Systems Developmental Dependency Analysis for Schedule and Decision Support

Abstract — In this paper, we introduce the Systems Developmental Dependency Analysis methodology. Its purpose is to assess the impact of partial developmental dependencies between components in a monolithic complex system, or between systems in a System-of-Systems. We propose a parametric model of the developmental dependencies. This approach results in a simple, intuitive model, whose output is a schedule of the development of the system, or System-of-Systems, accounting for partial dependencies. Using the proposed method, the expected lead times, and the beginning and completion time of the development of each system can be automatically scheduled and re-scheduled. These times are evaluated based on the current and expected performance of each system (in terms of development time), on the model of the dependencies, and on the amount of accepted risk. Rules that dynamically change the parameters of the dependencies, based on stakeholders’ decision, or on development deadlines, can be added to the basic model for further analysis. The Systems Developmental Dependency Analysis method supports educated decision making in the development and update process of systems architecture. In particular, throughout the whole development phase, the information given by the method can be used to identify criticalities, and bottlenecks, to quantify possible partial delay absorption, and to assess the best time to begin the development of each system, accounting for development cost, decision by other stakeholders, and risk. Application of the proposed method to a small example is used to demonstrate the capabilities of the method for system development analysis.

Keywords — dependencies, development, schedule, risk, system architecture, System-of-Systems

Introduction

As the complexity of systems grows, the management of their development schedule becomes more difficult. This is particularly true in the context of Systems-of-Systems (SoS). Maier (1998) 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 behavior emerges only through interaction of components, and this interaction will drive part of the development schedule. In such settings, and when a large number of systems is involved, it may not be possible to decompose into smaller entities to analyze the developmental dependencies. The user may need a high-level viewpoint of
development, which includes development decisions by independent stakeholders. Moreover, the individual entities under development may exhibit partial dependencies, where only a fraction of the development of a system is dependent on the development of other systems. Thus, a system $j$ can be developed with a lead time, that is a period of development of $j$ occurring before the full development of a system $i$, on which $j$ is partially dependent. Finally, knowing the impact of delays on the overall development, and the “right time” to start the development of a system, is a result more important than just knowing the schedule of the development of the complex system. By “right time” we mean a time that allows for trade-off between the exploitation of the possible lead time for early completion of development, or delay absorption, and the higher cost and risk associated with an early start, and a possible longer time to develop the system. This capability would have direct impact on acquisition of complex systems like aircraft and shed light on the approach of concurrency (Birchler et al. 2010, Fleischer and Lik er 1997, Garland and Colombi 2011).

Traditional approaches often comes short in managing such features. Dependencies between systems are usually modeled as absolute, or “on-off”, like in Project Evaluation Research Technique (PERT), and Critical Path Method (CPM), where a system has to wait until the predecessor is fully developed (Wiest and Levy, 1977). In other cases, experts in management plan the lead time according to their experience, but without dynamically changing it based on the reliability and delays of the current development schedule.

To address the limitations of traditional approaches in development schedule when dealing with complex systems, we propose Systems Developmental Dependency Analysis (SDDA). The goal of the method is to model and analyze developmental dependencies between systems, and to assess the impact of such dependencies, via the cascading propagation of delays. We model the system as a directed dependency network, where nodes represent the component systems (or subsystems). The edges represent the developmental dependencies between the constituent parts. A developmental dependency means that a certain system needs input (information, data, energy, etc.) from another system in order to be developed. Each dependency is modeled with two parameters, one accounting for the strength of the dependency, and one for the criticality of the dependency. Using a parametric model, we have two advantages: first, the parameters have an intuitive meaning, directly related to the features of the dependency. The result is a more accurate model than PERT, a model capable of representing and analyzing partial dependency and to produce dynamically changing schedule, while at the same time keeping a simple analytical form, useful for fast computation and simulation. The second advantage is that the model is simple enough to be used to quickly analyze the impact of a wide variety of delays and disruptions, either deterministic or stochastic, supporting high-level management decision for the development of complex systems. The parameters of the model may come from various sources, including expert evaluation, considerations about stakeholder decisions, and historical data.
With SDDA, we compute the impact of delays and stakeholder decisions on the overall development, based on the topology of the development network, and on the features of the dependencies. Based on evaluation of impact and propagation of delays, the user can identify critical systems and dependencies, quantify the robustness of the development network, in terms of delay absorption, quickly analyze the effect of stakeholder decisions, and evaluate different developmental choices under various levels of reliability and risk acceptance. The user can also perform trade-off between competing desired features, for example, time to complete the overall development, risk, capability to absorb delays, and time spent on each system. The analysis can be repeated during the system development, and the schedule be dynamically re-computed via SDDA.

After describing the methodology, the meaning of the parameters representing the dependencies, and the analytical model, we show examples of analysis with SDDA on a synthetic case study.

**Related Work and Prior Research**

**Dependencies and risk propagation**

Various authors have proposed methodologies to deal with systems dependencies and with the analysis of risk propagation among systems. The effect of dependencies in SoS development has been analyzed by Mane and DeLaurentis (2010a, 2010b), and by Mane et al. (2011) by means of a Markov network approach, meant to evaluate delay propagation before absorption. Whereas the method can be used to rank the systems based on the criticality of their impact on the development of the System-of-Systems, it does not account for partial dependencies, and the delay can propagate only to one of the systems dependent on the delayed system.

Other authors analyze risk propagation in a network of interdependent systems, but the model is often simple, with a binary propagation of risk or delay (Yan et al. 2012, Fang and Marle 2012), or a qualitative rather than quantitative description of the impact of delays (Gaonkar and Viswanadham 2007). These methods constitute a valid foundation upon which to assess the parameters of the dependencies in the SDDA model, which can then be used for more thorough analysis of development schedule, and associated risks.

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 supports SDDA users in deciding the adequate set of nodes required to model the system development, and constitute a valid tool to represent the matrices of parameters of the SDDA model and to identify clusters in SDDA networks. Adding to this process, our model can identify criticalities inside and among clusters, and suggest ways to shape the
dependencies and schedule the development of the required systems, in order to improve developmental architectures, and support decision policies to decrease cost and risk.

**Scheduling methodology and impact of delays**

Since 1959, industry heavily relied on PERT/CPM techniques for schedule, and analysis of time and cost of projects development. In 1959, Malcolm, Roseboom, Clark, and Fazar published the description of the PERT technique (Malcolm et al. 1959). The concepts of expected times, and latest time, are still used with little changes from the original formulation, that assumes absolute dependencies of events, and certain forms of probability density of the expected time, to treat the stochastic case. In the same year, Kelley and Walker added coordination of activities, revision, and analysis of critical path, in their CPM methodology (Kelley and Walker 1959).

Studies and modifications and studies have been added to PERT/CPM (Fulkerson 1962, Robillard and Trahan 1976). Cinicioglu and Shenoy proposed a way to solve stochastic PERT networks analytically, using Bayesian networks (2006), and Azaron and Tavakkoli-Moghaddam suggested a method for time and cost trade-off in dynamic PERT networks (2006).

A literature survey of project management, decision support, and scheduling shows that authors recognize the need for lead times accounting for possible partial dependencies (Sharma 2013, Lienz and Rea 2002), and for intelligent scheduling and rescheduling (Brown et al. 1995, Smith 1995, Zaveri and Emdad 1997). However, the common approach is to use static values for lead times (Ambriz 2008), and reactive rescheduling, based on the current behavior of the systems under development, in terms of delays, rather than in the expected behavior. Boehm (2014) surveys estimation methods for development schedule in Software engineering. The common approach in this case is to compress schedule by allocating an adequate number of workers to each activity. However, he also underlines the fundamental importance of project networks when there are known developmental dependencies, and when dealing with constituent systems in a SoS. The suggested approach in this case is the traditional PERT/CPM methodology (Wiest and Levy 1977, using expert data for estimation, augmented by on-demand scheduling and improved visibility of work-in-progress and system status (Turner 2013). These publications, while not accounting for partial dependency, or limited work breakdown structure due to complexity, underline the problems involved with uncertainty and large systems, and give methodologies to estimate development times.

Krishnan, Eppinger, and Whitney propose a framework to model overlapping in a basic scenario, without accounting for delays (Krishnan et al. 1997). The model is based on the required exchange of information to maximize parallel development. This framework does not deal with estimation of future behavior, and consequent dynamic programming, yet it constitutes a solid background to evaluate the features of developmental dependencies between systems.

Building upon the ideas, and previous research described in literature, the methodology described in this
paper addresses some of the needs in complex systems development scheduling, and risk management. SDDA constitutes a simple model, suitable to analyze and assess the impact of delays in the development, identify criticalities, deal with partial dependencies and their features, and dynamically perform intelligent scheduling and rescheduling, based on the features of the dependencies, and on the amount of acceptable risk input by the user.

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

- The beginning time of development of a system depends not only on its dependency from the development of another system, but also on the parameters that model this dependency.
- The possible lead time is a function of the parameters of the dependency, and can be automatically updated, based on the reliability of predecessors in terms of development according to the schedule.
- The completion time of a system is a function not only of the beginning time and development time, but also of the parameters of its dependencies.
- If a predecessor shows low reliability in development, SDDA suggests a lead time equal to 0, i.e. the development of successors must wait until the predecessor is fully developed. The parameters of the dependencies can be used to model the amount of risk that the manager is willing to accept, ranging from policies of high lead times (allowing for more delay absorption, but increasing the risk of long development times and waste of resources) to policies of low lead times (similar to PERT, allowing for delay absorption only outside the critical path, but with less risk of waste of resources).

**Systems Developmental Dependency Analysis**

Systems Developmental Dependency Analysis (SDDA) is a method to analyze the impact of partial developmental dependencies between systems on the schedule and development of the whole complex system or System-of-Systems (SoS). The method borrows the concepts of Strength of Dependency (SOD) and Criticality of Dependency (COD) from previous work by Garvey and Pinto (2009, 2012), who proposed Functional Dependency Network Analysis (FDNA), a two-parameters model of dependencies between capabilities. Likewise, in SDDA we model the developmental dependencies between systems with two parameters, having an intuitive meaning that facilitate the modeling process. The parameters are described below.

The outcome of SDDA 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. The lead time, i.e. the amount of time by which a system can begin to be developed before a predecessor is fully developed, is calculated based on the parameters of the dependencies, and the performance of the predecessors. SDDA allows for deterministic or stochastic
analysis. In deterministic analysis, SDDA evaluates the impact of a single instance (i.e., one given amount of delay in each system), resulting in one set of beginning time and completion time of each system. In stochastic analysis, the amount of delay in each system follows a given probability density function. Consequently, even the beginning and completion time of each system will be a probability density function. SDDA methodology evaluates the most critical nodes and dependencies with respect to development time and propagation of delays. Results from the analysis are used to compare different architectures in terms of development time, capability to absorb delays, and flexibility.

**Developmental dependencies**

The method is most useful for large systems and SoS development networks. These are directed networks where the nodes represent systems to be developed. The links, like in PERT networks, represent developmental dependencies between systems (Fig. 1). A dependency of a system \( j \) from a system \( i \) means that system \( j \) needs some input (information, or other deliverables) related to the development of system \( i \), to be able to complete its own development. Differently from PERT, however, the dependencies are not absolute and account for partial independency of development of each system. Usually, this partial independency is accounted for by assigning a fixed lead time, based on expert judgment. SDDA gives a simple, yet more realistic model of this kind of dependencies.

In SDDA, each system \( i \) requires three pieces of input data. The first two inputs are the minimum independent development time \( t_{\text{min}}^i \), and the maximum independent development time \( t_{\text{max}}^i \). These are respectively the minimum and maximum development time of system \( i \), not accounting for the dependencies. The third input is a reliability state, representing the system timeliness, or punctuality \( P_i \). The level of punctuality, normalized between 0 and 100, constitutes an assessment of the timeliness of the development. For example, 5 weeks could be the shortest time to develop a system, corresponding to punctuality of 100, 12 weeks could be the longest time to develop the same system, corresponding to punctuality of 0, but 10 weeks could correspond to a satisfaction of 50%, therefore a level of punctuality equal to 50. In this work, we used a simple linear correspondence between development time and punctuality (with high punctuality corresponding to short time, and vice versa. See Fig.2, line with crosses). In deterministic analysis, the punctuality of each system will be a single number, while in stochastic analysis the punctuality will follow a probability density function.

![Fig. 1. Synthetic SDDA network. N: node. SOD: strength of dependency. COD: criticality of dependency. P: punctuality. t: development time](image-url)
Each link, that is each dependency, requires two parameters: *Strength of Dependency* (SOD), and *Criticality of Dependency* (COD), that affect the development schedule of the whole system in different ways. These parameters, described in detail in the next section, can come from expert judgment and evaluation or can be computed based on historical data. The framework to overlap product development activities, proposed by Krishnan, Eppinger, and Whitney (1997), suggests a model based on the required exchange of information for parallel development. This approach gives important insights into the parameters of developmental dependencies, and can be used as a base for SDDA parameters evaluation.

**Parameters of the model**

Each developmental dependency between two systems is model with two parameters, SOD and COD. The low number of parameters, and their intuitive meaning, make them suitable to be assessed by knowledgeable designers and managers. At the same time, they overcome PERT/CPM’s inability to manage partial dependencies and dynamic lead time. Fig. 2 shows the relation between completion time of a predecessor system $i$ (function of its punctuality $P_i$) and beginning time of a successor system $j$. We describe the parameters used in the figure in the next sections.

![Fig. 2](image)

*Fig. 2. Completion time of system $i$, and beginning time of system $j$ in function of the parameters of the developmental dependency between the two systems. Due to partial dependency (SOD lower than 1), system $j$ can begin its development before completion of system $i$, unless the latter is critically late.*

**Strength of Dependency**

The Strength of Dependency (SOD), ranging between 0 and 1, evaluates the fraction of development time of a system that is dependent on inputs by its predecessor. As shown in Fig. 2, a system can begin its development earlier than the completion of the development of its predecessor. When the predecessor is developed in its shortest development time ($P_i$ equal to 100), the amount of lead time of the successor is equal to its own minimum development time, multiplied by a factor of 1 minus the SOD. This means that, while the predecessor completes its development, the successor will be able to complete the fraction of its development that does not depend on its predecessor. The value of the SOD parameter will trade-off between the risk associated with the decision to begin the development early and the possibility to partially absorb delays thanks to this lead time. A delayed development of the predecessor will affect the beginning time of the successor in two ways. First, the delay will directly be added to the expected beginning time of the successor. Second, the lead time computed by SDDA will decrease proportionally to the decrease in the predecessor punctuality (the development of the predecessor is considered to be less reliable), until the punctuality reaches a critical level, under which the lead time is equal to 0.
Criticality of Dependency

The Criticality of Dependency (COD), ranging between 0 and 100, is the normalized level of punctuality of a predecessor under which a successor cannot begin its development before the predecessor is fully developed. A partial dependency with a certain strength implies some risk, since the decision to have a lead time, that is to begin the development early, must be taken while the predecessor is still under development. Independently from the strength of the dependency, the criticality defines the amount of risk that the manager is willing to take. A high criticality means that even a small delay in the development of the predecessor will highly decrease the lead time, or wipe it out completely.

The model: basic formulation of Systems Developmental Dependency Analysis

The required input and parameters of SDDA have been described in the previous sections. The output of the model is the beginning time $t_B^i$ and the completion time $t_C^i$ of the development of each system $i$.

For a root node $i$, that is a node without any predecessor, the beginning time is 0

$$t_B^i = 0$$

and the completion time, depending on the punctuality, is

$$t_C^i = t_{\text{min}}^i + (1 - P_i/100)(t_{\text{max}}^i - t_{\text{min}}^i)$$

For a node $j$ having predecessors, we first compute the time necessary for its development, $t_D^j$, based on its punctuality

$$t_D^j = t_{\text{min}}^j + (1 - P_j/100)(t_{\text{max}}^j - t_{\text{min}}^j)$$

We then calculate the beginning and completion times based on each dependency from a system $i$. These are the actual beginning and completion times of system $j$, if it is depending only on one system. $S_{ij}$ is the SOD between system $i$ and system $j$, and $C_{ij}$ is the COD between system $i$ and system $j$.

If $P_i < C_{ij}$, system $i$ has critical delay, therefore the beginning time of $j$ based on its dependency on $i$, $i_t_B^j$, is equal to the completion time of $i$

$$i_t_B^j = t_C^i$$

otherwise, the beginning time of $j$ based on its dependency on $i$ is computed as

$$i_t_B^j = t_C^i - t_{\text{min}}^j (1 - S_{ij}) (P_i - C_{ij})/(100 - C_{ij})$$

In (5), the term that is subtracted from the completion time of $i$ is the lead time of $j$. It depends on the SOD between $i$ and $j$ and on the punctuality of $i$ (See Fig. 2, line with circles).

In this basic formulation of SDDA, the actual beginning time of a system $j$ that has more than one
dependency is the average of the beginning times resulting from each dependency. This prevents a single predecessor from critically influence the beginning time.

\[ t_B^j = \frac{1}{n} \sum_{k=1}^{n} k t_B^i \]  

(6)

The completion time of \( j \) based on its dependency on \( i \) is

\[ i t_C^j = \max(t_B^j + t_D^j, t_C^i + S_{ij}t_{\text{min}}^j) \]  

(7)

The first term in the brackets is the sum of beginning time and development time, therefore it is the completion time that \( j \) would have, not accounting for the dependencies. However, the strength of each dependency gives a measure of the fraction of development time of \( j \) that depends on the full development of \( i \). The second term in the bracket accounts for this dependency factor, stating that node \( j \) cannot be completed before a certain amount of time elapses after the completion of node \( i \). The actual completion time of system \( j \) is the maximum of the completion times given by each dependency.

\[ t_C^j = \max_{n}(it_C^j) \]  

(8)

Computation of the beginning and completion time for each node results in a complete schedule of the development of the complex system or SoS, showing the effect of partial developmental dependency on the development time.

The model: conservative formulation of Systems Developmental Dependency Analysis

In the basic model of SDDA, the actual beginning time of development of a system \( j \) is computed as the average of the beginning times resulting from each dependency that system \( j \) has. If the completion time of the predecessors of \( j \) is very different, this choice could result in an excessive amount of lead time, with consequent increase in cost. To avoid this effect, a more conservative formulation can be used. In this model, called SDDAmax, the beginning time is the maximum of the beginning times resulting from each dependency

\[ t_B^j = \max_{n}(n t_B^i) \]  

(9)

In SDDAmax, (9) is used instead of (6).

Deterministic analysis

In deterministic analysis, SDDA evaluates a single instance of the developmental dependencies. Given the parameters of the dependencies, and minimum and maximum development times, a single value of punctuality of each system is used to compute the resulting beginning and completion times of the development of each system.

This kind of analysis is useful if the user is interested in the impact of the timeliness of a specific system
on the overall development of the complex system. In this case, the user may assign values ranging between 0 and 100 to the punctuality of the system under consideration. The user may then assess the resulting impact on schedule through the analysis of beginning and completion times, listed in tables or plotted in graphic format. Another example may involve delays in several systems, to evaluate the combined effect of multiple postponements.

With deterministic analysis, the user can identify the most critical nodes under specified conditions, i.e. the nodes that most affect the development schedule. The user can compare different architectures, based on their response to delays. It must be noted that the results of this analysis also depend on the output of interest: for example, a manager might be interested in intermediate deadlines, completion time of development of intermediate systems, partial development of certain systems, delay absorption in case of low reliability. Stochastic analysis better characterizes the overall impact of delays on schedule, and on the risk associated with unreliable systems, in terms of punctuality.

Fig. 3 shows a Gantt chart for the simple network from Fig. 1, comparing results from SDDA, SDDAmax, and PERT when nodes 1 and 3 have punctuality equal to 100, and node 2 has punctuality equal to 70. The matrices of strength and criticalities of the dependencies are

\[
S = \begin{bmatrix}
0 & 0 & 0.6 \\
0 & 0 & 0.75 \\
0 & 0 & 0 \\
\end{bmatrix}
\quad C = \begin{bmatrix}
0 & 0 & 25 \\
0 & 0 & 40 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

The minimum and maximum development time are (in weeks)

\[
t_{\text{min}} = \begin{bmatrix}
10 \\
12 \\
10 \\
\end{bmatrix} \quad t_{\text{max}} = \begin{bmatrix}
16 \\
15 \\
14 \\
\end{bmatrix}
\]

From Fig. 3, we can notice that in PERT analysis Node 3 must wait until its two predecessor are fully developed. SDDA and SDDAmax exhibit a lead time for the development of Node 3. Due to the high strength and criticality of the dependency of Node 3 on Node 2, and since Node 2 is experiencing delays (P2=70), the lead time is small. SDDAmax is more conservative than the basic SDDA model for what concerns the risk of early development of Node 3.

Table I shows the results of various instances of delays in the simple 3-node network from Fig. 1. The user is in this case interested in the completion time of the whole complex system, that is the completion time of Node 3.

The results listed in Table I give some interesting insight:

- Under the assumption of partial dependencies, a schedule that follows the
SDDA model allows for partial or total delay recovery, and results in a completion time that is lower or equal to that given by a PERT model. This possible delay recovery must be traded against the cost of longer development times of individual systems, and the increased risk due to early decision.

- A schedule that follows the SDDAmax model is more conservative than SDDA, yielding less delay recovery, but also less risk of wasting resources due to early beginning of development. The development of Node 3 is shorter in SDDAmax than in SDDA model, but the overall development time is longer.

- Node 2 is the most critical if it experiences a short delay (e.g., when punctuality is equal to $[100 \ 75 \ 100]$, the delays are higher than other instances with single small decrease in punctuality). Node 1 is the most critical if it experiences a long delay (e.g., when punctuality is equal to $[25 \ 100 \ 100]$, the delays are higher than other instances with single large decrease in punctuality). Delays in Node 1 also result the most critical when coupled with delays in other nodes.

### Stochastic analysis

A more complete understanding of the impact of dependencies on development can be obtained by means of a stochastic analysis with SDDA. In stochastic analysis, we use a probability density function of the punctuality, rather than a single instance. The corresponding output is a probability density function for the beginning and completion times of each system, accounting for the parameters of the model, and the overall effect of topology. The expected value of the beginning and completion times can be used as a first guideline to decisions, while the variance of these times gives insight into the risk

<table>
<thead>
<tr>
<th>Punctuality of the systems</th>
<th>Model</th>
<th>$t_C^2$ (weeks)</th>
<th>Max delay in single systems</th>
<th>Overall delay in complex system</th>
</tr>
</thead>
<tbody>
<tr>
<td>[100 100 100]</td>
<td>SDDA</td>
<td>19.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>19.5</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>[75 100 100]</td>
<td>SDDA</td>
<td>19.5</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>19.5</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>22</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>[25 100 100]</td>
<td>SDDA</td>
<td>22</td>
<td>4.5</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>24.5</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>24.5</td>
<td>4.5</td>
<td>2.5</td>
</tr>
<tr>
<td>[100 75 100]</td>
<td>SDDA</td>
<td>20.25</td>
<td>1.6</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>21.29</td>
<td>0.75</td>
<td>1.79</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>22.75</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>[100 25 100]</td>
<td>SDDA</td>
<td>21.75</td>
<td>2.25</td>
<td>2.25</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>24.25</td>
<td>2.25</td>
<td>4.75</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>24.25</td>
<td>2.25</td>
<td>2.25</td>
</tr>
<tr>
<td>[100 100 75]</td>
<td>SDDA</td>
<td>19.5</td>
<td>1.75</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>20.5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>23</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>[100 100 25]</td>
<td>SDDA</td>
<td>20.75</td>
<td>3</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>22.5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>25</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>[50 50 100]</td>
<td>SDDA</td>
<td>22.38</td>
<td>3</td>
<td>2.88</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>23.08</td>
<td>3</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>23.5</td>
<td>3</td>
<td>1.5</td>
</tr>
<tr>
<td>[50 100 50]</td>
<td>SDDA</td>
<td>22.58</td>
<td>3</td>
<td>3.08</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>23.67</td>
<td>3</td>
<td>4.17</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>25</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>[100 50 50]</td>
<td>SDDA</td>
<td>21.54</td>
<td>2</td>
<td>2.04</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>25.08</td>
<td>2</td>
<td>5.58</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>25.5</td>
<td>2</td>
<td>3.5</td>
</tr>
<tr>
<td>[50 50 50]</td>
<td>SDDA</td>
<td>24.38</td>
<td>3</td>
<td>4.88</td>
</tr>
<tr>
<td></td>
<td>SDDAmax</td>
<td>25.08</td>
<td>3</td>
<td>5.58</td>
</tr>
<tr>
<td></td>
<td>PERT</td>
<td>25.5</td>
<td>3</td>
<td>3.5</td>
</tr>
</tbody>
</table>

---

a with respect to the baseline $t_C^2$ of the same model

b partial or total delay absorption
c effect of criticality: since lead time is decreased, the overall delay with respect to the baseline can be higher than the delay in a single system, but the overall completion time will be lower than or equal to the PERT baseline
associated with the development and delays. These outputs show patterns and features of the whole architecture. For example, given the expected distribution of punctuality, the user can compute the probability that development deadlines will be met. The user can then compare alternative architectures, identify their critical systems, and identify the topological patterns that cause the observed criticality.

Since SDDA is computationally inexpensive (Table II), Monte Carlo simulation appears to be the best choice to perform this type of analysis. We generate a large number of instances of punctuality, based on given distributions, and then we compute the beginning and completion times with SDDA.

In this paper, we used the following model of uncertainty:

- The user inputs an expected level of punctuality for each system, as in the deterministic analysis.
- The input level of punctuality will be the mode of a symmetric\(^1\) Beta probability density function (PDF), multiplied by a spreading factor. This input level might not be the mean and median of the PDF, because the tails might be cut, to respect the range of punctuality.
- The user inputs one of three level of uncertainty for each system. The lower the uncertainty, the more reliable the assumption of the punctuality. We model this higher confidence with a lower variance of the PDF.
- The user inputs the time instant, on the development schedule, at which SDDA has to compute the expected development performance. As this time instant gets closer to a system’s completion time, the uncertainty on the system punctuality will decrease (lower variance of the PDF), until the chosen time is equal or greater than the system’s completion time, at which point the uncertainty on the completion time is zero.
- At this point, even if the range of the Beta function is limited (and so is the resulting PDF with modified variance), it might be partially outside the allowed range of punctuality. Hence, we cut the PDF over the range 0 to 100, and normalize it so that its area is equal to 1.

Fig. 4 shows the results of development analysis via the stochastic SDDA model (10000 Monte Carlo trial). The punctuality was generated according to the PDF of the stochastic SDDA model, and we computed the consequent beginning and completion times of each system. The same stochastic model was also applied to PERT analysis. The matrices of strength and criticality of dependencies, minimum and maximum development times, and levels of punctuality, are the same as in the deterministic example of Fig. 3. Node

\(^{1}\) The initial probability is symmetric because we choose the two parameters of the Beta density function, \(\alpha\) and \(\beta\), to be equal.
1 has medium uncertainty level, Node 2 has high uncertainty level, and Node 3 has low uncertainty level.

At time 0, the beginning and completion times show the largest uncertainty. At time 8 weeks, the uncertainty has decreased. Node 2 has higher uncertainty than Node 1, according to the input by the user. At time 16 weeks, two systems are fully developed, and therefore exhibit no uncertainty. Node 3 has low uncertainty, according to the input by the user. A development schedule according to the SDDA basic model allows for early completion, but the graph of the expected development at time 8 shows that there is still risk associated with the uncertainty. The user can make informed decision thanks to this methodology, choosing an appropriate beginning time, based on result of this analysis, and on the amount of accepted risk. For example, the mean of the PDF of beginning time can be used as scheduled value.

The user can also calculate the probability that each system will be fully developed by given deadlines.

In this example, the expected levels of punctuality did not change. Of course, these levels, as well as the levels of uncertainty, may change over time. The user can repeat the analysis later, during the development of the complex system, when further decisions are required. SDDA results suggest possible times at which to perform the analysis again, with current information.

Table III lists some of the results of stochastic analysis with SDDA, SDDAmax, and PERT model, including the expected completion time, and the tenth, fiftieth, and ninetieth percentiles of the beginning time, representing more or less conservative choices. The results are computed based on the information available at time 0. The values of the parameters and inputs are the same as in the previous cases.

As expected, PERT exhibits the latest final expected completion time $E(t^2_c)$, because no development
overlapping is considered. On the other hand, the percentiles show a smaller spread in the values of beginning and completion times in PERT (Table III shows percentiles of the beginning development time of Node 3, on 10000 instances). The larger spread in the values of beginning and completion times in SDDA and SDDAmax occurs because the uncertainty in these values is not only due to the stochastic nature of the punctuality (as it happens in PERT), but also to the lead time assigned to the systems, based on the parameters of the models and on the expected punctuality.

Case Study: Littoral Combat Warfare

In this section, we show results of the application of SDDA for development schedule and risk analysis of a small Naval Warfare scenario SoS ([22]). This SoS is composed of three main systems: a Surface Warfare system, an Anti-Mine system, and an Anti-Submarine system. Each of these systems has ships, helicopters, unmanned vehicles, and other subsystems, listed in Table IV. In the same table, we also list the task performed by each subsystem, and a number that represents the subsystem in the different development networks. The objective of this case study is to illustrate how to perform deterministic and stochastic analysis, how to use SDDA results to obtain insights into the development time and risk of delays associated with various architectures and disruptions, and the value of SDDA approach for architecture design and selection, and for development policy support.

We analyzed three development architectures, representing different approaches, where

<table>
<thead>
<tr>
<th>Node number</th>
<th>System</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ship weaponry</td>
<td>Surface</td>
</tr>
<tr>
<td>2</td>
<td>Ship radar</td>
<td>Anti-submarine</td>
</tr>
<tr>
<td>3</td>
<td>Ship weaponry</td>
<td>Anti-submarine</td>
</tr>
<tr>
<td>4</td>
<td>Ship radar</td>
<td>Anti-mine</td>
</tr>
<tr>
<td>5</td>
<td>Ship weaponry</td>
<td>Anti-mine</td>
</tr>
<tr>
<td>6</td>
<td>Helicopter radar</td>
<td>Surface</td>
</tr>
<tr>
<td>7</td>
<td>Unmanned Air Vehicle radar</td>
<td>Surface</td>
</tr>
<tr>
<td>8</td>
<td>Helicopter weaponry</td>
<td>Anti-submarine</td>
</tr>
<tr>
<td>9</td>
<td>Remotely Controlled Vehicle radar</td>
<td>Anti-submarine</td>
</tr>
<tr>
<td>10</td>
<td>Unmanned Water Vehicle radar</td>
<td>Anti-submarine</td>
</tr>
<tr>
<td>11</td>
<td>Helicopter weaponry</td>
<td>Anti-mine</td>
</tr>
<tr>
<td>12</td>
<td>Remotely Controlled Vehicle radar</td>
<td>Anti-mine</td>
</tr>
<tr>
<td>13</td>
<td>Unmanned Water Vehicle radar</td>
<td>Anti-mine</td>
</tr>
</tbody>
</table>
various stakeholders participate into the SoS at different times. Development choices, and parameters of dependency are based on availability of resources, and efficacy of deployed systems during the development (architectures where only radars or only weapons are developed first are unwanted).

In architecture A, the ships are developed first, followed by development of the surface systems, then the anti-submarine system, and finally the anti-mine system.

In architecture B, the surface and anti-mine systems are developed first, followed by the anti-submarine system.

In architecture C, the surface system is developed independently from the others. Anti-mine and anti-submarine systems begin their development in parallel, but the completion of the anti-mine system depends on the completion of the anti-submarine system.

The networks representing these development choices, and their parameters, are shown in appendix.

**Deterministic analysis**

We first performed deterministic analysis, to compute the best and worst time of development, and to identify the most critical systems and dependencies in the development network. Fig. 5 shows the basic schedule, with SDDA, SDDA\textsubscript{max}, and PERT models, for architecture A. Fig. 6 shows the basic schedule with SDDA\textsubscript{max} model, for all three architecture. The schedule according to the three models, without any delay, exhibits the same aspect already underlined when describing the
models: SDDA and SDDA_{max}, exploiting partial dependencies, result in a schedule that allows for early completion of the development of the SoS. SDDA, less conservative than SDDA_{max}, shows a longer development time for many of the component systems. This approach, that uses more resources and entails more risks, may however allow for partial recovery, when delays occur.

The schedule according to SDDA_{max} for all three architectures shows how different component systems complete their development at different times. Therefore, different capabilities are achieved at different times. These considerations, together with criticalities, risk, delay absorption, and resources availability, give support to the user decisions.

**Best time of development**

We use deterministic analysis, with punctuality of the systems at the maximum level, to compute the best time of development. We also compute the completion time of each system, useful to evaluate possible partial capabilities achieved, and the development time of each system, useful to evaluate the use of resources. Results are shown in table V.

As expected, since delays do not occur, both SDDA and SDDA_{max} exploit the partial dependencies at the highest possible level, and therefore achieve the same completion time. However, since SDDA is less conservative, and involves longest lead times, the development times of each system (and, consequently, the use of resources and the cost) are larger. This feature will allow for partial or total delay absorption, in case the systems will not exhibit maximum punctuality.

<table>
<thead>
<tr>
<th>Architecture and overall time</th>
<th>SDDA</th>
<th>SDDA_{max}</th>
<th>PERT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDDA: 106</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 106</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>PERT: 150</td>
<td>62.8</td>
<td>34.2</td>
<td>62.8</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDDA: 107.5</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 107.5</td>
<td>107.5</td>
<td>107.5</td>
<td>107.5</td>
</tr>
<tr>
<td>PERT: 185</td>
<td>63.6</td>
<td>33.7</td>
<td>63.6</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDDA: 108</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 108</td>
<td>78</td>
<td>60.7</td>
<td>78</td>
</tr>
<tr>
<td>PERT: 183</td>
<td>97.2</td>
<td>43.8</td>
<td>97.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecture and overall time</th>
<th>SDDA</th>
<th>SDDA_{max}</th>
<th>PERT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 106</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 106</td>
<td>62.8</td>
<td>34.2</td>
<td>62.8</td>
</tr>
<tr>
<td>PERT: 150</td>
<td>106</td>
<td>64</td>
<td>106</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 107.5</td>
<td>107.5</td>
<td>107.5</td>
<td>107.5</td>
</tr>
<tr>
<td>SDDA_{max}: 107.5</td>
<td>95.8</td>
<td>43.8</td>
<td>95.8</td>
</tr>
<tr>
<td>PERT: 185</td>
<td>79.2</td>
<td>43.8</td>
<td>79.2</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 108</td>
<td>78</td>
<td>60.7</td>
<td>78</td>
</tr>
<tr>
<td>SDDA_{max}: 108</td>
<td>91.2</td>
<td>22</td>
<td>91.2</td>
</tr>
<tr>
<td>PERT: 183</td>
<td>103.3</td>
<td>37.5</td>
<td>103.3</td>
</tr>
<tr>
<td><strong>Architecture and overall time</strong></td>
<td>SDDA</td>
<td>SDDA_{max}</td>
<td>PERT</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 106</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 106</td>
<td>62.8</td>
<td>34.2</td>
<td>62.8</td>
</tr>
<tr>
<td>PERT: 150</td>
<td>106</td>
<td>64</td>
<td>106</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 107.5</td>
<td>107.5</td>
<td>107.5</td>
<td>107.5</td>
</tr>
<tr>
<td>SDDA_{max}: 107.5</td>
<td>95.8</td>
<td>43.8</td>
<td>95.8</td>
</tr>
<tr>
<td>PERT: 185</td>
<td>79.2</td>
<td>43.8</td>
<td>79.2</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 108</td>
<td>78</td>
<td>60.7</td>
<td>78</td>
</tr>
<tr>
<td>SDDA_{max}: 108</td>
<td>91.2</td>
<td>22</td>
<td>91.2</td>
</tr>
<tr>
<td>PERT: 183</td>
<td>103.3</td>
<td>37.5</td>
<td>103.3</td>
</tr>
<tr>
<td><strong>Architecture and overall time</strong></td>
<td>SDDA</td>
<td>SDDA_{max}</td>
<td>PERT</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 106</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA_{max}: 106</td>
<td>62.8</td>
<td>34.2</td>
<td>62.8</td>
</tr>
<tr>
<td>PERT: 150</td>
<td>106</td>
<td>64</td>
<td>106</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 107.5</td>
<td>107.5</td>
<td>107.5</td>
<td>107.5</td>
</tr>
<tr>
<td>SDDA_{max}: 107.5</td>
<td>95.8</td>
<td>43.8</td>
<td>95.8</td>
</tr>
<tr>
<td>PERT: 185</td>
<td>79.2</td>
<td>43.8</td>
<td>79.2</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>SDDA: 108</td>
<td>78</td>
<td>60.7</td>
<td>78</td>
</tr>
<tr>
<td>SDDA_{max}: 108</td>
<td>91.2</td>
<td>22</td>
<td>91.2</td>
</tr>
<tr>
<td>PERT: 183</td>
<td>103.3</td>
<td>37.5</td>
<td>103.3</td>
</tr>
<tr>
<td><strong>Architecture and overall time</strong></td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Delays and criticalities

In this case, we use instances of deterministic analysis, with decreased punctuality, to obtain insights into the capability of recovering delays through exploitation of partial dependency and lead times. We also identify the most critical nodes in terms of impact on development delays. The indicated delayed node has punctuality of 50, while the others have punctuality of 100. Delay recovery is computed with respect to the initial system delay. Results are shown in table VI.

Architecture A allows for more delay recovery, especially with the SDDA model. As expected, the SDDA model, which has the longest lead times, reacts better to delays. This comes at the cost of longer development times of each system. When delays occur in critical systems, the SDDAmax model results in a completion time similar to that of PERT. Table VII shows the most critical systems for each architecture, and each model.

Example of stochastic analysis and decision support

In this section, we show an example of the use of stochastic SDDA model for analysis and decision support. Based on the results of the deterministic analysis, we focus on architecture A. First of all, we perform analysis according to the model described in section III.F. The uncertainty is low for nodes 1 to 5, medium for nodes 6 to 10, and high for systems 11 to 13. The baseline punctuality is 80 for each system.

Fig. 7 shows the GANTT charts resulting...
from stochastic analysis of this baseline case. The parameters of the dependencies, and the shortest and longest development time for architecture A are shown in appendix. The figure shows the larger uncertainty, with information at time 0, in the SDDAmax model. However, this model suggests higher beginning times than SDDA, therefore it is less prone to risk in terms of excessive cost, and use of resources due to early beginning of development, followed by unexpected delays.

Table VIII demonstrates a possible use of SDDA stochastic analysis to support management and programming. Using architecture A, we make the following assumptions:

- The initial estimate of punctuality is equal to 80 for each system. With this estimate, and the level of uncertainty described at the beginning of this section, we generate PDFs of the beginning and completion times of each system.
- The initial decision for the scheduling of the beginning times can be the expected value of the beginning times resulting from the initial estimate, or the 10th percentile (meaning a less risky choice of late start), or the 90th percentile (meaning a more risky choice of early start).
- The manager can also decide if the programming policies will keep the initial beginning times, even when more information about the actual punctuality of the systems under development becomes available, or if the schedule will be reviewed after 30 weeks.
- The actual punctuality of the system can be 80 (as estimated), 70, or 90. If the schedule is reviewed, information about the actual punctuality will be available.
- The minimum sum of the development time of each system is what we would have in a PERT model. SDDA model accounts for partial dependencies, using a lead time that might cause longer development times. The actual sum of the development time of each system is used as an indicator of use of

Fig. 7. Gantt chart showing the schedule of development of the Naval Warfare SoS, according to the SDDA, SDDAmax, and PERT stochastic models, for architecture A. The baseline punctuality is 80 for each system, and the uncertainties are computed with information at time 0. The resulting PDF of beginning and completion times is shown above the corresponding bar of the Gantt chart. The darker shadows on the bars indicate the zones of higher probability.
Based on each combination of programming policies and actual punctuality of the systems, we compute the expected completion time resulting from the initial estimate, the expected completion time of an SDDA model having the correct information about punctuality, and the actual expected completion time resulting from the policies. This analysis has been performed over ten thousand instances. We then compare the actual completion time in each case to the completion time resulting from having full information, and determine the percentage of cases in which a policy results in a longer completion time, or in a shorter

<table>
<thead>
<tr>
<th>Manager decision for $t_b^i$</th>
<th>Actual $P_i$</th>
<th>Schedule review at $t = 30$ weeks$^b$</th>
<th>A priori$^i$ $E(t_{i3}^i)$</th>
<th>$E(t_{i3}^i)$ correct info$^d$</th>
<th>Actual$^h$ $E(t_{i3}^i)$</th>
<th>% actual instances later than correct info$^e$</th>
<th>Avg. delay$^g$</th>
<th>% actual instances earlier than correct info$^e$</th>
<th>Avg. gain$^i$</th>
<th>$\sum_{i=1}^{13} t_b^i$</th>
<th>$\sum_{i=1}^{13} E(t_b^i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>No</td>
<td>118.9</td>
<td>119.6</td>
<td>119.5</td>
<td>46.6</td>
<td>1.351</td>
<td>53.4</td>
<td>1.391</td>
<td>409.6</td>
<td>481.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>118.9</td>
<td>119.6</td>
<td>119.6</td>
<td>44.5</td>
<td>1.063</td>
<td>55.5</td>
<td>1.198</td>
<td>409.6</td>
<td>481.8</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>No</td>
<td>118.9</td>
<td>131.8</td>
<td>121.5</td>
<td>0</td>
<td>-</td>
<td>100</td>
<td>10.304</td>
<td>428.4</td>
<td>505.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>118.9</td>
<td>131.8</td>
<td>125.4</td>
<td>0</td>
<td>-</td>
<td>100</td>
<td>6.173</td>
<td>428.4</td>
<td>489.1</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>No</td>
<td>118.9</td>
<td>112.7</td>
<td>117.6</td>
<td>100</td>
<td>4.942</td>
<td>0</td>
<td>-</td>
<td>390.8</td>
<td>459.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>118.9</td>
<td>112.7</td>
<td>117.2</td>
<td>100</td>
<td>4.454</td>
<td>0</td>
<td>-</td>
<td>390.8</td>
<td>479.6</td>
<td></td>
</tr>
<tr>
<td>10th percentile (late start)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>No</td>
<td>126.2</td>
<td>119.6</td>
<td>124.6</td>
<td>98.8</td>
<td>5.003</td>
<td>1.2</td>
<td>0.728</td>
<td>409.6</td>
<td>473.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>126.2</td>
<td>119.6</td>
<td>121.8</td>
<td>92.7</td>
<td>2.290</td>
<td>7.3</td>
<td>0.607</td>
<td>409.6</td>
<td>484.2</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>No</td>
<td>126.2</td>
<td>131.8</td>
<td>126.6</td>
<td>1.6</td>
<td>0.790</td>
<td>98.4</td>
<td>5.291</td>
<td>428.4</td>
<td>496.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>126.2</td>
<td>131.8</td>
<td>128.0</td>
<td>0.7</td>
<td>0.463</td>
<td>99.3</td>
<td>3.609</td>
<td>428.4</td>
<td>492.2</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>No</td>
<td>126.2</td>
<td>112.7</td>
<td>122.7</td>
<td>100</td>
<td>10.111</td>
<td>0</td>
<td>-</td>
<td>390.8</td>
<td>451.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>126.2</td>
<td>112.7</td>
<td>119.1</td>
<td>100</td>
<td>6.372</td>
<td>0</td>
<td>-</td>
<td>390.8</td>
<td>481.7</td>
<td></td>
</tr>
<tr>
<td>90th percentile (early start)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>No</td>
<td>115.3</td>
<td>119.6</td>
<td>117.0</td>
<td>4.7</td>
<td>0.534</td>
<td>95.3</td>
<td>2.769</td>
<td>409.6</td>
<td>493.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>115.3</td>
<td>119.6</td>
<td>117.5</td>
<td>5.3</td>
<td>0.562</td>
<td>94.7</td>
<td>2.512</td>
<td>409.6</td>
<td>479.3</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>No</td>
<td>115.3</td>
<td>131.8</td>
<td>119.1</td>
<td>0</td>
<td>-</td>
<td>100</td>
<td>12.658</td>
<td>428.4</td>
<td>517.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>115.3</td>
<td>131.8</td>
<td>122.8</td>
<td>0</td>
<td>-</td>
<td>100</td>
<td>8.830</td>
<td>428.4</td>
<td>485.9</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>No</td>
<td>115.3</td>
<td>112.7</td>
<td>115.2</td>
<td>96.3</td>
<td>2.683</td>
<td>3.7</td>
<td>0.521</td>
<td>390.8</td>
<td>471.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>115.3</td>
<td>112.7</td>
<td>115.3</td>
<td>95.6</td>
<td>2.642</td>
<td>4.4</td>
<td>0.489</td>
<td>390.8</td>
<td>477.4</td>
<td></td>
</tr>
</tbody>
</table>

$^a$ Based on the resulting PDFs of SDDA analysis, with initial estimate of $P_i = 80$, we assume that the manager can decide a beginning time for development of each system that is the mean of the PDF, or the 10th percentile, or the 90th percentile.

$^b$ The manager can decide if to review the development schedule, using updated information on the systems punctuality.

$^c$ The expected completion time of the whole system, given the initial estimate.

$^d$ Expected completion time, with information on the correct $P_i$.

$^e$ Expected completion time, given management decisions.

$^f$ Percentage of instances of development schedule using initial estimate and management decisions that end later than the corresponding schedule obtained with the correct $P_i$.

$^g$ Average delay of the actual schedule with respect to the schedule obtained with correct information.

$^h$ Percentage of instances of development schedule using initial estimate and management decisions that end earlier than the corresponding schedule obtained with the correct $P_i$.

$^i$ Average time gain of the actual schedule with respect to the schedule obtained with correct information.

$^j$ Minimum development time: sum of the development time of each system, given $P_i$ (without overlapping).

$^k$ Actual development time: sum of the actual development time of each system.
completion time. We also compute the average delay in the cases where the actual completion time is higher, and the average gain in the cases where the actual completion time is shorter. Finally, we compute the minimum and actual sum of development times.

The results yield many interesting points to support management decision:

- The decision of using the expected value for the beginning times results in completion time close to that resulting from the model with correct information.
- The decision of late start causes delays with respect to the model with correct information. However, late start without schedule review also has the shortest sum of development times. Therefore, this choice can be appropriate when review of the schedule is not possible, and the most important objective is low cost and use of resources.
- The decision of early start yield the shortest completion time and the smallest delays. However, this comes at cost of longer development times.
- Reviewing the schedule when the actual punctuality is higher than the initial estimate reduces the final delay only slightly (except in the case of late start policy), but at cost of much longer sum of development times. This policy is suggested if completion time is more important than use of resources.
- Reviewing the schedule when the actual punctuality is smaller than the initial estimate reduces the final gain, but also the sum of development times. This policy is suggested if use of resources is more important than completion time.
- Reviewing the schedule when the actual punctuality follow the initial estimate brings both gains and delays closer to the values resulting from the model with correct information. This policy reduces the sum of development times (but also increases the percentage of late instances) if the initial decision was an early start. It increases the sum of development times (but also decreases the percentage of late instances) if the initial decision was a late start.

We repeated the same analysis using the SDDAmax model. Results are listed in table IX. As expected, in general SDDAmax model results in longer completion times than SDDA, but with lower sum of development times. The same trade-off occurs among different policies based on the SDDAmax model: delays can be reduced at the cost of longer sum of development times.

In the case of SDDAmax model, schedule review is always suggested when the actual punctuality is better than the initial estimate, because the delays can be highly reduced at the cost of a small increase in the sum of development times. On the other side, reviewing the schedule when the actual punctuality is smaller than the initial estimate is almost never a good policy within the SDDAmax model, because only a little reduction in the percentage of late instances is possible, while the loss in gain may be substantial. Exception to this outcome is the policy of late start. In this case, reviewing the schedule when the actual
punctuality is smaller than the initial estimate will increase the probability of having a gain in the completion time.

**Conclusion, Contributions and Future Work**

Building upon the concepts of parametric model of dependencies, proposed by Garvey and Pinto (2009, 2012), and by Guariniello and DeLaurentis (2015) in the operational domain, we developed a method for analysis of the impact of developmental dependencies between elements in a system, or systems in a SoS. SDDA methodology allows user to model developmental interactions between systems, accounting for...
partial dependencies. This results in a simple model of complex and large systems, at a relatively high level of abstraction, based on a small number of input parameters. SDDA can thus be used to quickly analyze the impact of delays along a development network, identify criticalities, compare different development architectures, support decisions, and suggest policies to trade-off between delay absorption, time of completion, cost, and use of resources.

Compared to traditional approaches for development and scheduling, SDDA representation provides multiple advantages and improvements:

- SDDA has a very low computational cost, therefore it is useful to quickly analyze a large number of instances of an architecture, and to generate policy guidelines for a complex system with multiple dependencies.
- SDDA parameters have an intuitive meaning, and they may be easily related to the causes of observed results, when analyzing the impact of delays. Therefore, with SDDA model, the user may not only identify criticalities of the system, but also possible ways to improve the developmental architecture.
- SDDA parameters may come from a variety of sources, including expert evaluation, and historical data. One possible model to identify the parameters is based on the information that a process of system development needs to receive from another process, in order to start.
- Deterministic analysis with SDDA expands the PERT model, to include partial dependencies, and dynamic scheduling and rescheduling, accounting for the current punctuality with which the systems are being developed. The lead time and the partial overlapping of systems development are based on the model of their dependency, and on their punctuality.
- Stochastic analysis supports decision making in the developmental setting. Based on some initial estimate, including expected values and levels of uncertainty on the punctuality of each system, the user can analyze the effect of various policies, more or less conservative. Multiple cases can be analyzed, with the actual punctuality more or less different from the initial estimate, to assess the expected outcome given by each policy, in terms of times of completion, delay absorption, cost, and use of resources.

A small Naval Warfare SoS has been presented, to demonstrate the application of SDDA, both in the deterministic and in the stochastic version. Future improvements include the development of detailed metrics for desired features in specific applications (for example, weighted development time and cost), and integration of SDDA with other management and systems engineering methodology and tools. Steps in this direction have been performed, using SDDA in combination with operational analysis, to add the concept of partial capabilities, achieved during development according to a given architecture, to the trade-off.
Appendix

For all three architectures, the minimum and maximum development times for the systems are (in weeks):

\[
\begin{bmatrix}
30 \\
25 \\
30 \\
25 \\
30 \\
40 \\
20 \\
40 \\
22 \\
24 \\
40 \\
22 \\
24 \\
\end{bmatrix}
\quad \quad \quad \quad \quad \quad \quad \quad \quad
\begin{bmatrix}
48 \\
40 \\
48 \\
40 \\
48 \\
60 \\
32 \\
60 \\
32 \\
30 \\
60 \\
32 \\
30 \\
\end{bmatrix}
\]

\[
t_{\text{min}} =
\begin{bmatrix}
30 \\
25 \\
30 \\
25 \\
30 \\
40 \\
20 \\
40 \\
22 \\
24 \\
40 \\
22 \\
24 \\
\end{bmatrix}
\quad \quad \quad \quad \quad \quad \quad \quad \quad
\begin{bmatrix}
48 \\
40 \\
48 \\
40 \\
48 \\
60 \\
32 \\
60 \\
32 \\
30 \\
60 \\
32 \\
30 \\
\end{bmatrix}
\]

The matrices of the parameters modeling the developmental dependencies in architecture A are:

\[
S^A =
\begin{bmatrix}
0 & 0 & 0 & 0 & 0.6 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.15 & 0 & 0 & 0 & 0.5 & 0.7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.15 & 0 & 0 & 0 & 0.75 & 0.65 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.4 & 0.4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.55 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
C^A =
\begin{bmatrix}
0 & 0 & 0 & 0 & 40 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 60 & 0 & 0 & 0 & 0 & 0 & 35 & 45 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 60 & 0 & 0 & 0 & 0 & 0 & 30 & 40 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 65 & 30 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 35 & 35 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
The corresponding developmental network is shown in Fig. 8.

The matrices of the parameters modeling the developmental dependencies in architecture B are:

\[
S^B = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0.5 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.75 & 0.65 & 0 & 0 \\
0 & 0 & 0 & 0.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.5 & 0.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.55 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.55 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
The corresponding developmental network is shown in Fig. 9.

The matrices of the parameters modeling the developmental dependencies in architecture C are:

\[ C^B = \begin{bmatrix}
0 & 0 & 0 & 0 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 60 & 0 & 0 & 0 & 0 & 35 & 45 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 60 & 0 & 0 & 0 & 0 & 0 & 0 & 30 & 40 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 40 & 65 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 30 & 35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 40 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 35 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]


The matrices of the parameters modeling the developmental dependencies in architecture C are:
The corresponding developmental network is shown in Fig. 10.

References


