# CMPLXSYS 530 Final Report: An Agent-Based Model of Communication Dynamics in Complex Engineered System Design Teams

John Meluso

April 26, 2018

# Contents

# 1 Introduction

Large-scale complex engineered systems (LACES) are a class of technical designed and engineered system with a significant degree of complexity, significant costs, long-duration development and operation timelines, numerous design cycles, and dispersed supporting organizations which span critical infrastructure and national defense systems alike. [3, 14, 13, 15] Systems engineering addresses these challenges through techniques like the "Systems V" to break a system down into its constituent physical or functional pieces before integrating them back into a larger complex system. [7, 10, 5]

However, these and other systems methods do not consider the effects of human decision-making cognitively or socially despite their known effects. The design evolution and system performance of these increasingly-expensive systems appear to depend on individual and group decision-making. [15] Therefore, LACES architects must consider both the technical and social structures which constitute LACES in order to reach optimal system designs. Given that agent-based modeling effectively simulates individual and group decision-making, agent-based modeling could inform intentional design of engineering teams. This paper seeks to represent the ways that one particular social dynamic manifests in these teams—miscommunication through a networked, hierarchical organization—through an agent-based model, and to study the effects of that social dynamic on system performance.

While this problem affects nearly all complex systems, it can be extremely difficult to isolate the effects of social factors on system performance in engineering and management contexts where products and results may not come to fruition for years or even decades. Fortunately, a relatively simple example is that of designing a satellite. Most satellites are designed and manufactured in a limited two to five year period, then launched into space—an isolated context—without potential of servicing the spacecraft. The design of the craft must therefore be robust enough to withstand a variety of risks and potential failures ranging from internal design and manufacturing problems to external environmental and operational constraints, and even market threats.

To design for robustness, systems engineers optimize several core requirements including total time to launch, cost, performance, and mass. While the former three of these values are common enough in other industries, a fundamental property like mass is uniquely consequential to aerospace applications due to both the exponential costs and capabilities required to launch increasingly large spacecraft into orbit.

The study of spacecraft mass, the processes that contribute to mass allocation in teams, and estimating what a spacecraft's mass will be provide a simplified way of observing how complex system performance is affected by social dynamics (such as management strategies, corporate culture, communication methods, etc.).[15] Mass is an example of a performance parameter, any number of which can affect a system's success. An engineer's ability to *predict* or *estimate* what the value of a parameter will be at the end of the design process ultimately affects the performance of the system itself due to feedback loops within designs and design objectives. For example, if a satellite team delivers a spacecraft which has 50 kg of mass which is unnecessary to complete the spacecraft's mission, that 50 kg is an unnecessary expense both in launch costs (at more than $10,000 a lb in many cases) and material costs. What if that 50 kg was in the solar panels, though? It may provide unnecessary current to the batteries meaning the batteries are larger (and therefore more expensive) than they need

to be. That mass also affects the structural requirements of the spacecraft body (called the 'bus') and may have caused an increase in the structure dimensions. A common refrain in the interviews was that 'mass begets mass,' which means effects on performance, cost, etc. Thus, misestimation affects performance.

Systems engineering handles this estimation "Uncertainty"—the error between the estimated value of a parameter and the actual value—through margins, a probabilistic technique for assessing the risk of future design changes. [2, 19]. While formal mathematical methods exist, engineers frequently use heuristics and intuition to calculate and manage margins. [19, 20, 8, 9, 18] My previous paper showed that engineers also use social "gaming" strategies to manage uncertainty. My current analysis demonstrates that communicative dynamics may also effect estimation, such as miscommunication—a difference between communicating parties in their understandings of the contextually and socially constructed meaning of a common term. In this case, that term is shockingly critical: the word *estimate*. How can engineers provide each other with valuable information about a design if they don't agree on the point in time which an estimate is designed to reflect? Does it reflect the current status of a design? Does it reflect an engineer's intuition or mathematical prediction of where a design will be in several years, regardless of where it is right now? Or some combination of the two?

Tthis report will attempt to capture the effects of differing understandings and definitions of an estimate on system performance. A simple network of agents will exchange information of certain types with different probabilities while favoring certain values, which was previously shown may affect system performance. [15] The system converges toward a design (reduces uncertainty) by treating the designs as samples drawn from random variables of historical information about previous designs. The methodology and construction will be discussed further in Section 3.

# 2 Related Works

Several authors in engineering fields have used agent-based modeling to simulate the teams which design and implement LACES as a means of understanding how team structure and design practices affect system outcomes. [15, 11, 12] Most engineering modeling of design teams arises from the world of design optimization where the objective is to maximize the design solution search efficiency of a team through the actions of individuals. [6, 22, 21] McComb et al. broke from the larger-scale optimization approaches when they chose to simulate the psychology of designers including multi-agency, self-bias, satisficing, etc. to form a design team agent-based model. [11] McComb's heterogeneous simulated annealing teams agent-based model scales these individual decisions up into a simulation of heterogeneous design solution search strategies, now considered the benchmark of engineering design agent-based modeling (in our small community). The HSAT multi-agent simulation creates a set of heterogeneous design agents whose design methods are collectively employed through simulated annealing optimization methods to produce optimal system outputs. [12] My own work stepped back to a more rudimentary ABM which examines biased information passing which results from well-intentioned incentive structures dictated by management, and supported by optimization techniques in real world design contexts based on interviews

with industry practitioners. [15]

Authors in management and organizations have likewise used ABMs to simulate organizational behaviors through network effects and game theory. My foundation in network theory developed through Mark Newman's course on Networks, their analysis, and modeling. [16] We see some of these characteristics exemplified in ABMs like that of Borgs et al. where information overload is simulated in social networks to simulate celebrity from posting quality and quantity. [4] But returning to teamwork, Anjos and Reagans demonstrate using a networked agent-based model of game theoretic actions and commitment strategies to optimize partnership outcomes. [1] A recent presentation by Reagans at the Ross School of Business demonstrated the negative effects which can result from biased learning in terms of optimal solution identification. [17] They created an ABM "maze of ideas" through which an agent navigated to reach an optimal answer to a problem using game theoretic searches and networked exchange of information to bias agents.

Reagans' model conceptually mirrors my own interests in biased information passing (similar to incentives) in team contexts, and provides a foundation if not for the optimization search (which is better drawn from the engineering algorithms which navigate complex variable spaces) then at least in terms of game theory choices and networked exchange of information. The real-world sociological dynamics are being drawn from a series of more than 95 interviews and 190 survey responses on social dynamics and other practices in design team contexts.

# 3 Model Description

The agents in this model are engineers in a large-scale complex engineered system. They interact with each other via a hierarchical reporting structure in which 7 agents report to an agent at the level above. This models a system breakdown in which a systems engineer oversees a group of subsystems engineers, or a systems engineering team oversees subsystems. The systems engineer communicates requests for information downward to their reporting agents who, with a varying probability, pass either information about their current design or their historical design.

Language is a negotiated activity in which both parties participate and must readily engage to achieve mutual understanding. My current work indicates that as much as 50% of the population of the team consistently misinterpret a standard quarterly request from the systems engineering team (hence the investigation). The estimates broadly fall into two categories which will be represented in the model:

1. The current state of design. When they receive a request for an estimate, some of the engineers interpret that request as an inquiry about what their current parameter status is. While they perform additional decision-making in some cases as to whether to communicate exactly where they think their design is today versus adding a little extra margin to protect themselves against future changes, they still generally conceive of estimates as something that captures the design as it is "right now."

2. A prediction of the production design. Others interpret the request for an estimate as an inquiry about what the parameters of their design will be upon completion of

the design process, essentially to make a prediction about what the values of their parameters will be at a future point in time. Again, people use a complex set of both quantitative and heuristic judgments to determine what that value will be, not least of which is a SWAG (or "Scientific Wild-Ass Guess"). But these individuals appear to distinguish between their design's current state of being and this projection or estimate of where their design will be at a later point in time, "at production" of their design.

Given these two categories, the model will assume that one group of agents just provide their design's current status. The other will provide the historical mean of their parameter's value because the primacy, anchoring, and adjustment heuristics are known to have effects on how people predict events.[15]

Once these estimates are provided by the subordinate agents to the superior, the superior agent will aggregate the parameter estimates via a simple sum, as one would a mass estimate for example, and evaluate the quality of the estimate with respect to a its own set of historical data. If it deems that the estimate is "not good enough," compared to a certain fraction of their historical dataset, they will send information to the subsidiary agents in the next model iteration to lower their own thresholds for activity. The subsidiary agents likewise have a utility associated with each estimate and if they deem the estimate "good enough" they choose not to expend additional resources to update the estimate; however, the superior can shift their threshold upward so that the subordinates all update their estimates

## 3.1 Agents

Every agent in the Python ABM is an instance of a single class **engineer** which can receive requests for estimates, assess the quality of their estimate, update their estimate, request estimates of any existing subordinates, and supply an estimate to a superior. So class **engineer** has methods:

**init** Initialize the class with parameters for its utility offset, probability of selecting one of two communication types as compared to a uniform random variable, and subsidiary agents in the network. If the communication probability comm_prob is greater than the uniform random variable drawn for this agent, then the agent returns the current status, otherwise it returns the predicted outcome. The subsidiary agents input is a list of agent names which this agent oversees in a directed network.

**repr** Returns the agent's name for referencing in loops.

**calc_util** Calculates the utility of the agent's current estimate. The utility function is a standard normal distribution which is offset from 0 by the offset parameter set in the init method.

**adj_util** Updates the thresholds of every agent subsidiary to this agent.

**gen_hist** This function generates a historical mean for the agent if the agent doesn't have any subsidiary agents, and sums the historical data from its subsidiary agents if it does. It generates its own historical mean by drawing a single random normal value for the mean and a single chi square value for the variance. It continues to return that

information to superior agents whenever requested, regardless of the threshold. This function also defines the utility threshold as the utility associate with its historical mean.

**give_est** Chooses to give the appropriate estimate to the superior agent depending on which type of estimate this agent is providing.

**update_est** Updates the agent's estimate. First checks whether or not the estimate is above the specified threshold or not. If above, it returns its latest estimate. If below, update the estimate with either its contributors' estimates or by generating its own new estimate if applicable.

## 3.2 Interaction Topology

The agents in this model form a directed network with one agent at the root with edges to 7 agents, where each edge contains a request for an estimate. Each of those 7 agents then has an edge back to the root agent wherein it passes its current estimate. The root agent then evaluates the estimates and, if the estimates are not good enough, it increases the utility threshold required for stopping. The root therefore has a second set of 7 directed edges to the subsidiary agents upon which it increases their utility thresholds.

In a sense, this is an unnecessarily complex way to describe the interactions between 8 agents. However, it could become fruitful later as the model is expanded to multiple managers, multiple layers in the tree, and multiple objectives which affect the agents which they must then balance across.

A physical environment is not applicable for this model.

## 3.3 Schedule

Each run of the model performed 10 system-level estimate updates, simulating a request from a manager to their subordinates.

1. Ask system agent to give estimate. Triggers system to update its estimate.
2. Update system estimate. Triggers system to request estimates from subsystems.
3. Ask subsystem agents to give estimates. Triggers subsystems to update estimates.
4. Subsystem evaluates whether to update its estimate or not and returns an estimate.
   (a) If already meeting own utility threshold, return the last estimate. (otherwise, not meeting threshold)
   (b) If not meeting own utility threshold, update own estimate.
   - If agent has contributors:
     i. Update contributors' thresholds.
     ii. Get contributor estimates.
   - If agent doesn't have contributors:

i. Update own estimate.

(c) Return new estimate to system agent.

5. System evaluates estimate utility and updates thresholds of subsystems if needed.

6. Perform 10 loops (9 more times).

# 4 Results and Analysis

A Monte Carlo simulation performed 1000 runs for each combination of two parameters: the communication type probability from 0 (all agents return predictions) to 1 (all agents return current values), and the utility offset of the utility distribution, from 1 standard deviation below the historical mean to 1 standard deviation above, as I demonstrated in my last paper had the effect of potentially increasing system uncertainty via subsystem-level decision making in light of system-level responses. The total number of runs for the Monte Carlo was 441,000 executions.

The objective of this experiment was to determine whether varying the types of communication and their prevalence in the system would have any substantive effect on system performance or not. The follow are a set of plots showing first histograms of a few parameter combinations from the Monte Carlo (Figure 1), then contour plots of the parameter sweep averages and variances for the system performance (Figure 2), and single-parameter graphs averaged across all other values (Figure 3).

# 5 Discussion and Future Work

Figure 1 first shows the unbiased case and that it performs roughly as expected. There is more variance in the distribution than with a standard normal distribution, but after only 10 runs for each model (simulating real-world design cycles called the Deming Cycle), it's no surprise that the sample variance remains larger than the population variance. The middle, representing that all agents passed only their current estimates, shows no discernible difference from the unbiased case. The final offset case shows the bias identified in my previous paper, a skew in the negative direction with approximately the same distribution width.

Most clearly visible from the upper-left and middle-left contour plots in Figure 2, the communication probability (the vertical axis) has no noticeable effect on system performance which is not already accounted for by the utility offset. As the model is configured today, it seems virtually random, further evidenced by the lower-left and upper-right contour plots which appear highly randomized. It's possible that some small effect is present in the lower-left plot of system utilities wherein the system utility is slightly higher at a communication type probability of 0 (all current values, no predicted values), but even that seems tenuous.

Figure 3 seems to reaffirm that suspicion, that while there may be an extremely slight increase in the utility values associated with low communication probability values in this model, the magnitude of the change in the utility value is negligible compared to that of mean utility. Contrasting against the lower-right graph in Figure 3, for example, shows three-fold

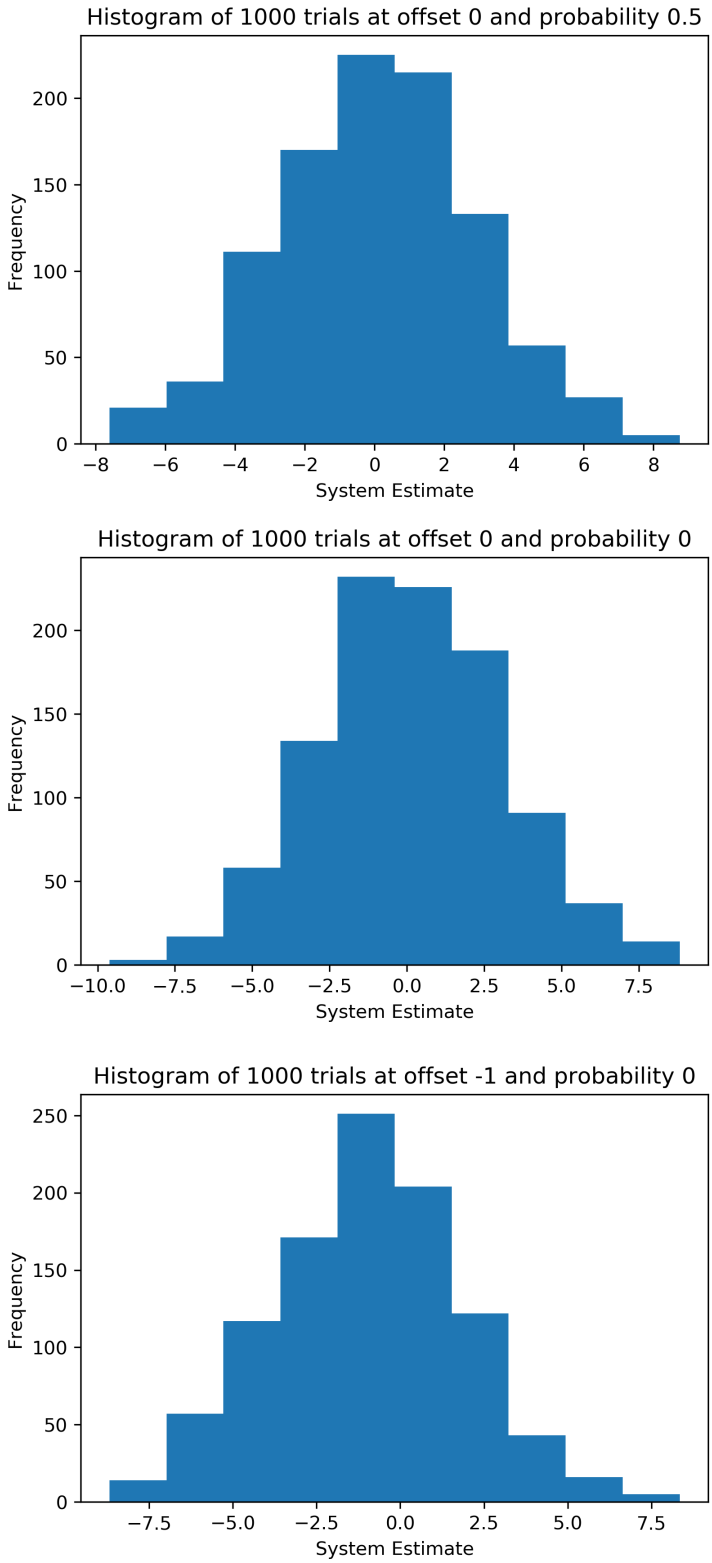


Figure 1: Histograms of the unbiased case (top), communication only edge of the swept space (middle), and offset only edge of the swept spaces (bottom).

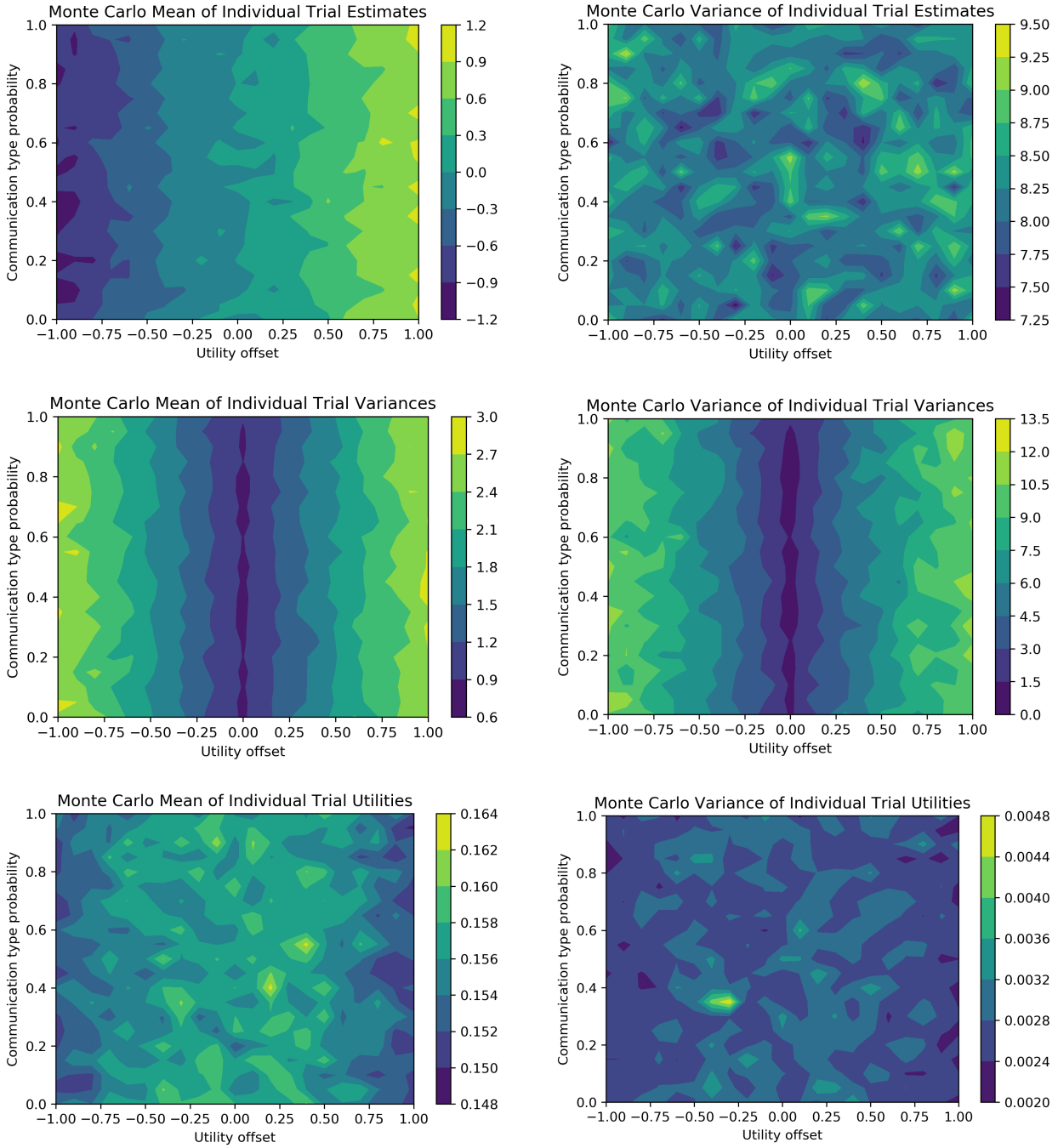


Figure 2: Contour plots of the system means (estimate), variance (uncertainty), and utility (quality).

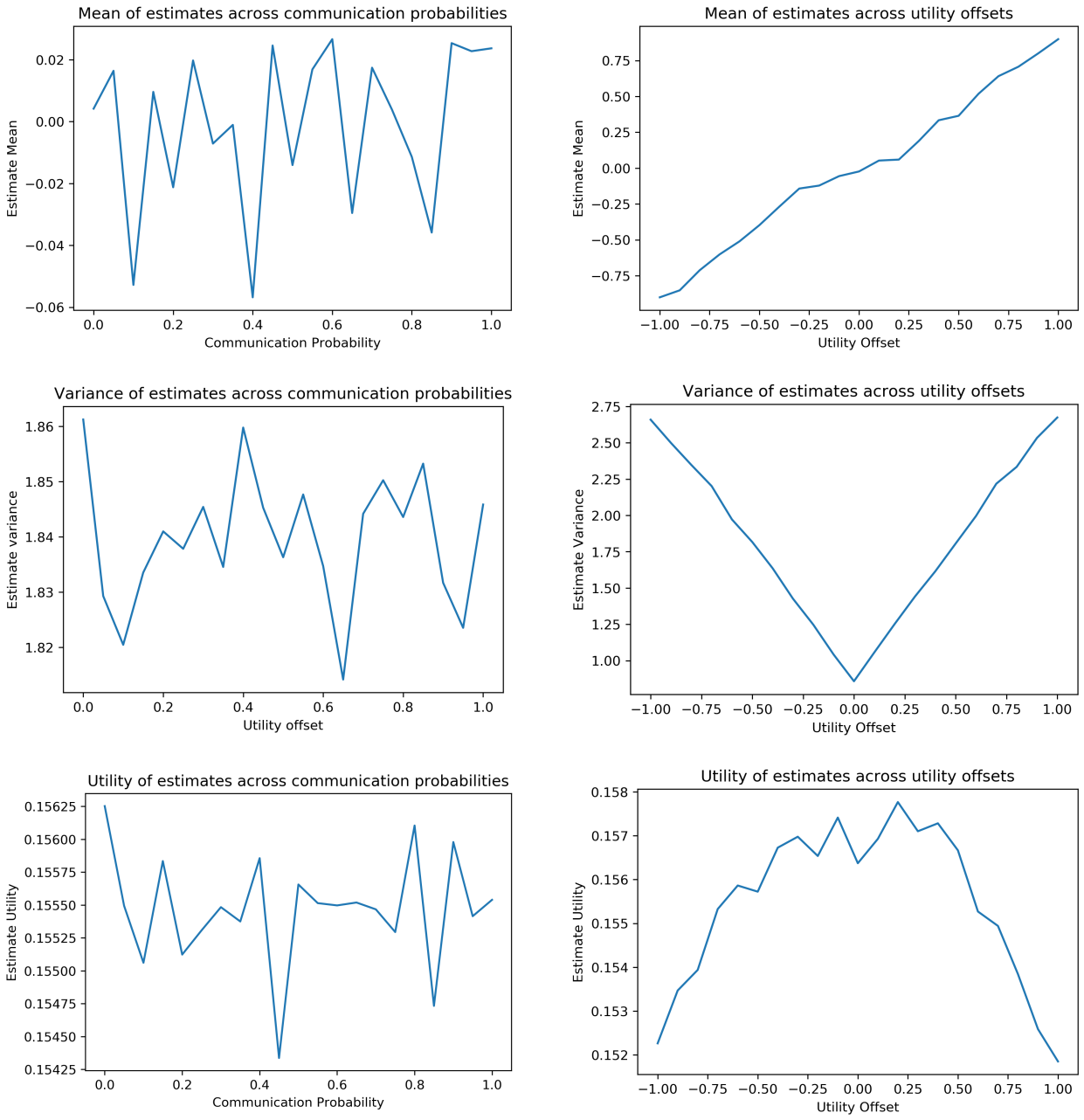


Figure 3: Graphs of averages across all variables for each parameter.

variation in a meaningful negative-parabolic form compared to the seemingly-random effects shown in the lower-left plot.

All of this is to say, there is little to no discernible evidence that variation in the types of estimates communicated between team members have substantive consequences for system performance. I'm skeptical of that finding because intuitively, if two people are talking about different designs as though they are talking about the *same* design, those designs are unlikely to interface with each other correctly. There is some, albeit not extensive, evidence in systems engineering to indicated that the interfaces between subsystems and to the overarching system are the most vulnerable for system failures. That would suggest that communication should play a more significant role than is shown here. Otherwise, perhaps the randomness itself shown in many of the figures is noteworthy, though the magnitudes are still very small.

Therefore, future work should re-examine the model assumptions, construction, parameter sweeps, and schedule to identify flaws in the reasoning, etc. However, it is also possible that this type of communication difference isn't substantive in and of itself. Perhaps another avenue would be to explore other types of communication in these systems which could lead to system variation. Finally, this network was ultimately smaller than what I wanted to model. I originally hoped for a tree network with a height of 4-6, but the computational challenges quickly became too large because of the number of agents. Nevertheless, real systems often have that type of depth even if with smaller out degree from each superior entity to subsidiary entities. I suspect that introducing a greater degree of network effects in scale and reporting structure may increase the degree of variation seen in system performance outcomes.

# References


[1] Fernando Anjos and Ray Reagans. Commitment, Learning, and Alliance Performance: A Formal Analysis Using an Agent-Based Network Formation Model. *J. Math. Sociol.*, 37(1):1–23, jan 2013.

[2] Jesse Austin-Breneman, Bo Yang Yu, and Maria C. Yang. Biased Information Passing Between Subsystems Over Time in Complex System Design. *J. Mech. Des.*, 138(1):011101, 2015.

[3] Christina L. Bloebaum and Anna-Maria Rivas McGowan. The Design of Large-Scale Complex Engineered Systems: Present Challenges and Future Promise. In *AIAA Aviat. Technol. Integr. Oper. Conf. 14th AIAA/ISSM*, number September, pages 1–19, Reston, Virigina, sep 2012. American Institute of Aeronautics and Astronautics.

[4] Christian Borgs, Jennifer Chayes, Brian Karrer, Brendan Meeder, R. Ravi, Ray Reagans, and Amin Sayedi. Game-Theoretic Models of Information Overload in Social Networks. pages 146–161. Springer, Berlin, Heidelberg, dec 2010.

[5] John O. Clark. System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective. *2009 3rd Annu. IEEE Syst. Conf.*, pages 381–387, mar 2009.

[6] Paul Egan, Jonathan Cagan, Christian Schunn, and Philip LeDuc. Synergistic human-agent methods for deriving effective search strategies: the case of nanoscale design. *Res. Eng. Des.*, 26(2):145–169, apr 2015.

[7] Kevin; Forsberg and Harold Mooz. The Relationship of System Engineering to the Project Cycle. *12th INTERNET World Congr. Proj. Manag.*, 1(June 1994):12, oct 1994.

[8] X. Gu, J.E. Renaud, S.M. Batill, R.M. Brach, and A.S. Budhiraja. Worst case propagated uncertainty of multidisciplinary systems in robust design optimization. *Struct. Multidiscip. Optim.*, 20(3):190–213, nov 2000.

[9] Jon C. Helton. Quantification of margins and uncertainties: Conceptual and computational basis. *Reliab. Eng. Syst. Saf.*, 96(9):976–1013, 2011.

[10] Charles Keating, Ralph Rogers, Resit Unal, David Dryer, Andres Sousa-Poza, Robert Safford, William Peterson, and Ghaith Rabadi. *System of systems engineering*, volume 36. 2008.

[11] Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Lifting the Veil: Drawing insights about design teams from a cognitively-inspired computational model. *Des. Stud.*, 40:119–142, 2015.

[12] Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Drawing Inspiration From Human Design Teams For Better Search And Optimization: The Heterogeneous Simulated Annealing Teams Algorithm. *Submitt. to J. Mech. Des.*, 138(April):1–6, mar 2016.

[13] Anna-maria Rivas McGowan. *Interdisciplinary Interactions During R & D and Early Design of Large Engineered Systems*. PhD thesis, 2014.

[14] Anna-Maria Rivas McGowan, Shanna Daly, Wayne Baker, Panos Papalambros, and Colleen Seifert. A Socio-Technical Perspective on Interdisciplinary Interactions During the Development of Complex Engineered Systems. *Procedia Comput. Sci.*, 16:1142–1151, 2013.

[15] John Meluso and Jesse Austin-Breneman. Gaming the System: An Agent-Based Model of Estimation Strategies and Their Effects on System Performance. In *Proc. ASME 2017 Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, number 58127, page V02AT03A050, Cleveland, Ohio, 2017.

[16] M E J Newman. Networks: an introduction, 2010.

[17] Ray Reagans. Negative Knowledge Transfer and Learning: Too Little Search or Too Much Noise.

[18] Kari Sentz and Scott Ferson. Probabilistic bounding analysis in the Quantification of Margins and Uncertainties. *Reliab. Eng. Syst. Saf.*, 96(9):1126–1136, 2011.

[19] Takeichiro Takamatsu, Iori Hashimoto, and Hiromu Ohno. Optimal Design of a Large Complex System from the Viewpoint of Sensitivity Analysis. *Ind. Eng. Chem. Process Des. Dev.*, 9(3):368–379, jul 1970.

[20] Daniel P. Thunnisen. Method for Determining Margins in Conceptual Design. *J. Spacecr. Rockets*, 41(1):85–92, jan 2004.

[21] Yiwen Zhong, Jing Ning, and Hui Zhang. Multi-agent simulated annealing algorithm based on particle swarm optimisation algorithm. *Int. J. Comput. Appl. Technol.*, 43(4):335–342, 2012.

[22] Yiwen Zhong, Lijin Wang, Changying Wang, and Hui Zhang. Multi-agent simulated annealing algorithm based on differential evolution algorithm. *Int. J. Bio-Inspired Comput.*, 4(4):217–228, 2012.