The following paper will be published and presented at the 5th Annual IEEE Systems Conference in Montreal, Canada, 4-7 April, 2011.

System Architecture Pliability and Trading Operations in Tradespace Exploration

Brian Mekdeci, Adam M. Ross, Donna H. Rhodes and Daniel Hastings
Systems Engineering Advancement Research Initiative (SEAri)
Massachusetts Institute of Technology
77 Massachusetts Avenue, Building E38-572
Cambridge, MA 02139
http://seari.mit.edu

***Abstract*-The concept of operations is often assumed when assessing different design variables in a tradespace study for a particular system architecture, The way a system operates, however, has a large effect on its performance, and can often be the only variable through which stakeholders can influence a system after the system is implemented. The concept of pliable system architectures is introduced so that operational variables can be explicitly considered and incorporated into tradespace studies. System transitions can be predicted by pliability, and these transitions can provide insight into other system "ilities" such as changeability, adaptability, flexibility and survivability. Two techniques are introduced in order to demonstrate the usefulness of the pliability concept; (1) a step-by-step process by which operational variables can be identified within a system architecture, and (2) a process by which very large tradespaces can be sampled into a manageable set of system instances that provide maximum insight for the level of effort to model them. As these new concepts and methodologies are new and part of ongoing research, they will need to be tested and validated in future work.**

Keywords-tradespace exploration; system architecture; operational variables; concept of operations; pliability; systems design

I. INTRODUCTION

Over the last decade, tradespace exploration techniques such as Multi-Attribute Tradespace Exploration (MATE) and Responsive Systems Comparison (RSC) have been developed to assist decision makers during early concept phases, in generating and assessing different system designs [1-4]. In these studies, systems were parameterized into different physical design variables, such as wing span or shield thickness, and then models were developed that assessed these different designs in terms of attributes, which decision makers consider for determining overall system utility. By using parameterized models and simulations, a tradespace of design alternatives can be explored, where decision makers can compare the utilities against the costs of various system designs [5]. Emphasis on selecting an appropriate physical design as early in the lifecycle as possible is a reasonable approach since it is often very costly, even prohibitively so, to make physical design changes to a system after it has been implemented.

In addition to selecting physical design alternatives, system designers have choices available regarding the operation of all but the simplest systems. Research has shown that operational choices can have a significant impact on the overall utility of a system. For example, operational changes to the observing schedule of the Terrestrial Planet Finder satellite mission resulted in a significant impact on utility [6], as did changes in the train frequency of an airport shuttle transportation system [7]. Thus, it is important to consider operational variables as well as physical design variables when enumerating a possible system solution set. *Operational variables* are choices available to operators, system designers, system architects or decision makers in how the system is operated. Similar to physical design variables, operational variables can be discrete choices from enumerated sets, or continuous variables. They are particularly important for systems of systems (SoS), and systems with long lifecycles. In many SoS, the SoS "designers" do not have the ability to alter the physical designs of the components of the SoS, which are often systems in their own right and under the control of other stakeholders. In many cases, the only way to influence the performance of the overall system is either though operating the system themselves, or by setting policies, standards and directives. As a system's lifecycle increases, so too does the likelihood that the context in which the system operates changes as well, making cost-effective and timely operational modifications even more important. Ideally, these operational changes will be made by the appropriate stakeholders to take advantage of a new opportunity that increases the value the system delivers. In many cases, however, operational changes are necessary to minimize decline in value caused by disturbances in contexts that arise.

Although available operational variables are a very important aspect of a system design, they are often overlooked, for several reasons. First, operational variables tend not to be as salient as physical design variables. Often, physical design variables are the result of component availability. When choosing the payload for an unmanned vehicle, for example, a system designer will often select from a set of existing (or upcoming) cameras that satisfy the physical (e.g. height, weight, connections) and capability (e.g. image resolution, color accuracy) requirements of the system. For operational variables, there are often an uncountable number of alternatives. For example, to get from point A to point B, an aircraft can fly an infinite number of alternative paths with various speeds, altitudes and routes. Another major reason

explicitly trading off operational variables is overlooked is because the implementation and impact of changes in operational variables tend to be difficult to assess. In order to compare one design against another, particularly in a tradespace exploration, it is necessary to be able to quantify the performance of various system designs. Typically, alternative physical designs can be assessed through simple parameter changes in existing physical models. Operational changes, however, tend to not follow known models, unless they are those that fall within the range of problems that have mathematical models, such as queuing theory and linear programming, found in Operations Research (OR) literature. Operations Research is an interdisciplinary branch of industrial engineering that uses mathematical methods to provide analysts with a quantitative basis for making decisions under their operational control [8]. Since quantitative methods dominate OR research, OR is typically concerned with finding the optimal solution to a given problem [9]. However, before these quantitative models can be used to assess the system, some fairly strict assumptions must be made about how the system operates. For example, the manner in which customers arrive and line up at a bank is an important assumption made prior to the quantitative analysis that determines the optimal number of bank tellers. For many complex systems, the assumptions required for analytical solutions are too strict and numerical simulation is necessary. Ideally, all of the operational as well as physical design variables would be enumerated, evaluated and included in a tradespace exploration. Unfortunately, every simulation takes a finite time to execute, so very large (often infinite) numbers of variables are impossible to consider. Instead, a finite set of plausible alternatives should be selected for comparison. Since many physical design differences can be captured as parameter changes while operational differences, such as changing the roles and responsibilities between various components, often require more substantial changes to simulations, it is not surprising that physical design alternatives dominate typical tradespace explorations. Of course, not all physical design alternatives are simply parameter differences, and some require substantial changes to the models and simulations. Therefore, a tradeoff exists between limiting the number of physical and operational alternatives under consideration, and modeling effort.

There are three goals for this paper. First, the concept of pliable system architectures (SA) is introduced as a way of incorporating operational variables into holistic system solutions that can be used in tradespace exploration studies. Next, an approach is introduced for sampling the large tradespace of possible system alternatives so that a tractable set of system solutions can be assessed. Finally, a step-by-step process is proposed where operational variables can be identified, to generate system architectures that will provide value under a variety of contexts.

II. System Architecture

The Department of Defense defines system architecture as a framework or structure that portrays relationships among all the elements of the system [10]. Crawley defines system architecture as the description of the components of a system and the relationships between them [11]. In this sense, the system architecture includes both what the system is composed of, and how these components work together (see Figure 1). At a minimum, the system architecture should describe the functional behavior of the system, i.e. which tasks the components perform, as well as when and how the tasks are executed, similar to the Capability and Operational Views found in the Department of Defense Architectural Framework 2.0 [12]. For simple systems, emergent behavior of the system is likely to be predictable from the system architecture, through aggregating the functional behaviors of the components. For more complex systems, emergent behavior is more difficult to predict. Engineered systems are designed for a purpose and thought must be given about how the system will operate as a whole to achieve that purpose, including how each component will function independently and together, as well as how the system interacts with its environment. A system architecture should be designed, as much as possible, such that its emergent behavior helps achieve value for its stakeholders.

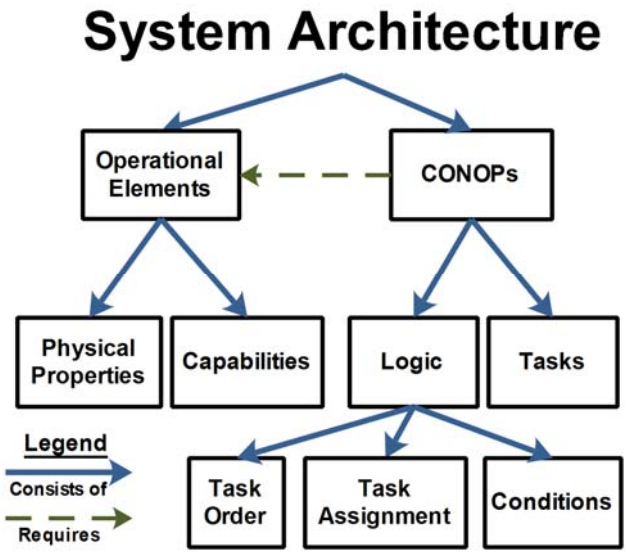

Figure 1: System architecture as operational elements with CONOPs

For the purposes of this paper, a *system architecture* consists of two core elements: (1) what the system is composed of, known as the *operational elements (OEs)*, and (2) how the system operates, known as its *Concept of Operations (CONOPs)*.

A. Operational Elements

The *operational elements* (OEs) describe what the system architecture is composed of. The OEs includes a description of all the components that (could) perform some function in the system. OEs are defined by their *properties* and *capabilities*. Properties are the physical characteristics of the operational elements, such as length and mass. The capabilities describe the functional behavior that the components can perform, such as fly, transmit data, or process video. The capabilities are what the OEs can do, but not necessarily what they actually do do (unused capabilities are latent capability, representing potential value in the system architecture). For example, a football player may have a good voice, but does not sing as part

of his job playing football, however this capability may provide value at a team fund-raising concert. The *form* is an instance of the OEs.

B. Concept of Operations

The *Concept of Operations* (CONOPs) describes what the system does. Specifically, the CONOPs describes the *tasks* that the OEs perform, as well as the *logic* for how the OEs perform these tasks, within a particular context. While some definitions of CONOPs assume that operators are external to the system, this definition includes operators as part of the system (in fact, they are considered OEs). Thus, the actions the operators perform within the system are part of the CONOPs, as well as the actions of other OEs.

1) Tasks: All systems do something and that something can be referred to as tasks. Tasks can be broadly defined, such as "perform surveillance" or they can be specific, such as "accelerate to 93.7 mph". Typically, broadly defined tasks are broken up into a set of smaller, specific sub-tasks. Each of these sub-tasks can then be further broken down into more specific sub-tasks. Typically, broader tasks have more OEs involved in performing those tasks.

2) Logic: Merely specifying a set of tasks will not be sufficient to describe the manner by which the system achieves its purpose. Additional information, in the form of logic, must be specified.

a) Order: The tasks must be given an order in which they are performed. Order does not have to be strict, as some tasks may be performed concurrently or even randomly.

b) Task Assignment: The logic of the CONOPs specifies which OEs perform which task. Sometimes the output from one OE becomes the input to another OE. Specifying which OE performs a task is especially important when multiple OEs share the same capabilities, such as one might expect in a complex SoS.

c) Conditions: Many tasks have conditions associated with their execution. Conditions may exist to determine whether a task gets performed at all. Conditions may also specify how a task gets executed. For example, if a hostile, anti-aircraft gun is present in the area, a UAV will fly at a higher altitude than it normally would. Other common conditions include starting and stopping conditions for tasks that are not being performed continuously.

III. Pliability in System Architecture

A system is an instance of a system architecture. For example, a Nimitz-class aircraft carrier is the system architecture of which both the USS Ronald Regan and USS George Washington carriers are instances. There are differences between these carriers, however, so they are not the same system, but rather they have the same system architecture. While both carriers have similar crews, in terms of the number of officers and their relationships, the actual crew members onboard the USS Ronald Reagan and the USS George Washington are different. They have different properties (e.g., height, weight, IQ), which can lead to different performances, even under identical conditions. Although both carriers will follow the same basic CONOPs for a particular mission as specified by the US Navy, small deviations, likely due to differences in the captain or crew, will occur. The *mode of operation* is the specific instance of CONOPs that a particular system is executing. Even if both systems were identical in form and mode of operation, they would not be the same system, just as two identical twins are not the same person. However, two systems with the exact same form and mode of operation are said to be *equivalent*.

A. Parameters and Architecture Pliability

Since the system architecture *SA* of a particular system describes exactly how the OEs work together to perform tasks, the performance of a system for a particular context *C* can be determined by modeling and/or simulating the system architecture for a specific contexts. The minimally acceptable value threshold of the system $\hat{V}$ is assessed by stakeholders, according to their needs, perhaps varying by context. In a perfectly value robust system, the context-embedded value V_C that the system generates is greater than or equal to the acceptable value threshold $\hat{V}_C$ determined by the stakeholders for any particular context *C*. In other words,

$$V_C \geq \hat{V}_C \; for \; any \; context \; C \qquad (1)$$

However, very few (if any) systems can provide acceptable value under all contexts, and many systems will only provide acceptable value under a small range of conditions. In many cases, as the environment or stakeholder needs change over time, systems require some modifications to their form and/or mode of operation accordingly. System architectures can be designed to allow variations in system form and/or mode of operation, by allowing system parameters to describe the specific instance of system architecture and the functions it performs. A parameter is a quantitative means for representing choices in a physical property, capability, task order, task assignment or condition of a system. For example, the wingspan is a physical property of an airplane, which is an OE included in the form of some air traffic control system architecture. The fact that the wingspan of a specific Predator UAV happens to be 12 ft, is a physical parameter and the fact that the UAV flies at a cruising speed of 98 mph is an operational parameter. All parameters of a system architecture are either *fixed* or *pliable.* Fixed parameters remain constant for all instances of that system architecture and for all time. The fact that the flight deck on a Nimitz-class aircraft carrier is angled at nine degrees is fixed, allowing for simultaneous takeoffs and landings. Parameters of a system architecture that are allowed to change, either between alternative systems, or within the same system over time, are known as pliable parameters. The *pliability* of a system architecture is how much a system can change in operational elements and CONOPs, while still adhering to the same system architecture. Nimitz aircraft carriers are built with two A4W nuclear reactors, but the architecture is designed to operate with just one if necessary. In this sense, the number of A4W nuclear reactors is pliable, although the number can only vary between

one and two. This range of acceptable values is called the *pliable range*. Pliability is measured both by the number of pliable parameters and the magnitude of each parameter's valid range. In some cases a parameter "level" may be a natural integer, such as the number of reactors in a carrier, it may be a continuous variable, such as the cruising speed of an airplane, or it may be a choice from a set of options, such as whether operators handle tasks in a first-come, first-served manner or highest priority first.

B. Defining Pliability of Parameters

Whether a parameter is pliable or not, depends upon whether or not it is *allowed* to change (as defined by the system architecture). All parameters can change, voluntarily or involuntarily. For example, a Nimitz-class aircraft carrier can suddenly be without a flight deck because it was destroyed in an attack. In many circumstances, this change will be a violation of the system architecture because the system was designed to allow aircraft to take off and land, and now the system cannot perform that function. Hence, the value being delivered by this damaged carrier is significantly reduced, and will likely fall below the required minimum threshold for many scenarios. The final decision about what parameter changes are allowable is up to the system architects, but there is risk in allowing systems to change to instances that provide little or no value to most, if not all, considered contexts. A system architect might not be able to anticipate every possible context a system may have to operate in, and certain instances that underperform in known contexts may prove to be invaluable in unknown contexts. However, there may be a risk in allowing these architecture instances, making it easier for external agents to force a system transition in the wrong context. Additionally, there should be a realistic, cost-effective way for the system to change from one instance of a particular system architecture, to another. The intent of designing a good system architecture is to minimize the cost of making changes down the road in the event of context changes. Although in theory a cellphone could turn into a UAV if enough changes were made to the form and CONOPs, there exist no realistic, cost-effective mechanism for transitioning from one of these systems into the other, and thus such a change should be seen as one from one instance of a particular system architecture into one instance of a different system architecture (see Figure 2).

C. Architecture vs. Instance Transitions

A system, a_1, is an instance of a particular system architecture, *A*. If a system a_1 changes in ways allowed by the pliability of the system architecture, then the system takes on a different instance of that architecture, i.e., $a_1 \rightarrow a_2$. A system's ability to make this type of transition is referred to as *system changeability*. If the system a_1 changes in ways not allowed by the system architecture, then that system has become a new instance, b_1 of a different system architecture, *B* (defined or undefined) (see Figure 3). A system has an *adaptable architecture*, if an internal agent can deliberately alter the system's architecture. A system has a *flexible architecture* if an external agent can do the same. An important distinction thus raised is that there are two "levels" of changeability that need to be considered: changes at the system architecture level, and changes at the system instance level. All things being equal, changes at the system instance level are likely to be less costly than changes at the system architecture level.

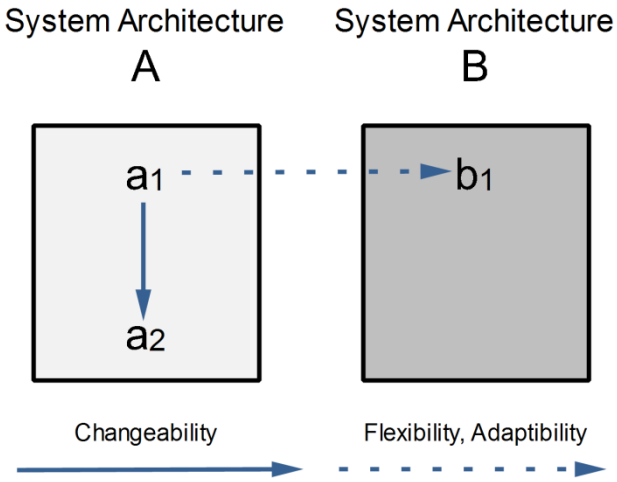

Figure 2: Architecture Transitions

D. Architecture Transitions and Survivability

Survivability is defined as the ability of a system to produce value before, during and after some finite duration disturbance [4]. Survivability is achieved through three mechanisms: (1) avoiding the disturbance before it happens, (2) mitigating the damage caused by the disturbance as it occurs, and (3) recovering to some acceptable value after the disturbance is over [13]. Disturbances may break the architecture of a particular system by altering one of the fixed parameters, or by forcing a parameter to go outside its pliable range. Since the system was not designed to operate outside of its architecture, there is a risk that the disturbance will not be survivable if the new, architecture non-conforming instance is not viable. In this situation, the system needs to either return to its original architecture through repair or replacement, or adapt to a new, viable system architecture. In other cases, disturbances may not break a system architecture, but instead change the context in such a way that the system will not be survivable unless it adapts to a new, viable system architecture (see Table I).

TABLE I. RECOMMENDED ACTIONS FOR DEALING WITH DISTURBANCES

Effect of Disturbance	Recommended Action
Changes a system's parameters that violate the pliability of SA	Repair system to original SA, validate new SA, or change to an alternative, viable SA
Changes context so current SA not producing value	Change SA
Change context so current system not producing value	Change system instance

IV. Generating System Architectures

Variations in quantifiable design and operational parameters, such as engine horsepower or flight speed, are often straightforward to specify in tradespace explorations. Variations in CONOPs, on the other hand, tend to be more difficult. In order to do a tradespace exploration, the choice of CONOPs must be defined in order to evaluate performance. Historically such choices have often been made implicitly through design evaluation model development, but in order to trade-off CONOPs, CONOPs must be explicitly stated. Thus, a method for identifying, enumerating and evaluating CONOPs is needed in order to determine the impact of varying CONOPs on system performance as well as on temporal properties, such as survivability.

A. Steps to Identifying CONOPs Variables

There are several steps for specifying CONOPs variables. As an illustrative example, consider a maritime surveillance scenario that involves three UAVs in charge of patrolling a particular Area of Interest (AOI).

1) Identify the Context: The first step is to identify the context under which the system is operating. This includes any relevant key exogenous factors that affect the success of the system, such as geographic details, weather, and other entities entering the AOI.

2) Identify the Mission Objectives and Attributes: Since any system under consideration is being designed for a purpose, it is necessary to state that purpose and the metrics, or stakeholder attributes, by which performance of the system is assessed. For the UAV system performing maritime surveillance, the mission objective is "to visually identify boats as they enter an area of interest." The attributes that the stakeholders will use to assess value will be *time-to-identification* (how long it takes the UAVs to identify a target once it enters the body of water) and *percentage of targets missed* (the total number of boats not identified divided by the total number of boats that entered the area of interest). A high performing system will be one that minimizes one or both of these attributes. In addition to performance, cost is always something to be minimized for any real system under consideration.

3) Specify the factors that directly affect the attributes: Several factors affect the performance of the system and it will be necessary to identify them to guide variables in the form and mode of operation. In order for a target to be identified, the target must be within the field of view (FOV) of a camera onboard a UAV and the UAV operator at the ground control station must spend a finite amount of time reviewing the visual information to make the identification. A camera has a limited FOV, which means there is a finite area that can be monitored at any moment in time. The *time-to-identification* and *percentage of targets missed* will be directly proportional to the AOI and inversely proportional to the FOV. If the geographic AOI is larger than the combined FOVs of the cameras onboard the UAVs, then the cameras will have to move their FOV to cover the entire AOI. To simplify this example, the assumption that "the camera does not pan" is made. Therefore, the only way the FOVs move across the AOI, is due to the movement of the UAVs themselves. Thus, the attributes will be inversely proportional to the speed of the UAV. The FOV is primarily a property of the camera type, the desired resolution of the photographs and the altitude of the UAV. Typically, the higher the desired resolution and the lower the UAV flies, the smaller the FOV.

4) Identify the System Form: In this step, the components of the system and their functions are specified, relative to the factors identified in Step 2 and drivers specified in Step 3. For the UAV example, the system form consists of the aircraft, the camera payload, and the operators. The aircraft flies the camera around the AOI, the camera provides video of targets within its FOV and the operator controls the UAV as well as identifies targets within the UAV's FOV.

5) Identify the CONOPs: In this step, the method by which the system achieves its mission objectives is specified. The system is described by the form, so the mode of operation should only reference components and functions that are part of the specified form. If there is a discrepancy between the form and the mode of operation, then perhaps one or both of them should be reconsidered.

Research is ongoing for defining a set of rules for specifying modes of operation. The preliminary rules are:

1. Start with one task that accomplishes each system purpose or objective.
2. Specify the logic surrounding the task:
   - When does that task get performed? i.e., what are the conditions necessary to perform that task (Order)?
   - Which components perform that task? (Assignment)
   - List *how* the task gets performed, i.e. under what conditions is the task performed? (Conditions)
   - What are the sub-tasks necessary to describe how the task works? (Tasks)
   - List the operational parameters (levels) for the various functions and conditions. (Operational Parameters)
3. Repeat Step 2 for each sub-task, breaking them down until only elementary sub-tasks remain. An elementary sub-task, is a basic task that does not need to be broken down into sub-tasks in order to be able to assess its impact on the attributes. Walking, for example, can be considered an elementary task, since for most systems, specifying that a person is walking at a particular speed and in a particular direction (the operational parameters for walking) is all the information needed to access its impact on attributes such as time-to-target. Breaking down the task of walking into sub-tasks, such as placing one foot in front of the other, is typically not necessary unless more detailed models and analysis are needed.

6) Look for Options in the Mode of Operation. In this step, each task, logic statement, and operational parameter is examined to determine if there are any options (i.e., alternative viable choices). Examples of questions designers should be asking include: Is there another, reasonable way to accomplish the same task? Can the system form be something else and the CONOPs will still work? Can a different set of sub-tasks, accomplish the same high-level task? Can different components perform the tasks instead? If the answer is "yes" to any of the above questions, then alternatives become variables that should be modeled and included in a tradespace exploration.

B. Illustrative Example: Maritime Security using UAVs

In order to illustrate the differences between these system constructs, consider a maritime security system composed of various unmanned aerial vehicles (UAVs) where there are two different types of payloads (electro-optical and synthetic aperture radar) and two different methods of communication between the ground control station and UAV (line-of-sight and satellite relay). Line-of-sight (LOS) typically allows higher data bandwidth than satellite relays, at the expense of range. Additionally, operators can choose the altitude at which the UAVs fly. Flying at high altitudes makes the UAV less susceptible to enemy detection, but results in poorer quality video surveillance. The number of UAVs performing the mission can be altered, as well as whether target detection is done automatically by the UAVs or manually by the human operators. Automatic target detection is faster, but at the expense of accuracy. Operators at the ground control stations can specialize and control a single type of unmanned vehicle (known as a mechanistic team) or they can diversify and control all types (known as an organic team). Typically, mechanistic teams perform better when the workload is evenly distributed, but when the task load is erratic, organic teams can be more effective [14].

The environment, in which this system operates, consists of an AOI that is 50 miles x 50 miles. Targets are randomly distributed within the AOI according to a spatial Poisson process, spaced five miles apart (on average). The form consists of three fixed-wing UAVs and a base located within the AOI at a particular location. The purpose of the system is to provide maritime surveillance. The attribute of interest is time-to-target, or the amount of time it takes the UAV to start video footage of a target, after such footage has been requested by a friendly entity outside the system. The UAV can only detect entities within 20 miles of itself, so the time-to-target attribute is determined by measuring how long it takes the UAV to be within 20 miles of the target, once the target has been requested.

At the highest level, the main task of the system is to provide maritime surveillance. Since the system is always expected to provide maritime surveillance, there are no conditions, i.e., the system is always supposed to be performing this task. This task is very broad and thus, there are no operational parameters to be specified at this level. Likewise, it is impossible to assess the time-to-target with a CONOPs at this broad level of specificity. Thus, this task needs to be broken down further. The next highest level of abstraction would be to say that there are five sub-tasks; (1) take off from base, (2) fly to the AOI, (3) loiter in the AOI, (4) take pictures of targets, and (5) return to base. Here, there are some conditions that need to be specified, such as when does the plane take off or when does the plane take a picture of a target? Also, there are some operational parameters that need to be specified, such as the altitude at which the UAV flies (see Table II). Even with these conditions and operational parameters set, it would be impossible to be able to determine the time-to-target, since the time-to-target requires knowledge of exactly where the UAV is located at a particular time and this cannot be determined with the CONOPs specified. Thus, the tasks need to be broken down further, particularly the loiter task. Here, the designer will discover several options. What flight path does the fixed wing take, e.g., racetrack or figure-8? Assuming that once targets are identified they are "removed" from the AOI, should the UAV dynamically try to anticipate where the next target is going to be based on where they have been in the past? How is the AOI divided up among the UAVs? Do they search their own sub-areas or share the entire AOI? These are all variables that can be explored by specifying different CONOPs.

TABLE II. EXAMPLES OF PARAMETERS FOR UAV SYSTEM

Parameters	Examples
Physical Properties	Weight of UAV, maximum altitude of UAV, camera FOV, number of UAVs
Capabilities	Fly, take picture, communicate with ground control station, communicate with UAVs
Task Order	1. Take off from base 2. Fly to AOI 3. Loiter in AOI 4. Take pictures in AOI 5. Transmit pictures to ground control station 6. Return to base
Task Assignment	UAV1 → Take picture UAV2 → Relay communications between UAV1 and ground control station
Conditions	If enemy is present, fly at 15,000 ft, else fly at 8,000 ft

Once the CONOPs has been specified to a level at which it can be modeled by some quantitative technique such as queuing theory or computer simulation, then it is at the right level of specificity and should not be broken down any further. At this stage, the system instances can be assessed and incorporated into a tradespace. However, the process of specifying the CONOPs will force the designer to consider many different questions. In this example, the designer is forced to consider how many UAVs would be necessary for this AOI and whether or not a Vertical Take-off and Landing (VTOL) UAV would be more suitable than a fixed-wing UAV, since a VTOL can hover. Does the fact that the UAVs have to keep flying somewhere at a minimum speed affect their ability to respond to targets quickly? At this stage, the designer may want to change either the form of the system (e.g. adding or removing UAVs, changing them to VTOLs) or CONOPs (changing the flight path). This process of discovery highlights

the iterative nature of system design, and may generate new instances of a specified system architecture, or even suggest new system architectures for evaluation.

V. TRADESPACE SAMPLING

As previously mentioned, the range of possible ways a system can be architected, from physical design alternatives to different modes of operation, is typically infinite. Since simulation is usually the only way in which these alternatives can be evaluated, the full range of possible system solutions must be sampled down into a finite set worthy of consideration. The concept of pliable system architectures provides designers with an approach for tradespace sampling. When designing a system, the architect should be able to confidently attest to the fact that the system will provide adequate value, according to stakeholder needs, for a particular context. The architect can do this by fixing all of the parameters in a proposed architecture and evaluating the system instance of this architecture for the specified conditions. Once satisfied, the architect can consider other possible contexts or disturbances and evaluate the system instance. If the system fails to provide adequate value, then the architect can look at the system architecture for parameters that could change through some reasonable, cost-effective mechanism and begin to make them pliable. The designer will set limits and evaluate system instances at those limits. Depending on the results, the architect can expand the limits, reduce the limits or make other parameters pliable as well. At some point, the proposed architecture will likely reach a pliability maximum, where "reasonable" changes do not result in system instances with the necessary performance required. At this point, "unreasonable" changes can be made, that is, the architect can consider different system architectures altogether and begin the iterative process of discovering appropriate pliable ranges again. New system architectures will typically open up further, reasonable parameter changes and the tradespace exploration continues.

Since there will typically be a considerable amount of pliability in a specific system architecture and different system architectures vary in large ways (i.e. non-trivial changes), it is expected that performance will be more correlated to system architecture than to context conditions. Similarly, shows a hypothetical graph where changes in system architecture (represented by different colors) have a large effect on cost and utility than changes in parameters (represented as point design 'dots'). This hypothesis can be easily tested and if shown, would further emphasis the need for architecture exploration in early phase design.

VI. NEXT STEPS

The concepts presented in this paper are still preliminary and need to be tested. The next steps in the research will be to begin validating the concepts introduced in this paper using a maritime security system of unmanned systems. First, the methodology for generating system architectures (section IV) will be performed for several different maritime security scenarios. Then, the maritime security scenarios and associated system architectures will be implemented in a discrete-event simulation. Point design instances of the initial system architecture will be evaluated for each scenario, by creating, testing and expanding parameter pliability (section V). As new point designs, generated by parameter changes, fail to increase performance or decrease cost, new system architectures will be generated by analyzing the simulation logs to determine areas of improvement. Then, the new system architectures are explored by making parameter changes and evaluating point designs, until no obvious improvements in system architecture can be seen. From these tradespaces, correlations between pliability and performance for these scenarios and system architectures can be determined, to help identify the relationship between system architecture, pliability, performance and cost. Additionally, some of the scenarios used to test the methodologies will include disturbances and thus, will allow researchers to explore the relationship between pliability of system architecture and survivability of systems. Some of the questions to be answered include: does increasing the pliability of a system architecture increase its survivability as well? If so, what is the relationship between cost, survivability and performance? When is it better for a system to change within its architecture and when is it better for a new architecture? Of course, answering these questions assume that the methodologies in section IV and V were successful in generating CONOPs and evaluating system architectures. Future work should also look at ways of improving these methodologies, by applying them to other systems of interest and also by running studies where alternative methods of CONOPs generation and system architecture evaluations are performed as a comparison. Finally, as research in this area progresses, it is hoped that methods can be established that would allow system architects to design pliable architectures that easily allow transitions from one instance into a better performing instance as the context changes, but disallows transitions, due to disturbances or otherwise, that degrade system performance.

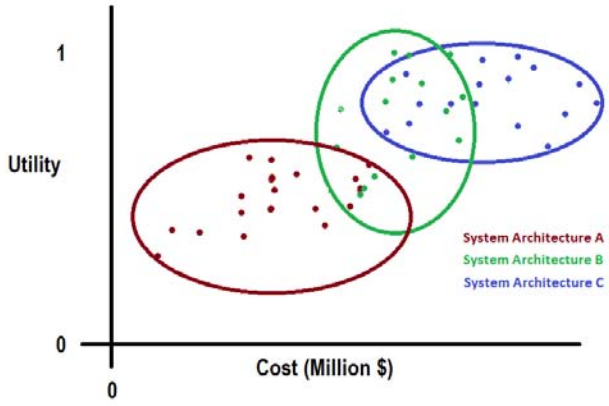

Figure 3: Hypothetical Correlation between System Architecture and Value

VII. CONCLUSIONS

Compared with physical designs, varying system operations is both an important consideration for system performance and an often neglected aspect of tradespace exploration studies. In this paper, the need for including operational variables in tradespace explorations was discussed and the concept of pliable system architectures was introduced. Pliable system architectures are those that allow some variation in form and/or mode of operation, while

maintaining an adequate level of performance. A preliminary methodology for decomposing a system into operational elements and a concept of operations, as well as generating pliable system architectures was presented. Finally, an iterative process for sampling the infinitely large tradespace of system solutions was introduced using the limits of pliability as design points of interest. The concepts and methodologies introduced in this paper are evolving through ongoing research, but hopefully will empower architects and analysts with an explicit approach for proposing and evaluating system architectures, as well as trading operations on par with trading design features. Future work includes validating and refining these methodologies with discrete event simulations of maritime security scenarios.

REFERENCES

[1] A.M. Ross and D.E. Hastings, "The Tradespace Exploration Paradigm," in INCOSE International Symposium, Rochester, NY, July 2005.

[2] A.M. Ross, H.L. McManus, D.H. Rhodes, and D.E. Hastings, "Revisiting the Tradespace Exploration Paradigm: Structuring the Exploration Process," AIAA Space 2010, Anaheim, CA, September 2010.

[3] A.M. Ross, H.L. McManus, et al., "Responsive Systems Comparison Method: Case Study in Assessing Future Designs in the Presence of Change," AIAA Space 2008, San Diego, CA, September 2008.

[4] M.G. Richards, "Multi-Attribute Tradespace Exploration for Survivability," Ph.D., Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA, 2009.

[5] A.M. Ross, D.E. Hastings, J.M. Warmkessel, and N.P. Diller, "Multi-Attribute Tradespace Exploration as Front End for Effective Space System Design". Journal of Spacecraft and Rockets, 2004, vol. 41 (1), pp. 20-28.

[6] A.M. Ross, "Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration", Ph.D., Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, 2006.

[7] J. Nickel, "Using Multi-Attribute Tradespace Exploration for the Architecting and Design of Transportation Systems", S.M., Massachusetts Institute of Technology, Cambridge, MA, 2010.

[8] P.M. Morse, "Methods of Operations Research", Wiley, New York, 1951.

[9] F. Hillier, G. Lieberman, et al., "Introduction to Operations Research," McGraw-Hill, New York, 1990.

[10] Department of Defense, "Dictionary of Military and Associated Terms", Joint Publication 1-02, 2010.

[11] E. Crawley, O. de Weck, et al., "The Influence of Architecture in Engineering Systems", MIT Engineering Systems Symposium., Cambridge, MA, 2004.

[12] Department of Defense Architectural Framework 2.0. http://cio-nii.defense.gov/sites/dodaf20/. Accessed on January 21, 2010.

[13] R. Westrum, "A Typology of Resilience Situations". Edited by E. Hollnagel, D.D. Woods and N. Leveson, Resilience engineering: concepts and precepts. Aldershot, England, 2006.

[14] B. Mekdeci, and M.L. Cummings, "Modeling Multiple Human Operators in the Supervisory Control of Heterogeneous Unmanned Vehicles. 9th Conference on Performance Metrics for Intelligent Systems, Gaithersburg, MD, 2009.