THE SYSTEM SHELL AS A CONSTRUCT FOR MITIGATING THE IMPACT OF CHANGING CONTEXTS BY CREATING OPPORTUNITIES FOR VALUE ROBUSTNESS

Adam M. Ross
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139
617-253-7061
adamross@mit.edu

Donna H. Rhodes
Massachusetts Institute of Technology
77 Massachusetts Ave
Cambridge, MA 02139
617-324-0473
rhodes@mit.edu

Abstract - One of the primary goals in developing systems is to create a system that delivers desired capabilities, or value, to system stakeholders. In practice, system development requires both creative design for meeting the system objectives on paper, as well as technical competence to ensure system value delivery in practice. At least several failure mechanisms exist that may prevent a system from delivering value in practice. These mechanisms include poor concept design, failure in design implementation, changes in system operating context, and changes in stakeholder expectations. Whereas traditional robust design deals with changes in operating context, designing for Value Robustness is a technique to ensure value delivery in spite of changes in a more generalized context and expectations. The concept of system shell is introduced as a value robust construct for mitigating the effect of changes in context and expectations by decoupling the system from the sources of change. The system shell consists of two parts: the system mask and the system shelter. The system mask changes how the system is "seen" by the external context and stakeholders. The system shelter changes how the system "sees" its external context and stakeholders. Examples of system shell applications are presented, as implemented in both software and hardware, and across applications from consumer products to aerospace systems. Implications for the design of systems and systems of systems are discussed, including customizing perception of value delivery, integrating legacy components, and balancing changeability with robustness. Given uncertainty in future system context and use, the purposeful addition of system shells as a part of system design is proposed as a cost-effective approach to maintaining system value in spite of changes in context and stakeholder expectations.

INTRODUCTION

Designers are faced with significant challenges in accommodating the nature of the contemporary engineering environment. This environment is characterized by an ever increasing pace of change, high degrees of uncertainty, complex interconnectedness of technologies and enterprises, and diversity of system stakeholders. Modern engineering also demands the simultaneous deployment of systems to deliver legacy functionality, while also contributing to collaborative system of systems (SoS) functionality.

Ross [8,9] addresses the challenge of designing for changeability, and proposes several new constructs for architectural decision making and design. A formal *change taxonomy* elaborates the various types of real and perceived system changes. A key distinction is drawn between scalability, modifiability, and robustness concerning the type of change necessary to achieve system value.

Robustness is related to an apparent lack of change in perceived value delivery, in spite of changes either internal or external to the system itself. This concept of robustness motivates the construct of system shell as an architectural approach for designing systems that will be robust in the face of change.

MOTIVATION

The desire for “robustness” extends from the fact that change is inevitable, both in reality and in perception. Approaches such as Axiomatic Design and Taguchi Robust Design methods have advanced understanding of how to develop robust systems to deal with real-world changes [6,11]. It is important to note that the goal of system design is not robust systems per se, but rather the delivery of value to system stakeholders. The motivation for changeability over a system lifecycle is categorized into three major drivers: dynamic marketplace, technological evolution, and variety of environments [2]. These drivers result in two key aspects for system architectures to address: 1) they must be able to be changed easily and rapidly, and 2) they must be insensitive or adaptable towards changing environments. In this paper, a construct is proposed that leverages the latter to minimize the need for the former, offering a more cost effective and efficient approach to sustained value delivery.

The complexity and interconnected nature of modern systems drives the cost of system changes to ever higher levels. Careful consideration must be taken when weighing the high cost for customization of a system versus the motivation for that customization. No system can be ‘all things to all people’, yet the contemporary engineering environment does often demand that an individual system simultaneously serve the needs of multiple stakeholder communities that may have divergent perceptions of system value. This need to serve many stakeholders is increasingly experienced by system designers. New constructs are needed to minimize costly changes to the architecture while also providing responsiveness to context shifts and emergent needs.

The motivation for robustness in perceived value in large scale defense systems has been explored during a recent workshop. In the 2003/04 timeframe, the US Air Force and Department of Defense issued new systems engineering policies related to the revitalization of systems engineering and the need to deliver more robust system solutions. In support of these policies, an Air Force/MIT Lean Aerospace Initiative (LAI) Workshop on System Engineering for Robustness was held in June 2004 that challenged the aerospace community to develop a process that enables “systems engineering for “robustness” [7]. “Robustness” according to Dr. Marvin Sambur, Assistant Secretary of the Air Force for Acquisition at the time of the workshop, was defined as:

- Capable of adapting to changes in mission and requirements;
- Expandable/scalable, and designed to accommodate growth in capability;
- Able to reliably function given changes in threats and environment;
- Effectively/affordably sustainable over their lifecycle;
- Developed using products designed for use in various platforms/systems; and
- Easily modified to leverage new technologies.

These goals are similar to the separately defined “ilities” of adaptability, scalability, robustness, sustainability, and flexibility. Experts at the workshop admitted no comprehensive approach existed for designing for “robustness” in this sense and that further research was required in order to adequately address the defense industry needs.

PROBLEM OF ROBUST SYSTEMS

System designers are faced with the challenging problem of creating robust systems in a world that is increasingly dynamic and highly interconnected. Effective design strategies are needed to create systems that can operate in multiple contexts, adapt to changing stakeholder needs and perceptions, and leverage uncertainty. In the modern environment, the designer’s challenge is to develop and select an architectural design that can best deliver a sustained level of value to its stakeholders as context changes, stakeholder perceptions shift, and new demands and opportunities arise. Ross refers to this approach as *designing for value robustness* [8].

We can describe a system in terms of a set of parameters capturing physical and functional aspects. An important aspect of change is the difference in states before and after a change has taken place. *Scalability* is the ability to change the level of a parameter. *Modifiability* is the ability to change the membership of a parameter set. *Robustness* is the ability to maintain parameter values in spite of external or internal context changes.

As an example, consider the following parameter set for a vehicle, which includes both function and form: {number of wheels, color of vehicle, quietness of cabin}. Suppose a design under consideration has the following particular parameter values: {4, “red,” and “moderately quiet”}. The possible ranges for these parameters include: {[3, 4, 6, 8], [“black”, “red”, “blue”], [“very quiet”, “moderately quiet”, “little quiet”, “not quiet”]}. If the current system can maintain its {4, “red,” and “moderately quiet”} in spite of its operating environment changing, such as due to driving on unpaved roads, or past construction sites, then it is robust in these parameters to those particular environments. The more environments to which the system is insensitive, the more robust the system is considered to be.

The concept of system shell is a value robust construct for mitigating the effect of changes in context and expectations by decoupling the system from the system external sources of change. It is not always a viable or affordable approach to make changes to the fundamental system structure or function as value expectations shift. Thus, the shell concept provides an alternate design approach to deliver value by making the system insensitive to external changes, rather than directly adapting the fundamental system for the external change.

The system shell approach is distinct from traditional robust design approaches in that it decouples the robustness mechanism from the system. The decoupled shell can be adapted to past, present, and future contexts, and modified as uncertainty grows or diminishes. Development cost and maintenance responsibilities for the shell can be separated from the system itself, thereby distributing the cost burden for robustness and allowing for customized ownership.

SYSTEM SHELL DESCRIBED

The system shell is comprised of two layers: the inner shell, or “shelter,” and the outer shell, or “mask,” each serving a unique purpose in designing a system for value robustness. The *system shelter* changes how the system “sees” its external context and stakeholders. The *system mask* changes how the system is “seen” by the external context and stakeholders. The system shelter is a protective-based mechanism, while the system mask is a perceptual-based mechanism. Figure 1 below illustrates the concept.

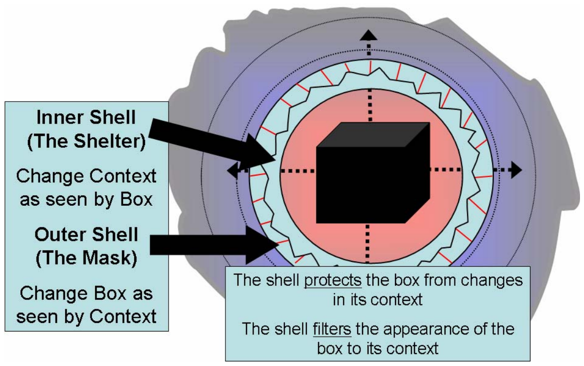
Inner Shell (The Shelter)
Change Context as seen by Box
Outer Shell (The Mask)
Change Box as seen by Context
The shell protects the box from changes in its context
The shell filters the appearance of the box to its context

Figure 1. System shell construct defined.

The purpose of the shelter is to prevent the system from experiencing changes in its context. It “protects” the system from change, thereby ensuring constancy of operating environments. A simple example of a shelter is a protective mechanism for sheltering humans from the elements, which could be a habitat or protective clothing. Instead of redesigning the human to be able to operate in temperature extremes, well-designed clothing and buildings provide shelter to the human system, creating an artificially stable context in which to ‘operate’. Another example, the space suit permits an astronaut to venture into the extreme environments in space. Well designed buildings permit the scientists in Antarctica to live and work in a climate of extreme weather conditions.

The mask changes the system as seen by the context. It “masks” the true system to prevent the system itself from having to change to meet external changing perceptions. This construct is similar to Klir’s definition of “mask” where “different masks lead generally to different behaviors for the same system” [4]. An example of a mask is a software wrapper module, which standardizes the appearance of code so that it can be manipulated by programmers who do not need to know the specific of that particular code. A construction example is when a franchise business builds in a town with a controlled architecture environment. Within the building itself, they still construct the standard structure using the same specifications and pre-fabricated construction parts, but will then ‘wrap’ it with a historic looking brick façade to achieve the required ‘look and feel’ for new constructions in the controlled environment.

The system shell provides a construct for system designers to consider when making architecture decisions during the concept phase, initial design, and/or during the subsequent evolution of the operational system.

EXAMPLES

To further elaborate the system shell concept, several examples are explored, and benefits and costs associated with these are discussed. There are three options for implementing the system shell construct: (1) shelter only; (2) mask only; or (3) shelter and mask together.

Shelter

The shelter prevents the system from experiencing changes in its context, and requires a protective mechanism. Consider the simple problem where a new construction house has been built with water pipes in an unheated garage. The owner elected to use an architectural design that was originally used in a warm climate, however the new construction has been built in a cold climate. The design worked fine in a warm climate, but in the more extreme temperature environment, a problem has occurred where the water pipes are freezing when the temperature drops below a certain level. Since this is a case where the 'system' has already entered operational use, the cost of relocating the water pipes would be a very costly solution though it may still offer the most reliable solution to the problem. An alternate solution that a builder often invokes that uses the shelter construct is to wrap the water pipes in foam insulation. In most cases, this design solution will address the problem of the freezing pipes and the cost to the owner is very low as compared to the cost that would be involved in relocating the pipes to a heated area, or the cost of heating the garage where the pipes are located.

Another example in the building regime is the use of protective films on glass to reduce solar penetration into internal rooms, likewise accomplished through tinting windows on cars. Radiation shielding for people and equipment serves the shelter function by isolating the people and equipment from the radiation. In a smaller sense, earplugs serve a similar purpose by preventing sound from penetrating the ear canal. In the area of software, firewalls serve the purpose of insolating computers from malicious streams of data on networks.

Mask

The mask changes the system as seen by the context, that is, it "masks" the true system to prevent the system itself from having to change to meet external changing perceptions. The mask construct is increasingly used in consumer products as a strategy to address the need for satisfying diversity of stakeholder stylistic preferences. An early example was the watch design with many variations of faceplates (Swatch™). Nokia™ phones are another example, where an underlying core architecture is used in a family of products which appear to the customer to be different products. The company's strategy to implement variation in the "mask" layer of the phone is a cost effective approach to offering a product line that accommodates market diversity and change drivers.

Indirectly, the interface components of systems serve the purpose of a system mask. An example are customizable software interfaces, both simple, such as switching between "basic" and "advanced" modes, or complex, such as customizing themes in the Windows™ operating system environment. Likewise, the Global Positioning Satellite (GPS) system and satellite radio systems consist of large, expensive, and complex remote space components, which are very difficult to modify or customize to particular users. The use of simple hand-held receivers with various appearances and capabilities, in a sense, approximates a system mask, customizing a user's experience with the system.

Dual Use of Shelter and Mask

In some cases, the designer may find advantages in the dual use of the shelter and mask constructs. The shelter's protective mechanism and the mask's perceptual adaptation mechanism offer complementary strategies. As a simple example, a consumer may find a personal apparel item to offer value robustness as it may impress one's aesthetically driven audience. For example, a person interviewing for an executive position would find a sweat suit could protect them from the elements, but a better apparel choice would be a business suit as both protection from the cold and as the 'mask' most suited for the standard business interview situation.

Another example is the wall socket and common plugs that permit appliances to be connected to any electrical system. Consider the case of wanting to use an electrical appliance in a country in which it was not designed to operate. Through the use of an adapter with converter, the device is able to function properly. The adapter serves as a mask to change the appearance of the appliance to match the "expectation" of the outlet. The converter serves as a shelter to protect the appliance from experiencing any effects for the non-standard electrical current. Together, the adapter with converter shell enables the appliance to continue to function in a new environment, continuing to deliver value to the appliance's user.

DISCUSSION

Implications for Design

Using the system shell construct for system value robustness has several implications to consider during design. First, as in traditional robust design, the system shell allows for the extension of system usefulness into new and different contexts or operating environments. Second, the system shell allows for new possibilities for customizing system perception and experience across multiple stakeholders even across time. Third, the appropriateness of when and where to use a system shell must be considered, as sometimes the cost of using the construct may exceed its benefit.

The first implication for extending operating ranges is multiplied when using the system shell over traditional robust design as multiple system shells could be developed to allow for customized or upgraded performance in new environments. As an example, consider the ability to add "skins" to the Apple iPod™. Such skins could be purely aesthetic, performing the system mask role, or could be both aesthetic and protective, preventing damage from impacts or temperature extremes, and thus performing the function of a full system shell.

The second implication of customizing the experience relates principally to the system mask role. Instead of struggling with determining how to compromise or aggregate diverse customer desires into a single system, the mask allows for a customer-specific experience. Masks could be developed to alter the appearance of the system to appear "red" to one person, but "blue" to another. Figure 2 below gives a notional example of such a system mask.

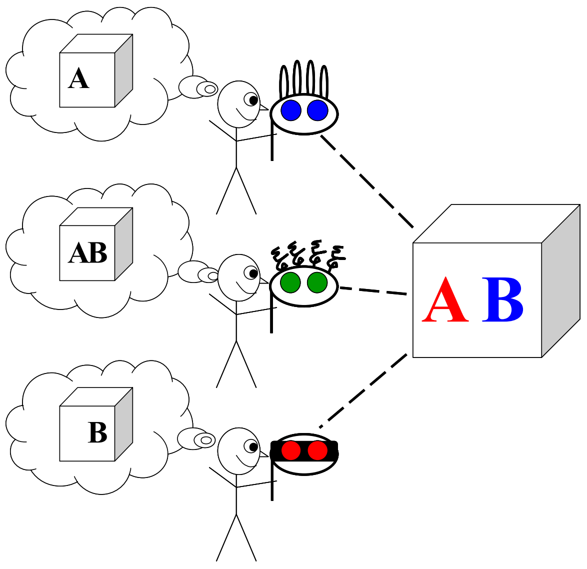

Figure 2. System mask customizes perception of same system to different stakeholders.

The third implication is the additional analysis needed to determine the appropriateness of when and where to use a system shell or one of its facets. Sometimes the cost for developing the system shell will exceed the benefit for having the shell. Such would be the case if the cost to develop a new system with the new desired values is less than developing a system shell that modifies the old system to give those same new values. Another reason for not pursuing a system shell is when extending the useful life of a system is not desired, such as the desire for new performance, technologies, concepts, user expectations, etc. As an example, Saleh discusses a type of analysis needed when trying to decide between replacing and repairing a legacy satellite system [10]. In essence the net benefit of the old system with a system shell must exceed the net benefit of the old system plus a system replacement in order for the system shell concept to make sense, as shown in the equation below.

$$Benefit_{old\_sys+shell} - Cost_{shell} > Benefit_{new\_sys} - Cost_{new\_sys}$$

System of Systems

Many of the engineering challenges today involve taking a system of systems approach which requires the simultaneous deployment of systems to address their legacy function, while also contributing to collaborative systems functionality of the SoS. Maier describes system of systems, also known as collaborative systems, noting “the interfaces, whether thought of as the actual physical interconnections or as higher level service abstractions, are the primary points at which the designer can exert control” [5]. He describes “leverage at the interfaces” as one of the fundamental principles for architecting system of systems. The system shell provides an enabling design concept to implement this principle through shelter and/or mask constructs.

The concept of a system shell is powerful in that it decouples the system itself from changes in its context. In order to make design sense, system shells should be developed to be able to be modified much more readily and for lower cost than the system itself. This may be particularly important as systems increasingly participate in SoS operations. In this case, the system retains its legacy function and behavior while simultaneously taking on a contributing role in the SoS. In such cases, the strategy of making changes to the system itself to accommodate the SoS unique needs might negatively impact the ability to continue to perform in its legacy role. By wrapping legacy components, the system shell concept can allow the incorporation of legacy systems into a larger system without having to perform costly modifications or redesign. In effect, the design of SoSs can be through the design of systems shells and their interactions, rather than on the components themselves, a useful approach, especially when dealing with legacy components or components outside of a designer’s control or influence

A recent study on the need for new systems of systems engineering methods is described in a recent report of the US Air Force Scientific Advisory Board [1]. This report stresses the important role of “convergence protocols” to provide mechanisms for the linkage of legacy systems to address emergent needs. These protocols are intended to guide SoS designers to develop SoS component shells that allow each component to evolve without negatively impacting the overall SoS value delivery. Permitting the independence of components in SoS design, as described in Keating, as an essential and distinct aspect in the successful development of an overall SoS built from existing systems [3].

Separating Changeable System Parts

One common perception is that the ability to change a system and robustness are in tension, where one must give up one to have more of the other. Such is not necessarily the case; it depends on the parameters under consideration. For example, making a computer robust to noise and physical impacts does not reduce its modifiability for changing components.

At a higher level is the concept of value robustness. If the goal for system design and development is to deliver value to stakeholders over the system lifecycle, then value robustness is the ultimate goal for the designers. Value robustness can be achieved through either passive or active means, with the former more akin to traditional robust approaches, and the latter embracing changeability as a dynamic strategy for value sustainment. Passive value robustness delivers value through the development of “clever” designs, or designs insulated by system shells, which are perceived to maintain value over time. Successful development of passively value robust systems, including those with system shells requires anticipation of future system contexts, including potential value perceptions and the competitive environment with alternative systems. Active value robustness requires less omniscience, but does have the added complexity of needing a change agent changing the system over time to maintain high value perception.

CONCLUSION

System designers are increasing challenged to design systems to be value robust in a dynamic and demanding world. Some traditional robust design methods have proven effective for addressing changes in the operating context, but designers can benefit from additional approaches, particularly to address highly complex and interconnected systems. In order to ensure sustained value delivery, designers need enhanced ways of thinking for architecting new systems or augmenting existing systems.

Designing for Value Robustness is a technique to ensure value delivery in spite of changes in a more generalized context and expectations. One approach of this overall technique has been presented in this paper, the concept of system shell as a value robust construct for mitigating the effect of changes in context and expectations by decoupling the system from the sources of change. Several areas of research are ongoing to further the design for value robustness methodology, including case studies that are expected to yield new insights toward this goal.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge support for this research from the MIT Systems Engineering Advancement Research Initiative (SEA^{RI}), Air Force Office of Scientific Research (AFOSR) and Lean Aerospace Initiative (LAI).

REFERENCES

[1] Air Force Scientific Advisory Board, “Report on Systems of Systems Engineering for Air Force Capability Development,” Air Force SAB-TR-05-04, July 2005

[2] Fricke, E. and Schulz, A.P., “Design for Changeability (DfC): Principles to Enable Changes in Systems Throughout Their Entire Lifecycle,” *Systems Engineering* 8(4), 2005

[3] Keating, C., Rogers, R., Unal, R., Dryer, D., et al., “Systems of Systems Engineering,” *Engineering Management Journal* 14(3), September 2003

[4] Klir, G., *An Approach to General Systems Theory*, New York: Van Nostrand Reinhold Company, 1969, pp. 118

[5] Maier, M. W. "Architecting Principles for Systems-of-Systems," *Systems Engineering* 1(4): 267-284, 1998

[6] Park, S. H., *Robust Design and Analysis for Quality Engineering*, New York: Chapman & Hall, 1996

[7] Rhodes, D.H., “Air Force/LAI Workshop on Engineering for Robustness,” Technical Report, Lean Aerospace Initiative, http://lean.mit.edu, July 2004

[8] Ross, A.M., “Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration,” Ph.D. in Engineering Systems, Massachusetts Institute of Technology, Cambridge, MA, June 2006

[9] Ross, A.M., and Hastings, D.E., “Assessing Changeability in Aerospace Systems Architecting and Design Using Dynamic Multi-Attribute Tradespace Exploration,” AIAA Space 2006, San Jose, CA, September 2006

[10] Saleh, J. H. "Weaving Time into System Architecture: New Perspectives on Flexibility, Spacecraft Design Lifetime, and On-orbit Servicing," Ph.D. in Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, June 2002

[11] Suh, N.P., *Axiomatic Design--Advances and Applications*, New York: Oxford University Press, 2001