Application of KEEL® Technology to Computer Science, Signal and Data Processing

Compsim White Paper

Keywords: Cognitive Technology, Information Fusion, Adaptive Control, Analog Circuits, Dynamic Graphical Language, Reasoning System, Advanced Mathematics

Objective: The need to address complex, dynamic, non-linear, inter-related problem sets as part of a missile defense system is a continuously evolving problem. This need requires systems that can address complex models in real-time. The economics of creating and maintaining these systems must also be considered part of the problem.

Often the domain experts (those that understand the problem) are different individuals than the programmers and engineers that are tasked with solving the problems. This requires significant dialog and trial-and-error to match the solution to the problem. There would be significant value in a solution that allowed the domain expert to create and test the executable models without passing the problem from person to person. There would also be significant value in a solution that simplifies the entire design.

This paper suggests the use of KEEL (Knowledge Enhanced Electronic Logic) Technology as a key component in addressing these complex data fusion problems.

KEEL Technology Overview

KEEL is a new "technology"; it is not just a new "tool". It is a fundamentally new way to process information. In one implementation it processes information on a digital computer as if it was executing on an analog computer. In this computer-based implementation the very small memory footprint may be a key advantage[1]. In another mode, the model can be implemented as an analog circuit when very high performance is needed. Creating and packaging this technology is accomplished with a new "dynamic graphical language". This is not just another "flow charting language" where one is defining data flow. Information items are modeled by using graphical items where size indicates the instantaneous importance of information. Wires define specific functional relationships. The KEEL "dynamic graphical language" allows one to interact with the design while the models are being developed. By simulating changes to inputs and "seeing" the system adapt.

KEEL was developed to respond to what Dr. Horst Rittle (UC Berkley) described as "wicked problems". He described wicked problems as those that were difficult or impossible to effectively address by writing a "formula". Humans commonly solve these kinds of problems by using judgment or reasoning. Humans "interpret" information and

[1] KEEL Engines are table driven functions. The code to implement a KEEL engine is approximately 3K words (no matter which computer language is chosen) and no matter how large the problem domain. The problem is described with data in tables. The size of the tables grows as the scope of the problem grows.

"balance alternatives". They coordinate the allocation of resources as they react to a changing environment. KEEL Technology operates the same way. When incorporated into systems, KEEL "Engines" (cognitive engines) are triggered to respond to changing events. The KEEL Engines can run continuously, or they can be triggered by change of state, or they can be periodically polled. When implemented as analog circuits they can operate at "VERY" high speeds. The KEEL Engines themselves are architecture independent. System engineering tools are provided that allow multiple KEEL engines to be integrated into a single application on one computer. Alternatively, distributed decision-making can be accomplished with agents distributing information across any network.

The KEEL dynamic graphical language has been identified as a new form of mathematics. Unlike conventional mathematics (formulas), KEEL can ONLY be developed on a computer. This is because it is a "dynamic" language that can only be described on dynamic media (a computer display).

Addressing complex problems

Another key difference in developing solutions with KEEL is that one builds models by considering how information is related and observing how the system responds, rather than thinking about what a formula would look like in order to solve a problem and then testing it to see if you get the correct answer. This is a subtle difference, but in the second case the designer is more concerned with the formula than with how the overall system will react or adapt.

Because KEEL is used to address complex models, it is helpful to know that the design is being monitored as it is being developed. Solving these inter-related problem sets, may cause the designer to create unstable designs. The KEEL Toolkit monitors the design as it is being created, warns the designer of the instability, and then blocks the bad design.

KEEL Engines

KEEL Engines are implemented as a "class with methods" or as "two or three functions and a series of tables" depending on the source code language selected for productization. There is one primary information-processing function that reviews data from input tables and saves output results to output tables. KEEL Engines process information by iteratively executing this core routine until a stable answer to all inputs and outputs (internally and externally visible) is achieved. Two versions of the execution model are available: one that is optimized for speed, and the other that is optimized for memory space. The version optimized for speed is used to design the analog circuit for deployment.[2]

[2] While a patent application for the analog circuit implementation has been granted, it has not yet been applied in practice.

One service that can be automatically integrated into a solution is one that allows inputs and outputs to be published (as XML) such that the decision-making of that final device can be audited. This file can be used to animate the KEEL dynamic graphical language so one can "see the device think". This can be used to audit simulations or operate like a black box in an airplane. This is extremely useful as KEEL Engines can be used to address very complex situations. There are no limits to the number of inputs and outputs that KEEL Engines can handle.

KEEL Development Process

1. Identify the outputs (control variables) that will be driven by the system.
2. Identify the inputs (information items that will ultimately drive the outputs).
3. Identify the interactions between inputs that control the outputs. This is commonly done within the KEEL Toolkit where the domain expert is defining (graphically) how the values interact. In this area the domain expert is "thinking in curves" which define inter-relationships. This is done completely without resorting to complex mathematics in the conventional sense.

Designing in KEEL is performed with common drag and drop practices. Size of graphical items is an indication of their importance. Scroll bars allow the designer to interact with the design by simulating changing inputs. Wires between information items define specific functional relationships. For example: The designer thinks about how one piece of information will control the importance of another, or how one piece of information will contribute to another, or how when these pieces of information combine to control another event… This is all done without conventional "formulas". The designer thinks in terms of curves and how curves might bend as the environment changes (curves controlling curves).

Even though the programming paradigm is completely different, we have recently held a two-day training class that was attended by SAF/XCO officials (not programmers). By the end of the class they were able to model a multi-variable, non-linear system using the KEEL language.

Development process <u>without</u> KEEL:

1. Domain expert describes the problem
2. Mathematician writes formulas (differential equations…)
3. Software engineer translates the formulas to computer code
4. Software engineer debugs the code
5. Software engineer writes a wrapper around the algorithm for testing by mathematician
6. Mathematician changes the formula - Go back to step 3
7. Software engineer documents the design (maybe)
8. Algorithm is integrated in a simulator for further testing by domain expert
9. Domain expert changes the description of the solution – go back to step 2
10. Environment changes – go back to step 1

Development process <u>with</u> KEEL:

1. Domain expert models and tests a solution to the dynamic problem from within the KEEL Toolkit using the dynamic graphical language
2. When testing is complete, the domain expert gives automatically created code (conventional computer language: C, C++, Java, C#, VB, Flash…) to software engineer for integration in application.
3. Environment changes – go back to step 1

Complex non-linear systems can be created in hours and days, rather than months and years. And once they have been created, they can easily be modified and extended.

Publications

KEEL Technology has been presented at numerous conferences including: at the Phoenix Challenge War Fighters' Conference in Monterey and Los Angeles, at IEEE High Assurance Systems Engineering Conference in Tampa, at a Sandia Labs sponsored Cognitive Conference in Santa Fe, and at ISA (Instrument Society of America) in Chicago. It has been on the cover of "Scientific Computing" magazine and been covered in the ISA "InTech" magazine. An article has been published on the Frost & Sullivan Defense and Aerospace website. There are numerous papers and application notes on the Compsim website that describe the technology in more detail.

Summary

KEEL Technology should have broad applicability to a wide variety of C4ISR and real-time information fusion applications.

Compsim

Compsim is a small, woman-owned business. Compsim is a "technology provider" (as defined by DARPA); not a "solution provider". Compsim created KEEL technology to address a wide variety of complex, dynamic, non-linear, inter-related problem sets. KEEL technology is covered by granted and pending patents 1) for the methodology for integrating driving and blocking signals, 2) for implementing a KEEL Engine in a device or software application, 3) for implementing a KEEL Engine as an analog circuit, and 4) for the KEEL dynamic graphical language (which is the only effective way to create KEEL Engines). KEEL Technology should be considered TRL 4-5. The technology is supported with a broad range of development tools. A variety of application demonstrations are available on Compsim's website: http://www.compsim.com.

Compsim LLC
PO Box 532
Brookfield, Wisconsin 53008

(262) 797-0418