systems

MDPI

Article

An Integrated Method for Selecting Architecture Alternatives and Reconfiguration Options Towards System-of-Systems Resilience

Zhemei Fang [1,2,*], Hang Li [3] and Dazhi Chen [1,2]

[1] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China; m17803865869@126.com
[2] National Key Laboratory of Science &Technology on Multi-Spectral Information Processing, Huazhong University of Science and Technology, Wuhan 430074, China
[3] Beijing Institute of Aeronautical Systems Engineering, Beijing 100076, China; hanglee_1987@smail.hust.edu.cn
* Correspondence: zmfang2018@hust.edu.cn

Abstract: Delivering persistent values in a dynamic environment is a challenging but imperative capability for a system-of-systems (SoS). Practitioners in the SoS and defense domains are exploring the benefits of the operational-level reconfiguration strategies via new operational concepts such as mosaic warfare. However, an architecture design that allows reconfiguration is also a crucial task, but has not yet received adequate attention, not to mention accounting for the mutual impact between architecture design alternatives and reconfiguration options. Therefore, this paper proposes an integrated method that can select the architecture with a specific inherent structure in the design phase that supports dynamic reconfiguration during the operational phase. This method firstly builds a structural framework that connects architecture design and reconfiguration, and identifies the enablers for SoS architecture reconfiguration. After developing an SoS effectiveness evaluator, the method constructs an integrated multi-objective formulation for the initial architecture selection and reconfiguration process, and provides a solution algorithm based on a fast non-dominated sorting genetic algorithm. An application to an air and missile defense SoS illustrates the effectiveness of the proposed method. The generated Pareto optimal set of solutions that have non-dominated recoverability and survivability provide useful decision support for SoS composition and initial architecture configuration, based upon which an SoS can also respond effectively to disruptions by computing the reconfiguration decisions.

Keywords: architecture alternatives; effectiveness evaluation; system-of-systems; reconfiguration; resilience

check for updates

Academic Editors: Michael Henshaw and Siyuan Ji

Received: 5 December 2024
Revised: 25 December 2024
Accepted: 26 December 2024
Published: 27 December 2024

Citation: Fang, Z.; Li, H.; Chen, D. An Integrated Method for Selecting Architecture Alternatives and Reconfiguration Options Towards System-of-Systems Resilience. *Systems* **2025**, *13*, 9. https://doi.org/10.3390/systems13010009

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Although the system-of-systems (SoS) has been studied for over two decades, the design and development of a well-performing SoS that delivers persistent values in a dynamic and uncertain environment remain a significant challenge [1]. Either strategic changes of missions and technologies or operational changes of system aging, malfunction and damage could lead to unexpected SoS failures. The unique traits of an SoS, such as autonomy, belonging, and connectivity, further aggravate the difficulty of persistent value delivery. For example, a component system leaving an SoS might bring a disaster to an SoS. Many organizations and researchers have recognized the importance of building non-functional attributes or "ilities" (e.g., flexibility, adaptability, resilience) [2] into SoSs

to cope with the uncertain and unpredictable changes. The "ilities" can be considered as a proactive design that enables rapid reactive strategies in an uncertain operational environment, with the goal of maintaining the vitality of an SoS.

Resilience as a salient "ility" has attracted considerable attention in the past few decades. Resilient systems are expected to sustain and recover functions or services in the face of unexpected and unknown disruptions [3]. The resilience-related topics have received heated discussion across a variety of disciplines, including transportation systems, infrastructure systems, ecological systems, defense systems, to name a few. Researchers have devoted large effort to develop methods for resilience quantification and resilience enhancement.

To achieve a sustainable SoS, the resilience enhancement strategies encompass both the design phase and the operational phase (i.e., runtime). Many defense programs propose reconfiguration as a promising solution in operational phase. For example, the key idea behind mosaic warfare, proposed by the U.S. Defense Advanced Research Projects Agency (DARPA), is to increase the agility, resilience, adaptability and survivability of U.S. forces through swift configuration and reconfiguration of smaller-sized functional units [4]. The U.S. Navy is exploring a similar concept of distributed maritime operations through studies conducted by the Naval Postgraduate School (NPS) [5]. The essential principle behind these efforts is to develop the internal capability of an SoS, allowing reconfiguration towards resilience instead of external actions such as repair and replenishment; this is especially important for expedition warfare and time-critical warfare.

An SoS architecture that supports reconfiguration during the operational phase must include specifically designed features such as functional redundancy, adequate functional capacity, and component redundancy, among others, during the design phase. An architecture defines the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time. As structured by architecture frameworks such as the Department of Defense Architecture Framework (DoDAF), the architecture design determines the system selection, connection pattern, and operational process, among others. When it comes to the operational phase, reconfiguration refers to re-sequencing operational flow, re-connecting physical systems, re-allocating systems to activities, etc., in the case of external and internal changes.

Although reconfiguration has been proposed as an important principle for designing resilience [3], the definition, enablers, and implementation strategies of reconfiguration in SoSs, especially combat SoSs, remain vague. Plus, the SoS architecture selection methods do not usually account for the operational-time reconfiguration options, while the reconfiguration optimization methods for combat SoSs often make an architecture hypothesis in advance. There is a lack of an integrated method that selects architecture alternatives and reconfiguration options together and suggests a proper architecture that can facilitate optimal reconfiguration strategies for better resilience.

Therefore, this paper contributes to the SoS and defense domains by proposing a method that can select architecture alternatives and reconfiguration options in an integrated manner towards enhanced SoS resilience. This method defines the reconfiguration at the architecture level and identifies the enablers for SoS architecture reconfiguration. Meanwhile, the method develops an integrated formulation for the initial architecture selection and reconfiguration process with an SoS effectiveness evaluator using a functional dependency network, and employs an intelligent optimization algorithm to solve the problem. The goal of the method is to support SoS decision-makers in designing resilient SoS architecture at the design phase by exploring the possible outcomes of reconfiguration.

This paper consists of five sections in total. Section 2 reviews the existing literature associated with SoS resilience and reconfiguration strategies. Section 3 describes the proposed quantitative reconfiguration framework. This section begins with the problem

description regarding SoS architecture configuration, reconfiguration enabler identification, effectiveness evaluator, and the reconfiguration process. This section further develops the mathematical formulation and corresponding solution algorithm based on the problem description. The next section applies the proposed method to a naval-integrated air and missile defense SoS problem and demonstrates the effectiveness of the method. The final section concludes the paper and indicates future research.

2. Literature Review

2.1. System-of-Systems Architecture Selection

The ISO/IEC/IEEE 42010:2011 (Systems and Software Engineering—Architecture Description) [6] defines architecture as "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution". Researchers in the SoS domain [1,7,8] follow this definition and consider SoS architecture as an allocated architecture generated by allocating a particular functionality from a functional architecture to the physical resources of a physical architecture. In order to describe an architecture more comprehensively and formally, architecture frameworks have been developed in both the defense and industry sectors. Commonly used architecture frameworks include the Department of Defense Architecture Framework (DoDAF), Ministry of Defense Architectural Framework, the Open Group Architecture Framework, and the Unified Architecture Framework. Guided by the architecture frameworks, along with relevant decision support methods and tools, the output of architecture design becomes a set of description models (e.g., operational activity model, state-transition model) that illustrate the capability mission, operational tasks, services, physical systems, functions, and the inter-relationship among these elements. The variety of elements could yield a large number of alternatives. Selecting out the appropriate alternatives based on the mission requirements and all sorts of constraints is the key task of SoS architecture selection and associated methods can refer to the survey paper [1].

2.2. System-of-Systems Resilience

Resilience, first introduced to the academic research by Holling [9], is a multi-faceted capability that encompasses avoiding, absorbing, adapting to, and recovering from disruptions [3]. Madni and Jackson [3] proposed a conceptual framework for resilience engineering that covers disruptions, system attributes, methods, and metrics. This is consistent with the recent research focuses in resilience engineering. Based on our understanding, the current resilience literature mostly concentrates on one or a few of the following aspects—developing resilience metrics, building models for calculating resilience, and developing resilience-enhanced strategies.

A variety of resilience metrics have been developed in the literature, but there is no consistent one yet. While some studies aim for a generic metric of quantifying resilience across disciplines [10], the majority have developed resilience metrics for a particular type of system, such as infrastructure systems [11] and transportation systems [12]. In the military and defense domain, a resilient system is expected to be trusted and effective in a wide range of mission contexts, be easily adapted to many others through reconfiguration and/or replacement, and have predictable, graceful degradation of function [13]. It is often equivalent to survivability in the aerospace and military context [14]. Due to the fact that a large body of SoS research revolves around the aerospace and defense context, many scholars have developed resilience metrics for combat SoSs that are also applicable to other fields. For instance, Tran et al. [15,16] proposed a resilience metric for command and control architecture based on information entropy-based battlespace awareness, and later developed an improved metric that accounts for the repeated disruptions and performance

volatility that commonly exist on a battlefield. Watson et al. [17] recognized the interfaces as the most potent SoS leverage point and proposed a new resilience metric based on graph theory to compare SoS architectures. Chatterjee et al. [18] developed a bio-inspired metric based on ecological network analysis that can help balance SoS resilience and affordability. Uday et al. [19] indicated that instead of using a single metric for evaluating SoS resilience, it would be more practical to use a set of System Importance Measures (SIMs) that not only assess the impact of disruptions on SoS performance but also evaluate the usefulness of the mitigation strategies. A few researchers have developed resilience metrics purely for combat SoSs. For example, Jiang et al. [20] proposed a combat SoS resilience index based on combat effectiveness considering the number of combat loops, combat capability, and time efficiency.

In addition to metrics development, researchers have employed a variety of modeling methods to support resilience quantification, such as system dynamics [17], agent-based modeling [19], network theory [21], Bayesian network [22], and Petri net [23]. Regarding the resilience enhancement strategies, Madni et al. [3] and Jackson et al. [24] indicated 14 representative resilience heuristics, namely functional redundancy, physical redundancy, reorganization, human backup, human-in-loop, predictability, complexity avoidance, context spanning, graceful degradation, drift correction, neutral state, inspectability, intent awareness, and learning/adaptation. However, the suggested design heuristics are qualitative descriptions and lack a quantitative framework to support informed decision-making. Considering the characteristics of SoS, this paper plans to develop an architecture-level reconfiguration framework to enhance SoS resilience.

2.3. Reconfiguration Strategies

Reconfiguration strategies have been used in various domains. Wang et al. [25] modeled the post-disruption reconfiguration strategy as direction reversing in road networks and line switching in power networks. The authors obtained system optimal dynamic traffic assignment and optimal power flow distribution. Zhao et al. [26] considered reconfiguration in manufacturing systems as a combination of reassignment and replacement, in which reassignment refers to changing the positions of components without changing the component reliability. The authors obtained optimal reconfigurations with the objectives of maximum system reliability and minimum reconfiguration cost. Peiravi et al. [27] proposed a universal redundancy strategy that can change the system structure at any time by inserting or removing redundant components individually or simultaneously. The strategy can help select the optimal reconfiguration instant, along with the optimal numbers of components to be activated or inactivated for each subsystems. Chen et al. [28] considered reconfiguration as a re-arrangement of satellites in a constellation by the addition or subtraction of satellites and orbital maneuvering to achieve the desired coverage. As a summary, most reconfiguration problems have been treated as allocation/assignment problems, although the mathematical formulations demonstrate significant differences.

In the SoS context, a number of researchers have explored the potential benefits of reconfiguration, and can be categorized into two groups. One group employs network theory to explore reconfiguration strategies. For example, Tran et al. [16] employed rewiring and preferential attachment in network theory as reconfiguration strategies to achieve system resilience. The authors used simulation to compare different reconfiguration strategies. Sun et al. [29,30] proposed a cooperative reconfiguration model for unmanned swarm SoSs that allows two nodes with same function in two kill chains to work as a substitution pair. The other group employs operational–physical architecture allocation to examine the reconfiguration policies. For instance, Uday and Marais [31] considered reconfiguration as pro-active function-system re-tasking under “stand-in redundancy”. Fang et al. [32] pro-

vided an approximate dynamic programming (ADP) method to conduct function-capability re-allocation for achieving agile response under disruptions. Lee et al. [33] proposed a reconfiguration method for SoSs that develops a knowledge base of adaptation strategies at the design time while generating particular adaptation strategies when receiving disruptions during the run time. Interestingly, an early study by Siddiqi and De Weck [34] explored the relationship between a system's architecture and reconfigurability and proposed several useful principles and guidelines for design for reconfigurability in the early design phase. Although the study focused on the system level, it offers us a deeper understanding of SoS reconfigurability.

Reconfiguration is a promising method for improving SoS resilience and vitality, especially given the current popularity of software-defined equipment in the defense and aerospace domain. However, many questions still remain. For example, what are the optimal reconfiguration decisions for maximizing the SoS capability, what kind of architecture is more suitable for reconfiguration, and what proactive strategies are needed to realize the optimal reconfiguration? This paper aims to improve SoS resilience by developing an integrated architecture selection method to support design-time architecture selection and run-time reconfiguration decision-making.

3. Integrated Method for SoS Architecture Selection and Reconfiguration

3.1. SoS Architecture Configuration

SoS architecture design in this paper concerns DoDAF-based architecture modeling as well as design decision-making that is often neglected in architecture development. This paper integrates architecture design decision-making at the design time and reconfiguration decision-making when faced with disruptions at the operational time to support resilient SoS architecture design. Both types of decisions are governed by the DoDAF-based architecture models that are developed following the major steps of mission analysis, operational architecture, physical architecture, and synthetic (i.e., allocated) architecture development. Figure 1 describes the architecture design decision types that are often needed to complete the DoDAF models, responding to the four major steps. The most common design decisions include selecting/connecting operational activities for operational architecture, selecting/connecting systems for physical architecture, and allocating systems to activities for synthetic architecture, as shown in the red dashed box in Figure 1.

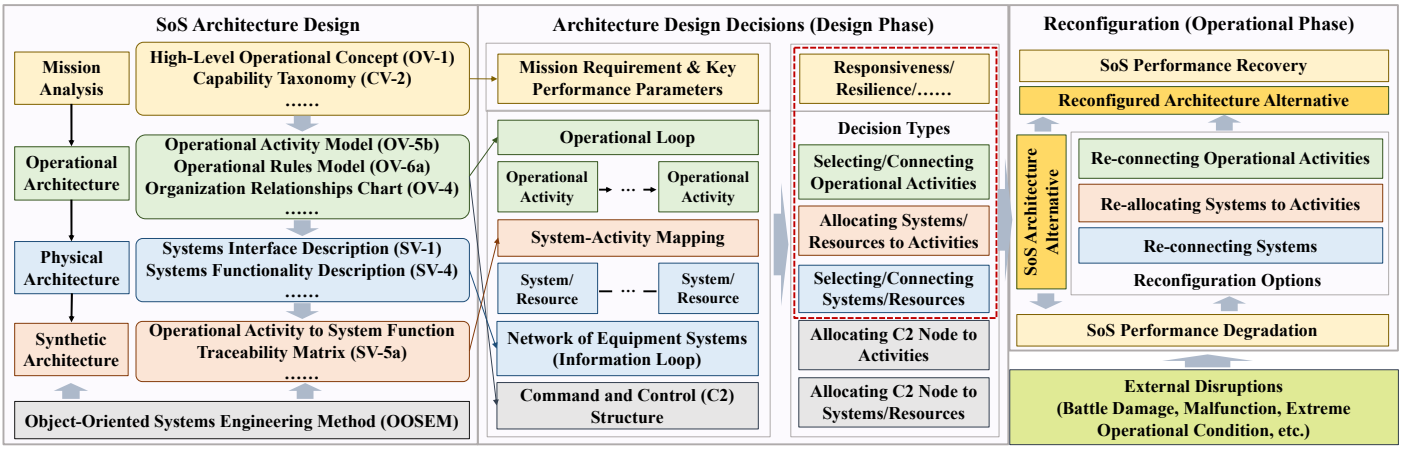


Figure 1. SoS architecture and potential reconfiguration options analysis.

3.2. Architecture Reconfiguration Process and Enablers

Consistent with the decisions at the design time, the reconfiguration decisions at the operational phase primarily considers re-arranging the architecture, including re-

connecting operational activities, re-connecting systems, and re-allocating systems to activities when disruptions occur and performance degrades, as illustrated in Figure 1. The reconfiguration decisions are constrained by the reconfigurability of the designed architecture and are expected to recover SoS performance responsively.

An architecture reconfiguration example is illustrated in Figure 2. If disruptions occur to system Sys2 that consist of sense (S in Figure 2), decide (D in Figure 2), influence (I in Figure 2, referring to firing/striking) functions and fail Sys2, the performance of function/activity FcnS2 would be degraded. In this case, the surviving systems can be re-allocated to the operational architecture to recover partial SoS capability. Meanwhile, the connections between systems might change accordingly to optimize the information exchange process.

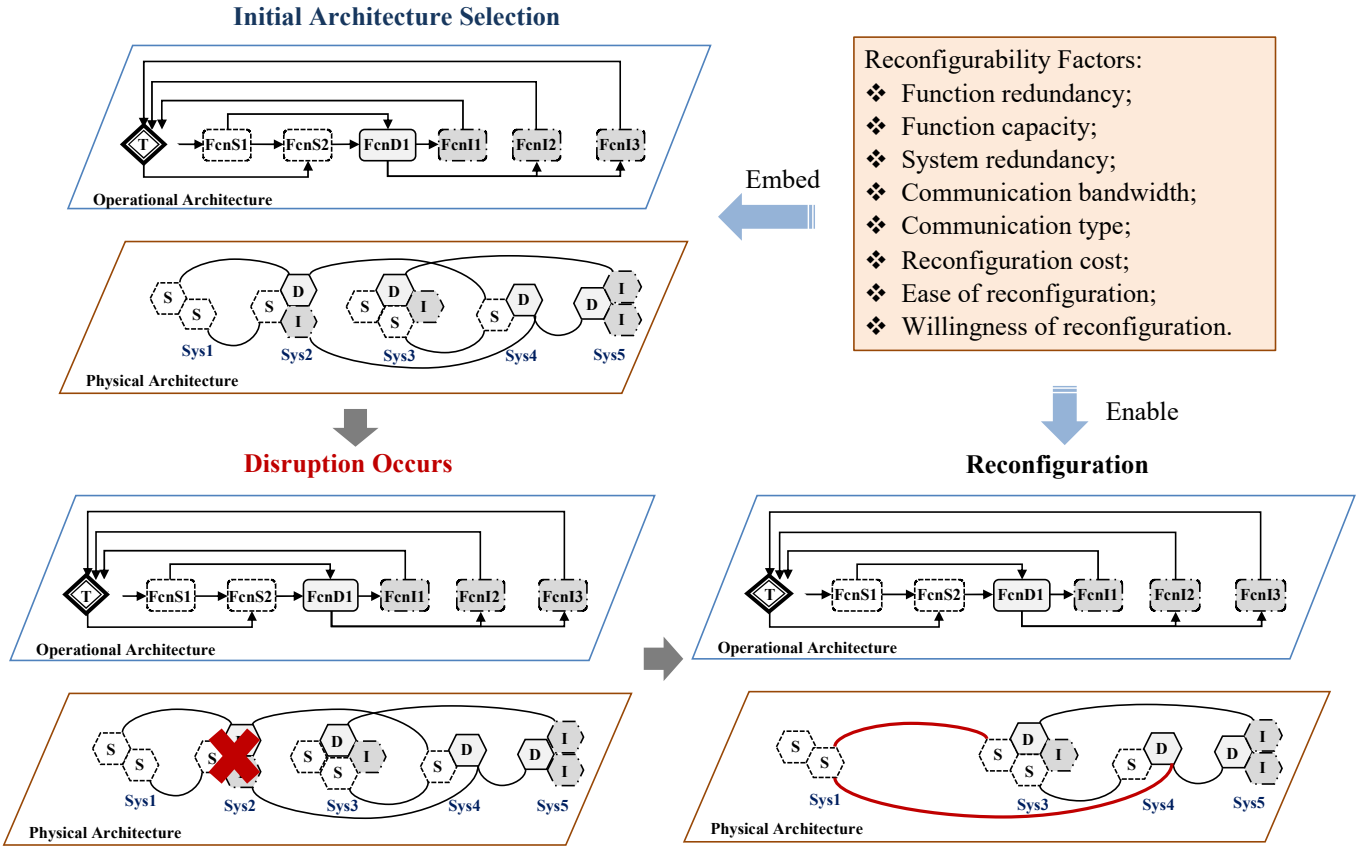


Figure 2. SoS architecture reconfiguration process.

The reconfigurability of the initial architecture is a prerequisite quality for conducting SoS architecture reconfiguration. Many factors can influence reconfigurability, such as function redundancy, function capacity, system redundancy, communication bandwidth, communication type, reconfiguration cost, ease of reconfiguration, and willingness of reconfiguration.

Function redundancy indicates the number of alternative ways in an SoS to perform a particular function. For example, long-range detection of an air threat can be performed by early warning aircraft, fighters, patrol aircraft, ground-based radar, etc. Function capacity is a system-level ability that affects the number of functions/activities to be supported simultaneously by a particular system. For instance, a phased array radar could support the target searching function and target tracking function at the same time. System redundancy refers to the number of the same backups for a system in an SoS; this is one of the most commonly applied methods for reliability and resilience.

Communication bandwidth influences the reconfiguration because different activity-system mapping and system–system connections differ in their requirements for bandwidth resources. Communication type determines whether two systems can be re-connected. If two systems are equipped with different types of communication links (e.g., Link 16, satellite communication), they cannot be connected during reconfiguration.

Reconfiguration costs affect reconfigurability from the perspective of economic constraints. Ease of reconfiguration that depends on the physical and control structure of component systems can influence the rapidness of the reconfiguration process. The modular open system approach (MOSA) that has been advocated again by the U.S. DoD explains the two important prerequisites for the ease of reconfiguration—modular design and open standard. Willingness of reconfiguration is a special SoS consideration due to the operational and managerial independence of the constituent systems. If a participant refuses to reconfigure the current structure, the reconfiguration process affected by this participant cannot proceed. This paper does not plan to include all the enablers because some factors impact the reconfiguration in similar ways and it is difficult to explain the results when too many intertwined factors are involved.

3.3. SoS Effectiveness Evaluator

SoS resilience evaluation requires proper resilience metrics as well as an effective method to assess SoS operational effectiveness. We propose a three-layer SoS effectiveness evaluator based on the functional dependency network analysis (FNDA) method [35,36], as illustrated in Figure 3. SoS effectiveness herein is a representative utility index for evaluating the overall functional capability of a given SoS under a specific condition from a decision-maker's perspective. It is a prerequisite for an SoS resilience evaluator.

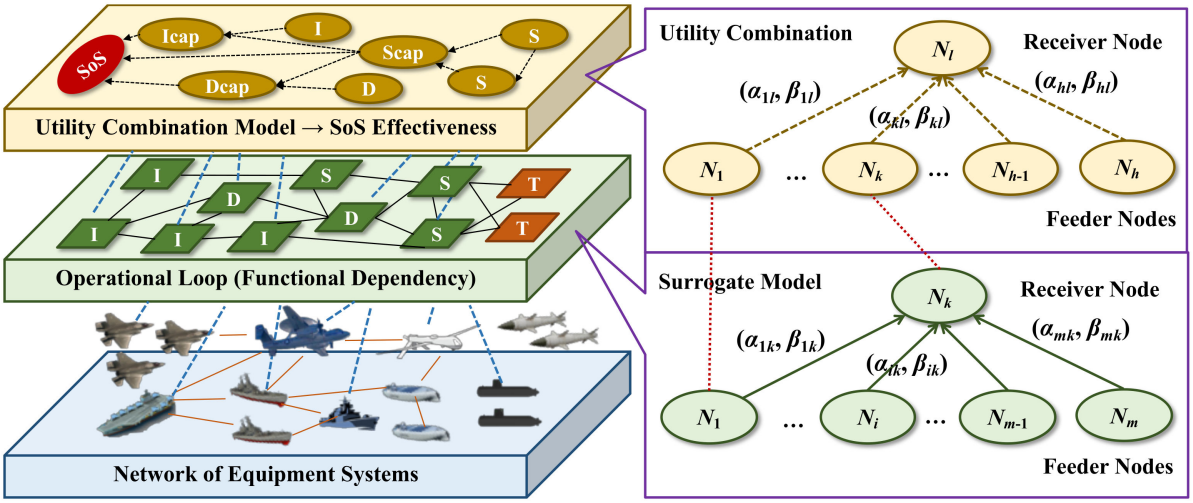


Figure 3. Structure of SoS effectiveness evaluator.

As shown on the left of Figure 3, the bottom layer is the network of equipment systems (reflecting the physical architecture in Figure 2), which yields combat performance through operational activities (e.g., detect, classify, command, control, strike) in the mid-layer (reflecting operational architecture in Figure 2). The connection between the bottom layer and the mid-layer are the functions provided by the equipment systems. The output of the mid-layer includes various SoS performances, such as task completion time and ratio of neutralized targets. While the performances vary in units and scales, the upper level aims to aggregate the values of different performance indicators into one representative indicator based on decision-makers' preferences.

The FDNA and its extensions are typically used as either surrogate models or utility combination models. While the original FDNA method focuses on the utility perspective, the extended methods regarding SoS capability (i.e., operability, effectiveness), often consider themselves as surrogate models that can partially replace high-resolution simulation models in the early design phase. However, the current research does not clearly distinguish between these two perspectives, leading to concept confusions and method misuses. In fact, when using a utility combination model, the utility or effectiveness of each node should be computed based upon the operational output information (e.g., ratio of detected targets) instead of the operational process information (e.g., location and speed of a target). In this paper, we adopt the FDNA method as both the utility combination model and surrogate model. We apply the surrogate FDNA model to the mid-layer and bottom layer, as shown on the right bottom of Figure 3, in which the ellipses represent operational activities in the mid-layer that are supported by the physical systems in the bottom layer. At the upper level, we use the FDNA method as the utility combination model, in which the values of the leaf nodes are generated by the mid-layer and the non-leaf nodes are virtual effectiveness nodes (e.g., detection capability).

The mathematical formulation of the regular FDNA method [35] with an extension regarding the self-effectiveness (SE) [36] is given as in (1). The right bottom network in Figure 3 is an example. The effectiveness O_k and SE value of each node are both normalized between 0 and 100. The SE values of the virtual nodes in the utility combination model should all be set to 100. The effectiveness O_i of the leaf nodes are equal to their self-effectiveness values SE_i.

$$O_k = \min\left(\sum_{i=1}^{m} \frac{O_i \alpha_{ik} \delta_{ik} + SE_i(1-\alpha_{ik})}{m}, \min(O_1 + \beta_{1l}, \ldots, O_i + \beta_{ik}, \ldots, O_m + \beta_{mk})\right), \quad (1)$$

$$0 \le \alpha_{ik} \le 1,\ \ 0 \le \beta_{ik} \le 100,\ 0 \le \delta_{ik} \le 1,\ 0 \le SE_i, O_i, O_k \le 100, \quad (2)$$

In a two-node pair (i.e., m = 1), the FDNA method is a piece-wise linear parametric model using two dependency defining parameters—strength of dependency (SOD) α_{ik} and criticality of dependency (COD) β_{ik}. The SOD defines the fraction of the effectiveness of node k that depends on the effectiveness of node i. The COD defines the remaining effectiveness of node k when the effectiveness of node i is equal to zero. The determination of the dependency parameters in the surrogate model and utility combination model are different. In the surrogate model, the SOD and COD parameters rely significantly on the actual operational process. For instance, the threat assessment function in command and control is likely to have a higher dependency level on radars with more precise detection. A new parameter δ_{ik} is introduced to capture the operability of the communication links. A value of 1 means that the link works perfectly while 0 indicates a complete failure of the communication link. In the utility aggregation model, the SOD and COD parameters largely depend on the decision-makers' preferences. For instance, if the feeder nodes are search capability and classification capability, and the virtual receiver node is overall detection capability, the decision-makers determine the importance of search capability and classification capability.

In a multi-feeder, with one receiver case as in Figure 3, the effectiveness of the receiver node O_k is an output impacted by all the feeder nodes and the dependency parameters. It selects out the smaller of the two effectiveness values caused by SOD and COD, respectively. The SOD part employs the weighted average method while the COD part employs the weakest link rule to select the value with the largest negative impact.

3.4. Mathematical Formulation for Reconfiguration

The formulation structure of the SoS architecture design problem with reconfiguration options is illustrated in Figure 4. Given a set of determined but probably incomplete DoDAF based architecture models, the architecture design decision model and reconfiguration decision model can, on one hand, use the provided information, and on the other hand, eventually help develop comprehensive DoDAF models. The focus of Figure 4 is developing and solving the design decision model that requires an appropriate SoS performance evaluator, in addition to the DoDAF models. Plus, as a novelty, the model also considers reconfiguration decisions in advance in the architecture design process. In the reconfiguration decision model, multiple scenarios concerning different forms of disruptions would require different reconfiguration strategies to achieve the desired performance, based upon the reconfigurability of the generated architecture design.

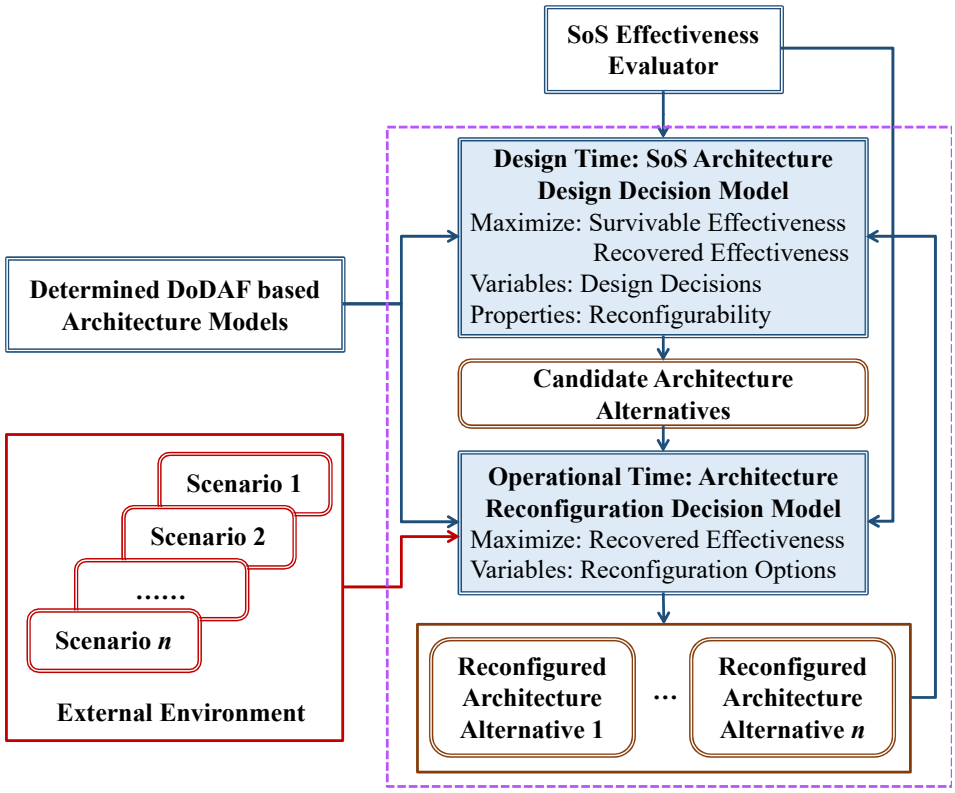


Figure 4. SoS architecture design problem with reconfiguration options.

The current literature offers numerous methods for defining the resilience metrics, based upon the performance (or utility, effectiveness) change curves as shown in Figure 5. A variety of curves have been explored for different types of system. In a combat SoS facing enemy attacks, we make several assumptions: (1) once a system or a functional module receives attacks, the system or functional module stops functioning immediately; (2) the constituent systems within an SoS are able to reconfigure themselves while providing the original functions continuously; (3) the new configuration starts to work when all the reconfiguration procedures are completed. The three assumptions respectively explain the plummeting effectiveness, the stable effectiveness during reconfiguration, and the perpendicular effectiveness rise in Figure 5.

Considering resilience as a combination of survivability and recoverability, we develop two objective functions for this problem. The mathematical expressions are written as in (3). One is to maximize the remaining SoS effectiveness O_n^{disrup} when attacked systems are completely destroyed, implying the essence of survivability. The other is to maximize

the difference between SoS effectiveness after reconfiguration O_n^{reconf} and after disruptions O_n^{disrup}, indicating the recoverability part of resilience. The time for taking reconfiguration actions also matters, but this study considers it as constraints instead of incorporating it into the resilience metrics.

$$\begin{cases} \max\limits_{y_m, x_{ik}^{init}, x_{ik}^{reconf}} E\left[O_n^{disrup}\right] \\ \max\limits_{y_m, x_{ik}^{init}, x_{ik}^{reconf}} E\left[O_n^{reconf} - O_n^{disrup}\right] \end{cases} \tag{3}$$

Although the reconfiguration process concerns both the decisions of system-activity re-allocation and system–system re-connection, the decision variables only include system-activity re-allocation x_{ik}^{reconf} that indicates the degree that system $i \in I$ supports activity $k \in K$. Systems that are allocated to linked activities are assumed to be connected automatically. The other decision variables include the binary variable y_m for initial system selection and architecture allocation x_{ik}^{init} that can both influence the reconfiguration decision options. All decision variables impact the objective via changing the self-effectiveness values of the activity nodes. The calculation of O_n^{disrup} and O_n^{reconf} uses (1) repeatedly until reaching the sink node. The initial self-effectiveness values are based on decision variable y_m and x_{ik}^{init}, as well as $unit_fc_{ik}$ that refers to the function provided for unit support level if allocating system i to activity k.

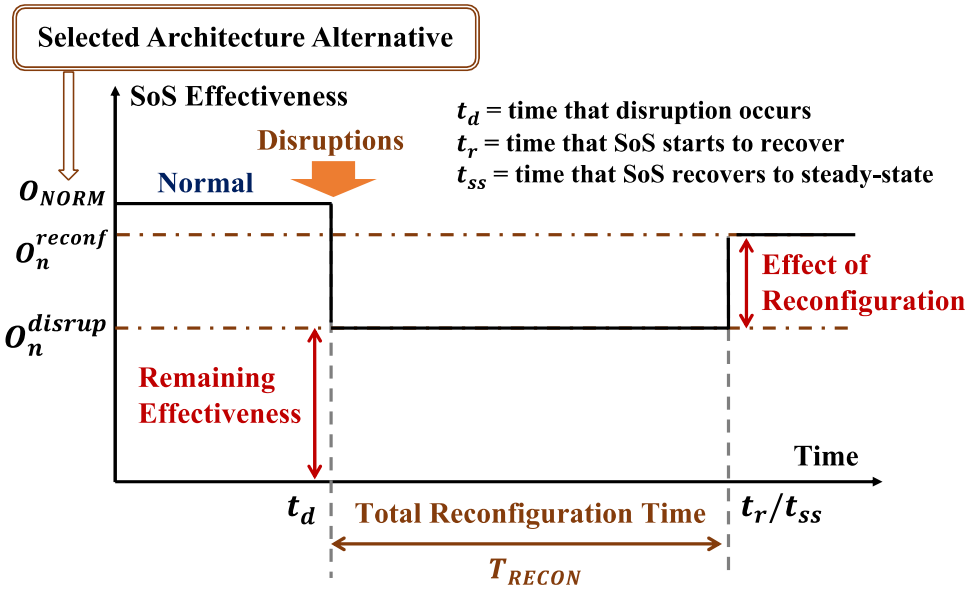


Figure 5. SoS effectiveness change process during disruption and reconfiguration.

The objective functions are subject to the following constraints. The first three depend on reconfigurability while the fourth determines reconfigurability. That is, during the initial architecture design, different potential systems vary in function capacity, communication bandwidth, and reconfiguration time; the SoS architect needs to select appropriate systems so that the reconfiguration strategy might be more effective and lead to better resilience. We currently select only three of the reconfigurability enablers, while others can be considered in the future.

Function capacity. The number of activities that system i can support is constrained by this system's function capacity FC_i. Both the initial architecture configuration and reconfiguration are constrained by this reconfigurability enabler. Function capacity here is a normalized abstract value. A concrete example would be the maximum target tracking

capacity (e.g., 20 targets). This value can be converted to a normalized value between 0 and 100 (e.g., tracking capacity of 20 is normalized to 90 and $unit_fc_{ik}$ is 4.5). That is, $FC_i = 90$. If a radar system i is assigned to support 10 target tracking activities ($K = 10$), it does not violate the following constraint.

$$\sum_{k \in K} \left(unit_fc_{ik} \cdot x_{ik}^{init,reconf} \right) \leq FC_i, \forall i \in I \tag{4}$$

Communication bandwidth. Whenever a physical system j is selected to perform a particular function/activity l, it consumes some portion of the communication bandwidth, depending on the support level, for exchanging information with adjacent systems i that conduct connected activities k. The total amount of consumed bandwidth cannot exceed the limit BW_j.

$$\sum_{i \in I \& i \neq j} \sum_{k \in K} \sum_{l \in K} \left(unit_comm_{kl} \cdot x_{ik}^{init,reconf} \cdot x_{jl}^{init,reconf} \right) \leq BW_j, \forall j \in I \tag{5}$$

Ease of reconfiguration. We consider the ease of reconfiguration as the reconfiguration time T_{RECON}, as shown in Figure 5. The reconfiguration process refers to adding new connections between two nodes while disconnecting existing links between two nodes. The total reconfiguration time cannot exceed a pre-determined threshold RT^{upp}. The "logic" function is to convert the integer values to the logicals. The following inequation depends on the assumption that the reconfiguration actions are in series rather than in parallel. When the supportive techniques for reconfiguration become more developed, simultaneous reconfiguration actions all-at-once will be more likely to occur.

$$\sum_{i \in I} \sum_{k \in K} \begin{pmatrix} time_{ik}^{conn} \cdot \text{logic}\left(x_{ik}^{reconf} \right) \cdot \left(1 - \text{logic}\left(x_{ik}^{init} \right) \right) + \\ time_{ik}^{disconn} \cdot \text{logic}\left(x_{ik}^{init} \right) \cdot \left(1 - \text{logic}\left(x_{ik}^{reconf} \right) \right) \end{pmatrix} \leq RT^{upp} \tag{6}$$

Budget for initial acquisition. Systems that vary in function capacity, communication bandwidth, and reconfiguration time have different development costs. The acquisition of systems should be subject to economic constraints.

$$\sum_{m \in M} \left(cost_m^{acq} \cdot y_m \right) \leq BG^{acq} \tag{7}$$

Function availability. Each function is able to perform, that is, each function has at least one physical system to support it.

$$\sum_{i \in I} x_{ik}^{init,reconf} \geq 1, \forall k \in K \tag{8}$$

The explanations of the mathematical notations in this section are listed in Table 1.

Table 1. Explanations of mathematical notations.

Notation	Description
k, l	Index of activities in operational architecture
i, j	Index of systems in physical architecture
m	Index of potential systems
K	Set of activities
I	Set of systems in the architecture
M	Set of all potential systems

Table 1. *Cont.*

Notation	Description
y_m	Binary variable indicating if system m is selected into the architecture
x_{ik}^{init}	Integer variable indicating the support level of system i for activity k for initial allocated architecture
x_{ik}^{reconf}	Integer variable indicating the support level of system i for activity k during reconfiguration
SE_l	Self-effectiveness of activity node l
O_l	Operational capability of activity node l
O_n	Operational capability of final Influence activity node n
$\alpha_{kl}, \beta_{kl}, \delta_{kl}$	Parameters of dependency between activities k and l
$unit_fc_{ik}$	Function provided for unit support level if allocating system i to activity k
FC_i	Function capacity of system i
$unit_comm_{kl}$	Communication bandwidth need for unit support level between activity k and l
BW_j	Maximum bandwidth of system j
$time_{ik}^{conn}$	Re-connection time of allocating system i to activity k
	Disconnection time of de-allocating system i from activity k
RT^{upp}	Upper limit for reconfiguration time
$cost_m^{acq}$	Development/acquisition cost of system m
BG^{acq}	Acquisition budget for designing initial architecture

3.5. Solution Algorithm

The problem with the objective functions in (3) and constraints in (4)–(8) is an optimization problem with multiple non-linear objective functions. Therefore, we employ a classic multi-objective evolutionary algorithm—the Fast Nondominated Sorting Genetic Algorithm (NSGA-II)—to resolve the problem. In addition to the regular single-objective genetic algorithm following the process of encoding, crossover, mutation, and selection, NSGA-II introduced the Pareto-ranking approach for fitness evaluation, the crowding distance for achieving diversity, and a combination of parent and offspring population for maintaining elitist solutions. This paper aims to develop a quantitative, integrated architecture selection and reconfiguration framework rather than proposing an advanced algorithm. Thus, a classic algorithm could be a good fit for the preliminary exploration.

The algorithm includes the following steps: (1) an architecture alternative is first selected in some way (e.g., randomly); (2) random disruptions then attack the SoS under a given architecture; (3) optimal reconfiguration decisions are made for each scenario and each designed architecture; (4) go back to step (1) for another architecture and continue the loop; (5) pick out the desired/optimized architecture for the design phase. The decision variables for initial architecture development include binary variable y_m indicating system selection and initial configuration x_{ik}^{init} that indicates the support level of system i for activity k for initial allocated architecture. Decisions for reconfiguration x_{ik}^{reconf} depend on the initial architecture configuration and the adversary attacks. Thus, we formulate the reconfiguration decision-making in an inner-loop optimization while the initial architecture configuration is in an outer-loop optimization. The initial architecture optimization is a bi-objective optimization problem while the architecture reconfiguration optimization is a single-objective optimization problem. The step-by-step procedures are illustrated in Algorithm 1 and Algorithm 2, respectively. Algorithm 1 employs a typical NSGA-II procedure while Algorithm 2 uses a regular genetic algorithm.

Algorithm 1: Initial Architecture Optimization Algorithm

Set initial hyper-parameters (e.g., crossover rate, mutation rate, maximum generation t_{max})
Create a random parent population P_0 of size N and set generation $t = 0$
Use the fast non-dominated sorting algorithm (Perform non-dominated sorting P_0 to classify individuals into different non-domination levels; Compute the crowding distance for each individual in P_0)
Conduct binary tournament selection, crossover, and mutation to P_0 and create offspring population Q_0 of size N
While the stopping criteria are not satisfied (e.g., $t \leq t_{max}$), do
    Combine parent and offspring population, set $R_t = P_t \cup Q_t$
    For each solution in population R_t
        If the solution satisfies the constraints in (4)–(8) except (6)
            Conduct a random attack to systems within the SoS
            Compute SoS effectiveness after disruptions $f_1 = O_n^{disrup}$
            Implement architecture reconfiguration following Algorithm 2
            Compute SoS effectiveness after reconfiguration O_n^{reconf}
            Calculate $f_2 = O_n^{reconf} - O_n^{disrup}$
        Else if
            Set $O_n^{disrup} = 0$, $O_n^{reconf} = 0$
        End if
        Obtain the fitness values for each solution
    End for
    Use the fast nondominated sorting algorithm and identify the nondominated fronts $F = \{F_1, F_2, \ldots\}$ in R_t
    Calculate crowding distance of the solutions in $F = \{F_1, F_2, \ldots\}$
    Create P_{t+1} by repeating $P_{t+1} = P_{t+1} \cup F_i$ until $|P_{t+1}| + |F_i| \leq N$ and adding the least crowded $N - |P_{t+1}|$ solutions to P_{t+1} if $|P_{t+1}| + |F_i| > N$
    Conduct binary tournament selection, crossover, and mutation to P_{t+1} and create offspring population Q_{t+1}
    Set $t = t + 1$
End while
Obtain Pareto-optimal set

Algorithm 2: Reconfiguration Optimization Algorithm (Single-Objective)

Set initial hyper-parameters (e.g., crossover rate, mutation rate, maximum generation t_{max})
Create a random parent population P_0 for architecture configuration after disruption and set $t = 0$
While the stopping criteria are not satisfied (e.g., $t \leq t_{max}$), do
    For each solution in the population
        If the solution satisfies the constraints in (4)-(8) except (7)
            Calculate SoS effectiveness after reconfiguration O_n^{reconf}
        Else if
            Set $O_n^{reconf} = 0$
        End if
    End for
    Apply selection, crossover, and mutation to generate next generation population P_{t+1}
    Set $t = t + 1$
End while
Return the maximum fitness value

4. Illustrative Example

4.1. Problem Setup

This paper employs a synthetic air and missile defense (AMD) example that mimics the U.S. Naval Integrated Fire Control Counter Air (NIFC-CA) system. As future warfare is moving towards more distributed mosaic warfare, smaller-unit systems, unmanned systems, and reconfigurable systems will be gradually added to the combat SoS. The operational architecture can be generated based on historical experiences, anticipated capability requirements, and expert engineering judgements. This case study builds a simplified operational architecture based on the current NIFC-CA operations [37]. As illustrated in Figures 6 and 7, the activities of F_S1, F_S2, and F_S3 refer to long-range detection, mid-range detection, and low-altitude detection respectively. F_D1 refers to the operational command and control (C2) function in the field, such as the C2 function provided by early warning (EW) aircraft. F_D2 represents the command and control function of the entire combat SoS. F_I1 and F_I2 refer to long-range and mid-range strike functions. The dependency parameters SOD and COD are assumed to be (0.5, 50) between different activities and (0.9, 10) between activities and capability. There are a couple of different methods to obtain the dependency parameters in the surrogate model form, including expert judgment based on marginal effect, simulation data based fitting, etc. Since the goal of this case study is to demonstrate the feasibility of the integrated method, we have simplified the selection of dependency parameters. Thus, we simply employ two mid-values to include the impact of both the feeder nodes and receiver nodes. Regarding the utility combination form, the dependency parameters are subjective values based on decision-makers' preferences.

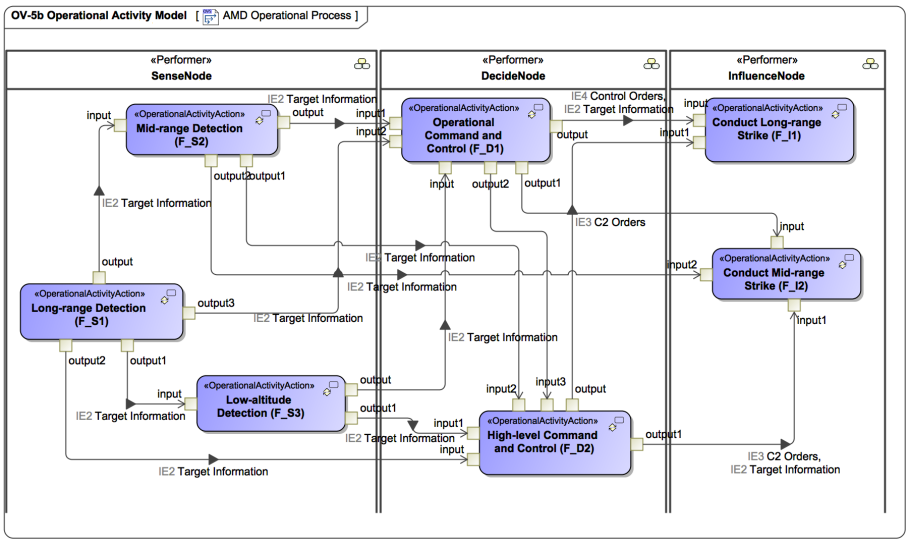


Figure 6. Operational architecture of a synthetic naval AMD SoS.

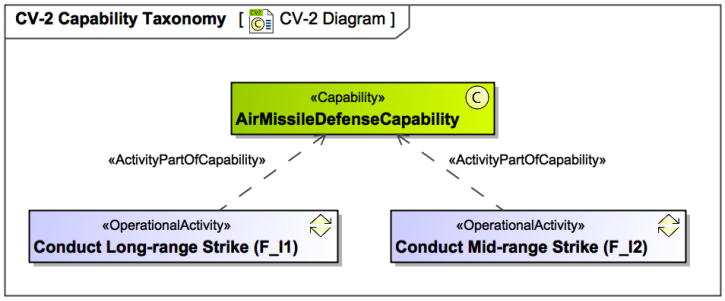


Figure 7. Operational activities mapping to capability.

This case study considers four types of candidate system, namely destroyer, EW aircraft (e.g., E-2D), stealth fighter (e.g., F-35), and unmanned aerial vehicle (UAV). Each type of system can support several activities in the operational architecture, as listed in Table 2. Each type consists of a series of classes (from class A to class E) that vary in technology advancement, acquisition cost, function capacity, communication bandwidth, and ease of reconfiguration. Specifically, from class A to class E, the technological advancement level increases so that the acquisition cost, function capacity, communication capability, and ease of reconfiguration (i.e., less reconfiguration time) also increase. Some of the input data are provided in Tables 2 and 3. The acquisition budget is given as 2000 million dollars and the upper limit of the reconfiguration time is assumed as two seconds. Although the data are currently synthetic, they are able to demonstrate the concept of the proposed framework.

Table 2. Functions provided by different types of system.

System Type	Activities						
	F_S1	**F_S2**	**F_S3**	**F_D1**	**F_D2**	**F_I1**	**F_I2**
Destroyer	0	1	0	0	1	1	1
EW Aircraft	1	1	1	1	0	0	0
Stealth Fighter	1	1	1	1	0	1	1
UAV	1	1	1	1	0	1	1

Table 3. Candidate systems and some of the synthetic input data.

System Type	Class	System Number	Acquisition Cost (m$)	Function Capacity	Communication Bandwidth
Destroyer	Class A	Sys1	200	70	50
	Class B	Sys2	250	80	75
	Class C	Sys3	300	95	100
	Class D	Sys4	350	105	125
	Class E	Sys5	400	120	150
EW Aircraft	Class A	Sys6	100	40	45
	Class B	Sys7	125	50	60
	Class C	Sys8	150	60	75
	Class D	Sys9	175	70	90
	Class E	Sys10	200	80	105
Fighter	Class A	Sys11	80	25	45
	Class B	Sys12	100	35	55
	Class C	Sys13	120	40	65
	Class D	Sys14	140	45	75
	Class E	Sys15	160	55	85
UAV	Class A	Sys16	50	25	30
	Class B	Sys17	70	30	35
	Class C	Sys18	90	35	45
	Class D	Sys19	110	45	50
	Class E	Sys20	130	50	55

If we assume a random attack following uniform distribution occurs to both systems and communications links, when an attack happens to a system, the SE value of this system falls to zero; when an attack occurs to a communication link, the parameter δ_{ik} falls to zero. After the random attacks, the SoS reconfigures itself to recover partial capability.

4.2. Experiments and Results

This case study employs Python 3.6 for the experiment in an environment with AMD Ryzen Threadripper 3970X CPU, 128 GB 2200 MHZ memory, and a Linux operating system.

Regarding the NSGA-II based Algorithm 1, we run the experiments with a crossover rate between (0.5, 1), a mutation rate between (0.01, 0.2), and a population size between (50, 200). With regard to the inner-loop Algorithm 2, we conduct experiments with a mutation rate between (0.01, 0.5), crossover rate between (0.1, 0.5), and population size between (50, 100). The selected parameters are listed in Table 4.

Table 4. Experiment parameters for the algorithms.

Algorithm	Parameters	Values
Algorithm 1 (NSGA-II)	Crossover rate	0.9
	Mutation rate	0.05
	Population size	100
	Maximum generation	100
Algorithm 2 (GA)	Mutation rate	0.05
	Crossover rate	0.3
	Population size	80
	Maximum generation	400

The algorithm takes around 33 h to achieve convergence. Since this method aims to solve the architecture design problem, it is still acceptable. Figure 8 illustrates the obtained Pareto sets at different generations when a system randomly receives attacks and falls to a complete failure. The plot shows clearly how the output reaches the Pareto front from the 25th generation to 100th generation. Since only one system is destroyed, a decent amount of capability value survives. In the 100th generation, if the surviving capability is small (e.g., if the attacked system significantly affects SoS effectiveness), the recovered capability is relatively large on the Pareto front. This means that the optimal reconfiguration strategy can help recover SoS capability as much as it can.

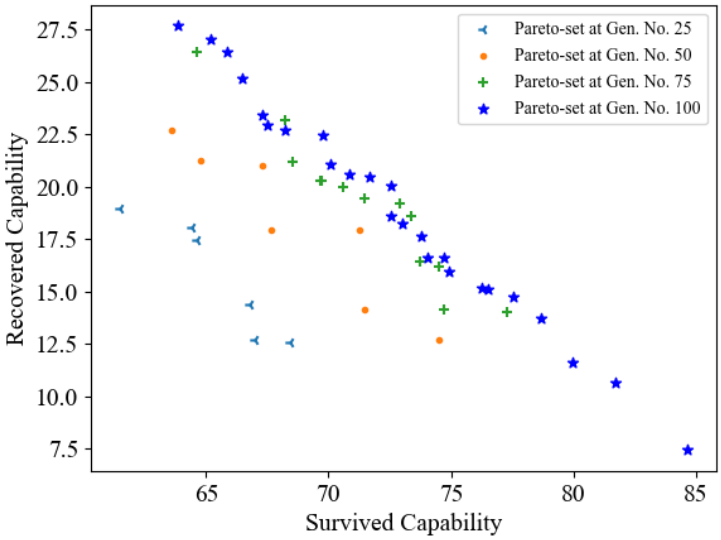


Figure 8. Pareto fronts of the experiment with one system disruption.

If multiple systems receive attacks, the surviving and recovered capabilities at the 100th generation are shown in Figure 9. The overall trend is not so different from the results in Figure 8, except for the lower capability values. Hence, the discussion of the results will focus on the situation with one system disruption.

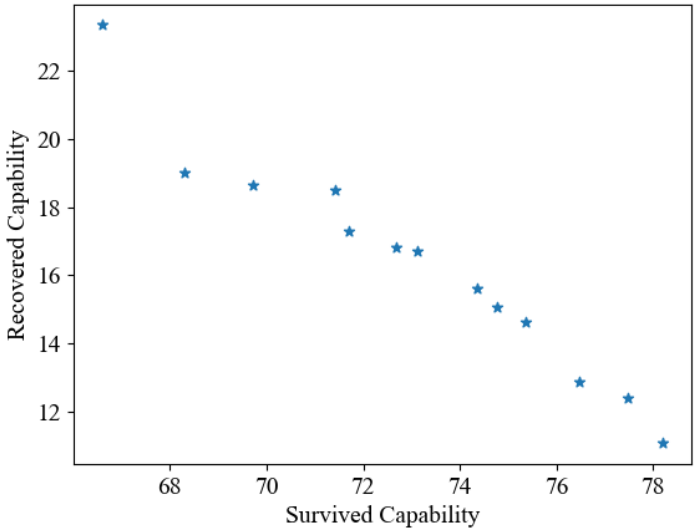


Figure 9. Pareto front of the experiment with disruptions to three systems.

Most of the solutions in the Pareto optimal set of the 100th generation in Figure 8 suggest an initial selection of systems of Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class B, EW Aircraft Class C, EW Aircraft Class D, and EW Aircraft Class E, as illustrated in Table 5. It might seem unreasonable in a real AMD SoS, but it makes sense in the current experiment, because the destroyer and EW aircraft can perform all the required activities and also provide higher function capacity and communication bandwidth. Although it comes with larger acquisition costs, the benefit of more reconfiguration options and high tolerance of failures still makes the destroyer and EW aircraft stand out. In fact, if the acquisition costs of UAVs and fighters in Table 3 are cut in half, Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class C, EW Aircraft Class D, EW Aircraft Class E, Fighter Class D, or Fighter Class E are more likely to be selected into the initial architecture. The UAVs are still not selected due to the communication bandwidth constraint. A lesson learnt here is that the low-cost UAVs need to be way more flexible (i.e., with far less reconfiguration time) or much cheaper, with adequate communication bandwidth, to outperform the giant multi-functional weapon platforms. UAVs might also be favored if the evaluation criteria change, such as the consideration of human safety. Concerning the future trend of mosaic warfare filled with unmanned systems, communication capability could be a major obstacle.

Table 5. Initial architecture selection at the 100th generation.

Number of Solutions	Selected Systems
18 Solutions	(Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class B, EW Aircraft Class C, EW Aircraft Class D, EW Aircraft Class E)
3 Solutions	(Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class C, EW Aircraft Class D, EW Aircraft Class E, Fighter Class E)
2 Solutions	(Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class C, EW Aircraft Class D, EW Aircraft Class E)
1 Solution	(Destroyer Class A, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class C, EW Aircraft Class D, EW Aircraft Class E, Fighter Class D)
1 Solution	(Destroyer Class A, Destroyer Class B, Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class B, EW Aircraft Class C, EW Aircraft Class E)

Additionally, in Figure 8, with the same set of selected systems, each solution lying on the Pareto front of the 100th generation has different initial architecture configurations and corresponding reconfiguration decisions. The SoS decision-makers can then pick

out the desired solution based on their own preference and acceptable level of survived capability. For example, if the acceptable surviving capability is 70 units, then the initial architecture configuration decisions are displayed in Table 6. The numbers in the table indicate the support level of a platform system to a particular activity. The destroyers take care of mid-range detection, command and control, and striking, while the EW aircraft are responsible of long-range detection, mid-range detection, low-altitude detection, and in-field operational command and control. After random attacks to a particular system, the SoS reconfigures itself, and the capability can be increased by 22 units. In this case, although a rational decision-maker might prefer the surviving capability of 85 units with a recovered capability of 7.5 units, a satisfying solution followed by an unexpected capability increase after reconfiguration could also be a good strategy for a combat SoS.

Table 6. An example solution of initial architecture configuration.

System Type	Functions						
	F_S1	F_S2	F_S3	F_D1	F_D2	F_I1	F_I2
Destroyer Class B	0	0	0	0	0	0	1
Destroyer Class C	0	3	0	0	1	2	1
Destroyer Class D	0	0	0	0	4	0	3
Destroyer Class E	0	8	0	0	3	3	1
EW Aircraft Class B	0	2	1	2	0	0	0
EW Aircraft Class C	1	3	1	2	0	0	0
EW Aircraft Class D	3	3	0	0	0	0	0
EW Aircraft Class E	1	3	3	1	0	0	0

Figure 10 demonstrates the Pareto fronts when communication links are attacked and completely ruined. We directly add the failure on the flow between operational activities. That is, multiple communication links fail in the physical architecture. Based on the plot, the solutions on the Pareto front have a fair amount of surviving capability and recovered capability, but lower than the values in Figure 8 due to the functional link disruption. The plot also demonstrates the process of output moves to a better Pareto set after 100 generations. In the Pareto set of the 100th generation, most of the solutions share the same set of initial system mixes that includes Destroyer Class C, Destroyer Class D, Destroyer Class E, EW Aircraft Class C, and EW Aircraft Class E. Again, this is reasonable under the current experiment setup because the fighters and UAVs are not adequately cheap or useful.

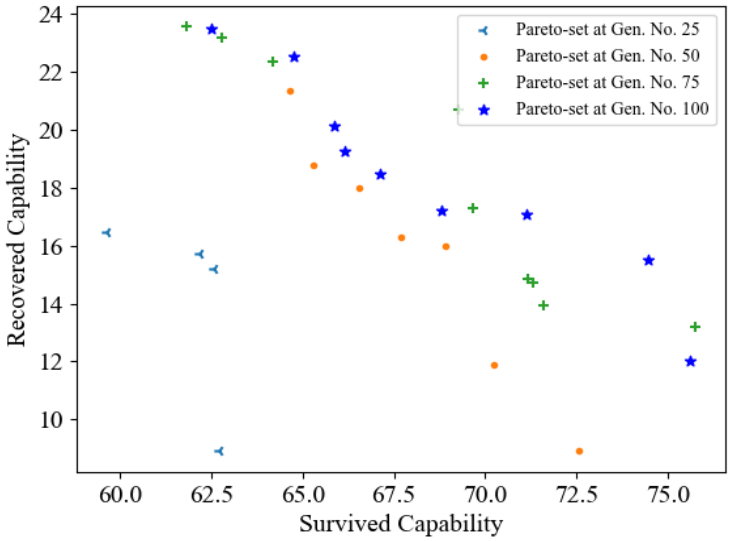


Figure 10. Pareto fronts of the experiment with communication link disruption.

The results in this synthetic example demonstrate that the proposed method is able to provide decision-makers in the defense domain with a quantitative, integrated tool to find the initial architecture selection and configuration that can handle disruptions more effectively, as well as reconfiguration decisions that can help recover more capability. The results also imply the importance of incorporating reconfigurability in the constituent systems and SoSs to achieve higher resilience. When transitioning from the theoretical method to real-world examples, we need to pay careful attention to the normalized values in the mathematical formulations of some reconfiguration enablers (e.g., function capacity). We need to apply techniques like multi-attribute utility theory to convert the concrete parameters to normalized values.

5. Conclusions

Delivering value persistently is a critical requirement for a combat SoS under fierce confrontation. The characteristics of SoSs provide a good opportunity to reconfigure the architecture during failures in operation with inherent features such as functional redundancy and substitution. Such an operation requires an appropriate architecture design at the beginning. This paper develops a quantitative architecture reconfiguration framework to enhance SoS resilience by considering the initial architecture development and architecture reconfiguration in an integrated manner. This framework considers DoDAF-based SoS architecture configuration. The resulting SoS effectiveness is evaluated using an extended three-layer functional dependency network analysis method. The effectiveness evaluator supports the formulation and solution development for optimal SoS architecture reconfiguration. The proposed framework is then applied to a synthetic naval air and missile defense SoS. The results provide decision-makers with a Pareto optimal set of solutions that have non-dominated recoverability and survivability. The decision-makers can then select the SoS composition and initial architecture configuration based on the suggestion and their own preferences and respond quickly to disruptions by computing the reconfiguration decisions.

Future work will focus on two perspectives. One perspective concerns theoretical enrichment from two directions. The first direction focuses on the side of architecture design. We plan to study the architecture reconfiguration enablers in depth, and develop quantitative evaluation metrics for the elements of reconfigurability. The reconfigurability metrics can be validated by the reconfiguration process during the operation phase and validated metrics can guide the architecture design as well. The second direction will concentrate on the reconfiguration process. The focus is on the solution algorithm aspect. We will conduct more thorough experiments to figure out the most suitable intelligent algorithm (e.g., particle swarm optimization, simulated annealing) and explore the approximation techniques to replace the genetic algorithm for the inner-loop reconfiguration process to increase the computational efficiency. Approximate dynamic programming or reinforcement learning might be a promising framework to manage the two-stage problem. However, it is noteworthy that the focus of the proposed method is more on the design phase than the operational phase, which reduces the impact of computational efficiency. The other perspective concerns extending the case study to include larger, real-world datasets to validate the proposed framework in operational scenarios. Additional sensitivity analyses of parameters (e.g., acquisition cost, communication bandwidth) are also necessary to highlight robustness under different scenarios.

Author Contributions: Conceptualization, Z.F., H.L. and D.C.; methodology, Z.F., H.L. and D.C.; validation, D.C. and Z.F.; writing—original draft preparation, Z.F.; writing—review and editing, H.L.; visualization, D.C.; supervision, Z.F.; project administration, Z.F.; funding acquisition, Z.F. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant No. 62103158.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

DURC Statement: Current research is limited to an integrated approach for selecting architecture design alternatives with potential reconfigurable options to achieve better resilience. As an important topic in the field of system architecture design and system-of-systems engineering, the method is beneficial for designing complex systems that are expected to deliver long-term values and does not pose a threat to public health or national security. Authors acknowledge the dual use potential of the research involving defense SoS architecture design and confirm that all necessary precautions have been taken to prevent potential misuse. As an ethical responsibility, authors strictly adhere to relevant national and international laws about DURC. Authors advocate for responsible deployment, ethical considerations, regulatory compliance, and transparent reporting to mitigate misuse risks and foster beneficial outcomes.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fang, Z. System-of-Systems Architecture Selection: A Survey of Issues, Methods, and Opportunities. *IEEE Syst. J.* **2022**, *6*, 4768–4779. [CrossRef]
2. Ross, A.M.; Rhodes, D.H.; Hastings, D.E. Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining System Lifecycle Value. *Syst. Eng.* **2008**, *11*, 246–262. [CrossRef]
3. Madni, A.M.; Jackson, S. Towards a Conceptual Framework for Resilience Engineering. *IEEE Syst. J.* **2009**, *3*, 181–191. [CrossRef]
4. Clark, B.; Patt, D.; Schramm, H. *Mosaic Warfare: Exploiting Artificial Intelligence and Autonomous Systems to Implement Decision-Centric Operations*; Center for Strategic and Budgetary Assessments: Washington, DC, USA, 2020.
5. Popa, C.H.; Stone, S.P.; Aw, E.H.; Teo, C.P.J.; Cai, L.E.; Chong, W.H.; Cline, R.; Hong, J.J.; Roger, K.C.; Lee, W.L.; et al. *Distributed Maritime Operations and Unmanned Systems Tactical Employment*; Naval Postgraduate School: Monterey, CA, USA, 2018.
6. *42010-2011: ISO/IEC/IEEE*; Systems and Software Engineering—Architecture Description. International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE): Geneva, Switzerland, 2011.
7. Levis, A.H.; Wagenhals, L.W. C4ISR Architectures: I. Developing a Process for C4ISR Architecture Design. *Syst. Eng.* **2000**, *3*, 225–247. [CrossRef]
8. Raz, A.K.; Kenley, C.R.; DeLaurentis, D. A System-of-Systems Perspective for Information Fusion System Design and Evaluation. *Inf. Fusion* **2017**, *35*, 148–165. [CrossRef]
9. Holling, C.S. Resilience and Stability of Ecological Systems. *Annu. Rev. Ecol. Syst.* **1973**, *4*, 1–23. [CrossRef]
10. Henry, D.; Ramirez-Marquez, J.E. Generic Metrics and Quantitative Approaches for System Resilience as a Function of Time. *Reliab. Eng. Syst. Saf.* **2012**, *99*, 114–122. [CrossRef]
11. Liu, W.; Song, Z. Review of Studies on the Resilience of Urban Critical Infrastructure Networks. *Reliab. Eng. Syst. Saf.* **2020**, *193*, 106617. [CrossRef]
12. Zhou, Y.; Wang, J.; Yang, H. Resilience of Transportation Systems: Concepts and Comprehensive Review. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 4262–4276. [CrossRef]
13. Goerger, S.R.; Madni, A.M.; Eslinger, O.J. Engineered Resilient Systems: A DoD Perspective. In Proceedings of the Conference on Systems Engineering Research (CSER 2014), Redondo Beach, CA, USA, 20–22 March 2014.
14. Richards, M.G.; Ross, A.M.; Shah, N.B.; Hastings, D.E. Metrics for Evaluating Survivability in Dynamic Multi-Attribute Tradespace Exploration. In Proceedings of the AIAA Space Conference & Exposition, San Diego, CA, USA, 9–11 September 2008.
15. Tran, H.T.; Domercant, J.C.; Mavris, D.N. A System-of-Systems Approach for Assessing the Resilience of Reconfigurable Command and Control Networks. In Proceedings of the AIAA SciTech Forum, Kissimmee, FL, USA, 5–9 January 2015.
16. Tran, H.T.; Balchanos, M.; Domercant, J.C.; Mavris, D.N. A Framework for the Quantitative Assessment of Performance-based System Resilience. *Reliab. Eng. Syst. Saf.* **2017**, *158*, 73–84. [CrossRef]
17. Watson, B.C.; Chowdhry, A.; Weissburg, M.J.; Bras, B. A New Resilience Metric to Compare System of Systems Architecture. *IEEE Syst. J.* **2021**, *16*, 2056–2067. [CrossRef]
18. Chatterjee, A.; Malak, R.; Layton, A. Exploring System of Systems Resilience Versus Affordability Trade-Space using a Bio-Inspired Metric. *J. Comput. Inf. Sci. Eng.* **2021**, *21*, 050905. [CrossRef]

19. Uday, P.; Chandrahasa, R.; Marais, K. System Importance Measures: Definitions and Application to System-of-Systems Analysis. *Reliab. Eng. Syst. Saf.* **2020**, *191*, 106582. [CrossRef]
20. Jiang, J.; Li, J.; Xia, B.; Yang, K. Modeling and Resilience Analysis of Combat Systems-of-Systems Based on Kill Web. In Proceedings of the 17th Annual System of Systems Engineering Conference, New York, NY, USA, 7–11 June 2022.
21. Chen, W.; Li, W.; Zhang, T. Complex Network-Based Resilience Capability Assessment for a Combat System-of-Systems. *Systems* **2024**, *12*, 31. [CrossRef]
22. Hossain, N.U.; Nagahi, M.; Jaradat, R.; Shah, C.; Buchanan, R.; Hamilton, M. Modeling and Assessing Cyber Resilience of Smart Grid Using Bayesian Network-based Approach: A System of Systems Problem. *J. Comput. Des. Eng.* **2020**, *7*, 352–366.
23. Pflanz, M.; Levis, A. An Approach to Evaluating Resilience in Command and Control Architectures. In Proceedings of the 2012 Conference on Systems Engineering Research (CSER), St. Louis, MO, USA, 19–22 March 2012.
24. Jackson, S.; Ferris, T.L. Resilience Principles for Engineered Systems. *Syst. Eng.* **2013**, *16*, 152–164. [CrossRef]
25. Wang, H.; Fang, Y.-P.; Zio, E. Resilience-Oriented Optimal Post-Disruption Reconfiguration for Coupled Traffic-Power Systems. *Reliab. Eng. Syst. Saf.* **2022**, *222*, 108408. [CrossRef]
26. Zhao, J.; Si, S.; Cai, Z. A Multi-Objective Reliability Optimization for Reconfigurable Systems Considering Components Degradation. *Reliab. Eng. Syst. Saf.* **2021**, *183*, 104–115. [CrossRef]
27. Peivari, A.; Nourelfath, M.; Zanjani, M.K. Universal Redundancy Strategy for System Reliability Optimization. *Reliab. Eng. Syst. Saf.* **2022**, *225*, 108576.
28. Chen, Y.; Mahalec, V.; Chen, Y.; Liu, X.; He, R.; Sun, K. Reconfiguration of Satellite Orbit for Cooperative Observation using Variable-Size Multi-Objective Differential Evolution. *Eur. J. Oper. Res.* **2015**, *242*, 10–20. [CrossRef]
29. Sun, Q.; Li, H.; Wang, Y.; Zhang, Y. Multi-swarm-based Cooperative Reconfiguration Model for Resilient Unmanned Weapon System-of-Systems. *Reliab. Eng. Syst. Saf.* **2022**, *222*, 108426. [CrossRef]
30. Sun, Q.; Li, H.; Zhong, Y.; Ren, K.; Zhang, Y. Deep Reinforcement Learning-based Resilience Enhancement Strategy of Unmanned Weapon System-of-Systems under Inevitable Interferences. *Reliab. Eng. Syst. Saf.* **2024**, *242*, 109749. [CrossRef]
31. Uday, P.; Marais, K. Exploiting Stand-In Redundancy to Improve Resilience in a System-of-Systems (SoS). *Procedia Comput. Sci.* **2013**, *16*, 532–541. [CrossRef]
32. Fang, Z.; Liao, J.; Zhou, X. Improving System-of-Systems Agility through Dynamic Reconfiguration. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020.
33. Lee, S.; Baek, Y.-M.; Hyun, S.; Bae, D.-H. Generation of Adaptation Strategies for Dynamic Reconfiguration of a System of Systems. In Proceedings of the 16th International System of Systems Engineering Conference (SoSE), Vasteras, Sweden, 14–18 June 2021.
34. Siddiqi, A.; Weck, O.D. Modeling Methods and Conceptual Design Principles for Reconfigurable Systems. *J. Mech. Des.* **2008**, *130*, 101102. [CrossRef]
35. Garvey, P.R.; Pinto, C.A. Introduction to Functional Dependency Network Analysis. In Proceedings of the Second International Symposium on Engineering Systems, Cambridge, MA, USA, 15–17 June 2009.
36. Guariniello, C.; DeLaurentis, D. Supporting Design via the System Operational Dependency Analysis Methodology. *Res. Eng. Des.* **2017**, *28*, 53–69. [CrossRef]
37. Bravo, T. *The Distributed Air Wing*; Naval Postgraduate School: Monterey, CA, USA, 2014.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.