Doctoral School: MADIS-631

Towards Modeling and Supervision of Multilevel Stochastic Systems of Systems: A Hypergraph Approach.

PhD THESIS

to obtain the degree of

Doctor of University of Lille

Speciality : Computer science and automation

Defended by

Abbass CHREIM

prepared at

CRIStAL CNRS-UMR 9189 defended on December 5, 2024

at CRIStAL, Lille

Jury

Reviewers :	Pr. François CHARPILLET	-	INRIA Est., France
	Pr. Hichem ARIOUI	-	Université d'Évry, France
President of the jury :	Pr. Olivier Simonin	-	INSA-Lyon, France
Examiners :	Pr. MO JAMSHIDI	-	University of Texas, USA
	Pr. Taha CHETTIBI	-	Université de Blida, Algérie
	Dr. Anne-Lise GEHIN	-	Université de Lille, France
	Dr. Yiwen CHEN	-	INSA-Hauts de France, France
Supervisor :	Pr. Rochdi Merzouki	-	Université de Lille, France

Ecole Doctorale: MADIS-631

Vers la modélisation et la supervision des systèmes de systèmes stochastiques multiniveaux : Une approche par hypergraphe.

THÈSE

soumise à l'Université de Lille en vue de l'obtention de

Doctorat

Spécialité : Informatique et automatique

par

Abbass CHREIM

préparée au laboratoire

CRIStAL CNRS-UMR 9189

soutenue le 5 Decembre 2024

à CRIStAL

Jury

Rapporteurs :	Pr. François CHARPILLET	-	INRIA Est., France
	Pr. Hichem ARIOUI	-	Université d'Évry, France
Président du jury :	Pr. Olivier Simonin	-	INSA-Lyon, France
Examinateurs :	Pr. MO JAMSHIDI	-	University of Texas, USA
	Pr. Taha CHETTIBI	-	Université de Blida, Algérie
	Dr. Anne-Lise GEHIN	-	Université de Lille, France
	Dr. Yiwen CHEN	-	INSA-Hauts de France, France
Directeur de thèse :	Pr. Rochdi Merzouki	-	Université de Lille, France

Towards Modeling and Supervision of Multilevel Stochastic Systems of Systems: A Hypergraph Approach.

Abstract

System-of-Systems (SoS) face significant challenges, including heterogeneity, scalability, and complex interactions among component systems (CSs). These systems typically operate in dynamic environments, introducing uncertainty and stochastic behavior. Many existing studies tend to oversimplify these complexities, with some focusing only on the dynamics of CSs without adequately addressing their structure, mission, and goals. Additionally, limited research has focused on supervising SoS under such conditions. Graph models, such as hypergraphs (HG), have proven effective in modeling the structure of SoS, while stochastic and weighted hypergraphs have been successfully employed to manage stochasticity in other complex systems. In this thesis, the Multi-Level Stochastic Hypergraph (MLSHG) model is introduced to address the challenges of modeling stochastic SoS. The model adheres to the key properties of SoS as defined by Maier, distinguishing it from traditional complex systems. A novel algorithm for supervising large-scale SoS is also proposed, integrating bottom-up monitoring with top-down reconfiguration to achieve long-term goals. The proposed framework supports resilience in these complex systems through recovery mechanisms based on redundancy. In a case study on a mushroom harvesting SoS, the model demonstrated clear advantages in addressing SoS modeling challenges compared to existing methodologies. The results showed that incorporating stochastic disturbances with an adaptive threshold enabled early reconfiguration during supervision, reducing deviations from the final goal. The capability-based reconfiguration method exhibited low computational time, scaling linearly with the number of CSs, thereby improving the system's scalability. The resilience scenario results further demonstrated that incorporating both stand-in and stand-by redundancy mechanisms enhances the resilience of these complex systems.

Keywords: System of Systems (SoS), Modeling and Simulation (M&S), Hypergraph (HG), Stochastic systems, Supervision, Resilience

Vers la modélisation et supervision de systèmes de systèmes stochastiques : Un modèle d'hypergraphe stochastique à plusieurs niveaux.

Résumé

Les Systèmes de Systèmes (SdS) font face à des défis majeurs, notamment en termes d'hétérogénéité, de scalabilité et d'interactions complexes entre les composants systèmes (CS). Ces systèmes fonctionnent généralement dans des environnements dynamiques, introduisant de l'incertitude et des comportements stochastiques. De nombreuses études existantes ont tendance à simplifier à l'excès ces complexités, certaines se concentrant uniquement sur la dynamique des CS sans aborder adéquatement leur structure, mission et objectifs. De plus, peu de recherches se sont concentrées sur la supervision des SdS dans de telles conditions. Les modèles graphiques, tels que les hypergraphes (HG), se sont révélés efficaces pour modéliser la structure des SdS, tandis que les hypergraphes stochastiques et pondérés ont été employés avec succès pour gérer la stochasticité dans d'autres systèmes complexes. Dans cette thèse, le modèle Hypergraphe Stochastique Multi-Niveaux (MLSHG) est introduit pour relever les défis de la modélisation des SdS stochastiques. Le modèle respecte les propriétés clés des SdS telles que définies par Maier, ce qui le distingue des systèmes complexes traditionnels. Un nouvel algorithme pour superviser les SdS à grande échelle est également proposé, intégrant une surveillance ascendante avec une reconfiguration descendante afin d'atteindre des objectifs à long terme. Le cadre proposé soutient la résilience dans ces systèmes complexes grâce à des mécanismes de récupération basés sur la redondance. Dans une étude de cas sur un SdS de récolte de champignons, le modèle a démontré des avantages clairs pour relever les défis de modélisation des SdS par rapport aux méthodologies existantes. Les résultats ont montré que l'incorporation de perturbations stochastiques avec un seuil adaptatif a permis une reconfiguration précoce pendant la supervision, réduisant ainsi les écarts par rapport à l'objectif final. La méthode de reconfiguration basée sur les capacités a montré un faible temps de calcul, évoluant de manière linéaire avec le nombre de CS, améliorant ainsi la scalabilité du système. Les résultats du scénario de résilience ont également démontré que l'incorporation des mécanismes de redondance stand-in et stand-by renforce la résilience de ces systèmes complexes.

Mots clés: Système de Systèmes (SdS), Modélisation et simulation, Hypergraphe (HG), Systèmes stochastiques, Supervision, Résilience

This thesis is dedicated to my parents, Fatimah HAMMOUD and Adnan CHREIM, for their love, endless support and everything they have given me.

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Prof. Rochdi Merzouki, for his unwavering support and guidance over the past four years, beginning with my Master's studies. His vast knowledge, insightful suggestions, and expertise have been invaluable to my learning experience. I am endlessly inspired by his professionalism and dedication. His mentorship has been indispensable in bringing this research work to fruition.

I am sincerely grateful to Dr. Yiwen Chen, Dr. Othman Lakhal, and Mr. Abdelkader Belarouci for their expertise and vast experience in this research field, which greatly aided me throughout my journey.

My heartfelt thanks also go to my thesis reviewers: Prof. François Charpillet and Prof. Hichem Arioui, and the other jury members: Prof. Taha Chettibi, Prof. Olivier Simonin, Dr. Anne-Lise Gehin, Dr. Yiwen Chen, and Prof. Mo Jamshidi, for their time and insightful comments, which have helped improve this work.

Many thanks to my colleagues in the SoftE group and other CRIStAL groups: Abdeslam Smahi, Jun Jiang, Loic Bal, Yehya Sharif, Rim Abdallah, Brayen, Alexandre D'hooge, and Francescan Maccarini. They have always been friendly and willing to help.

Special thanks to my friends in Lille: Rkein, Shmaysani, Hnayno, Achour, Khalil, Tekko, GH, Ismail, Taha, Khalaf, and Amhaz. Your support, joy, and laughter have been pillars of strength for me throughout this journey.

To my parents, your unconditional love and trust have been a constant source of strength. Thank you for always being there.

To my brothers, Ali and Mohamad, and my sisters, Zaynab and Walaa, your unwavering support and encouragement have motivated me to keep going, even during the most challenging moments. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 General introduction

As technology and industry continue to advance, systems are becoming increasingly complex and larger in scale. This complexity stems from the intricate nature of their components, which often operate and are managed independently, while cooperating, exchanging information, and evolving dynamically. These challenges are further compounded by the need for real-time collaboration and interaction among systems. In response to this growing complexity, the concept of a **System of Systems (SoS)** [Maier 1998] has gained widespread adoption. An SoS refers to an integrated collection of **Component Systems (CSs)** that function independently yet collaborate to achieve a common goal, resulting in emergent behavior that exceeds the capabilities of each individual system.

The integration of these complex systems, however, brings forth several significant challenges, such as **heterogeneity**, **complex interactions**, and **scalability**. Addressing these challenges becomes especially difficult

when attempting to model and analyze such systems. Various studies and models have been proposed in the literature to tackle these issues ([Kumar 2017, Khalil 2012, Soyez 2015]), but many of them tend to consider a specific behaviour of the complexity of SoS by focusing only on a subset of challenges. Furthermore, these systems often operate in dynamic environments characterized by uncertainty, which introduces **stochastic** elements that impact the performance and behavior of certain **CSs**. However, existing models lack the capability to effectively represent both **deterministic** and **stochastic CSs** in an integrated manner.

While **hypergraphs** have proven effective in modeling the structure of SoS [Khalil 2012], capturing the stochastic aspects of such systems remains a challenge. Recent advancements in **stochastic hypergraphs** have demonstrated their potential in modeling stochasticity in other complex systems, but they have yet to be fully explored in the context of SoS. To address this gap, we propose in this thesis the **Multi-level Stochastic Hypergraph (MLSHG) model**. This model is designed to capture the structure, heterogeneity, and scalability of SoS while integrating stochastic elements, providing a comprehensive framework for modeling and supervising complex, dynamic systems.

While it is important to model the structure and behavior of these systems, it is equally crucial to ensure that global missions and goals are consistently met, especially during critical events like disturbances. This creates a need for decision-making support and **supervision tools** that enable monitoring and reconfiguration, which are still limited in the current literature for SoS. To address this gap, this thesis focuses on developing a supervision strategy to manage system capacity during disturbances of a stochastic SoS, ensuring that long-term objectives are achieved. By combining monitoring and reconfiguration, the proposed framework aims to improve system **resilience** and **adaptability** in changing environments.

On the other hand, resilience, which refers to the ability of an SoS to recover from failure, has been addressed through various algorithms. These algorithms often rely on **redundancy mechanisms** that help the system recover by providing backup or alternative functionalities. Since this thesis focuses on performance-based resilience, it is important to quantify this resilience. Therefore, a resilience metric is adapted to measure and evaluate the resilience of an SoS, helping us assess the system's ability to maintain performance and recover from disruptions effectively.

To apply the proposed modeling, supervision, and resilience strategies, a case study is conducted on a **mushroom harvesting System-of-Systems**. In this case study, the framework is applied to the system, considering it a **Stochastic System of Systems (SSoS)** where the performance of both

mushrooms and human operators is influenced by uncertainty and can be modeled using probabilities. The supervision scenario focuses on managing the capacity of the mushroom farm—including both human operators and robots—to meet its objectives, while the resilience scenario aims to provide recovery mechanisms and quantify the farm's resilience in the face of disruptions.

1.2 Industrial Context

This thesis work is developed in the framework of the cooperation with University of Lille and La ferme **Gontière**, which is a mushroom production farm founded in 1961 in Wervicq, and has been relocated in Comines, in the north department of France in 1986. It produces mushrooms of all sizes, both white and pink, which are packaged in trays or sold in bulk. The company focuses on ultra-fresh mushrooms, harvested the same day to be available in stores the following day. Today, La Ferme de la Gontière is a mid-sized company employing 250 people, and its mushroom market is both national and international.

The mushroom production process is a prime example of a large-scale SoS. It involves multiple stages, including compost preparation, harvesting, discharge, cooling, storage, and transport. However, a key challenge lies in the harvesting phase, which is labor-intensive, repetitive, and generally unattractive to human workers. This makes it difficult to recruit staff for harvesting tasks, and worker fatigue often leads to early departures. Given the rapid growth of mushrooms, which double in size within a day, it is crucial to harvest them promptly once they reach maturity.

To address this issue, the farm plans to explore the feasibility of integrating collaborative robots to assist human workers in the mushroom harvesting process. The key challenge is managing the combined resources of human labor (operators) and robots to meet the farm's production targets. This large-scale system, comprising 30 chambers, 150 workers, and additional robots, must be designed to effectively handle this task.

Uncertainties arise from both the unpredictable growth of mushrooms and variations in labor performance, which can result in failures. Thus, it is crucial to adopt an appropriate model that accurately represents these subsystems and their interactions while ensuring scalability.

1.3 Contractual context of the thesis

This research work was conducted within the framework of the **CueiBot** project, funded by the regional **Hauts-de-France (HDF)** program on Industry, with the BPI-France[1]. The project aims to develop a robotic SoS (System of Systems) solution for mushroom harvesting at the Gontière farm. The objective is to support the picking teams in the event of a shortage of human labor, while maintaining the same quality standards as human operators. The robotic system will be designed to detect and locate mature mushrooms, perform the picking process, and assist in transferring the harvested mushrooms to a conveyor system.

An in-depth study will be carried out to determine the appropriate size and deployment strategy for the robotic systems to ensure optimal productivity, while integrating these systems with the picker teams to control and supervise the mushroom picking process.

The **CueiBot** project consists of the following parts:

- Development and design of a SoS robotic concept for mushroom handling.
- Study of mushroom gripping mechanisms.
- Vision system study for mushroom detection and localization.
- Management and planning of labor and the proposed robotic systems.

However, the thesis work focuses solely on the management and planning of the robotic SoS and human labor using Modeling and Simulation (M&S) techniques.

1.4 Scientific context of the thesis

This research work has been conducted at the Research Center in Computer Science, Signal and Automation of Lille (CRIStAL UMR CNRS 9189) within the SoftE (System of Systems Engineering) team[2] of the ToPSyS (Tolerance Prognosis System of Systems) group[3]. The research team main topic is the automation of large scale systems, known with its class of system of System Engineering (SE). The aim of this thesis is to address the scientific challenges related to the modeling and supervision of System of Systems (SoS) that are

[1] reference DOS0191049/00
[2] https://www.cristal.univ-lille.fr/equipes/softe/
[3] https://www.cristal.univ-lille.fr/gt/topsys/

large-scale, heterogeneous, adaptive, and contain stochastic components. The scientific context concerns the following:

- Provide a new framework that can describe the **structure** of a SoS, as well as the **performance**, **missions**, **capabilities**, and **interactions** of individual subsystems, while also handling **stochastic components**.
- Propose supervision strategies for managing such complex systems by **detecting disturbances** and performing **reconfiguration**, and study and **quantify their resilience**.

The research has been carried out under the supervision of Prof. Rochdi Merzouki, Professor at the University of Lille.

1.5 Thesis objectives

System of Systems (SoS) is a topic that has recently gained significant attention, particularly in applications that are large-scale and complex. However, existing studies on modeling such systems have notable **limitations**. Some focus exclusively on modeling the behavior of physical components without clearly defining the structure, objectives, or providing decision-making tools. Others fail to adequately address **heterogeneity**, and only a few explore the **stochasticity** present in certain components. Moreover, model-based approaches for SoS supervision in these contexts remain scarce in the literature. To overcome these challenges, we aim to achieve several key objectives:

- Model the structure, missions, and functions of different components within heterogeneous SoS, including stochasticity in some component systems.
- Manage the capacity of SoS by detecting failures (monitoring) and performing reconfiguration to achieve SoS objectives through supervision methodologies.
- Study the resilience of these SoS and quantify their resilience by proposing suitable metrics.
- Validate our proposed methodologies by applying them to the mushroom harvesting SoS, enabling the M&S of this type of large-scale system, managing its capacity in case of disturbances, and studying and quantifying its resilience.

1.6 Research problem statement and formulation

A System of Systems (SoS) is a collection of interacting and independent Component Systems (CSs) that collaborate to achieve a common goal, which surpasses the capabilities of the individual systems. Despite the increasing prevalence of SoS in various large-scale, complex applications, existing models struggle to fully capture the multi-dimensional nature of SoS, which includes heterogeneous, adaptive, and stochastic components. The operational capacity of such SoS must be managed through effective supervision methodologies to ensure that the desired mission is achieved. However, the literature lacks sufficient methodologies to adequately address this need.

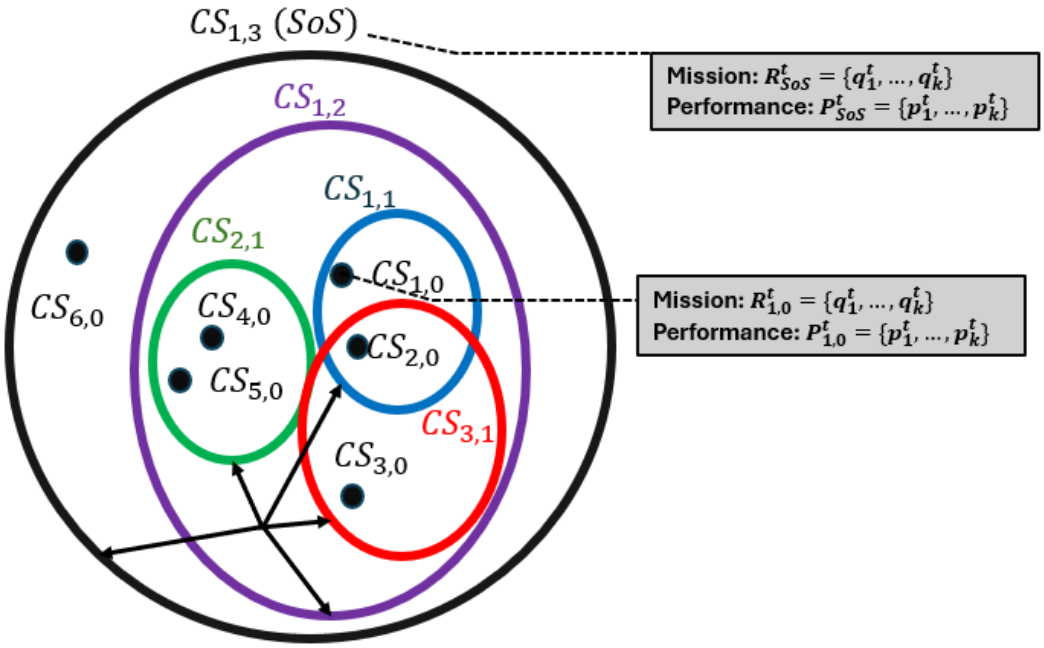


Figure 1.1: Modeling the SoS using a hypergraph (HG) [Khalil 2012]

Consider a multi-level SoS where the n-th CS at level l is denoted as $CS_{n,l}$. The architecture of this system can be described as a graph. In [Khalil 2012], a Hypergraph (HG) is proposed, represented as $H = (V, E)$, where V is the set of vertices and E is the set of hyperedges, to model the SoS. The vertex set $V = \{v_1, v_2, \ldots, v_{a_0}\}$ represents the $CS_{n,0}$ as nodes, where a_0 is the number of CSs at level 0 in the hypergraph. The hyperedge set $E = \{e_1, e_2, \ldots, e_m\}$ represents the relationships between CSs, where m is the total number of CSs at levels higher than 0, as shown in Fig. 1.1. Each CS is assigned a mission composed of k functions represented by the set: $R^t_{SoS} = \{q^t_1, \ldots, q^t_k\}$, and its

corresponding performance set: $P^t_{SoS} = \{p^t_1, \ldots, p^t_k\}$. The mission represents the desired performance, and by comparing the mission and performance, we can identify whether the global mission is satisfied. This performance can be affected by stochastic or deterministic behavior depending on each CS.

The key research statement is to **manage the operational capacity of the SoS by minimizing the residual between the desired mission and the current performance of the SoS, which is composed of heterogeneous, deterministic, or stochastic CSs, in the presence of major disturbances[4]. This ensures the satisfaction of the SoS mission while respecting the main properties of the SoS introduced by [Maier 1998]**.

This can be formulated as follows:

$$
\begin{aligned}
&\text{minimize } |r(t)| = \left| \sum_{s=1}^{k} \left(q^t_s \mid R^t_{SoS} - p^t_s \mid P^t_{SoS} \right) \right|, \\
&\text{subject to :} \\
&\qquad -\lambda \leq r(t) \leq \lambda, \\
&\qquad 0 \leq \varepsilon \leq \varepsilon_{\max}, \\
&\qquad R^t_{n,0} \neq R^t_{n',0} \quad \text{(managerial independence)}, \\
&\qquad q^t_s \mid R^t_{SoS} = \sum_{i=0}^{b} \sum_{p=1}^{a_i} q^t_s \mid R^t_{p,i} \quad \text{(emergent behavior)},
\end{aligned}
\tag{1.1}
$$

where $r(t)$ is the residual, λ is the threshold designed to indicate whether the desired mission is satisfied, ε is the number of disturbances, and $\varepsilon_{\max}$ is the maximum number of disturbances. b is the total number of levels, and a_i is the number of CSs at level i. $R^t_{SoS} = \{q^t_1, \ldots, q^t_k\}$ is the mission set of the SoS, and $P^t_{SoS} = \{p^t_1, \ldots, p^t_k\}$ is the corresponding performance set. It should be noted that the term $q^t_i \mid Y$ will be used frequently in this thesis, which denotes the value of function i in set Y at time t.

Thus, the challenge involves developing models and algorithms that represent different aspects of modeling, accounting for both deterministic and stochastic components, while managing the capacity by providing effective supervision and decision-making support to ensure that the SoS mission is satisfied.

[4]Definition: It refers to internal or external events that disrupt normal SoS operations, leading to abnormal or degraded performance, which requires intervention or adaptation to restore functionality.

1.7 Thesis methodology

To address the problematics of this thesis, the work is divided into three main stages: Modeling and Simulation (M&S), Supervision, and Resilience.

Modeling and Simulation: This stage involves proposing a new framework that describes individual CSs and their interactions. It aims to overcome challenges such as heterogeneity, stochasticity, scalability, and functional decomposition. The model should respect key SoS properties and be validated by modeling and simulating an appropriate case study.

Supervision: This stage focuses on developing algorithms for capacity management to ensure long-term objectives are met. The algorithm will be validated through simulation scenarios.

Resilience: The resilience stage consists of proposing new resilience mechanisms that allow the SoS to recover from failures. Additionally, the resilience of the SoS will be quantified using appropriate metrics.

1.8 Main contributions

Following the methodology presented in the previous section, the following contributions are discussed in this thesis:

- A new framework called the Multi-level Stochastic Hypergraph (MLSHG) model is developed for modeling SoS, addressing the challenges discussed earlier [Chreim 2024b, Chreim 2024a].
- A supervision algorithm, consisting of bottom-up monitoring for detecting disturbances and top-down reconfiguration, is proposed to ensure long-term objectives are met [Chreim 2024b].
- Resilience mechanisms (stand-in and stand-by redundancy) have been introduced for system recovery, and a resilience metric is proposed for quantifying resilience[Chreim 2024a].
- The MLSHG framework and the algorithms are applied to the mushroom harvesting SoS using the Flexsim software tool [Chreim 2024b, Chreim 2024a, Chreim 2023].

1.9 Disseminated results

Journals

- Chreim, Abbass, et al. "Towards Supervision of Stochastic System of Systems Engineering: A Multi-Level Hypergraph Approach." IEEE Access (2024).

International conferences

- Chreim, Abbass, Yiwen Chen, and Rochdi Merzouki. "Stochastic hypergraph model for resilient system of systems: A case study on mushroom harvesting system." 2024 19th Annual SoSE Conference (SoSE). IEEE, 2024.
- Chreim, Abbass, Abdelkader Belarouci, and Rochdi Merzouki. "AI-agent-based modeling for Supervision a System of Systems for Mushroom Harvesting." 2023 18th Annual System of Systems Engineering Conference (SoSe). IEEE, 2023.

1.10 Thesis organization

This thesis manuscript is organized as follows:

- **Chapter 2: State of the art on System of Systems (SoS).** This chapter presents the state of the art on SoS, covering their **history**, **definition**, **properties**, and **applications**. Existing **modeling methods** are discussed along with their limitations to outline the research gap. In terms of **supervision**, only a limited number of studies have addressed this issue, and relevant works are highlighted to guide the development of our methodology. Additionally, **resilience** methods are reviewed, and some existing **resilience metrics** are presented, along with their limitations.
- **Chapter 3: Contribution to organisational modeling of stochastic SoS.** The third chapter introduces our modeling framework, the **Multi-level Stochastic Hypergraph (MLSHG)** model, which describes the **hierarchical structure** of the SoS, the **individual CSs**, which may exhibit stochastic behaviors, and their interactions. A case study on a **mushroom harvesting SoS** is conducted to validate our framework. The M&S of this system are detailed, with a comparison between the proposed model and related studies, focusing on key SoS challenges such as **heterogeneity**, **complex interactions**, **stochasticity**, and **scalability**.

- **Chapter 4: Supervision of stochastic SoS**. In this chapter, we propose our supervision strategy by introducing an algorithm that combines **bottom-up monitoring** and **top-down reconfiguration**. This approach enables **disturbance detection** and implements appropriate **adaptations** to maintain normal operation that serves achieving long term goal. We explain the design of the threshold that triggers reconfiguration alarms. The results of the supervision scenario for the mushroom harvesting SoS are presented, along with a discussion comparing **computational time** with relevant studies.

- **Chapter 5: Resilience of stochastic SoS.** This chapter introduces how the proposed model enhances the resilience of SoS. The supervision algorithm is modified to integrate **resilience mechanisms**, such as **stand-in** and **stand-by redundancy**. Additionally, a **resilience metric** is proposed to effectively measure the system's resilience. The results of the resilience scenario applied to the mushroom harvesting SoS are presented, followed by an in-depth discussion.

- **Chapter 6: Conclusion.** In the final chapter, we summarize the PhD work and its **contributions**, discuss some **limitations**, and outline potential **future directions**.

Chapter 2

System of systems: State of art

Contents

2.1 Introduction

A model is an abstract representation of reality that aims to capture essential aspects of a system without perfectly reconstructing it [Muller 2000]. It focuses on representing the behavior and critical properties of components, while deliberately simplifying or omitting less relevant details. This selective representation is crucial, as models allow for the simulation of systems, which is more cost-effective and practical than real-world testing. In addition to

cost-efficiency, modeling aids in understanding system behaviors, predicting outcomes, and evaluating different strategies for supervision and control.

A model must effectively represent the complexity of the system and its components to evaluate and predict performance accurately, and to explore various supervision strategies. Multi-level modeling [Mostafavi 2011] is particularly valuable for managing this complexity, as it enables the breakdown of the system into more manageable layers or levels of abstraction. This is especially important in the context of SoS, which comprises large-scale components with diverse functionalities that interact dynamically. By using multi-level models, these interactions are better managed and evaluated, allowing for an understanding of how changes at different levels impact the overall system.

Throughout this chapter, the definitions of SoS proposed in the literature will be presented, along with a review of the historical development of this field. The various applications of SoS will be discussed, as well as the existing modeling methods, whether they focus on the dynamics of CSs or organizational aspects. Special attention will be given to graphical techniques and multi-level frameworks that support the development of the methodology later in this thesis. Additionally, since the literature on SoS supervision is limited, several methods and tools that could inform the development of our own supervision methodology will be examined. Finally, resilience methods for these systems will be presented, and the metrics used to quantify resilience will be discussed.

2.2 System of systems terminology

In the dynamic technology and industry world of today, the growing interdependence and complexity of various components require a holistic approach to problem-solving. This is where the concept of **System Engineering (SE)** comes in. A system is more than the sum of its components, it is a complex interconnected element working together to achieve a common goal. In engineering, science, and everyday life, the concept of a system encompasses a dynamic set of components with interdependent functions, collectively contributing to the achievement of specific objectives. [Lightsey 2001] has compiled a list of widely used definitions of SE, and proposes the following summary :

Definition 1 ***systems engineering*** *is an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs.*

As our understanding of systems deepens, it becomes necessary to extend our engineering methodologies beyond individual systems to meet the challenges

posed by their large-scale integration and interaction. This brings us to the field of **System of Systems Engineering (SoSE)**. Whereas traditional systems engineering focuses on optimizing the performance of individual systems, the shift to SoSE involves orchestrating the complexities that arise when these systems interface and collaborate.

2.2.1 History and definition

A System of Systems (SoS) is a collection of interacting, interrelated, and independent Component Systems (CSs), which collaborate to achieve tasks that no individual system can accomplish in isolation. The history of SoS can be traced back to the mid-20th century. While references to it have appeared in various research papers, such as [Ackoff 1971, Jackson 1984, Boulding 1956], the concept has not been clearly defined or comprehensively addressed in the manner we understand SoS today. The evolution of SoS began in the 1990s, driven by the escalating complexity of systems where the foundational theory of SoS was established, introducing crucial definitions and characteristics that recognized the need for a more integrated and collaborative approach to address intricate challenges stemming from the interconnected nature of modern systems, researchers have proposed numerous definitions of SoS based on their understanding of the concept, but there is no universally accepted definition. The first definition of SoS was established by [Eisner 1991], who considers SoS to be a set of independently acquired systems, each subjected to a systems engineering process, whose operations result in a multi-functional solution of an overall mission, his main contribution focused on the role of computer tools in SoS development. In addition, he listed some key features that allow SoS to be applied to any SE case in the context of computer engineering, namely:

- Subsystems are acquired under centralized control.
- Completely autonomous from the program manager.
- Subsystems are coupled and interoperate.
- System is largely uni-functional.
- Trade-offs are carried out to achieve optimal performance.
- System largely satisfies a single mission,

Subsequently, Kotov [Kotov 1997] considered SoS as large-scale, distributed, concurrent systems that are composed of complex systems. Maier [Maier 1998] has also identified a SoS as a set of collaboratively integrated

systems that possess operational and management independence of their components. He proposed a taxonomy and basic principles that are used today by many researchers and engineers in many fields when designing a SoS. In addition, Krygiel [Krygiel 1999] defines a SoS as a set of different systems so connected or related that they produce results impossible to achieve by the individual alone.

Besides defining the concepts of SoS, researchers have significantly contributed to the evolution of SoS through their work in various ways. For instance, Carlock and Fenton [Carlock 2001] highlighted that enterprise SoS focuses on integrating traditional systems engineering activities with the strategic planning and investment analysis of the enterprise. In the same year, Sage and Cuppan [Sage 2001] proposed that an SoS exists when the five Maier characteristics are applied, providing a comprehensive study of SoS in engineering and management terms.

Furthermore, Keating et al. [Keating 2003] defined SoS as a meta-system consisting of complex, autonomous, and integrated systems with diverse technologies, operations, and geographical areas. They also conducted a detailed review of SoS in terms of design, deployment, and operation, discussing various perspectives and potential future work. Bar-Yam et al. [Bar-Yam 2004] explored specific characteristics of SoS such as evolutionary development, emergent behavior, self-organization, and adaptation, drawing insights from the fields of biology, sociology, and the military.

DwLaurentis et al. [DeLaurentis 2005b] delved into the perspectives of SoS in decision-making within transportation systems, while Boardman and Sauser [Boardman 2006] introduced new characteristics of SoS, including autonomy, membership, connectivity, divergence, and emergence. They differentiated between systems and SoS in terms of structure, behavior, and realization. Sloane et al. [Sloane 2007] proposed an architecture for SoS modeling, where CSs act as service providers guiding the emergence of SoS.

Simpson and Dagli [Simpson 2008] analyzed the characteristics and attributes of SoS, encompassing flexibility, adaptability, modular design, open interfaces, and more. Gorod [Gorod 2008] provided a detailed overview of SoS, while DeLaurentis [DeLaurentis 2008] identified a taxonomy for modeling and analyzing SoS. Collectively, these contributions have enriched the understanding and development of SoS in various domains.

In 2008, Jamshidi made a significant contribution to the field of SoS by presenting two comprehensive books [Jamshidi 2008a, Jamshidi 2008b] covering various aspects, including principles, architecture, and applications. In these books, Jamshidi not only addressed the fundamental principles but also explored the complex architectures underlying systems of systems, highlighting the intricate interrelationships between the CSs. His work went beyond mere

compilation, as he provided insightful analyses and perspectives on how these principles and architectures manifest themselves in real-world applications. He has compiled a wealth of knowledge from previous work, synthesizing and refining existing definitions to present a more coherent and nuanced understanding of SoS:

Definition 2 *System of systems are large-scale integrated systems that are heterogeneous and independently operable on their own but are networked together for a common goal.*

Definition 3 *System of systems is a 'supersystem' comprised of other elements that themselves are independent complex operational systems and interact among themselves to achieve a common goal. Each element of a SoS achieves well-substantiated goals even if they are detached from the rest of the SoS.*

Furthermore, DiMario [DiMario 2009] explored the collaboration among autonomous systems within the SoS framework, setting a foundation for understanding system interactions. Building on this, Sauser et al. [Sauser 2010a] employed biological analogies to offer insights into SoS behavior, offering a fresh perspective on system interactions. In parallel, Ender [Ender 2010] introduced a M&S framework that supports architecture-level analysis, particularly in defense applications, where the use of neural networks plays a key role in simulating complex behaviors.

Similarly, Dauby and Upholzer [Dauby 2011] suggested an approach using evolutionary algorithms and ABM, which provided increased flexibility and autonomy in SoS simulations. Meanwhile, Cooksey and Mavris [Cooksey 2011] contributed by applying game theory to SoS modeling, expanding the range of strategic decision-making tools available. Complementing these efforts, Zhou et al. [Zhou 2011] addressed broader issues in System of Systems Engineering (SoSE) and proposed a computational method for SoS modeling after reviewing existing approaches.

Khalil *et al.* [Khalil 2012] further advanced SoS modeling by introducing a graphical approach based on hypergraphs, offering a visual and mathematical representation of system interactions. Similarly, Darabi and Mansouri [Darabi 2013] focused on competition and collaboration within SoS, modeling these dynamics to assess their effects on autonomy and belongingness among constituent systems. In a related effort, Klein and Vliet [Klein 2013] performed a systematic review of SoS architecture research, categorizing and analyzing the key contributions in this field.

In the context of mission-critical systems, Silva et al. [Silva 2015] developed mKAOS, a conceptual model that specifically addresses mission description in SoS, filling a gap in the existing literature. In addition, Vierhauser et

al. [Vierhauser 2016] tackled the challenges of SoS monitoring by proposing a flexible framework that can be applied to various architectures and technologies, enhancing system oversight. Similarly focusing on system architecture, Bondar et al. [Bondar 2017] explored emergent behavior in SoS and recommended the use of agent-based simulation and integration with SysML and UML to guide the development of SoS architectural models. Their work also evaluated several architecture frameworks such as Zachman, TOGAF, and FEAF.

Shifting to performance measurement, Bourne et al. [Bourne 2018] proposed a new SoS-based perspective on performance measurement and management (PMM), emphasizing key features like autonomy, connectivity, and emergence. This shift is argued to be more suitable for handling complex and uncertain environments compared to traditional PMM approaches based on control systems. In addition, Uslar et al. [Uslar 2019] contributed to the power and energy domains by offering a comprehensive framework for designing and validating complex systems.

Fortino et al. [Fortino 2020] expanded this research into the Internet of Things (IoT) domain, reviewing methodologies and tools within the SoSE framework. They highlighted the key characteristics of IoT systems such as interoperability, scalability, and autonomy. On a broader scale, Iwanaga et al. [Iwanaga 2021] called for a holistic SoS research approach, emphasizing the involvement of researchers, stakeholders, and policymakers in co-creating solutions for socio-environmental challenges.

Kozma et al. [Kozma 2021] introduced the System of Systems Lifecycle Management (SoSLM) approach, which extends the Product Lifecycle Management (PLM) model to address the challenges of deploying, operating, and maintaining large-scale SoS in automation systems. Their work bridged the gap between lifecycle management and SoS.

Chatterjee et al. [Chatterjee 2022] focused on designing SoS architectures to balance resilience and affordability, providing insights into achieving optimal trade-offs. In a more technical approach, Raman et al. [Raman 2023] used machine learning classifiers and formal methods to analyze emergent behavior in SoS. Their method, demonstrated through a case study involving autonomous UAV swarms, tackles the challenge of ensuring confidence in complex systems with emergent properties, especially in SoS scenarios. Finally, Huang et al. [Huang 2023] addressed the task-oriented SoS architecture selection problem by proposing a hybrid solution that combines reinforcement learning and evolutionary algorithms to optimize architecture design.

The previously reviewed works of literature in this discussion constitute invaluable chronicles that illuminate the historical evolution of SoS and SoSE over time. These works trace the path of these concepts as they respond to the

growing complexity of modern systems. Focusing on the development of SoS, these writings not only dissect the theoretical foundations, but also propose practical solutions to a whole range of challenges. From structural analysis and control to supervision, fault detection and isolation, M&S, optimization and decision-making, these works constitute a comprehensive exploration of the multifaceted dimensions of SoS and SoSE. In so doing, they not only contribute to the theoretical foundations of these fields, but also provide actionable insights and methodologies. Furthermore, these literary works delve into crucial aspects of resilience and sustainability, recognizing the evolving landscape of systems engineering and the imperative to address contemporary issues in a holistic, forward-looking manner.

2.2.2 Properties

Despite the numerous definitions of SoS proposed in the literature, a universally accepted definition remains elusive, highlighting the evolving and dynamic nature of the field. Various researchers have offered different conceptualizations, reflecting the complexity and multi-dimensionality of SoS. However, despite this diversity, certain common characteristics consistently emerge, although expressed differently across studies. Key properties such as emergent behavior, interoperability, and the integration of independent systems form the foundation upon which a coherent understanding of SoS can be established. By acknowledging and embracing these fundamental characteristics, researchers can better navigate the complexities surrounding SoS, paving the way toward a more refined and widely applicable definition. While a singular definition may remain elusive, the collective recognition of these essential properties provides a practical framework for the ongoing development and refinement of the SoS concept.

In this work, the focus is on considering the five fundamental properties proposed by Maier [Maier 1998] as a baseline for developing SoS. These distinctive traits set SoS apart from typical complex systems and serve as a guiding framework for a more precise understanding of their nature. These characteristics must be present in all constituent or Component Systems (CSs), namely:

- **Operational independence:** A CS is required to operate independently and function autonomously, irrespective of the presence or connection to other CSs within the SoS. Even when detached from the larger SoS framework, a CS maintains its distinct operational mode and possesses dedicated resources.
- **Managerial independence:** Each CS within the SoS manages its own

operations separately without regard of the other CSs. These operations serve a useful independent mission that is different from the mission of the other CSs and the SoS mission.

- **Geographic dispersion:** All CSs are dispersed over a vast geographical area and exchange information as part of their interactions, but no physical elements or energy are shared.

- **Emergent behavior:** The global mission and functionality of the SoS is achieved by the emergent behavior arising from the collaboration of all CSs. This crucial characteristic underscores the cooperative nature of CSs, emphasizing that the collective actions of multiple systems, rather than any single one, are essential for achieving the global objectives and functionality of the SoS.

- **Evolutionary and adaptive development:** SoS are constantly evolving over time. This implies that organizational structure and functions adapt dynamically, along with mission operations, in response to overall mission adjustments or external disturbances. This evolution may involve the addition or elimination of CSs to meet changing requirements and challenges faced by the SoS.

When modeling SoS, it is crucial to adhere to the set of fundamental properties stated above that collectively define the basis of these systems. These properties serve as a decisive test to determine whether an aggregation of elements and CSs truly constitutes a SoS. Indeed, any entity aspiring to be recognized as a SoS must demonstrate that it adheres to these properties. However, in cases where certain CSs show notable alignment with most of these properties while lacking others, an alternative classification known as a Virtual Component System (VCS), as proposed by Soyez [Soyez 2015], becomes relevant. It should be noted that some of these properties should be verified for individual CSs, including managerial independence, operational independence, and geographic distribution. Other properties, such as emergent behavior and evolutionary development, must be verified at the global SoS level.

2.2.3 Applications

SoS is applied in a variety of fields, offering solutions to complex problems by integrating multiple individual systems into a coherent whole. In defense [Owens 1996], SoS are used for command and control [Tran 2016], where interconnected military systems work in synergy to improve situational awareness

and decision-making. In the transport sector [DeLaurentis 2005b], SoS facilitates the efficient management of interconnected networks, optimizing traffic flow and enhancing safety. Furthermore, SoS principles are being applied in healthcare [Price 2022] to integrate various medical systems, enabling patient care and data exchange between healthcare providers in a seamless manner. In agriculture, SoS approach plays a crucial role in optimizing farming operations and increasing agricultural productivity. It enhances understanding of interoperability across heterogeneous infrastructure [Weinert 2020a], facilitating seamless integration and interaction among various agricultural systems. These applications and others are showcased in Table 2.1, highlighting the versatility and importance of the SoS paradigm for addressing complex challenges across various sectors.

Domain	Reference
Defence	[Owens 1996] [Tran 2016] [Lubas 2017]
Agriculture	[Weinert 2020b]
Healthcare	[Sloane 2008] [Price 2022]
Transportation	[DeLaurentis 2005a]
Energy management	[Zhao 2018]
Maritime	[Mahulkar 2009] [Mansouri 2009]
Civil aviation	[Mordecai 2016]
Biology	[Sauser 2010b]
Smart grid	[Ibne Hossain 2020]
Unmanned aerial vehicle swarm	[Kerr 2020]

Table 2.1: SoS applications

2.3 Modeling methods

In the field of complex SE, SoS modeling stands out as a vital discipline facilitating the understanding, analysis and design of interconnected and independent systems at different scales. SoS modeling encompasses a multidimensional approach, incorporating behavioral, and organizational methods. Each facet plays an essential role in fully capturing the complex dynamics and emergent behaviors inherent in SoS architectures.

Behavioral modeling is the fundamental framework for understanding how the individual components of an SoS interact and function. It examines the actions, processes and decision-making mechanisms of each CS, representing their operational characteristics, dynamic behavior and performance under

various conditions. By abstracting complex behaviors into manageable representations, behavioral modeling makes it possible to predict system responses and identify potential failures.

Organisational or Structural modeling, on the other hand, focuses on clarifying the physical and organisational structure of interconnected CSs within an SoS. It examines the relationships, dependencies, and interfaces among various CSs and their interactions, thereby facilitating the visualization and analysis of system architectures. Through structural modeling, engineers can gain insights into system composition, hierarchy, and connectivity, crucial for assessing scalability, adaptability, and resilience of SoS configurations.

In the literature, there are few studies on behavioral modeling in SoS, as it does not directly address the main objective of ensuring the common goal is achieved across the system. Instead, it tends to focus on individual components. However, most existing modeling methods describe the organizational structure of SoS, which can be divided into different classes. In the following section, we present the most relevant existing modeling methods.

2.3.1 Behavioral modeling

As previously stated, behavioral modeling refers to understanding the behavioral dynamics of physical components. Below, existing tools are cited, namely Bond Graph (BG) and System Dynamics (SD), for behavioral modeling.

2.3.1.1 Bond Graph

A **Bond Graph (BG)** is a graphical representation of a physical dynamic system, depicting the flow of energy between system components through power bonds. It can describe the flow of energy between Physical Component Systems (PCSs)), allowing us to understand the interaction between them using pairs of vertices and junctions $\{S, A\}$. For instance, [Kumar 2017] utilized BG to model the dynamics of vehicles , aiming to enhance traffic flow, reduce congestion, and contribute to sustainable transport solutions (Fig. 2.1).

2.3.1.2 System Dynamics

On the other hand, **System Dynamics (SD)** [Barbrook-Johnson 2022] is a computational modeling method used to analyze the behavior and dynamics of complex systems over time. This approach utilizes feedback systems theory and is commonly implemented through simulation models to understand, design, and predict system behaviors under various scenarios. SD provides a methodology for modeling and testing dynamic behavioral hypotheses, offering an equivalent model to mathematical formulations [Sohn 1985]. For

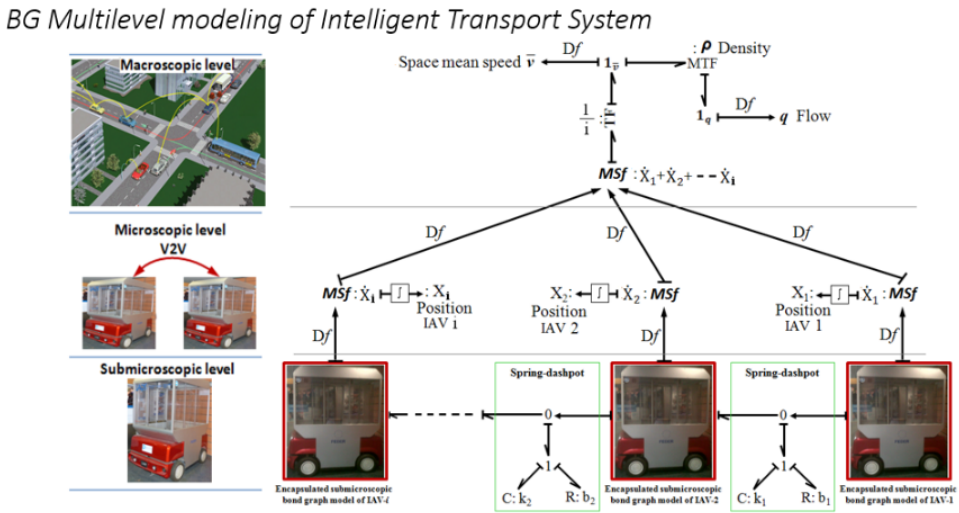


Figure 2.1: BG multi-level modeling of Intelligent Transport System [Kumar 2017]

example, [DeLaurentis 2005a] discussed the application of SD and Control Theory Robustness methods in understanding future transportation SoSs.

2.3.2 Organisational modeling

While some studies have addressed behavioral modeling, most researchers focus on **organizational modeling**, as it offers a better understanding of the structure of interconnected subsystems within complex and **highly heterogeneous SoS**. This approach facilitates mission and functional decomposition, leading to more efficient design, analysis, and optimization of the overall system. Researchers primarily concentrate on modeling the organizational aspects of SoS through various architectural frameworks, often without delving deeply into the behavioral dynamics of individual constituent systems.

To this end, various modeling tools and techniques have been developed and employed for the structural and organizational analysis of SoS and complex systems. These tools help visualize, analyze, and optimize the relationships and interactions among subsystems. Below are some commonly used modeling tools and methods, categorized into three main classes: **AI-based models**, **Agent-Based Modeling (ABM)**, and **graphical modeling**, as shown in Fig. 2.2.

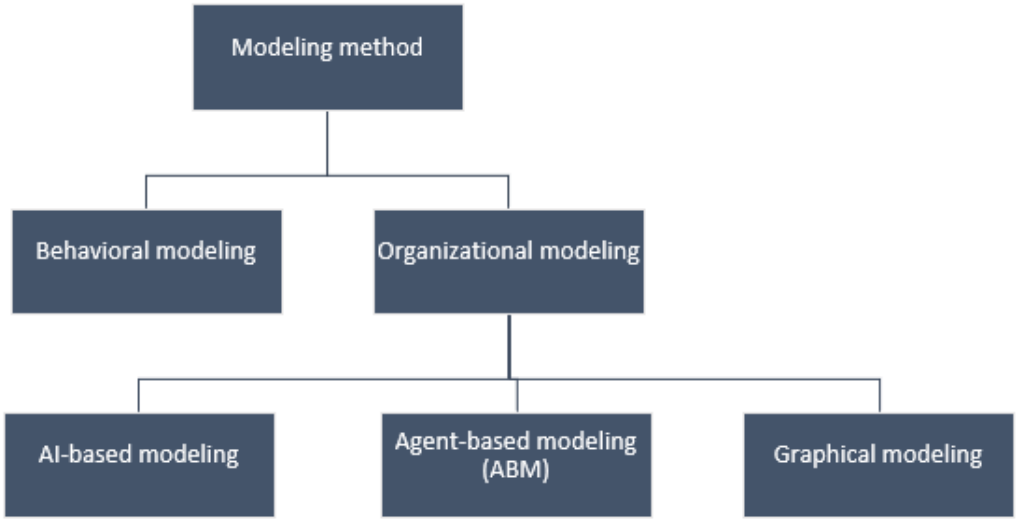


Figure 2.2: SoS modeling methods

2.3.2.1 AI-based modeling

AI-based modeling leverages artificial intelligence to optimize decision-making, communication, and coordination between interconnected CSs within a SoS. This approach significantly enhances the organizational aspects of SoS by enabling dynamic adaptability, efficient resource allocation, and seamless interoperability, which collectively improve the overall management and collaboration within complex system networks. For example, [Lin 2023] introduced a deep reinforcement learning approach to optimize task allocation and coordination between CSs, ensuring that systems are assigned tasks according to their capabilities, much like how roles are defined in an organization. Additionally, this approach supports dynamic reconfiguration, allowing systems to adapt to evolving conditions, which mirrors how organizations evolve to meet new challenges. This not only enhances the modeling of SoS in terms of structure but also in how systems collaborate to achieve shared objectives. However, the paper focuses on relatively simple interactions between systems performing the same task, whereas real-world interactions tend to be more complex.

Moreover, [Nilsson 2024] proposed AI-driven techniques to improve dynamic communication and adaptability between heterogeneous systems. These techniques include AI-based methods for automatic message translation, ontology alignment, and autonomous decision-making, enabling systems to collaborate seamlessly and reconfigure without manual intervention. Such capabilities help manage the coordination and interaction between systems,

ensuring that the SoS functions efficiently while adapting to changes, thus improving both the organizational structure and functionality of the SoS.

Despite these advancements, AI-driven solutions face notable **limitations**. They heavily depend on the availability of **large, high-quality datasets** for training, which may not always be accessible, especially in real-world SoS environments characterized by diverse systems. Furthermore, implementing AI solutions in large, heterogeneous SoS can require sophisticated architectures and significant computational resources, which can lead to substantial overhead and practical challenges.

2.3.2.2 Agent-based modeling

Agent-Based Modeling (ABM) has been widely used in modeling SoS [Feng 2023, de Amorim Silva 2020], where **agents** operate autonomously, interacting with their environment and communicating with other agents. ABM involves simulating the interactions and behaviors of autonomous agents, each representing a CS, which helps capture emergent behaviors and complex dynamics within the SoS. This approach enables a deeper understanding of how individual systems collaborate and adapt within a larger network. For example, [Soyez 2015] employed a multilevel ABM to explore both static and dynamic SoS, focusing on how systems evolve in terms of overall goals and missions. The study conducted a comparative analysis between ABM and Multi-agent Systems (MAS), where agents operate based on their individual perspectives, communicate with other agents, and adhere to their own rules and resources.

[Soyez 2015] described the SoS as a group of CSs (agents), with attributes that depict both static and dynamic behaviors within the SoS framework, as shown in Figure 2.3. The study classified Level 0 physical CSs as elementary CSs, while higher-level CSs were termed top CSs. Additionally, virtual CSs were introduced to represent components that temporarily deviate from the typical properties of SoS. This modeling framework was applied to intelligent autonomous vehicles within the European InTraDE (Intelligent Transportation for Dynamic Environment) project, aimed at automating port container logistics.

Although ABM is a powerful tool for modeling SoS, it faces challenges in accurately capturing the full range of emergent behaviors in large-scale systems, and decision-making becomes more complex as the scale and intricacy of the system increase.

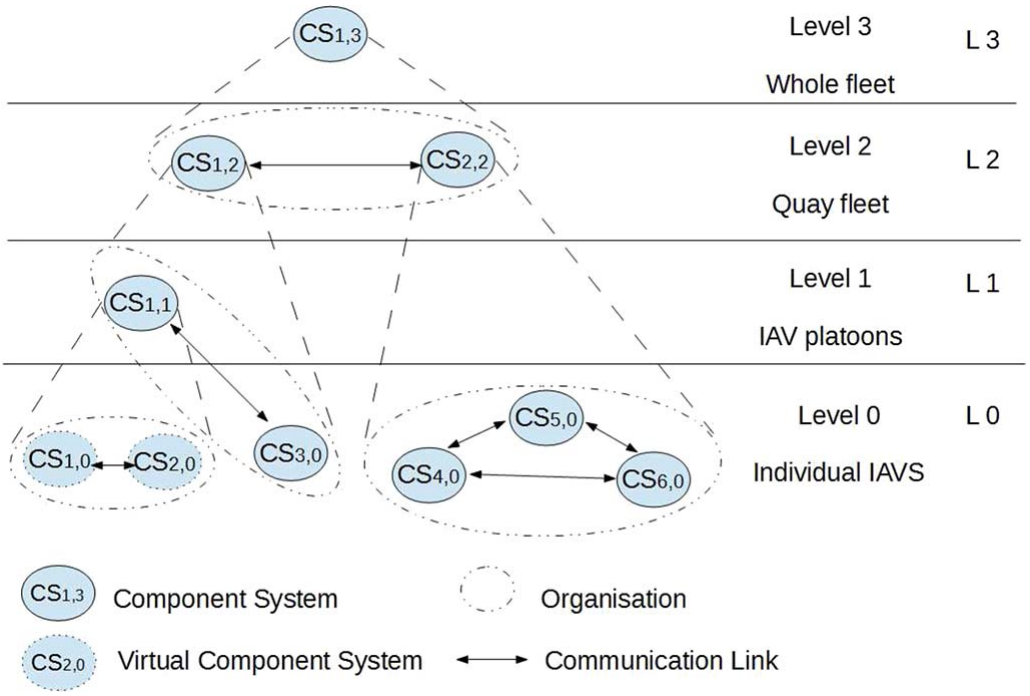


Figure 2.3: Graphical representation of multi-level ABM by [Soyez 2015]

2.3.2.3 Graphical modeling

Graphical approaches offer suitable solutions for describing the structure of a SoS. They aid designers in examining various organizational aspects and dependencies. It help to capture the relationships and interactions within complex systems [Christensen 2007]. By representing entities as **nodes** and their interactions as **edges**, graph models can effectively map out the intricate web of connections that helps in understanding system behavior, optimizing performance, and improving decision-making in the design and operation of SoS [Harrison 2016]. This method is particularly useful in SoS, where multiple independent systems collaborate to achieve higher-level objectives. In this section, various graphical models are discussed, including: **bi-graph**, **simple graph**, **directed graph**, **undirected graph**, **weighted graph**, **Hypergraph**, and **weighted and stochastic Hypergraph**. Additionally, it is explained how some of these models have been applied to SoS modeling.

- **Bi-Graph:** A bigraph is a graphical tool that consists of two graphs: one representing the placement of CS nodes and the other representing the relations between them. The link graph in bigraphs describes the interaction patterns among systems, enabling the representation of direct connections and interrelations between systems in a given scenario.

[Wachholder 2015] used bipartite graphs (Fig. 2.4) to address the lack of research in modeling approaches that consider both the structure and dynamics of SoS for emergent behavior. They explored the potential of bi-graphs in enabling context-sensitive interaction and reaction rule orchestration, which is essential for coping with the increasing complexity of modern software systems.

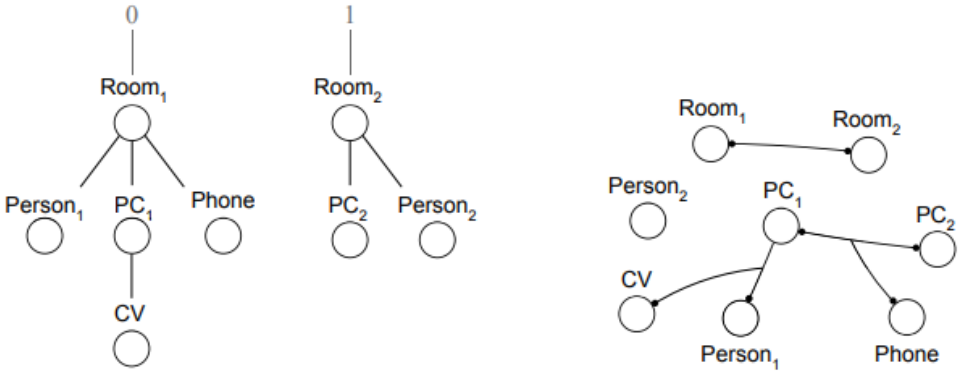


Figure 2.4: Bigraph proposed by [Wachholder 2015]: Place graph on the left, Link graph on the right

- **Simple Graph: Simple graphs** consist of a finite set of nodes (also called vertices) connected by edges (links) that represent elements and their relationships. Consider a graph $G = (V, E)$ with a vertex set V and an edge set E. The vertex set $V = \{v_1, v_2, \ldots, v_n\}$, where v_i for $i = 1, 2, \ldots, n$ represents the vertices in the graph. The edge set $E = \{e_{i,j} = \{v_i, v_j\} \mid v_i, v_j \in V \text{ and } v_i \neq v_j\}$ represents the relationships between two vertices.

  A **non-directed graph** (see Fig. 2.5) $G = (V, E)$ is a simple graph with no orientation in the edges, while a directed graph $G = (V, E)$, where $E = \{e_{i,j} = (v_i, v_j) \mid v_i, v_j \in V \text{ and } v_i \neq v_j\}$, is a simple graph with directions on the edges, as shown in Fig. 2.6.

  A **weighted graph** $G = (V, E, W)$ is a simple graph with information associated with each edge called weights. The weight set $W = \{w_{i,j} = w(v_i, v_j) \mid v_i, v_j \in V \text{ and } v_i \neq v_j\}$ provides supplementary insights about the relationship between the vertices, as shown in Fig. 2.7. Moreover, stochastic graph [Rezvanian 2016] in which weights associated with edges are random variables is a better graph model for real word application where the nature varies with time.

  Graphs have been used for modeling processes in complex systems [Jalving 2019]. For example, [Myhan 2023] used graphs for modeling

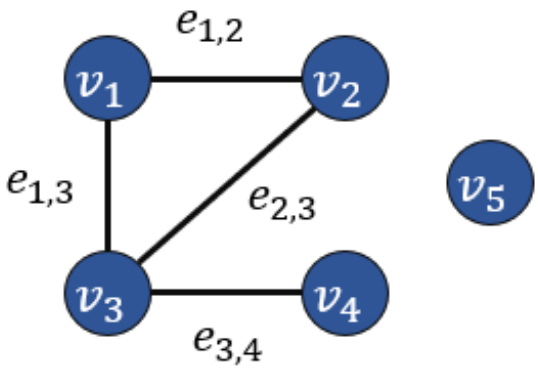


Figure 2.5: Non directed graph

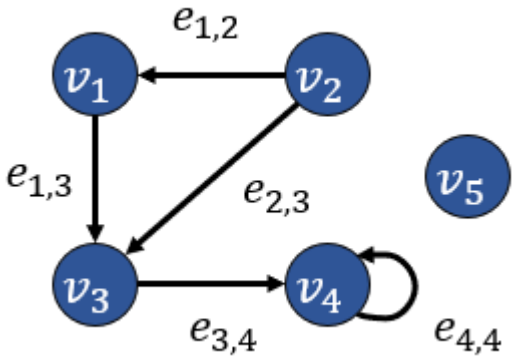


Figure 2.6: Directed graph

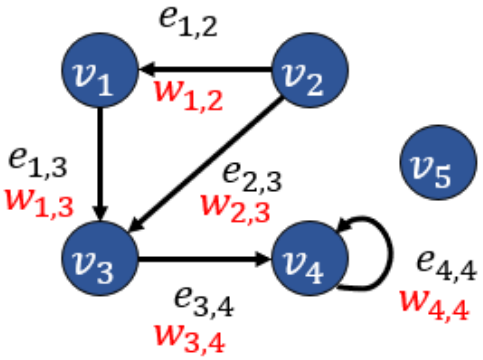


Figure 2.7: Weighted graph

and analyzing the structure of a grain drying line, demonstrating the effectiveness of these approaches in the agri-food industry. However, in complex systems where the structure possesses multiple dependencies, simple graphs cannot represent these dependencies, as edges only represent the relation between two vertices.

- **Hypergraph:** A **Hypergraph** (**HG**) is a generalization of a graph where edges, called hyperedges, can connect any number of vertices, not just two. In this context, [Khalil 2012] introduced a Hypergraph model for SoS modeling, which allows for a detailed representation of SoS by capturing multi-way relationships that traditional graph models cannot depict Fig. 2.8. This approach proves particularly useful in SoS, as it illustrates the intricate network of dependencies and interactions among various CSs.

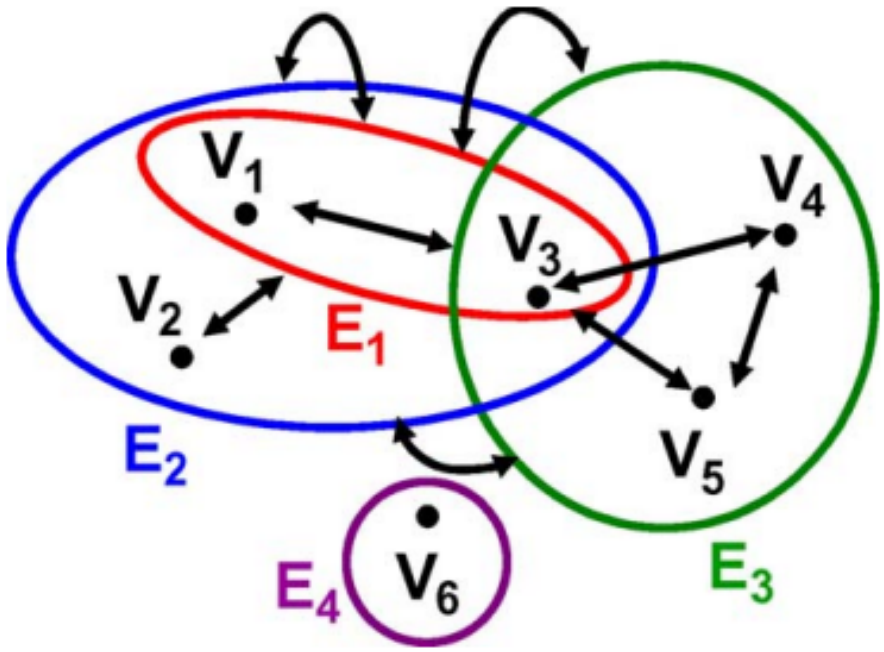


Figure 2.8: Hypergraph composed of six vertices and four hyperedges

From a mathematical point of view, a **HG** can be depicted as $H = (V, E)$ with a **vertex** set V and a **hyperedge** set E. The vertex set $V = \{v_1, v_2, ..., v_n\}$, where $v_i, i = 1, 2, ..., n$ represents a vertex in the HG. The hyperedge set $E = \{e_1, e_2, ..., e_m\}$, where $e_j = \{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$. The interaction and the information exchange between hyperedges is what differentiates hypergraphs from hypersets [Chemero 2008].

In the context of multi-level SoS modeling, the physical CSs in level 0 ($CS_{n,0}$) are represented as vertices, and the managerial CSs ($CS_{n',l}$) are presented as hyperedges in higher levels as shown in Fig. 2.9. There are two ways to represent graphically the HG for a specific SoS: the first one is similar to the **set-based representation**, where we add links between the hyperedges, and the second one is the **hierarchical representation** as illustrated in Fig. 2.10.

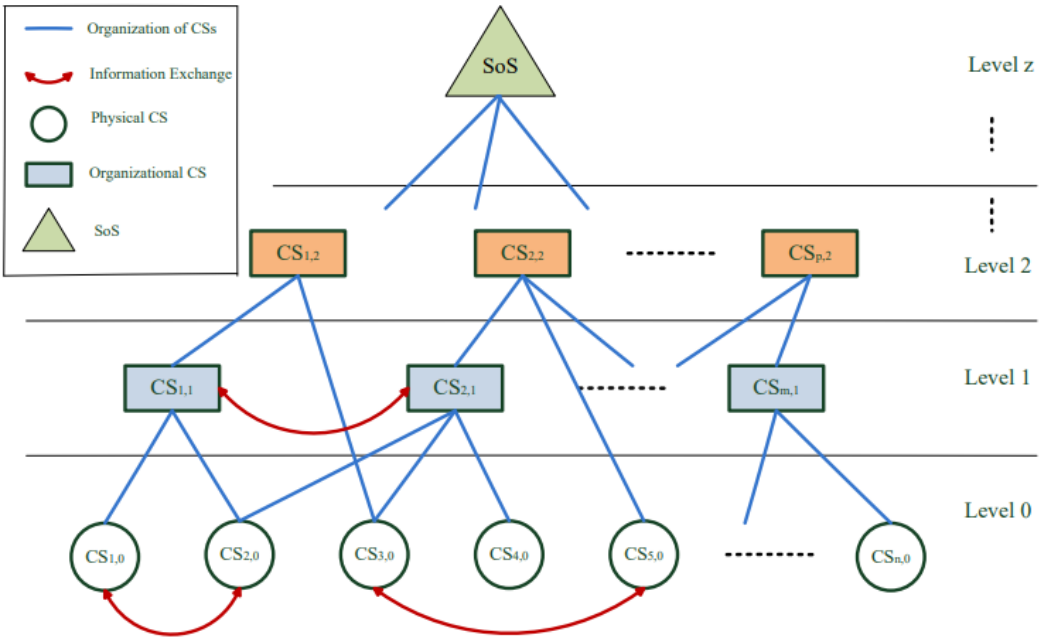


Figure 2.9: Hierarchical representation of HG by [Khalil 2012]

The Hypergraph (HG) proposed by [Khalil 2012] has been used to model a SoS as a Constraint Satisfaction Problem (CSP). This approach calculates satisfaction degrees and key performance indicators, facilitating Fault Detection and Isolation (FDI) of various CSs. Additionally, the author introduced an algorithm combining top-down reconfiguration with bottom-up monitoring for robust, online supervision. However, this method was limited to a specific type of SoS with low heterogeneity, specifically mechatronic systems, and cannot be easily applied to highly heterogeneous SoS that include stochastic components.

- **Stochastic and weighted Hypergraph:** A weighted HG $H = (V, E, W)$ has weights associated with its hyperedges, augmenting traditional hypergraphs by adding a quantitative measure to the connections between nodes. This enhancement offers supplementary insights into

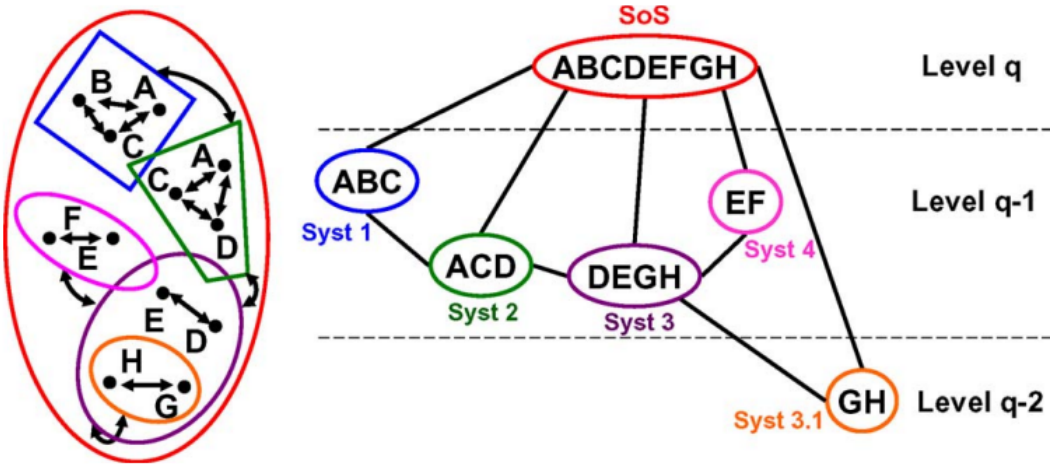


Figure 2.10: HG set-based representation (left) and the graphical Hierarchical representation (Right) [Khalil 2012]

the relationships among nodes within the same hyperedge and accommodates more complex associations among multiple participants.

The mathematical representation of weighted HG is represented by $H = (V, E, W)$ with vertex set V and hyperedge set E an weight set W . The vertex set $V = \{v_1, v_2, ..., v_n\}$, where $v_i, i = 1, 2, ..., n$ represents the vertex in HG. The hyperedge set $E = \{e_1, e_2, ..., e_m\}$, where $e_j = \{v_{i_1}, v_{i_2}, ..., v_{i_k}\}$. The indicator function of HG H is defined as follows:

$$I(v_i, e_j) = \begin{cases} 1, if\ v_i \in e_j, \\ 0, otherwise. \end{cases} \tag{2.1}$$

Let $W = [w_{ij}]$ denote the weight of vertex v_i in hyperedge e_j, where

$$w_{ij} \neq 0, if\ v_i \in e_j \tag{2.2}$$

In addition, w_{ij} may not be equal to $w_{ij'}$ if $j \neq j'$ because the weight of each vertex may vary at different hyperedge. Let $|e_j|$ denote the cardinality of hyperedge e_j, that is, the number of vertices in hyperedge e_j. If for arbitrary $j = 1, 2, ..., m$, $|e_j| = 2$, HG H degenerates to an undirected graph.

Weighted hypergraphs have been studied across various domains. For example, Chodrow [Chodrow 2023] introduced nonbacktracking spectral clustering specifically designed for nonuniform hypergraphs. Building upon this, Galuppi [Galuppi 2023] extensively investigated the spectral theory of weighted hypergraphs, while Stephan [Stephan 2022] explored

the non-backtracking spectra in sparse random hypergraphs. Additionally, Ko [Ko 2022] scrutinized growth patterns in real-world hypergraphs, significantly enhancing our comprehension of their evolutionary dynamics.

Recent advancements in weighted and stochastic hypergraphs have provided new insights into complex relational systems. For example, [Ran 2020] applied HG models in vehicular fog computing to represent cooperative networks of vehicles, underscoring the significance of stochastic modeling in dynamic and unpredictable environments. Similarly, [Bassoli 2021] investigated energy and latency in 5G cloud radio access networks, demonstrating how stochastic HG models can enhance system performance under uncertainty.

Weighted hypergraphs can also represent the **addition and retrieval of nodes**, as demonstrated by Guo [Guo 2014], making them suitable for SoS modeling where the number of CSs may change stochastically. They play an important role in capturing and understanding the inherent uncertainties, complexities, and dynamic behaviors of CSs and the SoS as a whole. In adition, Weighted hypergraphs help capture the probability-dependent interactions between these CSs, impacting the organizational modeling and emergent behavior of the SoS, as CSs can appear or disappear unpredictably or based on specific distributions [Mohsin 2019].

- **Bayesian network: Bayesian Networks** are graphical models that represent probabilistic relationships among a set of variables. They are widely used for analyzing component failures in complex systems [El-Awady 2023]. In these networks, nodes represent variables, and edges represent probabilistic dependencies between them, as shown in Fig. 2.11.

  In the context of SoS modeling, Bayesian Networks are used to account uncertainties presented in interactions and behaviors, for example, [Han 2013] utilized event tree methods and Bayesian Networks to quantify interdependencies and assess risks by analyzing how disruptions in one system can propagate to others. Bayesian Networks were employed to represent causal relationships between systems under uncertainty, helping to quantify interdependencies by including various parameters and factors to estimate how systems affect each other. Although this work addresses the quantification of interdependency with uncertainties, it lacks tools to assist decision-makers in making better decisions during the SoS development process.

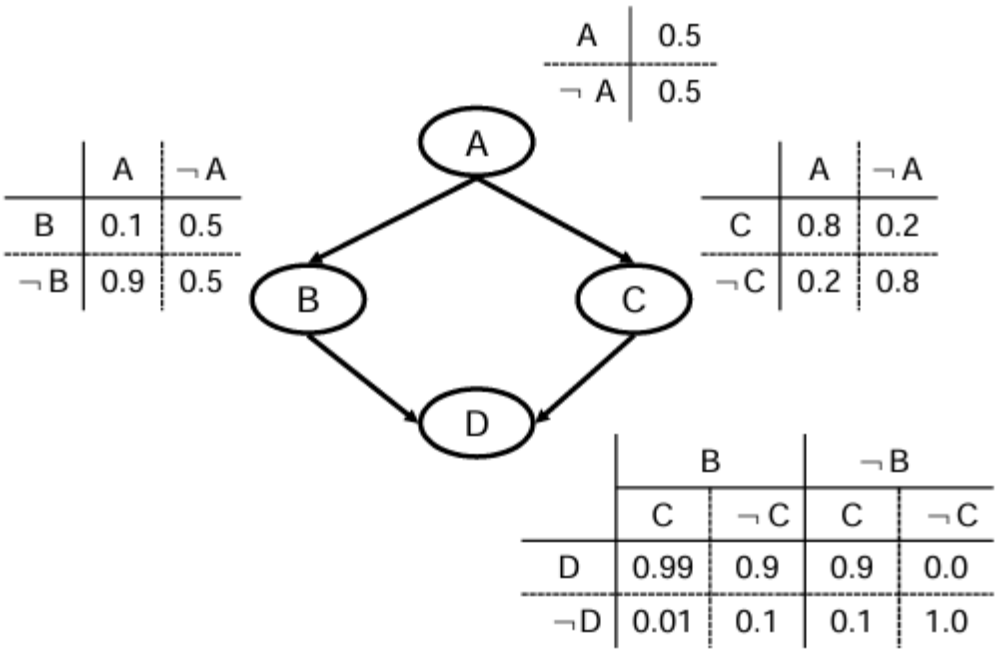


Figure 2.11: Baysian network

The summary of these SoS modeling methods is presented in Fig. 2.12.

2.4 Supervision of SoS

System supervision involves a series of activities that **monitor** and **manage** the performance, security, and functionality of technological systems. It ranges from the tracking of system operation to guarantee proper performance and identification of the root cause of problems in implementation, which updates and maintains security protocols. Proper supervision of the system will ensure the reliability, efficiency, and security of the components within the system, which translates to an operational support for organizations and prevention of downtime.

2.4.1 Definition

In **intelligent systems**, supervision encompasses not only capturing a snapshot of the process but also adapting to new situations where potential risks arise, requiring the detection of deviations from desired outcomes for effective adaptation [Gleirscher 2023].

In **social systems**, supervision is an intricate field encompassing various aspects of social management and control [Hämberg 2013] while some

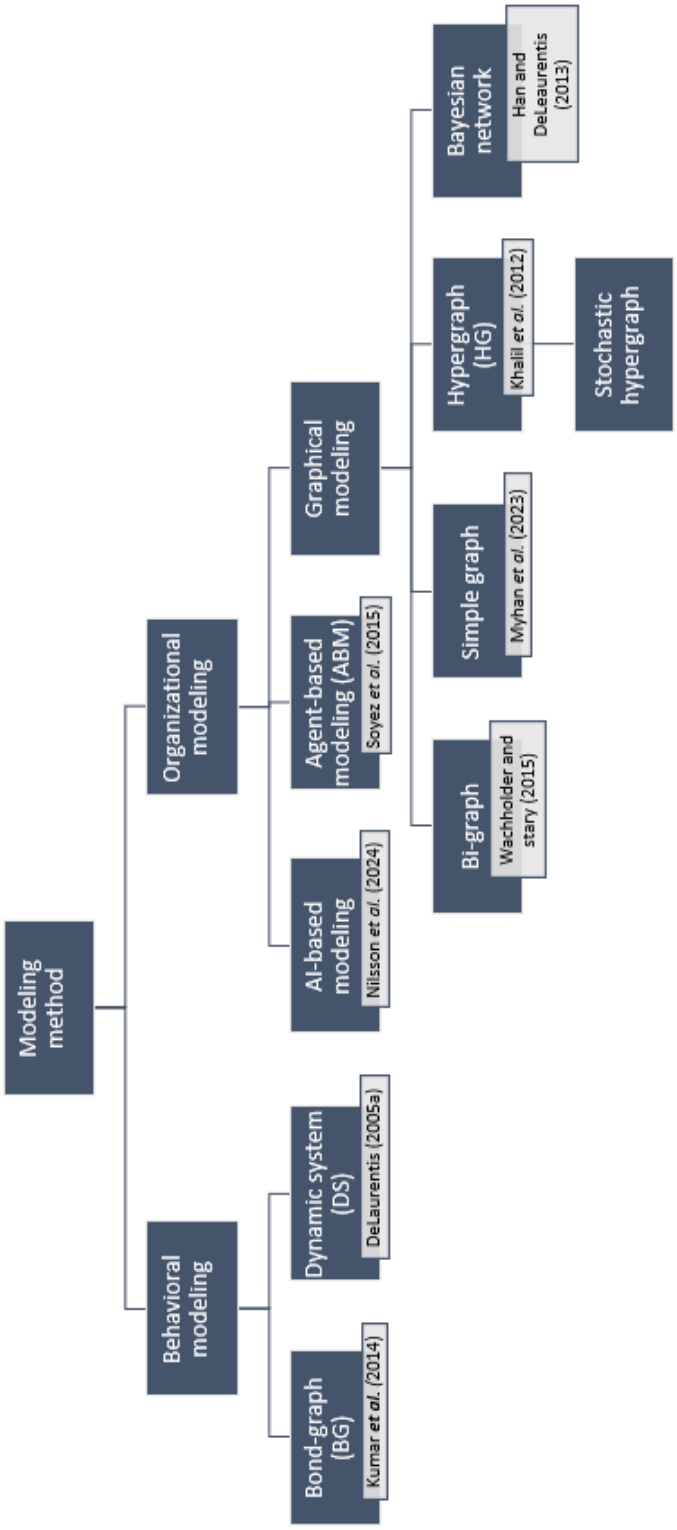


Figure 2.12: SOS modeling methodes

researchers differentiate supervision from control by emphasizing that supervision involves management and instruction, aiming to monitor the actual state of affairs.

There are other definitions of supervision. For example, **process supervision** is a set of activities for determining the state or condition of the process and controlling it to ensure that it remains in a safe regime [Natarajan 2012]. The condition determines whether the process is normal or abnormal, with deviations from normal behavior being detected as abnormalities.

While most studies focus on the Modeling and Simulation (M&S) of SoS [Mittal 2015], there is limited research on the supervision of these complex systems. **SoS supervision** involves **methods and tools** used to **maintain system performance**, ensuring that the SoS operates as expected and meets its global objectives, even in the presence of disturbances. It consists of handling disturbances during the **monitoring** stage and adapting the architecture during the **reconfiguration** stage [Khalil 2012]. This process requires a robust architecture or network to be effective [Li 2018]. Additionally, performance modeling plays a critical role in predicting stochastic system behavior during the early design stages and managing performance risks [Modaber 2024].

2.4.2 Monitoring

Disturbances can significantly impact the performance of complex systems that depend on intricate interactions and precise conditions. Even minor disruptions can compromise their functionality. To mitigate these risks and maintain system efficiency, robust monitoring mechanisms are essential.

Failure is a type of disturbance that necessitates the detection of failed CSs [Hyun 2023]. Prognostic and diagnostic methodologies are typically used in this detection, incorporating diagnostic systems characterized by quick detection, isolability, robustness, and error estimation [Venkatasubramanian 2005]. In SoS, this can be achieved by generating a set of fault indicators called Analytical Redundancy Relations (ARRs) [Staroswiecki 2001] that is the residual between the actual and the normal performance.

Another type of disturbance is the presence of extra CSs, leading to overcapacity. This requires evaluating the contribution of individual CSs to the overall SoS using specific metrics or attributes [Wang 2019].

Due to the unpredictable nature of disturbances, many studies have employed stochastic models to understand their effects in complex systems. For example, [Wang 2023] explores how power systems maintain synchronization despite random disturbances, which is particularly important with the integration of solar and wind power. In the context of single-system control, [Do 2016] examined how to control the trajectory of underactuated ships un-

der stochastic disturbances. The aim is to design controllers capable of handling both predictable and random disturbances, ensuring that ships follow a set path accurately. While most studies focus on single systems encountering random disturbances, there is a notable lack of research on the supervision of SoS under similar conditions.

Other models have employed stochastic approaches to represent disturbances in complex systems. For example, the Markovian model in [Oyewole 2020] effectively captures failures in cyber-physical power systems. Similarly, [Goldbeck 2019] utilized a stochastic asset failure model to assess failure propagation probabilities in urban infrastructure resilience. Binomial modeling is also a suitable approach for representing disturbances in independent components [Fan 2018], making it appropriate for stochastic SoS.

Monitoring is a crucial stage of supervision that addresses these disturbances. It involves the continuous observation of the system's performance to identify deviations from normal operation early. In multi-level frameworks, this stage is conducted in a bottom-up manner, starting with detecting disturbances at the physical level and propagating the information to higher-level organizations [Khalil 2012].

Monitoring distributed systems requires gathering and displaying information about process interactions to support performance evaluation. Due to the presence of multiple control points in distributed systems, traditional sequential monitoring techniques are often inadequate. A central control can be employed to monitor nondeterministic events more effectively [Joyce 1987].

Real-time monitoring, in particular, relies on models to define expected performance and detect deviations from it. Effective real-time monitoring combines advanced diagnostic methods, robust data collection, and sophisticated analytical tools to ensure systems continue functioning efficiently and reliably, even in the presence of disturbances. By rapidly identifying deviations from expected behavior, these models allow for timely interventions and adjustments, helping to maintain the integrity and functionality of complex systems despite unpredictable disruptions.

2.4.3 Reconfiguration

The dynamic nature of SoS, including the evolving behaviors of subsystems, changing goals and missions, and the disturbances that arise, requires a continuous response. **Adaptation** and **mission reallocation**, which involve reorganizing the SoS and reassigning functionalities to maintain normal performance, are essential for achieving global objectives [Chen 2023].

This process occurs during the reconfiguration phase, conducted in a top-down manner within multi-level supervision frameworks [Khalil 2012]. In

this phase, high-level organizations make decisions and reallocate functions to lower-level CSs. This hierarchical approach ensures that the system can efficiently adapt to changes and disturbances. High-level decision-makers evaluate the system's current status and decide on necessary adjustments, which are then implemented at the lower levels. By redistributing tasks and resources, the system can maintain optimal performance and continue to meet its global goals, even in a dynamic and changing environment.

Some reconfiguration methods, such as AI-based approaches, rely on data to reallocate missions and resources [Ibrar 2020], For example, [Gaiardelli 2024] used an AI-based framework for resource allocation and task offloading in a reconfigurable Internet of Vehicular Networks (IoV) to dynamically manage and distribute resources within a complex and dynamic network environment. However, these methods require large amounts of data, which may not always be available. However, these methods often require large datasets, which are not always available. Other methods focus on constraint satisfaction and optimization algorithms. For example, [Khalil 2012] applied the Constraint Satisfaction Problem (CSP) for reconfiguration, where constraints are assigned to high-level managerial CSs (represented as hyperedges in the HG) and evaluated using specific indicators. To solve the CSP, a backtracking algorithm was employed [Van Beek 2006], which has been widely used to find optimal solutions by exploring all possible options [Ayadi 2022, Li 2022, Mechqrane 2020]. The problem of these algorithms that are computationally expensive and poses challenges for large-scale systems.

Alternatively, capability-based reconfiguration [Antzoulatos 2015] is fast and less computationally intensive, which improves the scalability of SoS [Antzoulatos 2017]. In this context, capability refers to abstract descriptions of what a CS can accomplish [Scrimieri 2023]. Moreover, [Nelson 2019] emphasized the importance of mapping activities to system functions and assigning them to CSs, supporting missions reconfiguration and adaptation based on system capabilities where these missions must be evaluated to ensure that the global goal is achieved [Gao 2023].

2.5 Resilience of SoS

Recently, **resilience** has become an increasingly important in system design, as researchers prioritize it over robustness due to its ability to handle unforeseen challenges and disturbances [Tran 2015]. While robust systems are engineered to prevent failure through resistance, resilient designs emphasize adaptability and recovery. This shift in focus is essential because disruptions and shocks are inevitable, and systems cannot always avoid failures. Rather

than solely aiming to prevent breakdowns, resilience ensures that systems can absorb disturbances, adapt to changing conditions, and bounce back effectively. By designing with resilience in mind, systems can maintain stability and functionality even in the face of uncertainty, enhancing their long-term sustainability.

Resilience in SoS is crucial because it enables the system to adapt and recover from disruptions, ensuring continuous operation even in the face of failures. This adaptability is vital for maintaining the overall performance and reliability of complex, interconnected systems. Resilience and supervision are related concepts, particularly in these complex systems. Supervision primarily focuses on monitoring and reconfiguration for capacity management, ensuring that system resources are effectively allocated and disruptions are detected early. However, resilience goes beyond basic monitoring and capacity management. It emphasizes the system's ability to withstand, adapt and recover from disruption by employing different reconfiguration strategies [Dui 2023, Wang 2022].

2.5.1 Definition

Various definitions of resilience have been proposed in the literature. For example, [Tran 2015] offers the following definition of resilience:

Definition 4 (Resilience:) *The ability to maintain or recover desired capabilities in a timely manner when faced with a threat or disturbance, through well-informed design and adaptation.*

This definition is grounded in resilience characteristics such as the ability to provide the desired capability, the graceful and detectable degradation of function or capability when faced with disruption, the ability to maintain or recover degraded capabilities, and the ability to adapt to evolving threats and operating conditions—often through reconfiguration and replacement—while ensuring affordable and effective performance, as discussed in [Neches 2013, Madni 2009, Haimes 2009].

2.5.2 Methods

Researchers have focused on methods for designing resilient systems. In resilience engineering, there are two main approaches: **qualitative** and **quantitative** methods [Vugrin 2010]. Qualitative resilience methods emphasize descriptive, non-numerical analysis, assessing system properties, behaviors, and vulnerabilities using expert judgment, case studies, or scenario analysis.

These methods provide insights into how systems respond to disruptions and help identify resilience strategies.

In contrast, quantitative resilience methods involve numerical assessment and modeling, using measurable data, mathematical models, and statistical techniques to estimate system resilience, monitor performance, and predict recovery times. In SoS, where performance-based resilience is the primary focus, precise measurement of system performance and interactions is essential. Quantitative methods, which facilitate objective analysis, scalability, and predictive modeling, are particularly well-suited for managing the complexity of large, interconnected systems.

In SoS, reconfiguration methods focusing on **standby redundancy** rely on extra components as **backup systems** that replace failed ones. For instance, [Turnquist 2013] utilize redundant and backup systems, combined with linear programming and stochastic optimization, to design resilient SoS networks. However, these methods can be costly due to the need for additional components. Alternatively, adaptability-based methods utilize existing functional or **stand-in redundancy** to compensate for failed components by optimizing the performance of the SoS through reconfiguration. These methods usually rely on graphs or networks that can evolve to compensate for node failures [Chatterjee 2023]. For example, [Sun 2024] used deep reinforcement learning for resilient swarm UAV SoS, formulating it as an optimization problem. Similarly, [Uday 2015] employ stand-in (functional) redundancy, also framing it as a performance optimization problem. However, these network-based SoS formulations often address only a single layer and involve limited heterogeneity. Additionally, not all SoS possess sufficient functional redundancy to recover from failures, and at some point, it may be necessary to add external redundancies. This is why some researchers have suggested increasing redundancies in complex systems [Jackson 2013, Watson 2021].

2.5.3 Resilience metrics

SoS resilience quantification involves using specific metrics to measure the system's ability to **absorb**, **adapt** to, and **recover** from disruptions. These metrics often assess aspects such as robustness, recovery time, and performance degradation during failures. The need for such metrics arises from the increasing complexity and interdependence of modern SoS, where failures in one component can propagate across the entire system, leading to significant disruptions. Resilience quantification helps answer critical questions, such as which system—System A or System B—is more resilient under given conditions. By providing measurable data, these metrics enable a direct comparison of systems' resilience, helping in the design of more robust SoS, optimizing

resource allocation, and ensuring that the system can adapt and recover efficiently. This ultimately enhances long-term sustainability in dynamic and unpredictable environments.

For example, the authors in [Vugrin 2010] introduce quantitative metrics for assessing resilience by analyzing system performance over time. They evaluate resilience costs through the measurement of recovery effort. Figure 2.13 illustrates their conceptualization of system resilience and the associated recovery effort. The systemic impact SI is defined as an integral-based metric:

$$SI = \int_{t_0}^{t_f} [TSP(t) - SP(t)]\, dt, \tag{2.3}$$

where $TSP(t)$ represents the target system performance, $SP(t)$ represents the actual system performance, t_0 denotes the time of the disruption event, and t_f marks the time when recovery is complete. Similarly, the total recovery effort TRE is calculated as:

$$TRE = \int_{t_0}^{t_f} RE(t)\, dt, \tag{2.4}$$

where $RE(t)$ represents the recovery effort at time t. Additionally, the authors propose two supplementary metrics that combine a weighted sum of the systemic impact and the total recovery effort.

However, this approach faces challenges in differentiating between systems that may have similar integrated values but exhibit vastly different dynamic behaviors. Moreover, it does not distinguish between the absorption and recovery aspects of resilience.

On the other hand, [Balchanos 2012b] introduced a model called TIRESIAS, a resilience framework for complex and dynamic systems. This framework comprises a set of capability-based metrics designed to enable quantitative resilience comparisons between potential system designs [Balchanos 2012a, Balchanos 2014]. These metrics rely on measurements of system performance (see Fig. 2.14) and are used to assess and compare the absorptive and restorative capacities of different designs.

The ability of a system to absorb a disturbance is captured by the average degradation rate, ADR, and the time-averaged performance degradation, tMC_{deg}, which are calculated as follows:

$$ADR = \frac{MC_0 - MC_{\min}}{t_{\min} - t_0}, \tag{2.5}$$

$$tMC_{\text{deg}} = \frac{1}{t_{\min} - t_0} \int_{t_0}^{t_{\min}} [MC_0 - MC(t)]\, dt, \tag{2.6}$$

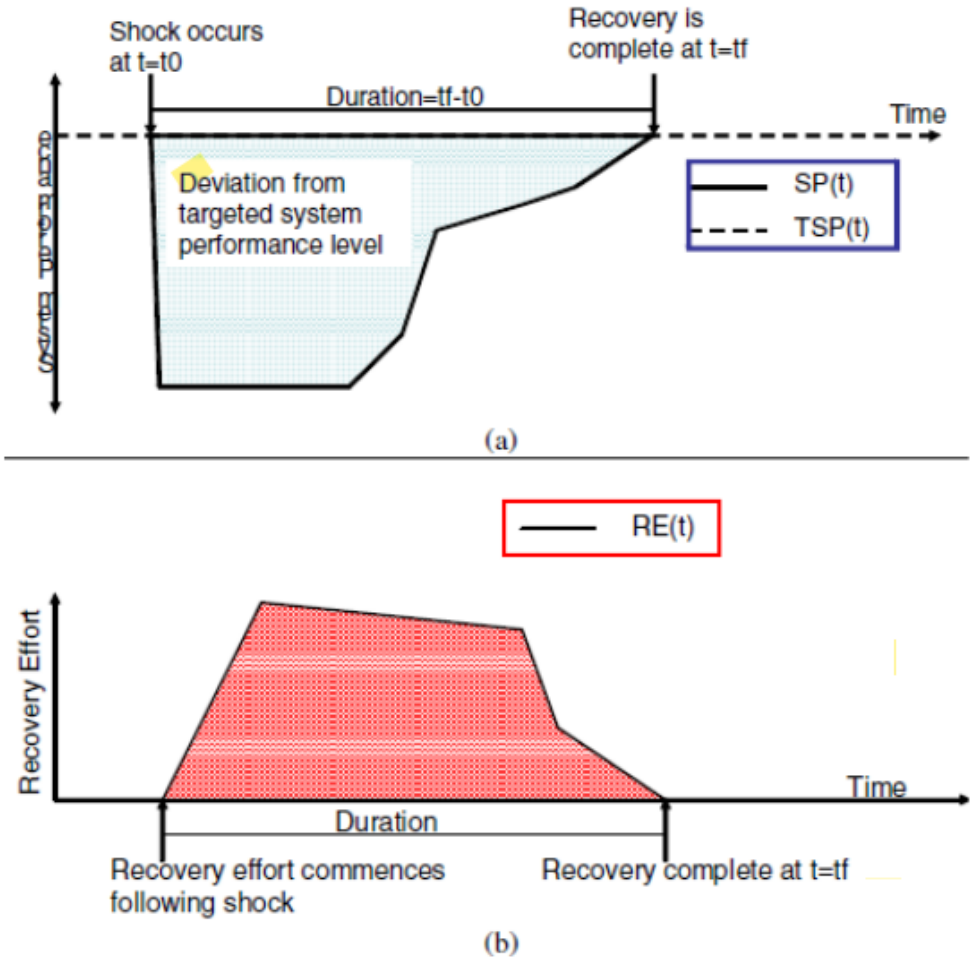


Figure 2.13: (a) System performance and (b) recovery effort [Vugrin 2010]

where MC_0 represents the original desired mission capability or performance level, $MC_{\min}$ is the minimum mission performance level, $t_{\min}$ is the time at which $MC_{\min}$ is reached, and t_0 marks the time when the disturbance occurs.

The ability of a system to recover capabilities lost due to a disturbance is measured by the average recovery rate, ARR, and the time-averaged performance recovery, tMC_{rec}, given by:

$$ARR = \frac{MC_{\text{SS}} - MC_{\min}}{t_{\text{SS}} - t_{\min}}, \tag{2.7}$$

$$tMC_{\text{rec}} = \frac{1}{t_{\text{SS}} - t_1} \int_{t_1}^{t_{\text{SS}}} [MC_{\text{SS}} - MC(t)]\, dt, \tag{2.8}$$

where MC_{SS} is the steady-state performance level, and t_1 marks the start of the system's recovery process. While this approach accounts for different resilience capabilities, it focuses on a single disruption and does not take into account the possibility of multiple disruptions.

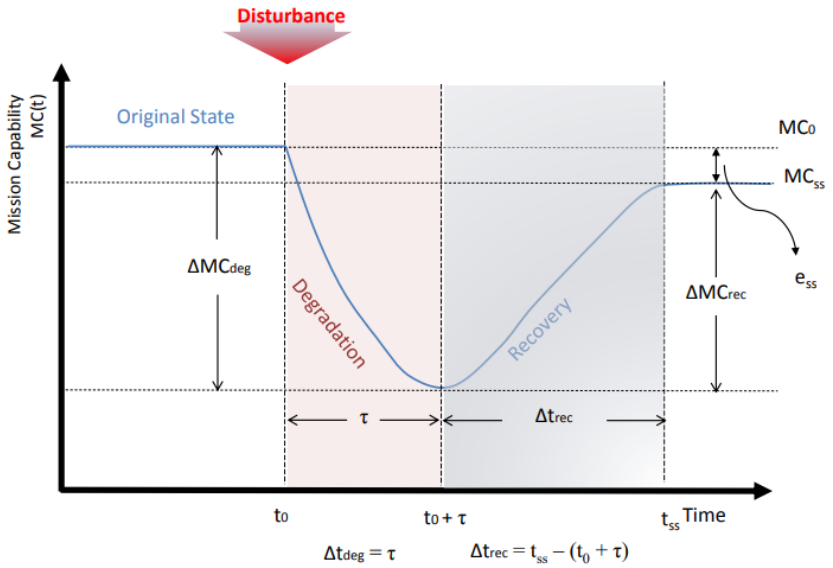


Figure 2.14: System performance used in [Balchanos 2014]

To overcome these challenges, [Tran 2016] proposed a new resilience metric calculated from SoS performance data, $y(t)$. This metric uses specific data points identified from the performance data, notionally shown in Fig. 2.15. Using the SoS performance data, the resilience metric R, is calculated as:

$$R = \rho\sigma \left[\delta + \zeta + 1 - \tau^{(\rho-\delta)}\right] \tag{2.9}$$

where $0 \leq R \leq \lambda$, and the terms in this equation are referred to as resilience factors. Each resilience factor captures an important aspect of a resilient system. The resilience factors are calculated as:

$$\sigma = \text{total performance factor} = \frac{\sum_{t_0}^{t_{\text{final}}} y(t)}{y_D(t_{\text{final}} - t_0)} \tag{2.10}$$

$$\delta = \text{absorption factor} = \frac{y_{\min}}{y_D} \tag{2.11}$$

$$\rho = \text{recovery factor} = \frac{y_R}{y_D} \tag{2.12}$$

$$\tau = \text{time factor} = \frac{t_{\text{SS}} - t_0}{t_{\text{final}} - t_0} \tag{2.13}$$

$$\zeta = \text{volatility factor} \tag{2.14}$$

where y_D is the desired system performance level, $y_{\min}$ is the minimum system performance level, y_R is the recovered performance level, t_0 is the time of disturbance, $t_{\min}$ is the time at which the system reaches the minimum performance level, and t_{SS} is the time when the system reaches steady-state performance.

Considering multiple failures, a total resilience metric, R_{total}, is calculated from individual R values as:

$$R_{\text{total}} = \frac{\sum_{i=1}^{N_T} w_i R_i}{\sum_{i=1}^{N_T} w_i} \tag{2.15}$$

where N_T is the number of threat events (i.e., CS failures) and the weights w_i are coefficients of an exponentially weighted moving average, defined as:

$$w_i = (1 - \alpha)^{N_T - i} \tag{2.16}$$

with a smoothing factor $\alpha = 0.06$.

This metric accounts for different resilience aspects considering multiple failures. However, given that most reconfiguration methods are based on redundancy (both stand-in and stand-by), adding redundancy factors to this metric can provide more insights into maintaining functionality during failures and recovering more effectively.

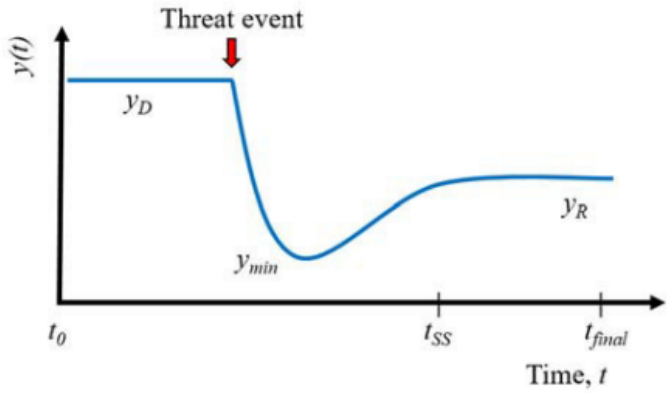


Figure 2.15: The performance for the SoS used in [Tran 2016], denoted as $y(t)$, represents the response to a single threat event occurring within the time interval from t_0 to t_{final}. The minimum performance level during this period is y_{min}. The system reaches steady-state performance at time t_{SS}, with a recovered performance level of y_R. The desired performance level is y_D.

2.6 Conclusion

The current literature highlights the growing interest of researchers in the SoS domain, particularly when addressing large-scale and complex systems. Although a unified definition of SoS is still lacking, certain characteristics and properties distinguish SoS from traditional complex systems. Most studies have focused on modeling the structure and organization of SoS, including the missions, functions, and interactions of their CSs. While recent research has shifted towards data-driven approaches, these methods require large datasets, which are not always available, especially in early-stage design. Graphical models have demonstrated efficiency in SoS modeling; however, they continue to face limitations in representing complex interactions, heterogeneity, scalability, decision-making processes, and stochastic components.

To address these challenges, there is a need for a comprehensive framework that can overcome these limitations. Such a framework should support the supervision of complex systems, ensuring they maintain normal operations. In terms of resilience, the literature shows that most recovery methods focus on either adaptation and stand-in redundancy or backups and stand-by redundancy. There is a growing emphasis on the importance of developing metrics to quantify resilience and further improve the robustness of SoS.

Chapter 3

Contribution to organisational modeling of stochastic SoS

Contents

3.1 Introduction

Modeling a SoS refers to the abstract definition of the structure, behavior, and interactions of its CSs. Compared with conventional single-system modeling, modeling SoS presents several challenges due to the inherent complexity of the relationships among multiple CSs [Maier 1998].

This chapter focuses on modeling the structure of the SoS in terms of mission and function allocation, along with their associated performance. It

also considers the existence of CSs within the SoS that may be affected by stochastic processes, as well as the capabilities of each CS. By proposing our framework, the Multi-level Stochastic Hypergraph (MLSHG) model, it will be demonstrated how heterogeneous CSs can be modeled while accounting for stochasticity. Inspiring from the formalism of [Soyez 2015], attributes will be incorporated for each CS to describe the dynamic organizational properties of these subsystems.

Additionally, the interactions will be discussed using edge-dependent weights [Chitra 2019], which encompass various attributes. These attributes include the existence of relationships, related functions, and both stochastic and deterministic dynamic interactions. The stochastic and deterministic dynamic interaction attributes are able to describe both the predictable and unpredictable interactions among CSs. The stochastic multi-way relationships among CSs can be well described with the help of the edge-dependent weight function.

The proposed framework will be verified against Maier's properties, and its application to a detailed case study will be explained. This verification demonstrates the framework's applicability and effectiveness in capturing the complexity and dynamics of an SoS.

3.2 Stochatic system of systems

Researchers have extensively studied SoS in the presence of stochastic behavior across various domains to incorporate uncertainty and model failures, and disturbances. Studies such as [Khabarov 2008, Doulgeris 2012, Panteli 2015, Mo 2016, Dhulipala 2021] have demonstrated the importance of considering stochastic elements to capture the real-world complexity and variability within SoS. These investigations primarily focus on understanding how randomness and unpredictability affect the performance and reliability of SoS. However, despite these efforts, there remains a lack of an explicit and comprehensive definition of what constitutes a Stochastic System of Systems (SSoS). To address this gap, a clear and precise definition is proposed, aiming to standardize the terminology and framework for future research in this critical area.

Definition 5 (Stochastic System of Systems (SSoS)) *A Stochastic System of Systems is a large-scale integrated system that is heterogeneous and independently operable on its own but networked together for a common goal, with inherent stochastic properties. These stochastic properties include random variations, uncertainties, and probabilistic behaviors that influence the interactions, performance, and the total number of Component Systems.*

In the following section, a novel framework for modeling SSoS will be proposed, along with an explanation of the methodology and the steps involved.

3.3 Multi-level Stochastic hypergraph modeling

Multi-level frameworks are crucial for handling the complexity of complex systems [Mostafavi 2011]. In a SoS, this means representing and analyzing the interactions and dependencies among multiple CSs that work together to achieve overall goals. By using multi-level models, researchers and engineers can study the organization, function distribution, and dynamic interactions within an SoS [Soyez 2015, Khalil 2012, Kumar 2017]. This approach helps identify potential weaknesses, optimize resource use, maintain normal operation and improve system resilience against both predictable and unpredictable disruptions.

To this end, the SoS is divided into a multi-level organization to better manage its complexity and interactions. At Level 0 are the Physical Component Systems (PCSs), which serve as the fundamental building blocks of the SoS. These consist of various physical systems that may be heterogeneous in nature, meaning they can differ significantly in terms of functionality, design, and operation. PCSs are responsible for executing the primary tasks and operations that directly contribute to the SoS's overall objectives. Examples of PCSs include mechatronic systems, social systems, and biological systems.

At levels higher than 0, Managerial Component Systems (MCSs) are introduced. These systems represent the organizational and managerial structures that oversee and coordinate the activities of the PCSs. MCSs have authority over their respective subsystems and are responsible for strategic decision-making, resource allocation, and ensuring that the various PCSs work together harmoniously to achieve the SoS's goals. The MCSs handle high-level planning, policy-making, and operational adjustments needed to maintain the efficiency and effectiveness of the SoS.

To represent this decomposition of the SoS, let $H = (V, E, W)$ denote the weighted hypergraph, where V is the vertex set, E is the hyperedge set, and W is the edge-dependent weight set.

The vertex set $V = \{v_1, v_2, \ldots, v_n\}$ represents the vertices in the hypergraph, with each vertex v_i for $i = 1, 2, \ldots, n$ corresponding to PCSs within the SoS. The hyperedge set $E = \{e_1, e_2, \ldots, e_m\}$ consists of hyperedges corresponding mainly to MCSs, where each hyperedge $e_j = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ represents a relationship or interaction among a subset of vertices. The following sections explain how each type of CS is represented through vertices and hyperedges by defining them and assigning dynamic properties and at-

tributes.

The weight set W is defined as edge-dependent weights to describe the interactions within the SoS. These weights capture not only the information exchange between CSs but also represent multiple properties of the interactions, such as the stochastic existence among the CSs, providing a detailed representation of the dynamics within the SoS.

Figure 3.1 depicts the graphical representation of the SoS using a hypergraph (set hypergraph and its hierarchical representations). This figure shows the different types of CSs, their indexes, levels $CS_{\text{index,level}}$, and the interactions. This formalism is proposed by [Khalil 2012], who also demonstrated different types of hypergraph representations. The interconnections between the hyperedges differentiate the nested hypergraph from a hyperset [Chemero 2008]. In this thesis, the focus is only on nested hypergraphs where there are interconnections between CSs. This representation presents a snapshot of the evolving SoS at a specific time and requires additional properties and attributes to describe the dynamics of a stochastic SoS.

By combining multi-level modeling with hypergraph theory and incorporating the organizational properties that describe a SSoS, the MLSHG model has been developed. This framework leverages the strengths of both multi-level modeling and hypergraph representations to effectively capture the complexity and dynamics of SSoS. The methodology for the MLSHG model is illustrated in Fig. 3.2 and is detailed in the following sections.

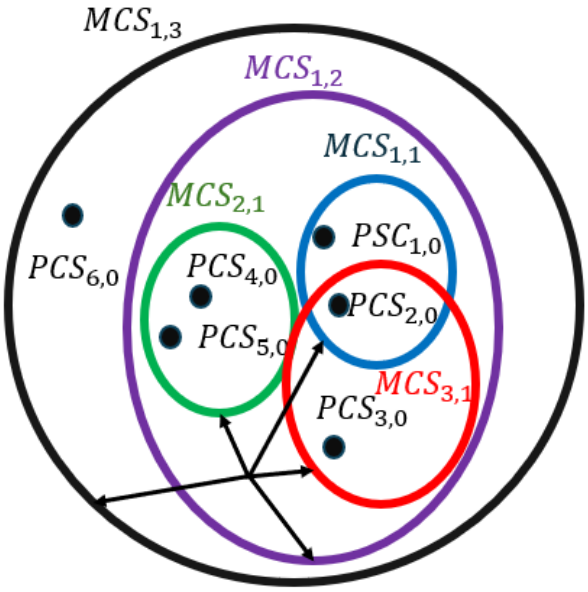
3.3.1 Individual CSs

Starting by defining individual CSs, these are subsystems within the SoS that vary based on their level and properties. These properties include organizational aspects such as missions, performance, capabilities, and whether they satisfy their desired missions. To capture these distinctions, the types of CSs in a multilevel SoS are defined below.

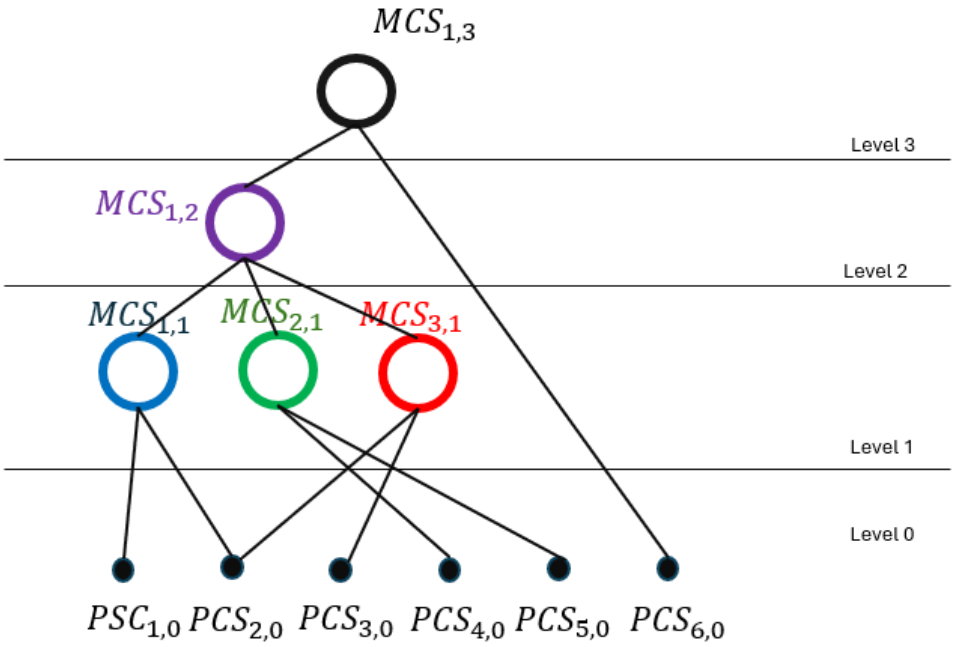
Definition 6 (Physical Component System (PCS)) *At physical level* 0*, a PCS can only be viewed as a vertex in hypergraph H. It represent social, mechatronic, or biological entity, to better describe heterogeneity in terms of behavior, the n-th CS at physical level* 0 *is denoted as $CS^j_{n,0}$, where j represents whether the CS is stochastic (SCS) (see Definition 8) or deterministic (DCS).*

$$CS^j_{n,0} = \begin{cases} SCS_{n,0}, if\ \ j = 1, \\ DCS_{n,0}, if\ \ j = 0. \end{cases} \tag{3.1}$$

To interpret the dynamics of a PCS, several attributes are assigned to represent its organizational properties as shown in the Fig. 3.3. A PCS can



(a)



(b)

Figure 3.1: Graphical representation of SoS using hypergraph (a) Set representation (b) Hierarchical representation

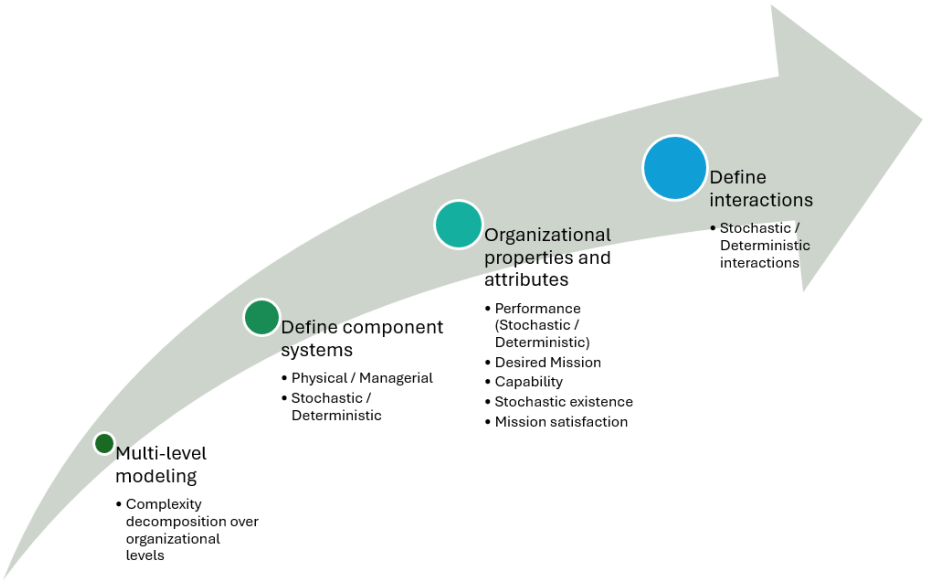


Figure 3.2: MLSHG methodology

be depicted as:

$$CS_{n,0}^{j} = \left\langle D_{n,0}^{t}, P_{n,0}^{t}, Q_{n,0}, R_{n,0}^{t}, A_{n,0}^{t}, C_{n,0}^{t} \right\rangle, \tag{3.2}$$

The attributes related to $CS_{n,0}^{j}$ are defined as follows respectively:

$D_{n,0}^{t}$: $D_{n,0}^{t} = \{d_1^t, ..., d_k^t\}$. A set of time-dependent attributes that represent the current deterministic performance of specific functions of $CS_{n,0}^{j}$ where k is the number of global functions, for SCS $D_{n,0}^{t} = \emptyset$. If $CS_{n,0}^{j}$ is DCS and does not perform Function 1 at time t, then $d_1^t = 0$.

$P_{n,0}^{t}$: $P_{n,0}^{t} = \{p_1^t, ..., p_k^t\}$. A set of time-dependent attributes that represent the current stochastic performance of specific functions of $CS_{n,0}^{j}$ where k is the number of global functions, for DCS $P_{n,0}^{t} = \emptyset$. If $CS_{n,0}^{j}$ is SCS and does not perform Function 1 at time t, then $p_1^t = 0$.

$Q_{n,0}$: $Q_{n,0} = \{q_1, ..., q_k\}$, A set of capability functions of $CS_{n,0}^{j}$ in the SoS that represents the capacity functionalities that $CS_{n,0}^{j}$ can realize, where q_1 is the quantification of function 1, assuming that all functions are quantified.

$R_{n,0}^{t}$: $R_{n,0}^{t} = \{q_1^t, ..., q_k^t\}$, A set comprising the quantification of desired functions for $CS_{n,0}^{j}$ at time t is assigned from a higher-level CS, where these functions represent the mission $M_{n,0}^{t} = \{function_1, ..., function_k\}$ of $CS_{n,0}^{j}$.

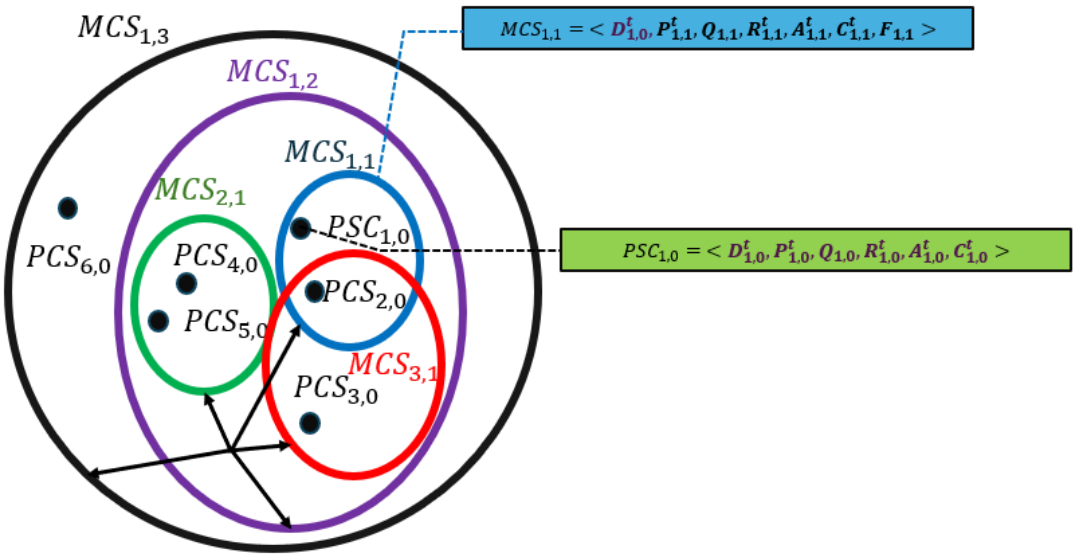


Figure 3.3: Hypergraph with assigned attributes

If $CS^j_{n,0}$ does not been assigned Function 1 at time t, then $q^t_1 = 0$.

$A^t_{n,0}$: Binary stochastic attribute that describes whether $CS^j_{n,0}$ is composed of the SoS. For an arbitrary fixed instant t_f, $A^{t_f}_{n,0} = 1$ if $CS^j_{n,0}$ is present in SoS at time t_f. Otherwise, $A^{t_f}_{n,0} = 0$.

$C^t_{n,0}$: A satisfaction attribute that indicates whether the constraint associated with $CS^j_{n,0}$ is met. This constraint is defined by the achievement of the desired mission $R^t_{n,0}$ based on the current performance $P^t_{n,0}$, and is formulated as follows:

$$C^t_{n,0} = \begin{cases} 1, & \text{if } |r(t)| \leq \lambda, \\ 0, & \text{if } |r(t)| \geq \lambda, \end{cases} \tag{3.3}$$

where $r(t)$ represents the residual between the desired functions of the mission $R^t_{n,0}$ and the current performance $P^t_{n,0}$. This formulation is similar CSP proposed by [Khalil 2012]. It is important to note that, since we assume performance disruptions only occur for SCS and not for DCS, this attribute is considered satisfied ($C^t_{n,0} = 1$) always for DCS.

Note that the assumption is that all functions can be quantified and the performance, capability, desired mission, and mission satisfaction attributes have the same length for all PCSs. Even if some of them do not perform specific functions, the value associated with these functions is set to zero. This

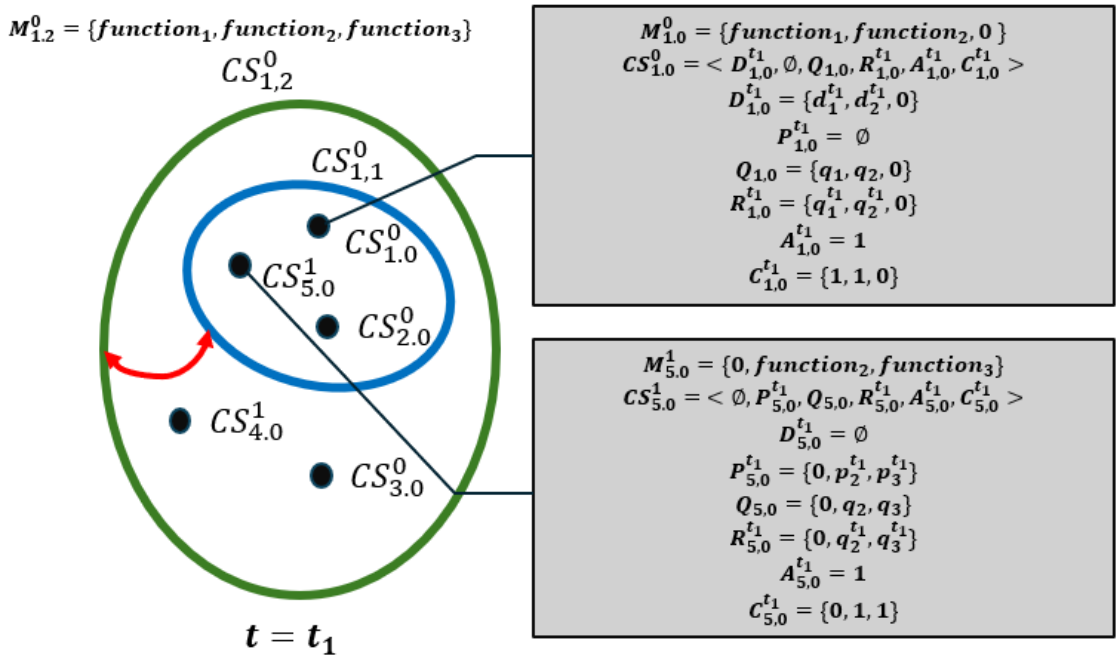


Figure 3.4: Example of attributes assigned to PCSs at t_1

formalism is considered in order to facilitate the development of algorithms for function decomposition and mission allocation later.

To illustrate this methodology, the example in Fig. 3.4 is used to compare the attributes assigned to two PCSs at t_1. The global mission is composed of three functions, as shown in the figure. For $CS^0_{1,0}$, which is a DCS ($j = 0$), it performs two functions (Function 1 and Function 2), where the quantified values of these functions are represented by $R^{t_1}_{1,0}$. The deterministic performance and capability are $D^{t_1}_{1,0}$ and $Q_{1,0}$, respectively. It is part of the SoS at t_1 ($A^{t_1}_{1,0} = 1$), and its desired functions (1 and 2) are satisfied ($C^{t_1}_{1,0} = \{1, 1, 0\}$).

For $CS^1_{5,0}$, which is an SCS ($j = 1$), it performs two functions (Function 2 and Function 3), where the quantified values of these functions are represented by $R^{t_1}_{5,0}$. The stochastic performance and capability are $P^{t_1}_{5,0}$ and $Q_{5,0}$, respectively. It is part of the SoS at t_1 ($A^{t_1}_{5,0} = 1$), and its desired functions (2 and 3) are satisfied ($C^{t_1}_{5,0} = \{0, 1, 1\}$).

Definition 7 (Managerial Component Systems (MCSs)) *A high-level organizational component for an arbitrary CS at managerial level l, where $l \geq 1$, can be both a vertex and/or a hyperedge in the HG H. Additionally, this CS can be a stochastic or deterministic hyperedge, represented by the index j, similar to that of the vertex.*

$$CS^{j}_{n,l} = \left\langle D^{t}_{n,l}, P^{t}_{n,l}, Q_{n,l}, R^{t}_{n,l}, A^{t}_{n,l}, C^{t}_{n,l}, F_{n,l} \right\rangle, \tag{3.4}$$

Since an MCS has authority over subordinate CSs, it has an additional attribute that represents its managerial properties, denoted by the $F_{n,l}$ attribute. This attribute addresses the question of how to assign functions and missions to subordinate CSs based on its own mission and functions and is defined as follows:
$F_{n,l} = \{f_{n,l,n',j}; (1 \leq j \leq l-1)(1 \leq n' \leq a_{l-j})\}$ where a_{l-j} is the number of CSs in level $l-j$, and $f_{n,l,n',j}$ is a mapping function that assigns the functions of the $CS^{j}_{n,l}$ to the subordinate system $(CS_{n',l-j})$, $f_{n,l,n',j} = 0$ if the n'-th CS at level l-j does not belong to the set of subordinate systems governed by $CS^{j}_{n,l}$.

Definition 8 (Stochastic Component System (SCS)) *A Physical Component System (PCS) at level* 0*, denoted as* $CS^{j}_{n,0}$*, or a Managerial Component System (MCS) at level* l*, denoted as* $CS^{j}_{n,l}$*, is considered stochastic if it has a stochastic performance attribute* $P^{t}_{n,l}$ *influenced by its stochastic behavior, which, in turn, affects its existence* $A^{t}_{n,l}$*.*

A SCS can be modeled using a stochastic process and is represented as $CS^{j}_{n,l} = SCS_{n,l}$ if $j = 1$. Otherwise, $CS^{j}_{n,l}$ is considered deterministic and is represented as $CS^{j}_{n,l} = DCS_{n,l}$ if $j = 0$.

3.3.2 Hierarchical Structure

A vertex $CS^{j}_{n,l}$ belongs to hyperedge $CS^{j'}_{n',l+k}$, i.e.,

$$CS^{j}_{n,l} \in CS^{j'}_{n',l+k}, \tag{3.5}$$

implies that the n-th CS in level l is supervised by the n'-th CS at level $l+k$. An example is shown in Fig.3.5. Because $CS^{0}_{1,0}, CS^{0}_{2,0}, CS^{1}_{5,0}$ are supervised by $CS^{0}_{1,1}$ in managerial level 1, $CS^{0}_{1,1}$ can be viewed as a hyperedge composed of vertices $CS^{0}_{1,0}, CS^{0}_{2,0}, CS^{1}_{5,0}$, that is,

$$CS^{0}_{1,1} = \{CS^{0}_{1,0}, CS^{0}_{2,0}, CS^{1}_{5,0}\}. \tag{3.6}$$

At the same time, $CS^{0}_{1,1}$ together with $CS^{0}_{3,0}$, $CS^{1}_{4,0}$ are supervised by $CS^{0}_{1,2}$ at a higher managerial level 2. In this occasion, $CS^{0}_{1,1}$ can be viewed as a vertex and $CS^{0}_{1,2}$ is a hyperedge, which can be expressed as

$$\begin{aligned} CS^{0}_{1,2} =& \{CS^{0}_{1,1}, CS^{0}_{3,0}, CS^{1}_{4,0}\} \\ =& \{\{CS^{0}_{1,0}, CS^{0}_{2,0}, CS^{1}_{5,0}\}, CS^{0}_{3,0}, CS^{1}_{4,0}\}. \end{aligned} \tag{3.7}$$

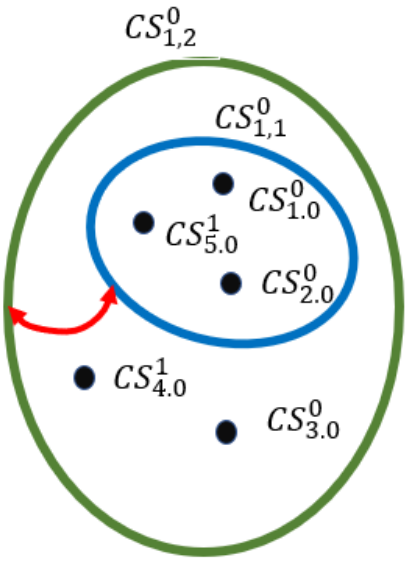


Figure 3.5: Hypergraph example

It can be observed that $CS^0_{1,2}$ is also a hyperedge of elements $CS^0_{3,0}, CS^1_{4,0}$ and the set $CS^0_{1,1} = \{CS^0_{1,0}, CS^0_{2,0}, CS^1_{5,0}\}$. In this way, the organizational relationships among multi-level CSs can be described flexibly. Note that this HG example is a snapshot at a specific moment, and the organization and structure of the HG may vary over time.

3.3.3 Interactions

The weight set W in the HG $H = (V, E, W)$ represents the edge-dependent weights assigned to vertices within the hyperedges. The weight of a vertex in a hyperedge is defined as:

$$W = \{W(v, e) \mid \forall v \in V, \forall e \in E\}.$$

In the MLSHG framework, the weight of a vertex $CS^j_{n,l}$ within a hyperedge $CS^{j'}_{n',l+k}$ is defined by the following attributes to describe the interactions among multiple CSs with both stochastic and deterministic characteristics:

$$\begin{aligned} W(CS^j_{n,l}, CS^{j'}_{n',l+k}) =< & R^t_{n,l,n',l+k}, DI^t_{n,l,n',l+k}, \\ & SI^t_{n,l,n',l+k}, A^t_{n,l,n',l+k} > . \end{aligned} \tag{3.8}$$

The attributes related to $CS^j_{n,l}$ are defined as follows respectively:

$R_{n,l,n',l+k}$: The quantification of sub-mission set of $CS^j_{n,l}$ consists of functions assigned by $CS^{j'}_{n',l+k}$. The function q^t_1 in $R^t_{n,l,n',l+k}$ depicted by

$(q_1^t|R^t_{n,l,n',l+k})$ is obtained by :

$$q_1^t|R^t_{n,l,n',l+k} = f_{n',l+k,n,k}(q_1^t|R^t_{n',l+k}), \tag{3.9}$$

which implies the functions of $CS^j_{n,l}$ that are monitored by $CS^{j'}_{n',l+k}$ is a part of its own functions performed in the entire SoS.

$DI^t_{n,l,n',l+k}$: Deterministic dynamic interaction vector of $CS^j_{n,l}$ with $CS^{j'}_{n',l+k}$ and other CSs supervised by $CS^{j'}_{n',l+k}$. Thus, the dimensions of $DI^t_{n,l,n',l+k}$ are equal to the number of CSs supervised by $CS^{j'}_{n',l+k}$, i.e., the cardinality of hyperedge $|CS^{j'}_{n',l+k}|$. That is

$$dim(DI^t_{n,l,n',l+k}) = |CS^{j'}_{n',l+k}|. \tag{3.10}$$

For example, refering to hyperedge (3.6) in Fig.3.5, the corresponding $DI^t_{1,0,1,1}$ can be described as

$$\begin{aligned} DI^t_{1,0,1,1} = [&DI_{CS^0_{1,0},CS^0_{1,1}}(t), \\ &DI^{CS^0_{1,1}}_{CS^0_{1,0},CS^0_{2,0}}(t), DI^{CS^0_{1,1}}_{CS^0_{1,0},CS^1_{5,0}}(t)], \end{aligned} \tag{3.11}$$

where $DI^{CS^0_{1,1}}_{CS^0_{1,0},CS^0_{2,0}}(t)$ and $DI^{CS^0_{1,1}}_{CS^0_{1,0},CS^1_{5,0}}(t)$ represent the deterministic dynamic interactions between $CS^0_{1,0}$ and $CS^0_{2,0}$, and $CS^0_{1,0}$ and $CS^1_{5,0}$, respectively, under the supervision of $CS^0_{1,1}$. Additionally, $DI_{CS^0_{1,0},CS^0_{1,1}}(t)$ represents the deterministic dynamic interaction between $CS^0_{1,0}$ and $CS^0_{1,1}$, since $CS^1_{5,0}$ is a SCS it implies that $DI^{CS^0_{1,1}}_{CS^0_{1,0},CS^1_{5,0}}(t) = 0$.

$SI_{n,l,n',l+k}$: Stochastic interaction vector of $CS^j_{n,l}$ with $CS^{j'}_{n',l+k}$ and other CSs supervised by $CS^{j'}_{n',l+k}$. Similarly, the dimensions of $SI_{n,l,n',l+k}$ are equal to the number of CSs supervised by $CS^{j'}_{n',l+k}$, i.e.,

$$dim(SI^t_{n,l,n',l+k}) = |CS^{j'}_{n',l+k}|. \tag{3.12}$$

Similarly, refering to hyperedge (3.7) in Fig.3.5, the corresponding $SI^t_{1,1,1,2}$ can be described as

$$\begin{aligned} SI^t_{1,1,1,2} = [&SI_{CS^0_{1,1},CS^0_{1,2}}(t), \\ &SI^{CS^0_{1,2}}_{CS^0_{1,1},CS^0_{3,0}}(t), SI^{CS^0_{1,2}}_{CS^0_{1,1},CS^1_{4,0}}(t)], \end{aligned} \tag{3.13}$$

where $SI^{CS^0_{1,2}}_{CS^0_{1,1},CS^0_{3,0}}(t)$ and $SI^{CS^0_{1,2}}_{CS^0_{1,1},CS^1_{4,0}}(t)$ represent the stochastic interactions between $CS^0_{1,1}$ and $CS^0_{3,0}$, and $CS^0_{1,1}$ and $CS^1_{4,0}$, respectively, under the supervision of $CS^0_{1,2}$. Additionally, $SI_{CS^0_{1,1},CS^0_{1,2}}(t)$ represents the stochastic interaction between $CS^0_{1,1}$ and $CS^0_{1,2}$. Since $CS^0_{3,0}$ and $CS^0_{1,2}$ are DCSs it implies that $SI_{CS^0_{1,1},CS^0_{1,2}}(t) = SI^{CS^0_{1,2}}_{CS^0_{1,1},CS^0_{3,0}} = 0$.

$A^t_{n,l,n',l+k}$: Binary stochastic attribute that describes whether $CS^j_{n,l}$ is under the supervision of $CS^{j'}_{n',l+k}$ at time t. For an arbitrary fixed instant t_f, $CS^j_{n,l}$ is supervised by $CS^{j'}_{n',l+k}$ only if both $CS^j_{n,l}$ and $CS^{j'}_{n',l+k}$ present in the SoS, which suggests

$$\forall t_f, A^{t_f}_{n,l,n',l+k} = 1 \text{ only if } A^{t_f}_{n,l} = 1 \& A^{t_f}_{n',l+k} = 1. \tag{3.14}$$

The relationship between $A^t_{n,l}$ of $CS^j_{n,l}$ in the entire SoS and $A^t_{n,l,n',l+k}$ of $CS^j_{n,l}$ under the supervision of $CS^{j'}_{n',l+k}$ can be expressed as follows:

$$\begin{aligned} & P(A^{t_f}_{n,l,n',l+k} = 1), \forall t_f, \\ =& P(A^{t_f}_{n,l,n',l+k} = 1 | A^{t_f}_{n,l} = 1 \wedge A^{t_f}_{n',l+k} = 1) \\ & * P(A^{t_f}_{n,l} = 1 \wedge A^{t_f}_{n',l+k} = 1), \end{aligned} \tag{3.15}$$

where $P(A^{t_f}_{n,l,n',l+k} = 1 | A^{t_f}_{n,l} = 1 \wedge A^{t_f}_{n',l+k} = 1)$ represents the conditional probability of $A^{t_f}_{n,l,n',l+k} = 1$ when $A^{t_f}_{n,l} = 1$ and $A^{t_f}_{n',l+k} = 1$. In addition, considering the independence between $A^{t_f}_{n,l}$ and $A^{t_f}_{n',l+k}$ for an arbitrary t_f, it follows that

$$\begin{aligned} & P(A^{t_f}_{n,l} = 1 \wedge A^{t_f}_{n',l+k} = 1) \\ & = P(A^{t_f}_{n,l} = 1) * P(A^{t_f}_{n',l+k} = 1) \end{aligned} \tag{3.16}$$

Substituting (3.16) into (3.15), one has

$$\begin{aligned} & P(A^{t_f}_{n,l,n',l+k} = 1), \forall t_f, \\ =& P(A^{t_f}_{n,l,n',l+k} = 1 | A^{t_f}_{n,l} = 1 \wedge A^{t_f}_{n',l+k} = 1) \\ & * P(A^{t_f}_{n,l} = 1) * P(A^{t_f}_{n',l+k} = 1). \end{aligned} \tag{3.17}$$

To better interpret the stochastic organization of SoS related to the attributes $A^t_{n,l}$, an illustrative example is provided in Fig. 3.6. The figure depicts SoS hypergraphs at three distinct instants: t_1, t_2, and t_3. At instant t_1, all CSs are present and colored black. This indicates that the stochastic existence attributes satisfy $A^{t_1}_{n,0} = 1, n = 1, 2, ..., 5$ and $A^{t_1}_{n,1} = 1, n = 1, 2$. Nevertheless, at time t_2, $CS^0_{1,0}$ and $CS^0_{2,0}$ are removed from the SoS. This is why vertices $CS^0_{1,0}$ and $CS^0_{2,0}$ are colored gray in the figure. This implies the stochastic existence attributes satisfy $A^{t_2}_{1,0} = 0$ and $A^{t_2}_{2,0} = 0$. Moreover, at instant t_3, the Component Systems $CS^1_{5,0}$ and $CS^0_{1,1}$ of higher level are further removed from the SoS (vertices $CS^0_{1,0}$ and hyperedge $CS^0_{1,1}$ are colored gray), which reveals that the stochastic existence attributes satisfy $A^{t_3}_{n,0} = 0, n = 1, 2, 5$ and $A^{t_3}_{1,1} = 0$.

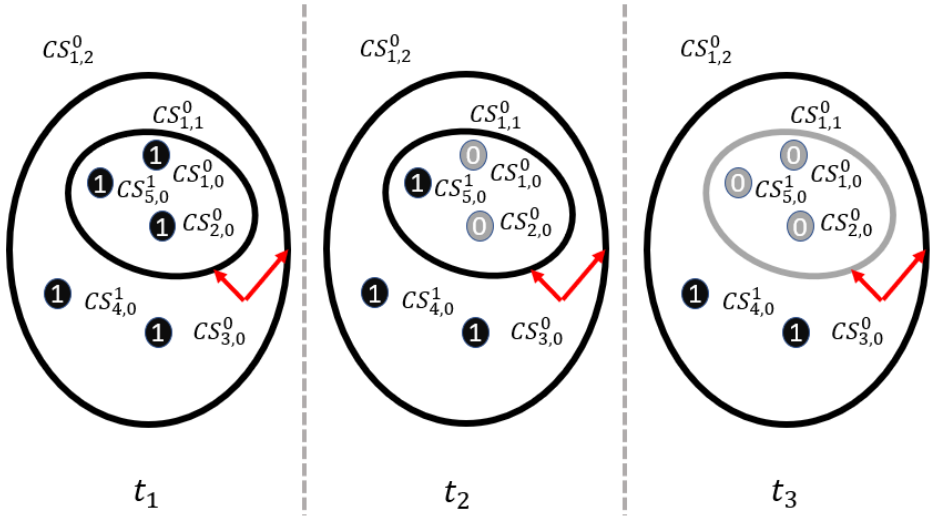


Figure 3.6: Hypergraph dynamics

3.3.4 Verifying Properties of System-of-Systems

As previously stated, an SoS exists when Mair's characteristics are respected [Sage 2001]. From the MLSHG perspective, this can be expressed as follows:

- Managerial independence: CSs within an SoS are not only capable of operating independently but are also managed independently, each CS has its own management structure, decision-making processes, and different mission or priorities that are separate from the SoS as a whole.

  In the MLSHG model,$CS^{j'}_{n',l'}$ and $CS^{j''}_{n'',l''}$ of a system $CS^{j}_{n,l}$ are managerially independent when their missions are different:

$$\forall j, j', j'' \in \{0,1\}, \forall CS^{j'}_{n',l'}, CS^{j''}_{n'',l''} \in CS^{j}_{n,l} : \quad R^t_{n',l'} \neq R^t_{n'',l''}. \tag{3.18}$$

  For example in Fig.3.5, $CS^0_{1,0}$ and $CS^1_{5,0}$ in $CS^0_{1,1}$ are managerial independent when:

$$R^t_{1,0} \neq R^t_{5,0} \tag{3.19}$$

- Operational independence: For a system to be considered a SoS, each of its CSs must be operationally independent. This means that if the SoS were broken down into its individual components, each one would still be capable of performing its functions and fulfilling its intended purposes independently.

  This is only applicable if it has its own resources that enable it to perform autonomously. In the MLSHG model, $CS^{j'}_{n',l'}$ and $CS^{j''}_{n'',l''}$ of a system

$CS^{j}_{n,l}$ are operationally independent if each possesses its own resources s.

$$\begin{aligned} &\forall j, j', j'' \in \mathbb{N}, \forall CS^{j'}_{n',l'}, CS^{j''}_{n'',l''} \in CS^{j}_{n,l} : \\ &s_{n',l'} \neq \emptyset \wedge s_{n'',l''} \neq \emptyset \wedge s_{n',l'} \cap s_{n'',l''} = \emptyset. \end{aligned} \tag{3.20}$$

In Fig.3.5, $CS^{0}_{1,0}$ and $CS^{1}_{5,0}$ in $CS^{0}_{1,1}$ are operational independent when:

$$s_{1,0} \neq \emptyset \wedge s_{5,0} \neq \emptyset \wedge s_{1,0} \cap s_{5,0} = \emptyset. \tag{3.21}$$

- Geographic distribution: Refers to the physical dispersion of the components of a SoS across different locations. In the MLSHG model, $CS^{j'}_{n',l'}$ and $CS^{j''}_{n'',l''}$ of a system $CS^{j}_{n,l}$ are geographic distributed if their physical state ϕ^{ph} are distinct:

$$\begin{aligned} &\forall j, j', j'' \in \mathbb{N}, \forall CS^{j'}_{n',l'}, CS^{j''}_{n'',l''} \in CS^{j}_{n,l} : \\ &\phi^{ph}_{n',l'} \cap \phi^{ph}_{n'',l''} = \emptyset. \end{aligned} \tag{3.22}$$

In Fig.3.5, $CS^{0}_{1,0}$ and $CS^{1}_{5,0}$ in $CS^{0}_{1,1}$ are geographic distributed when:

$$\phi^{ph}_{1,0} \cap \phi^{ph}_{5,0} = \emptyset. \tag{3.23}$$

- Emergent behavior: All CSs within $CS^{j}_{n,l}$ cooperate to achieve their global mission, which cannot be accomplished by each CS independently. From the MLSHG perspective, this implies that the sum of all functions of under CSs is equal to the global functions that represent the global mission:

$$\begin{aligned} &(\forall 1 \leq o \leq k) : \\ &(q^{t}_{o}|R^{t}_{n,l}) = \sum_{i=0}^{l-1} \sum_{p=1}^{a_i} (q^{t}_{o}|R^{t}_{p,i}) \end{aligned} \tag{3.24}$$

In Fig.3.5, suppose that $CS^{0}_{1,1}$ posses K functions, $CS^{0}_{1,0}$, $CS^{1}_{5,0}$ and $CS^{0}_{2,0}$ in $CS^{0}_{1,1}$ generate an emergent behavior when:

$$\begin{aligned} &(\forall 1 \leq o \leq k) : \\ &(q^{t}_{o}|R^{t}_{1,1}) = (q^{t}_{o}|R^{t}_{1,0} + q^{t}_{o}|R^{t}_{5,0} + q^{t}_{o}|R^{t}_{2,0}) \end{aligned} \tag{3.25}$$

It should be noted that $(q_i|Y)$ means the quantifies value of function i in set Y.

- Evolutionary development: The SoS evolves over time, meaning that the functions and sub-missions are adapted according to the change in the global mission through a reconfiguration strategy. This is possible

when the quantification set $R^t_{n,l}$ of mission $M^t_{n,l}$is changed over time. Therefore, for all CSs these changes are formulated as :

$$\begin{aligned} \exists t_i, t_j \in \mathbb{R}, t_i \neq t_j \\ R^{t_i}_{n,l} \neq R^{t_j}_{n,l} \end{aligned} \tag{3.26}$$

For example in Fig.3.5, the functions of $CS^0_{1,0}$ changes over time and:

$$\begin{aligned} \exists t_i, t_j \in \mathbb{R}, t_i \neq t_j \\ R^{t_i}_{1,0} \neq R^{t_j}_{1,0} \end{aligned} \tag{3.27}$$

In addition, new CSs can be added or removed from the SoS depending on certain stochastic processes, such that there is at least one CS whose existence is represented by the attribute $A^t_{n,l}$ where :

$$\begin{aligned} \exists t_i, t_j \in \mathbb{R}, t_i \neq t_j \\ A^{t_i}_{n,l} \neq A^{t_j}_{n,l} \end{aligned} \tag{3.28}$$

To illustrate this, the example in Fig. 3.6 is considered, where:

$$\begin{aligned} t_1, t_2 \in \mathbb{R}, t_1 \neq t_2 \\ A^{t_1}_{1,0} \neq A^{t_2}_{1,0} \end{aligned} \tag{3.29}$$

It should be noted that some of these characteristics were verified for individual CSs, including managerial independence, operational independence, and geographic distribution. Other characteristics, such as emergent behavior and evolutionary development, must be verified at the global SoS level. If a component system $CS^j_{n,l}$ lacks some of these properties, it is referred to as a **Virtual Component System (VCS)** [Soyez 2015].

3.4 Case study: Mushroom harvesting SoS

To validate this framework, a case study is conducted on a **mushroom harvesting system**, which includes **stochastic behavior** in **heterogeneous subsystems**. The system is considered an SSoS composed of **biological systems (mushrooms)**, **social systems (human operators)**, and **mechatronic systems (robots)**. Both stochastic and deterministic behaviors emerge within these subsystems, contributing to the overall emergent behavior of the entire system, with the primary mission being the successful harvesting of mushrooms.

3.4.1 Introduction

Mushroom production has emerged as an important agricultural sector, attracting attention for its unique characteristics and the valuable products it generates. The industry involves several stages, from the initial cultivation of fungal mycelium to the final harvesting of mushrooms.

As mushrooms grow on animal and plant wastes such as horse manure, cereal seeds, wheat straw, rice straw, sugarcane waste, and others, production begins with composting. The main aim of composting is to make nutrients more easily accessible to the mushroom mycelium [Panayi 2017]. The second stage consists of filling the compost-containing substrate into specific beds in several chambers, where environmental conditions such as temperature, humidity and oxygen levels are precisely controlled, after filling the chamber, The mushroom mycelium spawn is applied to the compost where the mushroom production cycle begins, when the mushroom mycelium reaches its maturity threshold, mushroom initiation occurs lasting a few days, These mushrooms grow very quickly, doubling in size within 24 hours, and we need to pick them at a very precise time, since if we pick them early, we won't get the expected yield. The total life cycle of mushrooms is estimated at around 6 weeks, with initiation starting after 4 weeks.

In the world of mushroom cultivation, the development of these mysterious fungus frequently takes the form of an intriguing interaction between deterministic and stochastic processes. There is still some intrinsic stochasticity in the mushroom growth cycle, despite growers' painstaking control of elements like temperature, humidity, and substrate composition to optimize conditions for mycelium development and fruiting. Randomness is influenced by environmental variability, genetic variety among spore or mycelial strains, and the dynamic nature of microbial interactions in the substrate. The timing and yield of mushroom flushes can therefore be unpredictable, even with careful environmental management. This stochastic aspect makes mushroom farming an exciting combination since it forces growers to adjust to changing results and highlights the adaptability and tenacity of these incredible creatures. To this end, taking these uncertainties into account when modeling allows to obtain more realistic results in the mushroom industry [Banasik 2019].

Unlike traditional farming techniques, where mushroom production relies heavily on the expertise of human operators, modern mushroom farms are entering a new era of agricultural innovation. These cutting-edge facilities are adopting advanced technologies such as sensor networks, automated environmental controls and the integration of autonomous harvesting systems, including conveyors and harvest-assist robots [Hu 2019] to compensate for the well-known labour shortage in mushroom production [Reed 2001]. These

technological advances are revolutionizing mushroom cultivation by improving precision and efficiency throughout the production process. Thanks to these new technologies, mushroom growers are ready to meet the growing demand for these mushrooms, while reducing labor-intensive tasks and costs and guaranteeing consistent quality [Huang 2021].

In this study, the focus is on the harvesting stage of mushroom production by considering the harvesting system as a complex system, which can be viewed as a SSoS composed of heterogeneous CSs. Human operators are the primary CSs in this industry, but the demanding and repetitive nature of harvesting tasks has led to a decline in the attractiveness of these manual jobs. Additionally, the occurrence of random injuries can impact the performance of these social systems. To address these challenges, robots are introduced as supplementary CSs. While robots are not intended to entirely replace human operators—since they are not as effective—they contribute to social resilience by compensating for the shortage of human operators required for mushroom harvesting. Furthermore, mushrooms themselves are considered biological CSs, possessing their own resources and exhibiting stochastic behavior.

As a result, the proposed framework is explored by applying it to this system. In the following sections, the complexity of the system is explained, each CS is modeled, and the application of the MLSHG model for organizational modeling of this SSoS is demonstrated.

3.4.2 System complexity

The use of this framework for modeling the system is necessary because of its complexity, which makes traditional modeling techniques inadequate. This complexity creates challenges in the modeling process, mainly due to the behavioral and organizational aspects. Because of these complex factors, a more advanced modeling approach is needed, as traditional methods may not fully capture the detailed interactions and structure within the system. These complexities can be summarized as follows:

- Heterogeneity of CSs: As previously mentioned, the system comprises social, biological, and mechatronic systems. For example, in the case of mushrooms, their growth rate depends on various parameters such as temperature, humidity, and compost quality. Additionally, robot performance is subject to specific constraints when performing the harvesting task. Human operators also perform according to specific schedules and rules, and they possess their own decision-making capabilities.

- Interaction and emergent behavior: Changes in the performance of CSs, as well as interaction between them, affect the overall objective and pro-

duction. For example, robots and operators must interact with mushrooms by inspecting their condition, and human-robot interaction must be guaranteed for better collaboration and to avoid collisions.

- Dynamics, Adaptability and learning: The system evolves rapidly (mushrooms double their size within 24 hours), and all CSs have to adapt to this dynamic.

- Scenario complexity: Different scenarios, such as variations in production quantity, number of robots and their capacities, or environmental conditions, can lead to very different results.

3.4.3 Multi-level Mushroom harvesting modeling

In modeling the mushroom harvesting process on a farm using the MLSHG approach, the complex network of components is systematically decomposed to ensure efficient and effective management. In Fig. 3.7, an overview of the multi-level decomposition is presented, where, at the highest level of abstraction, the farm itself is the primary system, composed of various chambers. Each chamber represents a distinct subsystem with specific organizational properties, designed to manage its functions. Within these chambers are several beds with different levels and bays where mushrooms grow. Human operators or labors are central PCSs, serving as the main resources assigned to harvest specific beds. To enhance efficiency and precision, robots are introduced as additional PCSs, working alongside human operators to automate tasks and carry out specific mushroom harvesting activities. The implementation of a multi-level organizational model facilitates adaptive functional decomposition, enabling efficient system simulation and addressing supervision and resilience challenges. This systemic approach provides a thorough and integrated methodology for mushroom farming, establishing a robust and adaptable SoS essential for successful agricultural operations.

3.4.4 Hypergraph representation

Figure 3.8 illustrates the distribution of CSs within the Farm SoS. In Figure 3.9, the corresponding HG representation at a specific instance, denoted as t, is presented. The mathematical nested set-based representation of hierarchical structure is depicted as follows:

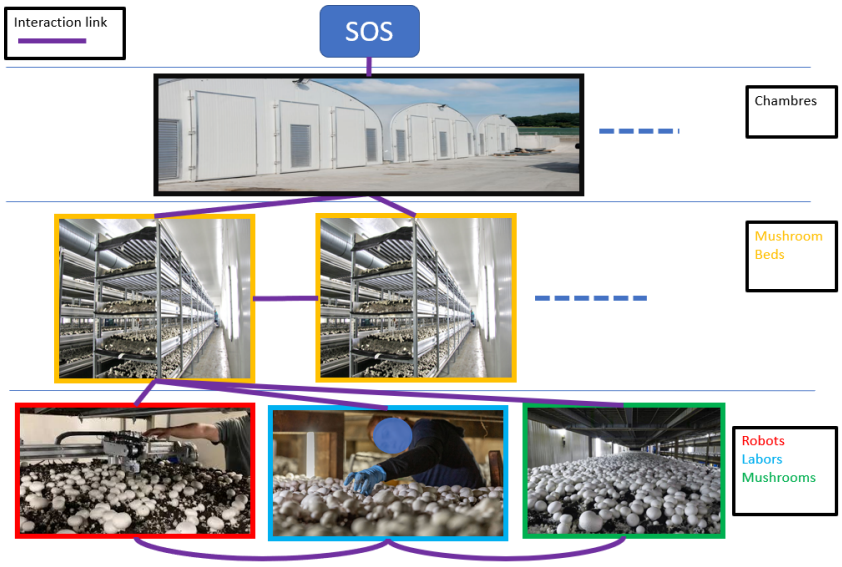


Figure 3.7: Multi-level Mushroom harvesting modeling

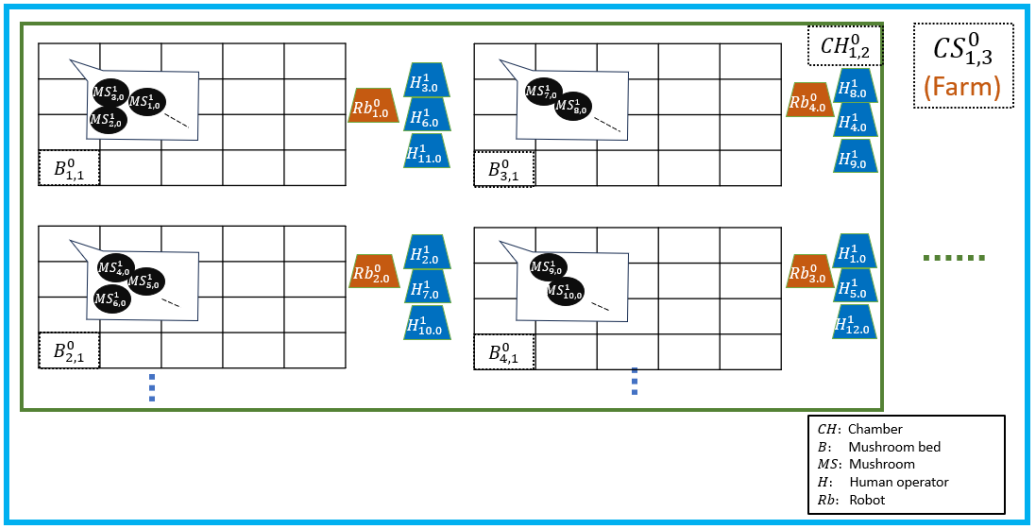


Figure 3.8: Mushroom farm component systems

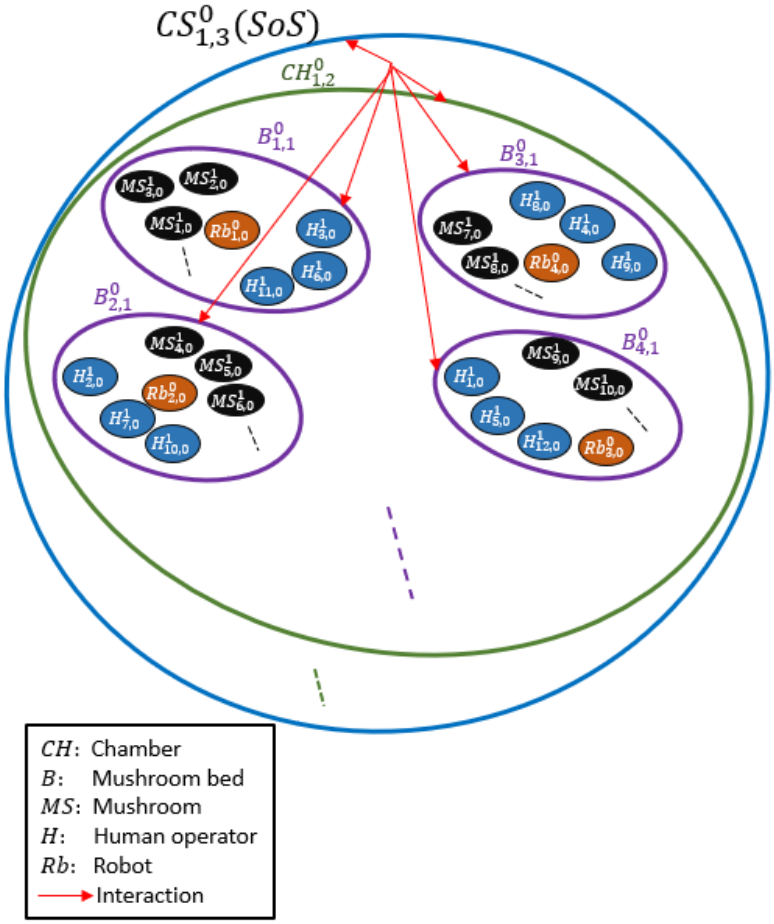


Figure 3.9: Hypergraph representation of musrhoom harvesting SoS

$$
\begin{aligned}
CS^0_{1,3} =& \{CH^0_{1,2}\} \\
=& \{\{B^0_{1,1}, B^0_{2,1}, B^0_{3,1}, B^0_{4,1}\}\} \\
=& \{\{\{MS^1_{1,0}, MS^1_{2,0}, ...H^1_{3,0}, H^1_{6,0}, H^1_{11,0}, Rb^0_{1,0}\}, \\
& \{MS^1_{4,0}, MS^1_{5,0}, ...H^1_{2,0}, H^1_{7,0}, H^1_{10,0}, Rb^0_{2,0}\}, \\
& \{MS^1_{7,0}, MS^1_{8,0}, ...H^1_{4,0}, H^1_{8,0}, H^1_{9,0}, Rb^0_{4,0}\}, \\
& \{MS^1_{9,0}, MS^1_{10,0}, ...H^1_{1,0}, H^1_{5,0}, H^1_{12,0}, Rb^0_{3,0}\}\}\}.
\end{aligned} \tag{3.30}
$$

It should be noted that all MCSs are considered as DCS ($j = 0$) in this case study, as it is assumed that these organizational components always exist in the SoS and do not exhibit proper stochastic behaviors. Howevere PCSs, may be either SCS or DCS based on their behavior.

The global mission of this SSoS is to achieve a mushroom harvesting rate that meets market demand. To ensure the success of this harvesting mission,

three main functions must be performed: growing, harvesting, and inspection. Mushrooms must grow at the desired rate to fulfill this mission, and the mature mushrooms should be inspected and detected before being harvested. The SoS mission is defined as $M^t_{1,3} = \{\text{growing}, \text{harvesting}, \text{inspection}\}$, and its corresponding quantifying set is $R^t_{1,3} = \{q^t_{\text{growing}}, q^t_{\text{harvesting}}, q^t_{\text{inspection}}\}$. These should then be decomposed across CSs based on their capabilities. The definitions of each CS are in the following subsections.

3.4.4.1 Mushroom

This is considered as SCS, and a vertex in level 0 (PCS) with the following representation:

$$MS^1_{n,0} = \left\langle \emptyset, P^t_{n,0}, Q_{n,0}, R^t_{n,0}, A^t_{n,0}, C^t_{n,0} \right\rangle, \tag{3.31}$$

Mushrooms exhibit stochastic behavior ($D^t_{n,0} = \emptyset$). Their primary functional capability is to grow in their bed, represented by $Q_{n,0} = \{q_{growing}, 0, 0\}$. The desired time-varying growth function is $R^t_{n,0} = \{q^t_{growing}, 0, 0\}$, whereas the actual grow rate performance is denoted by $P^t_{n,0} = \{p^t_{growing}, 0, 0\}$. A mushroom is considered to exist in the SoS only when it reaches its maturity threshold, indicated by $A^{t_m}_{n,0} = 1$, and disappears from the SoS when it is harvested, in which case $A^{t_h}_{n,0} = 0$.

The growth of mushrooms is influenced by several environmental factors, such as humidity and temperature. Various studies have attempted to optimize these factors [De La Croix 2022]. Other research has focused on statistically modeling mushroom behavior [Panayi 2023], while some studies have approached the problem through mathematical modeling [Chanter 1978]. Despite these efforts, modeling the behavior of mushrooms over time remains a significant challenge due to the extensive amount of data required, which is not currently available. As a result, detailed behavior modeling falls outside the scope of this study.

However, the growth of a group of mushrooms can be estimated by predicting the expected number of mature mushrooms over time. This estimation assumes that their growth behavior follows an exponential function, as suggested by [Chanter 1978]. By leveraging expert knowledge and historical data collected from a mushroom farm called "Ferme de la Gontière" in Comines, France, the number of mature mushrooms over time can be estimated.

The number of mature mushrooms, $MN(t)$, in a single chamber can be modeled using the curve shown in Fig. 3.10, which represents the expected number of mature mushrooms over time. This curve indicates that all mushrooms should reach maturity 7 days after the initial growth phase begins, with DM representing the final number of mature mushrooms.

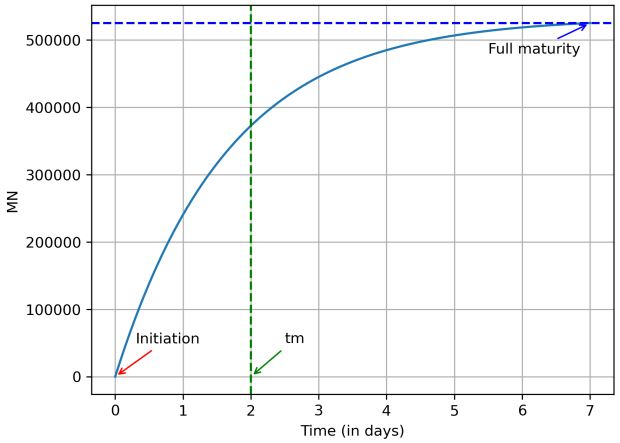


Figure 3.10: Mature Mushrooms in a single chamber (MN(t))

The stochastic existence attribute $A^t_{n,0}$ of each mushroom is assumed to be independent and identically distributed (i.i.d) for an arbitrary instant t. In addition, $A^t_{n,0}$ is assumed to obey the Bernoulli distribution at each time t, that is,

$$\begin{cases} P(A^t_{n,0} = 1) = p(t); \\ P(A^t_{n,0} = 0) = 1 - p(t), \end{cases} \tag{3.32}$$

where $p(t)$ is a time-varying function that represents the probability of a mushroom maturing at time t. The formulation of $p(t)$ was approximated using the number of mature mushrooms $MN(t)$. For arbitrary time t, the sum of all existence attributes $A^t_{n,0}$ obeys a binomial distribution, which can be depicted by

$$P(\sum_{n=1}^{DM} A^t_{n,0} = k) = \binom{n}{k} p(t)^k (1 - p(t))^{n-k}, \tag{3.33}$$

with $k = 0, 1, 2, ..., DM$. In addition, the expectation of $\sum_{n=1}^{DM} A^t_{n,0}$ satisfies

$$E(\sum_{n=1}^{DM} A^t_{n,0}) = DM * p(t). \tag{3.34}$$

Thus, the number of mature mushrooms $MN(t)$ can be used to approximate the maturity probability $p(t)$,

$$E(\sum_{n=1}^{DM} A^t_{n,0}) = DM * p(t) \approx MN(t), \tag{3.35}$$

which yields

$$p(t) \approx \frac{MN(t)}{DM}. \tag{3.36}$$

3.4.4.2 Human operator

A Human operator is treated as an SCS and a vertex in level 0 PCS with the following representation:

$$H^1_{n,0} = \left\langle \emptyset, P^t_{n,0}, Q_{n,0}, R^t_{n,0}, A^t_{n,0}, C^t_{n,0} \right\rangle, \tag{3.37}$$

Its stochastic attribute $P^t_{n,0} = \{0, p^t_{harvesting}, 0\}$ reflects the operator's harvesting performance. The desired harvesting function is $R^t_{n,0} = \{0, q^t_{harvesting}, 0\}$, and its capability function is $Q_{n,0} = \{0, q_{\text{harvesting}}, 0\}$. These social systems operate in shifts during the day only. Additionally, during their shift, they exist in the SoS only if they are performing normally ($A^t_{n,0} = 1$), meaning that the desired harvesting mission is satisfied ($C^t_{n,0} = 1$).

To enhance the modeling of stochastic performance degradation for $H^1_{n,0}$, a Bernoulli distribution with a probability of degradation is applied at each time step for each human operator, which may result in CSs failure. The probability of degradation is either assumed or based on expert input (e.g., 5%).

3.4.4.3 Robot

A robot system is considered as a DCS and a vertex at level 0 PCS with the following representation:

$$Rb^0_{n,0} = \left\langle D^t_{n,0}, \emptyset, Q_{n,0}, R^t_{n,0}, A^t_{n,0}, C^t_{n,0} \right\rangle, \tag{3.38}$$

Its deterministic performance attribute $D^t_{n,0} = \{0, d^t_{harvesting}, d^t_{inspection}\}$ includes the performance of the robot's harvesting and inspection functions since the robot is equipped with cameras that can inspect the mushroom's status and a gripper for harvesting. The desired harvesting and inspection functions are represented by $R^t_{n,0} = \{0, q^t_{\text{harvesting}}, q^t_{\text{inspection}}\}$, which are assigned from high MCSs. The capability functions are $Q_{n,0} = \{0, q_{\text{harvesting}}, q_{\text{inspection}}\}$, representing the maximum capacity the robot can perform. The robot operates continuously during the day and night, as indicated by $A^t_{n,0} = 1$, and since it is assumed to be a DCS, there is no stochasticity in the performance.

3.4.4.4 Mushroom beds

Mushroom beds are hyperedges (MCS) with a higher level of management. A bed has specific attributes and is described as follows:

$$B^0_{n,1} = \left\langle D^t_{n,l}, \emptyset, Q_{n,l}, R^t_{n,l}, A^t_{n,l}, C^t_{n,l}, F_{n,l} \right\rangle, \tag{3.39}$$

The mushroom bed is assigned harvesting and inspection functions, its deterministic attribute $D^t_{n,0}$ describes the related to these functions, in addition $A^t_{n,l} = 1$ if there is at least one mushroom in the bed, $F_{n,l}$ represents the function that assigns the under CSs functions .

3.4.4.5 Chambers

Chambers are hyperedges (MCS) with a higher level of management. The chamber has specific attributes and is described as follows:

$$CH^0_{n,1} = \left\langle D^t_{n,l}, \emptyset, Q_{n,l}, R^t_{n,l}, A^t_{n,l}, C^t_{n,l}, F_{n,l} \right\rangle, \tag{3.40}$$

This MCS has deterministic attribute $D^t_{n,l}$ representing the harvesting and inspection functions. $F_{n,l}$ is the mapping function of these functions under the CSs functions. A CS chamber always exists when at least one of its lower CSs exists ($A^t_{n,l} = 1$).

3.4.4.6 Interactions

A mushroom component system $MS^1_{n,0}$ in a hyperedge bed $B^0_{n',l}$ has a weight defined as follows:

$$\begin{aligned} W(MS^1_{n,0}, B^0_{n',l}) =< R^t_{n,0,n',l}, \emptyset, \\ \emptyset, A^t_{n,0,n',l} > . \end{aligned} \tag{3.41}$$

The mushroom is PCS with no information exchange with other CSs, thus no deterministic or stochastic interactions occur. The growing function of the mushroom within the bed is given by $R^t_{n,0,n',l} = \{q^t_{growing}, 0, 0\}$. The condition $A^t_{n,0,n',l} = 1$ holds true only if both $MS^1_{n,0}$ and $B^0_{n',l}$ exist, as stated previously.

A human operator $H^1_{n,0}$ in a hyperedge bed $B^0_{n',l}$ has a weight defined as follows:

$$\begin{aligned} W(H^1_{n,0}, B^0_{n',l}) =< R^t_{n,0,n',l}, DI^t_{n,0,n',l}, \\ SI^t_{n,0,n',l}, A^t_{n,0,n',l} > . \end{aligned} \tag{3.42}$$

Where $R^t_{n,0,n',l} = \{0, q^t_{harvesting}, 0\}$ represents the desired harvesting functions, $DI^t_{n,0,n',l}$ represents the interaction (information exchange) with other deterministic CSs, such as the robot's performance, and $SI^t_{n,0,n',l}$ represents the interaction (information exchange) with other stochastic CSs, such as the performance of other human operators. The condition $A^t_{n,0,n',l} = 1$ holds true only if $H^1_{n,0}$ performs the desired mission assigned by $CH^0_{n',l}$.

Similarly, a robot $Rb^0_{0,l}$ in a hyperedge bed $B^0_{n',l}$ has a weight defined as:

$$\begin{aligned} W(Rb^0_{n,0}, B^0_{n',l}) =< R^t_{n,0,n',l}, DI^t_{n,0,n',l}, \\ SI^t_{n,0,n',l}, A^t_{n,0,n',l} > . \end{aligned} \tag{3.43}$$

Where $R^t_{n,0,n',l} = \{0, q^t_{harvesting}, q^t_{inspection}\}$ represents the harvesting and inspection functions, $DI^t_{n,0,n',l}$ represents the interaction with deterministic CSs, such as the performance of other robots, and $SI^t_{n,0,n',l}$ represents the interaction with the stochastic performance of human operators. The condition $A^t_{n,0,n',l} = 1$ holds true only if $Rb^1_{n,0}$ performs the desired mission assigned by $B^0_{n',l}$. Note that it is assumed robots are equipped with cameras, allowing for the precise detection of mushrooms within the bed, which is why the inspection function is assigned to them.

3.4.5 Implementation

When applying the MLSHG framework to the mushroom harvesting SoS, **FlexSim**, a professional simulation tool for business systems across various industries, was used and adapted to reflect the multi-level SoS framework. The higher-level organizations (MCSs), represented by hyperedges (such as "chambers" or "beds"), differ from the Level 0 CSs in their supervisory roles and function assignment capabilities.

In FlexSim, the designed attributes of the framework—representing performance, mission, satisfaction of mission, and stochastic existence—are reflected by the labels associated with each CS. The exchange of performance information between CSs models their interactions. The functionality of each CS is developed using the Process Flow tool, which enables the creation of logic models and custom functionalities through the use of blocks and custom code snippets in C++ to simulate the logic and flow of processes within a complex system.

The managerial independence property is maintained by assigning each CS an independent mission that is performed based on its own logic, developed using logic flow and reflecting the real heuristics of the farm. Each CS in the simulation possesses its own resources, and even if the SoS farm and central management are separated, each CS will continue its operation independently, thereby respecting the operational independence property.

Fig. 4.4 presents a 3D visualization of a single chamber containing four beds (each pair positioned together with similar dimensions of real work farm), along with human operators and robots, the description of these CSs in the simulation is decribed below:

- Mushrooms (Fig.3.12): At time 0, representing the initiation , mature mushrooms are generated in the beds at random positions based on a time-dependent binomial distribution, as explained before, each one weighs 30 g.

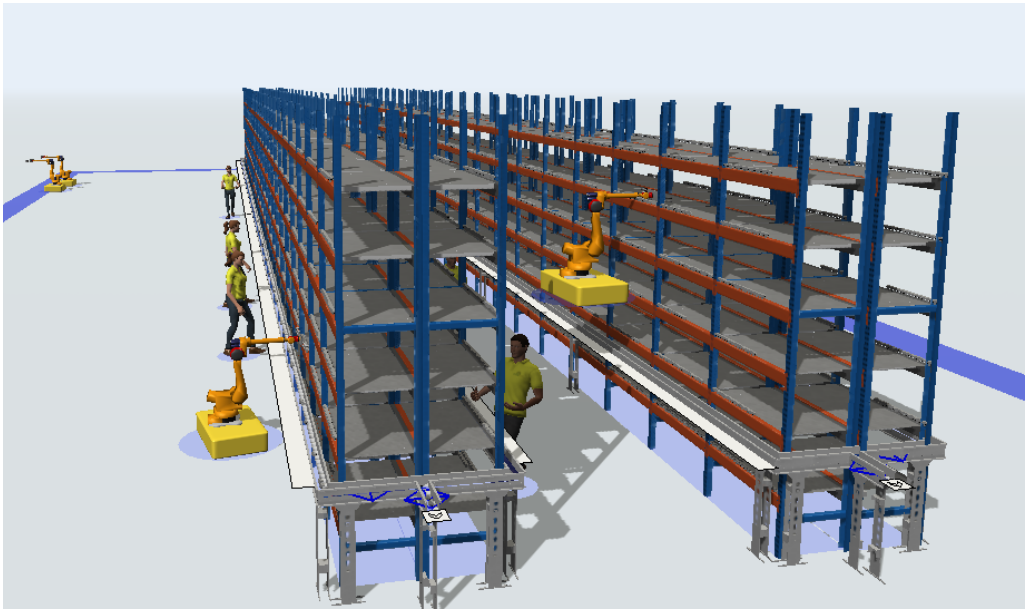


Figure 3.11: Simulation environment

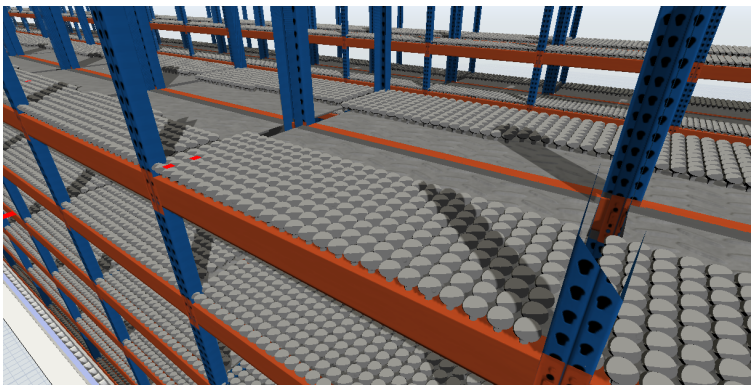


Figure 3.12: Mature mushrooms

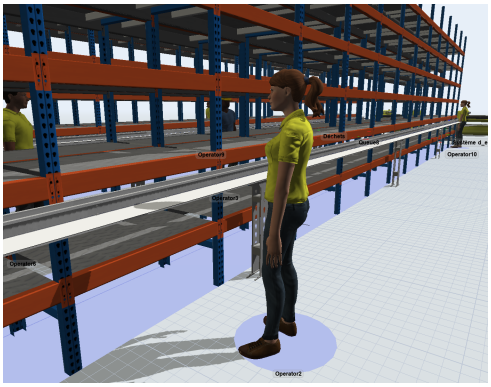


Figure 3.13: Human operator

- Human operator (Fig.3.13) : After two days from initiation, human operators begin harvesting mature mushrooms and placing them onto a conveyor system located next to the mushroom beds. These conveyors transport the mushrooms to the packaging and storage stages. The human operator CSs work in 8-hour shifts during the day, following specific harvesting heuristics. They harvest the mushrooms slot by slot, starting from one level of the bed, and once all mushrooms from that level are collected, they move to the next level. This logic is implemented in Flexsim using the logic flow tool, representing the managerial independence of these CSs within the SoS. The average harvesting capability of these CSs is shown in Table 3.1 (based on expert informations from the mushroom farm "Ferme de la Gontière" in Comines, France.). Since these are SCSs, their performance may decline, potentially leading to failures.

- Robot (Fig.3.14) : Robots begin harvesting alongside human operators, though with lower capabilities as shown in Table 3.1. However, unlike human operators, robots work continuously day and night without experiencing fatigue. They follow similar harvesting heuristics as the human operators, implemented in Flexsim's logic flow for each robot. Additionally, it is assumed that the robots are equipped with cameras to detect the status of the mushrooms. This information is then shared with other CSs by storing the data on mature mushrooms in a list accessible to all CSs within the Flexsim simulation.

3.4.6 Results and discussion

The simulation was run for 7 days, collecting data on the number of mature mushrooms in the chamber over time, the total harvested mushrooms, and

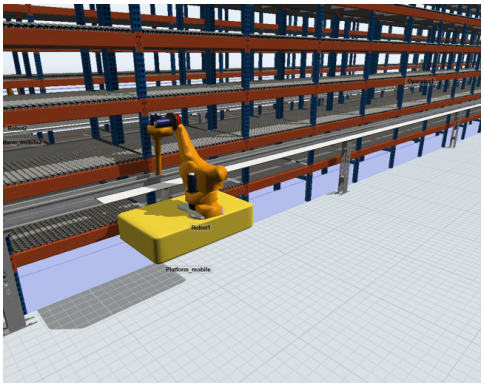


Figure 3.14: Robot

	Harvesting rate (Kg/h)
Human operator	30
Robot	10

Table 3.1: Harvesting capability

the harvesting rate. These results are plotted in Fig. 3.15 to compare the outcomes of different scenarios.

From the initiation day, the number of mature mushrooms in the chamber begins to increase, following a binomial process. After two days, the number of mature mushrooms starts to decrease as the harvesting process begins. The harvesting rate varies depending on the number of human operators and robots, as shown in the figure. In the scenario with 12 human operators, the harvesting rate drops to zero at night since no harvesting occurs during that time. However, in the other two scenarios, the process continues both day and night, as robots operate around the clock.

In scenario 3, where there are 12 human operators and 4 robots, the total mushroom yield reaches 13.3 tonnes. This increase is due to the higher number of CSs with harvesting functions. In contrast, reducing the number of CSs leads to lower yields. Scenarios 1 (12 human operators) and 2 (10 human operators and 2 robots) produce nearly the same yield of 10.6 tonnes, even though human operators are more efficient at harvesting. This is because robots can work continuously, day and night, compensating for the reduced number of human operators over the long term.

The fluctuations in the harvesting rate are caused by the stochastic nature of mushroom CSs, as they are randomly generated in different locations and vary in their growth stages over time. It was also observed that the higher the number of resources, the greater the performance fluctuations. This is due

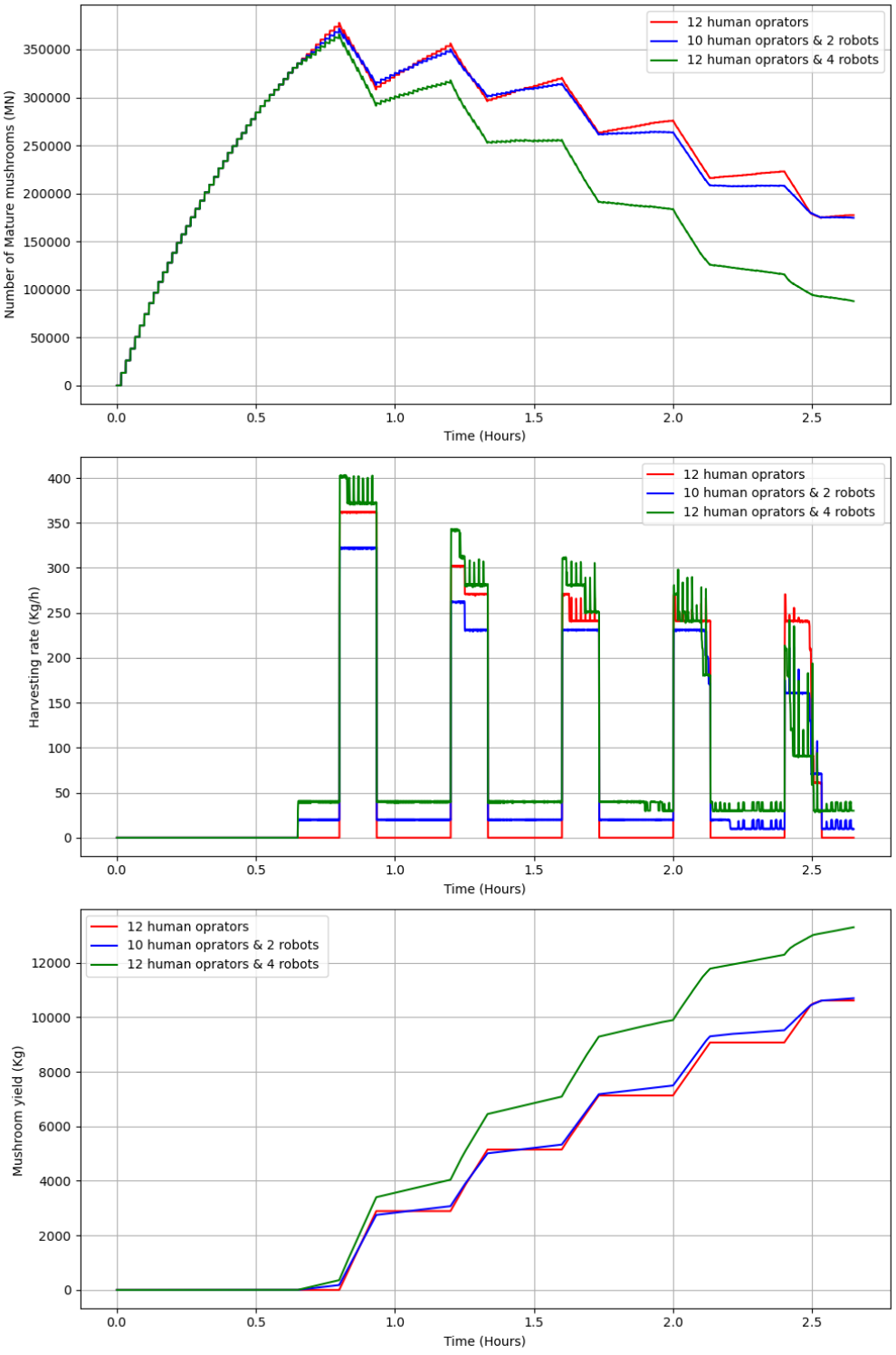


Figure 3.15: The number of mature mushrooms with time and the total mushroom yield with time and the harvesting rate in different scenarios

to resources occasionally pausing to avoid collisions with others and to locate unharvested slots. Additionally, since human operators function as SCSs, their performance may degrade, leading to CS failures and a subsequent drop in the harvesting rate, as shown in Fig. 3.15. In real-world scenarios, this decline is typically due to fatigue.

3.5 Conclusion

In this chapter, we introduced the **MLSHG** model for modeling **SSoS**. This model represents individual CSs based on their **performance**, **missions**, **capabilities**, **mission satisfaction**, and **presence** within the system. Additionally, interactions between these subsystems are captured using **edge-dependent weights**. The MLSHG model accommodates both stochastic and deterministic CSs (SCSs and DCSs), as well as physical and managerial CSs (PCSs) and MCSs).

The MLSHG model effectively addresses several significant challenges. A detailed explanation of these challenges is provided below, along with a comparison to other existing studies.

- **Behavior:** This pertains to the model's capacity to capture the dynamics of PCSs. The BG model utilized in [Kumar 2014, Kumar 2017] has been employed to analyze this dynamic behavior of PCSs.

- **Multi-level Organization:** This refers to the model's capability to depict a complex SoS within a multi-level hierarchical framework. This approach simplifies complexity and facilitates the distribution of missions and objectives. The models presented in [Khalil 2012, Soyez 2015, Kumar 2017, Dridi 2020, Mohsin 2020, Delsing 2022] effectively illustrate the organizational aspects of their methodologies.

- **Heterogeneity:** This pertains to the diversity among the CSs that comprise the broader SoS. The model must incorporate various functions or services among these CSs to meet the overall objectives. The models discussed in [Soyez 2015, Dridi 2020, Mohsin 2020, Delsing 2022] effectively manage these functionalities across different CSs.

- **Stochasticity:** This refers to the model's ability to capture and represent stochastic behavior CSs. While there are relatively few studies addressing this aspect, [Mohsin 2020] tackled the challenge using the SAM-SoS (Stochastic Software Architecture Model), with a specific focus on software architecture.

- **Scalability and Computational Efficiency:** Scalability pertains to the system's capacity to expand or contract its components, capabilities, and resources while preserving performance. It is crucial for the model to remain computationally efficient to be applicable in real-world scenarios. The ABM presented in [Soyez 2015] is scalable, as ABM typically decentralize computations to individual agents, effectively addressing these challenges.

- **Dynamic Adaptation:** This refers to the system's capability to modify its behavior, structure, or functions in response to real-time changes. The models presented in [Khalil 2012, Kumar 2017, Dridi 2020, Mohsin 2020, Delsing 2022] enable the SoS to adapt to particular conditions by adjusting its architecture.

- **Complex interactions:** This pertains to the capability to manage various relationships and dependencies among CSs, rather than concentrating solely on individual interactions. The studies in [Khalil 2012, Soyez 2015, Kumar 2017, Dridi 2020, Delsing 2022] effectively model these complex interactions.

The proposed MLSHG model addresses all these aspects except for behavioral modeling, as its multi-level framework emphasizes the structure, performance, functions, and capabilities of heterogeneous CSs, including mushrooms, human operators, and robots, while effectively capturing their complex interactions. Furthermore, the model accommodates both stochastic and deterministic CSs. It is designed to be scalable, computationally efficient, and capable of dynamic adaptation in response to disturbances. Table 3.2 summarizes the comparison between the proposed MLSHG and related studies.

In a case study, the **mushroom harvesting SoS**, which includes **social**, **physical**, and **biological components**, is modeled using the MLSHG framework. The model captures the system's functions, performance, capabilities, and interactions between its components. To improve function allocation and reduce complexity, additional managerial components such as mushroom beds and chambers are introduced. The simulation was performed using Flexsim over a single flush cycle (7 days) to analyze the outcomes of various scenarios.

Although the simulation results are close to what it was expected in a real farm environment, further calibration is required using additional data. This is necessary because certain assumptions were made, particularly in modeling the mushroom growth process. Despite these limitations, the simulation serves as a powerful tool for farmers during the early stages of system design. It enables them to evaluate and compare different outcomes under various

scenarios, providing insights into how different factors like different number of resources may impact production.

For effective **real-time management**, additional steps are necessary to enhance the model's practical use. Specifically, the system needs to be continuously monitored to detect any disturbances that could affect the growth and harvesting process. This monitoring will allow for adjustments to be made in real time, ensuring that the system operates efficiently and meets production goals. Moreover, the implementation of reconfiguration methods will be critical to maintain the performance. These techniques, which will be further developed in the following chapter, will allow the system to adapt dynamically and support farmers in managing their operations more effectively.

Method	Reference	Behavior	Multi-level organisation	Heterogeneity	Stochasticity	Scalability	Dynamic adaptation	Complex interactions
HG	[Khalil 2012]	✗	✓	✗	✗	✗	✓	✓
BG	[Kumar 2014]	✓	✗	✗	✗	✗	✗	✗
ABM	[Soyez 2015]	✗	✓	✓	✗	✓	✗	✓
BG & HG	[Kumar 2017]	✓	✓	✗	✗	✗	✓	✓
MeMSoS	[Dridi 2020]	✗	✓	✓	✗	✗	✓	✓
SysML	[Delsing 2022]	✗	✓	✓	✗	✗	✓	✓
MLSHG	Our method	✗	✓	✓	✓	✓	✓	✓

Table 3.2: Comparison between the MLSHG and the most relevant studies in terms of SoS modeling.

Chapter 4

Supervision of stochastic SoS

Contents

4.1 Introduction

In the previous chapter, the **modeling of SSoS** was explained using the proposed MLSHG model. This SSoS integrates multiple independent CSs, with the goal of achieving higher levels of functionality and performance. These CSs often operate in uncertain and dynamic environments, exhibiting stochastic behaviors due to inherent randomness and unpredictable interactions among their components. **Supervising** such **SSoS** poses a significant challenge, requiring advanced methodologies to ensure reliability and consistent performance.

The stochastic nature of an SSoS arises from various sources of uncertainty, including both internal and external random fluctuations. These uncertainties can propagate through interconnected subsystems, leading to emergent behaviors that are difficult to predict and control. **Internal fluctuations**

may cause performance degradation, potentially resulting in the failure of CSs. **External fluctuations**, on the other hand, can lead to mismanagement of the SoS's capacity. This mismanagement may result in an excessive number of CSs, causing system over-capacity and waste, particularly in supply systems. That separating internal and external fluctuations is crucial for **effective capacity management** as it helps in identifying the root causes of capacity issues, thereby enabling more precise adjustments to manage system loads in complex systems.

The objective is to manage the capacity of the SSoS by effectively controlling the number of CSs in **over-capacity** situations and ensuring that the SoS can adapt and reallocate functions in **under-capacity** scenarios, all while still achieving its overall goals through strategic supervision.

While supervision of deterministic SoS is achieved by maintaining a normal mode of operation [Khalil 2012], supervising an SSoS is more complex due to the difficulty in distinguishing between normal and abnormal modes of operation. In this study, the focus is on achieving the long-term goals of the SoS while maintaining a normal operational mode. This normal mode can be attained over time by verifying the performance of the SoS's time-varying mission and checking against specific thresholds that should account for these stochastic aspects.

Supervision in this context involves multiple stages, including **monitoring**, **reconfiguration**, and **control** of the SoS to ensure it functions as intended despite stochastic fluctuations. This includes the development of algorithms and strategies that can adaptively adjust the system's operation.

4.2 Threshold design

The design of the **threshold** for achieving **long-term goals** requires careful consideration of the time-varying performance of missions, as previously mentioned. It's crucial to account for the cumulative impact of these missions over time to ensure the successful attainment of long-term objectives. In this study, a novel approach to goal achievement is proposed by formulating it as the integral of these time-varying missions. This method allows for a comprehensive evaluation of each mission's contribution to the overall objective, considering both short-term and long-term impacts.

Once the **final goal** is established, it is important to determine an acceptable threshold for the residual of the **accumulated missions**, typically expressed as a percentage of the final goal. This percentage can also be applied to time-varying missions, ensuring that the accumulated missions remain aligned with the overall goal. It is important to note that, in this study, it is

assumed that all functions and missions are quantifiable.

By verifying the performance of the time-varying mission through the following equation:

$$|d(t)_{\text{ref}} - d(t)_i| < \lambda_0 d(t)_{\text{ref}}, \tag{4.1}$$

where λ_0 is a constant (e.g., $\lambda_0 = 0.1$), $d(t)_{\text{ref}} = d(t)_{\text{opt}} + \delta(t)$ is the reference performance of the mission, composed of the optimal performance $d(t)_{\text{opt}}$ and some time-varying uncertainty $\delta(t)$ used as a reference for comparison, and $d(t)_i$ is the current scenario performance.

we can prove that:

$$\int_{t_0}^{t} |d(t)_{\text{ref}} - d(t)_i| \, dt < \int_{t_0}^{t} \lambda_0 d(t)_{\text{ref}} \, dt \tag{4.2}$$

This inequality represents that the final goal condition is satisfied.

Given the performance verification condition:

$$|d(t)_{\text{ref}} - d(t)_i| < \lambda_0 d(t)_{\text{ref}} \tag{4.3}$$

Both sides of the inequality can be integrated, provided that both functions are continuous and integrable over $[t_0, t]$, with respect to t from t_0 to t:

$$\int_{t_0}^{t} |d(t)_{\text{ref}} - d(t)_i| \, dt < \int_{t_0}^{t} \lambda_0 d(t)_{\text{ref}} \, dt \tag{4.4}$$

This becomes:

$$\int_{t_0}^{t} |d(t)_{\text{ref}} - d(t)_i| \, dt < \lambda_0 \int_{t_0}^{t} d(t)_{\text{ref}} \, dt \tag{4.5}$$

This represents the **condition** that the integrated deviation from the reference performance remains within a fraction λ_0 of the integrated reference performance, ensuring that the cumulative performance adheres to the final goal condition.

To **account stochastic disturbances** and for early detection and reconfiguration, the constant threshold for time-varying missions can be adapted as follows:

$$\lambda_{SoS}^{t} = \lambda_0 d(t)_{\text{ref}} \times g(\varepsilon(t)), \tag{4.6}$$

where λ_0 is the fixed threshold set by experts and $g(\varepsilon(t))$ adjusts the adaptive threshold based on stochastic disturbances.

4.3 Supervision algorithm

Short- and long-term objectives are crucial for managing complex systems [Paut 2021]. To address these, a supervision algorithm is developed consisting of **bottom-up monitoring** and **top-down reconfiguration**, as illustrated in Fig. 4.1, to maintain performance and ensure the achievement of long-term global goals. While other studies focus on interaction failures [Maksuti 2021], the bottom-up monitoring in this study involves measuring performance, detecting, and eliminating failed CSs based on the satisfaction of missions, represented by the attribute $C^t_{n,0}$.

The top-down reconfiguration process includes function decomposition (see Fig. 4.2), eliminating extra CSs, and capability-based reconfiguration. By removing extra CSs in over-capacity scenarios and reconfiguring the SoS through capability-based adjustments in under-capacity scenarios, this approach effectively addresses issues of over-capacity and under-capacity in large-scale systems, improving capacity management.

Algorithms 1 and 2 detail the different stages of the proposed real-time supervision process. Starting with setting the number of levels b, the number of CSs in each level a_l, the declaration of CSs, and the establishment of the SoS global mission, which is composed of k functions. This SoS mission represents the desired performance $d(t)_{\text{ref}}$ for the system. Thresholds are then defined, and performance data are collected from the simulation. The detection and elimination of failed CSs are carried out by generating the residuals between the desired function set $R^t_{n,0}$ and the actual performance sets $P^t_{n,0}$ and $D^t_{n,0}$, followed by a comparison with the CS threshold λ_{CS} to determine whether the CS mission is satisfied. This process is conducted in a bottom-up manner.

In the top-down approach, functions are decomposed over the CSs through the function $F_{n,l}$, where the desired functions of a CS are the sum of all assigned functions from the higher-level CSs. During the reconfiguration stage, the residual of SoS performance to the SoS threshold λ^t_{SoS} is compared. If an over-capacity is detected, the hyperedges with the most over-capacity is identified and the last CS added will be eliminated . If under-capacity is detected, the capability of the existing CSs to compensate for the failed one is checked. This methodology ensures effective capacity management of the SoS.

It should be noted that since disturbances are only assumed in SCSs, the residual for failure detection is only checked for these subsystems, using the performance attribute $P^t_{n,l}$. Additionally, all MCSs are considered as DCSs, with the performance attribute $D^t_{n,l}$.

Further explanation, along with the corresponding mathematical representations, is described in the following sections.

Algorithm 1 Real time Supervision algorithm (Part 1)

///Declaration

$b \leftarrow levels$

$S \leftarrow \{a_l; (a_l \in \mathbf{N})(0 \leq l \leq b)\}$

//Input CSs sets and attributes

$V \leftarrow \{CS^j_{n,0}; 1 \leq n \leq a_0\}$

for $l \leftarrow b$ to 1 **do**

  $E_l \leftarrow \{CS^j_{n,l}; 1 \leq n \leq a_l\}$

//Define thresholds

$\lambda_{CS} \leftarrow CS_{threshold}$

$\lambda^t_{SoS} \leftarrow SoS_{threshold}$

//Initialize global functions

$R^t_{1,b} = \{q^t_1, ..., q^t_k\}$

$k \leftarrow |R^t_{1,b}|$

while True **do**

  ///Monitoring

  //Data performance collection from simulation

  $Pr \leftarrow \{P^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$

  $Dr \leftarrow \{D^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$

  //Detecting and eliminating failed PCSs

  $D \leftarrow \{d \in \mathbb{R} : (\exists 1 \leq n \leq a_0)(\exists 1 \leq j \leq k)$

  $(d = q^t_j | R^t_{n,0} - p^t_j | P^t_{n,0})\}$

  if $(\exists d \in D)(d > \lambda)$ **then**

    /Get the first failed PCS index

    $n' \leftarrow (\exists 1 < j < k)((q^t_j | R^t_{n',0} - p^t_j | P^t_{n',0}) > \lambda_{CS})$

    /Eliminate PCS with n' index from SoS

    $C^t_{n',0} \leftarrow 0$

    $A^t_{n',0} \leftarrow 0$

Algorithm 2 Real time Supervision algorithm (Part 2)

///Reconfiguration

//Function decomposition

for $l \leftarrow (b-1)$ to 0 **do**

  $R^t_{n',l} \leftarrow \{x \in \mathbb{R} : (\exists 1 \leq j \leq (b-l))(\exists 1 \leq n \leq a_{l+j})(\exists 1 \leq i \leq k)(x = \sum_{j=1}^{b-l} \sum_{n=1}^{a_{l+j}} (f_{n,l+j,n',j}(q^t_i|R^t_{n,l+j}))\}(\forall 1 \leq n' \leq a_l)$

//Check over-capacity

if $(\sum_{s=1}^{k}(q^t_s|R^t_{1,b} - d^t_s|D^t_{1,b})| < -\lambda^t_{SoS})$ **then**

//Check over-capacity MCS

  $B(x,y) = \sum_{i=1}^{k}(q^t_i|R^t_{x,y}) - (d^t_i|D^t_{x,y})$

  $n1, l1 \leftarrow (\forall 1 \leq l \leq b)(\forall 1 \leq n \leq a_l)(B(n1,l1) > B(n,l))$

//Eliminate extra PCS

  $n2 \leftarrow (\forall 1 \leq n \leq a_0)((CS^j_{n,0} \in CS^0_{n1,l1}) \wedge (\sum_{t_0}^{t} p^t_i|P^t_{n2,0} < \sum_{t_0}^{t} p^t_i|P^t_{n,0}))$

  $A^t_{n2,0} \leftarrow 0$

//Check under-capacity

else if $(\sum_{s=1}^{k}(q^t_s|R^t_{1,b} - d^t_s|D^t_{1,b})| > \lambda^t_{SoS})$ **then**

/Check PCSs with similar capabilities

  if $(\exists 1 \leq n \leq a_0)(\exists 1 \leq i \leq k)$ **then**

    $((q_i|Q_{n,0} = q_i|Q_{n',0}) \wedge (q^t_i|R^t_{n,0} = 0))$

/Obtain the first CS capable of replacing the failed PCS

    $n'' \leftarrow (\exists 1 \leq i \leq k)((q_i|Q_{n'',0} = q_i|Q_{n',0}) \wedge (q^t_i|R^t_{n'',0} = 0))$

/Reallocate failed functions

    $q^t_i|R^t_{n'',0} \leftarrow (q^t_i|R^t_{n'',0} + q^t_i|R^t_{n',0})(\forall 1 \leq i \leq k)$

  else

    Print('Recovering is not possible')

4.3.1 Monitoring

The monitoring stage of the supervision algorithm is a crucial initial phase, conducted in a bottom-up manner and composed of two primary components: performance measurement and failure detection.

Performance measurement, is conducted from level 0 to level b, as the performance of higher-level CSs depends on the performance of lower-level CSs. Data is collected from all CSs through simulation and stored in two distinct sets: one for SCSs, represented as $Pr \leftarrow \{P^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$,

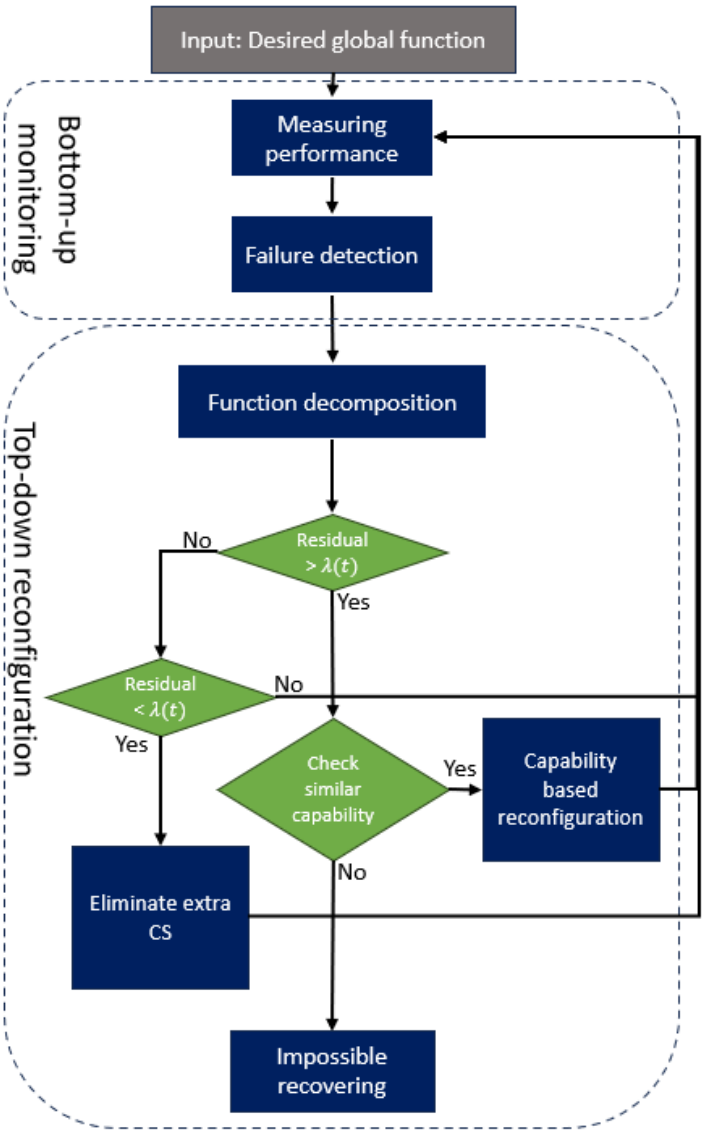


Figure 4.1: Supervision algorithm

and another for DCSs, represented as $Dr \leftarrow \{D^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$. This data is essential for detecting and verifying any deviations in global performance from the desired performance, particularly in scenarios of over-capacity and under-capacity.

Failure detection is performed at level 0 for PCSs, ensuring that any degraded PCSs are identified and removed early in the process. The process begins by calculating the residuals for all PCSs:

$$D \leftarrow \{d \in \mathbb{R} : (\exists 1 \leq n \leq a_0)(\exists 1 \leq j \leq k)(d = q^t_j | R^t_{n,0} - p^t_j | P^t_{n,0})\} \tag{4.7}$$

This expression represents the residual between the desired function j in set $R^t_{n,0}$, denoted by $q^t_j | R^t_{n,0}$, and its actual performance $p^t_j | P^t_{n,0}$. The residual d is stored in the set D for all functions and PCSs at level 0, as long as n and j satisfy their respective conditions.

These residuals are then compared against predefined thresholds to determine if any PCS has failed to satisfy its mission requirements through the condition:

$$(\exists d \in D)(d > \lambda) \tag{4.8}$$

If a failure is detected, the index of the first failed CS is identified as:

$$n' \leftarrow (\exists 1 < j < k)((q^t_j | R^t_{n',0} - p^t_j | P^t_{n',0}) > \lambda_{CS}) \tag{4.9}$$

The satisfaction attribute for the failed CS is then set to zero:

$$C^t_{n',0} \leftarrow 0 \tag{4.10}$$

Finally, the failed CS is eliminated by setting its stochastic existence to zero:

$$A^t_{n',0} \leftarrow 0 \tag{4.11}$$

Through this dual approach, the monitoring stage ensures that the system operates efficiently by maintaining data integrity for performance verification and promptly removing any failed CSs from the SoS, thus preparing the system for reconfiguration.

4.3.2 Reconfiguration

The reconfiguration stage is conducted in a top-down manner and is composed of three primary components: function decomposition, capability-based reconfiguration in the case of under-capacity scenarios, and the elimination of extra CSs in the case of over-capacity scenarios.

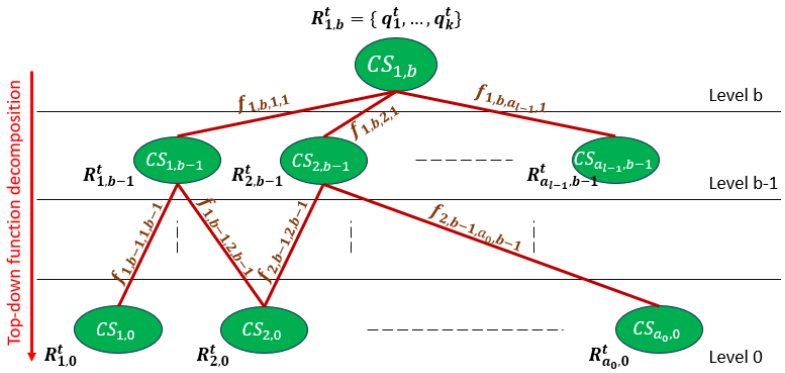


Figure 4.2: Function decomposition

In the function decomposition phase, the desired functions that represent the mission are decomposed from the top-down over all CSs, as illustrated in the Fig. 4.2. This decomposition is performed through the assigned functions $F_{n,l}$ of the MCSs ($F_{n,l}$ is defined in the modeling chapter), formulated as follows:

$$\begin{aligned} R^t_{n',l} \leftarrow \{x \in \mathbb{R} : (\exists 1 \leq j \leq (b-l))(\exists 1 \leq n \leq a_{l+j}) \\ (\exists 1 \leq i \leq k)(x = \sum_{j=1}^{b-l} \sum_{n=1}^{a_{l+j}} f_{n,l+j,n',j}(q^t_i | R^t_{n,l+j}))\} \\ (\forall 1 \leq n' \leq a_l) \end{aligned}$$

This expression represents the desired quantified functions for $CS^i_{n',l}$, assigned from all higher-level MCSs. It is formulated as the sum of the contributions from all relevant MCSs , noting that if $CS^i_{n',l}$ does not belong to $CS^{i'}_{n,l+j}$, then $f_{n,l+j,n',j} = 0$.

over-capacity checking is done by verifying that the residual between the reference performance (desired mission) and the current SoS performance is less than the negative of the threshold:

$$(\sum_{s=1}^{k} (q^t_s | R^t_{1,b} - d^t_s | D^t_{1,b})| < -\lambda^t_{SoS}) \tag{4.12}$$

If an over-capacity is detected, the system identifies the most over-capacityed MCS with index $n1$ and level $l1$, defined as:

$$B(x,y) = \sum_{i=1}^{k} (q^t_i | R^t_{x,y}) - (d^t_i | D^t_{x,y}) \tag{4.13}$$

$$n1, l1 \leftarrow (\forall 1 \leq l \leq b)(\forall 1 \leq n \leq a_l)(B(n1, l1) > B(n, l)) \tag{4.14}$$

Next, the system determines the index $n2$ of the last added PCS within this most over-capacityed MCS:

$$n2 \leftarrow (\forall 1 \leq n \leq a_0)((CS^j_{n,0} \in CS^0_{n1,l1}) \wedge (\sum_{t_0}^{t} p^t_i | P^t_{n2,0} < \sum_{t_0}^{t} p^t_i | P^t_{n,0})) \tag{4.15}$$

This means that all PCSs are checked, and the one in the most over-capacitated MCS with the lowest performance over time is identified. This helps confirm that the last added PCS was the cause of the over-capacity in the MCS.

Finally, the identified PCS is eliminated by setting its stochastic existence to zero:

$$A^t_{n2,0} \leftarrow 0 \tag{4.16}$$

In an under-capacity scenario, this condition is checked when the residual between the reference performance (desired mission) and current SoS performance is greater than the threshold:

$$\sum_{s=1}^{k} (q^t_s | R^t_{1,b} - d^t_s | D^t_{1,b})| > \lambda^t_{SoS} \tag{4.17}$$

If this condition is detected, capability-based reconfiguration is performed to compensate for the degraded performance. The system checks for CSs that have a similar capability with failed CS detected in the monitoring phase, with the following condition:

$$(\exists 1 \leq n \leq a_0)(\exists 1 \leq i \leq k)((q_i | Q_{n,0} = q_i | Q_{n',0}) \wedge (q^t_i | R^t_{n,0} = 0)) \tag{4.18}$$

This expression means that if any PCS has the same capability as the failed PCS but is not currently performing that function, it indicates that it is capable of taking over the failed functions.

Then, the system identifies the first CS with index n'' that is capable of performing the functions of the failed CS:

$$n'' \leftarrow (\exists 1 \leq i \leq k)((q_i | Q_{n'',0} = q_i | Q_{n',0}) \wedge (q^t_i | R^t_{n'',0} = 0)) \tag{4.19}$$

Finally, the functions of the failed CS are reallocated to this newly identified CS:

$$q_i^t|R_{n'',0}^t \leftarrow (q_i^t|R_{n'',0}^t + q_i^t|R_{n',0}^t)(\forall 1 \leq i \leq k) \tag{4.20}$$

This top-down reconfiguration approach ensures that the system remains flexible and responsive, adapting to changing conditions while maintaining the overall mission objectives.

4.4 Case study

To validate the proposed supervision methodology, it is applied to a mushroom harvesting system, conceptualized as a SSoS composed of heterogeneous CSs. These CSs exhibit both deterministic and stochastic behaviors that interact to achieve a common goal: efficient mushroom harvesting.

The stochastic behaviors include the growth patterns of mushrooms, which result in an unpredictable number of mature mushrooms being distributed randomly across the harvesting area at any given time. Another stochastic element is the degradation of human operators' performance, which can vary due to factors such as fatigue [Nicholls 2004], leading to fluctuations in the overall effectiveness of the SSoS.

On the other hand, the deterministic behavior is primarily exhibited by the robotic systems integrated into the harvesting process. These robots are designed with the specific purpose of compensating for the performance degradation of human operators. The deterministic nature of the robots' actions ensures a consistent level of performance, thereby stabilizing the overall functionality of the SoS despite the inherent uncertainties introduced by the stochastic components.

By applying the proposed real-time supervision methodology using the MLSHG model to the mushroom harvesting SoS, the goal is to demonstrate its effectiveness in managing the complex interactions between deterministic and stochastic behaviors, ultimately enhancing the reliability and efficiency of the harvesting process.

4.4.1 Real time Supervision of mushroom harvesting SoS

In the real time supervision of the mushroom harvesting SoS, the primary objective is to meet the daily harvesting yield within a specified time frame, typically within a single day, to ensure that market demands are fulfilled. This objective is achieved by maintaining a consistent harvesting rate throughout the day. Since human performance is critical to the reliability of agricultural

systems and harvesting operations [Yahia 2010, Purfürst 2011], fluctuations in performance and potential failures of human operators at certain times can lead to a reduction in the harvesting rate, resulting in an under-capacity that deviates from the target and threatens the achievement of the final yield goal.

Conversely, an over-capacity in the harvesting rate, which may occur if additional human operators are introduced, can cause an excessive deviation from the target, potentially leading to the loss of a portion of the mushroom yield. This imbalance could arise from improper handling or premature harvesting, which can negatively impact the overall yield.

Effective capacity management is crucial in agricultural operations and is a key factor in their success. Proper supervision and management of capacity are essential to prevent and address situations of both over-capacity and under-capacity [Vukelić 2014].

In this chapter, the focus is on achieving the target harvesting quantity as the long-term goal of the supervision methodology. At the same time, maintaining an optimal harvesting rate is identified as the critical mission of the SoS. By prioritizing these objectives, the supervision methodology is designed to ensure that the capacity of the mushroom harvesting SoS is managed efficiently and reliably, minimizing deviations from the desired outcomes.

Before applying the supervision algorithm, the mushroom harvesting SoS is modeled in terms of structural and HG representation. This includes modeling individual CSs and their interactions, covering aspects such as performance, capability, desired missions, and stochastic existence. The modeling is done using the MLSHG model, as detailed in previous Chapter.

4.4.2 Monitoring of mushroom harvesting SoS

4.4.2.1 Performance measurement

Monitoring the mushroom harvesting SoS is the initial stage in the supervision algorithm. This stage includes measuring the performance of the CSs, beginning with the primary PCSs at Level 0. At this level, the stochastic performance of human operators is assessed in relation to their harvesting tasks, while the deterministic performance of robots is evaluated in terms of their inspection and harvesting functions. Additionally, the performance of the mushroom growth mission is measured.

These performance data are then propagated to higher-level MCSs to assess their organizational performance. Specifically, this involves evaluating the mushroom beds at Level 1 and the chambers at Level 2. Subsequently, the performance data is aggregated to determine the current performance of the global SoS, which represents the entire mushroom harvesting operation.

It is important to note that in this work, its assumed that only the harvesting function's performance is susceptible to fluctuations, primarily due to failure of human operators. Further its assumed that other functions maintain consistent performance over time.

4.4.2.2 Failure detection

In addition to measuring performance, the monitoring phase should also include the detection of failed PCSs. The measured stochastic performance of human operators, $P^t_{n,0}$, the measured deterministic performance of robots, $D^t_{n,0}$, and the measured stochastic performance of mushrooms, $P^t_{n,0}$, are compared against the desired performance of the mission, $R^t_{n,l}$, to generate a residual d. This residual is then compared to their respective performance thresholds, λ.

These thresholds can vary depending on whether the system is a human operator, robot, or mushroom, as determined by experts in the field. However, for the sake of simplicity, it is assumed that all PCSs share the same threshold, which indicates whether they are reliable. If a PCS is detected as unreliable, the system removes it from the mushroom harvesting SoS by setting its satisfaction attribute $C^t_{n,l}$ and its stochastic existence $A^t_{n,l}$ to zero.

4.4.3 Reconfiguration of mushroom harvesting SoS

4.4.3.1 Function decomposition

Reconfiguration is the second phase of the supervision algorithm, performed after detecting and removing failed PCSs and acquiring the performance data of the SoS. This phase involves the decomposition of functions from the higher SoS level down to the PCSs. The functions of the SoS mission, $R^t_{1,3} = \{q^t_{\text{growing}}, q^t_{\text{harvesting}}, q^t_{\text{inspection}}\}$, are first decomposed across the chambers of the farm. These functions are then further decomposed over the mushroom beds within each chamber, and subsequently, over the PCSs (mushrooms, humans, robots) associated with each bed. This decomposition is guided by the $F_{n,l}$ attribute in each MCS, which assigns functions based on the capabilities of the respective CSs.

4.4.3.2 Reference scenario

While the monitoring and reconfiguration are performed in real-time, the generation of the reference scenario is done offline. This reference scenario represents the normal mode of operation on the farm and is generated by collecting real data from the mushroom farm "Ferme de la Gontier" in Lille, France,

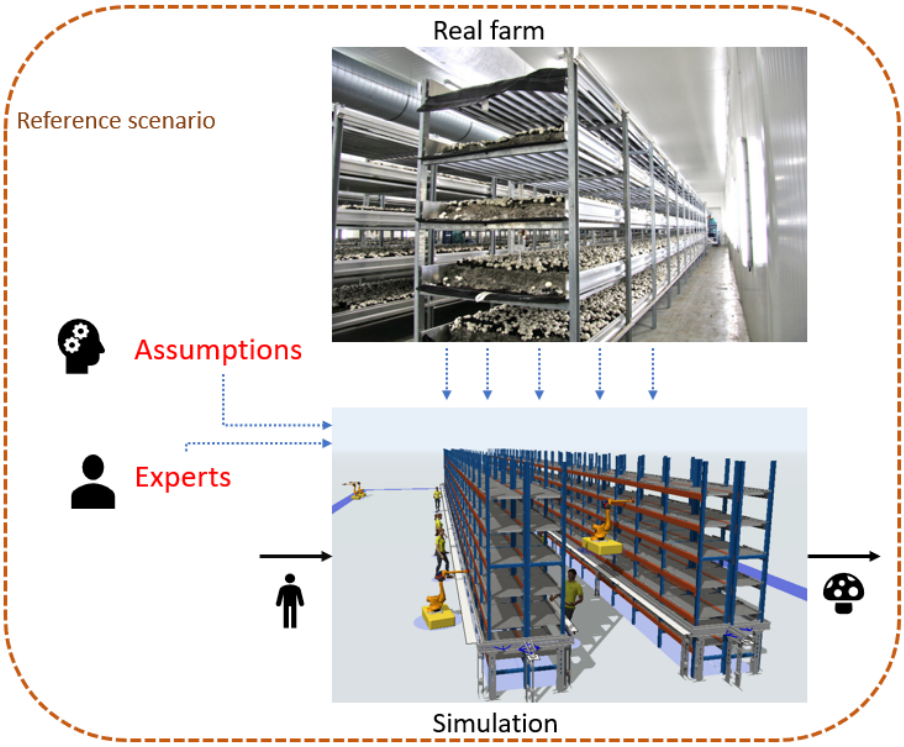


Figure 4.3: Reference scenario

along with information from experts and certain assumptions, as illustrated in the Fig.4.3.

- **Real Data:**
  - Chamber and bed dimensions.
  - Dimensions and weight of mature mushrooms.
  - Number of beds within a single chamber.
- **Expert Information:**
  - Average final number of mature mushrooms per bed.
  - Average harvesting rate of human operators.
  - Expected average harvesting rate of robots.
  - Number of human operators assigned to each chamber and bed.
  - Harvesting heuristics used by human operators.
  - Average daily harvesting yield.
- **Assumptions:**

- Exponential growth in the number of mature mushrooms over time.
- The capability of each PCS is assumed to be the average performance value for each one.

In this study, the reference scenario only includes human operators since the objective is to study the incorporation of robots to compensate for the performance degradation of the farm.

4.4.3.3 Threshold

To raise the alarm for over-capacity or under-capacity, the performance deviation should be compared with a threshold. The final acceptable deviation is considered 10% in the harvesting yield for a single day, representing the long-term goal. It has been previously verified that using this percentage for time-varying performance ensures the final goal is met. Therefore, the constant threshold considered for mission performance is defined as $\lambda^t_{SoS} = \lambda_0 d(t)_{\text{ref}}$, where $\lambda_0 = 0.1$.

To account for stochastic disturbances, the adaptive threshold in equation 4.6 is considered where:

$$\lambda^t_{SoS} = \lambda_0 d(t)_{\text{ref}} \times g(\varepsilon(t)), \tag{4.21}$$

where $\lambda_0 = 0.1$ is the fixed value, and

$$g(\varepsilon(t)) = \left(1 - \frac{\varepsilon(t)}{6}\right),$$

with $\varepsilon(t)$ representing the number of failed or additional human operators at each time step, and 6 being the maximum number of disturbances. This adaptive term accounts for the stochastic nature of disturbances, making the threshold more sensitive and allowing for robust reconfiguration. If the number of disturbances reaches 6, the threshold becomes zero, meaning the alarm is always raised as the maximum stochastic disturbance limit has been exceeded.

4.4.3.4 Over-capacity scenario

If the residual between the reference performance of the mushroom farm,and the current global performance of the farm, is less than the negative threshold, it indicates that the current performance will lead to an over-capacity. In this case, the system first identifies the mushroom bed (MCS at Level 1 in the HG) experiencing overcapacity. Then, the system identifies the human operator who was recently added and caused this overcapacity, and subsequently removes them from the system.

4.4.3.5 Under-capacity scenario

If the residual between the reference performance of the mushroom farm , and the current global performance of the farm, exceeds the threshold, it indicates that the current performance will lead to an under-capacity and that some human operators have failed. In this situation, the system searches for robots with similar capabilities that are not currently performing harvesting functions to compensate for the failed human operators. If no suitable robots are available, it means that reconfiguration is not possible.

While its known that maintaining performance degradation within the threshold ensures the final yield goal is achieved, it is also possible to meet the final target even if failures occur late in the harvesting process and no reconfiguration is made to compensate for them. This is because the final target is based on the integration of time-varying performance.

4.4.4 Implementation

The MLSHG model introduced in the modeling chapter is applied to the mushroom harvesting SoS and implemented in FlexSim, following a similar approach to the previous setup. Attributes of higher-level MCSs and lower-level PCSs are represented using labels, while the functionality of each component system is designed using the Process Flow tool. Furthermore, the proposed algorithm, which includes both monitoring and reconfiguration stages, is also implemented through Process Flow.

The simulation setup remains consistent with the previous implementation, using a single chamber with four mushroom beds, as shown in Fig. 4.4. However, this time, the simulation is run for a single shift (the first harvesting day) to monitor the system and implement necessary reconfigurations. The reference scenario is simulated first to establish baseline performance and set the final yield goal. This scenario involves only human operators performing the harvesting tasks, as is typical in real-world farms. In contrast, other scenarios introduce disturbances reflecting both over- and under-capacity conditions, allowing robots to compensate for the degradation in human operator performance.

Recognizing the importance of capacity management in the simulation, two scenarios are considered: under-capacity and over-capacity. In the under-capacity scenario, failures in human operator performance are modeled, which may occur due to fatigue in real situations. In the over-capacity scenario, the addition of extra operators is modeled, which could result from mismanagement on the farm in real scenarios. Both scenarios are modeled stochastically using a binomial distribution, and each is tested against the constant and

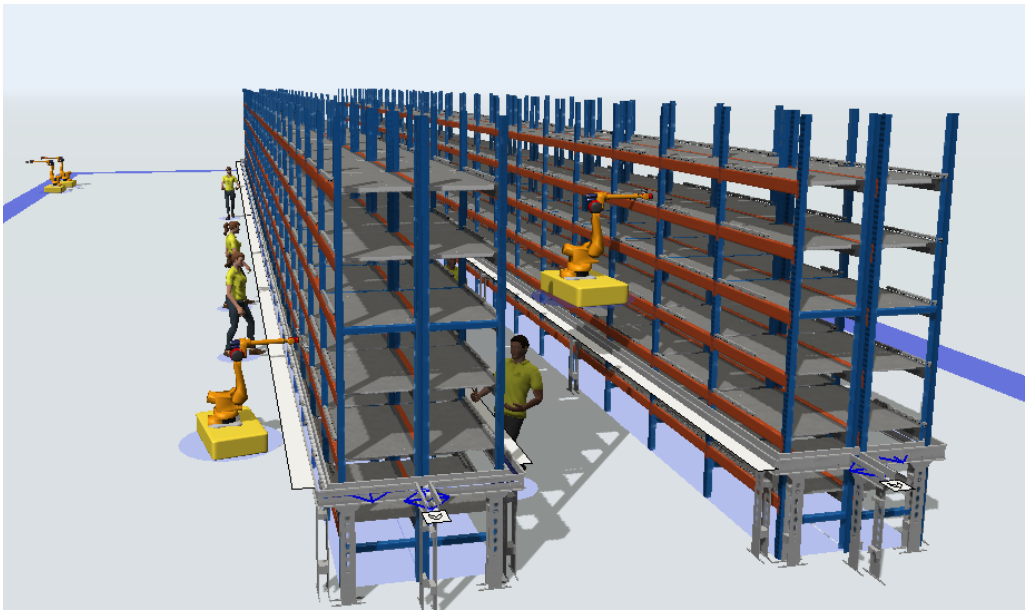


Figure 4.4: Simulation environment

adaptive thresholds introduced in Section 4.4.3.3.

- **Over-capacity:** At each time step, new human operators were added with a probability of 5%, with a maximum addition of six operators.

- **Under-capacity:** At each time step, the probability of human operator failure was 5%, with a maximum failure of six operators.

The initial number of CSs and the initial SoS architecture, representing the different levels of organization, are shown in Fig. 4.5. Additionally, separate tests were conducted to evaluate the scalability of the proposed MLSHG. Since computation time is primarily consumed during the reconfiguration stage, where resource and function reallocations are performed to satisfy the SoS threshold constraint, the focus is on comparing the capability-based reconfiguration proposed in this study with the backtracking method suggested by [Khalil 2012]. Khalil formulated the CSP as a SAT problem and listed several SAT solvers that could be used.

In this study, the method proposed in the algorithm is compared with the state-of-the-art SAT solver, CP-SAT-LP [Perron 2023], implemented using the Python library "OR-Tools," which has been recently applied to various optimization and constraint problems [Tsouros 2024]. Different scenarios, involving varying numbers of MCSs and PCSs, were tested to compare the computational time of both methods.

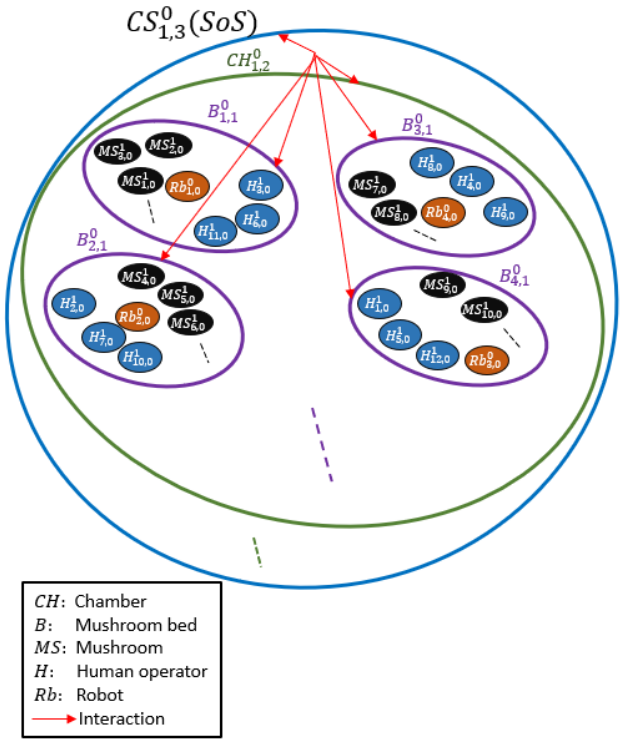


Figure 4.5: Mushroom harvesting hypergraph representation

4.4.5 Results and discussion

Afterward, the performance data of the mushroom harvesting SoS are collected, plotted and compared with the reference performance. It is important to note that although the global mission comprises three functions (growing, harvesting, and inspection), in this study, it is assumed that the growing and inspection functions are always maintained. Only the harvesting function is affected, as the fluctuations occur solely in the human operators who perform the harvesting tasks. Therefore, the plot and comparison focus exclusively on the performance related to the harvesting function.

In the over-capacity scenario with a constant threshold, four PCSs were added stochastically at various time steps: $H^1_{13,0}$, $H^1_{14,0}$, $H^1_{15,0}$, and $H^1_{16,0}$ at $t = 1$, 2, 4, and 6 hours, respectively, into MCSs $B^0_{2,1}$ (Bed 2), $B^0_{1,1}$ (Bed 1), $B^0_{4,1}$ (Bed 4), and $B^0_{3,1}$ (Bed 3), as shown in Fig. 4.6. For the first disturbance, the system did not initiate reconfiguration, as the SoS performance did not exceed the threshold. However, for the subsequent three disturbances, the system responded by eliminating the additional PCSs once the performance exceeded the threshold.

In the over-capacity scenario with an adaptive threshold, the same distur-

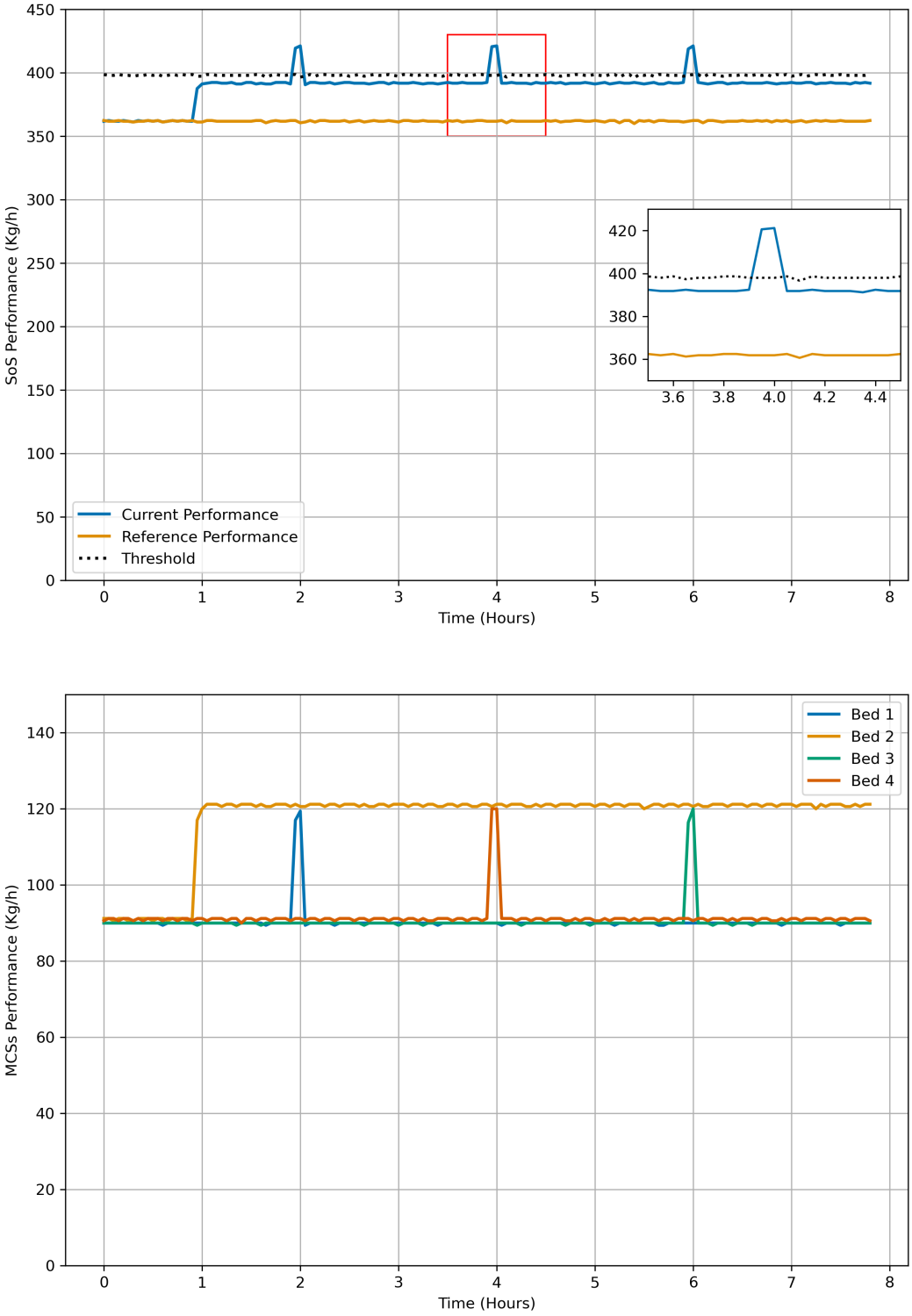


Figure 4.6: Performance of SoS and MCSs in the Over-Capacity Scenario with a Constant Threshold

bances were introduced using the same sampling seed. However, the system was able to eliminate the added CSs more effectively due to the greater sensitivity of the adaptive threshold, which facilitated earlier reconfiguration, as shown in Fig. 4.7. The impact of this adaptation was evident in the deviation from the final goal, measured by the total weight of harvested mushrooms. The deviation was 6.9% with a constant threshold, compared to just 1% with the adaptive threshold, as shown in Table 4.1.

Scenario	Harvested mushrooms (Tonnes)	Deviation
Reference (Optimal scenario)	2.87	0%
over-capacity adaptive threshold	2.9	1%
over-capacity constant threshold	3.09	6.9%
under-capacity adaptive threshold	2.65	7.6%
under-capacity constant threshold	2.61	9%

Table 4.1: Deviation from final goal for different scenarios

In the under-capacity scenario with a constant threshold, three failures occurred: $H^1_{7,0}$, $H^1_{3,0}$, and $H^1_{5,0}$ at $t = 1$, 3, and 5 hours, in MCSs $B^0_{2,1}$ (Bed 2), $B^0_{1,1}$ (Bed 1), and $B^0_{4,1}$ (Bed 4), respectively. The threshold was exceeded during the second failure, prompting the deployment of all available robots ($Rb^0_{1,0}$, $Rb^0_{2,0}$, $Rb^0_{3,0}$, and $Rb^0_{4,0}$) to compensate for the performance degradation in MCSs $B^0_{2,1}$ and $B^0_{1,1}$, as shown in Fig. 4.8. However, no reconfiguration was performed during the third failure since all available robots had already been deployed, which caused the SoS performance to decrease below the threshold.

In the under-capacity scenario with an adaptive threshold, the same failures were observed because the same seed was used. However, the SoS responded to the first failure by deploying two robots ($Rb^0_{1,0}$ and $Rb^0_{2,0}$) to the degraded MCS $B^0_{2,1}$ (Bed 2), and two robots ($Rb^0_{3,0}$ and $Rb^0_{4,0}$) to MCS $B^0_{1,1}$ (Bed 1) during the second failure, as shown in Fig. 4.9. No reconfiguration was performed during the third failure, as the system had already utilized all available resources. The deviation from the final goal, in terms of total harvested mushrooms, is summarized in Table 4.1 where the adaptive threshold scenario exhibited less degradation, with a 7.6% deviation compared to a 9% deviation in the constant threshold scenario.

These results demonstrate that using a constant threshold, as in [Khalil 2012], increases deviation from the final goal since stochastic failures are not considered. In contrast, incorporating stochastic disturbances into

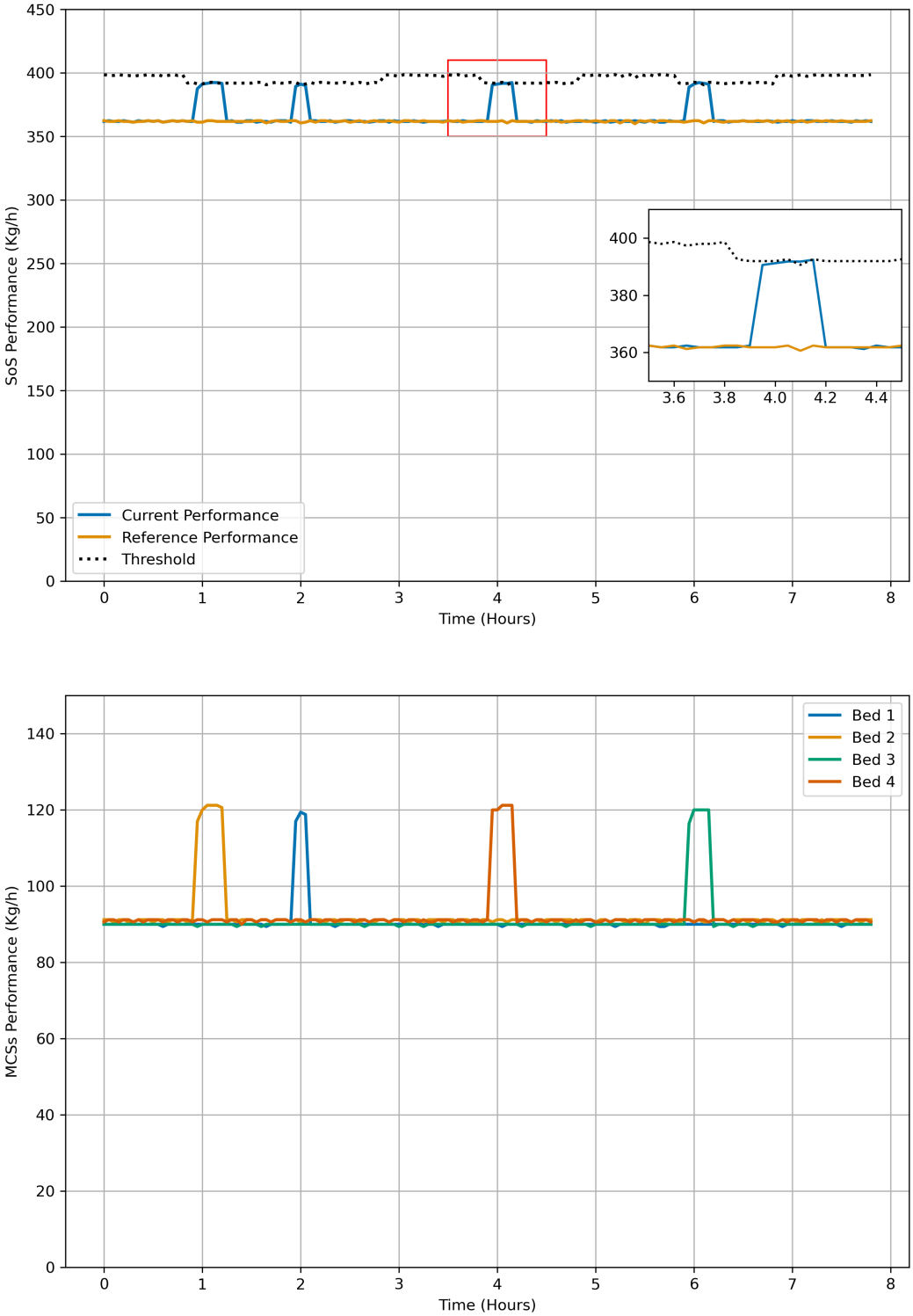


Figure 4.7: Performance of SoS and MCSs in the Over-Capacity Scenario with a adaptive Threshold

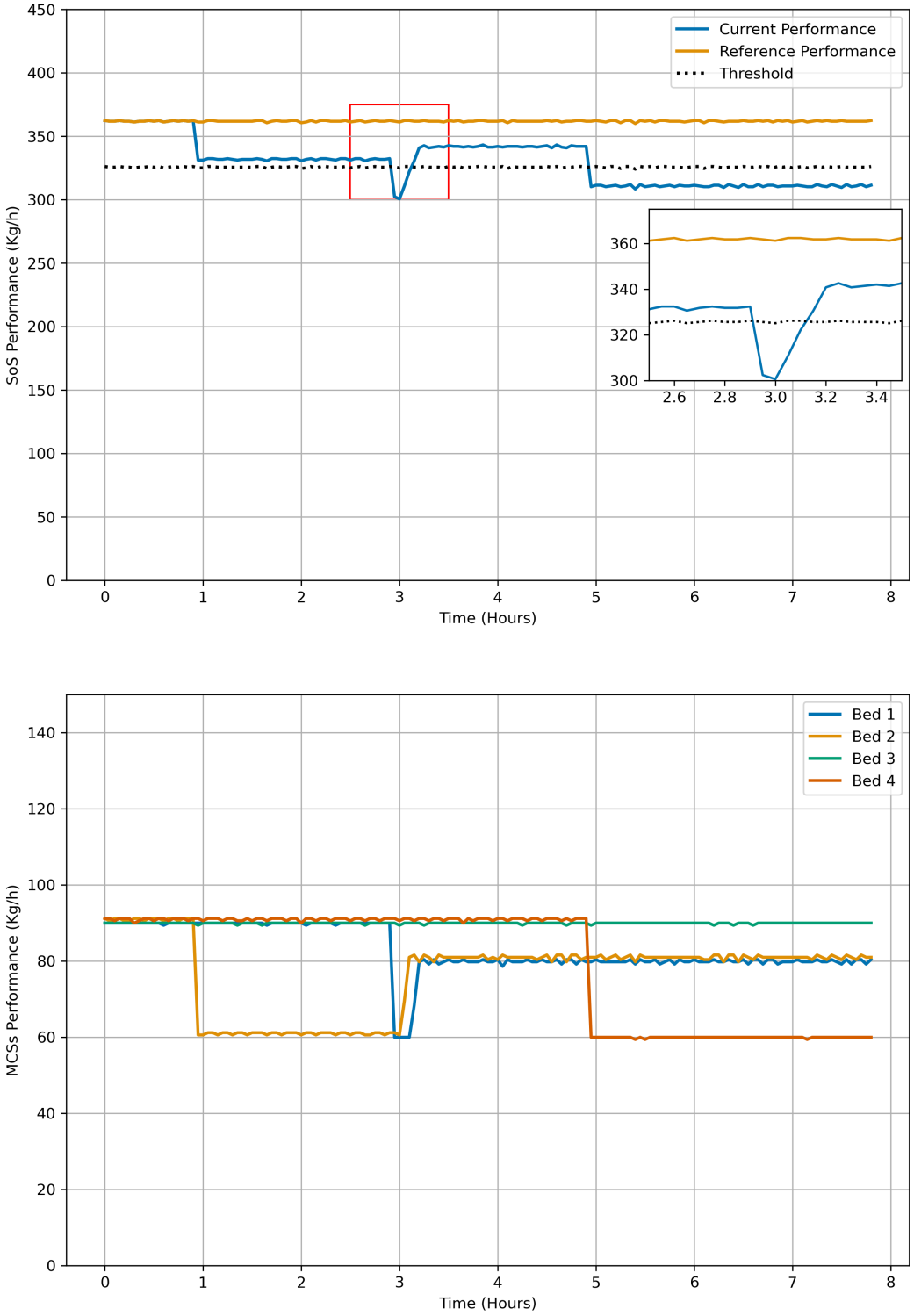


Figure 4.8: Performance of SoS and MCSs in the Under-Capacity Scenario with a Constant Threshold

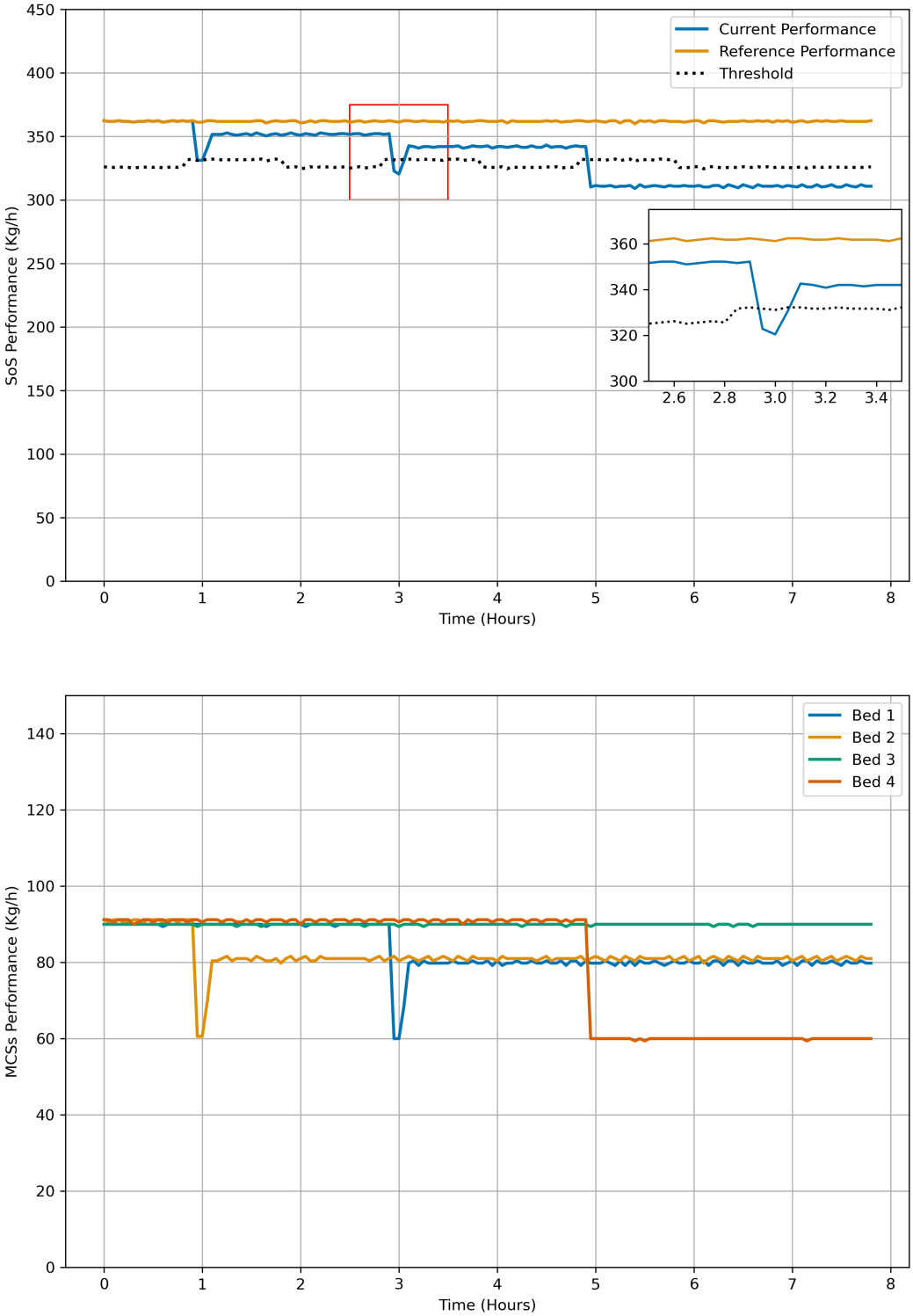


Figure 4.9: Performance of SoS and MCSs in the Under-Capacity Scenario with a adaptive Threshold

the adaptive threshold significantly reduces deviation from the final goal, enabling more robust and timely adaptation. Additionally, even if the drop in performance exceeds the threshold and no reconfiguration is implemented due to the lack of available resources, the final goal can still be achieved. This is because the goal is formulated as the integral of time-varying performance, and the timing of failures influences this deviation. Specifically, if failures occur near the end of the process, the deviation from the final goal will be less significant compared to failures that occur earlier in the process. The results from the FlexSim simulation were obtained for a single chamber. However, due to the high computational demands of this software and our limited computational resources, the remaining scenarios are implemented using a custom Python code to compare the computational time during the reconfiguration stage. The different scenarios are shown in Table 4.2, which includes varying numbers of MCSs (chambers and beds), PCSs (human operators and robots), different numbers of failures, and the average computational time for each method.

The graph in Fig. 4.10 shows that as the number of CSs increases, the computational time of the SAT solver used by [Khalil 2012] grows exponentially. This occurs because the method is based on backtracking algorithms, which attempt to find the best solution by checking all possible solutions. For example, with N MCSs and M PCSs, this requires checking M^N solutions, which is not suitable for large-scale SoS. On the other hand, the computational time for the capability-based reconfiguration used in the MLSHG model is significantly lower. This method checks the capabilities of the available CSs and assigns replacements for failed ones, resulting in computational time that scales linearly with the number of CSs.

Comparison of the average computational time between the SAT solver used in the HG model of [Khalil 2012] and the capability-based reconfiguration in the **MLSHG** model across different scenarios. Although the capability-based reconfiguration may not always provide the optimal solution that fully satisfies all mission constraints, it delivers an acceptable solution in a very short time. Additionally, it has been shown that the adaptive threshold, which accounts for stochastic disturbances and triggers early reconfiguration, helps minimize deviations.

4.5 Conclusion

In this chapter, the MLSHG framework is applied for **real-time supervision of SSoS**. The proposed methodology consists of an algorithm capable of **real-time monitoring** and **reconfiguration** using the defined attributes

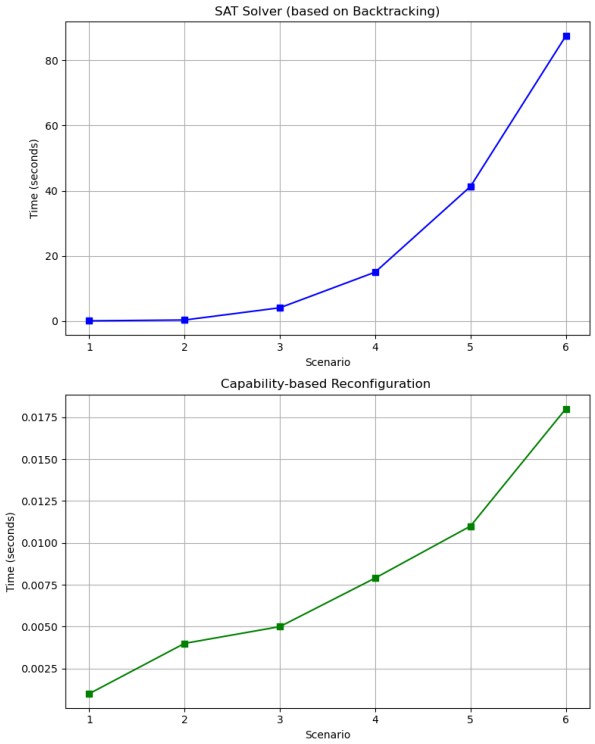


Figure 4.10: Comparison of the average computational time between the SAT solver used in the HG model of [Khalil 2012] and the capability-based reconfiguration in the **MLSHG** model across different scenarios.

associated with CSs in the MLSHG framework. This approach takes into account both **long-term goals** and **time-varying missions**.

In the monitoring stage, the objective is to measure the performance of all CSs, which helps evaluate the satisfaction of missions and detect failures. This process is conducted from the bottom up in the multi-level framework. In the reconfiguration stage, functions are decomposed from the top down, and the SoS performance is evaluated and compared with thresholds to determine whether the system is operating in a normal or abnormal mode. This approach aids in **capacity management** and raises alarms for reconfiguration in over- and under-capacity situations. If an over-capacity situation is detected, the system responds by removing extra CSs, while in an under-capacity situation, the system responds by reallocating functions and missions to other available CSs to compensate for degraded performance.

The proposed approach has been applied to the mushroom harvesting system, considered as an SSoS with heterogeneous CSs. The social and biological CSs were modeled stochastically to reflect uncertainty and stochastic disturbances in the SoS, while the robot was modeled as a deterministic CS to help compensate for and reconfigure the system. Various scenarios were tested, including under- and over-capacity situations with constant and adaptive thresholds.

The results showed that this approach effectively achieved long-term goals. As the number of CSs increased, the **computational time** of the proposed **capability-based reconfiguration** grew **linearly** and remained significantly lower compared to methods that seek the optimal solution. Although this method does not always provide the best solution for satisfying the mission, it demonstrated that early reconfiguration and accounting for stochastic disturbances through the **adaptive threshold** help **reduce deviations** from the final goal. While this framework was specifically applied to agriculture, it is suitable for other large-scale systems operating in uncertain environments, such as smart cities, supply chains, and military operations, due to its ability to handle stochasticity, scalability, and high complexity

To create the reference scenario, real data, expert insights from an actual farm, and some assumptions are used to simulate a comparable scenario involving a single chamber, similar to a real farm setup. However, for a more accurate and calibrated simulation, additional data would be required. Despite this, since our main objective was to compare performances, the developed approach was sufficient for evaluating the MLSHG framework and the supervision algorithm.

This supervision methodology enables the SSoS to recover; however, assessing the ability of the SSoS to recover will be addressed in the resilience study in the next chapter.

Scenario	MCSs (Chambers + beds)	PCSs (Human operators + robots)	Failures	SAT solver (Khalil et al. (2012) [Khalil 2012]): Avg. Time (seconds)	Capability-based reconfiguration (MLSHG): Avg. Time (seconds)
1	$1+4$	$10+4$	1	0.05	0.001
2	$2+8$	$15+4$	1	0.3	0.004
3	$3+12$	$26+4$	2	4.06	0.005
4	$4+16$	$36+6$	3	15	0.0079
5	$5+20$	$43+6$	4	41.3	0.011
6	$6+24$	$50+8$	5	87.5	0.018

Table 4.2: Different tested scenarios and the corresponding computational times obtained.

Chapter 5

Resilience of stochastic SoS

Contents

5.1 Introduction

In the previous chapter, the use of the MLSHG model for the supervision of SoS was explained, introducing a supervision methodology that consists of monitoring and reconfiguration. This methodology provides the essential tools that enable the system to adapt in the event of disturbances. The ability of the SoS to adapt and recover is what we define as resilience.

While some researchers focus on robustness [Davendralingam 2013], which is the ability to resist failures, others have suggested prioritizing resilience, as failures in SoS are unavoidable due to their complexity and emergent behavior [Madni 2009]. In this chapter, it is intended to build on the tools developed for supervision and introduce suitable recovery mechanisms to ensure the resilience of these complex systems.

Some resilience methodologies suggest including **anticipation** [Moradi 2018, Woods 2015]; however, this thesis focuses on SSoS, where

disturbances are inherently unpredictable. As a result, the emphasis shifts to **recovery and adaptation**. In highly dynamic and stochastic environments, anticipating every failure is impractical, making the system's ability to recover quickly and maintain its core operations even more crucial. Therefore, the following definition of resilient SSoS is proposed:

Definition 9 *The **resilience of a Stochastic System of Systems (SSoS)** is its ability to absorb, adapt, and recover from unpredictable failures, ensuring the continuation of normal operations, even with degraded performance.*

This resilience is achieved by proposing a methodology based on the presence of **redundancies** in the SoS and **functional adaptation**. To ensure comprehensive evaluation, this resilience should be quantified using suitable **metrics** that account for various resilience factors. In the following sections, we explain how the MLSHG model supports resilience strategies, introduce the proposed resilience algorithm and metric, and conclude by presenting the simulation results.

5.2 MLSHG for resilient SoS

The MLSHG model described in the modeling chapter is considered to support the resilience of SoS. The model, which captures the performance, mission, and capability of CSs, is well-suited for modeling **performance-based resilient** SoS, as the goal is to maintain desired performance. Since most recovery methods emphasize **redundancy** and **adaptability**, it is important to ensure that our model can incorporate these aspects. In this thesis, a distinction is made between **stand-by** and **stand-in redundancy**, as defined by [Uday 2014], with both being treated as dynamic properties that evolve over time. These redundancies, along with the adaptability aspects of our model, are detailed in the following sections.

5.2.1 Adaptability

It refers to the system's capacity to adjust its operations and structure to accommodate changes or disturbances. In SoS, it resembles **Maier's** [Maier 1998] fifth characteristic, **'Revolutionary and adaptive development,'** as an evolutionary perspective necessitates a system to be adaptable, capable of responding to unknown future conditions. In MLSHG, a SoS is deemed adaptable when its CS' functions change over time in response to

alterations in the global SoS goal or internal disturbances, for $CS^{j}_{n,0}$:

$$\forall j \in \{0,1\}, \exists t_1, t_2 \in \mathbb{R}, R^{t_1}_{n,0} \neq R^{t_2}_{n,0} \tag{5.1}$$

This implies that there are two points in time where the desired functions differ, indicating that the CS evolves over time.

5.2.2 Stand-in redundancy

It's also called **functional redundancy**, and it is noted that there are multiple ways to execute tasks and achieve functionalities. From the MLSHG perspective, for a given time t_1, the SoS has functional redundancy if at least one PCS has at least a quantified function $(q^{t_1}_i | R^{t_1}_{n,0})$ less than its capability functions $(q_i|Q_{n,0})$ where $(q_i|Y)$ means the quantifies value of function i in set Y :

$$\begin{aligned}(\forall j, j' \in \{0,1\})(\exists 1 \leq i \leq k)(((q_i|Q_{n,0}) \neq 0) \implies & \exists CS^{j}_{n,0} \in CS^{j'}_{n',l}, \\ & (q_i|Q_{n,0} > q^{t_1}_i|R^{t_1}_{n,0})),\end{aligned} \tag{5.2}$$

This formulation implies the existence of a PCS capable of performing certain functions at a higher level than its current performance. Therefore, it is **assumed** that the performance of all PCSs is **controllable**.

5.2.3 Stand-by redundancy

It refers to the presence of at least one **additional PCS** as a **backup CS** (not operating), capable of replacing the failing one due to the similarity in at least one function at a given time t_1:

$$\begin{aligned}((\forall q^{t_1}_i \in R^{t_1}_{n,0})(q^{t_1}_i = 0)) \wedge ((\exists 1 \leq j \leq k)((q^{t_1}_j|R^{t_1}_{n',0}) \neq 0 & \exists CS_{n,0}, CS_{n',0} \in V, \\ & \wedge (q_j|Q_{n,0}) \times (q_j|Q_{n',0}) \neq 0)\end{aligned} \tag{5.3}$$

5.3 Resilience algorithm

Based on the developed framework, an algorithm based on **Top-down reconfiguration** and **Bottom-up monitoring** is created for recovering from failures, it enables the model to adapt by considering the dynamics of both **stand-in** and **stand-by redundancies**. Given the costliness of operating backup systems, we **prioritize** recovery using functional redundancy over utilizing the backup systems. Fig. 5.1 illustrates the flowchart of the developed algorithm.

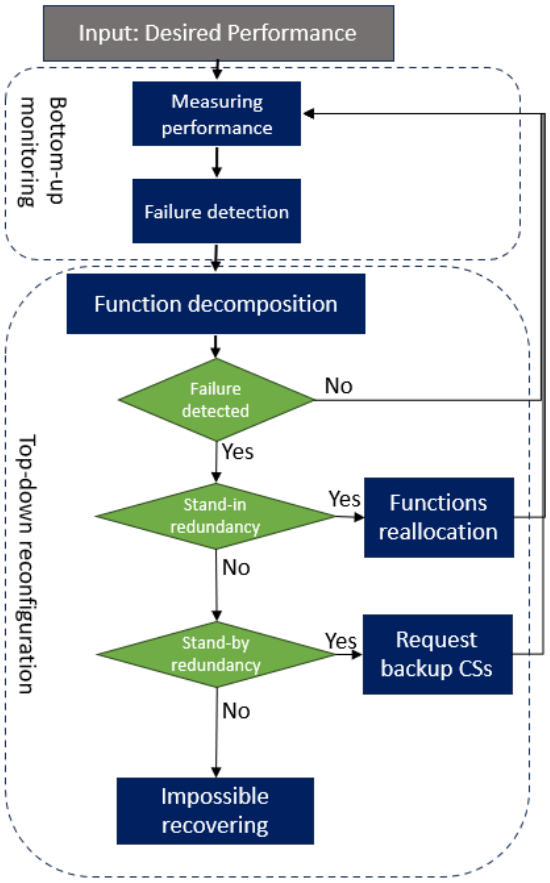


Figure 5.1: Flowchart of the resilience algorithm

The bottom-up monitoring is the same stage described in the supervision chapter, which includes performance measurement and failure detection. Failure detection is performed by generating the residual between the actual performance of the CSs and the desired performance, and then comparing this residual to a threshold. In top-down reconfiguration, the functions are decomposed similarly to the supervision algorithm. However, when failures are detected, it employ the stand-in and stand-by mechanisms for recovery. The following pseudo-algorithm (3,4) explains the different steps of the resilience algorithm.

The algorithm represents the different steps while accounting for b layers and a_l, the number of CSs in layer l. At each iteration, the first failed CS is detected, and both types of redundancy (stand-in and stand-by) are checked for PCSs to identify a functional or backup CS that can compensate for the failure, allowing the system to adapt to potential disturbances. The previously proposed definitions of stand-in and stand-by redundancies are applied, while

the remaining steps follow the same process as the supervision algorithm.

Algorithm 3 Resilience algorithm (Part 1)

///Declaration

$b \leftarrow levels$

$S \leftarrow \{a_l; (a_l \in \mathbf{N})(0 \leq l \leq b)\}$

//Input CSs sets and attributes

$V \leftarrow \{CS^j_{n,0}; 1 \leq n \leq a_0\}$

for $l \leftarrow b$ to 1 **do**

  $E_l \leftarrow \{CS^j_{n,l}; 1 \leq n \leq a_l\}$

//Define thresholds

$\lambda_{CS} \leftarrow CS_{threshold}$

$\lambda^t_{SoS} \leftarrow SoS_{threshold}$

//Initialize global functions

$R^t_{1,b} = \{q^t_1, ..., q^t_k\}$

$k \leftarrow |R^t_{1,b}|$

while True **do**

  ///Monitoring

  //Data performance collection from simulation

  $Pr \leftarrow \{P^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$

  $Dr \leftarrow \{D^t_{n,l}; 1 \leq n \leq a_l, 0 \leq l \leq b\}$

  //Detecting and eliminating failed PCSs

  $D \leftarrow \{d \in \mathbb{R} : (\exists 1 \leq n \leq a_0)(\exists 1 \leq j \leq k)(d = q^t_j | R^t_{n,0} - p^t_j | P^t_{n,0})\}$

  if $(\exists d \in D)(d > \lambda)$ **then**

    /Get the first failed PCS index

    $n' \leftarrow (\exists 1 < j < k)((q^t_j | R^t_{n',0} - p^t_j | P^t_{n',0}) > \lambda_{CS})$

    /Eliminate PCS with n' index from SoS

    $C^t_{n',0} \leftarrow 0$

    $A^t_{n',0} \leftarrow 0$

Algorithm 4 Resilience algorithm (Part 2)

///Reconfiguration

//Function decomposition

for $l \leftarrow (b-1)$ to 0 **do**

  $R^t_{n',l} \leftarrow \{x \in \mathbb{R} : (\exists 1 \leq j \leq (b-l))(\exists 1 \leq n \leq a_{l+j})(\exists 1 \leq i \leq k)(x = \sum_{j=1}^{b-l} \sum_{n=1}^{a_{l+j}} (f_{n,l+j,n',j}(q^t_i|R^t_{n,l+j}))\}(\forall 1 \leq n' \leq a_l)$

//Check SoS failure

if $|\sum_{s=1}^{k}(q^t_s|R^t_{1,b} - d^t_s|D^t_{1,b})| > \lambda_{SoS}$ **then**

  //Check Stand-in redundancies

  if $(\exists 1 \leq n \leq a_0)(\exists 1 \leq i \leq k)\ ((q_i|Q_{n,0}) \neq 0) \implies (q_i|Q_{n,0} > q^t_i|R^t_{n,0})$ **then**

    /Get the index of the most functionally redundant CS

    $B(x) = \sum_{i=1}^{k}(q_i|Q_{x,0}) - (q^t_i|R^t_{x,0})$

    $n'' \leftarrow (\forall 1 \leq n \leq a)(B(n'') > B(n))$

    /Reallocate failed CS functions to redundant CS

    $q_i|R^t_{n'',0} \leftarrow (q_i|R^t_{n'',0} + q_i|R^t_{n',0})(\forall 1 \leq i \leq k)$

  //Check Stand-by redundancies

  else if $(1 \leq n \leq a_0)(\forall q^t_i \in R^t_{n,0}) \quad (q^t_i = 0)) \wedge ((\exists 1 \leq j \leq k)((q^t_j|R^t_{n',0}) \neq 0 \wedge (q_j|Q_{n,0}) \times (q_j|Q_{n',0}) \neq 0)$ **then**

    /Get backup CS indexe

    $n''' \leftarrow (\forall q^t_i \in R^t_{n''',0})(q^t_i = 0) \wedge ((\exists 1 \leq j \leq k)((q^t_j|R^t_{n',0}) \neq 0 \wedge (q_j|Q_{n''',0}) \times (q_j|Q_{n',0}) \neq 0)$

    /Allocate failed CS functions to backup CS

    $R^t_{n''',0} \leftarrow R^t_{n',0}$

  else

    Print('Recovering is not possible')

5.4 Resilience quantification

To quantify resilience, **capability-based metrics** should be used to assess the system's ability to **absorb**, **adapt**, and **recover** from disturbances. These metrics must be suitable for the developed framework and aligned with the focus on performance-based resilience. They should encompass the different stages of performance resilience. In Fig. 5.2, the performance of a SoS is plotted, illustrating the pre-disturbance epoch, the absorption epoch, the recovery

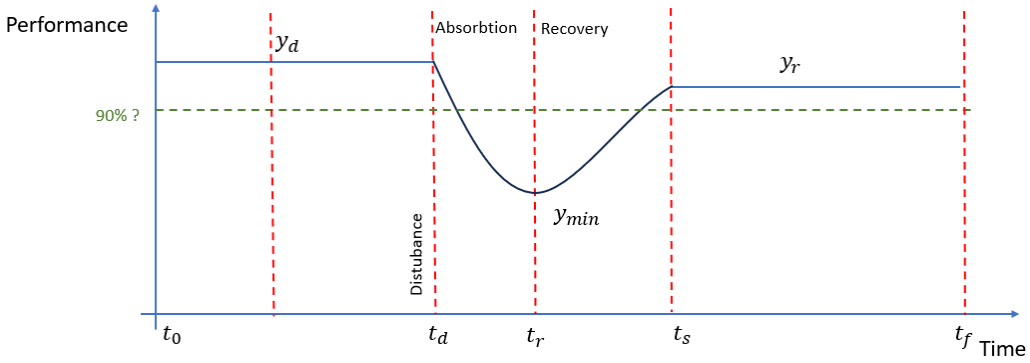


Figure 5.2: SoS performance over a period of interest for a single disturbance

epoch, and the steady-state epoch.

The resilience metric R, proposed by [Tran 2016], which includes performance, absorption, recovery, and recovery time factors, is adapted in our formalism to incorporate **redundancy** factors and is depicted as follows:

$$R = \rho\sigma(\delta + 1 - \tau^{(\rho-\delta)})\beta \tag{5.4}$$

where $0 \leq R \leq \infty$, and the terms in this equation are referred to as resilience factors. Each resilience factor captures an important aspect of a resilient system. The resilience factors are calculated as:

$$\sigma = \textbf{total performance factor} = \frac{\sum_{t_d}^{t_{\text{final}}} y(t)}{y_d(t_{\text{f}} - t_d)} \tag{5.5}$$

$$\delta = \textbf{absorption factor} = \frac{y_{\text{min}}}{y_d} \tag{5.6}$$

$$\rho = \textbf{recovery factor} = \frac{y_r}{y_d} \tag{5.7}$$

$$\tau = \textbf{recovery time factor} = \frac{t_{\text{S}} - t_d}{t_{\text{f}} - t_d} \tag{5.8}$$

$$\beta = \textbf{redundancy factor} = \frac{Re}{d_{SoS}} \tag{5.9}$$

where y_d is the desired system performance level, y_{min} is the minimum system performance level, y_r is the recovered performance level, t_d is the time of disturbance, t_{min} is the time at which the system reaches the minimum performance level, and t_{S} is the time when the system reaches steady-state performance. Re is the redundancy, encompassing both functional and backup capabilities in the SoS, and d_{SoS} is the performance of the SoS.

The performance, absorption, recovery, and recovery time factors are defined and detailed in [Tran 2016, Tran 2015], and can be summarized as follows:

- **Total performance factor:** This measures the total performance data over the period of interest.
- **Absorption factor:** This measures the system's ability to absorb the effects of a disruption.
- **Recovery factor:** This evaluates the system's ability to restore performance levels after a disruption.
- **Recovery time factor:** This measures the temporal aspects represented by the time required for the system to reach a steady state after a disruption.

The **redundancy factor** is added to represent the available capability relative to the desired performance of the System of Systems (SoS) mission. This factor captures the SoS's available capacity to compensate for degraded performance during operation.

The Re is the redundancy, encompassing both functional and backup capabilities in the SoS, which is defined as the difference between the total SoS capability and its current performance:

$$Re = \sum_{i=1}^{k} \left(\left(q_i^t \mid Q_{SoS} \right) - \left(d_i^t \mid D_{SoS}^t \right) \right) \tag{5.10}$$

Here, $q_i^t \mid Q_{SoS}$ represents the capability of function i in the SoS, where Q_{SoS} is the set of capabilities of all functions in the SoS. Similarly, $d_i^t \mid D_{SoS}^t$ denotes the actual performance of function i, and D_{SoS}^t is the set of performances of all functions in the SoS. The total number of functions is denoted by k.

The performance of the SoS, d_{SoS}, is given by:

$$d_{SoS} = \sum_{i=1}^{k} \left(d_i^t \mid D_{SoS}^t \right) \tag{5.11}$$

It should be noted that, this resilience metric, it ignores the volatility factor by assuming that the SoS possesses sufficient redundancy to enable recovery.

Similar to [Tran 2016, Tran 2015] and Considering multiple failures, a total resilience metric, R_{total}, is calculated from individual R values as:

$$R_{\text{total}} = \frac{\sum_{i=1}^{N_T} w_i R_i}{\sum_{i=1}^{N_T} w_i} \tag{5.12}$$

where N_T is the number of failures that require recovering (not all failures require recovering) and the weights w_i are coefficients of an exponentially weighted moving average, defined as:

$$w_i = (1 - \alpha)^{N_T - i} \tag{5.13}$$

with a smoothing factor $\alpha = 0.06$.

5.5 Case study

The **mushroom harvesting SoS** is used as a case study to apply our resilience methodology through the MLSHG model. The hierarchical structure of this system is detailed in the modeling chapter, representing the different levels and the physical subsystems (**PCSs: mushrooms, robots, human operators**) as well as the managerial subsystems (**MCSs: chambers and beds**) and their interactions. The scenario considered is adapted to reflect the resilience methodology, incorporating stand-in and stand-by redundancy mechanisms. The following sections explain the **importance of performance-based resilience** in mushroom farming, particularly in harvesting operations, and present the **simulation results along with a discussion**.

5.5.1 Resilient mushroom farm

Recent findings show that the current resilience of European farming systems is primarily focused on maintaining performance, but they lack the necessary adaptability for long-term resilience [Meuwissen 2020]. This highlights the importance of providing frameworks to shift toward resilience and recovery, as well as assessing resilience to evaluate the effectiveness of such methodologies [Meuwissen 2019].

Similar to other food services, the mushroom harvesting SoS is labor-intensive and faces sudden labor shortages, which can degrade performance [Karan 2021]. This sudden labor shortage can be modeled using stochastic methods, such as a binomial process, similar to the approach discussed in the supervision chapter.

In farming systems, **resilience practices** include labor reallocation through adaptability and the implementation of new technologies

[Manevska-Tasevska 2021]. In mushroom harvesting, the focus is on recovery by applying stand-in and stand-by redundancy mechanisms, **reallocating the functions of human operators**, **increasing their harvesting performance**, and **incorporating robots (backups)** when necessary. This adaptability will enhance the sustainability of these farming systems [Darnhofer 2010].

5.5.2 Results and discussion

The resilience scenario consists of a single chamber with four mushroom beds, similar to Fig. 4.4, involving 12 human operators with varying performance levels and 4 robots as backups. The simulation is run for a single shift (8 hours on the first harvesting day). The drop in performance of the human operators is modeled stochastically using a binomial distribution, similar to the supervision scenario. The resilience algorithm, which incorporates stand-by and stand-in redundancy mechanisms, is implemented using the logic flow tool in FlexSim.

The results of the simulation are plotted in Fig. 5.3, showing the performance of the MCSs (mushroom beds) and the overall SoS performance. Three failures occurred, leading to a drop in SoS performance. For **the first failure**, the system **did not respond** as the performance drop was within the threshold. For **the second** and **third failures**, the system incorporated **redundancy mechanisms to recover**. To better analyze these results, the simulation time is decomposed into six epochs, each representing a stage of the resilience scenario, as shown in the figure.

For the first failure, the system eliminated human operator 7 $H^1_{7,0}$ from mushroom bed 2 $B^0_{2,1}$, resulting in a performance drop in epoch 2. However, no response was required since the performance drop remained within the threshold, as mentioned previously.

For the second failure, the system eliminated human operator 3 $H^1_{3,0}$ from mushroom bed 1 $B^0_{1,1}$, which led to a performance drop in epoch 3. The system was able to recover by activating the stand-in redundancy mechanism. This involved increasing the performance of human operators 6 $H^1_{6,0}$ and 12 $H^1_{12,0}$ to their maximum capacity, which was 30 kg/h. The system thus successfully recovered from the failure.

For the third failure, human operator 4 $H^1_{4,0}$ was eliminated from mushroom bed 1 $B^0_{1,1}$ in epoch 5, leading to another drop in performance. The system employed both stand-in and stand-by redundancy mechanisms to recover. Initially, it increased the performance of human operator 11 $H^1_{11,0}$ to the maximum level. Since no functional redundancy was left, the robot backups $Rb^0_{1,0}, Rb^0_{2,0}, Rb^0_{3,0}, Rb^0_{4,0}$ were incorporated to aid in recovery during epoch

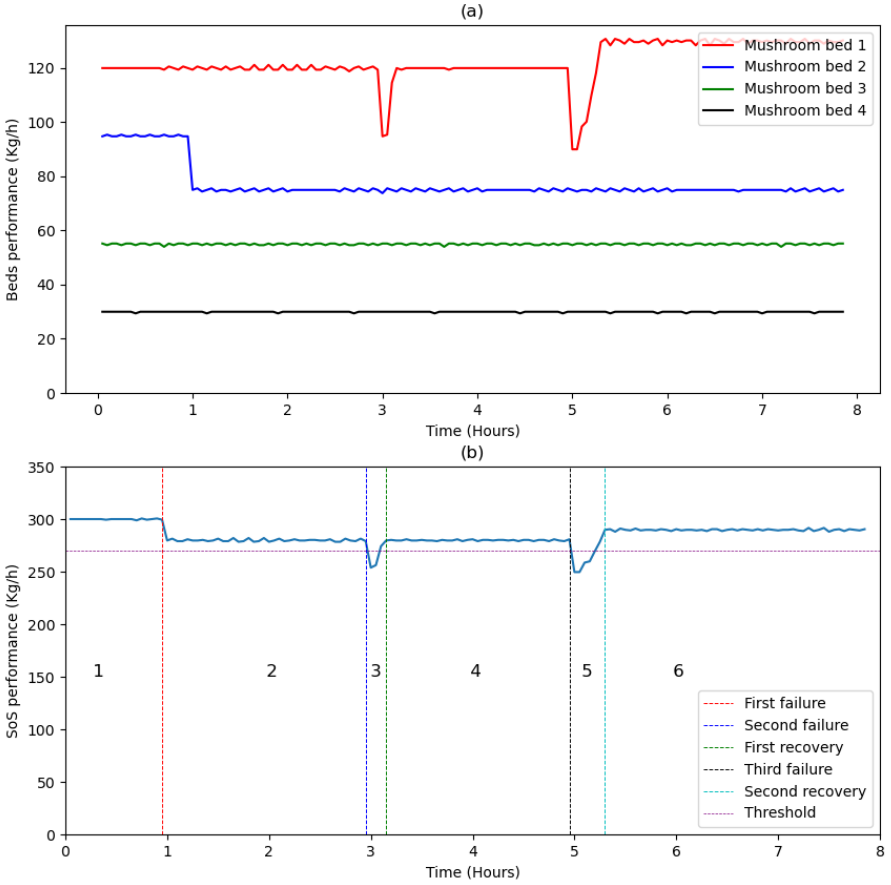


Figure 5.3: The performance of (a) mushroom beds and (b) SoS as a function of time for a multiple failure

5.

The changes in the performance of human operators are highlighted in red in Table 5.1, representing both the elimination (performance = 0) and the increase in performance when the stand-in redundancy recovery mechanism is activated. It should be noted that this is the average over a period of time, and the actual performance exhibits some uncertainty due to the stochastic nature of mushroom growth. Additionally, in this simulation, it is assumed that the performance of human operators can be controlled, whereas in real-life scenarios, this is more challenging.

Epoch	1	2	3	4	5	6
Operator 1	30	30	30	30	30	30
Operator 2	25	25	25	25	25	25
Operator 3	25	25	0	0	0	0
Operator 4	20	20	20	20	0	0
Operator 5	30	30	30	30	30	30
Operator 6	20	20	30	30	30	30
Operator 7	20	0	0	0	0	0
Operator 8	30	30	30	30	30	30
Operator 9	25	25	25	25	25	25
Operator 10	30	30	30	30	30	30
Operator 11	25	25	25	25	30	30
Operator 12	20	20	30	30	30	30

Table 5.1: Average mission performance (Kg/h) by operators

Moreover, the initial **HG representation**, along with its **evolution**, is shown in Fig. 5.4, illustrating the elimination of failed PCSs in epochs 2, 3, and 5, as well as the reallocation of backup PCSs in epoch 6. This evolution demonstrates the ability of the SoS to reallocate PCSs to MCSs, proving the adaptability of the proposed model.

To evaluate the proposed model and **quantify the resilience** of the mushroom harvesting SoS, the resilience metric proposed in Section 5.4 is calculated. This metric includes absorption, recovery time, redundancy, and performance factors for the first recovery (epoch 3), which incorporates the stand-in redundancy mechanism, and the second recovery (epoch 5), which incorporates both stand-in and stand-by redundancy mechanisms. The results are shown in Fig. 5.2.

For the **absorption factor**, both epochs exhibit similar values. However, for the **recovery factor**, the second recovery has a higher value, as backup CSs are incorporated to increase the SoS performance. Regarding **recovery**

time, the first recovery has a lower value than the second, since employing functional adaptation is faster than using both functional and backup redundancies. The **performance factor** is higher in the second recovery, demonstrating that using both stand-in and stand-by mechanisms improves performance. Finally, for the **redundancy factor**, the first recovery shows a higher value due to greater redundancy, while the second recovery, with fewer available redundancies, has a lower value.

This comparison highlights the differences between multiple failures; however, the main focus is on the overall resilience of the SoS over time. To quantify this, the **total resilience metric** is calculated as **0.2317**, using the proposed equation from Section 5.4. A higher value of this metric indicates greater resilience in the SoS. By utilizing this metric, the question of which system, A or B, is more resilient can be answered.

	First recovery (Stand-in redundancy)	**Second recovery** (Stand-in and stand-by redundancy)
Absorption factor δ	0.9097	0.8889
Recovery factor ρ	1.0022	1.0321
Time factor τ	0.4	0.7
Redundancy factor β	0.3333	0.2667
Performance factor σ	0.7607	0.8239
R_i	0.2517	0.2139
W_i	0.94	1
R_{total}	0.2317	

Table 5.2: Performance metrics for first and second recovery

5.6 Conclusion

In this chapter, the **MLSHG model** is applied and explained in the context of achieving **resilience in SSoS**. While the model demonstrates its capability for modeling and supervising SoS, it also shows support for **adaptability** and **redundancy** properties. Additionally, a resilience algorithm was proposed, consisting of bottom-up monitoring and top-down reconfiguration. While the monitoring process is similar to the supervision algorithm introduced in the previous chapter, the reconfiguration relies on stand-in and stand-by redundancy mechanisms. This algorithm prioritizes stand-in (functional) redundancy over backups, as incorporating additional components increases operating costs.

To quantify the performance-based resilience scenario and evaluate the proposed methodology, a **resilience metric** is proposed that incorporates total performance, absorption, recovery, time, and redundancy factors, accounting for multiple failures. In the FlexSim simulation, this resilience scenario is applied to the mushroom harvesting SoS, which experiences stochastic failures of human operators. The results demonstrated strong recovery capabilities through redundancy mechanisms, enabling quick recovery after failures. It is important to note that the resilience of the SSoS is highly dependent on redundancy (both functional and backup). While increasing redundancy makes the SSoS more resilient, the question remains regarding the additional cost required to implement more redundancies.

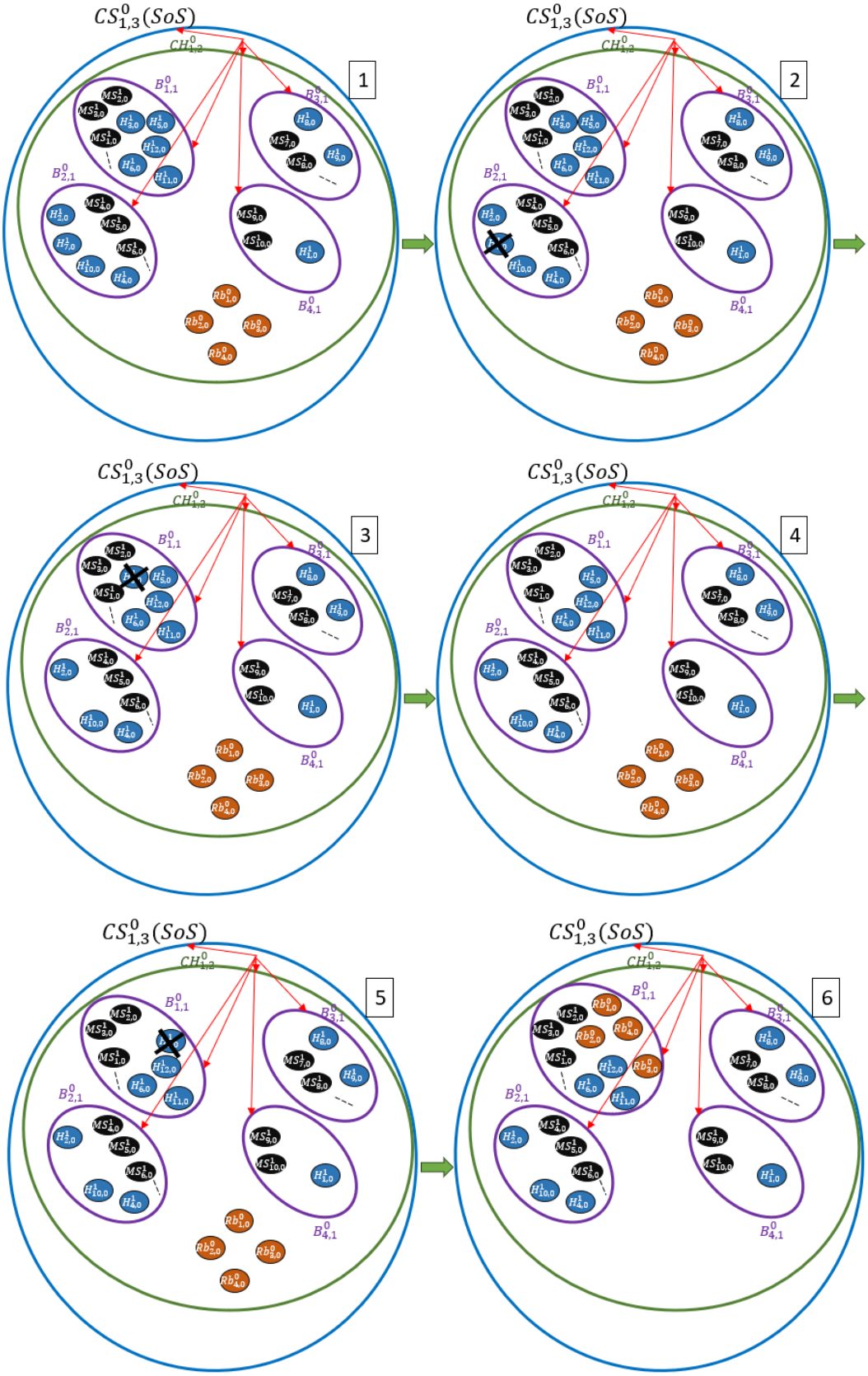


Figure 5.4: Hpergraph evolution of the resilience scenario for different epochs

Chapter 6

Conclusion

Contents

6.1 General conclusions

This thesis aims to contribute to the **modeling and supervision of System of Systems (SoS)**. The current literature highlights several key challenges in modeling SoS, including describing the heterogeneity of functions and the dynamic performance of Component Systems (CSs), understanding the complex interactions between these CSs and their structure, as well as accounting for stochastic CSs that may unpredictably join or leave the SoS. Existing works reveal the lack of a comprehensive framework that addresses these challenges simultaneously. To overcome these issues, the Multi-level Stochastic Hypergraph (MLSHG) model was presented in Chapter 3, with its key benefits summarized as follows:

- The structure of the SoS is described using multi-level modeling to manage complexity. At level 0, the Physical Component Systems (PCSs) are represented as vertices in the Hypergraph (HG). At higher levels, the Managerial Component Systems (MCSs) are represented as hyperedges in the HG, encompassing groups of PCSs.

- Each of these CSs is assigned attributes that describe its performance, functions, capabilities, stochastic existence, and mission satisfaction. This formalism allows the representation of heterogeneous CSs with varying functions and capabilities, while also distinguishing between

stochastic and deterministic CSs. For stochastic CSs, the stochastic existence attribute can be modeled using probabilities and distributions.

- The interactions between CSs are described using edge-dependent weights that account for their stochastic existence, information exchange, and sub-functions within the hyperedges.
- The proposed model is validated to ensure it adheres to the key properties outlined by [Maier 1998], which distinguish SoS from traditional complex systems.

While only a few studies have explored the **supervision** of this type of SoS, in Chapter 4, the supervision methodology is proposed by introducing an algorithm that integrates real-time monitoring and reconfiguration. During the monitoring stage, the objective is to assess the performance of all CSs, which helps evaluate mission satisfaction and detect failures. This process is conducted bottom-up within the multi-level framework. In the reconfiguration stage, functions are decomposed top-down, and SoS performance is evaluated against predefined thresholds to determine whether the system is operating normally or abnormally. This approach supports capacity management and triggers alarms for reconfiguration in both over-capacity and under-capacity scenarios.

In over-capacity situations, the system responds by removing excess CSs, while in under-capacity scenarios, it reallocates functions and missions to other available CSs to compensate for degraded performance. The results demonstrated that this approach effectively achieves long-term goals. As the number of CSs increases, the computational time of the proposed capability-based reconfiguration grows linearly and remains significantly low (≈ 0.02 s) compared to methods that seek optimal solutions, thereby enhancing scalability. Although this approach does not always guarantee the optimal solution for mission satisfaction, it shows that accounting for stochastic disturbances through an adaptive threshold allows for early reconfiguration and helps minimize deviations from the final goal.

This need for adaptability leads to the concept of **resilience**, defined as the ability to maintain normal operations and desired performance. While the supervision methodology laid the foundation for monitoring and reconfiguration, Chapter 5 expands on this by focusing on resilience strategies. Researchers have explored performance-based resilience through methods aimed at recovery after failures. In this context, key recovery strategies are introduced, such as stand-in and stand-by redundancy mechanisms, with a focus on prioritizing stand-in (functional) redundancy over backups, as adding extra components increases operational costs. These strategies require flexibility

and redundancy, which are inherently supported by the proposed MLSHG model.

To assess the effectiveness of these resilience strategies, a resilience metric was adapted by incorporating the redundancy factor alongside other factors such as performance, absorption, recovery, and recovery time. This metric evaluates the system's ability to recover from multiple failures and facilitates the comparison of resilience across different systems.

The practical application of these concepts is demonstrated in the case study on mushroom harvesting. Recent mushroom farming activities have highlighted the physically demanding and repetitive nature of harvesting tasks, leading to staffing difficulties and reduced productivity due to worker fatigue. By incorporating robots to assist human workers, the case study shows how effectively managing both human and robotic resources can improve productivity and meet yield goals, reinforcing the value of the proposed approach.

To model the mushroom farming system as a large-scale SoS, the MLSHG model is applied. The system consists of social components (human operators), biological components (mushrooms), and mechatronic systems (robots), all working together to achieve the desired yield. The MLSHG model captures the functions, performance, capabilities, and interactions of these components. To further manage complexity and enhance resource allocation, additional managerial components like mushroom beds and chambers were introduced into the model. While the simulation closely mirrors real-world scenarios, further calibration, particularly for mushroom growth modeling, is required. Nevertheless, this approach provides valuable insights to farmers in the early design stages, allowing them to evaluate and compare various resource allocation strategies and production outcomes, thereby supporting more informed decision-making

When applying the supervision algorithm to this SoS, the desired yield goals were achieved by detecting disturbances in human operators and adjusting performance accordingly. The final deviation from the goal was reduced using an adaptive threshold (from 9% to 7.6% in under-capacity scenarios and from 6.9% to 1% in over-capacity scenarios). Moreover, the resilience algorithm enabled us to explore various redundancy mechanisms, such as introducing additional robots as backups or increasing the capacity of human operators to support recovery. The system's resilience was assessed using the proposed metric, and the results showed that incorporating backups improved the performance factor, although the recovery time factor increased. It can be concluded that increasing redundancy improves the resilience of the mushroom harvesting SoS. However, the cost implications of implementing these redundancies remain an important consideration for future work.

6.2 Perspectives

6.2.1 Behavioral modeling

The proposed framework focuses primarily on the structural and organizational aspects of SoS, addressing many of the limitations identified in the literature. However, the dynamic behavior of Physical Component Systems (PCSs) is not explicitly modeled. This is a significant omission in the context of SoS, as understanding the dynamics of PCSs is essential for identifying and diagnosing the root causes of failures and performance degradation. In addition, the framework assumes that Managerial Component Systems (MCSs) are always observable by PCSs and that the links between them remain intact, which is not always the case in real-world scenarios.

By explicitly modeling the behavior of PCSs, it becomes possible to study failures in the interactions and links between CSs and to develop solutions for restoring functionality. For example, modeling the dynamics of PCSs using state-space models can help simulate and analyze their performance over time. Furthermore, by designing distributed observers, the observability of MCSs can be maintained, even in cases where communication links between CSs fail. This approach would allow the system to continue functioning under degraded conditions by reallocating tasks or reconfiguring its structure based on the observed state of the remaining components.

Incorporating behavioral modeling, such as state-space models, into the framework would significantly enhance its robustness, providing a more comprehensive solution for managing failures and ensuring continued performance within the SoS. It would also address critical issues, such as the failure of communication links and degraded interactions between CSs, offering practical solutions for maintaining system resilience and operational efficiency in complex environments. This is especially crucial when designing the SoS from scratch. However, this also brings additional challenges, such as shifting the modeling approach from a monolithic system to an SoS, while still ensuring that the SoS respects the key properties proposed by [Maier 1998].

6.2.2 Supervision and resilience

The reconfiguration algorithms proposed for resilience and supervision are heuristic-based, and while they provide good solutions in a short time, they do not guarantee the best possible solution. In more complex scenarios, where many functions must be considered within the SoS, these heuristic-based algorithms may not be sufficient. To address this, optimization methods with low computational complexity that can handle multi-objective optimization

functions should be further investigated, especially those that can account for stochastic failures. Methods such as Genetic Algorithms (GA), Markov Decision Processes (MDP) (reinforcement learning), and Particle Swarm Optimization (PSO) are promising candidates for future exploration. These methods can efficiently explore large solution spaces while balancing computational time and solution quality.

6.2.3 Mushroom harvesting simulation

The simulation of mushroom harvesting is based on certain assumptions, particularly when representing the growth of mushrooms. For a more accurate simulation, it would be beneficial to precisely model the growth process. While traditional methods rely on mathematical models, more recent approaches are primarily data-driven. In the future, AI models could be proposed for modeling mushroom growth. For instance, data in the form of images taken by cameras placed throughout the farm could be collected over time. With the application of appropriate AI models, this data could be analyzed to track the growth of mushrooms, enabling the development of more accurate growth functions.

Appendices

Appendix A

Flexsim simulation

The simulation in FlexSim consists of two branches: the 3D model and Process Flow. The 3D model is used to visualize the various components of the simulation, while the Process Flow defines the functionalities of each component.

A.1 3D Model

Fig. A.1 shows the 3D model that visualizes the different components of the mushroom harvesting SoS. The dimensions used in the simulation (specifically for the mushroom beds, see Fig. A.2) are based on the dimensions from "Ferme de la Gontière."

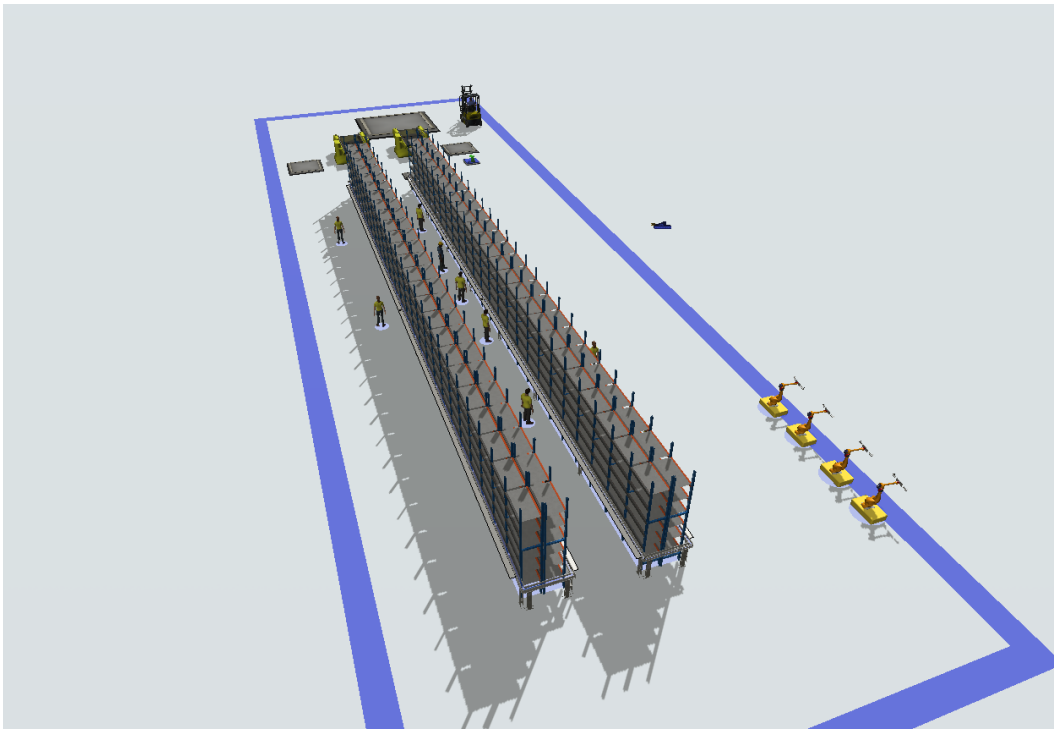


Figure A.1: 3D model.

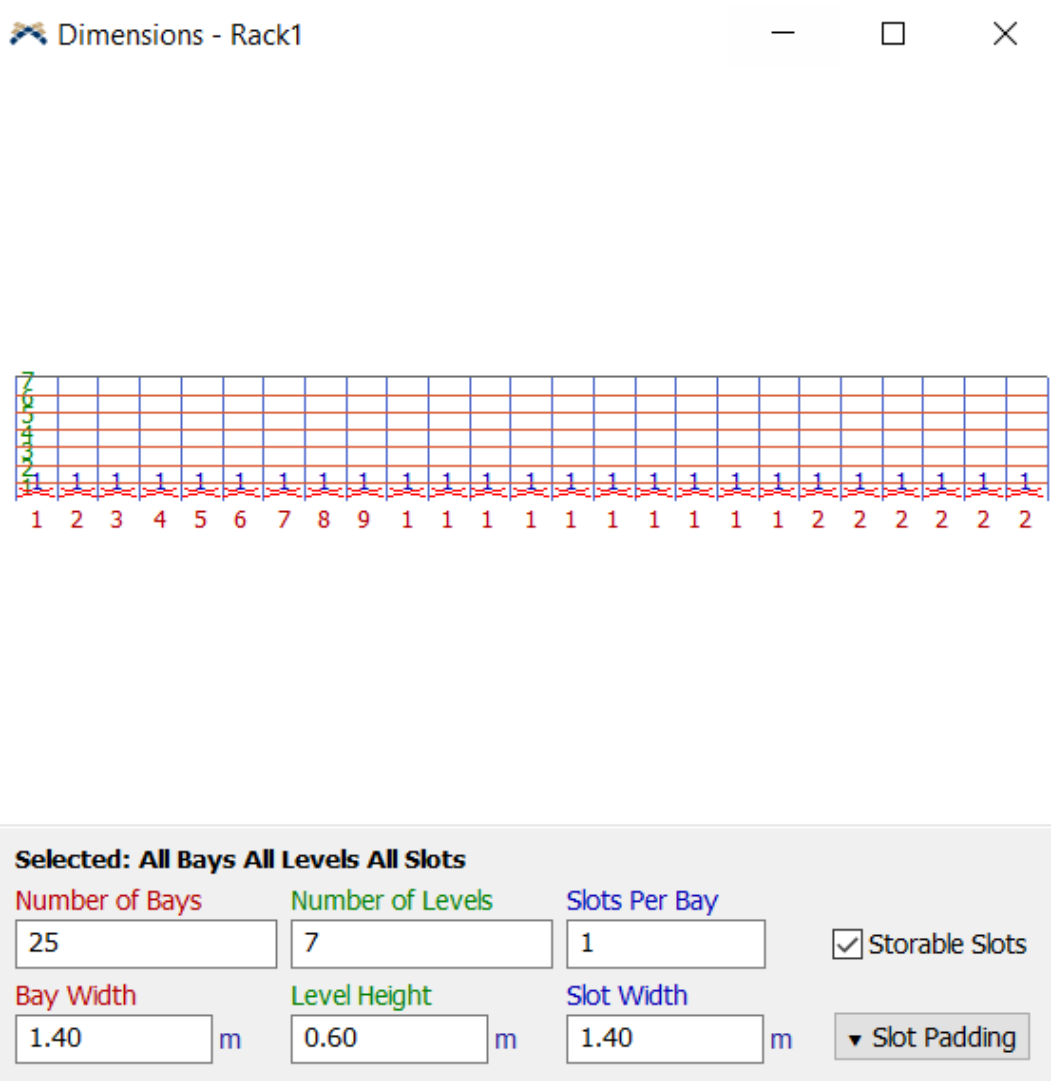


Figure A.2: Mushroom bed (Rack) dimension

Regarding the develloped attributes of the MLSHG model its implemented using attributes assigned as labels for each component, for example Fig. A.3 show the properties of human operator 8 that include the labels associated.

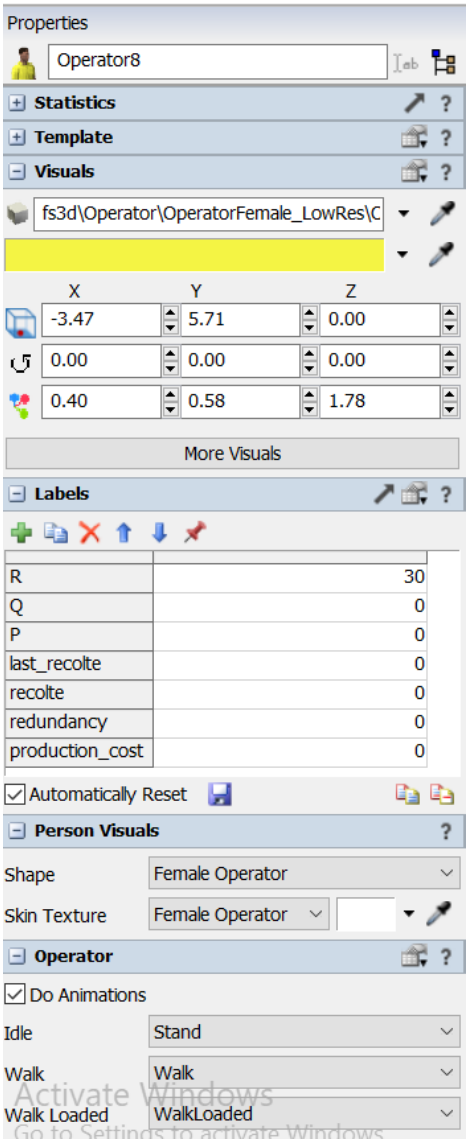


Figure A.3: Properties of human operator

The human operators work in 8-hour shifts during the day, 5 days a week. This schedule is implemented using the Time Table feature in FlexSim, as shown in Fig. A.4.

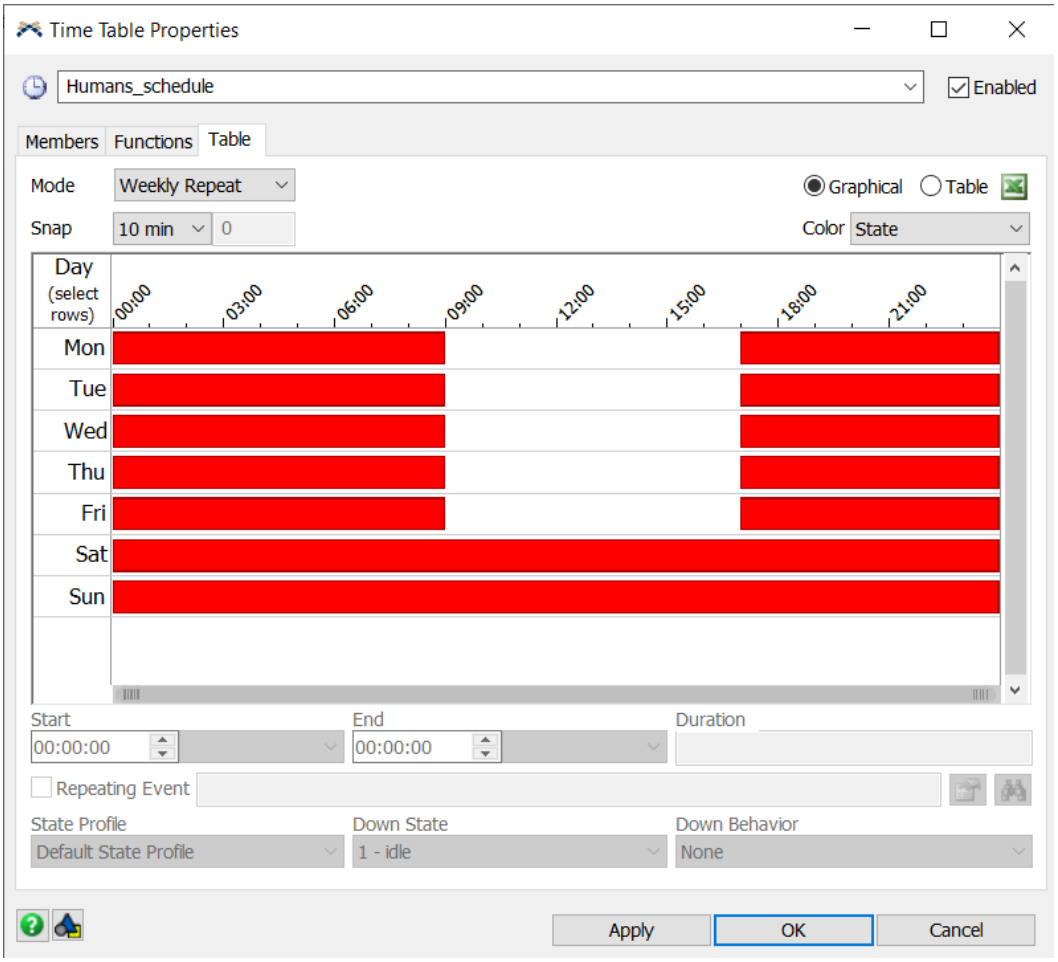


Figure A.4: Human operators scheduling

A.2 Process Flow

Using the process flow tool in FlexSim, we coded the functionalities of the different stages of the simulation. The first stage involves the generation of mushrooms, where mushrooms are considered as stochastic CSs and are generated using a binomial distribution. Fig. A.5 illustrates the process flow for generating mushrooms in different mushroom beds (racks). In the custom code, we implemented sampling from the binomial distribution, and the "create object" function is used to randomly generate mushrooms within the racks.

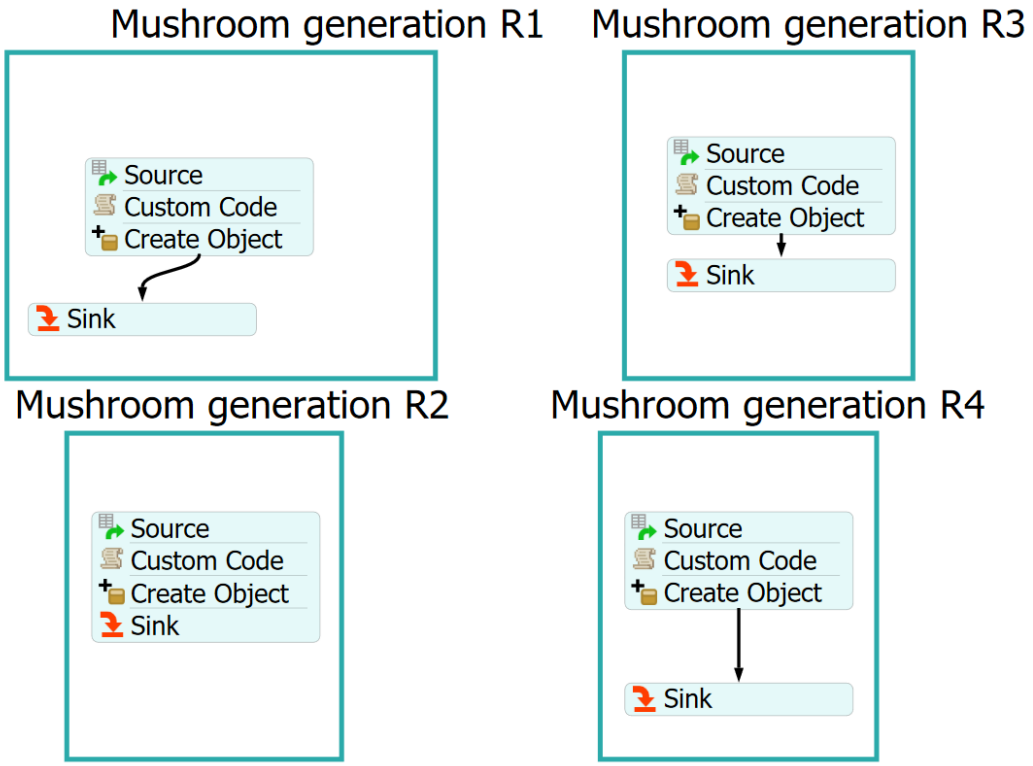


Figure A.5: Mushroom generation.

The mushrooms generated over time are considered mature and ready for harvesting. In real-life scenarios, these mushrooms are inspected before the harvesting process begins. In our modeling and simulation, we assume that robots are equipped with cameras to inspect the mushrooms. To represent this functionality, we store the generated mushrooms in a list that is accessible to different components. The Process Flow related to this inspection is illustrated in Fig. A.6.

After storing the mushrooms in a list, the functions are decomposed within

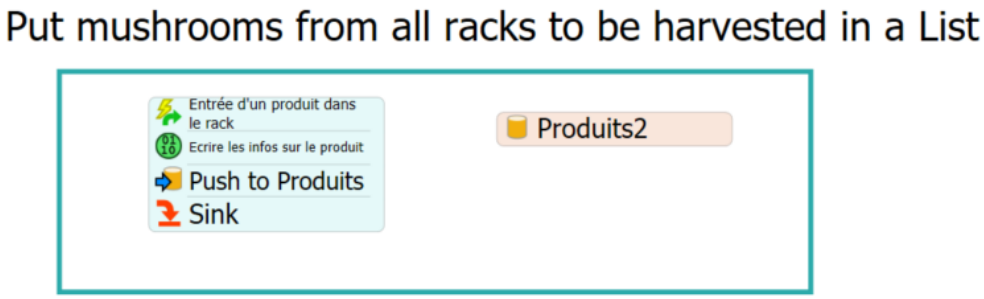


Figure A.6: Mushroom inspection.

the SoS, and human operators receive their assigned tasks and missions, primarily focusing on harvesting specific mushroom beds. Each operator performs their harvesting tasks based on their individual perspectives. To represent this, the heuristics of the human operators are coded using the process flow tool, as shown in Fig. A.7, ensuring that each operator functions independently, both operationally and managerially.

Similarly, robots have their own heuristics developed using the process flow tool, as shown in Fig. A.8. These heuristics focus primarily on harvesting a single level within the bed, slot by slot, starting either from left to right or right to left, and then moving to the next level once all mature mushrooms in the current level have been harvested.

Since human operators are considered stochastic CSs, their performance may degrade, leading to potential failure and their elimination from the SoS. This degradation is modeled using the process flow tool, with a binomial distribution implemented in custom code, as shown in Fig. A.9.

For the supervision algorithm, Fig. A.10 illustrates the monitoring stage, which is conducted bottom-up by measuring the performance of PCSs, propagating the information to higher levels, and detecting disturbances. The reconfiguration stage, conducted top-down, allows for function decomposition and reallocation.

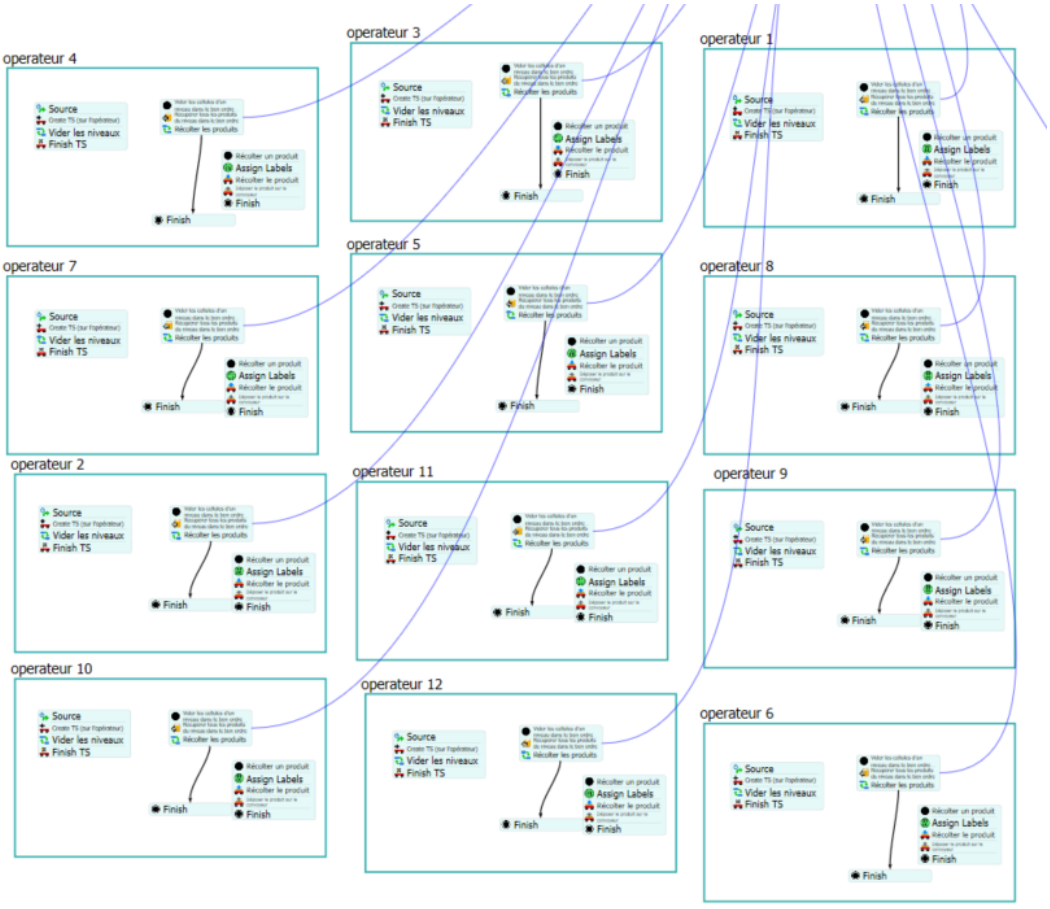


Figure A.7: Human operators heuristics

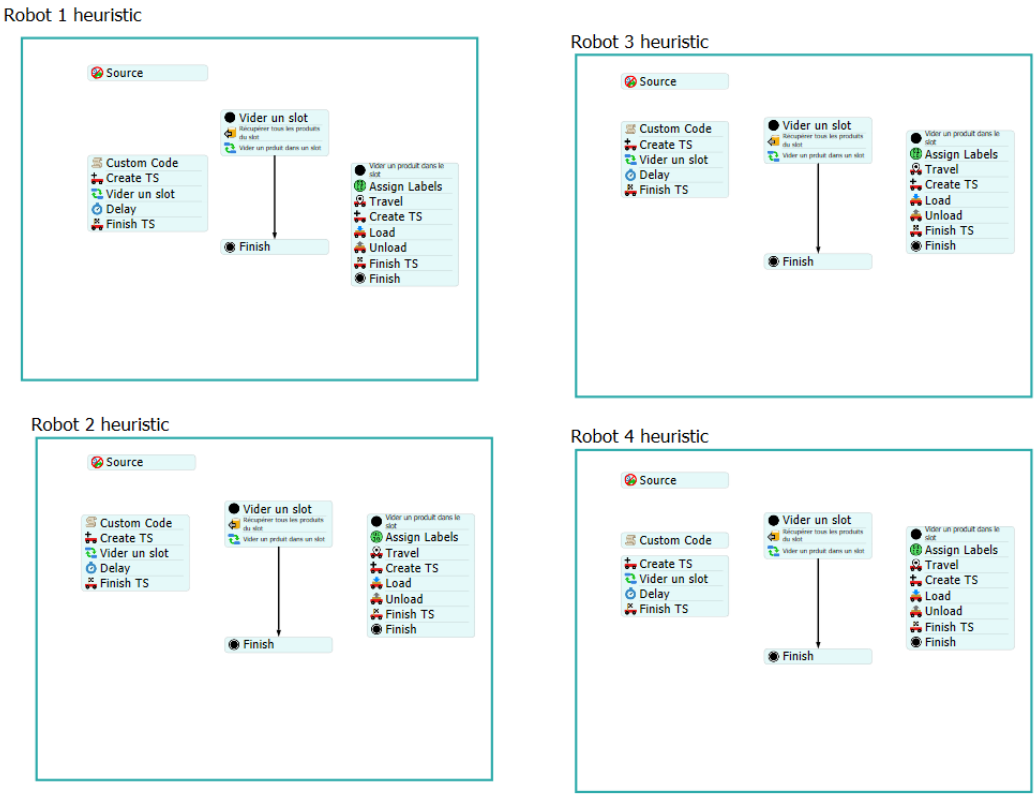


Figure A.8: Robot heuristics.

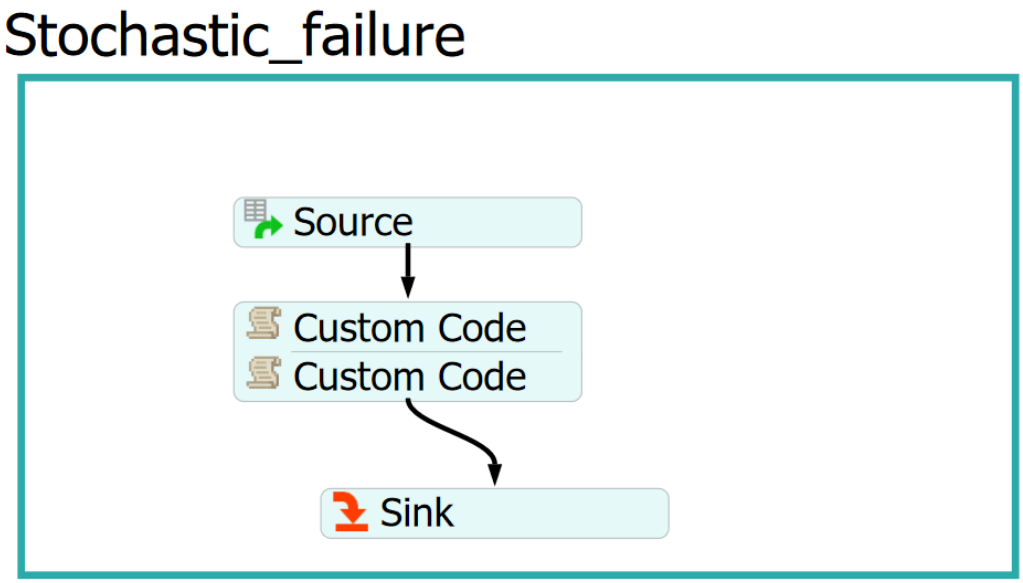


Figure A.9: Stochastic failures.

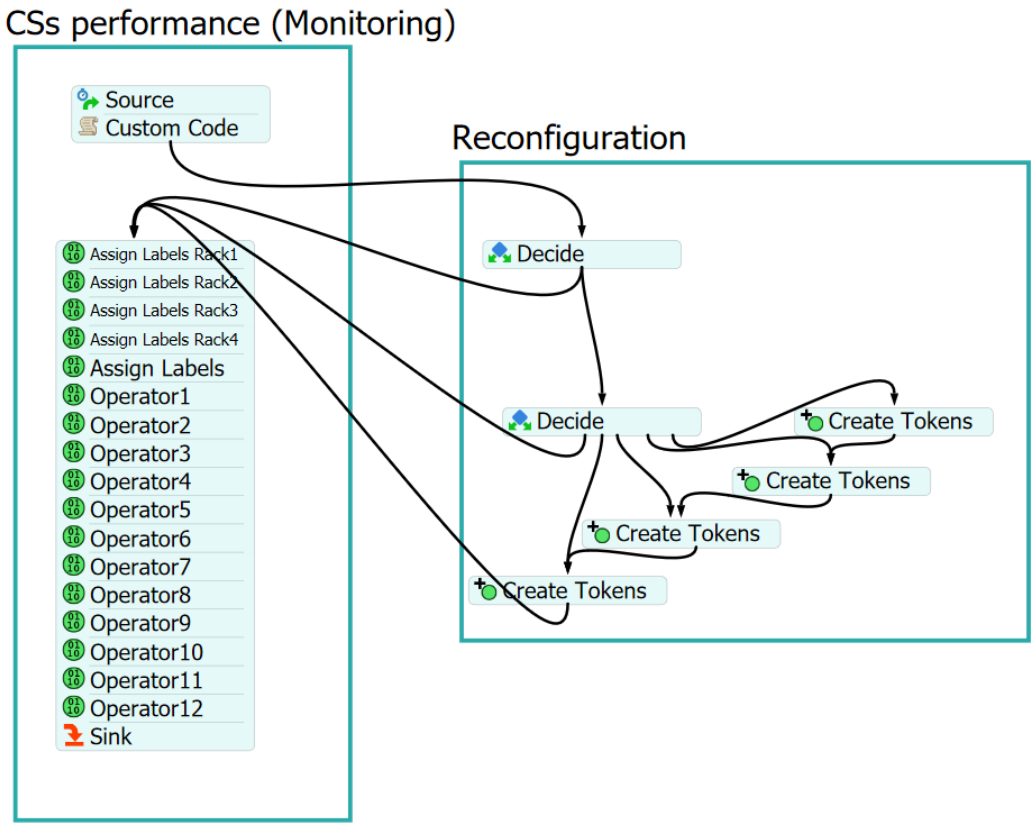


Figure A.10: Supervision algorithm.

Acronyms

ABM Agent-Based Modeling. 15, 21, 23, 73, 75

ARRs Analytical Redundancy Relations. 33

BG Bond Graph. 20, 72, 75

CS Component System. 6, 7, 17, 19, 23, 24, 35, 41, 44–46, 50, 51, 55–57, 59, 63, 66, 67, 72, 79, 83, 85, 86, 101, 104–106

CSP Constraint Satisfaction Problem. 28, 35, 49, 92

CSs Component Systems. 1, 6–9, 12–14, 17, 18, 20, 22, 23, 27, 28, 30, 33, 35, 42–46, 51–57, 59, 60, 63, 65–67, 69, 70, 72, 73, 76, 77, 79, 81, 83–88, 92, 95, 99, 101, 104, 106, 114, 118, 119, 121

DCS Deterministic Component System. 46, 48–50, 62, 65

DCSs Deterministic Component Systems. 53, 72, 79, 83

FDI Fault Detection and Isolation. 28

GA Genetic Algorithms. 122

HDF Hauts-de-France. 4

HG Hypergraph. 6, 24, 27–30, 35, 50, 52, 60, 75, 87, 90, 114, 118

IoT Internet of Things. 16

IoV Internet of Vehicular Networks. 35

M&S Modeling and Simulation. 4, 5, 8, 9, 15, 17, 33

MAS Multi-agent Systems. 23

MCS Managerial Component System. 51, 65, 66, 84, 85, 88, 90, 95

MCSs Managerial Component Systems. ix, 45, 50, 62, 65, 67, 72, 79, 84, 87, 91–93, 95, 97, 99, 102, 111, 112, 114, 118, 121

MDP Markov Decision Processes. 122

MLSHG Multi-level Stochastic Hypergraph. viii–x, 2, 8, 9, 44, 46, 48, 52, 55, 56, 59, 60, 67, 72, 73, 75, 76, 86, 87, 91, 92, 99–105, 111, 115, 118, 120

PCS Physical Component System. 46, 51, 63, 65, 66, 83, 85, 88, 90, 105

PCSs Physical Component Systems. viii, 20, 45, 49, 50, 60, 62, 72, 83, 85, 87, 88, 91–93, 99, 102, 105, 106, 111, 114, 118, 121

PLM Product Lifecycle Management. 16

PMM performance measurement and management. 16

PSO Particle Swarm Optimization. 122

SAM-SoS Stochastic Software Architecture Model. 72

SCS Stochastic Component System. 46, 48–51, 53, 62, 63, 65

SCSs Stochastic Component Systems. 69, 72, 79, 81

SD System Dynamics. 20, 21

SE System Engineering. 4, 12, 13, 19

SoS System of Systems. viii, ix, 1–10, 12–24, 27, 28, 30, 33–37, 40, 42–47, 49, 53–57, 60, 63, 67, 69, 72, 73, 77, 79, 83–88, 91–93, 95, 98, 99, 101, 103–105, 108–116, 118–121

SoSE System of Systems Engineering. 4, 13, 15–17

SoSLM System of Systems Lifecycle Management. 16

SSoS Stochastic System of Systems. 2, 44–46, 57, 59, 72, 76, 77, 86, 99, 101, 103, 104, 115, 116

VCS Virtual Component System. 18, 57

Bibliography

[Ackoff 1971] Russell L Ackoff. Towards a system of systems concepts. Management science, vol. 17, no. 11, pages 661–671, 1971. (Cited on page 13.)

[Antzoulatos 2015] Nikolas Antzoulatos, André Rocha, Elkin Castro, Lavindra de Silva, Tiago Santos, Svetan Ratchev and José Barata. Towards a capability-based framework for reconfiguring industrial production systems. IFAC-PapersOnLine, vol. 48, no. 3, pages 2077–2082, 2015. (Cited on page 35.)

[Antzoulatos 2017] Nikolas Antzoulatos, Elkin Castro, Lavindra de Silva, André Dionisio Rocha, Svetan Ratchev and José Barata. A multi-agent framework for capability-based reconfiguration of industrial assembly systems. International Journal of Production Research, vol. 55, no. 10, pages 2950–2960, 2017. (Cited on page 35.)

[Ayadi 2022] Zouhayra Ayadi, Wadii Boulila, Imed Riadh Farah, Aurelie Leborgne and Pierre Gancarski. Resolution methods for constraint satisfaction problem in remote sensing field: A survey of static and dynamic algorithms. Ecological Informatics, vol. 69, page 101607, 2022. (Cited on page 35.)

[Balchanos 2012a] Michael Balchanos, Yongchang Li and Dimitri Mavris. Towards a method for assessing resilience of complex dynamical systems. In 2012 5th International Symposium on Resilient Control Systems, pages 155–160. IEEE, 2012. (Cited on page 38.)

[Balchanos 2012b] Michael Gregory Balchanos. A probabilistic technique for the assessment of complex dynamic system resilience. 2012. (Cited on page 38.)

[Balchanos 2014] Michael G Balchanos, Jean Charles Domerçant, Huy T Tran and Dimitri N Mavris. Metrics-based analysis and evaluation framework for engineering resilient systems. In 2014 7th International Symposium on Resilient Control Systems (ISRCS), pages 1–7. IEEE, 2014. (Cited on pages viii, 38 and 40.)

[Banasik 2019] Aleksander Banasik, Argyris Kanellopoulos, Jacqueline M Bloemhof-Ruwaard and GDH Claassen. Accounting for uncertainty in eco-efficient agri-food supply chains: A case study for mushroom

production planning. Journal of cleaner production, vol. 216, pages 249–256, 2019. (Cited on page 58.)

[Bar-Yam 2004] Yaneer Bar-Yam. The Characteristics and Emerging Behaviors of System of Systems, 2004. [Online; accessed 17-January-2024]. (Cited on page 14.)

[Barbrook-Johnson 2022] Pete Barbrook-Johnson and Alexandra S. Penn. System dynamics, pages 113–128. Springer International Publishing, Cham, 2022. (Cited on page 20.)

[Bassoli 2021] R. Bassoli, F. Granelli, S. T. Arzo and M. Di Renzo. Toward 5G cloud radio access network: An energy and latency perspective. Transactions on Emerging Telecommunications Technologies, vol. 32, no. 1, page e3669, 2021. (Cited on page 30.)

[Boardman 2006] John Boardman and Brian Sauser. System of Systems-the meaning of of. In 2006 IEEE/SMC international conference on system of systems engineering, pages 6–pp. IEEE, 2006. (Cited on page 14.)

[Bondar 2017] Sergej Bondar, John C Hsu, Alain Pfouga and Josip Stjepandić. Agile digital transformation of System-of-Systems architecture models using Zachman framework. Journal of Industrial Information Integration, vol. 7, pages 33–43, 2017. (Cited on page 16.)

[Boulding 1956] Kenneth E Boulding. General systems theory—the skeleton of science. Management science, vol. 2, no. 3, pages 197–208, 1956. (Cited on page 13.)

[Bourne 2018] Mike Bourne, Monica Franco-Santos, Pietro Micheli and Andrey Pavlov. Performance measurement and management: a system of systems perspective. International Journal of Production Research, vol. 56, no. 8, pages 2788–2799, 2018. (Cited on page 16.)

[Carlock 2001] Paul G Carlock and Robert E Fenton. System of Systems (SoS) enterprise systems engineering for information-intensive organizations. Systems engineering, vol. 4, no. 4, pages 242–261, 2001. (Cited on page 14.)

[Chanter 1978] DO Chanter and JHM Thornley. Mycelial growth and the initiation and growth of sporophores in the mushroom crop: a mathematical model. Microbiology, vol. 106, no. 1, pages 55–65, 1978. (Cited on page 63.)

[Chatterjee 2022] Abheek Chatterjee, Richard Malak and Astrid Layton. Ecology-inspired resilient and affordable system of systems using degree of system order. Systems Engineering, vol. 25, no. 1, pages 3–18, 2022. (Cited on page 16.)

[Chatterjee 2023] Abheek Chatterjee, Cade Helbig, Richard Malak and Astrid Layton. A comparison of graph-theoretic approaches for resilient system of systems design. Journal of Computing and Information Science in Engineering, vol. 23, no. 3, page 030906, 2023. (Cited on page 37.)

[Chemero 2008] Anthony Chemero and Michael T Turvey. Autonomy and hypersets. BioSystems, vol. 91, no. 2, pages 320–330, 2008. (Cited on pages 27 and 46.)

[Chen 2023] Zhiwei Chen, Ziming Zhou, Luogeng Zhang, Chaowei Cui and Jilong Zhong. Mission reliability modeling and evaluation for reconfigurable unmanned weapon system-of-systems based on effective operation loop. Journal of Systems Engineering and Electronics, vol. 34, no. 3, pages 588–597, 2023. (Cited on page 34.)

[Chitra 2019] Uthsav Chitra and Benjamin Raphael. Random walks on hypergraphs with edge-dependent vertex weights. In International conference on machine learning, pages 1172–1181. PMLR, 2019. (Cited on page 44.)

[Chodrow 2023] Philip Chodrow, Nicole Eikmeier and Jamie Haddock. Nonbacktracking spectral clustering of nonuniform hypergraphs. SIAM Journal on Mathematics of Data Science, vol. 5, no. 2, pages 251–279, 2023. (Cited on page 29.)

[Chreim 2023] Abbass Chreim, Abdelkader Belarouci and Rochdi Merzouki. AI-agent-based modeling for Supervision a System of Systems for Mushroom Harvesting. In 2023 18th Annual System of Systems Engineering Conference (SoSe), pages 1–6. IEEE, 2023. (Cited on page 8.)

[Chreim 2024a] Abbass Chreim, Yiwen Chen and Rochdi Merzouki. Stochastic hypergraph model for resilient system of systems: A case study on mushroom harvesting system. In 2024 19th Annual System of Systems Engineering Conference (SoSE), pages 294–299. IEEE, 2024. (Cited on page 8.)

[Chreim 2024b] Abbass Chreim, CHEN Yiwen, Abdeslem Smahi, Jun Jiang and Rochdi Merzouki. Towards Supervision of Stochastic System of

Systems Engineering: A Multi-Level Hypergraph Approach. IEEE Access, 2024. (Cited on page 8.)

[Christensen 2007] Claire Christensen and Reka Albert. Using graph concepts to understand the organization of complex systems. International Journal of Bifurcation and Chaos, vol. 17, no. 07, pages 2201–2214, 2007. (Cited on page 24.)

[Cooksey 2011] K Daniel Cooksey and Dimitri Mavris. Game theory as a means of modeling system of systems viability and feasibility. In 2011 Aerospace Conference, pages 1–11. IEEE, 2011. (Cited on page 15.)

[Darabi 2013] Hamid R Darabi and Mo Mansouri. The role of competition and collaboration in influencing the level of autonomy and belonging in system of systems. IEEE Systems Journal, vol. 7, no. 4, pages 520–527, 2013. (Cited on page 15.)

[Darnhofer 2010] Ika Darnhofer, Stéphane Bellon, Benoît Dedieu and Rebecka Milestad. Adaptiveness to enhance the sustainability of farming systems. A review. Agronomy for sustainable development, vol. 30, pages 545–555, 2010. (Cited on page 112.)

[Dauby 2011] Jason P Dauby and Steven Upholzer. Exploring behavioral dynamics in systems of systems. Procedia Computer Science, vol. 6, pages 34–39, 2011. (Cited on page 15.)

[Davendralingam 2013] Navindran Davendralingam and Daniel DeLaurentis. A robust optimization framework to architecting system of systems. Procedia Computer Science, vol. 16, pages 255–264, 2013. (Cited on page 103.)

[de Amorim Silva 2020] Rafael de Amorim Silva and Rosana T Vaccare Braga. Simulating systems-of-systems with agent-based modeling: a systematic literature review. IEEE Systems Journal, vol. 14, no. 3, pages 3609–3617, 2020. (Cited on page 23.)

[De La Croix 2022] Ntivuguruzwa Jean De La Croix, Mukanyiligira Didacienne and Sibomana Louis. Fuzzy logic-based shiitake mushroom farm control for harvest enhancement. In 2022 10th International Symposium on Digital Forensics and Security (ISDFS), pages 1–6. IEEE, 2022. (Cited on page 63.)

[DeLaurentis 2005a] Daniel DeLaurentis. Understanding transportation as a system-of-systems design problem. In 43rd AIAA aerospace sciences meeting and exhibit, page 123, 2005. (Cited on pages 19 and 21.)

[DeLaurentis 2005b] Daniel DeLaurentis. Understanding Transportation as System-of-Systems Design Problem. 01 2005. (Cited on pages 14 and 19.)

[DeLaurentis 2008] Daniel A DeLaurentis. Appropriate modeling and analysis for systems of systems: Case study synopses using a taxonomy. In 2008 IEEE International Conference on System of Systems Engineering, pages 1–6. IEEE, 2008. (Cited on page 14.)

[Delsing 2022] Jerker Delsing, Géza Kulcsár and Øystein Haugen. SysML modeling of service-oriented system-of-systems. Innovations in Systems and Software Engineering, pages 1–17, 2022. (Cited on pages 72, 73 and 75.)

[Dhulipala 2021] Somayajulu LN Dhulipala, Henry V Burton and Hiba Baroud. A Markov framework for generalized post-event systems recovery modeling: From single to multihazards. Structural Safety, vol. 91, page 102091, 2021. (Cited on page 44.)

[DiMario 2009] Michael J DiMario, John T Boardman and Brian J Sauser. System of systems collaborative formation. IEEE Systems Journal, vol. 3, no. 3, pages 360–368, 2009. (Cited on page 15.)

[Do 2016] Khac Duc Do. Global robust adaptive path-tracking control of underactuated ships under stochastic disturbances. Ocean Engineering, vol. 111, pages 267–278, 2016. (Cited on page 33.)

[Doulgeris 2012] G Doulgeris, T Korakianitis, P Pilidis and E Tsoudis. Techno-economic and environmental risk analysis for advanced marine propulsion systems. Applied energy, vol. 99, pages 1–12, 2012. (Cited on page 44.)

[Dridi 2020] Charaf Eddine Dridi, Zakaria Benzadri and Faiza Belala. System of systems engineering: Meta-modelling perspective. In 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), pages 000135–000144. IEEE, 2020. (Cited on pages 72, 73 and 75.)

[Dui 2023] Hongyan Dui, Meng Liu, Jiaying Song and Shaomin Wu. Importance measure-based resilience management: Review, methodology and perspectives on maintenance. Reliability Engineering & System Safety, vol. 237, page 109383, 2023. (Cited on page 36.)

[Eisner 1991] Howard Eisner, John Marciniak and Ray McMillan. Computer-aided system of systems (S2) engineering. In Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics, pages 531–537. IEEE, 1991. (Cited on page 13.)

[El-Awady 2023] Ahmed El-Awady and Kumaraswamy Ponnambalam. Bayesian networks for failure analysis of complex systems using different data sources. In Engineering Reliability and Risk Assessment, pages 1–17. Elsevier, 2023. (Cited on page 30.)

[Ender 2010] Tommer Ender, Ryan F Leurck, Brian Weaver, Paul Miceli, William Dale Blair, Philip West and Dimitri Mavris. Systems-of-systems analysis of ballistic missile defense architecture effectiveness through surrogate modeling and simulation. IEEE Systems Journal, vol. 4, no. 2, pages 156–166, 2010. (Cited on page 15.)

[Fan 2018] Mengfei Fan, Zhiguo Zeng, Enrico Zio, Rui Kang and Ying Chen. A stochastic hybrid systems model of common-cause failures of degrading components. Reliability Engineering & System Safety, vol. 172, pages 159–170, 2018. (Cited on page 34.)

[Feng 2023] Yimin Feng, Chenchu Zhou, Qiang Zou, Yusheng Liu, Jiyuan Lyu and Xinfeng Wu. A goal-based approach for modeling and simulation of different types of system-of-systems. Journal of Systems Engineering and Electronics, vol. 34, no. 3, pages 627–640, 2023. (Cited on page 23.)

[Fortino 2020] Giancarlo Fortino, Claudio Savaglio, Giandomenico Spezzano and MengChu Zhou. Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools. IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 1, pages 223–236, 2020. (Cited on page 16.)

[Gaiardelli 2024] Sebastiano Gaiardelli, Michele Lora, Stefano Spellini and Franco Fummi. RRPDG: A Graph Model to Enable AI-Based Production Reconfiguration and Optimization. IEEE Transactions on Industrial Informatics, 2024. (Cited on page 35.)

[Galuppi 2023] Francesco Galuppi, Raffaella Mulas and Lorenzo Venturello. Spectral theory of weighted hypergraphs via tensors. Linear and Multilinear Algebra, vol. 71, no. 3, pages 317–347, 2023. (Cited on page 29.)

[Gao 2023] Yuan Gao, Jieyuan Cheng, Yongliang Tian and Hu Liu. Machine Learning-Based Evaluation of the Contribution Effectiveness in SoS Missions. IEEE Systems Journal, 2023. (Cited on page 35.)

[Gleirscher 2023] Mario Gleirscher. Supervision of Intelligent Systems: An Overview. Applicable Formal Methods for Safe Industrial Products: Essays Dedicated to Jan Peleska on the Occasion of His 65th Birthday, pages 202–221, 2023. (Cited on page 31.)

[Goldbeck 2019] Nils Goldbeck, Panagiotis Angeloudis and Washington Y Ochieng. Resilience assessment for interdependent urban infrastructure systems using dynamic network flow models. Reliability Engineering & System Safety, vol. 188, pages 62–79, 2019. (Cited on page 34.)

[Gorod 2008] Alex Gorod, Brian Sauser and John Boardman. System-of-systems engineering management: A review of modern history and a path forward. IEEE Systems Journal, vol. 2, no. 4, pages 484–499, 2008. (Cited on page 14.)

[Guo 2014] Jin-Li Guo and Xin-Yun Zhu. Weighted Hypernetworks. arXiv preprint arXiv:1408.4355, 2014. (Cited on page 30.)

[Haimes 2009] Yacov Y Haimes. On the definition of resilience in systems. Risk Analysis: An International Journal, vol. 29, no. 4, 2009. (Cited on page 36.)

[Hämberg 2013] Eva Hämberg. Supervision as control system: the design of supervision as a regulatory instrument in the social services sector in Sweden. Scandinavian Journal of Public Administration, vol. 17, no. 3, pages 45–64, 2013. (Cited on page 31.)

[Han 2013] Seung Yeob Han and Daniel DeLaurentis. Development interdependency modeling for system-of-systems (SoS) using Bayesian networks: SoS management strategy planning. Procedia Computer Science, vol. 16, pages 698–707, 2013. (Cited on page 30.)

[Harrison 2016] Willie K Harrison. The role of graph theory in system of systems engineering. IEEE Access, vol. 4, pages 1716–1742, 2016. (Cited on page 24.)

[Hu 2019] Xiaomei Hu, Zhaoren Pan and Shunke Lv. Picking path optimization of agaricus bisporus picking robot. Mathematical Problems in Engineering, vol. 2019, pages 1–16, 2019. (Cited on page 58.)

[Huang 2021] Mingsen Huang, Long He, Daeun Choi, John Pecchia and Yaoming Li. Picking dynamic analysis for robotic harvesting of Agaricus bisporus mushrooms. Computers and Electronics in Agriculture, vol. 185, page 106145, 2021. (Cited on page 59.)

[Huang 2023] Yang Huang, Aimin Luo, Tao Chen, Mengmeng Zhang, Bangbang Ren and Yanjie Song. When architecture meets RL+ EA: A hybrid intelligent optimization approach for selecting combat system-of-systems architecture. Advanced Engineering Informatics, vol. 58, page 102209, 2023. (Cited on page 16.)

[Hyun 2023] Sangwon Hyun, Jiyoung Song, Eunkyoung Jee and Doo-Hwan Bae. Timed pattern-based analysis of collaboration failures in system-of-systems. Journal of Systems and Software, vol. 198, page 111613, 2023. (Cited on page 33.)

[Ibne Hossain 2020] Niamat Ullah Ibne Hossain, Morteza Nagahi, Raed Jaradat, Chiranjibi Shah, Randy Buchanan and Michael Hamilton. Modeling and assessing cyber resilience of smart grid using Bayesian network-based approach: a system of systems problem. Journal of Computational Design and Engineering, vol. 7, no. 3, pages 352–366, 2020. (Cited on page 19.)

[Ibrar 2020] Muhammad Ibrar, Aamir Akbar, Syed Rooh Ullah Jan, Mian Ahmad Jan, Lei Wang, Houbing Song and Nadir Shah. Artnet: Ai-based resource allocation and task offloading in a reconfigurable internet of vehicular networks. IEEE Transactions on Network Science and Engineering, vol. 9, no. 1, pages 67–77, 2020. (Cited on page 35.)

[Iwanaga 2021] Takuya Iwanaga, Hsiao-Hsuan Wang, Serena H Hamilton, Volker Grimm, Tomasz E Koralewski, Alejandro Salado, Sondoss Elsawah, Saman Razavi, Jing Yang, Pierre Glynnet al. Socio-technical scales in socio-environmental modeling: Managing a system-of-systems modeling approach. Environmental Modelling & Software, vol. 135, page 104885, 2021. (Cited on page 16.)

[Jackson 1984] Michael C Jackson and Paul Keys. Towards a system of systems methodologies. Journal of the operational research society, vol. 35, pages 473–486, 1984. (Cited on page 13.)

[Jackson 2013] Scott Jackson and Timothy LJ Ferris. Resilience principles for engineered systems. Systems Engineering, vol. 16, no. 2, pages 152–164, 2013. (Cited on page 37.)

[Jalving 2019] Jordan Jalving, Yankai Cao and Victor M Zavala. Graph-based modeling and simulation of complex systems. Computers & Chemical Engineering, vol. 125, pages 134–154, 2019. (Cited on page 25.)

[Jamshidi 2008a] Mohammad Jamshidi. System of systems engineering: innovations for the 21st century. Deakin University, 2008. (Cited on page 14.)

[Jamshidi 2008b] Mohammad Jamshidi. Systems of Systems Engineering: Principles and Applications. 2008. (Cited on page 14.)

[Joyce 1987] Jeffrey Joyce, Greg Lomow, Konrad Slind and Brian Unger. Monitoring distributed systems. ACM Transactions on Computer Systems (TOCS), vol. 5, no. 2, pages 121–150, 1987. (Cited on page 34.)

[Karan 2021] Ebrahim Karan and Sadegh Asgari. Resilience of food, energy, and water systems to a sudden labor shortage. Environment Systems and Decisions, vol. 41, no. 1, pages 63–81, 2021. (Cited on page 111.)

[Keating 2003] Charles Keating, Ralph Rogers, Resit Unal, David Dryer, Andres Sousa-Poza, Robert Safford, William Peterson and Ghaith Rabadi. System of systems engineering. Engineering Management Journal, vol. 15, no. 3, pages 36–45, 2003. (Cited on page 14.)

[Kerr 2020] Chad Kerr, Raed Jaradat and Niamat Ullah Ibne Hossain. Battlefield mapping by an unmanned aerial vehicle swarm: Applied systems engineering processes and architectural considerations from system of systems. IEEE Access, vol. 8, pages 20892–20903, 2020. (Cited on page 19.)

[Khabarov 2008] Nikolay Khabarov, Elena Moltchanova and Michael Obersteiner. Valuing weather observation systems for forest fire management. IEEE Systems Journal, vol. 2, no. 3, pages 349–357, 2008. (Cited on page 44.)

[Khalil 2012] Wissam Khalil, Rochdi Merzouki, Belkacem Ould-Bouamama and Hafid Haffaf. Hypergraph models for system of systems supervision design. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 42, no. 4, pages 1005–1012, 2012. (Cited on pages viii, ix, 2, 6, 15, 27, 28, 29, 33, 34, 35, 45, 46, 49, 72, 73, 75, 77, 92, 95, 99, 100 and 102.)

[Klein 2013] John Klein and Hans van Vliet. A systematic review of system-of-systems architecture research. In Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures, QoSA '13, page 13–22, New York, NY, USA, 2013. Association for Computing Machinery. (Cited on page 15.)

[Ko 2022] Jisung Ko, Yeonwoo Kook and Kyomin Jung. Growth patterns and models of real-world hypergraphs. Knowledge and Information Systems, 2022. (Cited on page 30.)

[Kotov 1997] Vadim Kotov. Systems of systems as communicating structures, volume 119. Hewlett Packard Laboratories, 1997. (Cited on page 13.)

[Kozma 2021] Dániel Kozma, Pál Varga and Felix Larrinaga. System of systems lifecycle management—a new concept based on process engineering methodologies. Applied Sciences, vol. 11, no. 8, page 3386, 2021. (Cited on page 16.)

[Krygiel 1999] Annette J Krygiel. Behind the wizard's curtain: An integration environment for a system of systems. National Defense University, 1999. (Cited on page 14.)

[Kumar 2014] Pushpendra Kumar, Rochdi Merzouki, Blaise Conrard, Vincent Coelen and Belkacem Ould Bouamama. Multilevel modeling of the traffic dynamic. IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 3, pages 1066–1082, 2014. (Cited on pages 72 and 75.)

[Kumar 2017] Pushpendra Kumar, Rochdi Merzouki and Belkacem Ould Bouamama. Multilevel modeling of system of systems. IEEE transactions on systems, man, and cybernetics: systems, vol. 48, no. 8, pages 1309–1320, 2017. (Cited on pages viii, 2, 20, 21, 45, 72, 73 and 75.)

[Li 2018] Jichao Li, Jiang Jiang, Kewei Yang and Yingwu Chen. Research on functional robustness of heterogeneous combat networks. IEEE Systems Journal, vol. 13, no. 2, pages 1487–1495, 2018. (Cited on page 33.)

[Li 2022] Hongbo Li, Yaling Wu, Minghao Yin and Zhanshan Li. A portfolio-based approach to select efficient variable ordering heuristics for constraint satisfaction problems. In 28th International Conference on Principles and Practice of Constraint Programming (CP 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022. (Cited on page 35.)

[Lightsey 2001] Bob Lightsey. Systems engineering fundamentals. Defense acquisition univ ft belvoir va, 2001. (Cited on page 12.)

[Lin 2023] Menglong Lin, Tao Chen, Honghui Chen, Bangbang Ren and Mengmeng Zhang. When architecture meets AI: A deep reinforcement learning approach for system of systems design. Advanced Engineering Informatics, vol. 56, page 101965, 2023. (Cited on page 22.)

[Lubas 2017] Debra Greenhalgh Lubas. Department of defense system of systems reliability challenges. In 2017 Annual Reliability and Maintainability Symposium (RAMS), pages 1–6. IEEE, 2017. (Cited on page 19.)

[Madni 2009] Azad M Madni and Scott Jackson. Towards a conceptual framework for resilience engineering. IEEE Systems Journal, vol. 3, no. 2, pages 181–191, 2009. (Cited on pages 36 and 103.)

[Mahulkar 2009] Vishal Mahulkar, Shawn McKay, Douglas E. Adams and Alok R. Chaturvedi. System-of-Systems Modeling and Simulation of a Ship Environment With Wireless and Intelligent Maintenance Technologies. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 39, no. 6, pages 1255–1270, 2009. (Cited on page 19.)

[Maier 1998] Mark W Maier. Architecting principles for systems-of-systems. Systems Engineering: The Journal of the International Council on Systems Engineering, vol. 1, no. 4, pages 267–284, 1998. (Cited on pages 1, 7, 13, 17, 43, 104, 119 and 121.)

[Maksuti 2021] Silia Maksuti, Ani Bicaku, Mario Zsilak, Igor Ivkic, Bálint Péceli, Gábor Singler, Kristóf Kovács, Markus Tauber and Jerker Delsing. Automated and secure onboarding for system of systems. IEEE Access, vol. 9, pages 111095–111113, 2021. (Cited on page 79.)

[Manevska-Tasevska 2021] Gordana Manevska-Tasevska, Andrea Petitt, Sara Larsson, Ivan Bimbilovski, Miranda PM Meuwissen, Peter H Feindt and Julie Urquhart. Adaptive governance and resilience capacity of farms: The fit between farmers' decisions and agricultural policies. Frontiers in Environmental Science, vol. 9, page 668836, 2021. (Cited on page 112.)

[Mansouri 2009] Mo Mansouri, Alex Gorod, Thomas H Wakeman and Brian Sauser. Maritime transportation system of systems management framework: A system of systems engineering approach. International

Journal of Ocean Systems Management, vol. 1, no. 2, pages 200–226, 2009. (Cited on page 19.)

[Mechqrane 2020] Younes Mechqrane, Mohamed Wahbi, Christian Bessiere and Kenneth N Brown. Reordering all agents in asynchronous backtracking for distributed constraint satisfaction problems. Artificial Intelligence, vol. 278, page 103169, 2020. (Cited on page 35.)

[Meuwissen 2019] Miranda PM Meuwissen, Peter H Feindt, Alisa Spiegel, Catrien JAM Termeer, Erik Mathijs, Yann De Mey, Robert Finger, Alfons Balmann, Erwin Wauters, Julie Urquhartet al. A framework to assess the resilience of farming systems. Agricultural Systems, vol. 176, page 102656, 2019. (Cited on page 111.)

[Meuwissen 2020] Miranda PM Meuwissen, Peter H Feindt, Peter Midmore, Erwin Wauters, Robert Finger, Franziska Appel, Alisa Spiegel, Erik Mathijs, Katrien JAM Termeer, Alfons Balmannet al. The struggle of farming systems in Europe: looking for explanations through the lens of resilience, 2020. (Cited on page 111.)

[Mittal 2015] Saurabh Mittal, Mark Ruth, Annabelle Pratt, Monte Lunacek, Dheepak Krishnamurthy and Wesley Jones. System-of-systems approach for integrated energy systems modeling and simulation. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2015. (Cited on page 33.)

[Mo 2016] Hua-Dong Mo, Yan-Fu Li and Enrico Zio. A system-of-systems framework for the reliability analysis of distributed generation systems accounting for the impact of degraded communication networks. Applied Energy, vol. 183, pages 805–822, 2016. (Cited on page 44.)

[Modaber 2024] Mahtab Modaber, Martijn Hendriks, Marc Geilen, Twan Basten and Jeroen Voeten. A Method for Building Trustworthy Hybrid Performance Models for Cyber-Physical Systems of Systems. IEEE Access, 2024. (Cited on page 33.)

[Mohsin 2019] Ahmad Mohsin, Naeem Khalid Janjua, Syed MS Islam and Valdemar Vicente Graciano Neto. Modeling approaches for system-of-systems dynamic architecture: Overview, taxonomy and future prospects. In 2019 14th Annual Conference System of Systems Engineering (SoSE), pages 49–56. IEEE, 2019. (Cited on page 30.)

[Mohsin 2020] Ahmad Mohsin, Naeem Khalid Janjua, Syed MS Islam and Muhammad Ali Babar. SAM-SoS: A stochastic software architecture

modeling and verification approach for complex system-of-systems. IEEE Access, vol. 8, pages 177580–177603, 2020. (Cited on pages 72 and 73.)

[Moradi 2018] A Behrang Moradi, B Nicolas Daclin and C Vincent Chapurlat. Analysing eco-system of 'Ilities' for resilience evaluation in systems of systems. In Proc. Int. Conf. Model., Optim. Simul., pages 1–7, 2018. (Cited on page 103.)

[Mordecai 2016] Yaniv Mordecai, Ori Orhof and Dov Dori. Model-based interoperability engineering in systems-of-systems and civil aviation. IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 4, pages 637–648, 2016. (Cited on page 19.)

[Mostafavi 2011] Ali Mostafavi, Dulcy M Abraham, Daniel DeLaurentis and Joseph Sinfield. Exploring the dimensions of systems of innovation analysis: A system of systems framework. IEEE Systems Journal, vol. 5, no. 2, pages 256–265, 2011. (Cited on pages 12 and 45.)

[Muller 2000] Pierre-Alain Muller and Nathalie Gaertner. Modélisation objet avec uml, volume 514. Eyrolles Paris, 2000. (Cited on page 11.)

[Myhan 2023] Ryszard Myhan, Ewelina Jachimczyk and Marek Markowski. The Use of Graph Theory for Modeling and Analyzing the Structure of a Complex System, with the Example of an Industrial Grain Drying Line. Processes, vol. 11, no. 10, page 2812, 2023. (Cited on page 25.)

[Natarajan 2012] Sathish Natarajan, Kaushik Ghosh and Rajagopalan Srinivasan. An ontology for distributed process supervision of large-scale chemical plants. Computers & Chemical Engineering, vol. 46, pages 124–140, 2012. (Cited on page 33.)

[Neches 2013] Robert Neches and Azad M Madni. Towards affordably adaptable and effective systems. Systems Engineering, vol. 16, no. 2, pages 224–234, 2013. (Cited on page 36.)

[Nelson 2019] Travis Nelson, John M Borky and Ronald M Sega. System-of-systems quality attribute-based architectural alternatives. IEEE Systems Journal, vol. 14, no. 3, pages 3844–3854, 2019. (Cited on page 35.)

[Nicholls 2004] Andrew Nicholls, Leon Bren and Neil Humphreys. Harvester productivity and operator fatigue: working extended hours. International journal of forest engineering, vol. 15, no. 2, pages 57–65, 2004. (Cited on page 86.)

[Nilsson 2024] Jacob Nilsson, Saleha Javed, Kim Albertsson, Jerker Delsing, Marcus Liwicki and Fredrik Sandin. Ai concepts for system of systems dynamic interoperability. Sensors, vol. 24, no. 9, page 2921, 2024. (Cited on page 22.)

[Owens 1996] William A Owens. The emerging us system-of-systems. Number 63. National Defense University, Institute for National Strategic Studies, 1996. (Cited on pages 18 and 19.)

[Oyewole 2020] Peju Adesina Oyewole and Dilan Jayaweera. Power system security with cyber-physical power system operation. IEEE Access, vol. 8, pages 179970–179982, 2020. (Cited on page 34.)

[Panayi 2017] Efstathios Panayi, Gareth W Peters and George Kyriakides. Statistical modelling for precision agriculture: A case study in optimal environmental schedules for Agaricus Bisporus production via variable domain functional regression. PLoS One, vol. 12, no. 9, page e0181921, 2017. (Cited on page 58.)

[Panayi 2023] Efstathios Panayi, Gareth W Peters and George Kyriakides. Correction: Statistical modelling for precision agriculture: A case study in optimal environmental schedules for Agaricus Bisporus production via variable domain functional regression. Plos one, vol. 18, no. 1, page e0280374, 2023. (Cited on page 63.)

[Panteli 2015] Mathaios Panteli and Pierluigi Mancarella. Modeling and evaluating the resilience of critical electrical power infrastructure to extreme weather events. IEEE Systems Journal, vol. 11, no. 3, pages 1733–1742, 2015. (Cited on page 44.)

[Paut 2021] Raphaël Paut, Rodolphe Sabatier, Arnaud Dufils and Marc Tchamitchian. How to reconcile short-term and long-term objectives in mixed farms? A dynamic model application to mixed fruit tree-vegetable systems. Agricultural Systems, vol. 187, page 103011, 2021. (Cited on page 79.)

[Perron 2023] Laurent Perron, Frédéric Didier and Steven Gay. The CP-SAT-LP Solver (Invited Talk). In 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023. (Cited on page 92.)

[Price 2022] Alex Price, Nels Knutson and Cihan Dagli. Using System-of-Systems Optimization for Healthcare: A Use Case in

Radiation Oncology. In 2022 IEEE International Systems Conference (SysCon), pages 1–5. IEEE, 2022. (Cited on page 19.)

[Purfürst 2011] F Thomas Purfürst and Jörn Erler. The human influence on productivity in harvester operations. International Journal of Forest Engineering, vol. 22, no. 2, pages 15–22, 2011. (Cited on page 87.)

[Raman 2023] Ramakrishnan Raman, Nikhil Gupta and Yogananda Jeppu. Framework for Formal Verification of Machine Learning Based Complex System-of-Systems. Insight, vol. 26, no. 1, pages 91–102, 2023. (Cited on page 16.)

[Ran 2020] M. Ran and X. Bai. Vehicle cooperative network model based on hypergraph in vehicular fog computing. Sensors (Basel, Switzerland), vol. 20, no. 8, page 2269, 2020. (Cited on page 30.)

[Reed 2001] JN Reed, SJ Miles, J Butler, M Baldwin and R Noble. AE—Automation and emerging technologies: Automatic mushroom harvester development. Journal of Agricultural Engineering Research, vol. 78, no. 1, pages 15–23, 2001. (Cited on page 58.)

[Rezvanian 2016] Alireza Rezvanian and Mohammad Reza Meybodi. Stochastic graph as a model for social networks. Computers in Human Behavior, vol. 64, pages 621–640, 2016. (Cited on page 25.)

[Sage 2001] Andrew P Sage and Christopher D Cuppan. On the systems engineering and management of systems of systems and federations of systems. Information knowledge systems management, vol. 2, no. 4, pages 325–345, 2001. (Cited on pages 14 and 55.)

[Sauser 2010a] Brian Sauser, John Boardman and Dinesh Verma. Systomics: Toward a biology of system of systems. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol. 40, no. 4, pages 803–814, 2010. (Cited on page 15.)

[Sauser 2010b] Brian Sauser, John Boardman and Dinesh Verma. Systomics: Toward a Biology of System of Systems. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 40, no. 4, pages 803–814, 2010. (Cited on page 19.)

[Scrimieri 2023] Daniele Scrimieri, Omar Adalat, Shukri Afazov and Svetan Ratchev. An integrated data-and capability-driven approach to the reconfiguration of agent-based production systems. The International Journal of Advanced Manufacturing Technology, vol. 124, no. 3, pages 1155–1168, 2023. (Cited on page 35.)

[Silva 2015] Eduardo Silva, Thais Batista and Flavio Oquendo. A mission-oriented approach for designing system-of-systems. In 2015 10th System of Systems Engineering Conference (SoSE), pages 346–351, 2015. (Cited on page 15.)

[Simpson 2008] Joseph J Simpson and Cihan H Dagli. System of systems: Power and paradox. In 2008 IEEE International Conference on System of Systems Engineering, pages 1–5. Ieee, 2008. (Cited on page 14.)

[Sloane 2007] Elliot Sloane, Thomas Way, Vijay Gehlot, Robert Beck, James Solderitch and Elzbieta Dziembowski. A hybrid approach to modeling SOA Systems of Systems using CPN and MESA/Extend. In 2007 1st Annual IEEE Systems Conference, pages 1–7. IEEE, 2007. (Cited on page 14.)

[Sloane 2008] Elliot Sloane. Systems of systems (sose) engineering for the 21 st century healthcare enterprise. In 2008 2nd Annual IEEE Systems Conference, pages 1–4. IEEE, 2008. (Cited on page 19.)

[Sohn 1985] Tae-Won Sohn and Julius Surkis. System dynamics: A methodology for testing dynamic behavioral hypotheses. IEEE transactions on systems, man, and cybernetics, no. 3, pages 399–408, 1985. (Cited on page 20.)

[Soyez 2015] Jean-Baptiste Soyez, Gildas Morvan, Rochdi Merzouki and Daniel Dupont. Multilevel agent-based modeling of system of systems. IEEE Systems Journal, vol. 11, no. 4, pages 2084–2095, 2015. (Cited on pages viii, 2, 18, 23, 24, 44, 45, 57, 72, 73 and 75.)

[Staroswiecki 2001] Marcel Staroswiecki and G Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. Automatica, vol. 37, no. 5, pages 687–699, 2001. (Cited on page 33.)

[Stephan 2022] Lucas Stephan and Yuanzhi Zhu. Sparse random hypergraphs: Non-backtracking spectra and community detection. In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). IEEE, 2022. (Cited on page 29.)

[Sun 2024] Qin Sun, Hongxu Li, Yuanfu Zhong, Kezhou Ren and Yingchao Zhang. Deep reinforcement learning-based resilience enhancement strategy of unmanned weapon system-of-systems under inevitable interferences. Reliability Engineering & System Safety, vol. 242, page 109749, 2024. (Cited on page 37.)

[Tran 2015] Huy T Tran. A complex networks approach to designing resilient system-of-systems. 2015. (Cited on pages 35, 36 and 110.)

[Tran 2016] Huy T Tran, Jean Charles Domerçant and Dimitri N Mavris. A network-based cost comparison of resilient and robust system-of-systems. Procedia Computer Science, vol. 95, pages 126–133, 2016. (Cited on pages viii, 18, 19, 40, 42, 109 and 110.)

[Tsouros 2024] Dimosthenis Tsouros, Senne Berden and Tias Guns. Learning to learn in interactive constraint acquisition. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 8154–8162, 2024. (Cited on page 92.)

[Turnquist 2013] Mark Turnquist and Eric Vugrin. Design for resilience in infrastructure distribution networks. Environment Systems & Decisions, vol. 33, pages 104–120, 2013. (Cited on page 37.)

[Uday 2014] Payuna Uday and Karen B Marais. Resilience-based system importance measures for system-of-systems. Procedia Computer Science, vol. 28, pages 257–264, 2014. (Cited on page 104.)

[Uday 2015] Payuna Uday and Karen Marais. Designing resilient systems-of-systems: A survey of metrics, methods, and challenges. Systems Engineering, vol. 18, no. 5, pages 491–510, 2015. (Cited on page 37.)

[Uslar 2019] Mathias Uslar, Sebastian Rohjans, Christian Neureiter, Filip Pröstl Andrén, Jorge Velasquez, Cornelius Steinbrink, Venizelos Efthymiou, Gianluigi Migliavacca, Seppo Horsmanheimo, Helfried Brunneret al. Applying the smart grid architecture model for designing and validating system-of-systems in the power and energy domain: A European perspective. Energies, vol. 12, no. 2, page 258, 2019. (Cited on page 16.)

[Van Beek 2006] Peter Van Beek. Backtracking search algorithms. In Foundations of artificial intelligence, volume 2, pages 85–134. Elsevier, 2006. (Cited on page 35.)

[Venkatasubramanian 2005] Venkat Venkatasubramanian. Prognostic and diagnostic monitoring of complex systems for product lifecycle management: Challenges and opportunities. Computers & chemical engineering, vol. 29, no. 6, pages 1253–1263, 2005. (Cited on page 33.)

[Vierhauser 2016] Michael Vierhauser, Rick Rabiser, Paul Grünbacher, Klaus Seyerlehner, Stefan Wallner and Helmut Zeisel. ReMinds: A flexible runtime monitoring framework for systems of systems. Journal of Systems and Software, vol. 112, pages 123–136, 2016. (Cited on page 16.)

[Vugrin 2010] Eric D Vugrin, Drake E Warren, Mark A Ehlen and R Chris Camphouse. A framework for assessing the resilience of infrastructure and economic systems. Sustainable and resilient critical infrastructure systems: Simulation, modeling, and intelligent engineering, pages 77–116, 2010. (Cited on pages viii, 36, 38 and 39.)

[Vukelić 2014] Nataša Vukelić and Vesna Rodić. FARMERS'MANAGEMENT CAPACITIES AS A SUCCESS FACTOR IN AGRICULTURE: A REVIEW. , vol. 61, no. 3, pages 805–814, 2014. (Cited on page 87.)

[Wachholder 2015] Dominik Wachholder and Chris Stary. Enabling emergent behavior in systems-of-systems through bigraph-based modeling. In 2015 10th System of Systems Engineering Conference (SoSE), pages 334–339. IEEE, 2015. (Cited on pages viii and 25.)

[Wang 2019] Zhao Wang, Sifeng Liu and Zhigeng Fang. Research on SoS-GERT network model for equipment system of systems contribution evaluation based on joint operation. IEEE Systems Journal, vol. 14, no. 3, pages 4188–4196, 2019. (Cited on page 33.)

[Wang 2022] Hongping Wang, Yi-Ping Fang and Enrico Zio. Resilience-oriented optimal post-disruption reconfiguration for coupled traffic-power systems. Reliability Engineering & System Safety, vol. 222, page 108408, 2022. (Cited on page 36.)

[Wang 2023] Zhen Wang, Kaihua Xi, Aijie Cheng, Hai Xiang Lin, André CM Ran, Jan H van Schuppen and Chenghui Zhang. Synchronization of power systems under stochastic disturbances. Automatica, vol. 151, page 110884, 2023. (Cited on page 33.)

[Watson 2021] BC Watson, A Chowdhry, MJ Weissburg and B Bras. A New Resilience Metric to Compare System of Systems Architecture. IEEE Syst. J. 1–12, 2021. (Cited on page 37.)

[Weinert 2020a] Benjamin Weinert and Mathias Uslar. Challenges for System of Systems in the Agriculture Application Domain. In 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), pages 000355–000360, 2020. (Cited on page 19.)

[Weinert 2020b] Benjamin Weinert and Mathias Uslar. Challenges for system of systems in the agriculture application domain. In 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), pages 000355–000360. IEEE, 2020. (Cited on page 19.)

[Woods 2015] David D Woods. Four concepts for resilience and the implications for the future of resilience engineering. Reliability engineering & system safety, vol. 141, pages 5–9, 2015. (Cited on page 103.)

[Yahia 2010] W Ben Yahia, Philippe Polet, Frédéric Vanderhaegen and N Tricot. Human Factors in Studies of the Safety and Reliability of Agro-Equipment. IFAC Proceedings Volumes, vol. 43, no. 13, pages 13–18, 2010. (Cited on page 87.)

[Zhao 2018] Bo Zhao, Xiangjin Wang, Da Lin, Madison M Calvin, Julia C Morgan, Ruwen Qin and Caisheng Wang. Energy management of multiple microgrids based on a system of systems architecture. IEEE Transactions on Power Systems, vol. 33, no. 6, pages 6410–6421, 2018. (Cited on page 19.)

[Zhou 2011] Bo Zhou, Aleksandra Dvoryanchikova, Andrei Lobov and Jose Luis Martinez Lastra. Modeling system of systems: A generic method based on system characteristics and interface. In 2011 9th IEEE International Conference on Industrial Informatics, pages 361–368. IEEE, 2011. (Cited on page 15.)

Towards Modeling and Supervision of Multilevel Stochastic Systems of Systems: A Hypergraph Approach.

Abstract: System-of-Systems (SoS) face significant challenges, including heterogeneity, scalability, and complex interactions among component systems (CSs). These systems typically operate in dynamic environments, introducing uncertainty and stochastic behavior. Many existing studies tend to oversimplify these complexities, with some focusing only on the dynamics of CSs without adequately addressing their structure, mission, and goals. Additionally, limited research has focused on supervising SoS under such conditions. Graph models, such as hypergraphs (HG), have proven effective in modeling the structure of SoS, while stochastic and weighted hypergraphs have been successfully employed to manage stochasticity in other complex systems. In this thesis, the Multi-Level Stochastic Hypergraph (MLSHG) model is introduced to address the challenges of modeling stochastic SoS. The model adheres to the key properties of SoS as defined by Maier, distinguishing it from traditional complex systems. A novel algorithm for supervising large-scale SoS is also proposed, integrating bottom-up monitoring with top-down reconfiguration to achieve long-term goals. The proposed framework supports resilience in these complex systems through recovery mechanisms based on redundancy. In a case study on a mushroom harvesting SoS, the model demonstrated clear advantages in addressing SoS modeling challenges compared to existing methodologies. The results showed that incorporating stochastic disturbances with an adaptive threshold enabled early reconfiguration during supervision, reducing deviations from the final goal. The capability-based reconfiguration method exhibited low computational time, scaling linearly with the number of CSs, thereby improving the system's scalability. The resilience scenario results further demonstrated that incorporating both stand-in and stand-by redundancy mechanisms enhances the resilience of these complex systems.

Keywords: System of systems (SoS), Modeling and Simulation (M&S), Hypergraph (HG), Stochastic systems, Supervision, Resilience