Constellation-level Variability Modeling for Safety Constraint Refinement in System of Systems

Julieth Patricia Castellanos-Ardila, Sasikumar Punnekkat and Nazakat Ali
School of Innovation, Design and Engineering Mälardalen University - Västerås, Sweden
{julieth.castellanos, sasikumar.punnekkat, nazakat.ali}@mdu.se

***Abstract*—Safety assurance in *System of Systems* (SoS) orchestrations is challenged by the inherent variability of *Constituent Systems* (CS) and the dynamic environments in which they must operate. To address this challenge, we previously introduced SOSoS (*Safe Orchestration of Systems of Systems*), a process that integrates *System-Theoretic Process Analysis* (STPA) with principles from *Software Product Line Engineering* (SPLE) to support safety reasoning about the SoS variability at the macro-level, i.e., analysis of the SoS as a whole in its context. However, SOSoS lacked a clearly defined mechanism for managing variability at lower levels. Thus, in this paper, we expand SOSoS to the meso-level, i.e., a level focusing on the constellation formation within a SoS. A *constellation* is a subset of CS that together form operational links and interact to deliver a specific SoS capability. This constellation-level variability analysis employs the concepts of *Orthogonal Variability Modeling* (OVM) to identify key *Variation Points* (VP), *Variants* (V), and *Inter-Variant Constraints* (IVC) that govern the formation of viable constellations and can be used to generate *Refined Safety Constraints* (RSC). We demonstrate the method's applicability through a case study on a construction site. The meso-level analysis shifts the focus to constellation tactical configuration options, enabling a more precise characterization of orchestration logic and its associated safety constraints.**

***Index Terms*—Safety Constraints, Variability, System of Systems, SoS Constellation, Orchestration, Construction Domain.**

I. Introduction

A *System of Systems* (SoS) [1] refers to a complex integration of independently developed and managed *constituent systems* (CS), each offering a specific *capability* (i.e., ability of an entity to achieve its objectives) that contributes to a broader, shared mission. To coordinate such assemblies, the orchestration strategy has been proposed [2]. *Orchestration* is a form of composition in which a central element, i.e., the orchestrator, manages the behavior of the other elements [3]. In a SoS, the orchestrator is a mediator, i.e., an architectural component that enables CS communication, coordination, cooperation, and collaboration [4]. However, achieving safety assurance in orchestrated SoS remains a significant challenge due to the inherent variability of CS and the wide range of dynamic, heterogeneous operational contexts in which they operate. To structure this complexity, SoS behavior can be analyzed at different levels [5], i.e., macro (the SoS as a whole in its context), meso (the formation and evolution of interacting subsets of CS delivering joint capabilities), and micro (the internal behavior of a stable constellation).

This work is supported by the Vinnova-funded project SIMCON and SAILS, a pre-study aimed at investigating safety assurance of AI systems.

In our prior work [6], we proposed SOSoS (*Safe Orchestration of Systems of Systems*), a process integrating the System-Theoretic Process Analysis (STPA) [7] and the Software Product Line Engineering (SPLE) [8] approaches to support structured safety reasoning in the presence of SoS variability. The process is organized into three main activities. First, the design phase aims to establish the orchestrator's control logic. Second, the safety analysis focuses on identifying safety constraints for both core and variable features of the SoS. Finally, the third activity addresses the configuration and verification of SoS variants. Notably, SOSoS provides support for safety reasoning about SoS variability at the macro-level. However, it lacks a clearly defined mechanism for addressing variability at lower levels. As a result, SOSoS does not yet offer systematic support for refining safety constraints based on concrete configurations, which limits its applicability.

In this paper, we expand SOSoS to the meso-level by introducing a method for *Constellation-level Variability Modeling* to support *Safety Constraint Refinement in SoS*. For this, we use the SoS constellation concept [9] as a reference model to encapsulate the configuration-specific interactions within CS that work together, rather than traditional CS-level variability. To facilitate the analysis, this constellation-level variability employs the concepts of *Orthogonal Variability Modeling* (OVM) [8]. In general, for each *Variation Point* (VP) (i.e., feature in the constellation where a choice can be made), we define concrete *variants* (V) (i.e., specific options for the constellation VP) and specify *inter-variant constraints* (IVC) (i.e., dependencies that permit variants to be selected and form viable constellations). The resulting IVC act as constraint adaptation rules that can be used to tailor the generic SoS-level to *Refined Safety Constraints* (RSC). Finally, we demonstrate the method's applicability through a case study featuring machine coordination for mass removal in construction. In this context, the meso-level analysis provided a clear structure that permits shifting the focus to constellation tactical configuration options, enabling a more precise characterization of orchestration logic and its associated safety constraints.

This paper is organized as follows. Section II presents essential background. Section III presents related work. Section IV introduces the Constellation-level Variability Modeling approach. Section V illustrates the modeling approach with a case study. Section VI presents a discussion of our findings. Finally, Section VII presents conclusions and future work.

II. Background

A. Safe Orchestration of Systems of Systems (SOSoS)

SOSoS [6] is a structured process designed to support safety analysis in Systems of Systems (SoS) orchestrations. As illustrated in Fig.1, SOSoS integrates tasks from System-Theoretic Process Analysis (STPA) [7] (shown in blue) and Software Product Line Engineering (SPLE) [8] (shown in pink), while also incorporating verification activities (shown in white) to assess the correctness of system behavior. SOSoS is an iterative process, i.e., it allows refining the information from initial activities based on the feedback obtained in subsequent activities (see the blue arrows depicted in the figure).

This iterative process is composed of three main activities as follows. First, the *design*, where the orchestrator's control logic required for managing interactions among constituent systems (CS) is defined. Second, the *safety analysis*, in which the orchestrator's safety concept is developed based on potential hazards and unsafe control actions that may occur in the SoS. Third, the *configuration verification*, which aims at validating whether the orchestrator's control logic responds to the identified safety concept for specific configurations.

Inside the three main activities mentioned, there are specific granular tasks. In this paper, we focus on the task called **variability analysis** (highlighted in red in Fig. 1), which is included in the safety analysis activity. This task aims to systematically examine the variable features across CS, covering possible SoS configurations. It takes, as input, the high-level safety constraints defined for the orchestrator core (macro-level). The outcome is a set of variability-targeting constraints that enable the refinement of previously identified safety constraints, ensuring they remain valid and effective across different SoS configurations.

B. SoS Constellations

A key concept in SoS is the constellation. A *constellation* [5] refers to a temporary or ongoing subset of CS that actively collaborate to deliver a joint capability (i.e., the ability to achieve an objective) that none of them could achieve independently on their own. It represents an operational instantiation of the SoS, dynamically assembled based on current goals, situational context, and mutually beneficial outcomes for the participating CS. In general, constellations are defined as the "working horses" [9] of a SoS, directly realizing its mission through purposeful cooperation among sets of CS.

The formation and dissolution of constellations is inherently dynamic [9]. In this process, each CS makes operational (micro-level) decisions based on its goals, resources, and expectations about the behavior of others. These decisions are also informed by strategic (macro-level) reasoning, i.e., long-term, mission-wide, and often organizational considerations, as well as tactical (meso-level), i.e., planning and deployment decisions. A CS may participate in multiple constellations simultaneously in the absence of constraints like resource contention, timing conflicts, or contractual obligations.

The dynamic constellation formation and dissolution contribute to variability in SoS, often driven by differences in CS [10]. This form of variability may manifest dynamically at runtime. However, the variability addressed in this work is situated at design time, where tactical differences across constellation configurations can lead to different emergent behaviors, even if the mission objective remains the same.

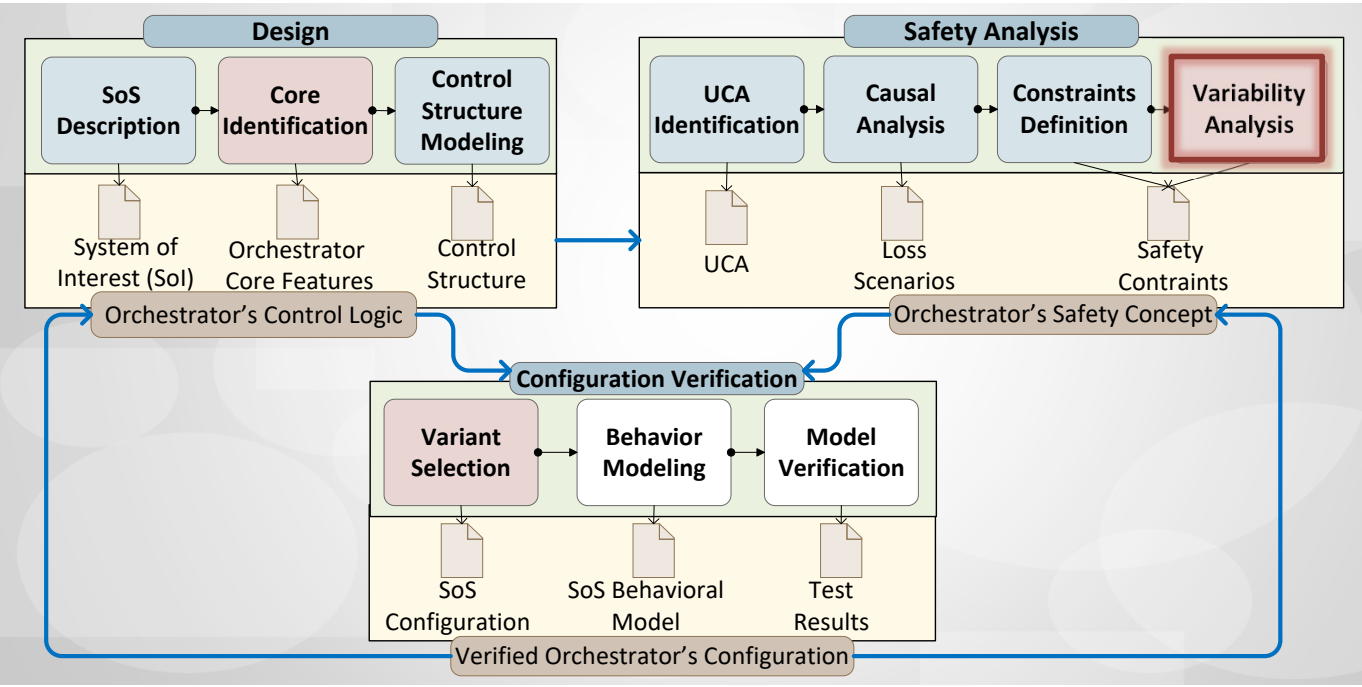

Fig. 1. SOSoS Process [6].

C. Orthogonal Variability Model (OVM)

Orthogonal Variability Modeling (OVM) [8] is a technique developed to explicitly capture and manage variability across a family of related systems. Generally, variability in software product line engineering refers to the capability of a software platform or set of artefacts to support differences between applications derived from the same product line. OVM introduces a dedicated variability model that exists orthogonally to other system features, rather than embedding variability directly into functional artifacts. In OVM, the central elements are:

1) **Variation Point:** Location where a choice can be made.
2) **Variant:** Specific option for a variation point.
3) **Dependency:** Relationships between variants.

In particular, the dependencies can be *mandatory*, and *optional*. A *mandatory relationship* means that a variant must be included whenever its associated variation point is selected, while an *optional relationship* allows the variant to be included or not, depending on specific configuration needs. *Variant constraint dependencies* are used to define logical relationships between variants in an orthogonal variability model. The *requires relationship* indicates that the selection of one element (variant or variation point) necessitates the inclusion of another, ensuring dependent features are always configured together. Conversely, the *excludes relationship* enforces mutual exclusivity, meaning that if one element is selected, the other must not be. These constraints help ensure valid and coherent product configurations by capturing logical dependencies and incompatibilities in the variability model.

III. Related Work

The product line approach, originated in manufacturing and industrial engineering, refers to the systematic management of differences among products while sharing a common and reusable core structure. Software engineering adopted these principles in the 1980s as organizations recognized the potential for platform-based development, systematic reuse, and variability management in software products [11] as well as in their development processes [12]. Subsequent efforts in areas such as process compliance [13], compliance-checking artifacts [14], and knowledge-driven variability modeling [15] have further leveraged this variability-centric view. In the job done in [16] there is also an emphasis in reuse through modularization, traceability, and configuration-dependent assessment for managing risk across product variants.

Variability management has also been explored in the SoS context. Botterweck [10] highlights that SoS variability requires not only adaptations of SPLE methods but also new mechanisms for managing aspects such as dependency propagation, architectural layering, and multi-level evolution. Klein and McGregor [17] emphasize that platform scoping must balance current reuse with future mission flexibility. Fendali et al. [18] show that variability in SoS is tightly linked to long-term architectural change, where coordination-level modifications can impact system-wide coherence. Tekinerdogan et al. [19], Lonetti et al. [20], and Wagner et al. [21] further demonstrate the role of feature models in process configuration, testing, and digital twin instantiation. The INCOSE's Systems Engineering magazine, in its 2021 issue [22], has also presented several articles discussing feature-based PLE within the SoS contexts. In particular, it discusses industry-aligned insights into applying PLE techniques for modularity in DoD projects, hybrid product breakdown structures, variability modeling, and organizational challenges. Collectively, these works underscore the need for dedicated variability approaches tailored to the complexity and dynamics of SoS.

Our work in this paper builds on these foundations. In particular, we introduce a constellation-level variability modeling approach (see Section IV) that explicitly targets the SoS interaction. Rather than prior efforts, which focus on system components or process abstractions, we center our analysis on how constellation variability features, especially tactical dimensions, such as coordination patterns, autonomy distributions, and communication modes, influence the refinement of safety requirements. With this approach, we extend the scope of existing SoS variability approaches by integrating safety refinement into the variability modeling process, resulting from the interplay of CS at the SoS constellation level. This idea contributes to the growing body of research that seeks to make SoS variability not only a design concern but a driver for configuration-aware, safety-informed SoS engineering.

IV. Constellation-level Variability Modeling

The SOSoS process (recalled in Section II-A) integrates safety analysis to support the orchestration of variable SoS configurations. It was created with a macro-level mindset, i.e., the focus is on the SoS as a whole in its context, providing answers to questions related to system-wide properties at an aggregated level. A central task in this process is the **variability analysis** (highlighted in red in Fig. 1), which focuses on refining safety constraints based on identified SoS variability. This task is essential for ensuring that safety requirements remain valid across different SoS configurations. However, there is currently no sufficient methodological support for analysing the type of variability required at this level.

In an orchestrated SoS, hazards often arise not from individual CS, which we assume are safe in isolation, but from their interactions during mission execution. To capture these interaction-driven analyses, we adopt the SoS constellation concept (as described in II-B) as a reference model for representing tactical configuration dimensions. This reflects a meso-level perspective, focused on the coordination and interplay among groups of CS, bridging the gap between macro-level SoS orchestration and micro-level CS behavior. We utilize OVM concepts (as recalled in Section II-C) as the foundation for modeling tactical constellation-level variability. OVM provides a structured notation for capturing variation points, their associated variants, and inter-variant constraints, enabling the representation of alternative tactics for constellation configuration. This modeling approach produces tactical-specific insights that inform the refinement of safety con-

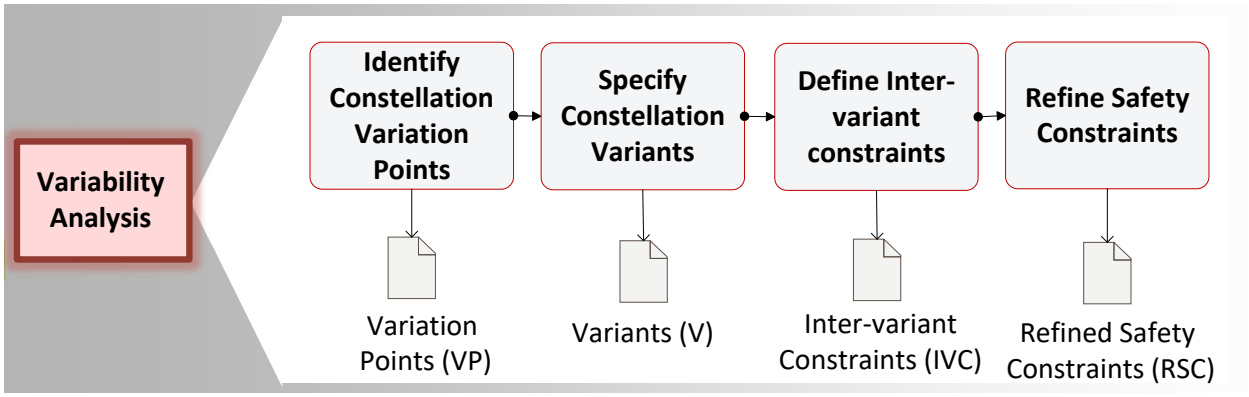

Fig. 2. Constellation-level Variability Modeling.

straints, ensuring that the requirements align with the structural assumptions of each constellation (see Fig. 2).

1) **Identify Constellation Variation Points:** We identify key aspects of SoS constellations that influence safety during orchestration. In particular, Variation Points (VP) represent the tactical dimensions that differ in a constellation. VP are elements whose variability can significantly impact coordination, control, and safety mechanisms. For this, we need to analyze the constellation's design and operational context to determine which aspects are flexible, configurable, or subject to change.
2) **Specify Constellation Variants:** After identifying the VP at the constellation level, the focus shifts to defining the specific variants (V), i.e., the concrete alternatives available for each variation point. These V represent the alternative features in a VP that a SoS constellation can exhibit. The specification of these V facilitates a systematic analysis of design alternatives and supports the differentiation of possible constellation configurations.
3) **Define Inter-Variant Constraints:** Once constellation-level VP and their associated V are defined, it is crucial to reason about their possible dependencies, as not all V combinations are actually valid. To manage this aspect, we define a set of inter-variant constraints (IVC) that restrict undesirable combinations and ensure design compatibility. IVC reflect domain knowledge, operational assumptions, and safety concerns, and are expressed as logical rules, e.g., requires, excludes, or conditionals, capturing compatibility relationships.
4) **Refine Safety Constraints:** While initial safety constraints provide a high-level understanding of required safety measures, they must be adapted to reflect the specific constellation configuration in use. Thus, this step focuses on refining the safety constraints previously identified in the SOSoS methodology by incorporating the IVC defined at the constellation level. The IVC act as refinement rules that permit the tailoring of generic safety requirements to refined safety constraints (RSC).

These tasks are essentially sequential. However, its final output, i.e., the RSC, is subject to verification in the next SOSoS activity, specifically the configuration verification. Based on the feedback, new analysis may be needed.

V. Case Study

In this section, we describe the case study and then we use it to illustrate the method described in Section IV.

A. Mass removal in construction sites

This case study focuses on the orchestration of a constellation of CS for mass removal operations at a construction site. Mass removal refers to the large-scale excavation, loading, and transportation of earth materials, e.g., soil, rock, or debris, to reshape the terrain in preparation for infrastructure development. The primary CS involved in these operations include excavators, wheel loaders, and haulers (see Fig. 3), each contributing distinct capabilities such as excavation, material loading, and transport. These CS are typically developed independently, with varying degrees of autonomy and heterogeneity in control logic and sensing capabilities. For example, a fully autonomous hauler may operate alongside a manually operated excavator. The operational environment is dynamic and spatially constrained, making the coordination of shared zones critical to ensure safe and efficient operations.

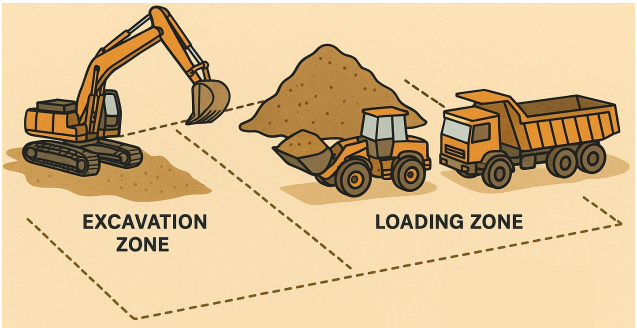

Fig. 3. Available set of CS for Mass Removal in a Construction Site (The figure was generated with the assistance of ChatGPT [23]).

In this context, the orchestrator must manage the constellation holistically, accounting for both the individual CS behavior and the emergent interactions among them. Particular

attention is given to physical interactions, such as passing, loading, or waiting, in shared operational zones, which introduce a significant risk of unsafe proximity. If not properly managed, unsafe proximity can result in severe consequences, including injury to personnel or damage to equipment. This hazard is inherently systemic, requiring the definition of SoS-level safety requirements during SoS design (see below).

SR-1: The orchestrator shall include a monitoring mechanism to ensure the safety distance between CS.

SR-2: The orchestrator shall have the ability to stop a CS in case of safety distance violation.

B. Constellation-level Variability Modeling

In this section, we apply the process depicted in Figure 2.

1) Identify Constellation Variation Points: In this step, we identify dimensions that influence safety through the design of a constellation for the mass removal operation (see Section V-A). We focus on VP that, when instantiated differently, cause constellation variation, even when addressing the same operational goals. Table I summarizes a representative set of mandatory tactical dimensions, which, though not exhaustive, are sufficient to illustrate the proposed modeling approach.

TABLE I
CONSTELLATION VARIATION POINTS (VP)

ID	Variation Point	Safety Implications
VP1	Coordination Pattern	Defines temporal and logical coordination. It can lead to coordination-related hazards, e.g., race condition between haulers that can cause unsafe proximity.
VP2	Autonomy Levels	Defines the distribution of autonomy levels between the CS, for example, autonomous haulers vs manual wheel loaders and excavators. It influences aspects like operator involvement.
VP3	Zone Complexity	Refers to the operational intricacy of the worksite layout. For example, as presented in Fig. 3, there may be zones, like the excavation one that only permits one machine. Zone complexity increases the likelihood of unsafe proximity.
VP4	Safety Control Allocation	Determines responsible ownership regarding safety controls. It may be the source of potential latency in reacting to safety threats.
VP5	Inter-CS Communication	Specifies how the machines involved in the mass removal operation communicate safety-critical information. Critical for maintaining awareness and timely conflict resolution.

2) Specify Constellation Variants: Following the identification of VP relevant to the mass removal constellation (see Table I), the next step involves specifying the concrete variants (V) that represent the possible configuration options associated with each VP. In this case study, the selected V are summarized in Table II. These V are informed by patterns commonly observed in real-world deployments of construction machinery and represent operational optional choices that directly influence constellation configurations.

TABLE II
CONSTELLATION VARIANTS (V)

VP	Variant	Description
VP1	V1.1 Sequential	Tasks are performed one after another with clear handovers between CS.
	V1.2 Parallel	Multiple CS perform tasks simultaneously in the same or nearby zones.
VP2	V2.1 Autonomous	All CS operate autonomously with no human involvement in the loop.
	V2.2 Manual	The constellation is composed entirely of manually-operated machines.
	V2.3 Mixed	Some CS are autonomous while others are manually operated.
VP3	V3.1 Single	Every CS operates within a single, well-defined work area.
	V3.2 Multiple	The work area has multiple overlapping or concurrent operational zones.
VP4	V4.1 Centralized	All safety decisions or actions are made by a central orchestrator.
	V4.2 Hybrid	Safety responsibilities are split between the central orchestrator and CS.
	V4.3 CS-local	Each CS manages its own safety using local perception and logic.
VP5	V5.1 Mediated	All communication between CS is routed through the orchestrator.
	V5.2 Direct	CS exchange information directly, without orchestrator intervention.

3) Define Inter-Variant Constraints: Each mandatory VP identified in Table I requires a selection, and the choice itself introduces flexibility in how the SoS constellation can be configured to address different operational needs. However, not all combinations of V, as listed in Table II, are appropriate or safe for deployment. Some V combinations may be inherently incompatible, potentially introducing critical risks such as delayed safety responses, communication breakdowns, conflicting control decisions, or coordination mismatches among the CS in the constellation. These issues are especially concerning in dynamic environments, where timely and dependable interaction between CS are essential for safe SoS operation.

For the purpose of illustrating this step, in this paper we present a set of IVC related to our case study (see Table III). Such IVC provide a foundation for guiding the selection of compatible configurations in an SoS constellation. They capture critical dependencies between key VP, such as the relationship between autonomy levels and control allocation, or between inter-CS communication modes and coordination patterns. These IVC effectively rule out combinations that could lead to latency or coordination mismatches. However, while the current set of IVCs is necessary, it may not be fully sufficient to guarantee the validity of all configurations. In particular, some constraints are unidirectional and may benefit from symmetric or transitive extensions to more precisely define the design space. To ensure completeness, it is advisable to evaluate all possible variant combinations using constraint-solving tools. Overall, the IVCs provide essential guidance but should be complemented with formal analysis to ensure configuration correctness at scale.

TABLE III
INTER-VARIANT CONSTRAINTS (IVC)

ID	Constraint	Rationale
IVC1	If `V1.2`, ⇒ includes `V3.2`.	Parallel coordination (V1.2) requires complex zoning (V3.2) to manage concurrent operations.
IVC2	If `V2.1`, ⇒ excludes `V4.1`.	Fully autonomous constellations (V2.1) exclude centralized safety control(V4.1), as autonomy implies independent decision-making.
IVC3	If `V2.2 or V2.3` ⇒ includes `V4.1`.	Manual or mixed-autonomy setups (V2.2 or V2.3) require centralized safety (V4.1) for coherence and operator-in-the-loop awareness.
IVC4	If `V5.1`, ⇒ excludes `V1.2`.	Mediated communication (V5.1) introduces latency, which is unsafe for parallel task execution (V1.2).
IVC5	If `V5.2` ⇒ excludes `V4.1`.	Direct communication (V5.2) implies more distributed control, which is incompatible with centralized safety (V4.1).
IVC6	If `V2.1` ⇒ includes `V5.2`.	Autonomous CS (V2.1) favors direct communication (V5.2) for distributed coordination to support real-time, decentralized coordination.
IVC7	If `V3.2` ⇒ excludes `V4.3`.	In complex, shared zones (V3.2), fully local safety (V4.3) is not sufficient. It requires hybrid or centralized mechanisms.
IVC8	If `V2.3` ⇒ excludes `V4.3`.	Mixed autonomy (V2.3) setups need centralized/hybrid safety. Local safety logic (V4.3) is not sufficient to human-machine interactions.
IVC9	If `V4.1` ⇒ includes `V5.1`.	Centralized safety control (V4.1) includes mediated communication (V5.1) to maintain orchestrator awareness.

TABLE IV
REFINEMENTS FOR SR1 (MONITORING FOCUS) BASED ON IVC

IVC	Impact	Monitoring-oriented Refinement
IVC1	Parallel execution in complex zones requires continuous, spatially aware monitoring across zones.	If V1.2 is selected, the orchestrator shall monitor safety distances across multiple concurrent zones.
IVC2	In fully autonomous setups, monitoring is decentralized, i.e., each CS must monitor distances locally.	If V2.1 is selected, each CS shall include a local mechanism to monitor safety distances.
IVC3	Centralized safety implies centralized monitoring, possibly with human-in-the-loop.	If V2.2 or V2.3 is selected, the orchestrator shall centrally monitor and aggregate safety distance data across all CS.
IVC4	Mediated communication introduces latency, making it unsafe for real-time peer-to-peer coordination.	If V5.1 is selected, the orchestrator shall perform safety distance monitoring in a sequential task execution mode.
IVC5	Direct CS-to-CS communication is incompatible with centralized safety control in the orchestrator.	If V5.2 is selected, monitoring of safety distances shall be supported via direct inter-CS communication.
IVC6	Safety distance enforcement relies on real-time, local communication among autonomous CS.	If V2.1 is selected, each CS shall monitor distances and exchange proximity data directly with others.
IVC7	Some level of central oversight is needed in zones where CS paths overlap dynamically.	If V3.2 is selected, the orchestrator shall assist local monitoring with an integrated zone-level view.
IVC8	Mixed autonomy needs hybrid monitoring, handling both automated and human-operated CS	If V2.3 is selected, the orchestrator shall integrate data from both autonomous and human-operated CS for proximity monitoring.
IVC9	Centralized control requires mediated data flows, possibly increasing monitoring latency.	If V4.1 is selected, the orchestrator shall ensure timely monitoring despite communication latency.

4) Refine Safety Constraints: In this section, we present the refinements for the general safety constraints established earlier in the SOSoS methodology (see SR1 and SR2 in Section V) in the light of the possible configurations given by the IVC identified in Table III. For this, we consider every IVC, examine its impact in the safety-requirements and provide a possible path for refinement (see Table IV). In general, four refinements can be derived from the IVC to ensure the SR1 focus is maintained (see Fig. 4).

The four refinements are explained below.

- **RSC1-1 (Centralized Monitoring):** If the orchestration includes manual or mixed-autonomy CS (V2.2 or V2.3), centralized safety control (V4.1) is required (per IVC3 and IVC8). This implies that the orchestrator must perform centralized monitoring of all CS to maintain global awareness and ensure consistent enforcement of safety distances. Mediated communication (V5.1) may be used, but care must be taken to manage latency (IVC9). This is further reinforced by IVC8, which excludes local safety control (V4.3) when mixed autonomy (V2.3) is present.
- **RSC1-2 (Decentralized Monitoring):** In fully autonomous configurations (V2.1), centralized safety control (V4.1) is excluded (via IVC2), and direct communi-

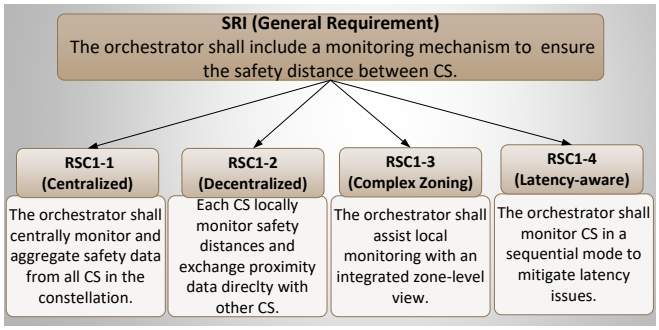

Fig. 4. Refined Safety Constraint SR1

cation between CS is included (V5.2 via IVC6). Similarly, direct communication (V5.2) implies more distributed control (via IVC5). Thus, each autonomous CS is responsible for monitoring its safety distances and directly sharing this data with peers to ensure safe operation.

- **RSC1-3 (Complex Zoning Supervision):** When the worksite involves parallel task execution (V1.2) or multiple zones (V3.2), as in IVC1 and IVC7, the monitoring burden exceeds what local systems can manage alone. Multiple zones involve overlapping paths, unpredictable interactions, and shared space usage. Thus, the orchestrator must support local monitoring efforts with zone-level supervision to maintain global situational awareness.
- **RSC1-4 (Latency-Constrained Monitoring):** If communication is mediated through the orchestrator (V5.1), direct and timely inter-CS communication is not possible (as per IVC4). This latency makes real-time monitoring for parallel tasks infeasible. As a result, the monitoring mechanism must assume a sequential coordination model where CS operate in isolation within shared zones.

This variability-aware reasoning also applies to SR2, which requires the orchestrator to have the ability to stop a CS in case of safety distance violation (see Fig. 5). Similar to SR1, the allocation of authority for triggering an emergency stop must adapt to the constellation configuration. In centralized stop control, typical of manual or mixed-autonomy setups, the orchestrator is responsible for issuing stop commands to all CS, ensuring an authoritative intervention. In contrast, local stop control applies to fully autonomous configurations, where each CS is capable of independently detecting hazards and executing self-halt actions without the involvement of the orchestrator. Between these extremes lies the hybrid stop approach, required in mixed environments where centralized and local mechanisms must work in tandem. For example, in mixed-autonomy systems, autonomous CS may self-stop while human-operated ones rely on the orchestrator to intervene.

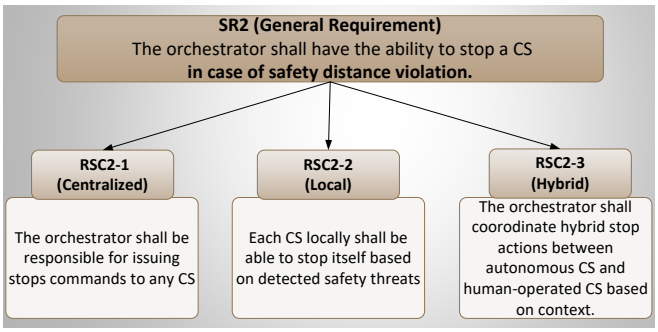

Fig. 5. Refined Safety Constraint SR2

Spatial and architectural conditions, e.g., complex zoning or mediated communication, can also influence how safety enforcement mechanisms must operate at runtime. Therefore, refinements for these aspects may also be needed. For example, in overlapping work zones, the risk of cascading interactions may necessitate synchronized stop coordination, even in systems with hybrid or centralized control. In a similar manner, communication latency introduced by mediated architectures may require preemptive or buffered stopping strategies.

VI. Discussion

From the application of our approach, despite its simplicity, we have identified some points that are worth discussing.

First, we have presented a set of aspects that are intended to represent design-time variability points (see Section V-B1) that are well-suited for configuration modeling and pre-runtime validation. Each VP corresponds to a structural decision made before deployment, affecting, to some extent, how the system will be constrained at runtime. However, more VP can be considered depending on the needs in the SoS constellation. One example of those additional design-time VP is the operator presence, which can be used to distinguish between fully automated, remotely supervised, and on-board human involvement. However, this VP could also be folded into VP2 if generalized. Another example of a possible missing VP is the fallback strategy, which represents the model's behavior in degraded modes. However, this VP may be reflected in the refined safety constraints instead. At first glance, there is redundancy between VP4 and VP5, as both influence safety management in the constellation. However, they address distinct concerns. In particular, VP4 defines who is responsible for making safety decisions, while VP5 specifies how safety-critical information is exchanged. In general, identifying an appropriate VP is a non-trivial task that requires a deep understanding of the system architecture and operational context. As such, the involvement of domain experts is essential to ensure that the selected VPs are both relevant and meaningful.

Second, our case study has not yet addressed run-time variability, i.e., the aspects of a SoS constellation that may change dynamically during operation, such as environmental and terrain conditions, or the presence of moving obstacles. Unlike design-time variability, these changes occur in real time and require the system to adapt on the fly. Critically, run-time variability is often enabled or constrained by design-time decisions, such as the degree of flexibility in zone definitions or the activation of specific CS capabilities. To manage this effectively, it is essential to incorporate operational design domain considerations, i.e., the environmental and operational conditions under which the CS is designed to operate safely, into the design-time analysis. This will allow potential runtime conditions to be anticipated. Furthermore, each CS brings its own capabilities and limitations, i.e., sensing accuracy, control latency, and actuation dynamics, which shape its response to evolving conditions. These factors are best analyzed through micro-level analysis, which focuses on the internal behavior of each CS in context. Therefore, delegating the assessment of such CS-specific variability to the micro level is both practical and appropriate. However, the insights gained from this analysis must be integrated into the constellation-level reasoning to ensure that assumptions about local adaptability inform coordination strategies and refined safety constraints.

Third, the definition of variants (see Section V-B2) reveals key architectural differences that shape the structure of the SoS constellation. These variants reflect choices for the VP that influence how CS interact. For instance, decentralized coordination in shared zones, combined with direct inter-CS communication, increases coordination demands and the potential for timing mismatches. The presence of mixed autonomy may add complexity, as it requires coordination and shared control between humans and machines. In this context, IVC (see Section V-B3) play a critical role by explicitly defining valid variant combinations. However, the IVC defined in this case study may not capture all relevant dependencies. Thus, to ensure completeness, these constraints should be complemented with formal analysis to evaluate the whole space of variant combinations.

This meso-level analysis represents a significant turning point in the process of designing SoS. Even in its current, partial form, this approach has enabled the derivation of safety requirements that are more aligned with the tactical variability inherent in SoS constellations. Through the structured examination of configuration diversity and interaction patterns within the constellation, the analysis has enabled a clearer understanding of how system design choices impact safety requirements (see Section V-B4). Thus, in general, this analysis has helped bridge the gap between high-level safety goals and their practical implementation. However, while this methodology improves the alignment between system design and safety assurance, it is not exhaustive. To fully capture the nuances of SoS behavior, additional forms of analysis, such as micro-level analysis of CS capabilities, real-time performance assessment, and ODD modeling, are necessary.

VII. Conclusions and Future Work

In this paper, we extended the SOSoS to the meso-level by introducing a *constellation-level variability modeling* to support *safety constraint refinement in SoS*. For this, we used the SoS constellation concept as a reference model to encapsulate tactical considerations. With this approach, we identified relevant *VPs*, defined concrete *V*, and determined *IVC* applicable to SoS configurations. These IVC served as adaptation rules for refining generic SoS-level safety constraints into configuration-aware *RSC*. The approach was illustrated through a case study featuring mass removal in the construction domain, highlighting its ability to provide a structured view of constellation-level variability and its influence on the orchestration logic and safety constraints.

Future work will focus on several directions. First, we plan to explore tool support for our approach by using feature models in FeatureIDE[1]. Second, we aim to integrate this approach with runtime monitoring frameworks to assess how configuration-specific constraints evolve under operational dynamics. Third, further empirical validation is needed across additional domains where more tactical dimensions are considered to assess generalizability. Finally, we envision linking constellation-level variability with micro-level analysis to enable traceability between local CS characteristics and SoS-wide safety constraints.

[1]https://www.featureide.de/

References

[1] ISO/IEC JTC 1/SC 7, *ISO/IEC/IEEE 21841:2019. Systems and Software Engineering — Taxonomy of System of Systems*, Std., 2019.

[2] T. Nordstrom, L. R. Sutfeld, and T. Besker, "Exploring Different Actor Roles in Orchestrations of System of Systems," in *19th Annual SoSE Conference*, 2024, pp. 190–196.

[3] ISO/IEC JTC 1/SC 38, *ISO/IEC 18384-1:2016. Information technology — Reference Architecture for Service Oriented Architecture (SOA RA)*, Std., 2016.

[4] L. Garcés, F. Oquendo, and E. Y. Nakagawa, "Towards a Taxonomy of Software Mediators for Systems-of-Systems," in *Brazilian Symposium on Software Components, Architecture, and Reuse*, 2018, pp. 53–62.

[5] J. Axelsson, "A Refined Terminology on System-of-Systems Substructure and Constituent System States," in *14th Annual Conference System of Systems Engineering*, 2019, pp. 31–36.

[6] J. P. Castellanos-Ardila, N. Ali, S. Punnekkat, and J. Axelsson, "Making Systems of Systems Orchestrations Safer," in *European Safety and Reliability and Society for Risk Analysis Europe*, 2025.

[7] N. G. Leveson and J. P. Thomas, *STPA Handbook*, 2018.

[8] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2011.

[9] P. Svenson and J. Axelsson, "Should I Stay or Should I Go? How Constituent Systems Decide to Join or Leave Constellations in Collaborative SoS," in *16th International Conference of System of Systems Engineering*, 2021, pp. 179–184.

[10] G. Botterweck, "Variability and Evolution in Systems of Systems," *arXiv preprint arXiv:1311.3627*, 2013.

[11] F. J. Van der Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer Science & Business Media, 2007.

[12] H. Washizaki, "Building Software Process Line Architectures from Bottom Up," in *International Conference on Product Focused Software Process Improvement*. Springer, 2006, pp. 415–421.

[13] T. Varkoi, T. Mäkinen, B. Gallina, F. Cameron, and R. Nevalainen, "Towards Systematic Compliance Evaluation Using Safety-oriented Process Lines and Evidence Mapping," in *24th European Conference on Systems, Software and Services Process Improvement*. Springer, 2017, pp. 83–95.

[14] J. P. Castellanos Ardila and B. Gallina, "Reusing (Safety-oriented) Compliance Artifacts while Recertifying," in *9th International Conference on Model-Driven Engineering and Software Development*. SciTePress, 2021, pp. 53–64.

[15] B. Gallina, T. Y. Olesen, E. Parajdi, and M. Aarup, "A Knowledge Management Strategy for Seamless Compliance with the Machinery Regulation," in *European Conference on Software Process Improvement*. Springer, 2023, pp. 220–234.

[16] R. Schmitt, B. Falk, M. Rüßmann, C. Brecher, W. Herfs, and A. Malik, "Risk Management Across Variants Requirements and Outlook for an Efficient Risk Assessment of Machines," in *International Symposium on Systems Engineering (ISSE)*, 2015, pp. 206–211.

[17] J. Klein and J. D. McGregor, "System-of-Systems Platform Scoping," in *International Workshop on Product Line Approaches in Software Engineering*, 2013, pp. 1–4.

[18] M. H. Fendali, D. Meslati, and I. Borne, "Understanding Evolution in Systems of Systems," in *International Systems Engineering Symposium*, 2017, pp. 1–6.

[19] B. Tekinerdogan, S. Duman, H. Caner, and B. Durak, "Customizing a Feature Ontology for Product Line Engineering within a System-of-Systems Context," in *International Symposium on Systems Engineering*, 2019, pp. 1–6.

[20] F. Lonetti, V. de Oliveira Neves, and A. Bertolino, "Designing and Testing Systems of Systems: From Variability Models to Test Cases Passing through Desirability Assessment," *Journal of Software: Evolution and Process*, vol. 34, no. 10, p. e2427, 2022.

[21] H. Wagner and C. Zuccaro, "Managing Variability in Digital Twins and System Development through Product Line Engineering," in *International Symposium on Systems Engineering*, 2024, pp. 1–7.

[22] INCOSE, "Special Feature Issue: Feature-based Product Line Engineering (PLE)," *INCOSE Insight*, vol. 24, no. 1, April 2021.

[23] OpenAI, "Chatgpt (aug 26 version)," https://chat.openai.com/, 2025.