Third International Engineering Systems Symposium
CESUN 2012, Delft University of Technology, 18-20 June 2012

Controlling Change Within Complex Systems Through Pliability

Brian Mekdeci[1], Adam M. Ross[2], Donna H. Rhodes[3], and Daniel E. Hastings[4]

Massachusetts Institute of Technology, Cambridge, MA 02139, USA

mekdeci@mit.edu, adamross@mit.edu, rhodes@mit.edu, hastings@mit.edu

Abstract. *As systems become larger, more complex, and operate for longer periods of time, some change within the system often becomes inevitable. Particularly in systems of systems, with diverse stakeholders, evolutionary development and managerial independence, it is not unusual for constituent systems to change in form or the way they operate. Changeability, the ability of a system to change, is often considered to be a desirable attribute that allows systems to be robust and to adapt in response to changes in context. However, involuntary changes, such as those that occur as a result of a disturbance, are more often problematic than favorable. In some ways, the survivability of a system depends on its ability to prevent, mitigate and recover from unintentional changes within the system brought about by disturbances. For certain large systems of systems, where there are complex interactions and a diverse set of stakeholders, even voluntary changes may be frowned upon, since it may be an expensive and time consuming process to approve changes. This paper discusses pliability, a new "-ility" that places constraints on the changes a system is allowed to make. Pliability is the ability of a system to change, without "breaking" or violating an architecture that the system architects intended and validated. Like changeability, pliability increases robustness by allowing systems to voluntarily change in response to changing contexts, and increases survivability by increasing the likelihood that unintentional changes are still within the set of allowable instances. It also distinguishes allowable changes from those that would require validation and approval from decision makers, making it easier to actually implement those changes in large, complex systems.*

Keywords. *Pliability, value robustness, survivability, changeability, flexibility, agility, CONOPs, concept of operations, system architecture.*

1 Engineering for Change

"Nothing endures but change."

-Heraclites (c. 535 BC – 475 BC).

As systems are expected to do more, their complexity often increases as well. They become more expensive to design, test, build, launch, operate and maintain. With shrinking budgets, particularly in public projects, stakeholders expect large, complex systems to have long lifecycles and provide value in spite of changes in context. The longer a system exists, the more complex its behavior is, or the more dynamic the context in which it operates, the more likely *something* within the system will change. Perhaps there will be a change in components, either in the number and type of components, or their attributes and capabilities. This is particularly an issue with systems of systems (SoS), since they are often never fully formed, but rather go through evolutionary development as components are added, removed and changed over time (Maier, 1998). Even if the components themselves don't change, what they do within the overall system might change. Certain F-16s, acting as components of a larger military SoS, may change from air-to-air combat roles to air-to-surface attack roles. An emergency response dispatcher may process calls in a priority queuing manner (based on location or some other factor), instead of a first-in, first-out (FIFO) sequence. Of course, changes often do not occur in isolation. A change in components, will often require operational changes as well, and vice-versa. If the wireless LAN suddenly fails, then mobile system components may be forced to operate near an available Ethernet port. Similarly, a decision to operate certain components in a mobile fashion may require batteries or some other portable power source instead of an AC power supply.

There has been increasing research studying how, when and why systems need to change in response to shifts in context (McManus and Hastings, 2006). In particular, research has areas such as

- Designing latent capabilities that grant the ability to change at a later date (e.g., de Neufville and Scholtes, 2011)
- The ability to change during operational phases (e.g., Gupta and Goyal, 1989)
- The ability to change easily / rapidly (e.g., McGaughey, 1999)
- The ability to change in size only (e.g., Elkins et al., 2004)

The terms "flexible", "adaptable", "agile", and "modifiable" are some of the labels that have been used to describe the ability to change. Unfortunately, most of these terms are not consistently defined in the literature, and there is ambiguity as to their exact meaning when applied to systems engineering. This makes it very difficult for stakeholders to communicate the properties that they desire, and for architects to know that they've met those requirements. The ability to change, known as *changeability* (Ross et al., 2008), is an important quality of a system that allows stakeholders to reduce the impact of uncertainty of future contexts. However, not all changes are beneficial and care must be taken by architects to ensure that any modifications to the components or the way they operate, will not reduce value delivery to its stakeholders.

2 Problems with Change

While it may seem that the more a system can change, the better it will be able to respond to shifts in context, there are limits. In their minds, and sometimes in documents, system architects have a *concept of operations* (CONOPs)[1] that describes how the components within the system function and interact with each other to produce value to the stakeholders within the context of an operational environment (Mekdeci et al, 2011) . A change in one of the components, their capabilities, or in the way the components interact or function, may violate an aspect of the CONOPs, and cause a reduction in value delivery. It is important then to only change what should be changed, and not change what shouldn't. There are many types of change to consider, but they can be broadly categorized into intentional changes and unintentional changes.

Intentional Changes. Intentional changes are those that decision makers choose to execute, often in response to a shift in context or stakeholder needs. An example of an intentional change would be if airline engineers added AC power ports to all the seats in an existing aircraft to keep up with market demand. Intentional changes are typically the types of changes described by researchers when they discuss changeability, flexibility, agility and other types of change-related system properties. However, the concept of "intentional" changes should be clarified further than has traditionally been the case. Sometimes, decision makers may want to implement a change to improve value delivery, but end up reducing it instead. This is because as the system complexity grows, and as more changes are made to the system, it becomes increasingly difficult to verify and validate the effects that changes will have. Sometimes, the original architects and engineers who designed the system and really understood the CONOPs may no longer be available to evaluate impacts of planned changes. Subtle assumptions that were not made explicit, may be violated with any new changes and could prove to be not only disastrous, but difficult to find until it's too late. This is particularly a problem with systems of systems that have autonomous constituent systems with emergent behavior that is notoriously difficult to model. The larger the system, or the more expensive, or the more stakeholders involved, the likelihood of someone wanting to accept responsibility for any one particular change decreases. Thus, there may be a substantial bureaucratic process involved for any complex system that makes it extremely time-consuming and costly for all but trivial changes to be made. This "red tape" can seriously impair the ability of the system to respond quickly to changes in context, especially if it takes years to approve any significant changes.

Unintentional Changes. Unintentional changes are those that the system is "forced" to undergo, that is, changes that occur whether the decision makers want them to or not. Some unintentional changes just happen without any intervention or "approval" from the stakeholders. A collision with a bird may cause an engine to malfunction, or a power outage may cause a monitoring system to fail. Other unintentional changes happen after decision makers authorize them, but only because they have to, in response to some other event beyond their control. A labor strike may force a city's engineers to shut down the subway, or a new environmental regulation may cause a chemical plant to replace certain older equipment with newer, more environmentally-friendly models. Since unintended changes are likely to be problematic, system

[1] To some researchers and practitioners, the word "CONOPs" refers to *documentation* that describes how a system works, rather than the *concept* of how the system works itself.

architects typically try to minimize their causes and effects as much as possible. Unintentional changes that are the result of endogenous forces, such as random component failure, are the focus of *reliability* engineering (Leveson, 1995), whereas system *survivability* and *robustness* strive to prevent, mitigate and recover from unintended changes caused by exogenous forces such as lightning strikes and resource shortages.

3 Change in Tradespace Studies

Tradespace studies are often used to help decision makers assess and compare various system designs in the conceptual stage of a system's lifecycle (Stump et al., 2004). Typically, designs are modeled and simulated and their overall utility (as determined by stakeholder/decision maker preferences) is plotted against cost. For a system that does not change, the utility and cost is constant for a particular context (Figure 1). However, if the system can change, then for a given context, there can be a range of utility/cost that a system can achieve or change to (Figure 2). Several interesting questions arise. Suppose in Figure 2 that system B differs from system A only in that an extra component is added (providing additional value at additional cost). Are these systems just two different instances of a similar architecture? Should the changeability of systems A and B be represented somehow in the tradespace (Ross and Hastings, 2006)? When does a change cause a system to become another system? What defines a simple transition change from an evolution change?

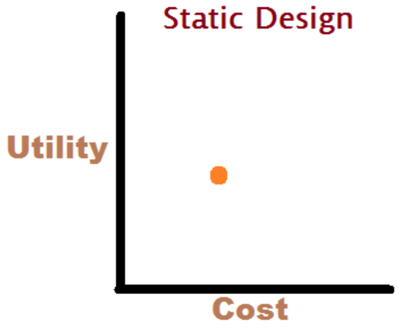

Fig. 1. Sample plot of static system.

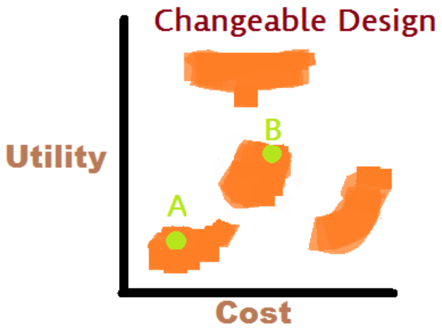

Fig. 2. Sample plot of changeable system.

Changes are not always as discrete and obvious as adding components. Sometimes small changes in the way a system operates could have large cost and performance consequences. As an example, if a 911 call center prioritizes calls based on location, it may have vastly different performance results that one whose mode of operation is to respond to calls in a first-in, first-out manner. Explicitly trading the modes of operation may result in multiple points in the tradespace for the same set of components. If the modes of operation are not traded, then system architects must recognize that if operators use the same system in a different way, it may appear to "move" in the tradespace. It is for this reason that system architects should explicitly consider an operations "envelope" around points to account for various modes of operation.

It is clear that systems may need to change in response to varying contexts and do change (intentionally or not). What is lacking is a system property that explicitly addresses the need to specify limits on what should be allowed to change, in order to ensure that any modifications of the system's form or mode of operation does not adversely affect its value delivery. The authors feel that a new "ility" is needed – one that recognizes the need for change, but also recognizes that there are limits to what should be allowed (i.e. there are "bad" changes that should be avoided or prevented). The term we use for this new ility, *pliability*, has not been widely used in the literature before. Our research seeks to precisely define the term and make it useful to researchers and practitioners alike.

4 Defining Pliability

The English word *pliable* is defined as "capable of being bent or flexed or twisted without breaking" (WordNet 3.1, 2012). For systems engineering, *pliability* can be defined as "…the ability of a system to change, without breaking its system architecture". We define a *system architecture* to be a collection of components and an associated concept of operations (CONOPs), whose instances provide some value, within a particular context. Since a system architecture can allow different sets of components and CONOPs, the *form* of a system is a specific collection of components (and their associated capabilities), while a *mode of operation* is a specific CONOPs (a way that those components are functioning and interacting with each other). An *instance* is a specific form and mode of operation pair, that belongs to the system architecture. Thus,

$$I_{ij} = [F_i, MO_j] \qquad \textbf{(1)}$$

where I_{ij} is the instance consisting of the *ith* form specified in the set of components and the *jth* mode of operations specified in the CONOPs of some particular system architecture. The *pliable set* of a system architecture, is the set of all possible instances that are allowed by the system architects to belong to that system architecture. Thus, we can write that for system architecture *X*:

$$Pliable\ set\ of\ X = \{I_{11}, I_{13}, \dots I_{nm}\} \qquad \textbf{(2)}$$

where *n* is the total number of sets of components and *m* is the total number of modes of operation specified in the form and CONOPs respectively. We can also say that a *realized system*, is an actual physical realization of a system architecture that provides value to stakeholders. Therefore, pliability is the property of a system to be able to switch to other allowable instances of a system architecture, specified by the architecture's pliable set.

If a system architecture has multiple instances, then a system always assumes one of these instances at any time *t*, and can transition to the other instances defined in the pliable set of its system architecture, while remaining the same system. If a system transitions to an instance outside of its system architecture, then it becomes an unapproved system. Whether this new system will "work" (i.e. provide adequate value to the stakeholders) is unknown (at best), since it does not belong to the set of allowable instances, defined by the architects who designed the system to begin with. Thus, it is usually in the best interest of the architects and decision makers to not let systems change into instances outside of their system architecture.

5 Using Pliability

After we have defined what a system architecture is and its relationship to an instance and a realized system, the concepts and usefulness of pliability can be illustrated with a simple example. Suppose a port authority wants to develop a maritime security system of systems (SoS) that will identify targets as they pass through a particular area of interest (AOI). The stakeholders identify two choices that they wish to explore: The first choice is the number of unmanned aerial vehicles (UAVs) to include in the SoS (either 4 or 8), and the second choice relates to the use of a manned patrol aircraft (MPA). If a MPA is added to the SoS, then it is possible to operate in a double target confirmation mode by requiring both manned and unmanned vehicles to positively identify a target. This mode takes longer, but typically has more accurate results as opposed to a single target confirmation done by UAVs only. Therefore, the stakeholders initially want the system architects to develop and test four designs:

1. D_1 = [4 UAVs / 0 MPA, Single target confirmation]
2. D_2 = [8 UAVs / 0 MPA, Single target confirmation]
3. D_3 = [4 UAVs / 1 MPA, Double target confirmation]
4. D_4 = [8 UAVs / 1 MPA, Double target confirmation]

Using Pliability in Architecting Systems. Given the requirements, the system architects can model and simulate the four systems that satisfy the component and CONOPs considerations that stakeholders are interested in. However, the acquirers ask the architects to make the systems pliable, then the architects must specify exactly what changes can be made to the systems without breaking their system architecture. Based on their initial concepts, the architects realize that switching between 4 and 8 UAVs is trivial. If this is the case, then perhaps a system architecture can be defined that has two instances – one with 4 UAVs and one with 8 UAVs, and a system can transition between the two, as necessary. However, due to safety concerns, mixing manned and unmanned vehicles requires a more complex air traffic control (ATC) implementation than just having unmanned vehicles alone, and therefore it is not possible for a system to transition from one type of ATC to another without a very high cost. For the stakeholder preferences under consideration, this change cost would be considered prohibitive. A purely unmanned vehicle system can work with the complex ATC, so transitions are possible if such a system were to be implemented. Therefore, there are actually two distinct system architectures, defined by the type of ATC they use (illustrated with their connected instances in Figure 3) meaning there are two distinct systems, S_S and S_C not four. One system can transition between two instances (4 or 8 UAVs), while the other can transition between four instances (4 or 8 UAVs, 0 or 1 MPA).

1. Simple ATC SoS (Ss):
   - I_{1S} = [4 UAVs / 0 MPA, Single target confirmation]
   - I_{2S} = [8 UAVs / 0 MPA, Single target confirmation]
2. Complex ATC SoS (Sc):
   - I_{1C} = [4 UAVs / 0 MPA, Single target confirmation]
   - I_{2C} = [8 UAVs / 0 MPA, Single target confirmation]
   - I_{3C} = [4 UAVs / 1 MPA, Double target confirmation]
   - I_{4C} = [8 UAVs / 1 MPA, Double target confirmation]

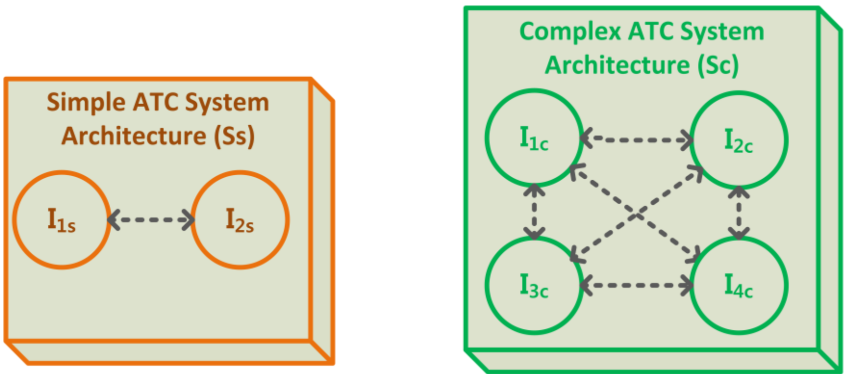
Simple ATC System Architecture (Ss)
I_{1s}
I_{2s}
Complex ATC System Architecture (Sc)
I_{1c}
I_{2c}
I_{3c}
I_{4c}

Fig. 3. Comparison between two different system architectures of the maritime security SoS.

When the two systems are modeled and simulated, they generate tradespaces as shown in Figure 4. Note, that a system with multiple instances doesn't just generate a single utility/cost point in a tradespace for a given context. Instead, it is a collection of utility/cost points generated by each of the instances in its architecture that it can transition to. Similarly, a tradespace can be generated "top-down" from a system architecture, by generating utility/cost points for all allowable instances within its pliable set.

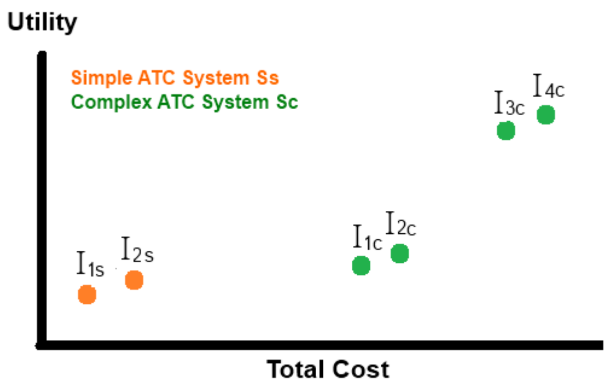

Fig. 4. Tradespace for two different system architectures of the maritime security SoS.

6. Using Pliability to Achieve Survivability, Value Robustness and Agility

Survivability has been defined as the ability of systems to minimize the impact of finite-duration disturbances on value delivery (Richards, 2009). This requires the effective handling of change within systems – either by preventing, mitigating or recovering from unwanted change caused by disturbances, or intentionally changing in response to new contexts. By requiring that systems be pliable, survivability and value robustness can be increased in three different ways; (1) By requiring system architects to go through a design cycle that explores the limits of their systems, (2) by increasing the number of safe instances that system can transition to, and (3) pre-validating change options to reduce the time and effort required for stakeholders to approve changes, allowing them to be implemented quicker and easier. These benefits are discussed below:

Disturbance Discovery and the Pliability Design Cycle. In defining the pliability of systems, the architects must specify what is allowed to be changeable within the system, which means they must provide some guarantee that such changes will not adversely harm its value delivery. To do this, they need to examine the parameters within the system architecture, both in components and in CONOPs, to see what can vary, what can't, and what the limits should be. This exercise forces architects to think about the causes and effects of change within their system, perhaps at a level they normally would not have. In the maritime security example, system architects would have to determine how many UAVs the SoS would support beyond the 4 and 8 suggested by the stakeholders. This is because it's possible that some disturbance may cause UAVs to be unavailable, or perhaps an increase in traffic would necessitate adding additional vehicles to meet the demand. In this example, they might realize that due to range limitations, they have to divide the AOI into two areas, meaning the minimum number of UAVs necessary to cover the AOI would be two. Similarly, due to bandwidth constraints, the maximum number of UAVs would be 12. At this point, they can analyze whether these constraints will satisfy their value robustness requirements. What is the likelihood, given labor shortages, bad weather, random component failures, and other disturbances, that the SoS may find itself with less than two available UAVs? Similarly, what conditions would have to exist for the system to need more than 12 UAVs? In an iterative fashion, as more disturbances and change agents are considered, changes in a candidate architecture are made, which lead to an expansion or reduction in the pliable set, from which feasible systems are evaluated. Thus, the process of defining an architecture and its pliable set becomes a cycle (Figure 5), where system architectures and systems are analyzed in the presence of disturbances and change agents, through the concept of pliability.

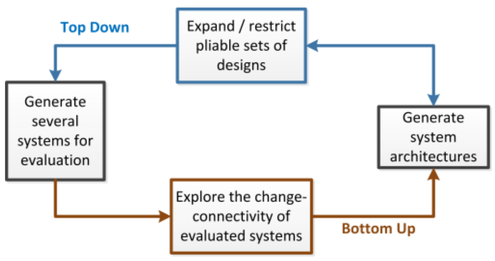
Top Down
Expand / restrict pliable sets of designs
Generate several systems for evaluation
Generate system architectures
Explore the change-connectivity of evaluated systems
Bottom Up

Fig. 5. System cycle for pliability

Increasing Available Options. If a system is pliable, then that means it can change (to some extent) and still maintain acceptable performance under the original contexts that were considered. Survivability is increased automatically as the pliable set of a system architecture expands, simply because the outcome system state of an unintentional change is more likely to be contained within the pliability of the system it affects. Returning to the maritime security example, a system that provides acceptable value with between 2 and 12 UAVs is going to be able to survive hostile attacks, component failures, increases in fuel prices and all sorts of other endogenous and exogenous changes that impacts the number of operational UAVs better than a system that is only designed to be able to accommodate either 4 or 8 UAVs.

Increasing Agility. By pre-validating reachable instances in the conceptual phase, the amount of approval necessary for changes after design should decrease. This reduces the “red tape” and allows complex systems to respond quicker to context shifts, increasing the ability for a system to change quickly (i.e. increasing agility (McGaughey, 1999)) in response to changes in context.

7 Conclusion

Change happens, and it is something that system architects must consider for complex systems with long lifecycle, operating in dynamic environments. Not all change is valuable, and successful systems will be able to avoid, mitigate and recover from harmful changes, and implement beneficial ones in a timely and cost-efficient manner. Incorporating change into engineering design is something that many researchers and practitioners realize is necessary; however, pliability distinguishes systems from their architectures, and introduces the concept that the architecture should be validated so that it has multiple valid instances to which a system may transition, should the need arise. In this way, pliability increases survivability, value robustness, and possibly agility, by requiring architects to consider disturbances, context changes and changes that might not have been considered, increases the safe options available to a system in the event that a change occurs (intentional or not), and decreases the red tape involved in implementing intended changes.

References

De Neufville, R., Scholtes, S. (2011). *Flexibility in Engineering Design*. MIT Press, Cambridge.

Elkins, D.A., Huang, N., Alden, J.M. (2004). Agile manufacturing systems in the automotive industry. In: *International Journal of Production Economics*, pp. 201-214. Elsevier, Burlington.

Leveson, N. (1995). *Safeware: System Safety and Computers*. Addison-Wesley, Boston.

Gupta, Y.P., Goyal, S. Flexibility of manufacturing systems: Concepts and measurements. In: *European Journal of Operational Research*, pp. 119-135. Elsevier, Burlington, MA.

McManus, H., Hastings, D. (2006). A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems. In: *IEEE Engineering Management Review*, pp. 81-94.

McManus, H.M., Richards, M.G., Ross, A.M., Hastings, D.E. (2007). A Framework for Incorporating "ilities" in Tradespace Studies. *Proceeding of AIAA Space 2007*, Long Beach, CA.

McGaughey, R. E. (1999). Internet technology: contributing to agility in the twenty-first century. In: *International Journal of Agile Management Systems*, pp. 7-13. Emerald, Bingley, UK.

Mekdeci, B., Ross, A.M, Rhodes, D.H., Hastings, D.E. (2011). System Architecture Pliability and Trading Operations in Tradespace Exploration. In: *Proceedings of IEEE International Systems Conference*. Montreal, PQ.

Richards, M. G. (2009). *Multi-Attribute Tradespace Exploration for Survivability*. PhD Thesis. Massachusetts Institute of Technology, Cambridge, MA.

Ross, A.M., Hastings, D.E. (2006). Assessing Changeability in Aerospace Systems Architecting and Design Using Dynamic Multi-Attribute Tradespace Exploration. In: *Proceedings of AIAA Space 2006*, San Jose, CA.

Ross, A.M., Rhodes, D.H., Hastings, D.E., Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining Lifecycle Value. In: *Systems Engineering*, pp. 246-262. Wiley, Hoboken, NJ,

WordNet 3.1 Online Lexical Database, wordnetweb.princeton.edu. (*last accessed on 27 Feb 2012*).

Stump, G.M., Yukish, M., Simpson, T.W., O'Hara, J.J. (2004). Trade space exploration of satellite datasets using a design by shopping paradigm. In: *2004 IEEE Aerospace Conference Proceedings,* pp. 3885-3895. Big Sky, MT.