Measuring Compsim's Knowledge Enhanced Electronic Logic (KEEL®)[i] Technology against Autonomous Technology Metrics

1. Deterministic / Auditable Behavior

Addresses the "trust" issue. In this case, auditability means the ability to reverse engineer decisions or actions that are made by the autonomous system. For a system to be auditable, it must be able to show with mathematical precision how every information item is valued and how those values are integrated to make a decision or take an action.

KEEL Technology provides 100% explainable / auditable behavior. Two processing models are provided.

By feeding input variables into the KEEL Toolkit, this information can be used to drive the KEEL Dynamic Graphical Language, where it is easy to trace exactly, and see how, information is integrated to make decisions and allocate control signals. Using KEEL you can show custom "views" of data including "parent view" (where one can see all the factors that drive an information fusion point), or "child view" (where one can see all of the information fusion points that are influenced by an input factor). This visual representation of data can also be driven by real-time data from simulation, or black box recorded data, so one can watch the system think and trap on decisions or actions. It can also be used to trace exactly why decisions were made so that tuning (or changes) can quickly and easily be accomplished.

Two processing models are provided (this is selectable by the designer): One view (the auto-generated code is slightly smaller) operates like a pure analog circuit where information is "balanced" until a stable set of inputs and outputs are achieved. The worst case cycle count is documented in the source code. During normal operation inputs and outputs whose data does not change is only processed two times to confirm stability. This processing model is more appropriate for systems where many items remain stable as it uses fewer processing cycles of the CPU.

The second processing model (selected by the designer) auto-generates code that is slightly larger. The KEEL Toolkit evaluates the model and calculates an optimal processing model (one additional table). This model is more deterministic as every input and every output is processed every "cognitive cycle".

Knowledge
Enhanced
Electronic
Logic

2. Support for Adaptive Behavior

Addresses the basic need for autonomy.

> **KEEL Technology was developed to provide adaptive behavior for small embedded real-time devices.**
>
> In a human, adaptive behavior is processed by the right-hemisphere of the brain where alternatives are balanced and information is collectively processed to address sometimes conflicting goals.
>
> The KEEL "dynamic graphical language" facilitates the creation of complex (dynamic, non-linear, inter-related, multi-dimensional) problem sets.
>
> Using the KEEL language, one models and tests how information is to be interpreted, rather than how to solve specific problems. One "thinks in curves" as one describes the relationships between information items.
>
> The dynamic nature of the graphical language makes it relatively easy to create and modify the functional relationships between information items.
>
> As one stimulates inputs, one can get immediate feedback on how the system processes information.
>
> It is easy to integrate temporal information into the models. As the model is being created and tested with the KEEL Dynamic Graphical Language, a KEEL Engine is being created behind the scenes that can be captured in a conventional computer language at the click of a menu item.
>
> The same model can be coded in multiple languages if desired: one for an embedded device, another for M&S, another for a training system and another for a demonstration system. This can also be described as: design once, deploy many times using your code of choice.

3. Support for Autonomous Learning

Some systems may benefit from the ability to integrate items / concepts that have never before been seen or sensed. Learning is different than adapting. When learning methods are used, new terms or concepts need to be integrated into the system without any manual / human intervention. Example: a UxS encounters a new type of object. Learning happens with a trial and error process and an associated categorization process as the attributes of the object are accumulated.

KEEL Technology can be used as part of a Learning System, where learning is defined as changing how the system interprets information.

An example of this would be where external trending information is used to bend a curve, or change the importance of an internal factor.

From the purest perspective, KEEL is "adaptive". It does not learn.

Inputs and outputs must be defined in advance.

4. Supports Non-linear Control

The exertion of many control decisions requires decisions of "how much" or relative control adjustments.

KEEL Technology supports the definition of adaptive, non-linear control as one of its key attributes (all without demanding higher level mathematics, or software engineering code / debug efforts).

The KEEL "Dynamic Graphical Language" could be defined as a new type of mathematics, where one is describing complex functional relationships between information items (the definition of calculus).

Because it is easy to "see" the results of "wiring" information items together using simple drag and drop methods, it is easier to "see" functional relationships and stimulate inputs and "see" the outputs immediately (interactively). This makes it easy to think in terms of the entire system and how the "system" operates, without going through multiple translations using conventional approaches.

Conventional approach: The SME conceives of a problem solution. The mathematician translates the concept into a formula. The software engineer translates the formula into code. The solution is tested; returning back to the SME. Iterate the cycle until the solution matches the concept. Then test the concept.

With KEEL, the SME can create and test the model. The SME gives the code to the software engineer to integrate into the system.

Knowledge
Enhanced
Electronic
Logic

5. Support for a Dynamic Environment

Seemingly obvious, the technology must support operation in a very dynamic environment were potentially a continuous re-evaluation of "behavior" is needed whenever any change is detected.

Dynamic problems are supported with the KEEL "Dynamic Graphical Language". The KEEL DGL emulates the KEEL Engine that processes dynamic information.

Dynamic problems are difficult to articulate and test. The KEEL Dynamic Graphical Language and the KEEL "cognitive" engines were designed to address complex (dynamic, non-linear, inter-related, multi-dimensional) problem sets.

KEEL Engines "balance" inter-related information items to generate adaptive control signals. KEEL Engines operate as if they were analog computing engines.

6. Handles Temporal Data

The technologies ability to handle time and space components of decision-making, as well as the aging value or trust of information.

Time and distance are common input variables that participate in the decision-making / action taking process. To KEEL Engines, these are just additional pieces of information that can be used to define how the information is to be interpreted.

Aging, or missing information can also be addressed with KEEL. One simply defines how the SME wants to treat missing or aging information. One might decrease the value of a piece of intelligence over time. Different problems, different degradation rates.

7. Support Human-in-the-Loop Decisions (Configurable Policies)

Addresses the "trust" issue and the need to be able to adjust the behavior in real-time.

It is common to integrate configuration inputs into a KEEL design. These configuration items could be adjusted by a human (if desired), or could come from a database or any other system.

By adding configuration inputs, one can enable external control of almost any items in the KEEL Engine. These could be simple human authorization functions to autonomous behavior determined by the KEEL-based system, or they could be parameters that change how the system thinks.

KEEL allows unlimited flexibility in this area.

It is likely that a system will go through several phases: 1) provide advice to humans (explaining why, how, how much, when); 2) autonomous behavior with authorization by humans; 3) autonomous behavior audited by humans in after mission reviews.

8. Support for "Qualified Data"

The technology must have a way to qualify data that may include partial data, or corrupted data.

Data for a UxS may be collected from a variety of sources. Some of these sources may be able to provide a confidence factor.

To KEEL Technology a confidence factor is just another piece of information that can be used to control how that information impacts the decision-making process.

There may be thresholds or curves that define how a confidence factor impacts the models. All of these services can be easily integrated into a KEEL-based model.

This is similar to a system where communication with the system is lost. One models this behavior, just as if a human was making the decisions in real-time. The SME defines how the system is to react if communication is lost, or if one or more sensors stop working.

9. Ability to Handle "Surprise" Situations

Addresses unexpected occurrences. The ability to handle "surprise" situations is identified as a needed attribute for technical solutions.

Unlike conventional approaches where one encodes how to solve a problem, with KEEL, one defines the functional relationships between information items.

This is a significant paradigm shift in programming. The code for a KEEL Engine is auto generated. No thought needs to be given to a "formula to solve a problem", one only identifies the outputs (control signals) and the inputs (factors influencing the control signals) and how they are inter-related (accomplished with the KEEL Dynamic Graphical Language).

Using this methodology, one is automatically capturing how to handle surprise in a 100% explainable / auditable manner.

Testing in a simulated environment, one can stimulate any set of events. One can trap on any behavior and see why / how / what happened.

10.Deterministic Performance

The time it takes to integrate information must operate within a known limited amount of time. Only a certain amount of jitter is acceptable.

KEEL decisions and actions are 100% deterministic in terms that the same input will always yield the same output. KEEL Engines created with the (A) processing model runs the same instructions for every call yielding very little jitter.

The KEEL (A) processing model has a "sequence table" that identifies the optimal processing sequence to address changes to any/all inputs.

11.Brittleness

The Task Force report highlights the issues of "brittle" code that can be broken or tricked by unexpected situations or breaches of boundary conditions.

By capturing "how to interpret information using the KEEL development methodology", rather than "developing solutions to specific problems", a solution is created that avoids problems of scripted, sequentially processed "code".

Brittle code is hard to manage. Adding instructions in the middle of scripted code can have unintended consequences. KEEL "code" is auto-generated. The KEEL Toolkit "watches" the development and displays warnings/blocks design issues, like circular references.

12.Extensibility

Addresses the need to have a technology where autonomous policies need to change, or when new sensors or data sources need to be added to the design.

KEEL Technology and the KEEL Dynamic Graphical Language makes it easy to extend models without starting over.

To add a new sensor, one simply adds a new input to the system and wires it into the web to define new functional relationships. The user gets immediate feedback to insure the "system" operates correctly. If not, double click on a wire to remove the functional relationship and wire it somewhere else.

When extending models in KEEL, one is looking at the entire system and observing how the entire system responds.

13.Effort to Implement / Extend

Addresses the cost issue and issues associated with new scenarios presented by adversaries. It is expected that autonomous systems will need to be extended many times over their life cycle.

Defining functional relationships using the KEEL Dynamic Graphical Language is accomplished by a simple drag and drop. Complex relationships are created using this simple process.

As soon as a wire (that defines a functional relationship) is dropped on the screen, the underlying KEEL Engine that is developed behind the scenes is operational. Removing a functional relationship is accomplished by double-clicking on a connection point. Testing the new relationship is available immediately.

In complex models where the best solution is not readily apparent, having the ability to test different configurations in minutes is helpful to explore alternative models.

Development of KEEL models is commonly an iterative process that may include new inputs and new outputs through the life of the model.

It is a common practice to include configuration inputs in a design, so the model can be tuned to achieve the optimal policy for a given situation. The configuration parameters can be "locked" to restrict or limit run-time configuration.

When starting a new project, one should begin testing within the first minute after starting the design.

14.Ease of Use: API (Application Interface)

Addresses issues related to integration of autonomous features into new and existing systems.

KEEL Engines have a very simple API. Load the inputs; execute the information fusion process; retrieve the outputs.

KEEL Engines can be delivered as one or two functions (depending on certain functionality utilized) in many standard computer languages; among them C, C++, C#, Objective C, Java, Flash Actionscript, Octave (MATLAB), Python, SCILAB, Visual Basic, and others.

For object oriented languages you are provided with a class with methods to load, process, and retrieve values.

KEEL Engines can also be packaged as Web Services, where the source code is provided.

A new language or packaging scheme can usually be developed and added to the list of languages in about two weeks.

15.Learning Curve

The time from technical introduction, to the time when one can be productive, to the time when one is an expert in applying the technology. Includes special skills needed as prerequisites.

There are no prerequisites to learning how to use the KEEL Dynamic Graphical Language. One can expect to be productive after a two day class / workshop. One can expect to be an expert after 6 months of use.

A KEEL workshop includes a one day introduction to the KEEL Dynamic Graphical Language and a one day discussion of architectural considerations which include: KEEL Engine code walkthrough, an introduction to other KEEL System Support Tools, and play time creating non-linear systems.

Sometimes a third day is spent focusing on a customer problem, creating and testing a working model.

16.Ease of Use: Development Tools

Addresses life cycle cost issue

KEEL development, using the KEEL Dynamic Graphical Language can be compared to an interactive computer game. The graphical / interactive nature of the language is "fun" because it is interactive.

The development process is "very" rapid.

3-Dimensional graphs help evaluate and document complex models.

Because the user does not have to translate concepts into formulas, or create code to interact with the models, most of the time is spent testing the system, rather than debugging code.

Once one becomes familiar with the "language" one quickly learns how to trace the dynamic models and "see" the logic.

17.Ease of Use: Problem Exploration

Addresses the need to provide a way to investigate solutions to complex problems that are not well understood.

It is very easy to explore decisions and actions. One simply traces wires. Clicking on a connection point highlights the related functional relationship.

Named "views" allow the user to focus attention on a certain part of the design. Parent views allow the user to select an output (or information fusion collection point) and see all the logic driving that point. Child views allow the user to select an input and see all the logic driven by that input.

18.Testability

Addresses the "trust" issue.

KEEL-based designs target complex problem sets. Decisions and actions are 100% explainable and auditable.

KEEL models are first tested within the KEEL Toolkit. The language facilitates interactive testing. One can stimulate inputs and "see" how information propagates through the system to make decisions and allocate resources.

Once the design has been tested in the KEEL Toolkit, it can be packaged for M&S environments. A KEEL capability called "Language Animation" allows the simulated device to publish what it sees in almost real time and drive the KEEL Toolkit hosting the KEEL Dynamic Graphical Language.

This same capability can be used with "black box" testing, where recorded views of the data can be used to drive the KEEL Toolkit. Using Language Animation, the tester can trap on decisions and actions to review system operation.

19.Certifiable Code

Addresses the "trust" issue. Autonomous solutions are expected to be derived from computer "code" at some point. To certify the code, this code must be reviewed for boundary cases, unterminated loops, and good coding practices. It should be possible to quantify / qualify the effort needed to certify the code.

Code certification should be simpler with KEEL Technology than (many/all?) other approaches because the KEEL Engine code is "always the same" and "very small" (approximately 3KB).

KEEL Engines have a very small memory footprint.

Solving complex problems using conventional IF/THEN/ELSE code: The more complex the problem, the longer and more complex the code. This code is processed sequentially,

On the other hand, KEEL Engines capture the problem space as table data. The KEEL Engine "code" processes the data in the tables. The "code" is always the same.

The only exception is that KEEL Engines are optimized to eliminate code for certain functional relationships that are not used in the design. When certain functional relationships are not used, the tables are not included and the code to process the tables is not included.

The same KEEL code structure is used for all languages. This means that once the concept is understood, and the code has been validated, it should never have to be certified again.

NOTE: This is different than certifying an application or policy.

20.Certifiable Application

Addresses the "trust" issue. The process required to determine if the autonomous application satisfies the need must be evaluated. This is somewhat associated with the auditability requirement, where every decision or action needs to be evaluated. This means the ability to understand what impacts every decision, action, asset/resource allocation.

Certifying a KEEL-based policy or application is more complex than certifying the KEEL "code", but the KEEL Toolkit provides a number of services to support application validation.

The KEEL Toolkit supports external control of the language. The KEEL Toolkit also insures that the designs do not include circular (unstable) characteristics.

Language Animation that includes the ability to trap on decisions or relative actions is another technique that one can use to validate operation.

An understanding of "how KEEL Engines process information" is also helpful in establishing test criteria and test cases.

21.Compatibility with COTS (Commercial Off The Shelf) or GOTS (Government Off The Shelf) Solutions

Addresses the cost issue as the government strives to be independent of any contractor. Issues with COTS technology available to adversaries should be addressed.

KEEL Engines are platform and architecture independent.

KEEL Engines are provided as text files in the language of choice: C, C++, C#, Objective C, Flash ActionScript, Java, Octave (MATLAB), Python, SCILAB, Visual Basic and others, as well as two forms of "web services".

KEEL Engines do not have any dependencies on external libraries.

KEEL is proprietary technology that "could" be acquired by the DoD and be made GOTS or COTS.

KEEL Technology is covered by a series of granted US Patents.

22.Platform Independence

Independent of any hardware restrictions. Not tied to a particular manufacturer or hardware set.

KEEL Engines are platform independent.

KEEL Engines are delivered by the KEEL Toolkit as text files in the source code of choice, and can be used in the development environment of choice.

The small size of the KEEL Engines have been tested in 8-bit microcontrollers, allowing complex problems to be addressed in very small form factors.

23.Architecture Independence

Technology independence of system architecture: single micro, multiple segments on the same micro, multi-processor, parallel processing, distributed across multiple devices, independent of communications.

KEEL Engines are low level functions (or classes) depending on the language selected. The KEEL Engines can be placed anywhere they are needed to support information fusion.

KEEL Engines can be thought of as individual human experts that interpret information and exert control that they have been authorized to perform. Collaboration / information exchange techniques / timing / methodology is outside the scope of the KEEL Engines. These decisions are left to the system architect.

KEEL Engines can account for broken communication links or degraded information. The interpretation models are part of the design process that describe how to perform under degraded conditions; just like policies would describe how humans operate when the control structure degrades.

24.Ability of the DoD to Acquire Full Rights to the Technology

Addresses the cost to the government issue. Also addresses the ability of DoD to control access to the technology.

DoD can acquire full unlimited rights to KEEL Technology from Compsim.

25.Dependencies

Technology dependencies on "any" factor may impact the value of a potential approach. Dependencies on any hardware chip set, or software libraries, or memory, or CPU clock speed may limit the value of a solution.

KEEL Engines for deployment in devices and software applications are not dependent on any particular platform, library, or technology. Inputs and outputs are normalized values.

The KEEL Toolkit requires ADOBE Air and runs on PC (XP and beyond) and Mac.

Installation is accomplished over the internet.

The KEEL Toolkit requires an internet connection to access license validation and user's manuals.

The KEEL Toolkit requires at least 1024x768 display resolution. Multiple, higher resolution displays are beneficial.

Mouse and Keyboard are required to work with the KEEL Toolkit and the dynamic graphical language.

26. Weaknesses Specific to a Technical Approach

The identification of any weakness may impact the value of a particular approach.

KEEL is an "expert system" technology and requires a Subject Matter Expert to create the models / policies.

While this is identified as a weakness, it can also be used to "help" the SME describe complex problem sets, where he/she may not be sufficiently familiar with higher level mathematics that "might" enable similar solutions to be explored.

KEEL Technology is proprietary (covered by granted US Patents) that have, as yet, not been acquired by DoD.

[i] More about Compsim's KEEL Technology: http://www.compsim.com/AboutKEEL.htm