

Applicability of KEEL[®] Technology to the 4D/RCS Reference Model Architecture for Unmanned Vehicle Systems

Tom Keeley
Compsim LLC
tmkeeley@compsim.com

Abstract

The 4D/RCS Reference Model Architecture for Unmanned Vehicle Systems provides a methodology for conceptualizing, designing, engineering, integrating, and testing intelligent systems software for vehicle systems with any degree of autonomy. Many of the components of this architecture require the integration of cognitive functions to provide this autonomy. KEEL[®] (Knowledge Enhanced Electronic Logic) technology provides a mechanism for defining and executing some of these cognitive functions.

This report targets organizations that are interested in developing products compatible with the 4D/RCS Reference Model and that may find value in KEEL technology for development, test, and execution.

1. Problem Statement

The 4D/RCS Reference Model Architecture defines “Value Judgment” and “Behavior Generation” services that will be distributed across a hierarchy of systems. As the 4D/RCS Reference Model is deployed in support of the Future Combat System, there will likely be thousands, if not millions of cognitive decisions required.

The 4D/RCS Reference Model suggests some approaches to package and deliver these cognitive functions. Section 4.4.12 (Knowledge of Rules of Mathematics and Logic) identifies the use of conventional programming concepts (IF, THEN, ELSE) to address logical problems that can be solved by numeric or symbolic algorithms. The report also recognizes that in the brain, rules can be represented by functional mappings implemented by neural nets.

Section 4.4.13 (Knowledge of Rules of Physics) recognizes both a static and a dynamic nature of the system. Section 4.4.14 (Knowledge of Value) recognizes the concept of information “value”. These two sections together require that the system support

the recognition of the dynamic changing of importance of information in a cognitive process.

2. KEEL Overview¹

KEEL Technology provides a mechanism for incorporating human-like reasoning in embedded systems and software applications. KEEL Technology encapsulates three concepts: The KEEL Toolkit and support tools, a model for decision-making / reasoning, and an execution engine that is created from the design developed with the KEEL Toolkit.

KEEL Technology has some specific benefits that may make it valuable to implementers of devices that comply with the 4D/RCS Reference Model. When a developer for 4D/RCS begins developing specific solutions, they may encounter several types of problems: 1. Complex problems where it is inefficient or virtually impossible to create a specific formula to define the solution. 2. Problems where neural net (pattern matching) solutions are at risk, because they cannot be adequately audited or understood by human supervisors. 3. Problems where a small memory footprint is required that cannot be addressed with alternative solutions. 4. Problems that must be addressed quickly and cannot suffer from long schedule impacts before a solution is provided.

To address the complexity issue, KEEL Technology incorporates a graphical development environment that allows the designer to develop the system by considering how the system should perform, not by writing textual formulas to describe functionality. This allows the cognitive problems to be addressed at a higher level, where the solution can be addressed without as much concern for the coded implementation. The graphics representing the KEEL source code are also dynamic (decisions continually executed during development). This allows the designer can see the impact of the design while it is being developed.

KEEL Technology can be considered in the “expert system” category, because it requires a human designer

to define the rules. Because it is a rules-based system, the decisions or actions can be viewed, understood, and audited. When the design is complete and satisfactorily evaluated in the design environment, it is translated to “code” which describes a KEEL engine. The KEEL engine is very small and does not require any support libraries. The rules are maintained in tables that are iteratively processed to determine a result or results. The API for the KEEL engine is very simple and appears to be compatible with the 4D/RCS Reference Model as defined in the 4D/RCS Section 3.14 (Example Data Structures).

2.1 Differentiation of KEEL

KEEL was based on a *process* for humans to address “wicked problems” rather than on a *mechanism* duplicating a specific structure. The term “wicked problem” was coined by Dr. Horst Rittel (UC Berkley) in the 70’s. Dr. Rittel addressed “wicked problems” associated with city planning and authored a paper process called IBIS² (Issue-Based Information Systems) to organize the information to address these problems. Rittel compared his “wicked problems” to “tame problems”. He suggested that tame problems could be addressed with formulas to get *correct* answers. He suggested that the objective for wicked problems was to find the *best* answer. He suggested that it would be difficult or impossible to write a formula for city planning.

So KEEL started from organization decision-making roots where people were involved in the process. KEEL concepts were not burdened with thoughts of a von Neumann “computer” structure or a model of the human brain. We focused on *process* rather than *mechanism*. Our objective was to create a system that could give the “same” result as a human expert, given the same input information. Because we are using a human expert to validate the results, the solution had to be visible and explainable to a human. It also meant it had to be explainable to the domain expert and not just a mathematician or software engineer.

The designer’s thought process is different when addressing a cognitive problem using the KEEL environment (as compared to a programmer writing linear code). Using KEEL technology, the designer is dealing with subjective judgmental decisions. The designer spends time defining how one piece of information relates to and interacts with another. The designer thinks of continuous functions (curves), rather than points in time. Verbal / textual languages, like conventional programming languages, do not easily depict continuous functions that need to be discussed by human domain experts. The programmer, writing

linear code, has to translate concepts from the non-linear domain to linear code, compile and package it before it can be tested.

The KEEL Toolkit graphical programming environment allows the creation of these relationships (curves) without writing textual formulas. We have authored a paper titled “Right Brain Programming – Case for a New Programming Paradigm”³ that explains the concept in more detail. Programming in curves does not require the translation of sensor information to human terms. While using human terms to define linguistic uncertainty (fuzzy logic) is helpful when taking inputs from humans, it does require the fuzzification / defuzzification process that is sometimes difficult to develop or interpret in detail.

This discussion is not intended to discourage the use of alternative techniques, but only to highlight how KEEL is different and where it might provide additional value.

3. Architecture Discussion

The 4D/RCS model appropriately recognizes a hierarchy of decisions or actions that must be exercised in both “Value Judgment” and “Behavior Generation” domains. KEEL engines are created as individual functional segments of code and would be embedded as methods and properties of classes according to 4D/RCS. They are architecture neutral; meaning that they can be implemented in any architecture or any level in the hierarchy. The system designer is responsible for positioning and scheduling the decision-making process. The 4D/RCS model defines the architectural considerations for performance and scheduling. While the KEEL engine doesn’t care where it fits within the hierarchy, the system designer will still have to determine whether performance is satisfactory. The 4D/RCS model also discusses interoperability between components. Because KEEL engines would exist as methods within classes, this is beyond the scope of the KEEL technology, but within the 4D/RCS model.

The 4D/RCS model appears to suggest a potential library of cognitive functions and KEEL engines should be compatible with this concept.⁴

4. Why Select KEEL Technology for Cognitive Processing

Devices that conform to the 4D/RCS model are likely to be involved in life or death decisions for either the devices themselves or their human associates. In these cases there needs to be a way to audit the decisions and actions that take place. It is also likely that individual models will need to be

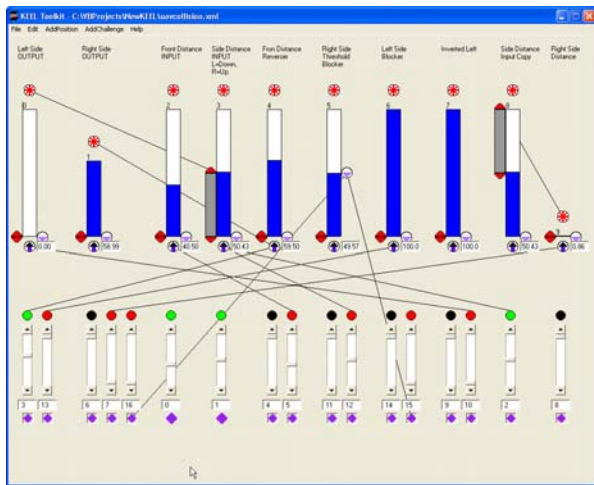
extended as new sensors or inputs are available to enhance the decision-making process. Both of these demands may be challenging to neural-net based solutions. Sometimes it is difficult to obtain good test data for neural nets. Insufficient or incorrect test data has the potential for generating incorrect results. Also, because the training effort is often time consuming, the impact of adding an input can be significant. In addition, since one cannot audit the individual decisions, it may be difficult to improve the process.

KEEL technology can be applied in cases where a human can define how inputs should be combined to contribute to an output or set of outputs. In very simple cases linear programming (IF, THEN, ELSE) might be used. However, when several pieces of information need to be integrated to determine the result, it is often difficult to generate the correct formula. When time and space also contribute to the decisions, the complexity of the system also increases. In these cases, using the KEEL toolkit, the solutions can be developed in hours, rather than days, weeks and months.

5. Services for the Domain Expert

The KEEL Toolkit uses a graphical paradigm for defining rules. Rules are 1. the identification of the importance of information and 2. the identification of relationships between information items. Outputs from the system are information items with a value. In this way an output “value” can “trigger an event” or can direct a “relative action”.

The graphical paradigm (as shown below) provides an easy-to-use mechanism for defining rules. It is not a flow charting process. KEEL information processing is a balancing activity that moves information through the engine until stability is achieved.



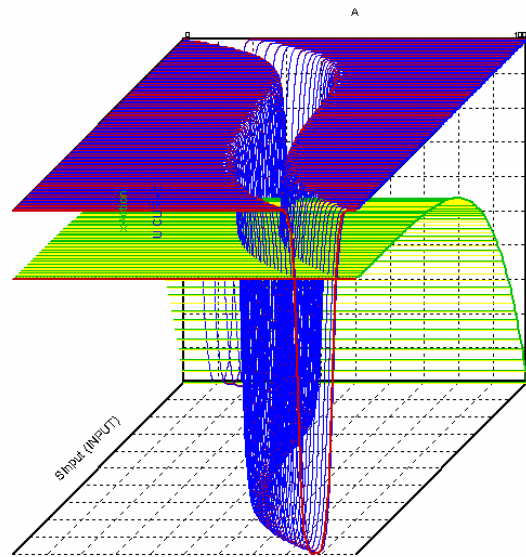
The user interface to the programming paradigm has some game-like qualities that make it “fun” to define logic. The user gets immediate feedback as the design is developed.

To assist the designer, the system is constantly monitoring for instability. These are information loops that could cause the system to become unstable.

The KEEL development environment runs on a Windows based PC. This provides a limited drawing surface upon which to develop the graphical design. In addition to a scrollable window to address this limitation, the KEEL Toolkit allows the definition of a number of views of the data. Unimportant items can be hidden until they are needed.

In a KEEL design, there are almost no limits to the number of inputs and outputs that can be incorporated in a cognitive design.

Programming in the graphical environment is primarily done with drag and drop functionality that allows the designer to wire together information items. Text is used to tag inputs and outputs by name. This textual information is used only to add information to the documentation. No text from the design is included in the resulting engine, except for array comments. By removing the comments, the resulting source code would be almost impossible to reverse engineer. This may be valuable from a security standpoint.



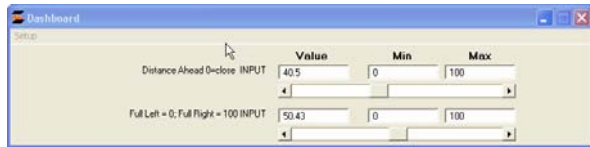
Numerous reports are available from the development environment. These provide complete documentation of each KEEL design, including worst case performance information.

Because the designer is interested in defining non-linear relationships between information items, the

KEEL Toolkit provides two and three dimensional graphing of information items (shown above).

In some cases exact probabilities will be known. While some of these exact numbers will be utilized in alternative linear coding solutions, techniques are provided to tune the curves to hit exact points.

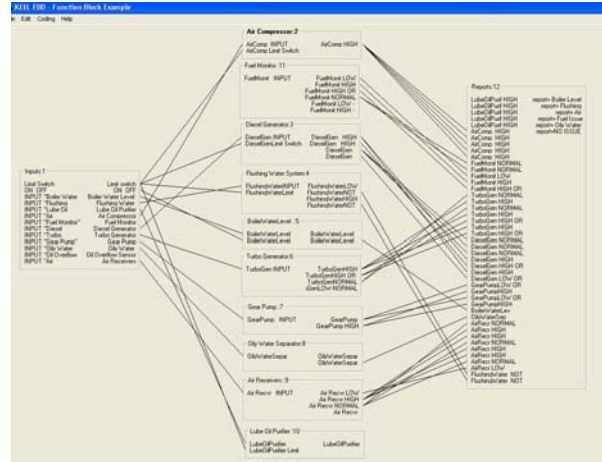
Since it is common for external sensor information to deal with ranges that differ from the normalized data that is supplied to the KEEL engine, the Toolkit provides a built in Dashboard builder that allows the developer to supply inputs in the range of the native sensor.



The source code for KEEL cognitive processes is the collection of graphical objects and their wired relationships displayed within the Toolkit. When providing the data to the software engineer for integration with the rest of the system, however, it must be translated to conventional source code. Today, the KEEL toolkit can translate the graphical design to conventional C, Microsoft C#, Java, Visual Basic Version 5/6, Visual Basic .NET, Macromedia Flash, Macromedia Flash (Object) format, and PLC Structured Text. Other languages will be developed as necessary.

6. Services for the Software Engineer

Since KEEL engines are often asked to address very complex decisions, the problems are often distributed to multiple KEEL engines working together. Multiple KEEL engines may be used to manage complexity when a problem is just too complex to be addressed with a single monolithic design. A modular decision-making approach may be more reasonable. To support this distribution of cognitive processing, a number of services are provided to assist in the integration of multiple KEEL engines. This allows the KEEL engines to be developed separately and compiled together, while avoiding any naming conflicts. Of course when the decisions are distributed across multiple devices or subassemblies, this would not be an issue. The image below shows a screen shot of the KEEL Function Block Tool that supports the integration of multiple KEEL engines into a single unit. Each of the boxes in the image represents a separate KEEL engine. The wires show linkages between engines.



The KEEL Toolkit maintains all information from a design in XML format. The KEEL Toolkit also uses XML as an import mechanism for bringing some of the external definitions into the KEEL environment. XML based data structures are also used to package data from devices so actual data from on-line devices can be brought back into the development environment for off-line analysis.

The interface to the KEEL engine's decision-making function is very simple. The following code snippet is for C. It sets a busy flag, loads the inputs, and then repeatedly calls the decision-making function until the busy flag is set false, at which time the outputs are unloaded and used for control or messaging.

```
if (busy==0) {
    busy=1;
    /* (add code to load external inputs into argvalues() table) */
    /* (call adjustarguments for each argvalues loaded from external sensors.) */
    while (busy) {
        busy=doDecisions();
    }
    /* (post the output values or derived values from modposvalues(), posvalues(), */
    /* thresholdvalues() to external control functions.) */
}
```

Other services are provided that allow data items to be rearranged and index values to be fixed. This allows most external glue logic to remain stable as new inputs are added to the system over time.

7. Compatibility with Alternative Cognitive Solutions

KEEL technology should be considered as just one possible solution in the Value Judgment and Behavior Generation areas. Straight line IF,THEN,ELSE code may be appropriate when the problem is straight forward, well understood, and can be modeled with a specific formula. Neural-nets may be the best option when the relationships between data items are not well understood and where there is appropriate training data available. Fuzzy logic appears to be a good fit when humans are asked to explain data to a computer and

linguistic terms are used to describe the situation. Learning algorithms integrated into neural-net based hardware devices may also be appropriate.

In a KEEL-based system, long term and short term memory is maintained external to the KEEL engine where it can be managed in any way appropriate. This differs from some neural-net based packages that may integrate “back propagation” into the solution. In a KEEL-based system, this allows *learning* (or more appropriately *adapting*) to be managed in the manner best suited to the problem being addressed. Some examples would be running averages or historical databases that are queried for specific views of data. In this way, the “system” (outside the KEEL engine) collects the information and provides the appropriate view as inputs to the KEEL engine. It is even likely that neural net, fuzzy logic and linear coded algorithms will provide inputs to KEEL engines and that the output from KEEL engines will supply data to neural net, fuzzy logic and linear algorithms.

8. Demonstrations using KEEL Technology

A number of demonstrations have been developed using KEEL technology. Several have focused on a hypothetical Weapon Control System that fits within the scope of the 4D/RCS Reference Model Architecture.⁵ This topic was developed to address two potential markets: Military / Command and Control Applications and the Electronic Games Market. While these demonstrations have not been packaged to conform with the 4D/RCS Reference Model, there does not appear to be any reason that this could not be done.

Since KEEL Technology focuses on the generic topic of cognitive processing, it is applicable to a wide variety of applications. Additional applications have been developed for automotive, medical, and financial processing.

A list of available application notes can be found on Compsim’s web site: <http://www.compsim.com>.

9. Summary

Compsim has developed KEEL Technology to address a wide variety of applications.⁶ It has focused on cognitive aspects of the target problem through the development of the KEEL Toolkit and associated tools, and on the needs of the software engineer that will integrate the KEEL engines into the complete solution.

We believe that KEEL technology has significant advantages when complex cognitive problems need to be addressed in a way that they can be audited by humans and incrementally improved. Because it is architecturally independent, it appears to fit well within the 4D/RCS Reference Model Architecture.

Compsim is a small woman-owned research oriented company that has developed KEEL Technology and created a patent portfolio around it. Compsim’s patent portfolio covers the development environment, the decision-making model, and the resulting engine architecture that results from the design.

Compsim is interested in partnering with companies with specific domain knowledge that have the ability to exploit KEEL Technology in the market place. In this line, Compsim is interested in licensing arrangements with those companies.



Compsim LLC is a provider of next generation cognitive technology for application in automotive, industrial automation, medical, military, governmental, enterprise software and electronic gaming markets. The company is headquartered in Brookfield, Wisconsin.

Compsim LLC
PO Box 532
Brookfield, Wisconsin 53008
(262) 797-0418
<http://www.compsim.com>

¹ “Decisions and Actions in KEEL”, Compsim,

http://www.compsim.com/papers/Decisions_and_Actions_in_KEEL.pdf

² “Issues as Elements of Information Systems”, Werner Kunz and Horst Rittel, Working Paper 131, July 1970

³ “Right Brain Programming – Case for a New Programming Paradigm”, Tom Keeley, Compsim,

http://www.compsim.com/papers/Right_Brain_Programming.pdf

⁴ “KEEL Engine Architectures – Discussion of Architectural Options for KEEL-based Solutions”, PowerPoint with Notes, Compsim, 2004

⁵ “KEEL-based Weapon Control System Status – January 2004”, Compsim, 2004

⁶ “Knowledge Enhanced Electronic Logic for Embedded Devices”, Compsim, 2003,

http://www.compsim.com/papers/About_KEEL.pdf