Res Eng Design (2017) 28:53–69
DOI 10.1007/s00163-016-0229-0

ORIGINAL PAPER

Supporting design via the System Operational Dependency Analysis methodology

Cesare Guariniello[1] · **Daniel DeLaurentis**[1]

Received: 30 March 2015 / Revised: 5 April 2016 / Accepted: 8 April 2016 / Published online: 22 April 2016

Abstract In this paper, we introduce the system operational dependency analysis methodology. Its purpose is to assess the effect of dependencies between components in a monolithic complex system, or between systems in a system-of-systems, and to support design decision making. We propose a parametric model of the behavior of the system. This approach results in a simple, intuitive model, whose parameters give a direct insight into the causes of observed, and possibly emergent, behavior. Using the proposed method, designers, and decision makers can quickly analyze and explore the behavior of complex systems and evaluate different architecture under various working conditions. Thus, the system operational dependency analysis method supports educated decision making both in the design and in the update process of systems architecture, without the need to execute extensive simulations. In particular, in the phase of concept generation and selection, the information given by the method can be used to identify promising architectures to be further tested and improved, while discarding architectures that do not show the required level of global features. Application of the proposed method to a small example is used to demonstrate both the validation of the parametric model, and the capabilities of the method for system analysis, design and architecture.

Keywords Dependencies · Design · Behavioral analysis · System architecture · Operability · Risk · System-of-systems

✉ Cesare Guariniello
cesare.guariniello@gmail.com

[1] School of Aeronautics and Astronautics, Purdue University, 701 W Stadium Ave, West Lafayette, IN 47907, USA

List of symbols

O_i	Operability of node i
O_i^C	Global term of operability of node i due to criticality of dependency
O_i^S	Global term of operability of node i due to strength of dependency
O_{ij}^C	Term of operability of node i due to criticality of dependency from node j
O_{ij}^S	Term of operability of node i due to strength of dependency from node j
SE_i	Self-effectiveness of node i
W^λ	Weight for the term based on criticality of dependency in multiple dependencies
α_{ij}	Parameter associated to the strength of dependency (SOD) between node i and node j
β_{ij}	Parameter associated to the criticality of dependency (COD) between node i and node j
γ_{ij}	Parameter associated to the impact of dependency (IOD) between node i and node j

1 Introduction

As technology keeps evolving, problems in systems engineering are growing increasingly big and complex. Traditional systems engineering methods are not always suitable when designing and architecting systems, and need to be supported by novel methodology, capable of addressing the new challenges. This is particularly true in the context of Systems-of-Systems (SoS), where the constituent systems have, at least in part, operational and managerial independence (Maier 1998; Sage and Cuppan 2001). Moreover, the behavior of most systems designed

and developed nowadays cannot always be directly evaluated based only on the behavior of the individual systems or subsystems. In this paper, we define *complex systems* as entities where interactions between the parts have a fundamental impact on the global behavior, and the effect of dependencies is difficult to ascertain by simple inspection during design (Mane et al. 2011; Nai Fovino and Masera 2006; Hsu et al. 2009). Throughout this paper, we will use the term systems to refer to this class of complex systems. The traditional systems engineering approach often comes short in managing such features. Dependencies between systems are dealt with by means of computationally expensive simulations, complex parametric models whose support functions lack intuitive meaning that may ease their use, or analytical tools that add qualitative traits to data that are sometimes already qualitative, thus decreasing the fidelity of the results. Therefore, new tools and methods are required, to analyze and quantify properties of the system as a whole, and to include considerations about possible emergent behavior when designing or updating complex systems (Dahmann and Smith 2012).

In systems engineering, the first step required to perform the desired analysis involves determining metrics that describe the features and the behavior of the system. When dealing with complex systems, however, metrics related to individual systems do not directly translate to an assessment of the overall behavior. Many authors recognize the importance of holistic high-level metrics (Rhodes et al. 2009; de Weck et al. 2012) and acknowledge the need to include thorough analysis of the impact of dependencies between components in the process of designing, architecting, and planning updates of systems and SoS.

To address some of the limitations of traditional systems engineering when dealing with complex systems, we developed a methodology called systems operational dependency analysis (SODA). SODA is based on previous work by Haimes and Jiang (2001), Haimes et al. (2005), Garvey and Pinto (2009, 2012) and Garvey et al. (2014), with the goal of analyzing operational dependencies between systems, and assessing the impact of such dependencies on the overall behavior of the system or SoS. We model the system as a directed dependency network, where nodes represent the component systems (or subsystems) and the capabilities that the whole entity is required to achieve. The edges connecting the nodes represent the operational dependencies between the constituent parts. An operational dependency means that a certain system needs input (data, material, and/or energy) from another system in order to reach its full operability. Each dependency is modeled with three parameters, one accounting for the strength of the dependency, and two accounting for the criticality of the dependency. The use of a parametric model of the system behavior yields multiple advantages: First of all, the parameters have an intuitive meaning, directly related to the features of the dependency. Thus, they give direct insight into the causes of observed, and possibly emergent, behavior. Parameters also support informed decision making in design and update of systems and SoS architecture, reducing the amount of simulation required. Moreover, parameters may come from various sources, including experiments, expert evaluation, and historical data.

The goal of this paper is to develop a framework to support decision in systems design and architecture. With SODA, we compute the operability of each system as a function of its own internal status, and of the operability of the other systems in the network, based on the topology, and on the features of the dependency. By this way, we can evaluate the impact and propagation of failures and disruptions from the nominal operability, and the effect of architectural design decisions on the global behavior. Based on these results, the user can quantify various metrics of interest, for example, robustness and resilience. Using SODA, designers and decision makers can quickly analyze the operational behavior of complex systems and SoS, and evaluate different architectures under several working conditions. The user can compare architectures based on metrics of interest, trade-off between competing desired features, identify the most promising architectures, as well as the causes of the observed behavior, and discard architectures that lack the requested features. This way, in the early design process, promising architectures can be kept into consideration, and improved based on the information given by the model parameters and the observed behavior, thus supporting the process of concept selection.

After describing the methodology and the meaning of the parameters of the model, we show the steps required to apply the method, and examples of analysis and synthesis with SODA on a synthetic Naval Warfare Scenario SoS.

2 Related work and prior research

2.1 Dependencies and behavioral models

Current practice uses various approaches to model and analyze complex behavior due to dependencies. Response Surface Methodology (Box and Wilson 1951) models the relationship between input and output variables with a polynomial. This methodology was widely modified and improved for years (Khuri and Mukhopadhyay 2010), and evolved into Robust Parameter Design (Taguchi 1986). These techniques result in detailed modeling of systems behavior, but the parameters of this model do not have intuitive meaning (they represent a function, but are not directly related to features of the dependencies), therefore

extensive simulation is required for analysis, and the design space exploration is executed by generating and analyzing all possible designs. Bayesian Networks (Pearl 1988; Moullec et al. 2013) deal with probabilistic dependencies between nodes in an acyclic network and are suitable to analyze interdependencies. However, the regression analysis required to identify the correct conditional probabilities is computationally (or experimentally) expensive and may require excessive time in the case of large networks. Also, these methods give another very detailed model of the systems behavior, but they do not readily provide information on the root cause of observed behavior, so that improvement of given architectures, and trade space exploration is not well informed. To overcome these limitations and to deal with the complexity of large systems, other network-based methods have been proposed, with the goal of facilitating the understanding of systems architectural features. Some of these are specific to a particular class of problems. For example, the generalized information network analysis (GINA) method models the flow of information in aerospace systems for communication and sensing (Shaw et al. 2001), with the goal of assessing cost, capability, and performance. A common framework used in systems engineering to deal with dependencies is the Design Structure Matrix (DSM) methodology. Analogous to adjacency matrix in graph theory, DSM is used to model and analyze system structural features and dependencies (Browning 2001; Eppinger and Browning 2012). DSM uses both metrics from graph theory, for example, betweenness centrality, and ad hoc algorithms for clustering systems and support organization in system development. DSM has been extended to multiple domain, with the Multiple Domain Matrix (MDM) methodology (Maurer and Lindemann 2008), and successfully applied by Bartolomei et al. (2012) to Engineering Systems. Engineering Systems Multiple Domain Matrix (ES-MDM) constitutes a conceptual framework to model interactions between systems, functions, objectives, and activities, for application of DSM analysis methods. SODA approach models the behavior of complex systems with a low number of parameters, having an intuitive meaning related to the physical systems. Therefore, it can give useful insights in the early stage of design, and identification of critical systems, without requiring extensive simulations. As a parametric model, SODA can be used in conjunction with existing frameworks for system analysis. For example, SODA adds more insights into the impact of the dependencies between systems in the operational and functional domain and can therefore be used in conjunction with DSM and ES-MDM framework. DSM and ES-MDM support SODA users in deciding the adequate set of nodes required to model the system behavior, and constitute a valid tool to represent matrices of SODA parameters and to identify clusters in SODA networks. SODA can identify criticalities inside and among clusters and suggest ways to shape the dependencies, and improve operational architectures.

2.2 Trade space exploration and architecture generation

Trade space exploration has been addressed by multiple authors. McManus et al. (2004) introduced a framework that allow for rapid architecture selection in the conceptual design phase. However, the qualitative considerations on risk and utility are based on classical systems engineering methodology and do not account for some complex features, for example, operational dependencies. Other authors include dependencies between variables in conceptual design (Nunez et al. 2012), or in change propagation (Hamraz et al. 2012), but only as a qualitative flow, or very simple model of binary or qualitative interdependencies (Augustine et al. 2012), or with probabilistic assessments requiring many simulations (Clarkson et al. 2004). SODA is a valid alternative to existing parametric models or to simulation-based models. While keeping the parametric model quite simple, SODA accounts for quantitative partial dependencies and can be used to improve existing frameworks for architecture selection and change propagation analysis. An advantage of this quantitative model is that it gives insight into the causes of the observed behavior, thus allowing for informed architecture generation and therefore improved trade space exploration.

2.3 Criticality and risk analysis

Similar approaches, involving extensive simulations, or qualitative assessment and binary dependencies, are used to identify critical nodes in interdependent networks (Rinaldi et al. 2001; Sosa 2008; Zio and Sansavini 2011; Nguyen et al. 2013), and to support decision in risk management (Gaonkar and Viswanadham 2007; Fang and Marle 2012). Besides giving a simple model of the behavior of complex networks in case of failure, SODA addresses other limitations of this current approach to failure and risk analysis. First of all, in current practice, failure modes are often considered only at the end of the design process. Kurtoglu et al. (2010) underlined the need to consider functional failures in the early design process, though the methodology they propose relies on simulations, which require previous knowledge of possible architectures. We propose an alternative approach, to similarly consider failures in early design, and to evaluate the impact of system architecture on operability, while reducing the need for simulations. SODA allows the designer to include the impact of dependencies and identify root causes of the observed global behavior, when failures occur, at the beginning of

the design process, replacing simulations with a parametric model of the behavior. Therefore, with SODA the designer can include considerations about risk when selecting the most promising architectures. Methods like Fault Tree Analysis (FTA) (Watson 1961; Mearns 1965) and Failure Mode and Effect Analysis (FMEA) (United States Department of Defense 1949) have limitations when dealing with multiple complex failures, and use ordinal scale (FMEA) or binary faults (FTA), while SODA deals with the analysis of the effect of system failures, including partial failures, and is able to model details of the interactions, including multiple failures and cascading effects.

3 Systems operational dependency analysis

In this section, we provide a general description of the systems operational dependency analysis (SODA) model, SODA parameters, model formulation, and basics of SODA analysis, introducing the novel concepts and ideas related to SODA. The next section offers a thorough description of how to use and apply the concepts described in this section. SODA is a method to analyze the result of possible cascading effect of dependencies between systems on the overall operability, in case of disruptions. The method is based on previous work by Garvey and Pinto (2009, 2012), Garvey et al. 2014), who proposed functional dependency network Analysis (FDNA), a 2-parameters model of dependencies between capabilities. This method is derived from the Leontief-based Input/Output model for infrastructures (Haimes and Jiang 2001; Haimes et al. 2005). Haimes proposed the use of a simple linear model of dependencies to analyze production and services in infrastructures. FDNA is a 2-parameter piecewise linear model of dependencies between capabilities. The parameters have intuitive meaning, and the method is used to analyze the impact of failures on the desired capabilities in complex systems. While FDNA results in a good input/output model for capabilities, adding value to a simple linear model, direct application of FDNA to model the behavior of complex systems and the impact of dependencies showed some limitations, which restricted its use to support decision making in systems architecture. We appreciate the power of a simple parametric model, and we kept the idea of separating the impact of a one-to-one dependency into a critical zone, where very low input is available, and a non-critical zone, where high input is available. However, since FDNA has been developed to deal with capabilities, it does not consider the influence of the internal status of a system. Also, it does not include stochasticity, and it fails to model some example of behavior that we observed in various applications. For example, FDNA cannot model dependencies where the criticality is rapidly absorbed as a result of a small increase in the input. Additionally, when multiple dependencies occur, FDNA can model only substitute inputs (OR-like). To address these limitations, building on FDNA and its previous application, and based on results from agent-based model (ABM) simulations of problems from different fields of application (e.g., warfare scenarios, GPS-based autonomous flocking of vehicles, space exploration architectures), we propose the SODA model. SODA is a 3-parameter piecewise linear model, suitable to analyze system dependencies, including partial dependencies, and their effect on system behavior. We recognize that the features of complex systems and the computational cost needed to perform analysis of these systems require analytical methods to keep some inherently qualitative aspect. However, one of the goals of SODA is to keep these qualitative features to a minimum. For this reason, SODA parametric model trades a more detailed behavioral analysis off for a quantitative representation of the systems behavior. In Sect. 1, we underlined how the parameters of SODA model may be evaluated via parametric regression analysis from experimental or historical data, but also be estimated—if such data are not available—by expert assessment. Tools are under development to support the latter; however, we suggest expert evaluation to be kept at a minimum, in order to reduce the possible introduction of subjectivity into the model. We also recommend performing sensitivity analysis to evaluate the possible impact of errors in the evaluation of the parameters of SODA model.

We validated the model through agent-based model (ABM) simulations, executed through the Discrete Agent Framework (Chow et al. 2012; Mour et al. 2013), and successfully applied the methodology to aerospace systems. This publication is mainly focused on the methodology, which is described in detail. The synthetic case study described in Sect. 5 is a small example of application of SODA analysis, and it is part of a larger study for a Naval Warfare Scenario. Future publications, focused on specific applications, will cover larger applications of SODA to a variety of problems, belonging to different fields.

3.1 Operational dependencies

In SODA, we model the architecture of complex systems and SoS as a directed operational network (Fig. 1). The nodes represent either the component systems or the capability to be acquired. Accordingly, the links represent the operational dependencies between the systems or between the capabilities.

In SODA, each node is characterized by its internal health status, or self-effectiveness (SE), ranging between 0

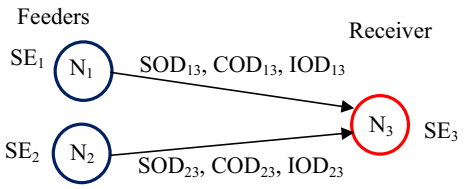

Fig. 1 Synthetic SODA network. *N* node, *SOD* strength of dependency, *COD* criticality of dependency, *IOD* impact of dependency, *SE* self-effectiveness

(system not working at all) and 100 (system having maximum performance). Each link is characterized by three parameters: strength of dependency (SOD), criticality of dependency (COD), and impact of dependency (IOD) that affect the behavior of the whole system-of-systems in different ways. These parameters, described in detail in the next section, can come from expert judgment and evaluation, and they yield a direct insight into the cause of the observed behavior. On the other hand, the parameters of the model can be computed based on historical data, or through a limited number of simulations and experiments.

SODA is used to evaluate the effect of topology and of possible degraded functioning of one or more systems on the operability of each system in the network. The analysis can be a deterministic evaluation of a single instance of the system status or a stochastic analysis of the overall behavior. In the deterministic analysis, given the SE of each system, and the properties of each dependency, SODA quantifies the operability O_i of each node, according to the model described in Sect. 3.3. The operability of a node, ranging between 0 and 100, is defined as the level at which the system is currently operating, or the level at which the desired capability is being currently achieved. Operability of a hypothetical system is related to performance by means of a given function, as shown in Fig. 2. It is thus related to the value or utility that the system is achieving (however, a value function may include other desired variables than operability). When designing the operational network, the user must also define the relationship between performance and operability. An example of the entire

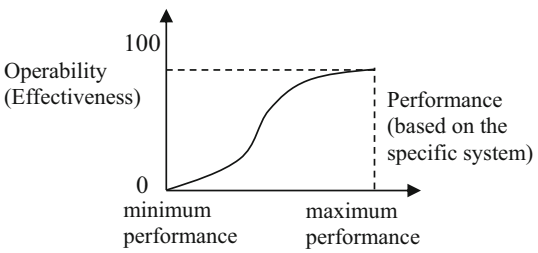

Fig. 2 Performance and operability. The minimum or worst possible performance corresponds to operability equal to 0. The maximum or best possible performance corresponds to operability equal to 100

procedure is described in Sects. 4 and 5. The operability of nodes of interest is used to analyze and evaluate properties of the overall system, such as robustness, resilience, risk.

In the stochastic version of SODA, the SE of each system follows a probability distribution. Consequently, the operability of each node is also probabilistic. Different from the deterministic version, this type of analysis deals with the overall behavior of the system, rather than with a single instance.

SODA can thus be used to identify the most critical nodes and dependencies in the network, in terms of impact on the operability, under different disruptive conditions. Designers can compare different architectures and use metrics based on operability, to quantify robustness and resilience of complex systems, and the risk of each architectural design in terms of impact of disruptions. These metrics may be defined based on the specific application and on the desired analysis. In this paper, we use the difference between 100 and the resulting value of operability following disruption (expected value in case of stochastic analysis) as a measure of the robustness of the architecture to the disruption. If we add flexibility to the architecture, by applying rules that allow the architecture to be reshaped in case of disruption (e.g., a system can support a disrupted system), we use the difference between the value of operability when actions are taken to counteract the effect of disruption and the original value of operability following the disruption as a measure of the resilience of the architecture (i.e., the capability to recover part of the loss in operability).

3.2 Parameters of the model

Each dependency between two systems is represented with three parameters. Experiments and simulations on real applications showed that this model is more accurate than other models of the same family (Haimes and Jiang 2001; Haimes et al. 2005; Garvey and Pinto 2012), yet still simple enough to guarantee fast analysis of a high number of architectures. The low number of parameters, and their intuitive meaning, make them both suitable to be assessed by knowledgeable designers and to be used to drive decision making in complex systems architectural design.

3.2.1 Strength of dependency

Strength of dependency accounts for how much the behavior of a system depends on the behavior of a feeder system. SOD is the predominant factor when the feeder has a high level of operability (SOD zone in Fig. 3). The parameter for SOD, α_{ij}, ranges between 0 and 1 and is defined as the fraction of operability of node j that is depending on the operability of node i. The rest of the

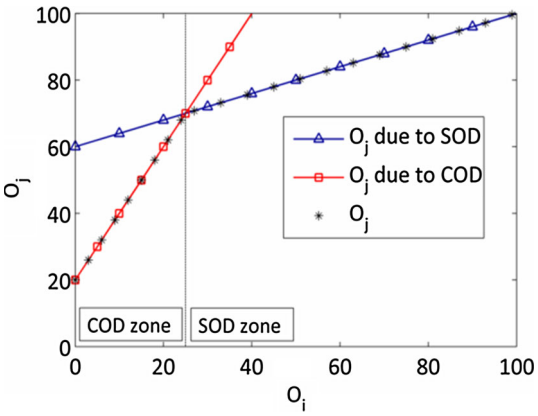

Fig. 3 Operability due to single dependency of system j from system i. $SE_j = 100$. *Line with triangles* term due to SOD (here, 0.4). *Line with squares* term due to COD (here, 80). *Stars*: O_j as a function of O_i. Criticality is prevalent for $O_i < 25$

operability of node j is dependent on its SE. α_{ij} is the slope of the line with triangles in Fig. 3.

3.2.2 Criticality of dependency

Criticality of dependency is one of the two parameters that quantify how the functionality of a system degrades when a feeder system is experiencing a major failure. COD is the predominant factor when the feeder has a low level of operability (COD zone in Fig. 3). The parameter for COD, β_{ij}, ranges between 0 and 100 and is defined as the maximum loss in operability of node j, that is, the drop in the operability level of node j, when node i has operability equal to 0. In Fig. 3, β_{ij} is the difference between 100 and the intercept of the line with squares with the y-axis. We underline that this definition is slightly different from the one given by Garvey and Pinto (2009, 2012) for FDNA. With the new definition, high β_{ij} corresponds to high criticality.

3.2.3 Impact of dependency

In the definition of FDNA (Garvey and Pinto 2009, 2012; Guariniello and DeLaurentis 2013a), the slope of the function relating the operability of nodes i and j in the COD zone was always equal to 1. This meant that, starting from the minimum operability of node j (for low levels of operability of node i), in the COD zone this operability O_j would depend solely on the operability of node i, without any contributions by the SE of node j. Results from ABM simulations showed that, starting from the baseline corresponding to operability of node i equal to 0, the operability of node j can increase faster than the increase in operability of node i. The critical dependency can thus have a lower impact and be restricted to a smaller zone. A small width of the critical zone models a dependency that may be highly critical, i.e., resulting in a large loss of operability if the input is completely disrupted, but that requires just a small level of operability of the feeder node to achieve high level of operability of the receiver node. Simple linear models and FDNA piecewise linear model fail to capture this feature—which we found to be very common in many systems dependencies—resulting in larger modeling error (Fig. 4). Due to this observation, we introduce a parameter for the Impact of Dependency, γ_{ij}. It ranges between 1 and 100 and is defined as 100 divided by the slope of the COD-dependent function (line with squares in Fig. 3). The resulting support function can model a wider spectrum of dependencies than FDNA (e.g., it can model dependencies that exhibit an input/output behavior similar to a step function). Other models, for example, nonlinear functions, or larger polynomials, may result in a better behavioral model of complex one-to-one dependencies than SODA, but will also increase both the computational cost of the analysis, and the complexity of the model setup.

3.3 The model: dependency from a single system

Based on the parameters described in the previous sections, we model the operability O_j of node j, depending only on node i, according to the following equations.

The operability of root nodes, i.e., nodes that are not dependent by any other node, is equal to their SE:

$$O_i = \mathrm{SE}_i \tag{1}$$

The operability of a node j, depending only on one feeder node i, is computed as the minimum of two terms (black stars in Fig. 3), one depending on the SOD, one depending on the COD:

$$O_j = \min\left(O_j^S, O_j^C\right) \tag{2}$$

The term depending on the SOD is computed based on the operability of the feeder, and the SE of node j:

$$O_j^S = \alpha_{ij} O_i + \left(1 - \alpha_{ij}\right)\mathrm{SE}_j \tag{3}$$

The term in the critical zone is computed based on COD, IOD, and the operability of the feeder:

$$O_j^C = \frac{100}{\gamma_{ij}} O_i + \left(100 - \beta_{ij}\right) \tag{4}$$

3.4 The model: dependency from multiple systems

When a node is dependent on more than one node, the equations of SODA are slightly modified. Following the formulation by Garvey and Pinto (2009, 2012) for FDNA, we compute the operability term depending on SOD as the

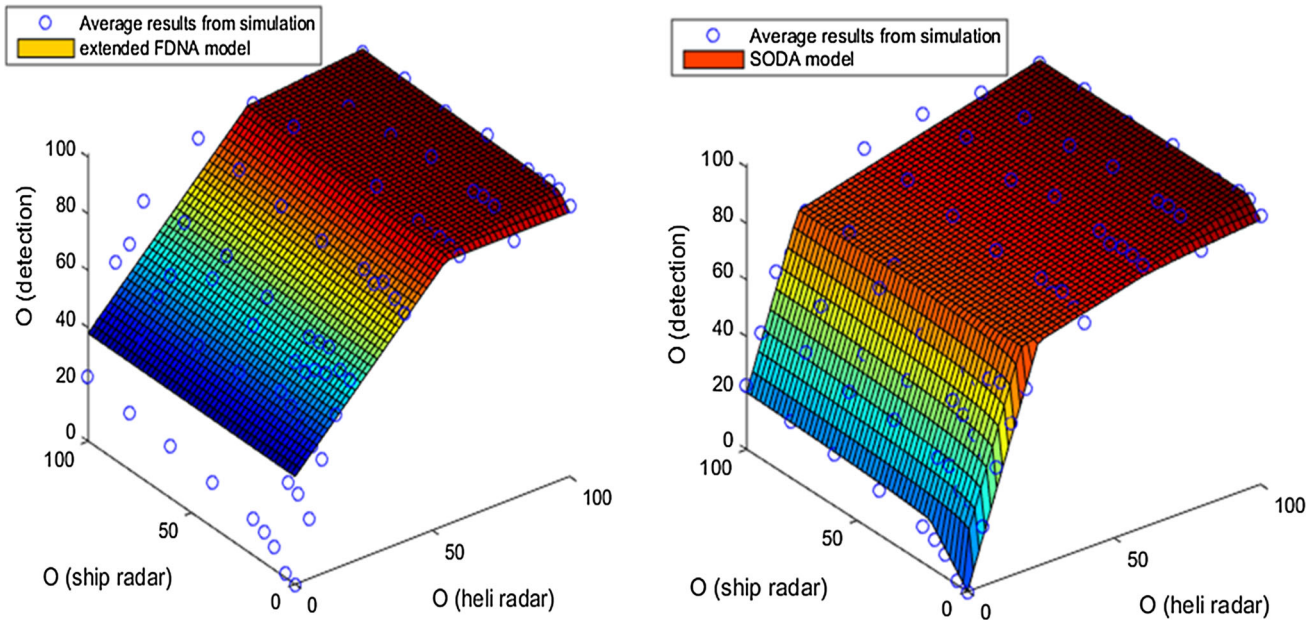

Fig. 4 Example of dependency of detection from ship and helicopter radar in a Naval Warfare Scenario. *Left* FDNA model modified to consider internal status of systems. The 2-parameter model does not capture the high increase in detection capability caused by slight increase in helicopter radar capability. Also, without corrective weight ($\lambda = 0$), the zone where the operability of both feeders is low is dominated by the most critical node (helicopter), and FDNA fails to model this behavior. *Right* SODA model better captures the input/output relationship, and with the corrective weight ($\lambda = 0.1$), it better models the combined effect of multiple dependencies

average of the corresponding terms for each dependency. We compute the operability term depending on COD as the minimum of the corresponding terms for each dependency, thus reflecting the intuitive idea of the overall impact of a critical dependency.

However, the use of this formulation results in a possible non-zero operability of a node even when all the feeders have 0 operability (Fig. 4, left). ABM simulations showed that there are cases of dependency from multiple nodes when operability of a node may decrease to 0 when all the feeders have 0 operability. To keep a simple model of one-to-one dependencies, we modeled this effect by applying a multiplicative weight W to the parameter that models the COD. For each feeder of the node under consideration, the weight is the average of the operability resulting from other feeders and has an exponent λ ranging between 0 and 1 (in the case presented in this paper, we assumed a value of 0.1 for the exponent). With the multiplicative weight, the parameter modeling the COD represents the loss in operability resulting from the total loss of one feeder when all other feeders are working properly. The exponent models how large the impact of other feeders in the critical zone is. An exponent of 0 models an "OR-like" dependency, where each dependency of a node j from a node i results in a certain level of operability, without accounting for dependencies of node j from other nodes. An exponent of 1 models an "AND-like" dependency, where the operability resulting from the dependency of j on i degrades to 0 if j is not receiving adequate input from its other feeder nodes. Intermediate non-zero exponents model different levels of impact of other feeders in the critical zone. Multiple dependencies are then modeled as follows.

The operability of root nodes, i.e., nodes that are not dependent by any other node, is equal to their SE:

$$O_i = \mathrm{SE}_i \tag{5}$$

The operability of node j, depending on multiple feeders, is computed as the minimum of two terms, one depending on the SODs and another depending on the CODs:

$$O_j = \min\left(O_j^S, O_j^C\right) \tag{6}$$

The term depending on the SODs is the average of SOD-based terms, computed based on the operability of each of the n feeders, and the SE of node j:

$$O_j^S = \frac{1}{n}\sum_{i=1}^{n} O_{ij}^S \tag{7}$$

$$O_{ij}^S = \alpha_{ij} O_i + \left(1 - \alpha_{ij}\right)\mathrm{SE}_j \tag{8}$$

The term in the critical zone is the minimum of terms based on COD, IOD, and the operability of each of the n feeders:

$$O_j^C = \min\left(O_{1j}^C, O_{2j}^C, \ldots, O_{nj}^C\right) \tag{9}$$

$$O_{ij}^{C} = \frac{100}{\gamma_{ij}} O_i + W_i^{\lambda}(100 - \beta_{ij}) \quad (10)$$

If the network has cycles, a variant of the same equations can be used. They constitute a set of nonlinear equations, whose solution can be found iteratively, using the SE of each node as initial condition for the operability. Stopping criteria include number of iterations, or change in operability with respect to the previous iteration. However, in this paper, the scope is limited to acyclic networks.

3.5 Deterministic analysis

The simplest analysis that can be performed with SODA method is a single-point, deterministic analysis. Values for the SE of each system, i.e., their possible degraded status, are fed into the equations, to compute the actual operability of each system.

This kind of analysis can be thought of as a way to answer *what if* questions: for example, if the user is interested in the impact of a specific system on the overall behavior of the whole entity, values ranging between 0 and 100 can be assigned to the SE of the system under consideration, and the subsequent operability of each of the other systems can be computed through SODA, and listed in tables or plotted in graphic format. Another example may involve partial failures in several systems, so as to evaluate the combined effect of multiple failures.

Deterministic analysis gives good insight into the influence of dependencies into the operational network. The user can identify the most critical nodes under specified conditions, i.e., the nodes that most affect the operability of other nodes. Some of the results may show unexpected behavior, for instance possible reduction of the impact of failures, due to the particular topology. The user can compare different architectures, based on their response to failures and accidents. It must be noted that the results of this analysis also depend on the output of interest: for example, a node might be critical on the operability of a low-interest node, while having a small impact over nodes of interest. Stochastic analysis better catches details about the overall impact of disruptions on the operability.

3.6 Stochastic analysis

A more realistic understanding of the systems behavior as a function of the dependencies between components can be achieved by means of a stochastic analysis with SODA. In stochastic analysis, a probability density function of the SE, rather than a single instance, is used. The corresponding output is a probability density function for the operability of each of the component systems, accounting for all the SOD, COD, and IOD values as well as the overall effect of topology. This behavior is condensed in a few probability distributions of interest instead of long tables, as with the deterministic analysis. In particular, the expected value of the operability of a system gives a measure of the robustness of such a system to failures of the feeders, while the variance of the operability evaluates the sensitivity of the system to failures of the feeders. Such outputs show behavioral patterns and features of a whole architecture and thus are valuable in design activities. For example, given the expected distribution of SE of the component systems over time (including aging, minor failures, major accidents), and a threshold for the minimum operability to be achieved by some system of interest, the user can compute the probability that these systems are operating above the given threshold, as time goes by. The user can then compare alternate architectures, identify their critical systems, and explain the role of topology in the observed criticality.

Based on SODA equations, an analytic expression for the expected value and variance of the operability can be derived (operability is a piecewise linear combination of the SE). However, due to the possible high number of nodes in the network, and since SODA is computationally inexpensive (Table 1), Monte Carlo simulation appears to be the best choice to perform this type of analysis. Stochastic analysis, similarly to deterministic analysis, can be used to analyze the combined effect of multiple failures. Since SODA formulation for multiple dependencies (Sect. 3.4) does not explicitly treat the correlation between systems feeding the same node, the PDFs of the self-effectiveness of these systems are statistically independent in the model. We suggest two ways to address correlation. The user can choose input for the Monte Carlo simulation from statistically dependent PDFs. A more accurate alternative is to model the correlation between the systems: since the internal status of the systems is correlated, it means that part of their operability depends on a common cause, which can be explicitly modeled in SODA with a node feeding the correlated systems.

3.7 Synthesis and architectural design updates

Whereas SODA analysis allows for comparison between various architectural designs, and for trade-off between competing features, the intuitiveness of the parameters used in SODA model can support decision making in design updates.

Since the parameters give insights into the causes of the observed behavior, they also suggest possible ways to effectively improve this behavior on an already existing architecture and allow for evaluation of the impact of architectural changes, and modified dependencies.

Table 1 Computational cost of SODA analysis

Number of nodes	Avg. number of edges (100 runs)	Avg. time (s)	Max time (s)
100	2484	0.0125	0.0143
500	62,361	0.887	0.922
1000	249,687	6.625	6.804
2000	999,785	61.41	67.82

Time to perform analysis of a single instance of the operational network, with variable number of nodes and edges. Processor Intel Core i3-2350 M 2.3 Ghz, 4 Gb DDR3 RAM

4 SODA problem setup

In this section, we list and describe the steps required to apply SODA analysis, and provide guidelines to the use of SODA. The baseline ideas, nomenclature, and mathematical formulation of SODA modeling have been introduced in Sect. 3. These concepts are applied in this section to illustrate a step-by-step description of SODA modeling, analysis, and synthesis.

4.1 Step 1: operational dependencies

The first step necessary to apply SODA is the conversion of the requirements and the systems involved into a network of operational dependencies. It must be underlined that the topology of this network will depend not only on the desired or required topology of interactions between the systems (exchange of information, matter, energy), but also on the desired output from the analysis. The network of operational dependencies will include nodes representing the systems that are assigned to perform a task, and nodes representing capabilities that the user is interested in quantifying. SODA is a model of the operational domain; therefore, the flow between systems is generalized in terms of operability. System Engineering methods for functional allocation can be used to perform this step (INCOSE 2015). Functional Modeling method (Stone and Wood 2000; Hutcheson et al. 2007) has the advantage of identifying possible failure modes (Tumer and Stone 2003; Stone et al. 2005) that can also be used as a basis for analysis of the impact of disruptions with SODA.

Alternative architectures may have different systems to perform the required function, different network topology, or different features of the dependencies.

4.2 Step 2: self-effectiveness and operability

The second step in the problem setup is the definition of the *internal status*, that will be represented by the SE of each system, and of the *performance*, that will be represented by the operability of each node. Both SE and operability are normalized between 0 (worst case) and 100 (best case). Multi-dimensional performance is represented by different capability nodes (and associated systems). The measure of internal status and performance, to be associated respectively with SE and operability, can be evaluated by experts, or computed from historical data, experiments, or simulations.

4.3 Step 3: dependency parameters (α_{ij}, β_{ij}, γ_{ij})

At this point, to be able to model the overall behavior through SODA, we need to determine the parameters of each dependency. In their FDNA formulation (Garvey and Pinto 2009, 2012), Garvey and Pinto describe a way to evaluate α_{ij} as the fraction of operability of a node depending on its feeder, but accounting for a *virtual baseline operability* of the system, when the feeders are not giving any input, and there is no criticality. Then they suggest to evaluate β_{ij} as the actual operability that a system achieves, when the feeders are not giving any input.

The parameters for the dependencies can be effectively evaluated by experts, based on these definitions and on the equations of SODA, if the users have a good understanding of the impact of each of these parameters on the operability dependency between a feeder node and a receiver node. Figure 5 shows the effect of α_{ij} on a single dependency of node N_j from node N_i, when β_{ij} is equal to 95, γ_{ij} is equal to 10, and α_{ij} increases from the left plot to the right plot. Each plot in the figure represents the operability O_j as a function of the operability O_i. As the SOD grows larger and larger, the operability of node N_j is increasingly dependent on that of node N_i, and less dependent on its SE.

Figure 6 shows the effect of β_{ij} on a single dependency of node N_j from node N_i, when α_{ij} is equal to 0.5, γ_{ij} is equal to 40, and β_{ij} increases from the left plot to the right plot. Each plot in the figure represents the operability O_j as a function of the operability O_i. As the COD grows larger and larger, the critical zone (low values of O_i) increases in size, and the loss due to criticality gets bigger.

Figure 7 shows the effect of γ_{ij} on a single dependency of node N_j from node N_i, when α_{ij} is equal to 0.5, β_{ij} is equal to 95, and γ_{ij} increases from the left plot to the right plot. Each plot in the figure represents the operability O_j as a function of the operability O_i. As the IOD grows larger and larger, the critical zone (low values of O_i) gets bigger,

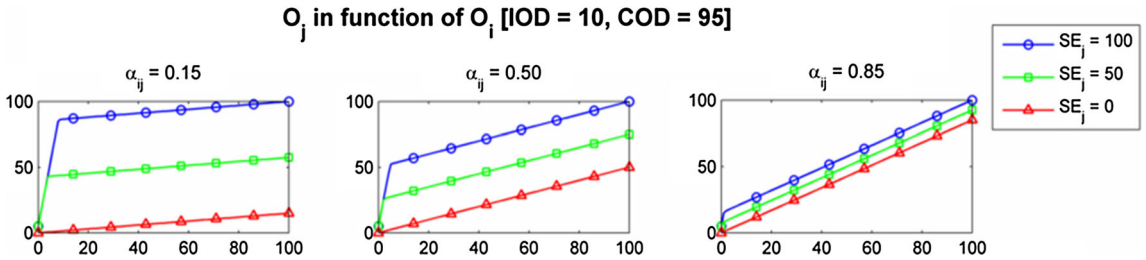

Fig. 5 Operability of node N_j in function of the operability of node N_i for different values of α_{ij}. *Left* $\alpha_{ij} = 0.15$ (O_j is most dependent on SE_j). *Center* $\alpha_{ij} = 0.5$ (O_j is equally dependent on SE_j and O_i). *Right* $\alpha_{ij} = 0.85$ (O_j is most dependent on O_i)

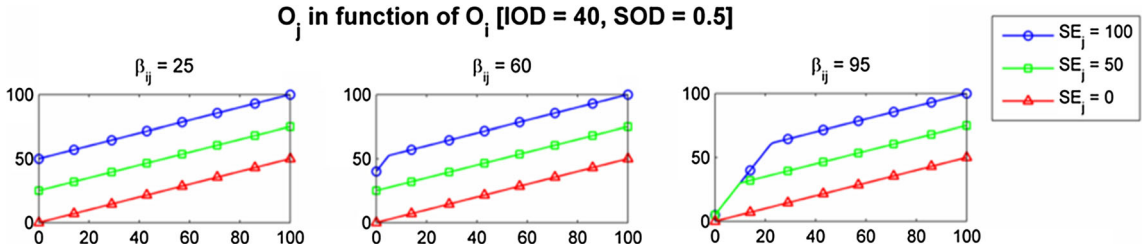

Fig. 6 Operability of node N_j in function of the operability of node N_i for different values of β_{ij}. *Left* $\beta_{ij} = 25$ (criticality does not occur because of low SOD). *Center* $\beta_{ij} = 60$ (criticality occurs, but the loss is not visible when SE_j is small). *Right* $\beta_{ij} = 95$ (criticality occurs, and the loss is big, yet it is not visible when $SE_j = 0$)

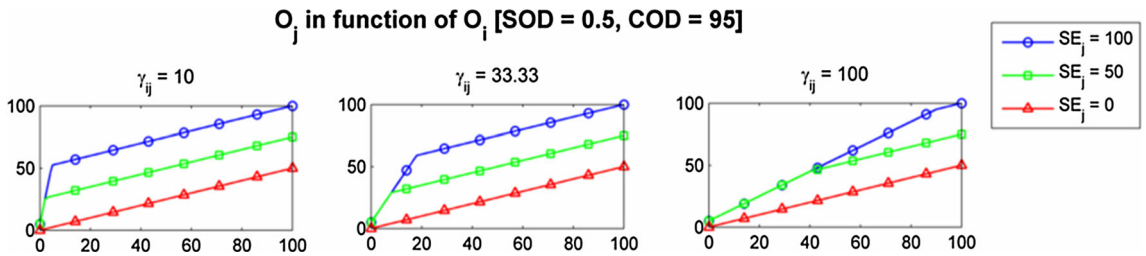

Fig. 7 Operability of node N_j in function of the operability of node N_i for different values of γ_{ij}. *Left* $\gamma_{ij} = 10$. *Center* $\gamma_{ij} = 33.3$ (criticality occurs over a wider range of O_i). *Right* $\gamma_{ij} = 100$ (criticality occurs over a wider range of O_i, and it is the predominant factor for high values of SE_j)

having impact even at high values of O_i, especially when SE_j is high.

The combination of the three parameters, and of multiple dependencies, results in a model of the systems behavior, according to SODA equations. The process can be reversed, and historical data, or results from simulation may be used to evaluate, by means of regression analysis, the parameters of the SODA structure that best fits the results. These parameters may then be used to perform more comprehensive analysis, without the need to execute full simulation. Therefore, if the user has historical data or a good knowledge of one-to-one dependency behavior, the parameters that model the global behavior can be computed with a cost of the same order of magnitude as the cost of other parametric models (e.g., Response Surface Methodology), that is generally lower than the cost of performing simulations. If data are not available, Design of Experiments techniques can be applied, so that a low number of simulation is used to identify the parameters of the model. In this case, the cost to compute the parameters is at most the same as simulation-based models (e.g., Bayesian Networks), while the cost of the analysis is very low (Table 1). Since SODA is a parametric model, it trades detail off for low cost, and for the advantage of having a representation capable of giving insight into the reason of observed results. The level of detail of SODA analysis will depend on the level of detail of available data or simulations. For the early design phase, we suggest to use simple models, which will be able to identify more and less promising architectures. For later design phases, or for update of existing architectures, more data will be available to support accurate parametric regression.

4.4 Step 4: analysis

Once the SODA network, with the relative parameters, is available, the user can perform the required analysis. In step 1, systems and capabilities of interest were included in the network. The user may now decide to perform deterministic analysis, setting up points of interest (each one being a set of SE for each node, corresponding to a specific failure), and use the results to evaluate the impact of different failures, including multiple failures, and to identify the critical systems and dependencies. Otherwise, the user may use models of failure probability in individual systems to perform stochastic analysis and analyze the behavior of the whole entity, and the main features of the impact of failures and functional dependencies. Metrics of interest, representing, for example, the robustness, and resilience of the entire system can be built based on the results of deterministic or stochastic analysis (systems operability), and their evolution over time.

4.5 Step 5: synthesis

SODA allows for fast analysis of the effect of various failures scenarios on systems behavior and capabilities, as well as comparison of architectures, and possible trade-offs between competing metrics (reliability, resilience, capabilities, cost). In addition, the parameters of the SODA model for operational dependencies give insight not only on the effect of the operational dependencies between the systems, but also on some of the reasons why an observed behavior occurs. Based on the relationship between some of the parameters, the topology of the network, and the observed results, the designer can decide the appropriate actions needed to improve the architecture (increase redundancy, add, or delete paths between node pairs, invest money to increase the robustness of critical nodes, etc.). New architectures can be generated based on these observations and analyzed again by means of SODA. This process can be iterated to improve architectures.

5 Case study: a small naval warfare scenario

In this section, we show results of the application of SODA for risk analysis of a small Naval Warfare scenario SoS. The scenario is comprised of an MH-60 helicopter, a ship equipped with detection and weapon systems, and an adversary boat approaching the coast. The helicopter and the ship can perform detection of the boat, and the ship can also engage the adversary. We are interested in time required to detect the adversary, and time required to engage it. We follow the steps described in the previous section.

5.1 Step 1: operational dependencies

The first step in SODA is the conversion of the requirements and the systems involved into a network of operational dependency. As noted in Sect. 4.1, the topology of this network depends not only on the desired or required topology of interactions between the systems (exchange of information, matter, energy), but also on the desired output from the analysis: in this case, two nodes will represent the capability of detection, and the capability of engagement, respectively. We use the intermediate node representing the capability of detection because we are interested in modeling the effect of the dependency of detection time from the operability of the detection systems. If the user is interested in different outputs, the nodes of the functional network can be changed accordingly. Three more nodes will represent the helicopter's detection system, the ship's detection system, and the ship's weapon systems. The resulting architecture is shown in Fig. 8. Alternative architectures may involve different systems to perform the required function, different network topology, or different features of the dependencies. As noted in Sect. 3.6, we may decide to model the correlation between operability of some systems explicitly. For example, the operability of the ship's radar and weapon systems is likely to exhibit some degree of correlation, since the systems are on the same ship (if the ship is damaged, both systems will be disrupted). For more accurate results, we could add one node, representing, for example, the ship hull, feeding both the ship's radar and the ship's weapon system. This node would have a simultaneous impact on the operability of the systems aboard the ship.

5.2 Step 2: self-effectiveness and operability

The second step in the problem setup is the definition of the *internal status* that will be represented by the SE of each systems, and of the *performance*, that will be represented by the operability of each node. In this case, we use simplifying assumptions for detection and weapon systems: they operate within a fixed range, and their SE is a measure of the probability to correctly detect or engage the adversary if it is within the operative range. The operability of

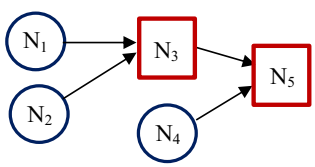

Fig. 8 The small naval warfare operational dependencies. N_1 ship's radar, N_2 helicopter's radar, N_3 detection of the adversary, N_4 ship's weapon system, N_5 engagement of the adversary

such nodes corresponds to their SE, since they are root nodes (they do not have feeder nodes).

The operability of the detection capability node is a measure of the time required to perform detection of the adversary boat, normalized between 0 (longest time necessary to detect the adversary. In this case, it is the maximum allowed mission time, before the adversary reaches the coast) and 100 (minimum time required to detect the adversary).

The operability of the engagement capability node is a measure of the time required to engage the boat, normalized between 0 and 100, similarly to what we did for the operability of the detection capability.

5.3 Step 3: dependency parameters (α_{ij}, β_{ij}, γ_{ij})

At this point, to be able to model the Naval Warfare SoS behavior through SODA, we need to evaluate the parameters of each dependency. These can be effectively evaluated by experts, assuming that a good description of the meaning of each parameter is familiar to the user. Based on the concepts described in the previous sections and represented in Figs. 3, 5, 6 and 7, experts can select values of the parameters resulting in appropriate curves for each dependency. However, relying only on expert opinion can be prone to error. In this case (especially when adequate information about the systems is not available), we suggest to perform sensitivity analysis to identify possible large errors resulting from a wrong choice of parameters. If historical data or some simulation results are available, the user can perform a regression to identify the appropriate values for the parametric model. All these approaches require less computational time than required to run extensive simulations of the behavior of the complex system. In the worst case, to obtain a high level of detail, the user will perform the same amount of simulation required in Response Surface Methodology or Bayesian Networks.

In this example, since data from a real problem were not yet available, we ran simulations with an ABM and used the results to perform regression and retrieve the required parameters, and to validate the SODA model. If this approach is chosen, design of experiment can help to reduce the amount of simulation required to obtain enough data to build the SODA model.

5.3.1 ABM and validation

We developed an agent-based model for the small Naval Warfare SoS, including an adversary boat moving toward the coast (and performing escape maneuvers in case of detection), a helicopter capable of detecting the adversary, hovering over it, and communicating its position to the ship, and a ship capable of detecting and engaging the adversary.

We used the Discrete Agent Framework ABM to perform a full-factorial design of experiments, with eleven discrete levels of SE for each system. Due to the uncertainties modeled in the ABM, we ran 10,000 instances per each design point. The results, comprising the time required to detect and engage the enemy in various working conditions of radars and weaponry, were used to determine the parameters of the SODA model, with a parametric regression (least square error between the results from the simulation, and results of a model having the SODA structure). A three-dimensional representation of part of the multi-dimensional fitting is shown in Fig. 4. The following matrices show the parameters identified for the Naval Warfare Scenario SoS:

$$\mathrm{SOD} = \begin{bmatrix} 0 & 0 & 0.28 & 0 & 0 \\ 0 & 0 & 0.88 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0.89 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{COD} = \begin{bmatrix} 0 & 0 & 9.03 & 0 & 0 \\ 0 & 0 & 85.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 39.7 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathrm{IOD} = \begin{bmatrix} 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 57.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 52.1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The use of an ABM to validate the SODA model confirmed the need for a third parameter (γ_{ij}) to be added to the original parameters derived from FDNA (α_{ij} and β_{ij}) to describe the operational dependencies. In FDNA, systems behavior in the region where the feeder nodes are highly disrupted is not well modeled, with errors of 15 % or more in the computed operability. This happens because of the lack of consideration for the effect of other feeders in multiple dependencies, and because FDNA imposes a slope of 1 on the critical portion of the input/output model. To model multiple dependencies in the region of high disruption of the feeders, we found out that including a weight in the operability term depending on criticality (Eq. 10) results in a better model of the systems behavior, capable of modeling OR-like and AND-like multiple dependencies.

5.4 Step 4: example of analysis

Based on the identified model, we performed analysis of criticality and reliability of the SoS. First of all, we

Table 2 Impact of systems failures on detection and engagement of the adversary

Failed system	O_3 (detection) (%)	O_5 (engagement) (%)
Minor disruption		
Ship radar	97.17	97.17
Helicopter radar	91.25	91.25
Ship weapons	100	91.06
Major disruption		
Ship radar	88.70	88.70
Helicopter radar	49.03	49.03
Ship weapons	100	64.25

performed a deterministic analysis, imposing a minor (operability degraded to 80 %) and a major (operability degraded to 20 %) disruption to one system at a time, to identify the systems that have the highest impact on the overall operability.

In this case, a measure of the overall behavior is given by the operability of nodes 3 and 5. For this problem, we used the loss in operability of the detection and engagement capability as measure of the robustness of the architecture. The loss in operability of the required capabilities is also used to rank the systems by criticality, with high loss corresponding to high criticality. Results (Table 2) indicate that this SoS is robust to failures in the ship radar, and even to limited failures in the ship weapon system, while the helicopter radar is the most critical system for detection, with consequent heavy impact on the adversary engagement.

In this kind of analysis, SODA has three advantages over traditional methodologies:

- For very large networks, analysis can be performed in very short time (Table 1).
- It constitutes a more accurate model than other piecewise linear models (Haimes Input/Output model, and FDNA), capturing effects and behavior that other model would disregard.
- Systems can be ranked based on their impact on the desired behavior, accounting for partial failures and disruption. Traditional approaches, that model only on/off status, would identify the impact of the ship weapon system as the most critical. However, this system is critical only when the failure is total, since it is the only weapon system. Under different circumstances, the helicopter radar is more critical to both detection and engagement capabilities.

An example of stochastic analysis is shown in Fig. 9. The SE of radars and weapon system has uniform probability density between 0 and 100. Monte Carlo simulation (20,000 runs) gives the probability density of the

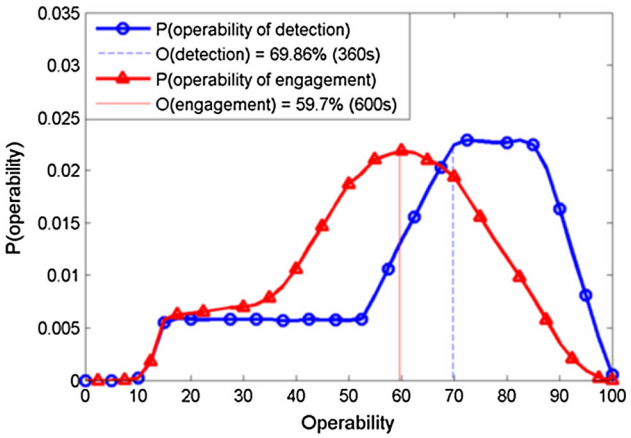

Fig. 9 Probability density of detection and engagement operability when the SE of radars and weapon systems has uniform density between 0 and 100. *Right dotted line* operability corresponding to detection time of 360 s. *Left dotted line* operability corresponding to engagement time of 600 s

Table 3 Basic metrics from stochastic analysis: expected value of operability (robustness), standard deviation of operability (sensitivity to failure), percentage of successful instances (robustness)

	Robustness % instances above threshold	Sensitivity $E(O_i)$	$\sigma(O_i)$
Detection (N_3)	52.8 % within 360 s	66	21.2
Engagement (N_5)	64.6 % within 600 s	56.2	18.4

operability of detection and engagement. Basic metrics related to robustness and sensitivity to failures are shown in Table 3, assuming a threshold of 360 s for detection, and 600 s for engagement. In this case, we use the expected value of operability and the percentage of successful instances following disruption as a measure of robustness of the architecture to the disruption. We use the standard deviation of the operability as a measure of sensitivity to the disruption.

In this kind of analysis, SODA has three advantages over traditional methodologies:

- For very large networks, SODA can be used to perform computationally inexpensive Monte Carlo simulation, based on known or realistic probability density functions of the SE of each system.
- The use of standard density functions, such as Gaussian or Beta, is not required.
- The global behavior, accounting for the impact of all dependencies, as well as the SE of each system, can be modeled and analyzed with SODA. Based on these results (and their evolution over time), measures of

robustness, resilience, and sensitivity can be computed and used to support design of the network evolution.

5.5 Step 5: example of synthesis. Improving the architecture

Deterministic analysis showed that the most critical system in the network is the helicopter radar. The parameters show that this system is independent from other systems, and the dependency of the detection from this system has very high strength, criticality, and impact. Furthermore, the detection capability has the highest criticality and impact on the capability of engagement of the adversary.

Stochastic analysis confirms these findings. Figures 10, 11 and 12, and Table 4 show the resulting probability density for detection and engagement, when the ship detection systems, the helicopter detection system, and the weapon systems have a low-valued internal status, here modeled with a *Beta*(2,11) density function.

Given the analysis performed in this simple case, possible improvement of the architectures may involve:

- Increasing the robustness of the helicopter radar.
- Supporting the helicopter radar with other systems.
- Decreasing the impact of the helicopter radar on the desired capabilities.

Table 5 shows the results of an option of redundancy of the radar (that increases the robustness), and an option of a support from an external system aboard the ship, capable of stepping in for the helicopter radar, in case of degraded capability, at the cost of a slight loss of SE in the other systems of the ship. We use the difference between the expected operability when actions are taken and the expected operability of the original disrupted system as a measure of the resilience of the architecture (capability to

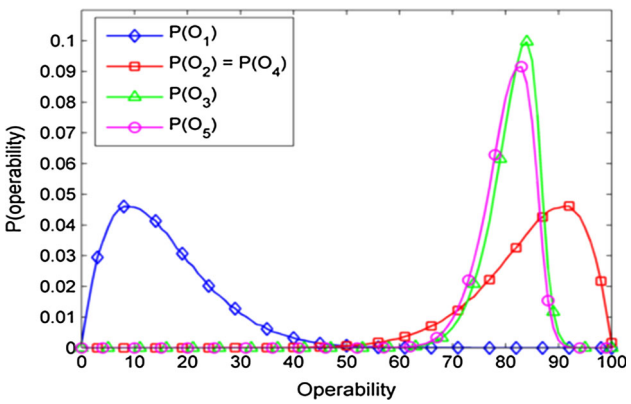

Fig. 10 Probability density of the operability of each system when the ship detection system is disrupted. The other systems keep a high level of operability. $SE_1 = Beta(2,11)$, $SE_2 = Beta(11,2)$, $SE_4 = Beta(11,2)$

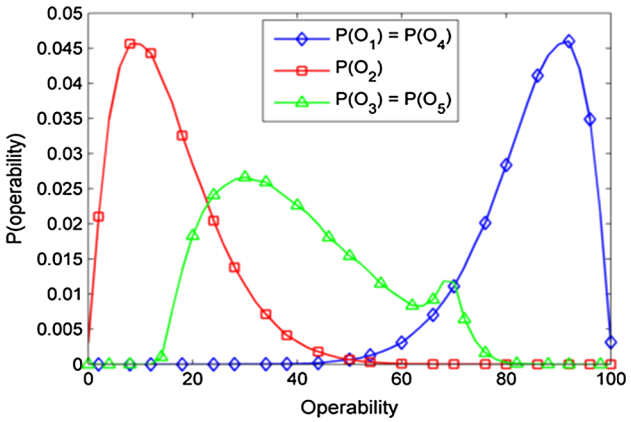

Fig. 11 Probability density of the operability of each system when the helicopter detection system is disrupted. The operability relative to the desired capabilities is highly impacted. $SE_1 = Beta(11,2)$, $SE_2 = Beta(2,11)$, $SE_4 = Beta(11,2)$

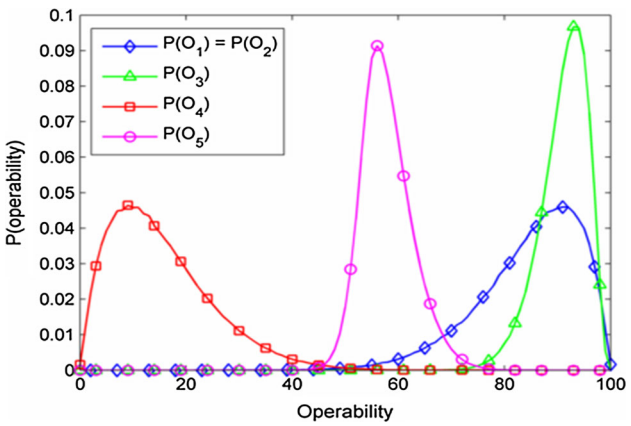

Fig. 12 Probability density of the operability of each system, when the ship weapon system is disrupted. $SE_1 = Beta(11,2)$, $SE_2 = Beta(11,2)$, $SE_4 = Beta(2,11)$

Table 4 Expected value and standard deviation of the operability of detection (O_3) and engagement (O_5), when one system is weak (SE modeled with a *Beta*(2,11) function)

Weak system	$E(O_3)$	$\sigma(O_3)$	$E(O_5)$	$\sigma(O_5)$
Ship radar	81.31	4.43	80.29	4.56
Helicopter radar	40.12	15.1	40.12	15.1
Ship weapons	91.09	4.44	57.73	4.85

Table 5 Expected value and standard deviation of the operability of detection (O_3) and engagement (O_5), when actions are taken to reduce the impact of helicopter's radar failure

	$E(O_3)$	$\sigma(O_3)$	$E(O_5)$	$\sigma(O_5)$
Original system	40.12	15.1	40.12	15.1
External support	69.87	5.20	69.69	5.17
Redundant radar	68.63	5.16	68.47	5.12

counteract the negative impact of disruption, and partially recover from the loss in operability).

The different architectures, that are modeled with different parameters than the original one, show an increased robustness to failures of the helicopter radar. The synthesis of these new architectures must be traded off with cost and, in the case of the external support, the ship must have the required flexibility.

5.6 Additional applications

Thanks to the concepts of SE and operability and to the normalization of parameters, SODA can be used in many different fields. Applications that have been tested and used to identify the weakness and limitations of previous parametric modeling applied to systems include cyber-security of complex networks, analysis of an on-orbit satellite servicing system-of-systems, and analysis for Solar System Exploration and for navigation satellites (Guariniello and DeLaurentis 2013b, c, 2014a, b; Wang et al. 2014). More complex analysis built upon this methodology involves the evolution of operability over time, based on the evolution of SE, and on possible minor and major disruption.

6 Conclusion, contributions, and future work

Building upon the Leontief-based Input/Output method originally proposed by Haimes, and the FDNA method proposed by Garvey and Pinto, we developed a method for analysis of the impact of functional dependencies between elements in a system, or systems in a SoS. The SODA methodology allows users to model the behavior of a complex system at a relatively high level of abstraction, based on a small number of input parameters whose values can be supported by limited simulation. The more traditional simulation-based approach suffers in this setting, because the amount of components and dependencies involved requires computationally expensive simulations or experiments. Furthermore, in methods like Response Surface Methodology, the modeling parameters often lack an intuitive meaning, so that it is not always possible to readily identify the root causes of the observed behavior. On the opposite extreme, some traditional models are quite simple and often fail to capture relevant behaviors of complex systems.

While SODA representation is less detailed than traditional physics-based or agent-based simulations, it provides multiple advantages and improvements:

- SODA has a very low computational cost; therefore, it is useful to quickly analyze a large number of instances of an architecture, and to generate design guidance for a complex system with multiple interdependencies.
- SODA parameters have an intuitive meaning, thus they are directly related to the causes of observed, and possibly emergent, behavior. This is particularly relevant in the case of SoS and complex systems, where the overall behavior is heavily affected by the dependencies between the component systems, and cannot be directly inferred by a mere analysis of the behavior of the individual components. With SODA model, the user can identify criticalities of the system, and ways to improve its global behavior.
- SODA parameters may come from a variety of sources, including expert evaluation, and historical data. This reduces the amount of simulation required to model interactions between systems. The three-parameter model has been validated through comparison with results of ABM simulations for various applications, and it models the global behavior of a system with more detail than previously proposed models (linear Input/Oputput, FDNA).
- Based on the result of analysis with the SODA model, metrics can be built to evaluate the robustness and resilience of a system architecture to disruptions and failures, accounting for the cascading impact of dependencies.
- The parametric model supports decision making in design and update of the systems architecture. SODA allows for quick analysis and exploration of different architectures; thus, the design space can be further explored, keeping and improving the most promising architectures, and discarding those that do not show an adequate level of desired features (robustness, reliability, resilience).

Being a surrogate model, SODA does not cover all the possible relationships between systems, and some models will perform better than SODA for more complex problems, where an increased level of detail is preferable. Likewise, when a pure deterministic or probabilistic analysis is required, and the user has no need to identify the underlying causes of the observed behavior, Surface Response Methodology or Bayesian networks can be a better choice. The identification of SODA parameters has a complexity which is at most equal to identifying the parameters of a surface or the conditional probabilities in a Bayesian network via simulations and experiments, but can be simplified by assumptions based on knowledge of some of the input/output relations. However, other methodology may be preferable (or used together with SODA) in problems where detailed and accurate modeling is deemed more important than simplicity, low computational cost, support

to decision making and management, and complex cascading failures and risk analysis.

A small Naval Warfare SoS has been presented to demonstrate the application of SODA. More complex applications involve the analysis of the evolution of operability over time, and rules to dynamically modify the network under given circumstances, so as to include the effect of flexibility on the behavior of the complex system. Other future improvements include the development of standardized metrics for desired features in specific applications, and formal integration of SODA with classic Systems Engineering methodology.

Acknowledgments This material is based upon work supported, in whole or in part, by the US Department of Defense through the Systems Engineering Research Center (SERC) under Contract HQ0034-13-D-0004 RT #108. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

References

Augustine M, Yadav OP, Jain R, Rathore A (2012) Cognitive map-based system modeling for identifying interaction failure modes. Res Eng Des 23:105–124

Bartolomei JE, Hastings DE, de Neufville R, Rhodes DH (2012) Engineering systems multiple-domain matrix: an organizing framework for modeling large-scale complex systems. Syst Eng 15:41–61

Box GEP, Wilson KB (1951) On the experimental attainment of optimum conditions (with discussion). J R Stat Soc Ser B 13(1):1–45

Browning T (2001) Applying the design structure matrix to system decomposition and integration problems: a review and new directions. IEEE Trans Eng Manag 48(3):292–306

Chow R, Braun D, Fry D (2012) DAF: discrete agent framework programmer's manual. Purdue University, West Lafayette

Clarkson J, Simons C, Eckert C (2004) Predicting change propagation in complex design. J Mech Des 126:788–797

Dahmann J, Smith K (2012) Integrating systems engineering and test and evaluation in system of systems development. In: IEEE international systems conference, Vancouver

de Weck OL, Ross AM, Rhodes DH (2012) Investigating relationships and semantic sets amongst system lifecycle properties (ilities). In: Third international engineering systems symposium. Delft

Eppinger S, Browning T (2012) Design structure matrix methods and applications. MIT Press, Cambridge

Fang C, Marle F (2012) A simulation-based risk network model for decision support in project risk management. Decis Support Syst 52:635–644

Gaonkar R, Viswanadham N (2007) Analytical framework for the management of risk in supply chains. IEEE Trans Autom Sci Eng 4(2):265–273

Garvey P, Pinto A (2009) Introduction to functional dependency network analysis. In: Second international symposium on engineering systems, Cambridge

Garvey P, Pinto A (2012) Advanced risk analysis in engineering enterprise systems. CRC Press, Boca Raton

Garvey P, Pinto A, Santos JR (2014) Modelling and measuring the operability of interdependent systems and systems of systems: advances in methods and applications. Int J Syst Syst Eng 5(1):1–24

Guariniello C, DeLaurentis D (2013a) Dependency analysis of system-of-systems operational and development networks. In: Paredis CJJ, Bishop C, Bodne D (eds) Conference on Systems Engineering Research. Procedia Comput Sci 16:265–274

Guariniello C, DeLaurentis D (2013b) Maintenance and recycling in space: functional dependency analysis of on-orbit servicing satellites team for modular spacecraft. In: AIAA Space Conference. San Diego

Guariniello C, DeLaurentis D (2013c) Dependency network analysis: fostering the future of space with new tools and techniques in space systems-of-systems design and architecture. IAF International Astronautical Congress, Beijing

Guariniello C, DeLaurentis D (2014a) Communications, information, and cyber security in systems-of-systems: assessing the impact of attacks through interdependency analysis. In: Madni AM, Boehm B (eds) Conference on Systems Engineering Research. Procedia Comput Sci 28:720–727

Guariniello C, DeLaurentis D (2014b) Integrated analysis of functional and developmental interdependencies to quantify and trade-off ilities for system-of-systems design, architecture, and evolution. In: Madni AM, Boehm B (eds) Conference on Systems Engineering Research. Procedia Comput Sci 28:728–735

Haimes YY, Jiang P (2001) Leontief-based model of risk in complex interconnected infrastructures. J Infrastruct Syst 7:1–12

Haimes YY, Horowitz BM, Lambert JH, Santos JR, Lian C, Crowther KG (2005) Inoperability input-output model for interdependent infrastructure sectors. I: theory and methodology. J Infrastruct Syst 11:67–79

Hamraz B, Caldwell N, Clarkson J (2012) A multidomain engineering change propagation model to support uncertainty reduction in risk management in design. J Mech Des 134(10):100905. doi:10.1115/1.4007397

Hsu J, Clymer J, Garcia J, Gonzales E (2009) Agent-based modeling the emergent behavior of a system-of-systems. INCOSE Int Symp 19(1):1581–1590

Hutcheson R, McAdams DA, Stone RB, Tumer IY (2007) Function-based behavioral modeling. In: Proceedings of the 2007 ASME design engineering technical conferences and computers and information in engineering conference. Las Vegas

INCOSE (2015) Systems engineering handbook: a guide for system life cycle processes and activities. Wiley, Hoboken

Khuri A, Mukhopadhyay S (2010) Response surface methodology. WIREs. Comput Stat 2(2):128–149

Kurtoglu T, Tumer I, Jensen D (2010) A functional failure reasoning methodology for evaluation of conceptual system architectures. Res Eng Des 21:209–234

Maier M (1998) Architecting principles for systems-of-systems. Syst Eng 1(4):267–284

Mane M, DeLaurentis D, Frazho A (2011) A Markov perspective on development interdependencies in networks of systems. J Mech Des 133(10):101009. doi:10.1115/1.4004975

Maurer M, Lindemann U (2008) The application of multiple-domain matrix. In: IEEE international conference on systems, man and cybernetics, pp 2487–2493

McManus H, Hastings D, Warmkessel J (2004) New methods for rapid architecture selection and conceptual design. J Spacecr Rockets 41(1):10–19

Mearns AB (1965) Fault tree analysis: the study of unlikely events in complex systems. Boeing/UW System Safety Symposium

Moullec ML, Bouissou M, Jankovic M, Bocquet JC, Requillard F, Maas O, Forgeot O (2013) Toward system architecture generation and performance assessment under uncertainty using

Bayesian networks. J Mech Des 135(4):041002. doi:10.1115/1.4023514

Mour A, Kenley CR, Davendralingam N, DeLaurentis D (2013) Agent-based modeling for systems of systems. INCOSE Int Symp 23(1):973–987

Nai Fovino I, Masera M (2006) Emergent disservices in interdependent systems and system-of-systems. In: IEEE international conference on systems, man, and cybernetics. Taipei

Nguyen D, Shen Y, Thai M (2013) Detecting critical nodes in interdependent power networks for vulnerability assessment. IEEE Trans Smart Grid 4(1):151–159

Nunez M, Datta V, Molina-Cristobal A, Guenov M, Riaz A (2012) Enabling exploration in the conceptual design and optimisation of complex systems. J Eng Des 23(10–11):852–875

Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco

Rhodes DH, Ross AM, Nightingale DJ (2009) Architecting the system of systems enterprise: enabling constructs and methods from the field of engineering systems. In: IEEE international systems conference. Vancouver

Rinaldi S, Peerenboom J, Kelly T (2001) Identifying, understanding, and analyzing critical infrastructure interdependencies. IEEE Control Syst Mag 21(6):11–25

Sage A, Cuppan C (2001) On the systems engineering and management of systems of systems and federations of systems. Inf Knowl Syst Manag 2(4):325–345

Shaw G, Miller D, Hastings D (2001) Development of the quantitative generalized information network analysis (GINA) methodology for satellite systems. J Spacecr Rockets 38(2):257–269

Sosa M (2008) A structured approach to predicting and managing technical interactions in software development. Res Eng Des 19:47–70

Stone RB, Wood KL (2000) Development of a functional basis for design. J Mech Des 122(4):359–370

Stone RB, Tumer IY, Van Wie M (2005) The function-failure design method. J Mech Des 127(3):397–407

Taguchi G (1986) Introduction to quality engineering: designing quality into products and processes. Asian Productivity Organization. American Supplier Institute, Dearborn

Tumer IY, Stone RB (2003) Mapping function to failure mode during component development. Res Eng Des 14(1):25–33

United States Department of Defense (1949) MIL-P-1629. Procedures for performing a failure mode effect and critical analysis

Wang Y, Zhang W, Li Q (2014) Functional dependency network analysis of security of navigation satellite system. Appl Mech Mater 522–524:1192–1196

Watson HA (1961) Launch control safety study. Bell Labs, Murray Hill

Zio E, Sansavini G (2011) Modeling interdependent network systems for identifying cascade-safe operating margins. IEEE Trans Reliab 60(1):94–101