TCU-AM-17-040

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Modeling Uncertainty and Its Implication in Complex Interdependent Networks

12 April 2017

Dr. Anita Raja

Albert Nerken School of Engineering
The Cooper Union

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. N00244-15-1-0006. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Abstract

The overall goal of this paper is to continue our efforts to forge new ground in identifying the effects of interdependencies in large complex networked applications and, if needed, uncovering early indicators of interdependency risk so that appropriate risk mitigation actions may be taken. Specifically, we seek to study and quantify the impact of network characteristics on cascading risk. Cascading risk is defined as the propagation of programmatic issues across networked programs due to the interdependency of one program upon the other. Harnessing the extensive data that has been collected over the years in the form of Defense Acquisition Execution Summary (DAES), Selected Acquisition Reports (SARs) and Contract data for Major Defense Acquisition Programs (MDAPS), we will present our intermediate results in our ongoing efforts on leveraging network structure and sequential data to study cascading risks. We will also identify the challenges to data acquisition.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgement

I am grateful to Professor Mohammad Hasan from UNL and Cooper Union students, Brendan Fernes, Hetian Wu, Sahil Patel, Kevin Yao and Eui Seong Han for their diligent work that ensured the progress of this project. Professor Hasan worked on the Probabilistic Risk Analysis study and network analysis. Brendan and Hetian for their work on methods to scale up and generalize data extraction, Sahil and Kevin for their work on the PCA+MC what-if analysis; Eui Seong Han for his assistance with the data organization and analysis.

I am thankful to Mr. Robert Flowe for his valuable insight on the MDAP domain and suggestions to consider contract data. Professor Stan Mintchev was instrumental in discussions on building a formal model of MDAPs to determine the dynamics of the networks and its effect on performance. I also thank Professor Maureen Brown for her insights into the MDAP data.

THIS PAGE INTENTIONALLY LEFT BLANK

About the Author

Anita Raja is Associate Dean of Research and Graduate Programs and Professor of Computer Science in the Albert Nerken School of Engineering at The Cooper Union. She directs the Distributed Intelligent Agents Lab. She received her Ph.D. in computer science from the University of Massachusetts at Amherst in 1998 and 2003, respectively. Raja's research focus is in the field of artificial intelligence, specifically as it relates to the study of decentralized control and reasoning in software agent systems operating in the context of uncertainty and limited computational resources.

THIS PAGE INTENTIONALLY LEFT BLANK

TCU-AM-17-040

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Modeling Uncertainty and Its Implication in Complex Interdependent Networks

12 April 2017

Dr. Anita Raja

Albert Nerken School of Engineering
The Cooper Union

Disclaimer: This material is based upon work supported by the Naval Postgraduate School Acquisition Research Program under Grant No. N00244-15-1-0006. The views expressed in written materials or publications, and/or made by speakers, moderators, and presenters, do not necessarily reflect the official policies of the Naval Postgraduate School nor does mention of trade names, commercial practices, or organizations imply endorsement by the U.S. Government.

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

The Joint Space of Major Defense Acquisition Programs Networks

Our work is motivated by the need for “what-if” analysis in large complex interdependent and networked applications such as the critical infrastructure network (electric, water, gas grids). The research goal is to develop methodologies and algorithms to proactively model and reason about non-linear cascading risks to facilitate this analysis. Networked applications often operate under uncertainty in environmental response and the temporal state and action choices of the nodes are captured in the form of structured and unstructured text data as well as image data.

We build on our previous work (Raja et al., 2012, Raja et al., 2013, Raja et al., 2014), where we used state-of the-art extraction technologies including Latent Dirichlet Allocation (LDA) topic modeling algorithms to develop automated text and image extraction techniques to extract features from various types of structured and unstructured text and image data. In addition to this automated data extraction module, we developed two executable modules: one to identify the relationships in a network (Network Identifier module) and the other to compute the weight of the links among the neighboring nodes (Interdependency Index Determiner). We tested and evaluated these algorithms on a small network and showed that the performance of the automatic extraction algorithms was comparable to the performance of manual extraction.

We use the MDAP network as a case study to study cascading risks and develop methodologies and algorithms that can be generalizable to similar networks. Individual MDAP performance across months and years has been captured by a combination of structured and unstructured temporal data including Selected Acquisition Reports (SARs), Defense Acquisition Execution Summary (DAES) reports and milestone reviews are evaluated from an individual program point of view without emphasizing the dynamics of joint space. The question of modeling cascading risk across programs with funding or data relationships is important since we conjecture that poor performance of the MDAPs (various breach conditions) can

be attributed to local (individual MDAP) as well as non-local (related MDAPs) sources that result due to interdependencies among the MDAPs.

In this paper, we present a network-centric approach that has the dual goal of contributing to advances in reasoning about uncertainty, large-scale text and image data analysis as well understanding of complex networks. This project breaks ground in the areas of a) defining a metric to quantify the influence of network characteristics on performance; b) identifying the type of data required to formulate appropriate mathematical models for understanding the dynamics of complex networks; and c) using analytic tools to determine cascading risk in networks.

Summary of Findings

In this paper, we view risk of MDAP failure from several different lenses. Our findings indicate that considering network effect of a program's performance in addition to individual performance metrics along with such PAUC increase and APB breaches provides a more accurate view of future risk of program failure. We have developed an extended Probability Risk Analysis model to capture this wholistic view of risk and shown its effectiveness using a MDAP subnetwork as a case study. We also conducted a feasibility study for modeling interdependent networks as a coupled dynamical system and potentially adapting the algorithms for feed-forward networks. A coupled dynamical system has properties that would facilitate what-if analysis. We found that while DAES data was not conducive for such an analysis, it was more useful to mine contract data to assist with this type of cascading risk analysis. We then used applied methodology often used for financial instruments that combines Principal Component Analysis and MonteCarlo simulations to contract data to generate and provide insight into risk scenarios.

Network Performance Study of an Interdependent Hierarchical Network

The joint space of major defense acquisition programs (MDAPs) creates interdependencies among MDAPs. These interdependencies contain the characteristics of a complex network (Brown, 2014). Programs in the MDAP network share diverse relationships. Mainly, there are two types of ties that exist among the MDAPs: (1) programmatic ties (also called programmatic interdependencies) are defined by the program managers in terms of inbound and outbound connections to support hardware/software requirement of the programs, and (2) funding ties that identifies the programs as funding neighbors if they draw funding support from the same "program element" (PE) account. These two types of ties result in two types of network relationships among the MDAPs, namely, programmatic network and funding network.

A systemic understanding of the performance of the MDAPs requires the understanding of these two types of networks. Therefore, the system of MDAPs can be considered as a multiplex network that is a superposition of both programmatic and funding network defined on the same set of programs (Szell, 2010).

Interdependencies among the program influence the performance of the MDAPs (Brown, 2014; Raja et al., 2012). However, the multiplex nature of MDAP networks has not been considered to examine the performance of the programs. Moreover, the effect of interdependency on the programs was not quantified previously. Our goal is to investigate the joint space of the MDAP multiplex network as it influences program performance and to define a metric (the risk parameter) that quantifies this influence. The values of this risk parameter for each program in the multiplex network would be useful to forecast potential cascading effect. Moreover, the program managers would be able to identify critical programs using this parameter and to take necessary measures to improve programs' performance. The risk parameter is formally defined in the following based on the Probabilistic Risk Analysis (PRA) methodology for networked systems.

Probabilistic Risk Analysis (PRA)

Probabilistic risk analysis (PRA) is a methodology (Lewis, 2009) to evaluate risks associated with a complex engineering entity. It systematically looks at how the pieces of a system work together to ensure safety. PRA allows analysts to quantify risk and identify what could have the most impact on safety (Lewis, 2009). Therefore, we use the risk parameter from PRA methodology to quantify the influence of interdependency in a complex network, specifically the MDAP network.

The PRA equations for risk in a system use the notion of vulnerability and consequence. Although the concept of vulnerability, risk, and consequence in non-network systems share standard definitions in financial and engineering communities, these terms are not well understood for networked systems. This is because network science is a new field and it is not very clear how to understand the failure of the assets in networks.

According to the standard definitions (Lewis, 2009) in non-networked systems, vulnerability V is the probability that a component or asset will be compromised after successful attacks. Risk R measures the expected loss due to the failure of an asset. Threat T is the probability that an attack will be attempted. Consequence C is the outcome of a successful attack. Therefore, standard risk is defined as the product: $R = TV\mathrm{C}$

These definitions, however, do not provide an appropriate measure for risk in networked systems. In a network, system failure is a function of the interdependence of the nodes. These definitions do not incorporate the interdependency of the various components of a system. Therefore, it is important to consider the connectivity among the nodes in a network for computing risk.

Lewis (Lewis, 2009) extended these standard definitions to networks containing many components or assets (nodes and links). Threat (t), vulnerability (v), consequence (c), and risk (R) in a networked system are an aggregation of individual component or asset threat, vulnerability, and consequences. Network risk is defined in the following PRA equation as an expected value by taking the sum

over all nodes (n) and links (m) of the individual components: $\boldsymbol{R} = \sum_{i=1}^{n+m} \boldsymbol{t_i}\, \boldsymbol{v_i c_i} = \sum_{i=1}^{n+m} \boldsymbol{v_i}\, \boldsymbol{c_i}$ assuming $t_i = 1$. Here, threat and vulnerability are a priori estimates of the probability of failure. Consequence is typically measured in dollars or lives. This PRA equation for risk is applicable for any system where a priori approximations of the probability of failure can be reasonably estimated. For reducing risk in a networked system, Lewis (Lewis, 2009) argues that it is important to identify the critical nodes that have higher risk values.

Earlier works by Albert, Jeong and Barabasi (Albert, Jeong & Barabasi, 2000) and others explored why highly-connected nodes were more critical nodes than others. However, these studies were done in the context of single-plex network. Al-Mannai and Lewis (Al-Mannai & Lewis, 2007) proposed a static technique for critical node analysis in a **multiplex network** where criticality of a node not only depends on the number of connections but also on other measures. They use a degree-weighted model of network risk to identify the most critical nodes in a network. Intuitively, critical nodes either have many connections or have larger target values. Based on this observation, Al-Mannai and Lewis extended the simple PRA definition of risk to define the target value of a node as g_iC_i , where g_i is the degree of the node and C_i is the consequence associated with the node's intrinsic value. Therefore, according to their model, extended risk ***r*** for an n-node network is related to network topology as follows, $\boldsymbol{r_{ext}} = \sum_{i=1}^{n} g_i V_i C_i$ where g is the degree of node *i*, while V and C are its vulnerability and consequence, respectively.

Example: PRA for a Small Synthetic Network

As an illustration of the above-mentioned extended PRA technique, let's consider the following network (Figure 1) of four nodes (A, B, C, and D). Connectivity among the nodes is shown for three years. We will use fictitious values for vulnerability and consequence of the nodes in this network in order to understand how the above-mentioned model helps to identify nodes that are most critical for the operation of the network. Also it will facilitate in understanding the various factors that contribute towards the criticality measure.

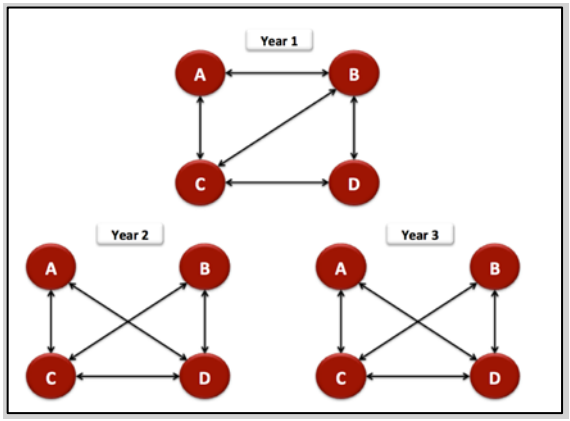

Figure 1: Critical Node Analysis for a Synthetic Network

Table 1 shows the results for extended PRA. In Year 1, we notice that node C is the most critical. Although it has the smallest consequence value in the network, its high connectivity and largest value for the vulnerability are responsible for its critical condition.

In Year 2, however, node C is not the most critical node anymore. This is due to the reduction in its consequence measure. Node D appears to be the most critical because of the increase in its degree. Its vulnerability and consequence values did not increase from the previous year.

In Year 3, Node A becomes the most critical node because of the increase in its vulnerability and consequence values. However, its degree did not increase.

Year 1

	g	V	C	r = gVC
A	2	0.01	15	0.3
B	3	0.02	20	1.2
C	*3*	*0.6*	*10*	***18***
D	2	0.3	15	9

Year 2

	g	V	C	r = gVC
A	1	0.07	10	0.7
B	2	0.01	30	0.6
C	3	0.5	2	3
D	3	0.3	15	***13.5***

Year 3

	g	V	C	r = gVC
A	1	0.6	20	***12***
B	2	0.01	30	0.6
C	3	0.5	7	10.5
D	3	0.1	10	3

Table 1: PRA of the synthetic network for three years

Figure 2 shows the change in extended risk values for the nodes that indicate node criticality during the three-year time-span. This simple illustration helps us to understand the significance of incorporating a node's degree (g) for the computation of its risk along with its vulnerability and consequence (Al-Mannai & Lewis, 2007).

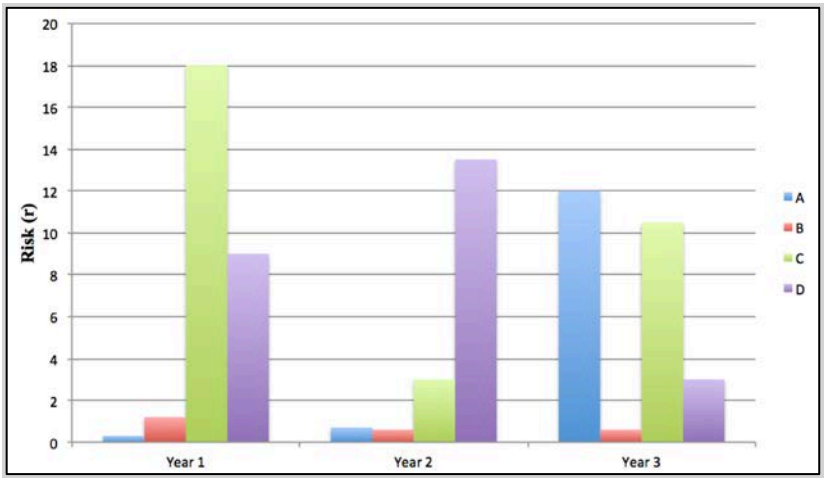

Figure 2: Critical node analysis for the synthetic network

Critical Node Analysis for a Small MDAP Network

Implementing the above-mentioned technique of extended PRA is a non-trivial task for MDAP networks. PRA requires a reasonable estimation of a priori approximations of vulnerability and consequence of the network assets. However, there is no guideline to do such estimation for MDAPs. Moreover, data on the MDAPs are complex artifacts and often times are either incomplete or fuzzy. Therefore, defining vulnerability and consequence parameters for MDAPs is a challenging task that we address below.

MDAPs operate on a multiplex network. At one hand, MDAPs share funding with other MDAPs (as a result they form a shared-funding network); on the other hand, MDAPs share hardware/software components with other MDAPs (as a result they also belong to a programmatic network). Therefore, performance of MDAPs can be examined in light of the performance of the individual program (program-centric) as well as its resulting performance in two different networks (network-centric): (1) a **programmatic network** and (2) a **funding network**. In our analysis, we consider both the program-centric and network-centric contributions.

Below, we first discuss how to discover diverse (programmatic and funding) network relationships among the MDAPs and form a multiplex network. Then we define the various parameters for the extended PRA model. Finally, to validate the approach for extended PRA of the MDAP network, we present a case study of critical node analysis for an MDAP enterprise.

Multiplex Network Formation

The interdependency of the MDAPs that influence their performance can be best understood via the programmatic network (Brown, M. M., 2014). In a programmatic network, individual MDAPs support other MDAPs by providing software or hardware components. Therefore, our network of interest is based on the programmatic relationships that exist among the MDAPs. We have gathered data on programmatic interdependencies from the DAES reports for the respective MDAPs. Typically, the last page of the DAES report records the inbound and

outbound connections.

Apart from their programmatic dependency, the MDAPs are also related via common PE accounts. In our network model, we capture this funding network relationship as well.

Both the programmatic and funding relationships on the same set of MDAPs are superimposed to define a multiplex MDAP network. For example, Figure 3 shows both the funding and programmatic interdependencies among the MDAPs in an MDAP multiplex network in 2009.

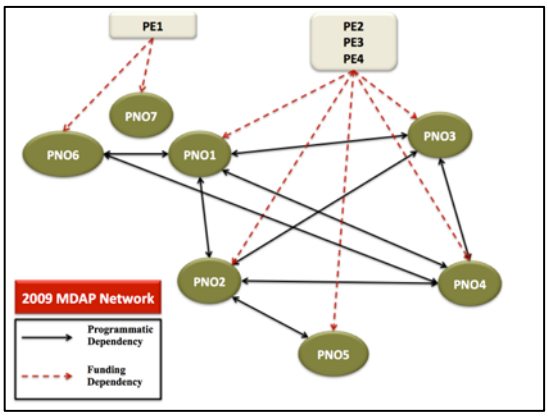

Figure 3: An MDAP Multiplex Network Model

Parameters of Extended PRA Model: Degree (g), Vulnerability (V) and Consequence (C)

We define the extended PRA model parameters below:

- **Degree (g):** It is defined by the number of outgoing edges from a node in the programmatic network. Therefore, degree measures the extent of influence of one program (node) on other programs. In a m-node network,

$$g = \sum_{i=1}^{m} n_i$$

- **Vulnerability (V):** It is a measure of weakness of a node in a network. It is defined as the probability of failure of a node if a successful attack is launched on it.

In the MDAP network, we notice that a program may become prone to failure; we call such a program a critical program. Our hypothesis is that breach incidences and other factors mentioned below are indicators of criticality of a program. Program failure is characterized by increased APB breaches and PAUC increase. Moreover, we hypothesize a program's criticality could potentially influence its neighbor's performance (increased breached condition and PAUC increase.

- First, from a program-centric point of view, a program may fail due to its intrinsic poor performance. For example, weapons procurement cut could lead to intrinsic poor performance (Raja et al., 2012). This program-centric view is captured in the "Program Status" page of the DAES report of the programs.
- Second, APB beaches and the percentage of increase in PAUC is also a measure of a program's performance. These values are recorded in SAR files.
- Third, from a program-centric perspective, the number of its funding and programmatic neighbors influences the performance of a MDAP. For instance, having a large number of funding neighbors (per PE account) makes a program susceptible to potential reduction in promised funding as funds could be siphoned to its neighbors.
- Fourth, having a large number of upstream programmatic neighbors (from which the edges fall on the program) increases its dependency of software/hardware components for successful completion of its tasks.
- Fifth, funding lag also affects the performance of a program and may make it prone to failure.

We propose that the above-mentioned five parameters provide a reasonable estimation for the probability of failure of a program and use these to define vulnerability (Lewis, 2009). Therefore, vulnerability should be considered as the cumulative effect of these parameters. We define the normalized vulnerability based on these parameters using a simple linear function and study its effectiveness:

$$\mathrm{V} = \frac{p+b+fNbor+pNbor+diffF}{1+1+1+1+1}$$

Each parameter in the numerator has a maximum value of 1. In the following the individual parameters are formally defined.

p: It refers to a program's intrinsic performance (captured in DAES reports) and is a linear combination of the factors contributing to Program Status. We use the December DAES report for the last reported month of a year for this computation. Last reported month's data is used as it provides that year's intrinsic performance level of the program. We use the data provided in the "Program Status" page of the DAES reports to compute this metric as described in Table 2.

b: It refers to the number of breaches that occurred in the current year (retrieved from SAR files).

fNbor: It is the normalized number of funding neighbors (retrieved from R docs).

pNbor: It is the normalized number of upstream programmatic neighbor (retrieved from DAES reports).

diffF: It is the normalized differential between received and promised funding amount (retrieved from SAR and R docs).

Parameters	Formula
p	$\frac{Cost + Schedule + Performance + Funding + Life\ Cycle\ Sustainment}{10+10+10+10+10}$ For the five "Program Status" variables, Cost, Schedule, Performance, Funding and Life Cycle Sustainment, we map the following quantitative values for the colored bubbles: Green: 0; Yellow: 5; Red 10 The value of p is normalized by the maximum numeric values (i.e., 10) for each status variable.
b	$\frac{APB\ SchedBreach + APB\ PerfBreach + APB\ Cost\ (RDT\&E)Breach + PAUC}{1+1+1+25}$ where APBSchedBreach =1 if APB Schedule Breach occurred, 0 otherwise; APB Cost (RDT&E) Breach = 1 if APB Cost (RDT& E) Breach occurred, 0 otherwise; APB PerfBreach if APB Performance Breach Breach occurred, 0 otherwise **PAUC**: it is captured from the "Unit Cost" section of the SAR. We use the Current year value. Since a "critical" Nunn-McCurdy breach occurs when the program acquisition or the procurement unit cost increases 25% or more over the current baseline estimate, we use 25 as the maximum value for PAUC.
fNbor	$\frac{\sum_{i=1}^{n} fNbor_i}{fNbor\_Max}$ Here, the subscript *i* refers to each PNO account and fNbor_Max is a predefined large value that is used for normalizing fNbor. We define fNbor_Max as follows, fNbor_Max = Total PE accounts in the network * Total number of MDAPs
pNbor	$\frac{Number\ of\ upstream\ programmatic\ neighbors}{pNbor\_Max}$ pNbor_Max is a predefined large value that is used for normalizing pNbor. We define pNbor_Max as follows, pNbor_Max = Total number of MDAPs in the network
diffF	$\frac{Promised\ Funding - Received\ Funding}{Promised\ Funding}$

Table 2: Formulas for the 5 parameters used in the computation of Vulnerability

- **C (Consequence):** Consequence measures the damage or loss (in dollars) of an asset when failure occurs. Therefore, it should be proportional to the RDT&E funding (from R Docs) and is determined by the breach condition. For example, if a program experiences 100% breach, then its Consequence would be tantamount to its entire RDT&E funding. We define it as follows:

C = b * Funding (RDT&E) NBC*007481075

The breach parameter b from the vulnerability computation is used to compute Consequence.

2.4 Case Study: An MDAP Network

We use the extended PRA to identify the most critical nodes for an MDAP enterprise that consists of six MDAPs: PNO1, PNO2, PNO3, PNO4, PNO5, and PN06[1]. These six MDAPS are funded by 4 program elements (funding sources): PE1, PE2, PE3, PE4 as shown in Figure 4.

Data for the years 2009 to 2011 are used for this case study. Figure 4 shows the MDAP enterprise multiplex network for these three years.

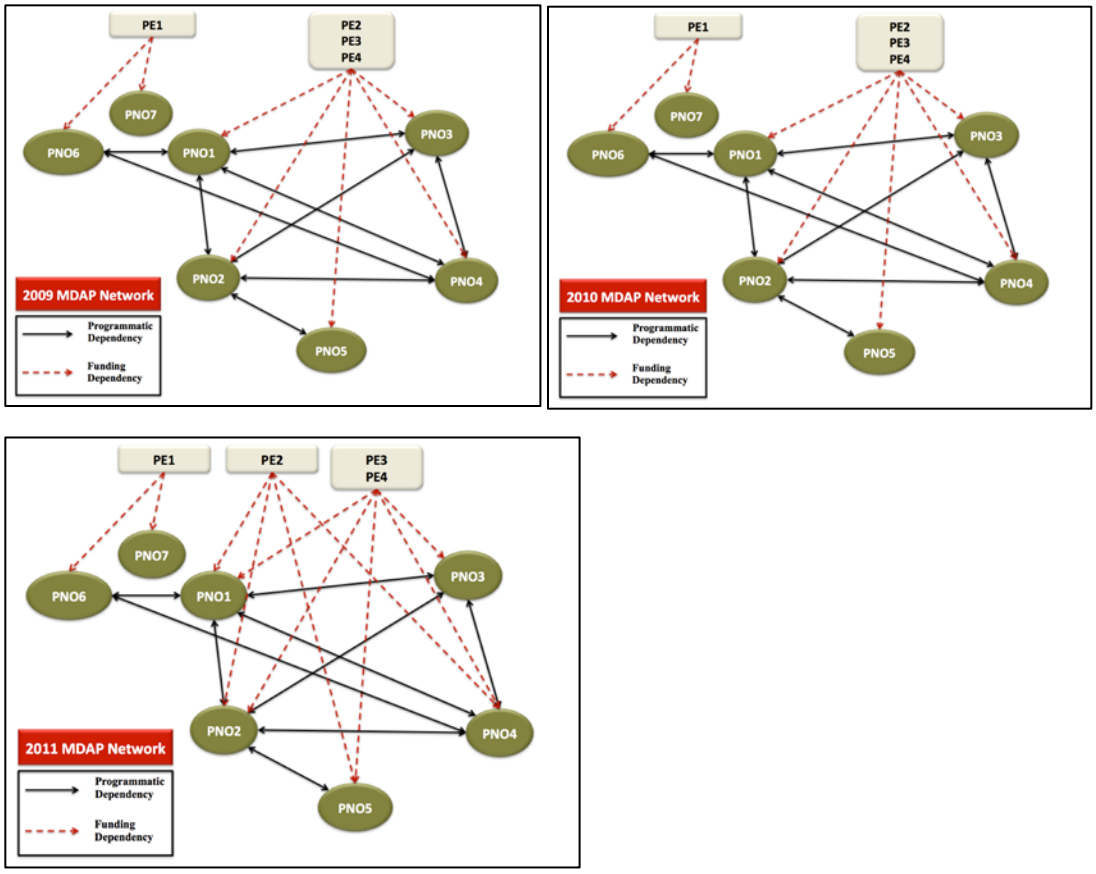

Figure 4: The MDAP Enterprise Multiplex Network from 2009 to 2011

[1] the name of programs PNO1 to PNO6 have been withheld since the "data is classified as official use only" FOUO

Detailed calculation of the risk values for each MDAP for three years were performed. Table 3 shows the summary of the results.

	Risk (r)		
	2009	**2010**	**2011**
PNO1	18.11	21.02	97.05
PNO2	0	6.97	3.75
PNO3	6.16	18.89	100.52
PNO4	4.88	11.39	0
PNO5	0	0.17	0.48
PNO6	24.22	24.9	19.7

Table 3: Critical node analysis for MDAP enterprise network

As an illustration of the calculations in Table 2, we show the detail calculation of the risk value for PNO1 in 2009 in Table 4.

Parameters	**Calculation**
p	$\frac{Cost + Schedule + Performance + Funding + Life\ Cycle\ Sustainment}{10+10+10+10+10}$ $= \frac{0+0+0+0+0}{10+10+10+10+10} = 0$
b	$\frac{APB\ SchedBreach + APB\ PerfBreach + APB\ Cost\ (RDT\&E)Breach + PAUC}{1+1+1+25}$ $= \frac{0+0+0+2.45}{1+1+1+25}$ $= 0.0875$
fNbor	$\frac{\sum_{i=1}^{n} fNbor_i}{fNbor\_Max}$ $= 12/28 = 0.429$
pNbor	$\frac{Number\ of\ upstream\ programmatic\ neighbors}{pNbor\_Max}$ $= 4/6 = 0.6667$
diffF	$\frac{Promised\ Funding - Received\ Funding}{Promised\ Funding}$ $= \frac{215.934(0604280N) - 212.6(SAR)}{215.934(0604280N)}$ $= 0.015439903$
Vulnerability (V)	$V = \frac{p+b+fNbor+pNbor+diffF}{1+1+1+1+1} = 0.2396356$
Consequence(C)	C = 18.894225
Risk(R)	gVC = 18.11091575

Table 4: Detail calculation of the risk value for PNO1 in 2009

From Figure 5, we observe that over the years PNO1 and PNO3 became the most critical programs in the network. PE6 retained its criticality level and we do not see significant improvement. A careful analysis of the data for PNO1 and PNO3 in year 2011 reveals that both programs have high breach incidence (that includes increased PAUC). As a result, their consequence values increased as well. Also these two programs were characterized by higher degrees. All these factors contributed to their high level of criticality. For PNO6, although its degree is relatively small, it has been experiencing schedule and cost breach as well as increase in PAUC for three consecutive years. The funding budget for PNO1 and PNO6 (over $300 million dollar) is also a contributing factor.

According to 2011 SAR files, PNO1, PNO3 and PNO6 experienced significant PAUC increase and APB breaches indicating their poor performance level. This observation confirms our risk computation measure is a step in the right direction.

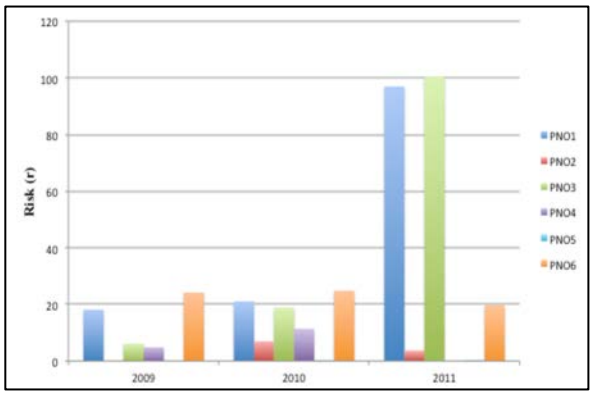

Figure 5: Critical node analysis for the MDAP enterprise network

Discussion of extended PRA model for risk computation

The objective for defining the risk parameter (R) in this paper is to capture the effect of multiplex network relations on a MDAP program's performance. As mentioned earlier, breach conditions (from DAES & SAR) are indicators of a program's intrinsic performance but do not account for the exogenous effects on a program. We have developed the PRA risk model with the potential to capture the network effect on a program's performance. In this model, the intrinsic parameters (p and b) tell us whether a program is "Vulnerable", while the MDAP program's

network status accounts for “Criticality”. The premise shown by the case study is that this criticality measure helps identify programs that are susceptible to future breaches more effectively than by simply using their intrinsic performance parameters (p or b values).

Manual analysis of MDAP data (done in previous phases of the project) facilitated the process of modeling a small MDAP network for extended PRA analysis. For this modeling, we considered the multiplex nature of the MDAP network and used various performance reports. The results indicate that the extended PRA technique has the potential to successfully identify risky programs and infer the performance of programs.

Also, the PRA analysis uses a network-based composite metric, instead of just individual program PAUC increase and APB breaches, to compute the risk level of a program. For example, both PNO1 and PNO3 have relatively high degrees of risk and share the same funding accounts, making them susceptible to poor performance. By looking at their increasing PRA risk values in 2009 and 2010, it can be inferred that these two programs are in critical conditions. Also by looking at the nearly stable high-risk values of PNO6 in 2009 and 2010, this program should be considered as critical as well.

Hence, with the aid of an automated information retrieval mechanism from the performance reports, it is possible to develop an algorithmic tool to identify risky programs. Recognizing the potential of these risky/critical programs to affect the performance of their neighbors could contribute towards predicting cascading effects. As future work, we plan to verify this empirically.

Also, we plan to use this model on another MDAP network to determine if it is able to identify the critical programs, we will modify the parameters of our PRA based model (if necessary) and use this knowledge to define a general model for the entire MDAP network as whole or more realistically, specialized PRA models for classes of similar MDAPs.

Studying the Feasibility of Mathematically Modeling the Phenomenology of MDAP Networks

We also conducted a feasibility study for modeling interdependent networks as a coupled dynamical system and potentially adapting the algorithms for feed-forward networks (Mintchev & Young, 2007; Lanford & Mintchev, 2013) to risk propagation interdependent networks like the MDAP network. To do this, we would have to determine the network model which includes determining network architecture properties including various centrality measure, strength of network connections including a precise form of the coupling formalism (strength can be seen as a precise rule that determines dynamical evolution), state features and action options which were already determined in Raja 2012, a reward optimization model that provides some dynamics to this network The model would allow us to investigate whether the system has any attractive equilibria, as well as determining the strengths and weaknesses of the basins of attraction. For example, if the steady state of the MDAP network is characterized by only one funded program, with all others having discontinued funding, this is probably not good. We hypothesize that if good equilibria were discovered, an outcome of this analysis could be to recommend a funding strategy that maintains equilibrium or guarantees a rapid convergence toward it.

The specific working hypothesis in the context of the MDAP network is as follows: The programmatic interdependencies between MDAPs have a profound influence on large-scale network performance over an extended period of time.

To determine a network model that is descriptive, predictive, and mathematically sound, we would need a collection of numerical quantities either measured, or somehow computed from other measurement recorded in time series over a sufficiently long period of time. This would involve:

(R1) determination of observable quantities measured numerically, i.e., real numbers on a well-defined scale. The key characteristic is to have some a priori evidence that the observables chosen evolve dynamically - i.e., change over time; also, it is absolutely necessary for these to be numerical, or to correspond to some sort of real number scale.

(R2) finding time series of data on the observables chosen in (R1). Usually a lot of data over a sufficiently long time scale is required to build this historical account of how the observables have changed over time. If the model is to be predictive in the short term, then the variables/observables must have been sampled at a sufficiently high rate.

Evaluating DAES data

We began by studying the DAES data of several MDAPs collected over a decade with the hope that the sequential monthly data would provide indicators of performance degradation. We have extensive experience with DAES data from our previous work, where we used DAES data to study local and non-local issues that affect the performance of the MDAP (Raja et al, 2012) and also developed sophisticated text and image extraction tools (Raja et al., 2013; Raja et al., 2014) to automatically extract the DAES data en masse.

Since changes in total cost could be considered as a useful observable, we constructed a few test time series based on the information captured on Top Cost Drivers in the DAES report. Figure 6 captures one such example. It became clear the cost driver time series was not sufficiently volatile enough to facilitate predictability.

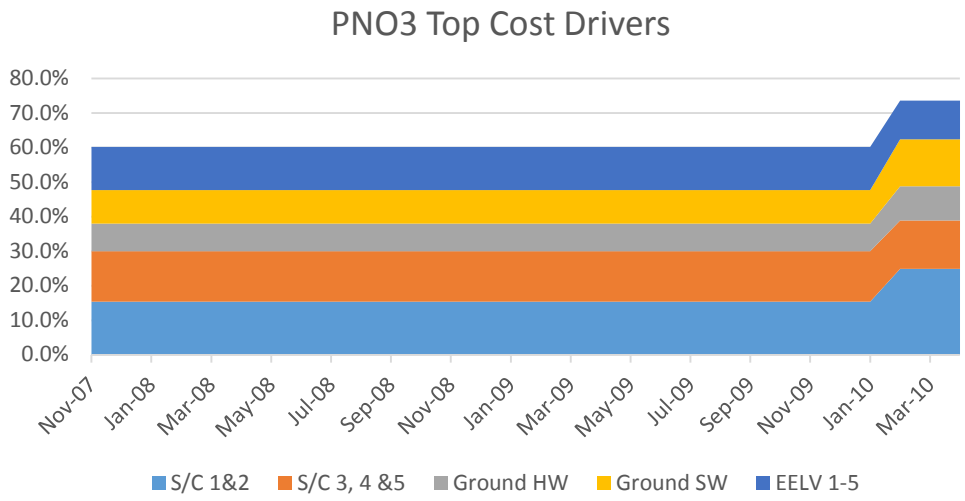

Figure 6: PNO3 Top Cost Drivers from November 2007 to March 2010

Figure 6: Stacked Area Time series data of 5 top cost drivers of PNO3. While there is some volatility in January 2010, the volatility is not frequent enough to capture the change in performance risk of the MDAP program over time

Moreover we ran into several challenges with the preciseness of the data as far as our goal of building a mathematical model is concerned. Some of our observations are captured below:

In the DAES Program Status page

- We could not ascertain the quantitative mechanism for color transitions of the red, yellow or green bubbles that capture the changes in value of Cost, Schedule, Performance etc. in going from one month to the next.
- The risk in the Risk Summary page describes the risk computation in somewhat of a quantitative way. However, it was still unclear how the risk quantity evaluated; it seems to be coded by a 2-dimensional vector, a (consequence, likelihood) pair; how (if at all) is each of those coordinates computed?

In the DAES page with Top Cost Drivers, Technology Readiness Assessment, Performance (kpps) and Acquisition Program Baseline (APB)

- All of the KPP diagrams seem to be set at T(or threshold); it was not possible to ascertain how these quantities were computed and whether they change over time?
- While the technology readiness assessment box is another potentially interesting measure with regards to building a state space model, the values did not change for long periods of time and so were at a course level of granularity.

In the Finley charts of the DAES reports,

- The knowledge gained from the Finley charts is that the dependencies are of some programmatic importance--they can affect the course of the subject program--otherwise they wouldn't be mentioned. So of all the potential types of interdependencies that could exist among programs, the Finley charts show those that present a potential risk to the program in question (subject to the limitations of the format and the awareness of the program manager). The dependencies described by the Finley charts generally relate to some component or subsystem in the subject program (or system) that must be provided by, or is somehow dependent upon the external program (or system). In many cases (actually, most cases) the external entity is a non-ACAT 1D program. There is no requirement for those programs to report their data to OSD via the SAR and DAES. In fact, the data for those programs will be held by the program office or their Program Executive Offices within the military department. This makes

getting detailed data about the external program difficult.

- Also, the challenge with the Finley charts is that the nature of the dependency is usually not defined: it could be funding, schedule, or some technical issue. Given the shortcomings of the Finley charts as a way to represent programmatic interdependencies, other more objective representations of system interdependencies have been explored, particularly artifacts that describe the interconnections between the system in question and external systems. These data are in the Information Support Plan (ISP) that each major program generates as part of its milestone approval documentation. The difficulty with the ISP, however, is that the reports are more difficult to obtain, and recent changes in policy have made the data less analytically useful.

The data acquisition challenges could be summarized as follows: although there are some allusions to the idea that various quantities presented in the reports are quantitatively obtainable through formulas or calculations, there is not much explanation as to how this is actually done or what the numerical values/ranges would be and whether these definitions are consistent across all programs. This information is crucial to building a state space model for the MDAP network. Also the strategy for determining interdependencies seems to be a difficult. Also given the time lag (DAES reports are generated monthly) and the level of data captured, often there was not variation in the data from one month to the next.

Analyzing Contract data

We then deliberated on whether contract data would probably be a better data set for the type of time series based risk analysis we were considering. Instead of focusing on metrics related to contract value (looking for indicators of cost growth), we would instead look at the frequency of contract transactions.

The idea is that when a program is running smoothly, there's probably a baseline rate of contract modifications in the normal course of business (i.e., as funding is added, tasks are completed, deliverables received, etc.). However, when something traumatic happens like a test failure, or other technical difficulties, we could probably expect significant contractual "churn", as previously-planned efforts are realigned to address the mission-critical issue.

The following is a possible scenario where the "churn" metric might be a more

reliable indicator of program distress than cost: Consider a program that is composed of multiple components, each being developed under separate contracts (e.g., a satellite and it's ground control segment). If, for example, the satellite has a problem in development (i.e., a test failure), the satellite contract will probably experience cost growth, but the ground control segment might actually experience a decrease in expenditures, as it has to slow down to accommodate delays in the satellite. So whereas costs might increase on one contract, they might be somewhat offset by temporary decreases in the other, which would muddy the "signal" seen at the overall program level. However, each contract would probably have to be re-scoped in order to increase the level of effort for the satellite, and to reduce the level of effort for the ground segment. Thus, both will incur additional contract "churn" as a result, which should be observable by plotting the frequency of contract modifications over time.

Figures 7, 8 and 9 are the time series of the contract “churn” for the three MDAPs. Each contract transaction reported in Federal Procurement Data System – next Generation (FPDS-NG) has an "issue date" indicating when the contract modification was signed. We plotted the frequency of contract actions over time.

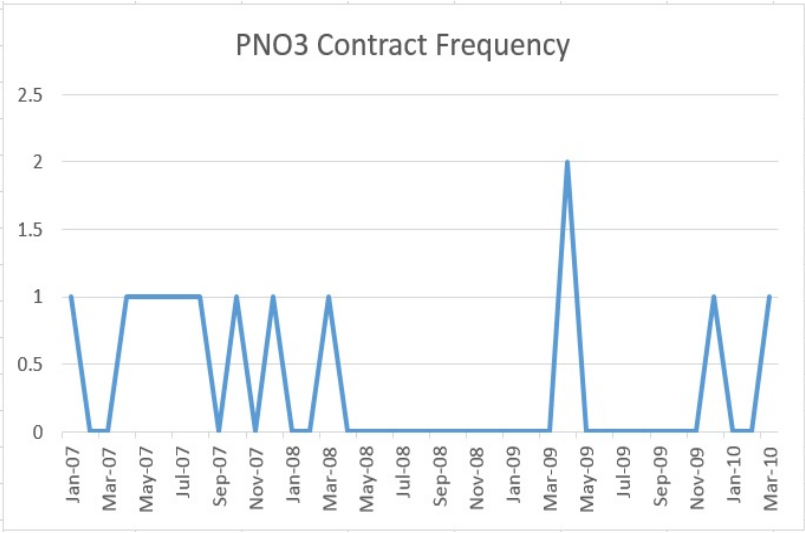

Figure 7: Time series data of PNO3-related issue dates.

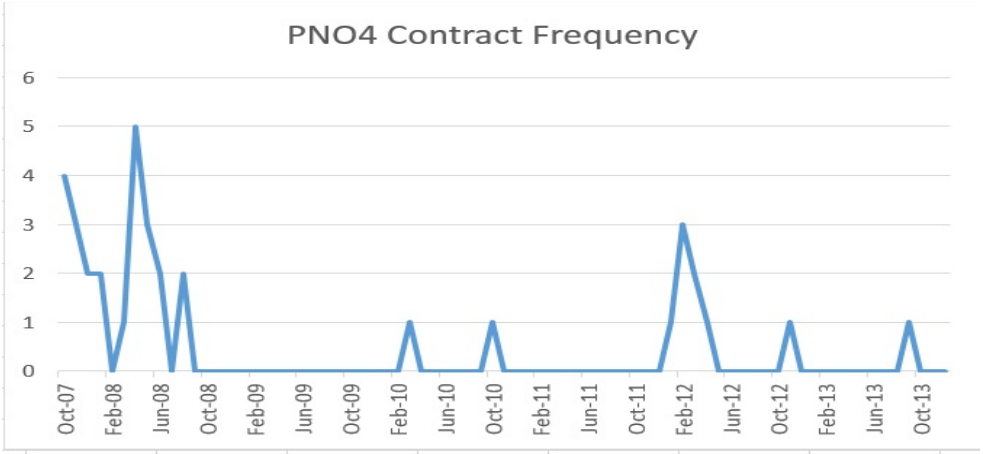

Figure 8: Time series data of PNO5-related issue dates.

In an effort to determine whether there is any type of correlation between the onset of significant contract churn in Figure 9 and program performance, we examined the breaches reported in the annual SARS data for PNO6. The December 2004, 2005, 2006 and 2007 SAR files show no APB or Nunn-Mccurdy breaches although the notes in the 2005 Threshold breach section states that there was a cost deviation from the key decision point-B approved APB even though there was no change in the total program cost as a result of the action. The 2009, 2010, 2011 SARS show Schedule and Cost RDT&E APB breaches with varying levels of explanations. The December 2012 SARS indicates no such breach. We are continuing to study the executive summaries as well SARS of future years in more detail.

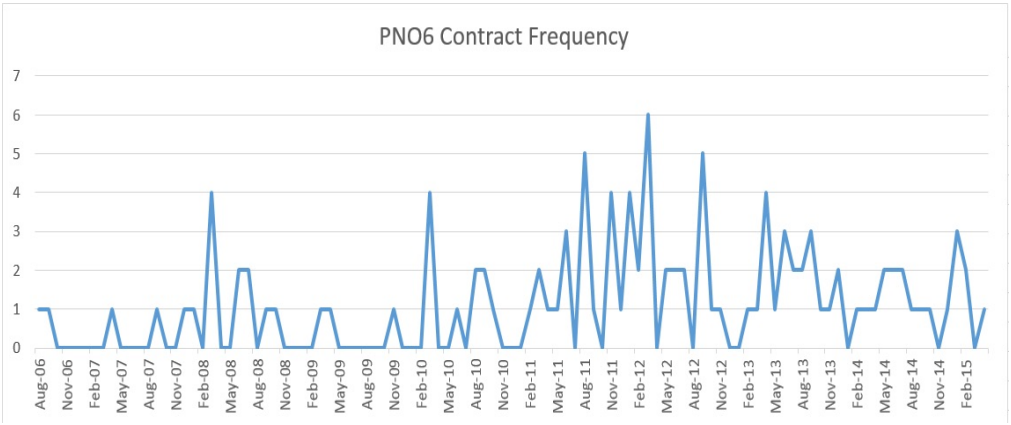

Figure 9: Time series data of PNO6-related issue dates.

Our observation from this examination of churn in contract data is that it does indeed have the volatility that could support the network modeling process. In addition to studying the PNO6 SARS data in greater detail as mentioned above, we are also trying to find contract data over a sufficiently long time scale to support our modeling analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

Reasoning about Uncertainty and Modeling What-If scenarios:

The goal of this task is to develop a statistical model that would address what-if scenarios in the MDAP network. For example, we seek to answer questions such as what if my partner reneges on a funding obligation or what if Congress alters my funding? These questions are not easily answerable due to the interdependent nature of the MDAP network. Many of these programs pool resources from many different sources, and there are many program dependencies as well. We investigate analytic tools to compute cascading risks and thus allow for mitigating actions that would avoid inefficiencies.

Prior work in What-If Analysis

In investment portfolio theory, investments are classified as sensible when their risks are justified and will be able to give output based on the given boundaries, or budget. Decision trees can be used to illustrate the different ways an investment may affect the portfolio. Real options analysis, for example, is a way for investment strategists to evaluate how to make long-term investment strategies. It has (Davendralingam and DeLaurentis, 2013) been shown that by mapping the MDAP network into a robust portfolio management problem while taking into account volatilities and uncertainties, the “warfighter performance index”, which was an indicator of the gains of selected MDAPs, can be maximized through optimization techniques that take into account risks (cost and developmental) and potential gains. The goal in the work was to find a way to balance capability and risk when adding systems, MDAPs, to the System of Systems.

Our approach is motivated by prior work in Risk Management of Large Option Portfolios via Monte Carlo Simulation (Avellaneda, 2016) which focuses on the volatility surfaces of large option portfolios. The goal was to present a statistical model of the data that could be used for tests and numerical implementations. The model is first created by identifying model risk factors (such as stocks) and then estimation techniques are used to determine correlations between the risk factors in

addition to the volatilities, or uncertainties of the risk factors. Furthermore, Monte-Carlo simulations (Mooney, 1997) are used to present risk scenarios that would give insight on risk associated with the portfolio.

To analyze the volatility surface, Principal Component Analysis (PCA) (Jolliffe, 2002) is used as a mathematical model that presents eigenvectors that contain an optimized amount of information concerning the options data and its volatility. By using PCA, risk factors were identified and correlations were found between them. MCSim was then implemented to predict outcomes of certain portfolios as an indicator of risk.

Background

Principal Components Analysis (PCA): It is a common statistical technique that is utilized in processing large quantities of data. Specifically, it finds patterns in data to simplify it for further analysis. Finding patterns in data could be useful in expressing data concisely by pointing out similarities and differences. Our project deals with high dimensional data. As a result, our research required us to understand efficient ways of analyzing it. PCA compresses the multidimensional dataset by reducing the number of dimensions without loss of information. This is done by taking principal components of the scatter matrix. Principal components are eigenvectors with respective eigenvalues; the highest eigenvalue is the first principle component of the data, and it represents the most significant relationship between the data dimensions. Eigenvectors are perpendicular. This means that they are uncorrelated with each other. By combining characteristics and weighing them based on influence, an eigenvalue is an optimized linear model of characteristics, or dimensions. As a result, this is the most efficient way to represent the data without superfluous correlated characters, etc. There is one eigenvector/eigenvalue for each dimension. By sorting them based on eigenvalues, from high to low, the most significant eigenvectors are found. At a certain point, eigenvalues become so small that they do not account for much of the information. As a result, it is efficient to set a lower bound on the eigenvalues and ignore the rest of the data from there. In most cases, components are cut off so as to leave data that accounts for at least 85-90% of the variance. Each individual eigenvalue, or principal component is a linear combination of the different characteristics. We apply this process to our set of contracts data. Principle components can be used to simplify the data. To do so requires the original data set to be described in terms of principle components as opposed to the known original components. By choosing optimal components that explain the most variance, the total amount of principle components needed to explain the data variance is generally much less than the original amount of components uses.

Monte Carlo Simulation: This method generates results by using repeated random sampling. This generally is used with a mathematical model that can be used for prediction or risk analysis. These models are statistical models, usually from experimental data or regression models. Linear modeling is another way to predict, or extrapolate, future data from experimental data. Process input values are entered into an equation that returns an output value. However, these kinds of models make it difficult to account for input variability, which is generally a natural symptom of real world situations. A relevant application of MCSim to this research problem lies in the relation to what-if analysis. Simulated data is used in real world applications when data collected is not sufficient or limited due to practical limitations. To begin, input values and their probability distributions must be defined. Typically, variables under observance will have a mean and standard deviation - this is easy to find in the data we have. What is different here from the initial one time run of an input value is that it accounts for a variance in the input data. By giving a specific probability distribution, output values are no longer always the same for each input value. Additionally, there is a need to set limits for the output values. It is assumed that there is a threshold that sets apart success and failure. This is a factor that should be decided, or calculated somehow, before the implementation. From every input value, many simulations are to be run. This will return a range of outputs to inputs ideally in a histogram. Naturally, there will probably be failure output values (values past the threshold) that are most likely towards the two tail ends of the histogram curve.

K-Means Clustering Algorithm: It is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. Understanding this algorithm will aide to the development of the risk predictor algorithm in this project. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because a different location leads to a different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest

centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function is a chosen distance measure between a data point x_i and the cluster center m_k, is an indicator of the distance of the n data points from their respective cluster centers.

Combining Principal Component Analysis and MonteCarlo Simulations

We use a combination of principal component analysis for the statistical model and MonteCarlo simulations for the numerical implementation (Jamshidian & Zhu, 1996) to generate risk scenarios to facilitate the what-if analysis using contract data. Principal component analysis allows us to replace a large number of variables with much fewer artificial variables that effectively represent the same data and are linear combinations of the underlying dataset. This is a necessary step due to the large number of variables and contracts to be analyzed. Then, we will identify the risk factors that could lead to MDAP breaches (schedule, cost, performance etc.) and estimate the correlations between the factors. Then, we will create a function that takes the data manipulated by PCA as input (an CSV file) and generates a probability model based on it. This will then be used to run a Monte Carlo simulation, which generates random values that will be plotted based on the probability model. The values will then be considered as breaches/non-breaches based on the nearest point's probability distribution.

Our approach is captured in the Risk_Scenario_Generation algorithm described in the appendix. To begin, MDAP contract data was collected with the expectation that it could provide insight of historical failure and risk measurements of

MDAPs. The contracts represent relationships between MDAPs and contractors who, instead of the DoD itself, produce new devices, vehicles, technological advancements, etc. Due to the large increase of these contracted development projects, the contract data set has become rather large—a potential wealth of information on indicators of risk. We examined the contract data closely in terms of one value of interest that is discussed before: the frequency of changes in contracts.

As shown earlier, changes in contracts, such as restatement of capabilities, change of requirements, or extension of time, could be possible indicators of risk of failure. We sought to dig deeper into ways failure could be predicted. The contract data had about 2000 contracts accounted for each year with 230 variables per contract —analyzing it all using typical methods would be tedious and time consuming. Nevertheless, the wealth of data held the potential for breach prediction and better risk management.

Experimental Setup and Results

The contract data has many fields, some of which have been deemed extraneous for the purposes of predicting project status (for e.g., an identifier as to whether the contracting business is owned by an Asian, Black, Hispanic person). *Csv_modifier.py* is a script which goes through the USA Spending Gov's CSV data files and extracts appropriate data fields (the appropriate data fields are determined by the user). The exact function that does this is *csv_modifier.modify_CSV(infile, outfile, company_name)*. This function takes an input CSV from USA Spending Gov, an outfile name for which to write the modified CSV file to, and the company name for which to extract data for. Continuous fields are left as is, while discrete fields with either yes or no indicators or string entries are converted to numerical entries. This new CSV file is now ready to have the PCA algorithm run on it, to further reduce field components.

The PCA algorithm is run on the modified CSV file using the *pca2.py* script. This script takes in the modified CSV file, runs the PCA algorithm on each "component" (i.e. column) and stores a list of eigenvalue fractions corresponding to each component. For example, if there were 4 components that the PCA algorithm

deemed as principal, and the first component had an eigenvalue of 1, the second had 2, third had 3, and fourth had 4, then the eigenvalue fraction list would be [0.10, 0.20, 0.30, 0.40]. This list of eigenvalue fractions is then written on the first line of the outputted CSV file, followed by a line stating Component 1, Component 2, Component 3, Component 4, and Breach. This line is then followed by appropriate values for each column (note that the Breach column is untouched during the PCA algorithm).

Going year by year, we then go into the USA Spending contracts data and look at all those same companies associated with program breaches. Since there is not enough information to show how an individual contract affects an MDAP's breach status, we assume that all contracts in that year under that company are potential contributors to a breach.

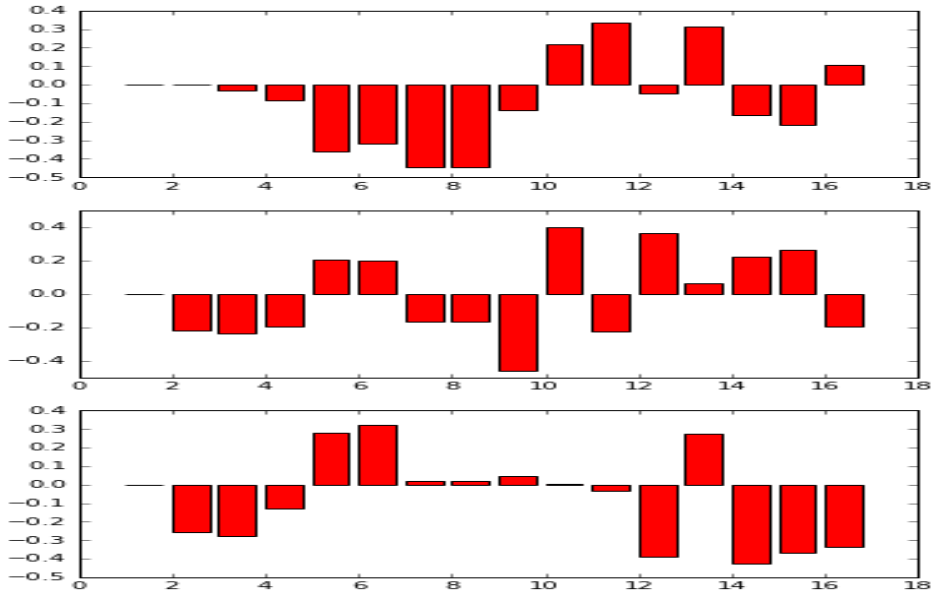

Figure 10: *From top to bottom*: First, Second, Third principal components; x-axis: Original components, y-axis: Relative proportions of original components.

Figure 10 is a plot of the first 3 PCs (more to be displayed) and the weights of the original components in each one. Together, they account for approximately 63% of the variance. 16 variables here can be analyzed quantitatively by noticing correlations. For example, columns 5 and 6 were negatively correlated with columns 11 and 13; this meant that signed/completion dates were negatively correlated with

contract types. These principal components will replace the original components and create a smaller, more efficient variable space for Monte Carlo simulation.

The user will be able to select a level of wariness for the risk generator, which essentially dictates how strict his/her thresholds are when deciding on breach probability. The CSV file (now PCA processed) will generate values that will be plotted in n dimensional space. According to the wariness level selected, each point plotted will have a corresponding n dimensional probability distribution. This will form somewhat of a core where the density is highest—the radius of this core will be decided on the wariness level.

Essentially, the riskier, or less wary, the user wants to be, the skinnier the distribution, or smaller the core, will be. Increasing wariness is, quantitatively, when the highest probability of the breach, which is in the space around the point, increases. For example, let's say that the boundary of a sphere with radius 1 centered at a breach point has a 25\% chance of a breach. By increasing the wariness, that boundary will now be increased to a radius greater than one. Since the Gaussian distributions trailing on the ends are further pushed, points outside the core experience an increase in their probability of breach. The core will maintain a uniform probability distribution, and its surroundings will look like a Gaussian distribution (subject to change). Each point will have an "influence region", discussed below.

After the PCA algorithm is run and a new CSV file is generated, a probability space can now be generated. *Dict_establish.dist_establish (input_csv_name, wariness, influence_region, core_radius)* is the function that handles the creation of a list of *dict_establish.Point* instances.

- Each Point instance represents a list of local probability distribution functions centered at a certain coordinate. Each Point has other properties as well, including a *core_radius* and an *influence_region*.
- The core represents an area with a uniform and maximum probability distribution function. The influence region represents two things: the first is the boundary demarcating when a new Point should be created (i.e. if an array of components is currently being considered, and it falls within the influence region, then it will be absorbed into the appropriate Point instance); the second is the value of *sigma*

for a Gaussian-like distribution. The distance between the core and the influence region for a specific component represents the value of *sigma* for that component. On an encountered array of components that falls within the influence region, the appropriate Point instance's core probability is scaled by an appropriate factor corresponding to the breach value for that array (note that the breach value can be a 0, 1, or 2 since the SAR data file has either a green, yellow, or red dot). The scaling factor is determined at point initialization such that the core probability will never exceed 1 (this is determined using the infinite geometric sum formula).

- These two fields will be user inputs into *dist_establish.* Upon initialization, a Point will take the *core_radius* argument, and will scale it appropriately, corresponding to each component's variance, and append the value in a list *core_fields_radii*, which represents the actual radii for each component.

A similar procedure occurs for *influence_region*, and the list created is *influence_region_radii*.

Bin_establish.bin_establish(input_csv_name, number_of_bins) takes in an input CSV file, which will be the CSV file after *pca2.py* is run and a number of bins to generate a list of bins, with each list of bins containing *number_of_bins* amount of bins. The purpose of this script is to establish the underlying distribution of the PCA data so that the simulation can generate values from a correct probability distribution. *Bin_establish.bin_establish* goes through the input CSV file, finds the minimum and maximum value for each column, and generates a list of bins for each column based on those minimum and maximum values. It then runs through the CSV file again, and marks how many times a point falls in each bin.

The reason for this procedure is to handle limitations of the data. We are hoping for the best continuous data distribution; however, it is inevitable that this is not the case. When looping through some values that are meant to be continuous, they will still end up as discrete probabilities. For example, if three contracts had values (12, 34, 50), then the probability distribution generated by simply looking at the discrete values would give us 1/3 probability of each of those values only. On the contrary, creating bins, each of which has a normal distribution within it, will allow us to have continuous values that address the ranges of the data better. Each bin has a probability based on the number of values that fell into the bin from the data. Each of these generated values will then be mapped to its closest plotted geometrical

neighbor (the ones that have a radius, core, and surrounding distribution) and will determine its probability of failure from that neighbor. These final probabilities will then be used to generate the final simulation report.

Conclusions and Future Work

In this paper, we have discussed our progress in our ongoing efforts to 1) study the impact of network topological characteristics on risk propagation and our methodology to quantify it; 2) evaluate the critical importance of quantifiable state features in order to assess network dynamics; 3) describe our investigation into time-series data that could facilitate our analysis.

Our initial results on PRA analysis for a case study and the contract data time series are encouraging and we plan to further investigate the scale-up of the PRA analysis as well as using the contract data towards building the network model.

The PCA and MonteCarlo based risk prediction algorithm has been shown to be viable in test cases. By generating random scenarios via Monte Carlo simulations, the algorithm is able to reproduce a risk index that matches the expected risk. In the future, we would apply this algorithm on processed DAES reports to generate a second round of testing on historic data.

THIS PAGE LEFT INTENTIONALLY BLANK

References

Albert, R., Jeong, H. & Barabasi, A. (2000). Error and attack tolerance of complex networks. *Nature*, 406, 378--382.

Al-Mannai, W. & Lewis, T. (2007). Minimizing network risk with application to critical infrastructure protection, J. Inform. Warfare 6 (2):52–68.

Avellaneda, M. (2016). Risk Management of Large Option Portfolios via Monte Carlo Simulations, Big Data Finance.

Brown, M. M. (2014). Acquisition Risks in a World of Joint Capabilities: A Study of Interdependency Complexity. Proceedings of Naval Postgraduate Schools 11th Annual Acquisition Research Symposium, pp 109-128. Monterey, CA.

Davendralingam, N., DeLaurentis, D.A., (2013) "Acquistion Management for System-of-Systems Affordability through Effective Portfolio Management", Proceedings of Naval Postgraduate Schools 10th Annual Acquisition Research Symposium, pp 15-17. Monterey, CA.

Jamshidian, F., Zhu. Y, Scenario simulation, (1996), Theory and methodology. Finance and stochastics, I., 43-67. Sage Publications.

Jolliffe, I, Principal component analysis. John Wiley & Sons, Ltd, 2002.

Lewis, T. G. (2009). Network Science: Theory and Practice, John Wiley & Sons, Inc., Hoboken, NJ.

Mintchev, S. & Young, L. (2009) Self-organization in predominantly feedforward oscillator chains. Chaos, 19(4):043131, 2009.

Mooney, C. Z, Monte Carlo simulation (Vol. 116). Sage Publications, 1997.

Lanford, O. & Mintchev, S. (2015) Stability of a family of traveling wave solutions in a feedforward chain of phase oscillators. Nonlinearity 28: 237-261, 2015.

Raja, A., Hasan, M. R., & Brown, M. M. (2012). Facilitating Decision Choices With Cascading Consequences in Interdependent Program Networks. Proceedings of Naval Postgraduate Schools 9th Annual Acquisition Research Symposium, pp. 197-220. Monterey, CA.

Raja, A., Hasan, M. R., Rajanna, S., & Salleb-Aoussi, A. (2013). Leveraging Structural Characteristics of Interdependent Networks to Model Non-linear Cascading Risks. Proceedings of Naval Postgraduate Schools 10th Annual Acquisition Research Symposium, pp. 293-318, Monterey, CA.

Raja, A., Hasan, M. R., Rajanna, S., & Salleb-Aoussi, A. (2014). A Scalable Approach to Modeling Risk in the MDAP Network, Proceedings of Naval Postgraduate Schools 10th Annual Acquisition Research Symposium, pp 293-318, Monterey, CA.

Szell, M., Lambiotte, R., & Thurner, S. (2010). Multi-relational organization of large-scale social networks in an online world. In *Proc. Natl. Acad. Sci. U.S.A. 107*, pages 13636–13641, Morgan Kauffmann, San Mateo.

Appendix

Code documentation for What-If Analysis:

Algorithm: Risk Scenario Generation (RSG)

```
//GIVEN FILES: input_csv.csv
//This decides how many principle components to keep, based on the cumulative variance they explained
variance_threshold = 0.95
//These values decide the 'wariness' of the model – risk_value (between 0, 1) explains the probability
//distribution of the plotted point, influence_region explains the size of the breach risk region associated //with the point plotted. By increasing influence_region, a simulated point will be more likely to be //plotted into the breach risk region. By increasing risk_value, a simulated point in that influence region //will be more likely to be considered as a breach.
risk_level = 0.35
influence_region = 20
//simulation points to be plotted
number_of_points = 2000

modify_CSV('input_csv.csv', 'output_csv.csv', 'COMPANY_OF_INTEREST')
PCA('output_csv.csv', 'output_PCA.csv', variance_threshold)
list<Points> probPoints = dist_establish('output_PCA.csv', risk_level, influence_region)
list<list<Bins>> probBins = generateBins('output_PCA.csv')
breach_prob = runSimulation(number_of_points, probBins, probPoints)
```

Where:

procedure void modify_CSV(input_csv_name, output_csv_name, company_name):
Modify CSV input file according to PCA algorithm, identify breaches using DAES Report

procedure void PCA(input_csv_name, output_csv_name, variance_threshhold):
Take CSV file and simplify using our PCA model.

procedure list<Points> dist_establish(input_csv_name, risk_level, influence_rgn):
Take input csv file and begin plotting data points—each point has the following properties, a core, a surrounding probability distribution, and a risk level. The core is a uniform probability distribution with the highest probability and the surrounding regions are gaussian distribution tails.

procedure list<list<Bins>> generateBins(input_csv_name):
Establish probability of falling into various bins for each component of the input_csv_file. Each bin has a superimposed uniform distribution.

procedure int runSimulation(number_of_points, list<list<Bins> probability_values, list<Points> list_of_points):
Generates n points whose components are selected according to probability_values. Each point generated gets corresponded to a point in the list_of_points, and the probability of breach vs no breach is obtained from there. The total number of breaches / total number of points will be the overall probability for a breach.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net