strategies for data collection will improve the efficiency of engineering context-aware functionalities. Once data collection is automated, the way will then be paved for the implementation of continuous context modeling, where context-modeling activities, which nowadays are performed at design time, will happen at runtime. Continuous context modeling will enable software engineers to discover innovative smart behaviors faster by helping them learn about the opportunities provided by the dynamic environment as soon as they emerge from the contextual data being continuously collected and fed into context-modeling tools.

Quality of contextual data
Based on initial contextual data or information, it is possible to derive new contextual elements. Consider, for example, a smart mobility solution that collects, among other data, the location of the user and the vehicles. A trivial example is the derivation of the contextual element distance (see Figure 20). This derivation is done through an operation that someone has to implement.

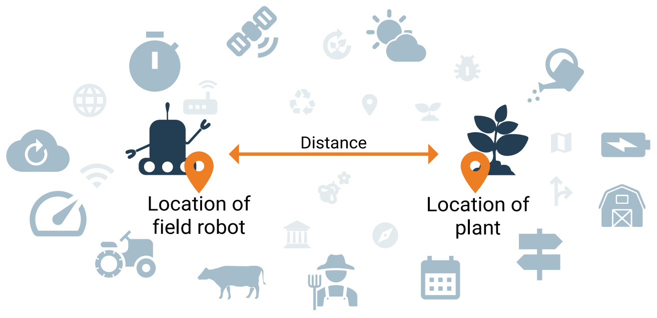


Figure 20. Derivation of the contextual element »distance« from two other contextual elements. When there are several contextual elements, deriving new contextual elements may not be trivial.

A comprehensive catalog of operators can play a decisive role in the definition of context-aware functionalities. In a scenario with dozens of contextual elements and multiple systems generating data that are relevant for the system-supported tasks, the ability to derive new contextual elements from primary ones can provide system designers with shortcuts to better understand how the context may influence system-supported tasks. As the possession of such a catalog would put companies who have it in an advantageous position compared to those who do not have it, we see the possibility for research to play a prominent role in fostering the development of an open catalog of contextual operators. These operators take contextual elements as input and also produce contextual elements. Therefore, it is also necessary to maintain an open vocabulary of contextual element types. Figure 21 illustrates two services, Service 1 and Service 2, that benefit from an open contextual operator catalog to derive more contextual elements.

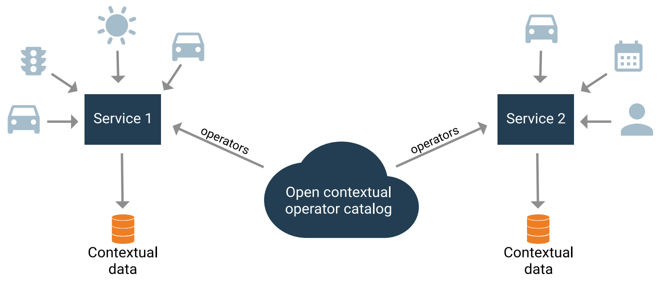


Figure 21. Two constituent systems, Service 1 and Service 2, in a smart mobility scenario collect contextual data from different sources. Both benefit from an open catalog of contextual operators to derive new contextual elements and store the corresponding contextual data. Arrows indicate data flow.

Middlewares for interoperable contextual data
We envision the development of middlewares to support the interoperability of contextual data, so that constituent systems in a dynaSoS can have a shared understanding of their (shared) context. Enabling such a shared understanding is important for at least two reasons. First, although constituent systems are operated independently, they interact with each other either directly or indirectly. The more these constituent systems are able to share a common understanding of their context, the better their chances of implementing smart behaviors at the system-of-systems level. The other reason is that all constituent systems are potential context sources. Therefore, if contextual information about each system can be made available to the others through a standardized protocol, it becomes easier for each part to provide their unique contributions to the overall context. For example, while more than one constituent system might be able to sense, by their own means, the current weather conditions, only each particular system knows about its internal operational status, which may include, for example, their energy autonomy or current limitations in its capabilities.

The design and deployment of such a middleware can lead to both edge and cloud solutions, meaning that further research can be developed in both directions. In the edge, we see some scenarios where either local infrastructure or a federated network of such a middleware hosted by the constituent systems can be provided to receive and distribute contextual information. On the other hand, the rise of 5G networks might enable cloud-based solutions for such middleware and support several smart scenarios, independent of their location and in a more cost-effective manner. Figure 22 illustrates a smart farming scenario where elements such as tractors, weed control robots, and an irrigation system send/get contextual information to/from a middleware for context.

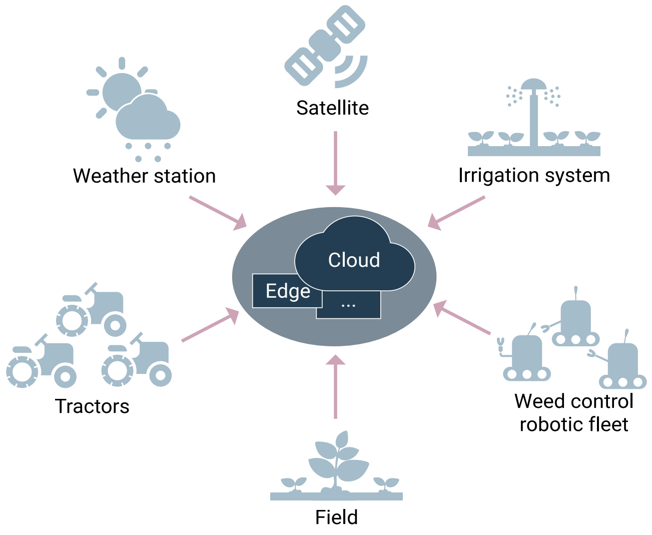


Figure 22. Machines, sensors, and other software-based elements send and retrieve context information from a middleware.

Decision-making under uncertainty
Machine learning and data-driven models are powerful means to collect relevant information about the context. A major issue with these models is that their output is subject to uncertainty. Many approaches exist to minimize uncertainty, but some residual uncertainty can hardly be avoided. DIN SPEC 92005 "Artificial Intelligence – Uncertainty Quantification in Machine Learning" is currently under development and will collect existing approaches to measure the residual uncertainty and clarify the various notions of uncertainty. Measurement also includes approaches for estimating the uncertainty during operation. Some proposals (Groß, et al., 2022) have already been made for the consideration of this uncertainty in decision-making. Further research is required to deal with dependencies of uncertainties. How can uncertainties be combined if there is a sequence of outputs with a related sequence of uncertainties? Can decision-making take advantage of this or is this not possible because of stochastic dependencies? How can uncertainties of different data-driven models be combined? Furthermore, it is worth investigating synergies between the measurement of uncertainty (Groß, et al., 2022) and the measurement of robustness (Siedel, et al., 2022) . Can these approaches be combined to make them beneficial for decision-making?

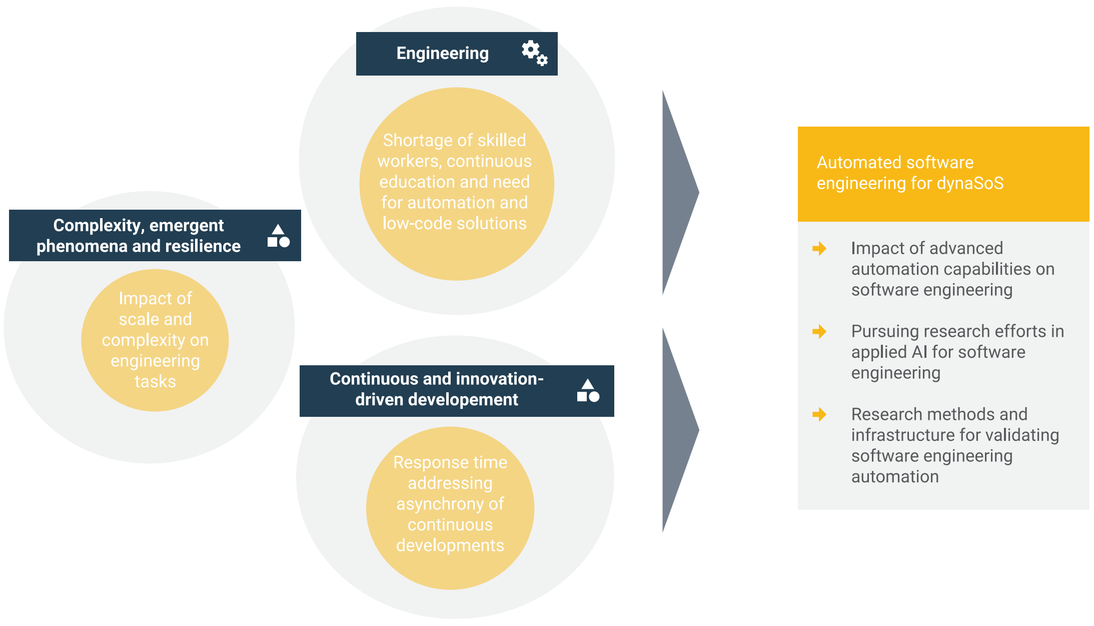

Engineering
Shortage of skilled workers, continuous education and need for automation and low-code solutions
Complexity, emergent phenomena and resilience
Impact of scale and complexity on engineering tasks
Continuous and innovation-driven developement
Response time addressing asynchrony of continuous developments
Automated software engineering for dynaSoS
Impact of advanced automation capabilities on software engineering
Pursuing research efforts in applied AI for software engineering
Research methods and infrastructure for validating software engineering automation

Figure 23. Clustering of challenges for automated software engineering and related research directions.

Recommendation: Automated software engineering for dynaSoS

The automation of software engineering tasks has been a long-standing goal for research and industry. The evolution of technologies (e.g., cloud, fog, edge, serverless, ...), the scale of systems being designed, and advances in analysis methods all influence each other and perpetuate the need for research in this area. For example, the availability of a large number of software repositories (e.g., Github) combined with new advances in code analysis based on very large deep learning models (such as OpenAI's Codex[30], Code2vec[31]) have triggered a wave of new work to automate tasks that were previously very difficult or impossible to automate (Yang, et al., 2022), such as generating code from natural language, extracting requirements from text, etc. (see, for instance, already available tools like Microsoft's Github Copilot[32], Tabnine[33], or the recent Amazon Web Service Codewhisperer[34]). One recommendation is to continue research efforts regarding the automation of software engineering tasks (including, but not limited to, AI-based approaches) while ensuring that these new methods can be empirically validated on realistic setups. Another important point is to ensure that the underlying analytics methods do not create a lock-in effect (for example, only a few organizations such as Microsoft, Nvidia, OpenAI, Facebook, etc. can train very large deep learning text models), to consider and mitigate the underlying costs in terms of power consumption, and to continue research in terms of data privacy and model explainability.

Motivating challenges

1. **Impact of scale and complexity on engineering tasks:** The scale of the systems to be designed as well as the multiplicity of stakeholders involved make engineering tasks more and more difficult. This makes it harder to automate them completely but generates demands to automate them as much as possible in order to implement them within cost constraints. Engineering tasks include verification and validation. Automated verification and validation during operation are needed to collect the required evidences for quality assurance.

30 https://openai.com/blog/openai-codex/
31 https://code2vec.org/
32 https://github.com/features/copilot
33 https://www.tabnine.com/
34 https://aws.amazon.com/codewhisperer/

2. **Response time addressing asynchrony of continuous developments:** Because dynaSoS are open heterogeneous systems continuously developed by different organizations, software engineering needs to be faster in order to react in time to unexpected events. In a safety-critical context, changes demand re-certification, which is too time-consuming for many dynaSoS use cases.
3. **Shortage of skilled workers, continuous education, and the need for automation and low-code solutions:** The technological trends underlying our dynaSoS vision such as AI, Big Data, data science, IoT, 5G & &G, cybersecurity, as well as green energy are seen as disruptive and will lead to radical changes, not only in the way we work, but also in the skills required (Li, 2022).

Research directions

The following research directions are envisioned to address this recommendation:

Pursuing efforts in research in applied AI for software engineering
The latest advances in the field of AI, in particular the development of deep learning models and representation-based learning methods, allow for applications that were previously impossible, whether in natural language processing, computational vision, or multimodal information processing (i.e., combining several modalities such as text, images, sound, video, etc.). Software engineering is, of course, taking advantage of these advances, and the applications of these AI techniques within software engineering are numerous (see, for example, sub-domains such as Artificial Intelligence for Software Engineering (AI4SE), Artificial Intelligence for IT Operations (AIOps), or Artificial Intelligence driven Development Environments (AIDE)). In the near future, new advances in deep learning will continue to infuse software engineering and new applications (especially multimodal ones) will emerge.

We recommend continuing research efforts in this direction, especially regarding multimodality. This should include the joint use of code as text but also of so-called intermediate representations (such as AST, or the SSA form, which are already in use) as well as other types of input data such as architectural diagrams, images, or even sound.

Research methods and infrastructure for validating software engineering automation
As systems to be engineered grow in size and complexity, empirically validating software engineering methods becomes harder and harder. This point was (and still is) regularly mentioned by the software engineering community (see (Briand, et al., 2017) (Méndez Fernández, et al., 2019)). As the scale, dynamics, and complexity of systems increase, it will be increasingly difficult to access and collect data on these systems (see, for example, the IBM CodeNet project[35]) and to properly set up experiments to try to falsify new methods.

It is especially important to recognize that advances in automated software engineering are likely to emerge from collaboration with industrial research (see, for instance, what Facebook is doing (Bader, et al., 2021)). This is due, on the one hand, to the increasing pressure in terms of software engineering that companies developing complex digital solutions are subject, but also to the fact that some of these companies have access to both skills and data that escape national research organizations today. In this respect, it is necessary to continue and develop the existing research and development strategy of involving industry, but it is also necessary to ask questions regarding the responsibility of the digital giants as research actors.

Impact of advanced automation capabilities on software engineering
Advances in analytics and AI are already having an impact on the way software engineering teaching and learning is done (Kästner, et al., 2020; Imai, 2022; Puryear, et al., 2022; Ernst, et al., 2022). Software engineers must increasingly rely on tools that automate laborious tasks, and must also understand in principle how these tools work and what their strengths and weaknesses are. Discussions about the impact of AI on software engineering are not unlike those of the late 1980s (before the second AI winter) (Ford, 1987; Partrdige, et al., 1987). At that time, AI was mostly synonymous with expert systems. It is worth noting that automation tools (whether AI-based or not) were integrated into software engineering tools where they made the most sense. In a sense, these are tools that need to be mastered. This is why further investigation is necessary on topics such as explainability, data protection, safety and security issues (such as adversarial attacks), and handling of causality (see Recommendation: Engineering of emergence and resilience). For this, transdisciplinary transfer is equally necessary to explore existing solutions in other fields

Recommendation: Reliable data management for dynaSoS

DynaSoS systems are data-intensive systems, where different organizations may need to share information and where value is created from data from various sources. As such, dynaSoS

35 https://developer.ibm.com/exchanges/data/all/project-codenet/

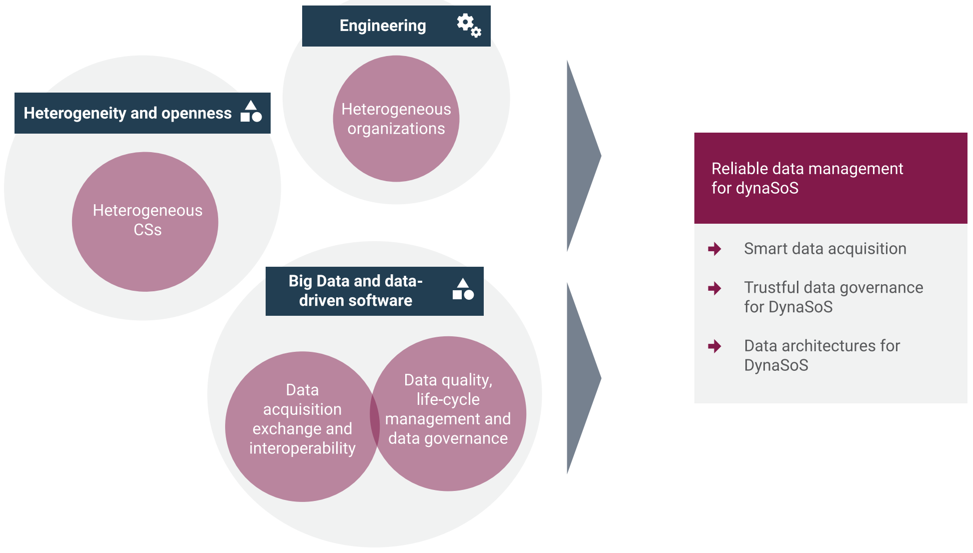

Engineering
Heterogeneous organizations
Heterogeneity and openness
Heterogeneous CSs
Big Data and data-driven software
Data acquisition exchange and interoperability
Data quality, life-cycle management and data governance
Reliable data management for dynaSoS
Smart data acquisition
Trustful data governance for DynaSoS
Data architectures for DynaSoS

Figure 24. Clustering of challenges for reliable data management and related research directions.

call for attention toward data management. »Data management« is an overarching term that covers multiple data-related aspects, such as data architecture, data acquisition, data storing, data quality, data integration, and data governance. In dynaSoS, large amounts of data are generated in diverse formats and with varying degrees of quality. This poses both technical and organizational challenges to the design, implementation, and operation of dynaSoS: On the one hand, the constituent systems need to fundamentally rely on data to implement their autonomous behaviors and smartness; on the other hand, data reliability is defied by a high level of heterogeneity and openness, which characterize dynaSoS. Therefore, we envision further research toward reliable data management for dynaSoS: There is a need to find ways of architecting, deploying, operating, and assuring reliable data for dynaSoS in order to make these systems more resilient to the open nature of the environments in which these systems operate. As potential research directions, we see investing in data acquisition methods optimized for dynaSoS; designing and developing data architectures for dynaSoS; and exploring adequate data governance designs for dynaSoS.

Motivating challenges

1. **Data acquisition, exchange, and interoperability:** First of all, because dynaSoS are data-intensive systems, they face challenges related to Big Data and require clear data management and a good governance strategy to be in place. Moreover, quantity is not a synonym of either »quality« or »suitability«. Different systems have different requirements and large volumes of data require adequate processing steps to be truly beneficial for systems.
2. **Heterogeneous constituent systems:** DynaSoS demand continuous integration of disparate data, which challenges not only the design of constituent systems but also the monitoring of high-level (i.e., SoS-level) behavior. Furthermore, since each constituent system is technically and operationally independent from the others, black-box integration is required.
3. **Heterogeneous organization:** In such diverse scenarios as we see in dynaSoS (several stakeholders, each having its own particularities – including, for example, business goals, operational procedures, technical stack), the variety in data formats and quality is enormous. Apart from the direct implication for interoperability, it is a major challenge to create and foster synergies among participants.
4. **Data quality, lifecycle management, and data governance:** Data ownership and usage rights are an open organizational issue, with legal/ethical implications.

Research directions

The following research directions are envisioned to address this recommendation:

Smart data acquisition

Research on data acquisition can help to make data management for dynaSoS more reliable as it contributes to improving the quality of the data made available. Where the amount of accessible data is larger than what systems are able to process within a reasonable amount of time, a desirable goal is to have constituent systems deal only with »as-much-as-needed« data. This requires the ability of filtering out what is not necessary (Younan, et al., 2020). The implementation of smart data filtering may be achievable through context-aware data acquisition, which should also be fully automated (Taivalsaari, et al., 2017).

From the SoS point of view, a holistic view on data acquisition (as well as data storage and data processing) may improve the overall system by deriving new domain-relevant data based on primary data sources. This idea can be illustrated by already existing traffic apps: Based on individual information about the position and speed of cars, such solutions are able to derive high-level traffic information. Likewise, based on the large streams of data being generated by distributed sources deployed in physical environments, high-level information can be generated and made available for all participants, which may increase the energy efficiency of the SoS as a whole. As participants might tend to optimize their individual efficiency, it becomes necessary to architect dynaSoS in such a way that collaboration is encouraged, easy, and rewarding.

Data architectures for dynaSoS
The study of alternatives and improvements in data architectures for dynaSoS is also an important research direction to follow for reliable data management. First, appropriate architecture designs determine how quality attributes are achieved, and interoperability is a core architecture driver for dynaSoS. In some particular scenarios where participation and geographic position of the constituent systems are entirely open, even enabling unconstrained data exchange among participants is an ambitious undertaking (Tsigkanos, et al., 2019). Besides that, further research on semantic interoperability may also add to the development of adequate data architectures, as the usage of semantically annotated data facilitates neutralizing data format restrictions (Franke, et al., 2021). Second, the more open a dynaSoS is, the higher the potential of conflicting goals between constituent systems and the SoS. Therefore, research on data architectures that motivate trust and cooperation among participants is needed. In this direction, there are at least two possible ways to go: One opportunity lies in architectures based on multi-agent systems, where trust and reputation models can be developed and provide references for participants (Fang, 2021); another is to invest in data trust platforms on which constituent systems could rely to share data in a trustworthy manner.

Concerning the data storage strategy, the usage of architectures that favor data replication across constituent systems might call for further research on eventual consistency. It is necessary to invest in methods aimed at making the implementation of eventual consistency both practical and suitable for the open and heterogenic nature of dynaSoS.

Finally, further advancements in edge computing may enable more alternatives in terms of data architectures. Because constituent systems may operate in resource-constrained environments, providing these systems with full-fledged edge infrastructure is a possible approach (Tsigkanos, et al., 2019), including the capability of handling and processing data (Younan, et al., 2020). Conversely, an edge infrastructure itself often has constrained resources that require investments in its efficiency. Tools and methods focused on building edge computing solutions will be of great help for engineers and should cover different dimensions of efficiency, including, for example, storage, energy consumption, and time behavior.

Trustful data governance for dynaSoS
The legal and ethical issues raised by data ownership call for research on data governance for dynaSoS. From a purely organizational perspective, investment in legal frameworks (both general and domain-specific ones) is necessary, as pure technical enforcement can be highly challenging. Thus, whenever technical enforcement cannot suffice, an appropriate legal framework should protect stakeholders from misuse of data. Either way – be it organizational or technical –, research on software engineering may also contribute to a data governance model that facilitates trust among participants.

From a technical perspective, it should be clear for participants in a dynaSoS who owns the data and what data usage policies are in place. Therefore, research on data usage policy enforcement for distributed systems is promising to foster the development of trustful dynaSoS. Research opportunities in this direction include the advancement of technologies that embed cross-company trust in their constructs. Whenever participants cannot trust each other, mediation of data exchange through specialized components that take responsibility for managing trust can be put in place. Therefore, research on the design, development, deployment, and operation of either central or distributed trustee platforms is recommended.

Alternatively, blockchain is a technology that is regarded as a promising data management enabler in many domains, in particular for establishing trust in a technical manner. Blockchain is expected to help supply/access data for/in different devices (Miloslavskaya, et al., 2019) and tackle security and privacy concerns related to collected data in the future (Ahlawat, et al., 2021). On the other hand, it is worth noting that blockchain technology can be highly energy-demanding, in particular in large-scale systems. Yet another option that has been investigated in recent years and that combines technical and organizational elements is the design of federated data spaces to enable trusted data exchange across heterogeneous parties (Bohlen, et al., 2018).

With respect to methods and frameworks, it is recommended investing more in inter-organizational data governance, as most research has neither focused on it (Nielsen, 2017) nor explained how data ownership and control can be ensured in inter-organizational settings, both at company and individual levels. Furthermore, given the inherent complexity of dynaSoS, studies are necessary to explore and identify adequate data governance designs »for one-to-one, one-to-many, and many-to-many inter-organizational settings« (Abraham, et al., 2019).

Roadmap

We organized the proposed research directions into a roadmap that features two dimensions: time and impact. Figure 25 illustrates a detailed view of the roadmap. The colors indicate to which of the six research recommendations each direction belongs. This roadmap is based on several workshops with external and internal experts that were organized throughout the project.

The horizontal axis features the time dimension. It is used to indicate whether a certain research direction is perceived as a short-, mid-, or long-term direction to be pursued. At this point, it is important to state that the research directions are, as the title indicates, »directions« -- and therefore neither starting nor finishing lines. When a particular research direction is placed on the right side of the roadmap, this does not imply that research on the direction should be delayed, but rather that it is perceived as a long-term research topic – such research directions are expected to demand more effort.

In general, short-term directions are characterized by two things: First, they are topics where research has already been actively developed and must be further pursued toward the realization of dynaSoS. Second, some of them can be understood as corner stones, or building blocks, for others. The boundaries between the research directions are not sharp, and aspects of each direction can be found to contribute to other directions. Two such examples are the research directions »Smart data acquisition« and »Quality of contextual data«. As dynaSoS are data-intensive systems, it is natural to think about their precedence over mid-term directions such as »Data-driven context modeling« and »Continuous context modeling«.

The vertical axis features the impact dimension. Impact here refers to the potential impact of the research direction in society. It is not meant to imply that the research directions indicated in the lower area of the roadmap are less important than those in the upper area. Instead, it is supposed to mean that the perceived impact of the directions in the upper area of the roadmap is expected to be more tangible for society and industry. Research directions that feature sustainability and applied software engineering research, for example, are regarded as high-impact directions.

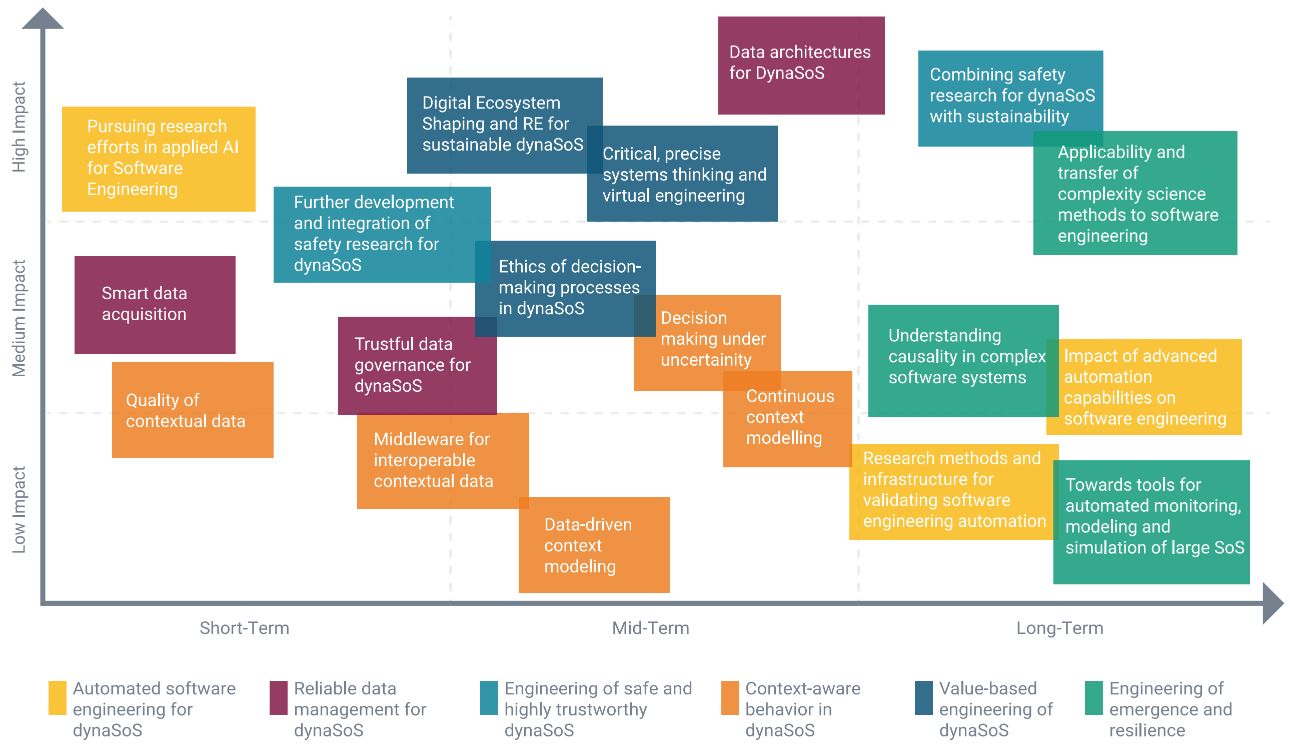

High Impact
Medium Impact
Low Impact
Short-Term
Mid-Term
Long-Term
Pursuing research efforts in applied AI for Software Engineering
Smart data acquisition
Quality of contextual data
Further development and integration of safety research for dynaSoS
Digital Ecosystem Shaping and RE for sustainable dynaSoS
Trustful data governance for dynaSoS
Ethics of decision-making processes in dynaSoS
Middleware for interoperable contextual data
Data-driven context modeling
Critical, precise systems thinking and virtual engineering
Data architectures for DynaSoS
Decision making under uncertainity
Continuous context modelling
Combining safety research for dynaSoS with sustainability
Applicability and transfer of complexity science methods to software engineering
Understanding causality in complex software systems
Impact of advanced automation capabilities on software engineering
Research methods and infrastructure for validating software engineering automation
Towards tools for automated monitoring, modeling and simulation of large SoS
Automated software engineering for dynaSoS
Reliable data management for dynaSoS
Engineering of safe and highly trustworthy dynaSoS
Context-aware behavior in dynaSoS
Value-based engineering of dynaSoS
Engineering of emergence and resilience

Figure 25. Roadmap for dynaSoS featuring 6 recommendations and their respective 19 research directions.

Summary and outlook

In this chapter, we conclude this report by reflecting on the results, discuss its limitations, and provide an outlook on future development.

Summary

In order to understand the broad role played by the topic of dynamic systems of systems, six different application areas were examined. Concrete example systems also helped us to understand the requirements for dynaSoS and were used to derive the research questions. A large number of relevant domains are affected by dynaSoS.

The research topics were then identified through a variety of interviews, expert workshops, and a broad literature review. The research architecture first helped us to better understand dynaSoS. This then allowed research challenges to be derived and sorted into clusters. Various themes from the clusters then led to recommendations when bundled together. Six core recommendations were derived for the following areas: Value-based engineering of dynaSoS; Engineering of emergence and resilience; Reliable data management for dynaSoS; Automated software engineering for dynaSoS; Context-aware behavior of dynaSoS; and Engineering of safe and highly trustworthy dynaSoS.

Each of these recommendation areas was further detailed and finally sorted into a roadmap in terms of potential (impact) and time perspective.

Reflection

We presented a research roadmap for the software engineering of trustworthy dynamic systems of systems. The roadmap we developed is based on interviews with experts and decision makers, workshops, and systematic literature reviews. The literature reviews include related roadmaps published before the project (e.g., (Northrop, et al., 2006; Carleton, et al., 2021; SafeTRANS, 2019; Kagermann, et al., 2017)) or during the project (see (Eletronic Components and Systems, 2023; INCOSE, 2023)). Some of those roadmaps focus on the future of software engineering while others focus on the future of systems engineering. Some highlight the large scale of systems while others focus on autonomy or safety and security. We focused on software engineering, dynamics, and autonomy in systems of systems, and on high trustworthiness. This focus is unique. Another differentiation concerns the people involved. Developments are driven or blocked by people who have the authority or resources to make other people follow them. Our roadmap has a clear focus on Germany. We involved authors from related German roadmaps (SafeTRANS, 2019; Kagermann, et al., 2017) and discussed existing roadmaps with stakeholders who understand what drives and what blocks envisioned innovations in particular domains. A majority of these stakeholders mentioned that serious technical challenges are often not the main obstacle to implementing visions; rather, the problem are complex, systemic issues of the current regime. We considered this main issue in our roadmap but focused on technical, software-related challenges.

Limitations

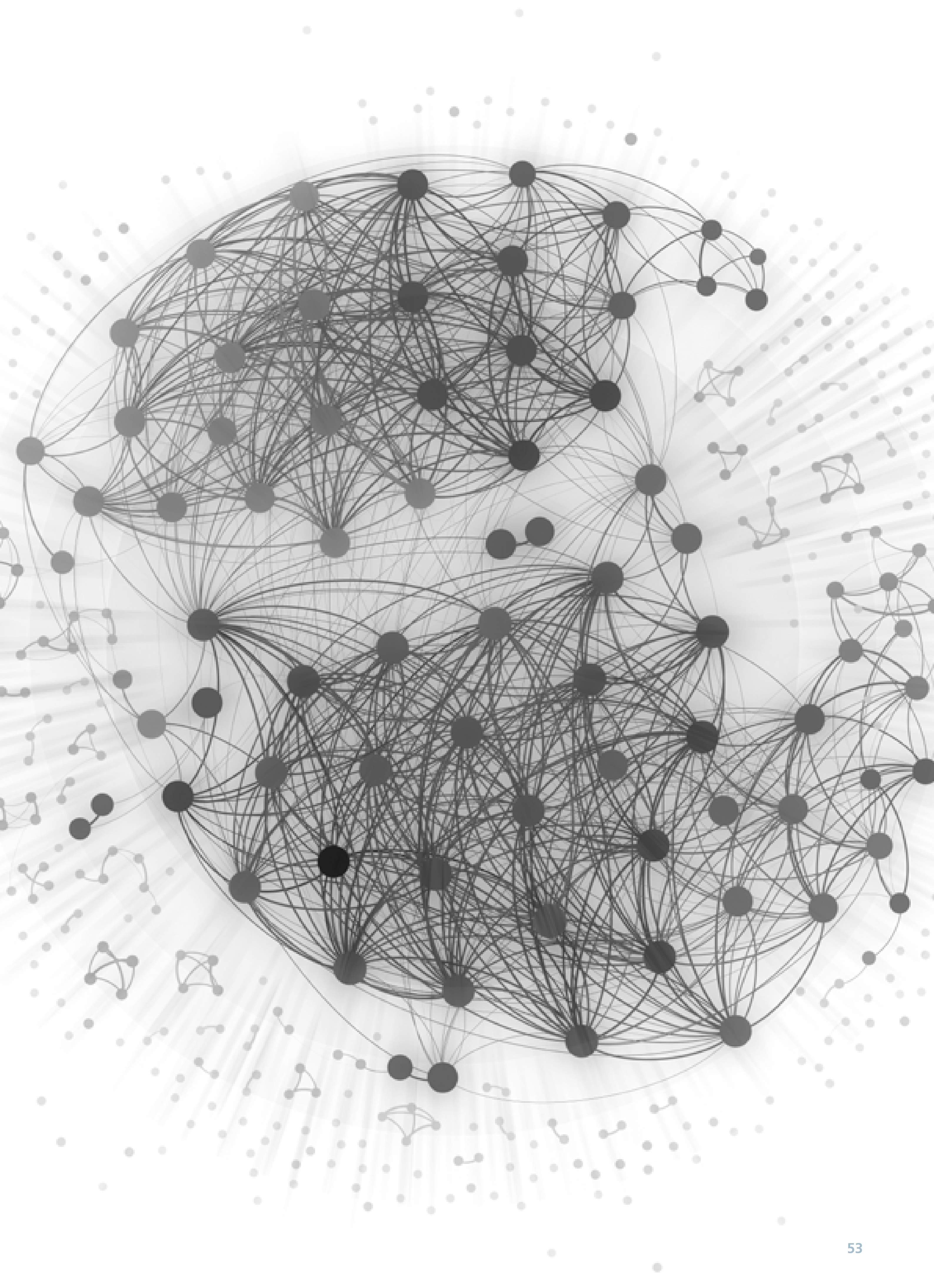
We limited our recommendations to six fields of action. We highlighted certain challenges and related research directions for each of the six recommendations. The recommendations as well as their challenges and research directions are relevant from our point of view, but not complete.

Our framework for describing dynaSoS in terms of dimensions and deriving clusters of challenges from this description is also not necessarily complete. However, it was sufficient to structure all the challenges we collected during the project. It also supported us in formulating the challenges by providing context.

Outlook

The framework could be filled with further research challenges in the future and help to identify further connecting points between research activities. It provides a terminology for discussing dynaSoS that is similar to the way RAMI 4.0 provides a terminology for discussing Industry 4.0. So far, there is no platform for dynaSoS. Germany has a platform for Industry 4.0 and a platform for learning systems. In the future, we may have a platform for dynaSoS. As long as no such platform exists yet, we invite researchers and practitioners to get in touch with us.

Funding agencies may use this work to identify research topics within their scope of funding or to get inspirations for their funding strategy. The amount of investment is an important factor for the implementation of the roadmap, but there are a lot of other factors that need to be considered in order to estimate the timeframe for realizing dynaSoS in Germany and beyond. These include, for instance, the gap between the state of the art and the state of the practice, systemic issues that hinder changes of current regimes (Geels, 2002), and collaboration with the global research community.



References

Abraham R., Schneider J. and Vom Brocke J. Data governance: A conceptual framework, structured review, and research agenda [Journal] // International Journal of Information Management. - 2019. - pp. pp.424-438.

Adedeji K. and Hamam Y. Cyber-physical systems for water supply network management: basics, challenges, and roadmap. [Journal] // Sustainability. - 2020.

Adler R. [et al.] Research Report SIL4CLOUD, Digitale Schiene Deutschland [Report]. - 2022.

Ahlawat P. and Rana C. An Era of Recommendation Technologies in IoT: Categorisation by techniques, Challenges and Future Scope [Journal] // Journal of Science & Technology. - 2021.

Ahmed A. [et al.] Service management for IoT: requirements, taxonomy, recent advances and open research challenges [Journal]. - [s.l.] : IEEE Access, 2019.

Ali O., Ishak M. and Bhatti M. Emerging IoT domains, current standings and open research challenges: A review [Journal]. - [s.l.] : PeerJ Computer Science, 2021.

Amade B. and Nwakanma C. Identifying Challenges of Internet of Things on Construction Projects Using Fuzzy Approach [Journal]. - [s.l.] : Journal of Engineering, Project & Production Management, 2021.

Antonino P. [et al.] Continuous engineering for Industry 4.0 architectures and systems [Article] // Software: Practice and Experience. - [s.l.] : Wiley Online Library, 2022.

Arnold R. D. and Wade J. A Definition of Systems Thinking: A Systems Approach [Journal] // Procedia Computer Science. - 2015. - pp. 669-678.

Atlam H. [et al.] Integration of cloud computing with internet of things: challenges and open issues [Conference] // EEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData). - 2017.

Avizienis A. [et al.] Basic concepts and taxonomy of dependable and secure computing [Journal]. - [s.l.] : IEEE Transactions on Dependable and Secure Computing, 2004.

Bader J. [et al.] AI in Software Engineering at Facebook [Journal] // IEEE Software. - 2021. - 4 : Vol. 38. - pp. 52-61.

Badr Y., Zhu X. and Alraja M. Security and privacy in the Internet of Things: threats and challenges [Journal]. - [s.l.] : Service Oriented Computing and Applications, 2021.

Balduf F. [et al.] System of Systems Engineering in Deutschland: Bestandsaufnahme und Ausblick. [Article] // Tagungsband zum Tag des Systems Engineering. - [s.l.] : Gesellschaft für Systems Engineering, 2022. - pp. 26-30.

Barabási A.-L. and Pósfai M. Network Science [Book]. - 2016.

Bauer T., Antonino P. O. and Kuhn T. Towards Architecting Digital Twin-Pervaded Systems [Conference] // 2019 IEEE/ACM 7th International Workshop on Software Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES). - 2019. - pp. 66-69.

Bleiholder J. and Naumann F. Data Fusion [Journal] // ACM Comput. Surv.. - 2009. - 1 : Vol. 41. - p. 41.

Bloomfield R. and Rushby J. Assessing Confidence with Assurance 2.0 [Report] : CSL Technical Report SRI-CSL-2022-02. - 2022.

Bohlen V. [et al.] Open Data Spaces: Towards the IDS Open Data Ecosystem [Report]. - [s.l.] : International Data Spaces Association (ed.), 2018.

Bolukbasi T. [et al.] Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings [Conference] // Advances in Neural Information Processing Systems 29 (NIPS 2016). - 2016.

Booch G. The history of software engineering [Journal] // IEEE Software. - 2018. - 5 : Vol. 35. - pp. 108-114.

Bosch J. Continuous Software Engineering [Book]. - [s.l.] : Springer, 2014.

Brandt S. and Siebert J. Challenges related to system-of-systems for greening and climate adaptation in smart cities [Report]. - 2022.

Briand L. [et al.] The case for context-driven software engineering research: Generalizability is overrated [Journal] // IEEE Software. - 2017. - 5 : Vol. 35. - pp. 72-75.

Brunello G. and Wruuck P. Skill Shortages and Skill Mismatch in Europe: A Review of the Literature [Journal] // IZA Discussion Paper No. 12346. - 2019. - p. 35.

Bundesgesetzblatt Product Liability Act of 15 December 1989 (Federal Law Gazette I, p. 2198), last amended by Article 5 of the Act of 17 July 2017. - [s.l.] : Federal Law Gazette [Bundesgesetzblatt] I p. 2421, 1989.

Burton R. M. [et al.] GitHub: exploring the space between boss-less and hierarchical forms of organizing [Journal] // Journal of Organization Design. - 2017. - 10.

Carleton A. [et al.] Architecting the future of software engineering: A national agenda for software engineering research and development [Report]. - [s.l.] : Software Engineering Institute., 2021.

Casañ M., Alier M. and Llorens A. Teaching ethics and sustainability to informatics engineering students, an almost 30 years' experience [Journal]. - [s.l.] : Sustainability, 2020.

Castellani B. and Gerrits L. Map of the complexity sciences [Report]. - [s.l.] : Art and Science Factory, LLC, 2021.

Castelluccia C. and Le Métayer D. Understanding algorithmic decision-making: Opportunities and Challenges [Report]. - [s.l.] : EPRS | European Parliamentary Research Service, 2019.

Chen Z. [et al.] Characteristics and technical challenges in energy Internet cyber-physical system [Conference] // IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). - 2016.

Cook R. I. How Complex Systems Fail [Report] / Cognitive technologies Laboratory ; University of Chicago. - 1998.

Das B. An overview on big data: characteristics, security and applications [Journal] // Journal of Network Communications and Emerging Technologies (JNCET). - 2020. - 9 : Vol. 10. - p. 6.

Daun M. [et al.] Collaborating multiple system instances of smart cyber-physical systems: a problem situation, solution idea, and remaining research challenges [Conference] // IEEE/ACM 1st international workshop on software engineering for smart cyber-physical systems. - 2015. - pp. 48-51.

Dey A. Understanding and using context [Journal]. - [s.l.] : Personal and ubiquitous computing, 2001.

Diène B. [et al.] Data Management Mechanisms for IoT: Architecture, Challenges and Solutions [Conference]. - [s.l.] : 5th International Conference on Smart and Sustainable Technologies (SpliTech), 2020.

Dörner D. Die Logik des Mißlingens. Strategisches Denken in komplexen Situationen [Book]. - 2003.

Dridi C., Benzadri Z. and Belala F. System of Systems Modelling: Recent work Review and a Path Forward [Conference] // 2020 International Conference on Advanced Aspects of Software Engineering (ICAASE). - [s.l.] : IEEE, 2020. - pp. 1-8.

Ebert C. and Hochstein L. DevOps in Practice [Journal] // IEEE Software. - 2022. - 1 : Vol. 40.

Eletronic Components and Systems Strategic Research and Innovation Agenda 2023 [Report]. - 2023.

Ernst N. and Bavota G. AI-Driven Development Is Here: Should You Worry? [Journal] // IEEE Software. - 2022. - 2 : Vol. 39. - pp. 106-110.

EU Commission EC Staff Working Document: Delivering on the UN's Sustainable Development Goals – A comprehensive approach [Report] / European Commission. - 2020.

Falcão R. [et al.] The practical role of context modeling in the elicitation of context-aware functionalities: a survey [Conference] // 2021 IEEE 29th International Requirements Engineering Conference (RE). - [s.l.] : IEEE, 2021. - pp. 35-45.

Falcão R., King R. and Carvalho A. xPACE and TASC Modeler: Tool support for data-driven context modeling [Conference] // REFSQ 2022. - Birmingham : CEUR-WS, 2022.

Falcão R., Pestana M.C. and Vieira V. TASC4RE: a data-driven context model to support the elicitation of context-aware functionalities [Conference] // ER Forum. - 2022.

Fang Z. System-of-Systems Architecture Selection: A Survey of Issues, Methods, and Opportunities [Journal] // IEEE Systems Journal. - 2021.

Farley D. Modern Software Engineering [Book]. - [s.l.] : Addison-Wesley Professional, 2022.

Fayollas C., Bonnin H. and Flebus O. SafeOps: A Concept of Continuous Safety [Conference] // 2020 16th European Dependable Computing Conference (EDCC). - 2020.

Feth P. Dynamic Behavior Risk Assessment for Autonomous Systems [Book] : PhD Thesis / PhD Theses in Experimental Software Engineering, Band 67. - [s.l.] : Fraunhofer Verlag, 2020.

Fitzgerald B. and Stol K. Continuous software engineering: A roadmap and agenda [Article] // Journal of Systems and Software. - [s.l.] : Elsevier, 2017. - Vol. 123. - pp. 176-189.

Ford L. Artificial intelligence and software engineering: a tutorial introduction to their relationship [Journal] // Artificial Intelligence Review. - 1987. - Vol. 1. - pp. 255-273.

Franke M., Hribernik K and Thoben K. Semantic Interoperability for Logistics and Beyond [Book Section] // Dynamics in Logistics. - [s.l.] : Springer, 2021.

Freitag C. [et al.] The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations [Journal] // Patterns. - 2021. - 9 : Vol. 2.

Gao J. [et al.] A Survey on Deep Learning for Multimodal Data Fusion [Journal] // Neural Computation. - 2020. - 5 : Vol. 32. - pp. 829–864.

Geels F. Technological transitions as evolutionary reconfiguration processes: a multi-level perspective and a case-study [Journal]. - [s.l.] : Research Policy, 2002.

Geihs K. and Wagner M. Context-awareness for self-adaptive applications in ubiquitous computing environments [Conference] // International Conference on Context-Aware Systems and Applications. - [s.l.] : Springer, 2012.

Geisslinger M. [et al.] Autonomous Driving Ethics: from Trolley Problem to Ethics of Risk [Journal]. - 2021.

George J and Nazeh M. Challenges Faced by CIOs in cloud and IoT based organizations-A Study on IT and Business Leaders [Journal]. - [s.l.] : International Journal on Informatics Visualization, 2019.

Glymour C., Zhang K. and Sprites P. Review of Causal Discovery Methods Based on Graphical Models [Journal] // Frontiers in Genetics. - 2019. - Vol. 10.

Goli T. and Kim Y. A Survey on Securing IoT Ecosystems and Adaptive Network Vision [Journal]. - [s.l.] : International Journal of Networked and Distributed Computing, 2021.

Gorod A., Sauser B. and Boardman J. System-of-systems engineering management: A review of modern history and a path forward [Journal] // IEEE Systems Journal. - 2008. - 4 : Vol. 2. - pp. 484–499.

Groen E. [et al.] Anwendungsfälle zu dynamischen Systemen der Systeme der Zukunft [Online]. - 2022. - https://www.iese.fraunhofer.de/content/dam/iese/dokumente/media/studien/whitepaper-dynamic_systems_of_systems-dt-fraunhofer_iese.pdf.

Gröger J. [et al.] Green Cloud Computing - Lebenszyklusbasierte Datenerhebung zu Umweltwirkungen des Cloud Computing [Report]. - [s.l.] : Umweltbundesamt, 2021.

Groß J. [et al.] Architectural Patterns for Handling Runtime Uncertainty of Data-Driven Models in Safety-Critical Perception [Conference] // SAFECOMP 2022: Computer Safety, Reliability, and Security. - 2022. - pp. 284-297.

Hauer M. P. and Zweig K. Chancen und Risiken algorithmischer Entscheidungen [Article] // Human Resources Manager.. - 2021. - pp. 48-53.

Hofmeister C. [et al.] A general model of software architecture design derived from five industrial approaches [Journal] // J. of Systems and Software.. - 2007. - pp. 106-126.

Humble J. and Farley D. Continuous delivery: reliable software releases through build, test, and deployment automation [Book]. - [s.l.] : Pearson Education, 2010.

IEEE Computer Society Model Process for Addressing Ethical Concerns During System Design, Standard IEEE 7000 [Journal]. - 2021.

Imai S. Is GitHub copilot a substitute for human pair-programming? an empirical study [Conference] // ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE ,22). - Ney York : Association for Computing Machinery, 2022. - pp. 319-321.

INCOSE System Engineering Vision 2035 [Report]. - 2023.

Jamshidi P. [et al.] Microservices: The Journey So Far and Challenges Ahead [Journal]. - [s.l.] : IEEE Software, 2018. - 3 : Vol. 35. - pp. 24-35.

Janssen N. The Data Science Talent Gap: Why It Exists And What Businesses Can Do About It [Online] // Forbes Technology Council. - October 11, 2022. - https://www.forbes.com/sites/forbestechcouncil/2022/10/11/the-data-science-talent-gap-why-it-exists-and-what-businesses-can-do-about-it/?sh=76439db23982.

Johnston D. Scientists Become Managers-The ‚T'-Shaped Man [Journal] // IEEE Engineering Management Review. - 1978. - 3 : Vol. 6. - pp. 67 - 68.

Jones M. Global Warming and System Safety [Journal] // Journal of System Safety. - 2022. - 3 : Vol. 57.

Kagermann H. [et al.] Fachforum autonome systeme im hightech-forum: autonome systeme–chancen und risiken für wirtschaft, wissenschaft und gesellschaft [Report]. - 2017.

Kästner C. and Kang E. Teaching software engineering for al-enabled systems [Conference] // 2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). - [s.l.] : IEEE, 2020.

Kim M. Research issues and challenges related to Geo-IoT platform [Journal]. - [s.l.] : Spatial Information Research, 2018.

Kläs M. and Sembach L. Uncertainty Wrappers for Data-driven Models - Increase the Transparency of AI/ML-based Models through Enrichment with Dependable Situation-aware Uncertainty Estimates [Conference] // WAISE 2019 at Computer Safety, Reliability, and Security (SAFECOMP 2019). - 2019.

Kleppman M. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems [Book]. - [s.l.] : O'Reilly Media, Inc., 2017.

Klotins E. and Gorschek T. Continuous Software Engineering in the Wild [Article] // Software Quality: The Next Big Thing in Software Engineering and Quality - 14th International Conference on Software Quality, {SWQD} 2022, Vienna, Austria, May 17-19, 2022, Proceedings. - Vienna, Austria, : [s.n.], May 17-19, 2022. - pp. 3-12.

Knauss A. [et al.] ACon: A learning-based approach to deal with uncertainty in contextual requirements at runtime [Journal]. - [s.l.] : Information and Software Technology, 2016.

Koorosh A. [et al.] SafeDrones: Real-Time Reliability Evaluation of UAVs Using Executable Digital Dependable Identities [Conference] // Model-Based Safety and Assessment. IMBSA 2022.. - 2022. - Vols. Lecture Notes in Computer Science, vol 13525..

Küçük Y., Henderson T. and Podgurski A. Improving fault localization by integrating value and predicate based causal inference techniques [Conference] // 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). - 2021.

Kuusisto M. Organizational effects of digitization: a literature review [Journal] // International Journal of Organization Theory and Behaviour. - 2017. - 3 : Vol. 20. - pp. 341-362.

LaBerge L. [et al.] How COVID-19 has pushed companies over the technology tipping point—and transformed business forever [Report]. - [s.l.] : McKinsey & Company, 2020.

Laney D. 3D data management: Controlling data volume, velocity and variety. [Report]. - Stamford : META Group Inc., 2001. - p. 4.

Li F. [et al.] Advances and emerging challenges in cognitive internet-of-things [Journal]. - [s.l.] : IEEE Transactions on Industrial Informatics, 2019.

Li L. Reskilling and Upskilling the Future-ready Workforce for Industry 4.0 and Beyond [Journal] // Information Systems Frontiers. - 2022.

Liu Z. and Wang J. Human-cyber-physical systems: concepts, challenges, and research opportunities [Journal]. - [s.l.] : Frontiers of Information Technology & Electronic Engineering, 2020.

Mackenzie D. and Pearl J. The Book of Why: The New Science of Cause and Effect [Book]. - 2018.

Maier M. W. Architecting principles for systems-of-systems [Journal] // Systems Engineering. - 1998. - 4 : Vol. 1. - pp. 267-284.

Martinez-Fernández S. [et al.] Software Engineering for AI-Based Systems: A Survey [Journal] // ACM Transactions on Software Engineering and Methodology. - 2022. - 2 : Vol. 31. - pp. 1-59.

Mascardi V. [et al.] Engineering Multi-Agent Systems: State of Affairs and the Road Ahead [Journal] // ACM SIGSOFT Software Engineering Notes. - 2019. - 1 : Vol. 44. - pp. 18-28.

Mattos D. and Liu Y. On the use of causal graphical models for designing experiments in the automotive domain [Conference] // Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022. - 2022.

Mattos D. and Liu Y. On the Use of Causal Graphical Models for Designing Experiments in the Automotive Domain [Conference] // International Conference on Evaluation and Assessment in Software Engineering 2022 (EASE ,22). - New York, NY, USA : Association for Computing Machinery, 2022. - pp. 264-265.

McDermott T. [et al.] AI4SE and SE4AI: A Research Roadmap [Journal] // Insight. - 2020. - 1 : Vol. 23. - pp. 8-14.

Méndez Fernández D. and Passoth J. Empirical software engineering: from discipline to interdiscipline [Journal] // Journal of Systems and Software. - 2019. - Vol. 148. - pp. 170-179.

Mennenga M. [et al.] Synthetic emergence as a functional unit for the environmental assessment of a system of systems [Conference] // Procedia CIRP. - 2020. - pp. 393-398.

Mennenga M. [et al.] Synthetic emergence as a functional unit for the environmental assessment of a system of systems [Journal] // Procedia CIRP. - 2020. - Vol. 90. - pp. 393-398.

Miloslavskaya N. and Tolstoy A. Internet of Things: information security challenges and solutions [Journal] // Cluster Computing. - 2019. - pp. pp.103-119.

Möller U. and McCaffrey M. Levels without Bosses? Entrepreneurship and Valve's Organizational Design [Book Section] // The Invisible Hand in Virtual Worlds: The Economic Order of Video Games. - [s.l.] : Cambridge University Press, 2021.

Nazish M. and Banday M. Green internet of things: a study of technologies, challenges and applications [Conference] // International Conference on Automation and Computational Engineering (ICACE). - 2018.

Nielsen B. A comprehensive review of data governance literature [Conference] // Selected Papers of the IRIS, n. 8. - [s.l.] : Association for Information Systems, 2017.

Niestroy I. and Meuleman L. Managing the implementation of the SDGs [Report]. - [s.l.] : European Public Administration Country Knowledge, 2020.

Northrop L. [et al.] Ultra-large-scale systems: The software challenge of the future [Report]. - [s.l.] : Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst., 2006.

O'Neil C. Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy [Book]. - [s.l.] : Crown, 2016.

Paris T. [et al.] Teaching co-simulation basics through practice [Conference] // 2019 Summer Simulation Conference. - 2019.

Parrend P. and Collet P. A Review on Complex System Engineering [Journal] // J Syst Sci Complex. - 2022. - Vol. 33. - pp. 755–1784.

Partrdige D. and Wilks Y. Does AI have a methodology which is different from software engineering? [Journal] // Artificial Intelligence Review. - 1987. - pp. 111-120.

Pasquale F. The Black Box Society: The Secret Algorithms That Control Money and Information [Book]. - [s.l.] : Harvard University Press, 2015.

Pearl J. The seven tools of causal inference with reflections on machine learning [Journal] // Communications of the ACM. - 2019. - 3 : Vol. 62. - pp. 54-60.

Peter C. and Swilling M. Linking complexity and sustainability theories: Implications for modeling sustainability transitions [Journal]. - [s.l.] : Sustainability, 2014.

Pinheiro M. and Souveyet C. Supporting context on software applications: a survey on context engineering [Conference] // Modélisation et utilisation du contexte. - 2018.

Puryear B. and Sprint G. Github copilot in the classroom: learning to code with AI assistance. [Journal] // J. Comput. Sci. Coll.. - 2022. - 1 : Vol. 38. - pp. 37–47.

Rinehart D. J., Knight J. C. and Rowanhill J. Understanding What It Means for Assurance Cases [Report] : NASA/CR–2017-219582. - [s.l.] : NASA, 2017.

Rockström J. [et al.] Planetary boundaries: exploring the safe operating space for humanity [Journal]. - [s.l.] : Ecology and society, 2009.

Rodrigues A. [et al.] Enhancing context specifications for dependable adaptive systems: A data mining approach [Journal] // Information and software technology. - 2019. - pp. 115-131.

Rodríguez-Pérez G., Nadri R. and Nagappan M. Perceived diversity in software engineering: a systematic literature review [Journal] // Empirical Software Engineering. - 2021. - 102 : Vol. 26.

Runge J. [et al.] Detecting and quantifying causal associations in large nonlinear time series datasets [Journal] // Science advances. - [s.l.] : Science advances, 2019. - 11 : Vol. 5.

Rushby J. Runtime certification [Conference] // 8th International Workshop, RV 2008. - [s.l.] : Springer, 2008.

SafeTRANS Safety, Security, and Certifiability of Future Man-Machine Systems [Report]. - 2019.

Scharinger B. [et al.] Can RE Help Better Prepare Industrial AI for Commercial Scale? [Journal] // IEEE Software. - [s.l.] : IEEE Software, 2022. - Vol. 39. - pp. 8-12.

Schlossnagle T. Monitoring in a DevOps World: Perfect should never be the enemy of better. [Journal] // Queue. - [s.l.] : Queue, 2017. - 6 : Vol. 15. - pp. 35-45.

Schmerl B. [et al.] Challenges in composing and decomposing assurances for self-adaptive systems [Conference] // Software Engineering for Self-Adaptive Systems. - 2017.

Schneider C. and Betz S. Transformation²: Making software engineering accountable for sustainability [Journal] // Journal of Responsible Technology. - 2022.

Schneider D. and Trapp M. B-space: dynamic management and assurance of open systems of systems [Journal]. - [s.l.] : Journal of Internet Services and Applications, 2018.

Schneider D. Conditional Safety Certification for Open Adaptive Systems. - [s.l.] : Fraunhofer Verlag, 2014.

Schölkopf B. Causality for Machine Learning [Report]. - 2019.

Schranz M. [et al.] Swarm intelligence and cyber-physical systems: concepts, challenges and future trends [Journal]. - [s.l.] : Swarm and Evolutionary Computation, 2021.

Scoones I. [et al.] Dynamic Systems and the Challenge of Sustainability, STEPS Working Paper 1 [Report]. - [s.l.] : Brighton: STEPS Centre, 2007.

Seyff N. [et al.] The Elephant in the Room – Educating Practitioners on Software Development for Sustainability [Conference] // 2021 IEEE/ACM International Workshop on Body of Knowledge for Software Sustainability (BoKSS). - 2020. - pp. 25-26.

Siebert J. Applications of statistical causal inference in software engineering [Report]. - 2022.

Siebert J., Ciarletta L. and Chevrier V. Agents and artefacts for multiple models co-evolution. Building complex system simulation as a set of interacting models [Conference] // Autonomous Agents and Multiagent Systems-AAMAS 2010. - 2010. - pp. 509-516.

Siedel G. [et al.] Utilizing Class Separation Distance for the Evaluation of Corruption Robustness of Machine Learning Classifiers [Conference] // IJCAI-ECAI-22 Workshop on Artificial Intelligence Safety (AISafety 2022) . - 2022.

Siegenfeld A. F. and Bar-Yam Y. An Introduction to Complex Systems Science and Its Applications [Journal] // Complexity. - 2020. - Vol. 2020. - p. 16.

Singh S. [et al.] Challenges of digital twin in high value manufacturing [Journal]. - 2018.

Skulmowski A. and Rey G. COVID-19 as an accelerator for digitalization at a German university: Establishing hybrid campuses in times of crisis [Journal] // Human Behavior and Emerging Technologies. - 2020. - Vol. 2. - pp. 212– 216.

Smite D. [et al.] Spotify Guilds: How to Succeed With Knowledge Sharing in Large-Scale Agile Organizations [Journal] // IEEE Software. - 2019. - 2 : Vol. 36.

Snafucatchers consortium STELLA. Report from the SNAFUcatchers Workshop on Coping With Complexity [Report]. - Brooklyn NY : [s.n.], 2017.

Strietska-Ilina O. Skill shortages. Modernising vocational education and training-Fourth report on vocational education and training research in Europe: background report [Report]. - 2008. - p. 72.

Sundberg N. Sustainable IT Playbook for Technology Leaders: Design and implement sustainable IT practices and unlock sustainable business opportunities [Book]. - [s.l.] : Packt, 2022.

Taivalsaari A. and Mikkonen T. A roadmap to the programmable world: software challenges in the IoT era [Journal] // IEEE software. - 2017.

Tavčar J. and Horvath I. A review of the principles of designing smart cyber-physical systems for run-time adaptation: Learned lessons and open issues [Journal] // IEEE Transactions on Systems, Man, and Cybernetics. - [s.l.] : IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018. - pp. 145-158.

Theobald S. and Diebold P. Interface problems of agile in a non-agile environment [Conference] // 19th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2018). - 2018. - pp. 123-130.

Tian L. [et al.] Federated learning: Challenges, methods, and future directions [Journal] // IEEE Signal Processing Magazine. - [s.l.] : IEEE signal processing magazine, 2020. - 3 : Vol. 37. - pp. 50-60.

Tisi M. [et al.] Towards Twin-Driven Engineering: Overview of the State-of-The-Art and Research Directions [Conference] // Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems (APMS 2021). - [s.l.] : IFIP Advances in Information and Communication Technology, 2021. - Vol. 630.

Tlili S., Mnasri S. and Val T. A survey on iot routing: Types, challenges and contribution of recent used intelligent methods [Conference] // 2nd International Conference on Computing and Information Technology (ICCIT). - [s.l.] : IEEE, 2022.

Tsigkanos C., Nastic S and Dustdar S. Towards resilient Internet of Things: Vision, challenges, and research roadmap [Conference] // IEEE 39th International Conference on Distributed Computing Systems (ICDCS). - 2019. - pp. 1754-1764.

Uday P. and Marais K. Designing resilient systems-of-systems: a survey of metrics, methods, and challenges [Journal] // Systems Engineering. - 2015.

Vosoughi S., Roy D. and Aral S. The spread of true and false news online [Journal] // Science. - [s.l.] : science, 2018. - 6380 : Vol. 359. - pp. 1146-1151.

Yang Y. [et al.] A Survey on Deep Learning for Software Engineering [Journal] // ACM Comput. Surv.. - [s.l.] : ACM Computing Surveys (CSUR), 2022. - 10s : Vol. 54. - p. 73.

Younan M. [et al.] Challenges and recommended technologies for the industrial internet of things: A comprehensive review [Journal] // Measurement. - 2020.

Zahidi S. [et al.] The Future of Jobs Report 2020 [Report]. - [s.l.] : World Economic Forum Platform for Shaping the Future of the New Economy and Society, 2020. - p. 163.

Zhang J. M. [et al.] Machine learning testing: Survey, landscapes and horizons [Journal] // IEEE Transactions on Software Engineering. - [s.l.] : IEEE Transactions on Software Engineering, 2022. - 1 : Vol. 40.

Zweig K. Ein Algorithmus hat kein Taktgefühl: Wo künstliche Intelligenz sich irrt, warum uns das betrifft und was wir dagegen tun können [Book]. - [s.l.] : Heyne Verlag, 2019.

Appendix A – German research landscape

Here we indicate German research organizations that, from our point of view, can help implement the research roadmap for dynaSoS. While this list is definitely not complete, it should provide a starting point for building networks and/or consortia of organizations with the purpose of fostering collaboration on dynaSoS.

For the sake of transparency, we have **put in bold face** the names of the organizations whose individuals contributed to the DynaSoS project. We take this opportunity to thank them once again for investing their invaluable time in sharing with us their thoughts on challenges, recommendations, and research directions for dynaSoS.

Computer Science Research at Max Planck Institutes

The Max Planck Society is composed of several research institutions in Germany and abroad that carry out basic research in different knowledge fields. Among these, some institutes focus on Computer Science research, including the Max Planck Institute for Informatics, the Max Planck Institute for Intelligent Systems, and the Max Planck Institute for Software and Systems.

DFKI

The German Center for Artificial Intelligence (DFKI) conducts research related to AI including cyber-physical systems and multi-agent systems.

Fortiss

Fields of research include Architecture and Services for Critical Infrastructures, Human-centered Engineering, Industrial IoT, ML, Model-based Engineering, and Software Dependability, among others.

Fraunhofer ICT Group

The Fraunhofer ICT Group as part of the Fraunhofer-Gesellschaft is the largest IT research organization in Europe. The Fraunhofer ICT Group currently has 21 member institutes throughout Germany including, for instance, the Fraunhofer Institute for Software and Systems Engineering ISST, the Fraunhofer Institute for Open Communication Systems FOKUS, the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, the Fraunhofer Institute for Cognitive Systems IKS, and the Fraunhofer Institute for Experimental Software Engineering IESE.

FZI

Research areas Software Engineering, Embedded Systems and Sensors Engineering, Intelligent Systems and Production Engineering, and Information Process Engineering.

German Institute of Urban Affairs

The institute carries out research on solutions for municipal challenges. It advises municipalities on different topics, such as intelligent transport systems, sharing economy, smart city, and resilient cities, among others.

German Systems Engineering Society

The German Systems Engineering Society represents INCOSE in German-speaking countries. Working groups include AI-based System, Model-based Systems Engineering, Moderate-Complex Systems, and Sustainability enabled by Systems Engineering.

Humboldt University Berlin

Software and Systems Engineering for Complex Safety-Critical Systems and Software Evolution

Institute for AI Safety and Security

Research fields include AI Engineering, Safety-critical Data Infrastructure, Execution Environments & Innovative Computing Models, and Business Development and Strategy.

Institute for Product Engineering at KIT

Research groups include Advanced Systems Engineering, and Human-Machine Systems.

Institute of Information Security and Dependability

Research groups for Decentralized Systems and Network Services, Modelling for Continuous Software Engineering, Logic of Autonomous Dynamical Systems, Dependability of Software-intensive Systems and Test, Validation and Analysis of Software-Intensive Systems.

Karlsruhe Institute of Technology

Software Design and Quality, including the research groups “Modeling for Continuous Software Engineering” and “Dependability of Software-intensive Systems”

Ludwig-Maximilians-Universität München (LMU)

Chair of Human-Centered Ubiquitous Media AG

National Academy of Science and Engineering

Topics include Energy & Resources, Healthcare technology, Circular Economy, Innovation, Digital & Self-learning, International Cooperation, and Mobility, among others.

RheinMain University of Applied Sciences (HSRM)

Working group “Learning and Visual Systems”, member of the research focal area “Smart Systems for Man and Technology”. Research topics include computer vision, ML, and data science, among others.

Rhineland-Palatinate Technical University (formerly Technical University of Kaiserslautern)

Chair of Software Engineering – Dynamic risk assessment and safety assurance under uncertainty; Chair of Artificial Intelligence; Algorithmic Accountability Lab

Saarland University

Explainability and Perspicuous Computing – Research on many topics including CPS, dynamic dependable systems, dynamic and hybrid systems, among others.

Safetrans

SafeTRANS is a not-for-profit association joining partners from industry and science across application domains. Its roadmap and position paper »Safety, Security, and Certifiability of Future Man-Machine Systems« is closely related to dynaSoS and the recommendation addressing safety and trustworthiness.

Technical University of Berlin

Information System Engineering – Research on many topics including automated driving, cloud-native architecture and engineering, AI-based data sovereignty, data-driven processes, and data management, among others.

Technical University of Munich (TUM)

Applied Software Engineering Group – Research fields include continuous software engineering, CPS, smart environments, and ML applications, among others.

Universität Hamburg

Among the foci of the Department of Informatics are Human-Centered Computing (see Ethics in Information Technology), Complex System Engineering (see Databases and Information Systems, Applied Software Technology, and Information System, Socio-Technical System Design).

University of Braunschweig

Chair of Sustainable Manufacturing & Life Cycle Engineering including an SoS Engineering Research Group

University of Cologne

Software and Systems Engineering addressing Requirements Engineering and Data-driven Systems Engineering

University of Oldenburg

Department of Foundations and Applications of Systems of Cyber-Physical Systems for modeling, verification, and synthesis of reactive, real-time, and hybrid dynamics in embedded and cyber-physical systems.

University of Rostock

Chair for Modeling and Simulation

University of Stuttgart

Institute of Software Engineering (Empirical Software Engineering Group); Institute for Visualization and Interactive Systems, Human-Computer Interaction and Cognitive Systems Department

University of Ulm

Institute of Software Engineering and Programming Languages

Impressum

31st of January, 2023

IESE-001.23/E | public

The research described in this paper was performed in the DynaSoS project (grant no. 01IS21104) of the German Federal Ministry of Education and Research (BMBF). We thank all external people who participated in the interviews and the workshops. We also thank Sonnhild Namingha for proofreading and the whole project team for their contributions.

Editorial office: Dr. Rasmus Adler, Dr. Frank Elberzhager, Prof. Dr.-Ing. Peter Liggesmeyer

Authors: Rasmus Adler, Frank Elberzhager, Rodrigo Falcão, Julien Siebert, Eduard C. Groen, Jana Heinrich, Florian Balduf

Graphics and layout: Stefanie Ludborzs, Patrick Mennig

Proofreading: Sonnhild Namingha

Image credits: Titelseite: © iStock.com/Anastasiia Shavshyna, Bearbeitung Fraunhofer IESE | Seite 7: © iStock.com/dirkpurz | Seite 9: © iStock.com/LegART2 | Seite 14: © iStock.com/Oleh_Slobodeniuk | Seite 19: © iStock.com/Mumemories | Seite 21: © iStock.com/nazar_ab | Seite 25: © iStock.com/redtea | Seite 34: © iStock.com/SHansche | Seite 50: © iStock.com/Sholikhul Bakhmid | Seite 53: © istock.com/ Alexander Vasilyev | Seite 68: arquiplay77 – stock.adobe.com

© Fraunhofer-Institut für Experimentelles Software-Engineering IESE, Kaiserslautern 2023

Fraunhofer IESE is an institute of the Fraunhofer-Gesellschaft.
The institute transfers innovative software development techniques,
methods, and tools into industrial practice.
It helps companies to build up software competencies that meet
their needs and to achieve a competitive market position.
Fraunhofer IESE is headed by Prof. Dr.-Ing. Peter Liggesmeyer

Contact

Prof. Dr.-Ing. Peter Liggesmeyer
Executive and Scientific Director
peter.liggesmeyer@iese.fraunhofer.de
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
www.iese.fraunhofer.de

Dr. Rasmus Adler
Research Program Autonomous Systems
rasmus.adler@iese.fraunhofer.de
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
www.iese.fraunhofer.de

Dr. Frank Elberzhager
Architecture-Centric Engineering
frank.elberzhager@iese.fraunhofer.de
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern
www.iese.fraunhofer.de