

KBArchitecting® - Knowledge Based Architecting for MORE EFFECTIVE SYSTEMS

This paper introduces KBArchitecting® a decision based engineering approach that moves beyond traditional systems engineering to address the many problems that are being encountered today’s 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 tend to overrun in schedule and cost.

At the root of these problems (and the premise for this paper) is the inadequacy of traditional system engineering to adequately define and document user requirements in a manner that supports effective and efficient implementation. Problems that are being encountered in current developmental programs and factors that contribute to these problems are identified in Table 1.

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

Table 1 – Program Problems and Their Contributing Factors

KBArchitecting® - a Novel Systems Engineering Approach

KBArchitecting® captures and documents user requirements in the form of DECISIONS that are made in the OPERATIONAL ENVIRONMENT. In addition, KBArchitecting® captures all the necessary data to make these decisions in Knowledge Elements (KEs). Using traditional engineering approaches, the IDEF0 charts capture inputs, outputs, control, and resource information. KBArchitecting® adds to this information by capturing limitations and constraints on processing, on control, and on resources. As can be seen from Figure 1, our IDEF0 differs in that we also capture the context in which decisions are to be made, its dependencies, and the required performance parameters, such as, Key Performance Parameters, (KPPs), Measures of Performance (MOPs), and Measures of Effectiveness (MOEs).

KBArchitecting® - Knowledge Based Architecting for MORE EFFECTIVE SYSTEMS

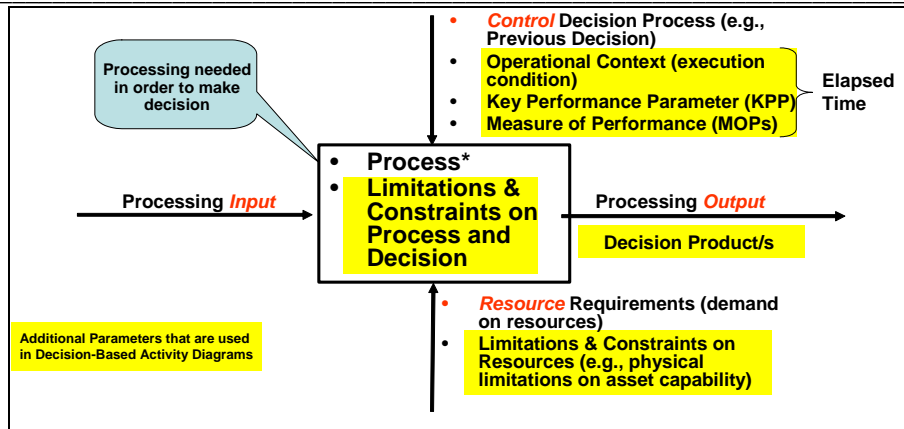


Figure 1 – KBArchitecting® IDEF0 Diagram Capturing Additional Performance Data

The decision information and its processing are then encapsulated in Decision Objects following Object Oriented Design (OOD) concepts.

This information is then used by FORELL’s advanced simulation and modeling capability (CASSim™ - Complex Adaptive Systems Simulation) to ensure that KPPs, MOPs, and MOEs are being satisfied. CASSim™ analyzes the interaction between decision objects, determines the emergent behavior, and generates a set of rules for decision object execution. These rules contain decision object control and resource utilization information for the end-system. Included in these rules is information about the context in which these decision objects can be executed. The context is defined by the system state, which is monitored by our intelligent agents. The specific system states define operational context. *By merely changing the rules and changing the sequence of the decision objects, one can easily create new systems with different behaviors.* The result is a flexible, scalable, and extensible system. Figure

KBArchitecting® - an Effective Test and Integration Approach

During the traditional system engineering processes, test plans and test procedures are generated to test and validate functional and interface requirements. There are at least three levels of testing: at the unit level, at the parameter and assembly testing level, and at integration testing or system testing level. Because functional tests are generally not aligned with operational threads, separate integration test plans and procedures are required in order to test and validate the operational threads, i.e., different test paths through the functionally oriented software are executed and tested in order to validate operational threads. Whether or not performance requirements are being satisfied is almost always uncovered during testing rather than during the design phases. In many programs performance is almost always treated as an afterthought, with the remedy of adding more hardware to the original configuration or figuring that by the time a program gets delivered, the hardware will have sufficient processing capacity through the numerous technology upgrades. KBArchitecting® on the other hand takes performance into consideration right from the beginning, thereby reducing overall program development risk.

Decision objects are “strung together” for execution to create operational threads that are executed along natural decision boundaries. Figure 2 provides a conceptual view of how decision sequences support operator tasks and how operator tasks then support missions during execution of decision objects in the end system.

KBArchitecting® - Knowledge Based Architecting for MORE EFFECTIVE SYSTEMS

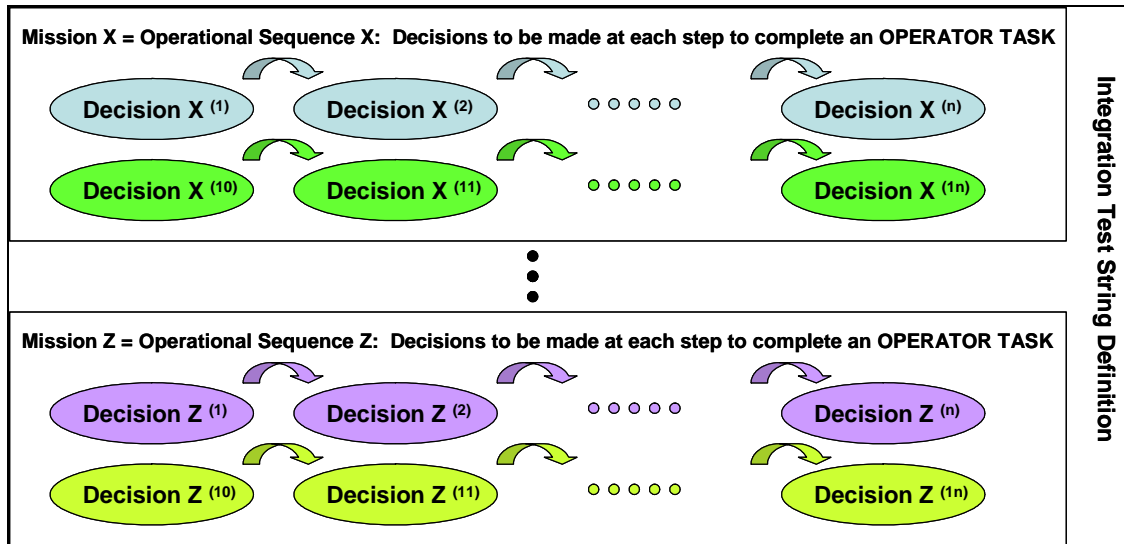


Figure 2 – Operational Decision Sequences Support Operator Tasks to Support Missions

KBArchitecting®, testing and validation is performed at the decision object level. Decision objects are tested and validated for *producing the “right” decision products with the desired performance*. Since KBArchitecting® defines and captures all the data that is needed for decision objects to execute, *decision objects stand on their own and are autonomous*. Where there are no dependencies between decision objects, these decision objects are modeled for parallel processing. The concept of *parallel processing is inherent in KBArchitecting®*, providing a solid foundation for leveraging of today’s parallel processor technology, with the possibility of significantly increasing system performance.

KBArchitecting® Decision Open Architecture

KBArchitecting® Objective Decision Architecture is an *Event-Driven Intelligent Agent-based Open Architecture System*. Figure 3 provides a conceptual view of our Decision Based Open Architecture.

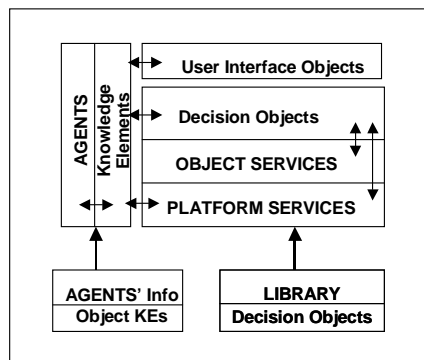


Figure 3 – KBArchitecting® Open Architecture View

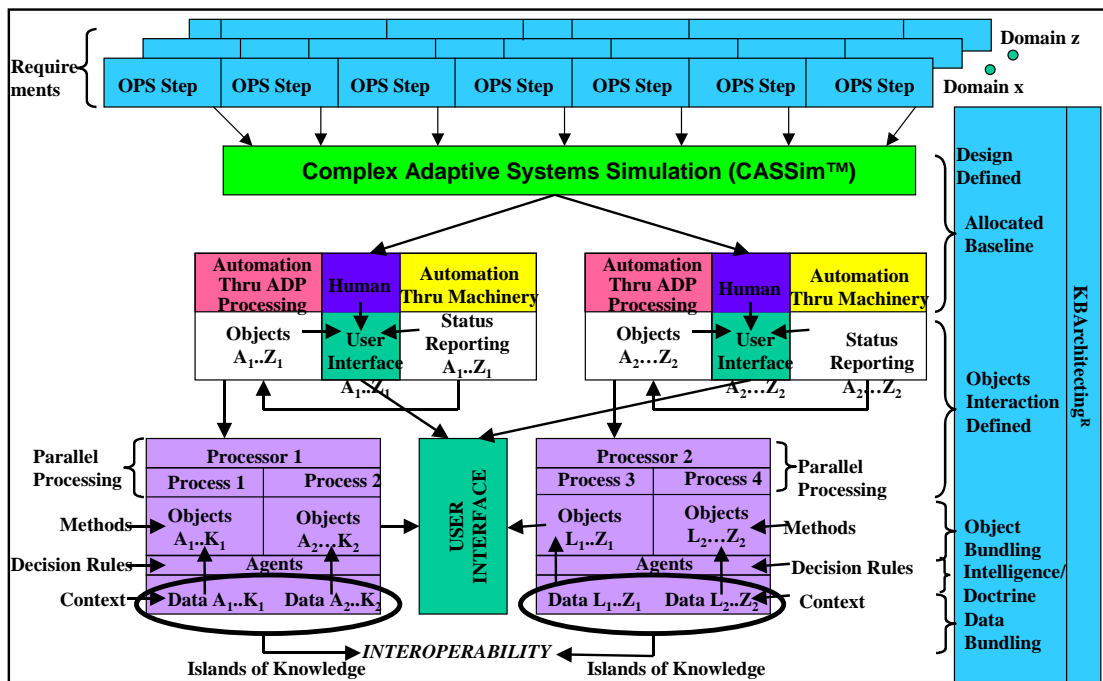
Intelligent Agents monitor the operational environment by monitoring system states that are correlated to operational context by our Intelligent Agents. *Decision objects execution occurs through operational*

KBArchitecting® - Knowledge Based Architecting for MORE EFFECTIVE SYSTEMS

context switching that is initiated by our Intelligent Agents rather than by the Operating System, which has no knowledge of the operational tempo. Operational context switching occurs based on the rules that have been generated during the KBArchitecting® engineering process. During the KBArchitecting® engineering process, an initial set of rules is generated and is modified for optimal performance through our simulation and modeling process. The set of optimized rules is then used in the end-system.

KBArchitecting® Logical System

KBArchitecting® Logical System is depicted in Figure 4. This system is composed of reusable decision objects, which are bundled, grouped, and allocated to hardware, software, and people to provide maximum system performance. To improve interoperability, decision objects with their



associated data

Figure 4 – KBArchitecting® Logical System – Provides Maximum System Performance

and rules for execution are exchanged between systems, leading to significant system performance.