

**Introduction**

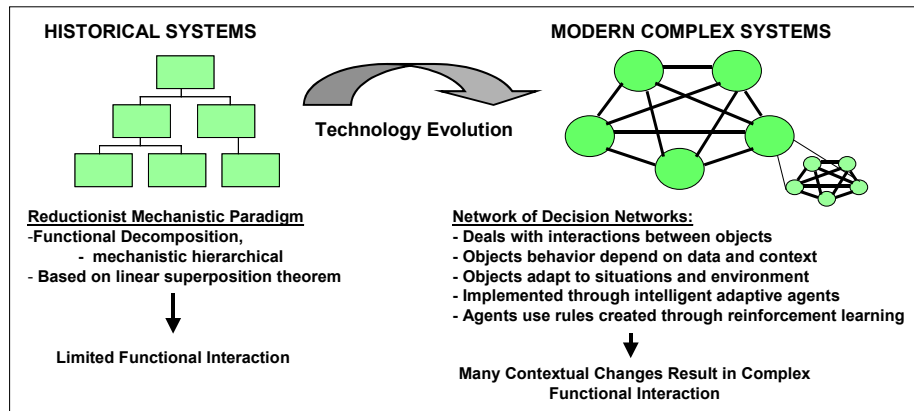
This paper introduces KBArchitecting® (Knowledge Based Architecting) a novel systems engineering approach that moves beyond traditional systems engineering to address the many problems that are being encountered in current day development of complex systems. These problems generally manifest themselves in the development of large software intensive systems as typically seen in large DoD (Department of Defense) acquisition programs. There are many reasons why such programs get into trouble or fail. At the root of these problems (and the premise for this paper) is the inadequacy of traditional system engineering approaches to understand user requirements and the problem space of Complex Adaptive Systems. Some of the resulting negative impacts on program development are typified by problems and factors that are identified in Table 1 below. KBArchitecting® by FORELL Enterprises, Inc., of Buena Park, CA. is a systems engineering approach that uses elements of Knowledge Engineering and Decision Engineering, an approach we assert is needed in mainstream systems engineering in order to overcome these problems.

<b>Program Problems</b>	<b>Contributing Factors</b>
The deliverable system does not satisfy the end-user	<ul style="list-style-type: none"> <li>• Inadequate domain expertise knowledge</li> <li>• Inaccurate capturing of <i>user requirements</i></li> </ul>
Requirements Creep	<ul style="list-style-type: none"> <li>• Inaccurate translation of <i>user requirements into system requirements and software requirements</i></li> <li>• Inadequate understanding of the <i>requirements</i> to a level of detail that allows implementation</li> </ul>
Schedule Overrun	<ul style="list-style-type: none"> <li>• Ambiguous and inadequate <i>requirements</i> definition</li> <li>• Inadequate process controls</li> <li>• Inadequate management controls</li> </ul>
Cost Overrun	<ul style="list-style-type: none"> <li>• Ambiguous <i>requirements</i> definition</li> <li>• Lack of understanding of scope of task due to lack of understanding of <i>user requirements</i></li> <li>• Lack of understanding of scope of task due to lack of understanding of <i>detailed requirements</i></li> <li>• Inability to correctly <i>allocate requirements</i> and design and implement a system that satisfies complex system performance needs</li> <li>• Inadequate process control</li> <li>• Inadequate management control</li> </ul>

**Table 1 – Program Problems and Their Contributing Factors**

The *inability to define requirements* to a level that allows *successful* system and software development is a major contributor to program failure. This problem has been exacerbated in recent decades by the increasing complexity and size of software in systems and the increasing levels of automation needed for effective operability and maintenance of systems. To overcome this problem, the complexity for such systems must be adequately addressed in the requirements definition for the system. Requirements definition falls squarely in the systems engineering domain.

We can define success as those programs that deliver systems that satisfy the end user and are delivered within schedule and cost constraints. We propose that KBArchitecting® is a systems engineering approach that uses elements of Knowledge Engineering to overcome some of the fundamental issues in systems engineering while still adhering to those engineering principles that are sound and have been proven over time.

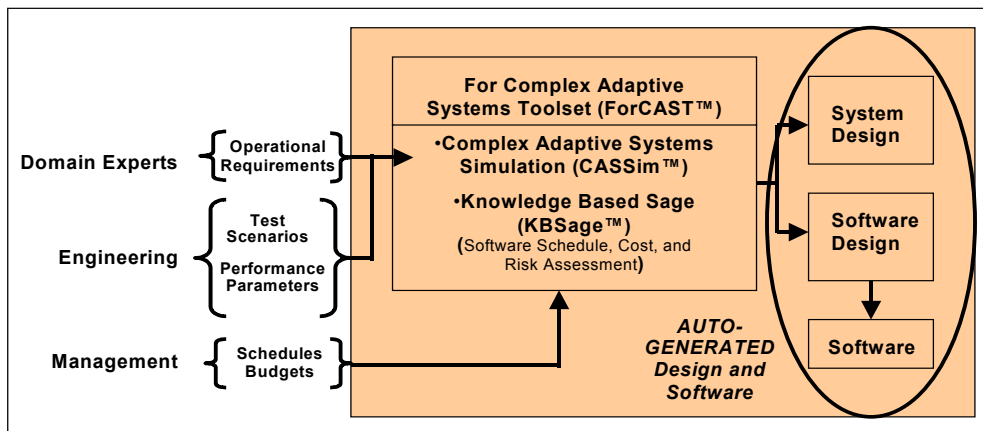


**Figure 1 – System and Technology Evolution**

Figure 1 contrasts the traditional reductionist approach to systems engineering with the modern context for complex adaptive systems. Current day systems are much more complex in terms of the functionality they provide and in terms of the many context sensitive interactions that occur. Earlier systems were characterized by well-defined behavior, whose inputs and outputs can be described in requirements and interface documents with reasonable certainty. Test plans and test procedures depended on predictable mechanistic behavior, i.e., the same system stimuli produce the same results. Today’s systems are networks of many other networks, whose interactions are many and complex. No longer does the same stimuli produce the same results, because the results depend on the “state of the system”, i.e., the context in which system components are executing. Defining functional requirements and their associated performance requirements for today’s complex context sensitive systems requires a new approach to engineering. This paper introduces such a concept, i.e., KBArchitecting®.

**Discussion**

**KBArchitecting® objectives** are to significantly reduce the manual activities between the information given by domain experts and the production of the deliverable system, providing an automated development bridge between the end-user and the delivery of the production system. In addition, KBArchitecting® is to facilitate delivery of a highly optimized system, whose performance is calibrated to the performance needs of the end-user. Figure 2 provides a conceptual view of KBArchitecting® and related tools for addressing complex adaptive systems. CASSim™ is based on simulation software



**Figure 2 – KBArchitecting® Conceptual View: An Auto-Generated System**

developed by Dr. John Clymer, CSUF in conjunction with his curriculum “Simulation Based Engineering for Context-Sensitive Systems”. KBSage™ is based on a tool by Dr. Randall Jensen, one of the original authors of SEER/SEM, a widely used software estimation tool.

As can be seen from this figure, there are three disciplines contributing to the input of KBArchitecting®: the domain experts, engineering, and management. Their responsibilities are limited to providing the input to KBArchitecting® as depicted in the figure and performing the associated analysis on the output. Additional realized benefits consist of increased software productivity, system scalability, flexibility, reliability, and maintainability, and reduced overall life cycle cost.

### KBArchitecting® Process Logical View

The basic supposition in KBArchitecting® is that defining requirements is best left to those that are considered to be domain experts. It is also generally known that when talking to two different people in the same domain, that each will have a different view on how to best accomplish his or her task. An effective engineering approach must therefore be flexible enough to allow for the different approaches (operational task sequencing), while still providing the robustness that allows each of the domain experts to achieve their end-objectives. Figure 3 provides a logical view of the KBArchitecting® engineering process.

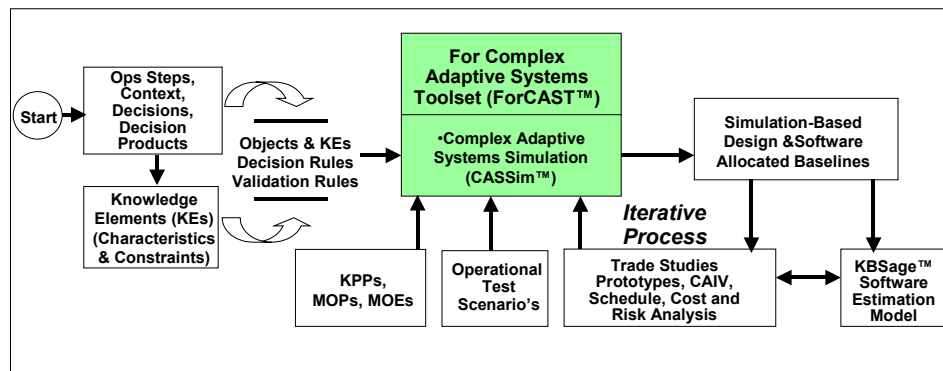


Figure 3 – KBArchitecting® Process Logical View

KBArchitecting® is based on elements of knowledge management and decision engineering, capturing decisions and their associated information. This information is collected for each step in the operational process for each domain. The information collected in the KBArchitecting® process is used in an advanced simulation capability, CASSim™, to examine the problem space at the system level, i.e., examine the interactions between the different system components and determine their emergent behavior. This modeling leads to:

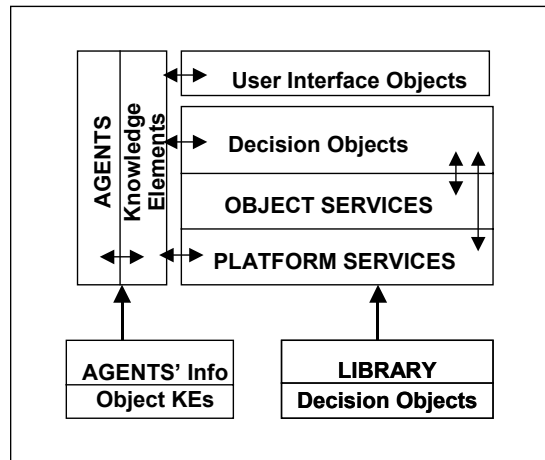
- 1) Identification of those components that contribute to accomplishing operator tasks in the most effective manner, i.e., tasks that do not contribute to the operational decision process are presumed to be unnecessary. Note that if two operators came up with two different sets of tasks and decision sequences, that those tasks that do not contribute to the decision outcome are eliminated in this step.
- 2) Allocation of components to the hardware for optimized performance.

The decision information for each step in the operational process is translated into a set of system components and a set of initial Artificial Intelligence (AI) rules, that define the decision constraints,

context, and system environment in which these components will act in the target environment. CASSim™ uses this initial information to explore the problem space and to determine the emergent system behavior. Through feature extraction and re-enforcement learning techniques, CASSim™ creates a set of AI rules that when used in the KBArchitecting® Objective Decision Architecture System will result in optimized system performance.

**KBArchitecting® Objective Decision Architecture**

KBArchitecting® Objective Decision Architecture (ODA) is an event-driven agent-based system in which the components are made up of decision objects, following the Object Oriented Design (OOD) paradigm. Agents in ODA operate on decision objects, using rules generated through CASSim™. System states and decision context cause events to occur. Upon occurrence of these events, agents will execute the associated decision objects with the required user interface. KBArchitecting® makes a distinction between automated decision objects and user interface objects to support Cognitive Task Analysis (CTA) and Cognitive Workload Analysis (CWA), which contribute to *reduced manning and automation analysis*. Figure 4 shows the KBArchitecting® ODA.



**Figure 4 – KBArchitecting® ODA – An Event-Driven Agent-Based System**

In ODA, special agents called System Monitors check the context and system states associated with the decisions in the operational system, arbitrating decisions in ambiguous situations via fuzzy logic. The System Monitor is a special monitor in that it only monitors for “True” conditions in order to trigger the execution of the appropriate decision objects, i.e., states are posted and are not sensed or polled. Figure 5 represents a logical view of the System Monitor in KBArchitecting® ODA System. This System Monitor

Decision Context	System State	Event	Trigger	Execution Sequence								
State Variables a, b, c	X	<table border="1" style="display: inline-table;"> <tr><td>a</td><td>b</td><td>c</td><td>X</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	a	b	c	X	1	1	1	1	=True	Decision Object A Decision Object B
a	b	c	X									
1	1	1	1									
State Variables d, e, f	Y	<table border="1" style="display: inline-table;"> <tr><td>d</td><td>e</td><td>f</td><td>Y</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> </table> <p>=False Apply Fuzzy Logic</p>	d	e	f	Y	1	1	0	1	=True	Decision Object A Decision Object C Decision Object D
d	e	f	Y									
1	1	0	1									

**Figure 5 – KBArchitecting® ODA System Monitor**

is special in that it applies fuzzy logic techniques to “force” a true condition under special circumstances, e.g., tactical military and time critical situations. When “forcing” a true condition, a probability of correctness ( $P_c$ ) is calculated based on a weighting factor for each state variable contributing to an event. The  $P_c$  factor can then be used to evaluate the results of the decision objects execution.

### **Information Bundling and Islands of Knowledge**

We have included in our approach, a basic concept i.e., “bundling” that we believe will enhance interoperability and increase transaction efficiency across complex systems. Using this concept, all information that is required for a decision object to execute in the target system is “bundled” and is referred to as *Islands of Knowledge*. The Islands of Knowledge include the user interface objects, which contain the data elements associated with any user displays and includes the system behavior and display behavior. To ensure data consistency and integrity, decision objects are aligned along database transaction boundaries. This may be supported utilizing existing agents or more robustly using modern database management systems with transaction rollbacks.

### **Decision Object Bundling**

While traditional systems engineering decomposes functions to their lowest level, KBArchitecting® objective is to bundle interdependent decision objects for optimized system performance.

### **Decision Object Grouping**

CASSim™ examines the interaction between the different decision objects and identifies those objects that can be bundled. In addition, CASSim™ examines the interaction between the different bundled decision objects to support an optimized system allocation.

### **KBArchitecting® Logical System**

KBArchitecting® Logical System is a system, which is made up of *reusable* decision objects, which are bundled, grouped, and allocated to provide maximum system performance. To improve interoperability, data is moved at the “Islands of Knowledge” level rather than at the data element level, yielding a

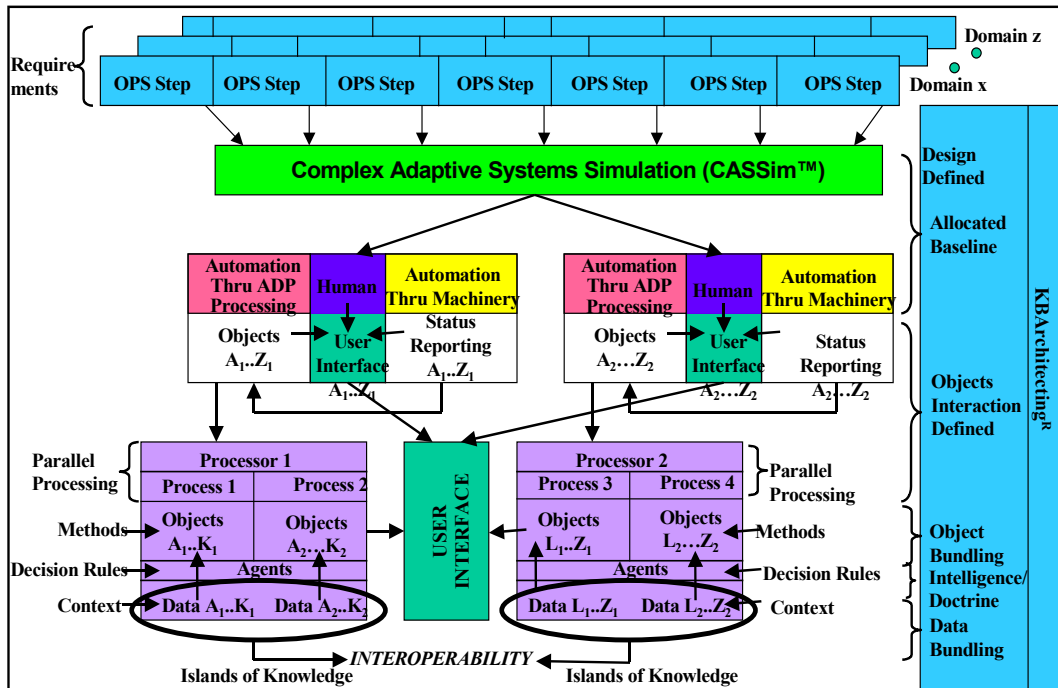


Figure 6 – KBArchitecting® Logical System – Provides Maximum System Performance

significant performance improvement. Figure 6 shows the KBArchitecting® Logical System.

### KBArchitecting® Benefits

KBArchitecting® engineering paradigm is based on operational decision sequences as defined by domain experts. This information is used directly to “auto-generate” decision objects for the operational system, thus reducing the problem associated with requirements definition, generation, and translation. These decision objects are executed in the required context by a set of AI rules that are generated by CASSim™. Decision objects can be readily reused in other systems by evaluating and changing the AI rules. New decision objects can be added without impacting existing objects, thus allowing for technology insertion. The net effect of KBArchitecting® is a highly optimized system that implements user requirements as defined by the domain experts. This engineering approach provides for significant cost reductions over the life of a program.

**For further information contact: Lary D. Smith at 714-690-7720 or laryds@eforell.com**