Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 16 (2013) 265 – 274

Procedia Computer Science

Conference on Systems Engineering Research (CSER'13)
Eds.: C.J.J. Paredis, C. Bishop, D. Bodner, Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013.

Dependency Analysis of System-of-Systems Operational and Development Networks

Cesare Guariniello[a,]*, Daniel DeLaurentis[a]

[a]*Purdue University, 701 West Stadium Avenue, West Lafayette, IN 47907, USA*

Abstract

In this research, a Dependency Network Analysis technique has been adapted to assess operability, reliability, and resilience in both operational and development networks, associated with System of System architectures.
The architecture is modeled as a directed network where nodes represent either the component systems or the capabilities to be acquired. Links on the network represent various kinds of dependencies between the constituent systems: functional dependency in an operational network, sequential development dependency in a development network. Each dependency is characterized by strength and criticality. The ultimate goal of the technique seeks to analyze effects of such dependencies -and of their strength and criticality- on operability, and to identify valid operating and developing strategies and architectures. For operational networks, Functional Dependency Network Analysis is used to assess the effect of topology and of possible degraded functioning of one or more systems on the operability of the network. For development networks, Development Dependency Network Analysis is used to assess how development time or capabilities are affected by the network topology and by delays in the development of component systems.
Each technique is evaluated with regard to amount and quality of necessary input, completeness and usefulness of results, and applicability to problems of diverse nature.

Keywords: dependencies; network; operability; development; SoS.

1. Introduction

The challenge of analyzing and architecting Systems-of-Systems (SoS) is very tough, not only due to complexity and size: dependencies between component systems are what most affect the behavior of the whole structure. Maier [1] defines SoS as a collection of systems that must have two features: its components must be able to operate independently by the whole system and they do operate independently, being managed at least in part for their own purpose. Since the elements are designed and developed independently, the aggregate emerges only through interaction of components. Also DeLaurentis and Callaway [2], when describing their proposed definition for a

* Corresponding author. Tel.: +1-765-490-1939.
E-mail address: cguarini@purdue.edu.

complex system and a SoS, underline the dependency of functionality on linkages between components in a SoS. Another feature of SoS is emergence ([1], [3]): SoS can show qualities that are irreducible to the constituent parts ([4]), and depend on how the component systems interact. Emergent behavior is what cannot be predicted through analysis at any levels simpler than that of the system as a whole ([5]). In [6], emergent behaviors are defined as characteristics that arise from the cumulative actions and interactions of the constituents of a SoS, and emergence can be used to develop flexible and robust SoS. Such definitions of emergent behavior will be used in this paper.

System engineers struggle with complex dependencies between systems and between capabilities to be achieved, in both development and operational relationships. A development relationship means that the development of a certain system is dependent from the development of another, but this relationship not necessarily affects their functioning. Instead, an operational relationship means that a certain system needs input (data, material, energy) from another system to operate. A brief historical survey of the use and definition of SoS by government agencies shows that such systems are usually not formally designed, planned or dealt with as SoS: instead, solutions are usually found for the specific problem of interest [7].

The conclusion of such considerations is the need for general design methods, guidelines, analysis techniques, and metrics to support decision making in different SoS contexts at different levels. Such tools should address complexity, and above all they should account for the importance of dependencies between systems and between requirements, and the consequent emergent behavior.

1.1. Dependency Network Analysis

In this research, a technique has been adapted, with the goal of assessing the impact of dependencies in a SoS. The architecture of SoS has been modeled as a directed network. The nodes represent either the component systems or the capability to be acquired. Accordingly, the links represent the dependencies between the systems or between the capabilities. Each dependency is characterized by strength and criticality, that affect the behavior of the whole SoS in different ways: strength of dependency accounts for how much the behavior of a system is affected by the behavior of a predecessor system, while criticality of dependency quantifies how the functionality of a system is degraded when a predecessor system is experiencing a major failure.

The Dependency Network Analysis technique has been first applied to operational networks based on the Functional Dependency Network Analysis (FDNA). As in the original formulation by Garvey and Pinto ([8], [9]), this method 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. To adapt the technique to SoS analysis, a term has been added to account for possible degraded functioning of any component system due to its own malfunctions (the formulation in [8] represents only malfunctions in nodes that do not depend by other nodes). Further novelty presented in this paper is the test of stochastic analysis with FDNA, involving a probability distribution for the operability of the systems. FDNA technique identifies the most critical nodes in the network, as well as the most important dependencies. Comparison of different architectures can be performed, by assessing the operability of the systems. The resilience of a SoS can be evaluated in terms of capability to reduce the loss of operability when single systems are affected by partial failures. Further analysis can be executed to assess the benefits of adding or removing systems.

Finally, a new technique, the Development Dependency Network Analysis (DDNA), has been proposed and tested, and preliminary results of analysis of dependencies in a development network are reported. DDNA borrows the concepts of strength and criticality of dependencies from FDNA, and uses a network representation of the dependencies, as in Program Evaluation and Review Technique (PERT) and Critical Path Method (CPM). DDNA is used to assess how development time or capabilities are affected by the network topology and by possible delays in the development of component systems.

2. Functional Dependency Network Analysis

2.1. The method

In FDNA, the SoS is represented as an operational network, with the nodes being component systems or capabilities to be achieved. The links represent a dependency of the operability of a node (successor node) from another (predecessor node).

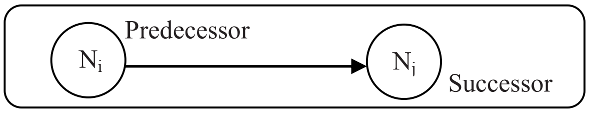


Fig. 1. Operational dependency of node N_j from node N_i

The operability of a node is defined as the "percentage" of effectiveness, that is the level at which the system is currently operating, or the level at which the desired capability is being currently achieved. Operability, ranging between 0 and 100, can be related to performance by means of an input function. In the example in fig. 2, a planetary probe communication system is evaluated based on the number of valid data downlinks per week (performance), with its operability (effectiveness) being 100 when the system performs 1000 valid data downlinks per week, and the degraded operability being 25 when the system performs 380 valid data downlinks per week.

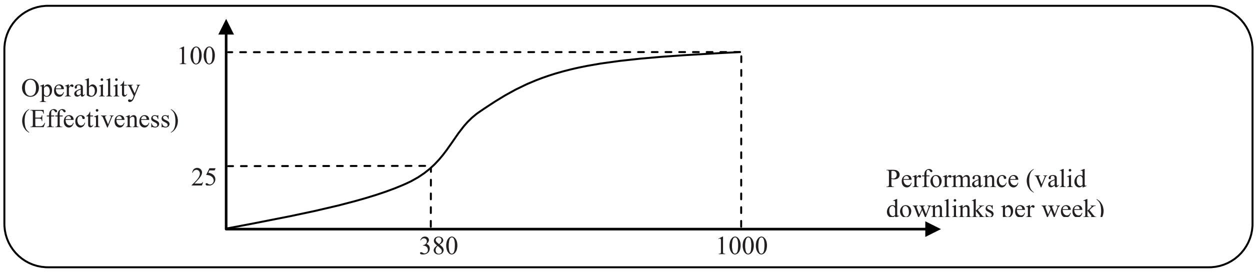


Fig. 2. Correlation between Performance and Operability of a system.

Using this relationship, the operability of root nodes, i.e. nodes that have no predecessors, can be evaluated based on their current performance. FDNA analysis, then, allows for the computation of successor nodes operabilities, which can in turn provide a value for the nodal performance. In this paper, the function relating performance and operability is assumed to be a given input, and the method is described on the basis of the operability.

Further input is required:

for each node N_i, a self-effectiveness level SE_i is needed, ranging between 0 and 100. For root nodes, this is just the operability; for nodes that have at least one predecessor, the self-effectiveness is the level of operability that the node would have, if it were a root node: therefore, the self-effectiveness assess the current status of a node, not accounting for its dependencies.

For each link, two values are needed. The strength of dependency (SOD) between node N_i and node N_j, α_{ij}, and the criticality of dependency (COD) between node N_i and node N_j, β_{ij}. α_{ij} must be between 0 and 1, and can be evaluated as the fraction of the operability level of node N_j due to the dependency by node N_i. For example, if the operability level of node N_j (working at self-effectiveness equal to 100) is 70, then α_{ij} is equal to 0.3. β_{ij} must be comprised of values between 0 and 100, and it can be evaluated as the maximum level of operability reachable by node N_j, when the operability of node N_i is 0. A lower value β_{ij} corresponds to a higher criticality of dependency.

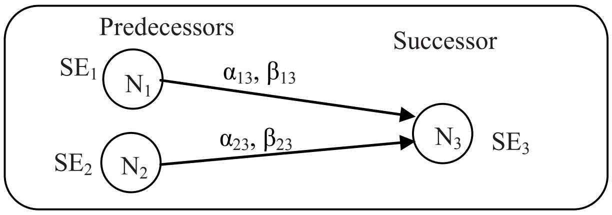


Fig. 3. A small network, showing the required input for FDNA

According to the method proposed in [8], the operability of root nodes is simply their self-effectiveness:

$$O_i = SE_i \tag{1}$$

The operability of nodes that have at least one predecessor is computed as the minimum of two terms, one depending on the SODs, one depending on the CODs:

$$O_j = min(SOD\_O_j, COD\_O_j) \tag{2}$$

For a node N_j having *n* predecessors, the two terms are computed according to equations 3-6:

$$SOD\_O_j = Average\ (SOD\_O_{j1,} SOD\_O_{j2}, \ldots , SOD\_O_{jn}) \tag{3}$$

$$SOD\_O_{ji} = \alpha_{ij}O_i + (1-\alpha_{ij})SE_j \tag{4}$$

$$COD\_O_j = Min\ (COD\_O_{j1,} COD\_O_{j2}, \ldots , COD\_O_{jn}) \tag{5}$$

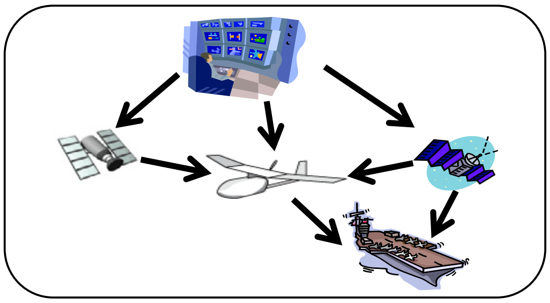
$$COD\_O_{ji} = O_i + \beta_{ij} \tag{6}$$

The term accounting for SOD is the average operability values of node $N_{j,}$ computed for each dependency from a predecessor node N_i, thus reflecting the relationships between the *n* predecessors and the node N_j. The term accounting for COD is the minimum of the values of the operability of node N_j computed for each dependency from a predecessor node N_i, thus precisely reflecting the importance of the most critical dependency.

Using equations 1-6, the operability of each node can be sequentially computed, starting from the root nodes, in a breadth-first way: after the roots, nodes directly depending from the root are analyzed, and so on.

2.1. Application to a simple example of Aerospace SoS

To assess the value and applicability of FDNA for SoS, the method is applied to a simple five-node aerospace network (fig. 4).



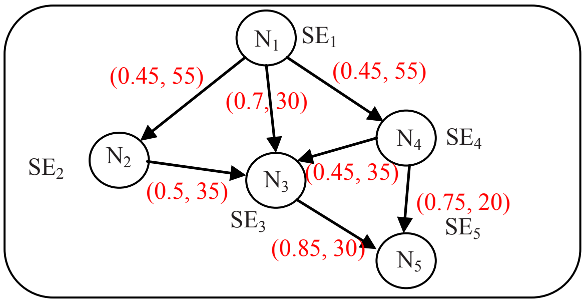


Fig. 4. (a) the five-node aerospace SoS; (b) the same SoS represented as a network, with the required input for FDNA: self-effectiveness of each node, and COD and SOD of each link.

The network includes ground facilities (N_1), two satellites (N_2 and N_4), a UAV (N_3), and a ship (N_5). The links represent communication and data dependencies for location: the satellites and the UAV need data from the ground facilities, the UAV also uses the satellites for navigation, and the ship gets data from the UAV and from one of the satellites. The input can be given in form of a vector for the self-effectiveness, and two matrices containing the terms α_{ij} and β_{ij} (nonzero entries in the SOD matrix correspond to dependencies in the network).

$$SE = [SE_1\ SE_2\ SE_3\ SE_4\ SE_5] \qquad SOD = \begin{bmatrix} 0 & 0.45 & 0.7 & 0.45 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.85 \\ 0 & 0 & 0.45 & 0 & 0.75 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad COD = \begin{bmatrix} 0 & 55 & 30 & 55 & 0 \\ 0 & 0 & 35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 35 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.1.1. Deterministic analysis

In this kind of simulation, different values for the self-effectiveness, i.e. degraded operability of system, are fed into the equations, to compute the actual operability of each system. Table 1 shows the results of degraded operability (level of operability equal to 75 or to 25) in a single system.

Table 1. Degraded self-effectiveness in single systems. O=operability of the nodes

Self-effectiveness [N_1 … N_5]	O_1	O_2	O_3	O_4	O_5
[75 100 100 100 100]	75	88.75	90.60	88.75	91.79
[25 100 100 100 100]	25	66.25	55	66.25	68.22
[100 75 100 100 100]	100	86.25	97.71	100	99.03
[100 25 100 100 100]	100	58.75	93.13	100	97.08
[100 100 75 100 100]	100	100	87.5	100	95.22
[100 100 25 100 100]	100	100	66.25	100	85.66
[100 100 100 75 100]	100	100	97.94	86.25	93.97
[100 100 100 25 100]	100	100	93.75	58.75	78.75
[100 100 100 100 75]	100	100	100	100	95
[100 100 100 100 25]	100	100	100	100	85

Analysis of these results give good insight into the influence of dependencies into such network: first of all, the five-node SoS appears to have high resilience in these cases: any single failure that degrades the operability of a system down to 25, can at most degrade the operability of the ship down to 68.22. The second result from FDNA analysis is the identification of the most critical nodes, in this case nodes N_1 and N_4 most affect the operability of other nodes. While some result is anticipated (for example, since there is a path from node N_1 to any other node, its influence is expected), other behaviors are captured by FDNA. For instance, node N_4 shows positive criticality for what concerns the dependency of node N_5: when the satellite N_4 is working at level 100, the operability of the ship is never lower than 85, even if other systems influencing N_5 are degraded. Finally, FDNA allows more complex analysis, based on the output of interest: for example, it can be noted that, if the operability of the ship is the object of the evaluation, highly degraded operability of node N_3 is worse than highly degraded operability of node N_2; if instead the global operability of the network, computed as the sum of the operability of each node, is the measure of interest, highly degraded operability of node N_3 gives better results than highly degraded operability of node N_2 (noticeably, this is not true if the operability decreases to 75. This is due both to the CODs and SODs, and to the topology, sources of complexity in the network. To better catch these details, stochastic analysis can be performed).

2.1.2. Stochastic analysis

A more realistic view of the behavior of a SoS as a function of the dependencies between the component systems can be achieved by means of a stochastic analysis with the FDNA technique. This kind of analysis is useful to capture and summarize all the aspects related to SOD and COD, as well as to topology, in a few probability distributions (as reported above, such details are spread in the tables for deterministic FDNA). Since the five-node network is very small, a Monte Carlo simulation has been executed.

The computation of the expected value for the operability of a system gives a measure of the resilience of such system to failures of the predecessors, while the variance evaluates the sensitivity of the system to failures of the predecessors. Differently from the deterministic analysis, this evaluation is not based on the simulation of single instances (that could for example neglect some COD), but it accounts for any possible combination of the effects of COD, SOD, and topology. The capability of this kind of analysis to capture behavioral patterns and features of a whole architecture makes it suitable to be used as a decision tool.

The analysis confirmed that the systems in the five-node network are resilient to single failures in the predecessors, with a probability distribution shifted towards high values of operability (fig. 5a).

The complexity of the interdependencies between systems, resulting in more complex behavior, arises even if only two system experience a degraded operability in the simple five-node network, as shown in fig. 5b.

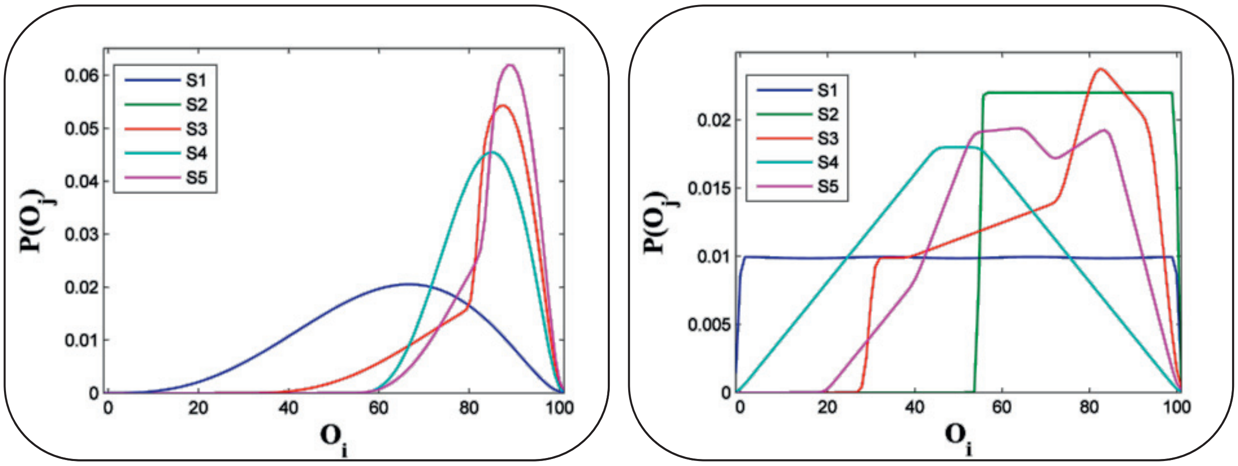


Fig. 5. Probability distribution for the operability in the five-node system. (a) single failure in node N_1, beta probability distribution for self-effectiveness of node N_1; the probability distribution of S2 coincides with that of S4; (b) failure in nodes N_1 and N_4, with independent uniform distribution: node N_3 is in this case more resilient than node N_5, differently from what appears in 5a

2.2. Example of greater complexity

An additional scenario is executed using a more complex twelve-node network (fig. 6), with three ground facilities (N_1, N_2, N_3), two satellites (N_4, N_5), two UAVs (N_6, N_7), a ship (N_8), an airplane (N_9), and three nodes corresponding to desired capabilities: long range detection (N_{10}), short range detection (N_{12}), and rescue (N_{11}). The results further demonstrate the power of FDNA in comparing different architectures, as well as the result of removal of nodes or dependencies from a system. Four deterministic tests are reported and results are shown in table 2.

Interesting outcomes have been achieved, showing unexpected behavior and patterns not directly predictable through the knowledge of the component systems, that is one of the definitions of emergence in the SoS. If node N_7, i.e. one of the UAVs, is removed from the SoS (meaning that the short range observation is based only on the ship and the airplane), when node N_1 has degraded self-effectiveness, the operability of node N_{11} unexpectedly increases. The operability of node N_{12}, instead, slightly decreases when nodes N_1, N_5, or N_8 are self-effective at level 20, but node N_{12} is not anymore affected by degraded self-effectiveness of node N_2. Therefore, such architecture could be preferable if nodes N_1 and N_2 is prone to failures.

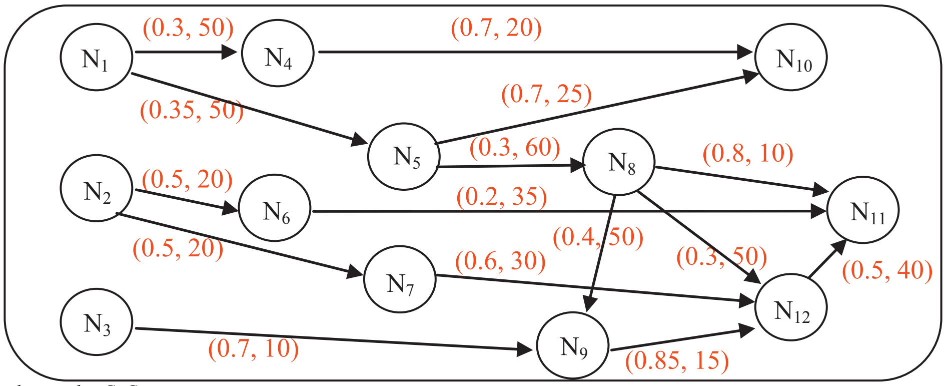


Fig. 6. The twelve-nodes SoS

When the dependency of node N_9 on node N_8 is removed, the operability of nodes N_{11} and N_{12} globally increases. Noticeably, if node N_9 is having a degraded self-effectiveness, its dependency from a single node (N_3) gives a high operability, resulting in better operability of nodes N_{11} and N_{12}.

Finally, a small architecture change has been tested: node N_5 is dependent from node N_2 instead than N_1, and node N_6 is dependent from node N_1 instead than N_2. This new architecture gives lower operability in some case, but it can be noticed that node N_{10} has a smoother behavior, with the operability being the same for degraded self-effectiveness of nodes N_1 and N_2. The architecture can then be evaluated based on the importance of the requested capabilities (N_{10}, N_{11}, N_{12}), and on the probability of failures of N_1 and N_2.

Table 2. Twelve-node SoS experiment: indicated system is having a self-effectiveness of 20 (single failure in other systems gave the same results for all the experiments), while all other systems have self-effectiveness equal to 100

Test	O_{10}, O_{11}, O_{12} (N_1=20)	O_{10}, O_{11}, O_{12} (N_2=20)	O_{10}, O_{11}, O_{12} (N_5=20)	O_{10}, O_{11}, O_{12} (N_8=20)	O_{10}, O_{11}, O_{12} (N_9=20)
Basic network	79, 96.36, 98.59	100, 75, 70	69, 93.2, 97.37	100, 54, 91.23	100, 96.5, 79
Node removal	79, 97.25, 97.89	100, 75, 100	69, 93.2, 96.05	100, 54, 86.84	100, 96.5, 79
Link removal	79, 97.45, 99.1	100, 75, 70	69, 93.2, 98.32	100, 54, 94	100, 98.5, 91
Diff. architecture	89.5, 75, 100	89.5, 92.6, 70	69, 93.2, 97.37	100, 54, 91.23	100, 96.5, 79

3. Development Dependency Network Analysis

A Development Dependency Network Analysis (DDNA) method, based on the concepts of SOD and COD from FDNA, is developed. It is applied to development SoS networks, where the links, like in PERT [10], represent development dependencies between systems. The outcome of such analysis is the beginning time and the completion time of the development of each system, as well as an assessment of the combined effect of multiple dependencies and possible delays in the development of predecessors. As in FDNA, this method evaluates the most critical nodes and dependencies, and can be used to compare different architectures in term of development time. Also, if some of the nodes are capabilities to be achieved, the method assess the time, or the expected time, in probabilistic analysis, by which each capability is available. The method has been developed to also account for the possibility of measuring partial capabilities attained during the development of a SoS.

3.1. The method

Compared to existing methods, such as PERT/CPM [10], DDNA provides more specific insight into the effects of multiple and diverse dependencies on the development of SoS. These include:

- The strength of dependency affects both the beginning time and the completion time of development of a system. Differently from PERT, development of a system can start before a predecessor is complete, if the dependency has not reached criticality. Therefore, a more realistic analysis of development time is achieved. Above all, this feature allows DDNA to assess partial capabilities during the development of a SoS.
- COD affects the beginning time in the same way as in PERT/CPM: a successor must wait until a critical predecessor is complete to begin development. Instead SOD results in a less absolute dependency.
- Differently from FDNA (where performance is related to operability), in DDNA time is directly used into the computation, and the level of performance constitutes an assessment of the quality of the development (for example, three weeks could be the best time for a system, ten weeks could be the worst, but eight weeks could correspond to a satisfaction of 50%).

In DDNA, each node requires three pieces of input data: the minimum independent time (MINIT), that is the minimum duration of development of the system; the maximum independent time (MAXIT), that is the maximum duration of development of the systems; the self-effectiveness (SE) that linearly evaluates how much the system is close to being developed with its minimum duration time, without accounting for dependencies. If SE=0, then the independent duration of development is equal to MAXIT. If SE=100, then the independent duration of development is equal to MINIT. Due to dependencies, the actual time to develop a system can be longer than MAXIT (but in this case the system must have begun its development before the completion of a predecessor, thus resulting in earlier partial capabilities), whereas it can never be shorter than MINIT.

Each link requires two input data: the strength of dependency (SOD) and the criticality of dependency (COD). The SOD evaluates how much a system can begin its development before the completion of the predecessor, it constitutes a parameter for the parabolas shown in fig. 7a. This shape has been chosen because it results in anticipated development for systems depending from a node with medium-high SE, while no early development is allowed when the predecessor is being developed either in its best time, or with a SE below the critical threshold. In future research, these curves will be compared to data from industrial projects and consequently adapted.

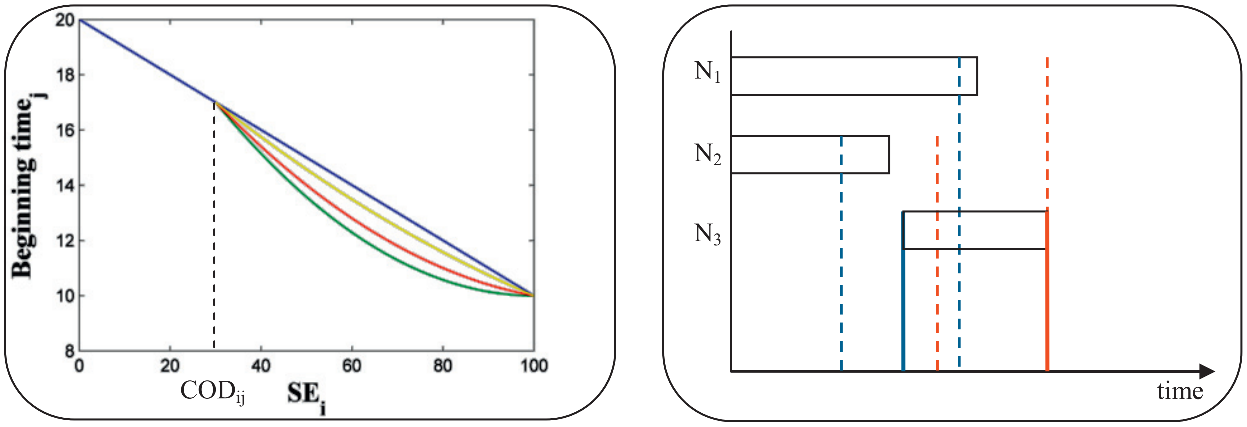


Fig. 7. (a) the dependency between node N_i and node N_j. If SE_i is lower than COD_{ij}, then the beginning time of N_j coincides with the completion time of N_i. Otherwise, the development of N_j can begin earlier: the blue line relates SE_i to the completion time of N_i (coincident with the beginning time of N_j when SOD_{ij}=1). The yellow, red, and green parabolas correspond respectively to SOD_{ij}=0.7, SOD_{ij}=0.3, SOD_{ij}=0; (b) Multiple dependency of N_3 from N_1 and N_2 in a Gantt diagram fashion. Blue lines are the beginning times, red lines are the completion times (dotted=due to single dependency, continuous=actual times due to multiple dependency)

The COD is the minimum level of SE of the predecessor that allows an early development of the successor. If the SE of the predecessor is lower than the COD, then the successor must wait until the predecessor is fully developed. Higher COD corresponds to higher criticality.

For root nodes N_i, the beginning time (BT_i) is 0, and the actual completion time (CT_i) is computed as

$$CT_i = MINIT_i + (100-SE_i)\,(MAXIT_i-MINIT_i)/100 \qquad (7)$$

That is, depending on its SE. For nodes having a single predecessor, the beginning time is computed according to the function shown in fig. 7a: if $SE_i < COD_{ij}$, then $BT_j=CT_i$. Otherwise, BT_j is computed as

$$BT_j = a\,SE_i^2 + (-a\,(100+COD_{ij})+(B-D)/(100-COD_{ij}))\,SE_i+(100\,D-B\,COD_{ij})/(100-COD_{ij})+100aCOD_{ij} \qquad (8)$$

Where B is the minimum actual completion time for node N_i, D is the completion time of node N_i corresponding to COD_{ij}, and a is equal to $(1-SOD_{ij})\,(D-B)/(100-COD_{ij})^2$. This formulation corresponds to the parabolas in fig. 7a, and guarantees that node N_j cannot have a beginning time lower than the minimum completion time of node N_i.

Given a development time of node N_j equal to

$$DT_j = MINIT_j + (100-SE_j)\,(MAXIT_j-MINIT_j)/100 \qquad (9)$$

that is a linear relationship between the SE of a node and its development time, the completion time is

$$CT_j = Max(BT_j + DT_j,\ CT_i + SOD_{ij}\,DT_j) \qquad (10)$$

The first term is the completion time that node N_j would have starting at BT_j and not having any dependency from N_i. However, the development time, added to the anticipated beginning time, could result in the successor node N_j being fully developed before the completion of the predecessor node N_i. Therefore, the second term accounts for this dependency, stating that node N_j cannot complete its development before a certain amount of time after the completion of node N_i elapses (this amount of time being dependent on SOD_{ij} and development time of N_j). Equations 9 and 10 are also used to compute the minimum and the maximum completion time for node N_j.

For nodes having multiple predecessors, the beginning time is computed as the average of the beginning times given by each dependency. Even if a node has got a critical dependency from one or more predecessors, it can still begin its development based on the dependency from other nodes. Criticality, however, affects the completion time. Using the average, instead than the minimum, prevents a single predecessor from critically influence the beginning time. The completion time is the maximum of the completion times given by each dependency.

3.2. Preliminary results of the method

The method has been tested with simple networks to get an insight into the result attainable through this analysis. A deterministic analysis of a five-node network is reported in table 3 (matrices for the network are stated below).

Table 3. Preliminary results with DDNA method. BT=Beginning Time. CT=Completion Time

Self-effectiveness	BT_1	CT_1	BT_2	CT_2	BT_3	CT_3	BT_4	CT_4	BT_5	CT_5
[100 100 100 100 100]	0	7	0	10	8.5	16	0	5	10.5	24
[20 100 100 100 100]	0	11	0	10	10.7	17.4	0	5	11.2	25.4
[100 20 100 100 100]	0	7	0	18	12.5	24	0	5	14.5	32
[100 100 20 100 100]	0	7	0	10	8.5	25.6	0	5	16.74	36.48
[100 100 100 20 100]	0	7	0	10	8.5	16	0	13	14.56	24
[100 100 100 100 20]	0	7	0	10	8.5	16	0	5	10.5	33.6
[10 10 100 100 100]	0	11.5	0	19	15.25	25	0	5	15	33
[90 10 100 10 100]	0	7.5	0	19	13.19	25	0	14	19.5	33

Various observations can be done even with this few results: for example, the high SOD between nodes N_4 and N_5, and N_3 and N_5 causes N_5 to have a longer development than N_3, even when N_3 starts to be developed later. Also, the capability of the SoS to partially recover from delays is evident, above all when delays strike the root nodes, and successor nodes have low SOD (so they can begin their development, adding partial capability to the overall SoS). In PERT/CPM analysis, a delay on a critical path would never be recovered, nor the analysis would account for the possibility that the System-of-System could be partially developed in the meantime.

$$[MINIT, MAXIT] = \begin{bmatrix} 7 & 12 \\ 10 & 20 \\ 6 & 18 \\ 5 & 15 \\ 8 & 20 \end{bmatrix} \qquad SOD = \begin{bmatrix} 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad COD = \begin{bmatrix} 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 30 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. Conclusions and future work

4.1. Applicability of the methods and quality of results

In this paper, two methods for the analysis of the effect of dependencies between the components of a SoS have been presented. Functional Dependency Network Analysis is applicable to operational networks, and Development Dependency Network Analysis is applicable to development networks. FDNA quantifies the effect of the dependencies on the operability, and DDNA quantifies the effect of the dependencies on the development time. Both techniques can identify the most critical nodes and dependencies, and their effect on the operability or development of successor nodes, and can compare different architectures. The analysis can be deterministic or probabilistic, and be used to analyze specific or global operability and development behavior. The relationship between performance and operability (or performance and time), and the use of a graph description allows these methods to deal with problems of different nature and various fields of application: for example, complex aerospace systems, design schedule, industrial production, social relationships can be analyzed by FDNA and DDNA. However, even if both methods are generally applicable to any SoS that can be represented by an operational or development network, virtual SoS (meaning that each component systems is completely independent and is not even partially designed according to the goals of the entire SoS), especially if asynchronous, are hard to treat with such methods, since the SOD and COD of the dependencies would be difficult to evaluate, and could change over time.

Both methods are being developed as part of research for the Department of Defense. The aim is to assess the effects of dependencies in SoS and to create analysis and decision tools specific to this kind of system. Those tools are required to account for typical traits of SoS: complexity, size, partial autonomy in development and operability of the component systems, emergent behavior caused by the complex relationships between the systems.

Output of FDNA and DDNA gives a detailed insight into the effects of dependencies, underlining unintuitive emergent behavior of the SoS, as described in the examples. FDNA can quantify the resilience of a SoS, and evaluate different architectures with respect to their operability when degraded functionality arises. DDNA uses the properties of dependencies (strength and criticality) to assess the development time, and the effect of delays on the development, more realistically than PERT/CPM. When criticality is not reached, partial development of a system can begin before a predecessor is complete. DDNA is able to capture the capability of a network to absorb a delay even along the critical path and is also suitable to assess partial capabilities available during the development of a SoS. However, both methods require many input data: strength and criticality of each dependency in the networks has to be evaluated (by experts, or getting data from simulations, or just using reasonable values), as well as the possible shape of the network. For DDNA, the minimum and maximum development time for each system are further required inputs.

4.2. Future work

Potential future work will address the following:

- FDNA analysis has to be tested with larger networks, featuring complex interdependencies between component systems; the results can be compared to those of other analysis techniques and metrics, looking for patterns and characteristics relating the features of nodes, links, and architecture of networks (like degree, centrality, weights) to the outcomes of FDNA.
- DDNA analysis is very promising: different shapes for the functions relating the development time of dependent systems can be tested. Also, DDNA will be directly compared to PERT/CPM analysis. Stochastic analysis, though very complex, can be added to this method, in the same way as it has been executed in FDNA. However, the two most important improvements that will be added to this technique involve a tool for optimization vs. cost analysis, similar to that of CPM, and a metric to asses partial capabilities achieved by the SoS during the development of its components.
- For both methods, analysis of data from SoS design found in literature, and comparison with the output from the methods will be performed. In the meanwhile, an Agent Based Model has been developed to represent a naval warfare SoS, and is currently being used to obtain the required input for FDNA.

Acknowledgement

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

References

1. Maier, M., “Architecting Principles for SoS”, Systems Engineering, Vol. 1, No. 4 1998, pp. 267-284.
2. DeLaurentis, D., and Callaway, R., “A SoS Perspective for Public Policy Decisions”, Review of Policy Research, Vol. 21, No. 6, 2004.
3. Sage, A., and Cuppan, C., “On the Systems Engineering and Management of Systems of Systems and Federations of Systems”. Information, Knowledge, Systems Management, Vol. 2, No. 4, 2001, pp. 325-345.
4. Laughlin, R., *A Different Universe: Reinventing Physics from the Bottom Down*, Basic Books, 2005.
5. Yang, K., and Chen, Y., and Lu, Y., and Zhao, Q., “The Study of Guided Emergent Behavior in System of Systems Requirement Analysis”, 5th International Conference on System of Systems Engineering, 2010.
6. Hsu, J., “Emergent Behavior of Systems-of-Systems”, INCOSE Mini-Conference, 2009.
7. DeLaurentis, D., and Crossley, W., and Mane, M., “Taxonomy to Guide SoS Decision-Making in Air Transportation Problems”, Journal of Aircraft, Vol. 48, No. 3, 2011.
8. Garvey, P., and Pinto, A., “Introduction to Functional Dependency Network Analysis”, MIT, 2009.
9. Garvey, P., and Pinto, A., *Advanced Risk Analysis in Engineering Enterprise Systems*, CRC Press, 2012.
10. Blanchard, B., and Fabrycky, W., *System Engineering and Analysis,* 3rd ed., Prentice Hall International Series in Industrial and Systems Engineering, 1998, Chaps. 1, 2, Appendix A.