When Robots Kill
White Paper
Compsim LLC

Abstract: There are many business drivers that are causing businesses to pursue the development of "intelligent" robots. In the medical / human services area, robots will provide a support mechanism for an aging population. In the military space, the robots will provide a way to amplify military capabilities. Two physical approaches will be concurrently developed: biological systems and mechanical systems. There will also be developments in the virtual world. These are commonly referred to as "software agents" and deployed as intelligent applications.

If we think of robots as any kind non-human mechanisms that can perform tasks, then it may be easier to consider some of the biological entities as robots and not have to consider Frankenstein's monster as the only representation. This allows one to include trained bacteria or single celled animals in the class of robots. Given this definition, it is easy to consider their (biological robots) ability to kill. They are already being used to carry medications to specific locations in the body. The concept of biological warfare is well recognized.

In the mechanical domain, it may be more common to consider robots as mechanical representations that are displayed in the movies. However, robots in the industrial space have been around for years; they are the tools used in flexible manufacturing. They have mechanical arms that are used to perform repetitive tasks. They are used to reduce costs and keep humans out of harmful situations. In the military space they are now being deployed as unmanned aerial vehicles, unmanned underwater vehicles, and unmanned land vehicles. Now they are performing surveillance activities and sometimes (in a remote control mode) firing weapons. The concept of swarming has been used to describe how groups of these devices can work together to accomplish a task.

This paper is intended to address the next generation of these devices, where these devices will exert the next level of intelligence: the ability to exercise judgment on their own to accomplish goals that have been assigned to them. It will discuss some significant issues that must be addressed in describing policies that can be interpreted by robots and why policies prepared for humans are not sufficient. It will also suggest how this task of describing policies for robots can be addressed using Knowledge Enhanced Electronic Logic (KEEL®) Technology and compare this approach with others that might be considered for controlling these intelligent robots (if they are to be controlled at all). This is a significant issue, since eventually robots will be tasked with killing; whether the task is killing cancer cells, or perhaps killing an enemy. It is mandatory that humans retain some level of control, because, unlike humans, these robots will be mass produced. They will all (or at least a production run) be the same and exhibit the same reasoning capabilities. They will have the potential of addressing problems in an undesirable way, so *how* their reasoning capabilities are addressed is going to be a very important factor. This issue "should" impact the requirements as to how they are created.

Policies for Humans

"Policies reflect the 'rules' governing the implementation of processes."

Good Policies targeting humans are suggested to have the following characteristics:

- "Policies are written in clear, concise, simple language.
- Policy statements address what is the rule rather than how to implement the rule.
- Policy statements are readily available to the community and their authority is clear.
- Designated policy experts (identified in each document) are readily available to interpret policies and resolve problems.
- As a body, they represent a consistent, logical framework for action."[1]

The previous description makes some assumptions: First, it assumes that the problem can be described in "clear, concise, simple language". It also assumes that a policy can be understood by the humans being targeted. It also recognizes that there is some chance that the target audience will not completely understand the policy, because it suggests that "policy experts" need to be available to interpret and explain the policy.

The issue is that policies are trying to describe how complex situations are to be addressed. These situations are usually more complex than "If you see a stop sign then stop". Complex situations require judgment. They require the actor to interpret the situation and make multiple value judgments. Absolute statements are "almost" never appropriate. If they can be made, then they are the easy problems to address.

Humans address these complex situations collectively because they have learned how to participate in society. Individuals make their own decisions concerning how to comply with policies (laws) based on their own interpretation of the pros and cons of their actions. Some humans make very bad decisions for society as a whole. Society puts in place laws (policies) and creates a judicial system to interpret the laws.

This approach is not appropriate for robots, because one cannot take the chance that very bad decisions are mass produced. This would be a strategy for disaster.

Policies for Robots

If we treat robots as "machines", which is what they are, than their operation should be treated as such. Consider a machine tool, for example. A machine tool is designed to manufacture precision parts. This means that it has well scripted (and mechanically restricted) boundaries that limit what it can and cannot do. The machine tool incorporates feedback mechanisms that allow some adaptability, but these are limited to well-controlled domains. Any attempts to work outside these boundaries trigger alarms, or shut down the system, so corrective action can be taken. The script or program that defines the operation of the machine tool can be compared to a mathematical formula. There are explicit outputs to explicit inputs. There is no room for "interpretation" by the

[1] "Guide to Writing Policy and Procedure Documents"; http://www.ucsc.edu/ppmanual/pdf/guide.pdf#s

machine tool. Sensors transform real world information into explicit numeric representations of the environment. Formulas embedded within the machine tools provide exact directions about how they should perform.

We would not expect machine tools to accept complete part designs and process information using human language (English for example) description, without an underlying mathematical foundation. Using the English verbal / written language alone would yield inconsistent results. An analogy would be to give verbal orders to a number of skilled craftsmen, and then to expect them to build identical chairs. If one did try to create a human language description with sufficient detail to explicitly define all of the features, it would be extremely cumbersome and difficult to maintain.

In the Industrial Automation world, we have created a broad range of intermediate tools that help humans explain what they want the machine tools to do. CAD (Computer Aided Design) tools allow designers to draw pictures that can be translated by computers into formulas (scripts) that can be understood by machine tools. In some cases, these software tools will evaluate the designs for their manufacturability. In this way, one might think of the design analysis tools built into CAD applications as a way for the machine tools to interpret design policies. Once the design script has been given to the machine tool, however, the machine tool will just attempt to do what it has been told to do.

Now consider what we are expecting "robots" to do in the future. They will be asked to fight our wars, search through hazardous situations, and perform to meet tactical objectives. In performing these tasks they will have to determine how to solve problems on their own, or in groups, without the continuous support of human controllers helping them perform every move. The situations they face will be much more complex that those faced by the machine tool. They will be asked to solve problems and seek goals, while they encounter situations that they have never encountered before, including situations that their designers have never planned for.

We will also be demanding more of the robots than just their ability to solve problems on their own. We will (or should) demand that they can explain why they do what they do. The requirement is similar to the need to put data recorders ("black boxes") in airplanes and, most recently, into our automobiles. The black box provides a record of the information that is used when the airplane or automobile operates. It is used to determine the cause of problems or accidents. This information is subsequently used to create better (safer) airplanes and automobiles.

When a robot is given challenging goals (kill, protect, destroy…) it will have to balance risks and rewards. It will have to consider collateral damage, friendly forces, non-combatants, and environmental impacts. It may have to consider impact to tactical or strategic initiatives when it looses communication with higher echelons. Even at very low levels, the subassemblies of the robot may have to consider the quality and timeliness of data from sensors. It may have to operate with degraded services. It most certainly will have to operate as part of a team or group that may or may not include humans. It

may have to extract a value for its own survival. Given that the robots will be operating in a continuously changing environment, the policies must be flexible enough to enable the robots to accomplish their tasks. But they cannot be given the flexibility of interpretation allocated to humans. In this light, the robots must be able to efficiently operate in a manner that can be completely audited by humans.

It is probably impossible to write a document in human terms that completely and explicitly defines all the actions performed by a human that would be acceptable for a machine tool (robot). For example: documenting a political decision or a statement of government policy. When a political action is taken there may be some form of explanation provided. That statement (if there is one) is then decomposed by the news media. Often multiple points of view are stated that speculate on the justification of the action: the action was done because…; they included these artifacts in the decision; these other actions carried some weight; this other problem may have been considered… The ultimate result is that no human really knows why an action is taken. Even the person that performed the action cannot explain *why* in an explicit manner that would prove acceptable to the machine tool builder. It would also be almost impossible to re-create exactly the same scenario so that one would be sure that the same (or different) action would be taken the next time.

Robots are Computers

We might step back and look at robots from another light. From an intelligence standpoint, they are just microprocessor based devices. They have computers in one form or another that process instructions according to some kind of von Neumann architecture (common computer architecture with a Central Processing Unit and Memory). They process information according to programs that are sequentially processed. If we consider that robots are just computers then one might suggest that we have "many" computer languages that can be used to translate human ideas into formats that can be processed by computers. We just compile the programs into machine language that the computers (robots) can understand, and we can assume that they will do what we want them to do.

The scenario above might be acceptable IF we limited the challenges of the robots to the realm of machine tools, with well-defined boundaries that limit their scope. This will not be acceptable, however, if these robotic devices of software applications are to achieve their expectations.

Alternative Points of View

The dominant point of view in certain research organizations is that we must develop systems that "learn". I believe the basis for this reasoning is that the problems that we are asking these systems to address are too complex to describe, therefore the systems themselves must be able to learn on their own how to solve them. The problem with this approach is that we may not define the problems appropriately, so the way that the "systems" address them may not prove acceptable. Remember robots will kill. They

may not consider the collateral damage, friendly forces, non-combatants… appropriately. If they are given the ability to "learn on their own" then they will develop their own value system. If they learn by just matching patterns, then they may be limited to the patterns they have observed. If they collaborate amongst themselves, then bad decisions can propagate similar to a virus. This approach is probably not sufficient or acceptable, in light of the problems that might be encountered. Most certainly, if we mass produce robots for war, then they must start out well-trained.

One approach that might be considered is that all life and death (significant) decisions will be reviewed by humans. The problem with this approach is that it would be almost impossible to have the robot "explain" the reasoning in real-time, and then get humans to react in a timely manner. A simple example can be shown by building a robot that maneuvers around a corner. The robot starts to slide at the same time multiple pedestrians jump in front. An instantaneous response is needed. There is no time for "discussions" with humans.

Another approach being promoted recently is the use of "rules engines". The thought is that one can effectively describe rules to complex models with IF | THEN | ELSE logic. Other than just suggesting that it is difficult or impossible to address complex, dynamic, non-linear problem sets with IF | THEN | ELSE logic, one can attempt to suggest that a human is an analog system, not a von Neumann computer. There have also been recent discussions suggesting the use of analog computers as a better approach for solving cognitive problems[2]. Consider the example of the robot maneuvering around the corner again, simultaneously controlling the accelerator, brakes, and turning radius, while considering slipping, pedestrian traffic, speed, surface conditions, performance characteristics….and that all of the variables should really be treated as analog states (continuous variables).

One might choose to allocate these types of problems to higher level mathematics. Humans invented differential calculus to solve these kinds of problems. Differential calculus provides a way to describe complex, non-linear, functional relationships on a piece of paper. One problem with this approach is that it limits the definition to those people who can understand the formulas. This may not be the domain expert or the one responsible for the policies. This would require that the policies to be described using the human (English or other) language, and then be interpreted by the mathematician who would be responsible for developing the formula. Then the equation may have to be passed to a different person: the software engineer that may have to interpret the equation and transform it to a computer program. Testing the solution, if it is to describe a complex policy, would be difficult. Auditing the real world results of actions would be difficult. The computer program would likely be very complex, and the structure of the software implementation would probably bear little direct resemblance to the original policy as it was described by the policy maker.

[2] Hedger, "Analog Computation: Everything Old is New Again" Research & Creative Activity, Indiana University, April 1998 Volume XXI Number 2

Summarizing the Requirements

1. A methodology must be provided that allows the domain expert to define the policy with sufficient granularity so that it can be exactly translated into a form that can be explicitly executed by a robot.

2. The methodology for describing the policy must support the efficient development of policies to address complex, non-linear scenarios.

3. The execution engine for the robot that will execute the policy must be suitable for embedded, real-time operation.

4. The methodology must be completely understandable so it can be efficiently tested before deployment.

5. Robot performance needs to be audited after deployment.

6. The efficiency of the entire policy life cycle must be considered (design, test, deploy, audit, extend).

7. The methodology must be architecture independent so it can be deployed on a variety of platforms, and in a variety of situations.

Introduction to Knowledge Enhanced Electronic Logic (KEEL®) Technology

KEEL is a new "technology"; it is not just a new "tool". It is a fundamental new method to process information. In one implementation it processes information on a digital computer as if it was executing on an analog computer. In robotic implementations the very small memory footprint may be a key advantage[3]. In another mode, the model can be implemented as an analog circuit (when very high performance is needed). Creating and packaging policies (rules of engagement) using this technology is accomplished using a new "dynamic graphical language". The KEEL graphical language is not a "flow charting language" where one defines data flow. Items that impact the policy 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 the domain expert to interact with the design while the models are being developed. By simulating changes to inputs, one can "see" how the policy will be interpreted, (the designer can "see the system think").

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

[3] 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 the changing environment. 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. The KEEL dynamic graphical language could also be described as a new form of mathematics; one that doesn't focus on the underlying formulas, but one that focuses on how the formulas are supposed to perform.

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 policies (designs). The KEEL Toolkit monitors the design as it is being created and warns the designer of the instability.

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.[4] This iterative processing of information is similar to how humans interpret

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

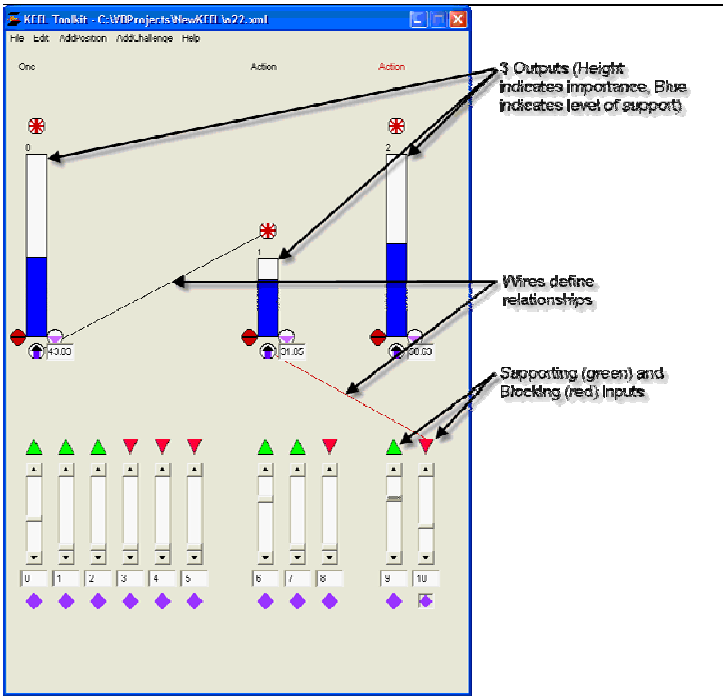
policies; by interpreting information and balancing alternatives while considering the impact of alternatives on policy directives.

One service that can be automatically integrated into robotic solutions allows for the decisions and actions of the robot to be audited. Real world inputs observed by the robot can be published (as an XML file). This file can be read by the KEEL Toolkit 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.

General KEEL Concept Development Process

1. Identify the outputs (control variables – actions that might be taken while following the policy) that will be driven by the system.
2. Identify the inputs (information items that will ultimately drive the outputs – items that are to be interpreted by the policy).
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 policy should be interpreted. 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, a successful two-day training class for SAF/XCO (Office of the Secretary of the Air Force) officials (not programmers) was held. By the end of the class they were able to model a multi-variable, non-linear system using the KEEL language.

Policy Development process without KEEL

1. Domain expert describes the policy in a written document
2. Mathematician interprets the written document and writes formulas (differential equations…) to describe the policy
3. Software engineer translates the formulas to computer code
4. Software engineer debugs the code representing how he interprets the formula.
5. Software engineer writes a wrapper around the algorithm representing the policy for testing by mathematician
6. Mathematician changes the formula or explains to the software engineer how the formula was incorrectly interpreted - Go back to step 3
7. Software engineer documents the design (maybe)
8. Algorithm representing the policy is integrated in a simulator for further testing by domain expert
9. Domain expert changes the description of the policy because testing highlighted missing components or because of a new scenario – go back to step 2
10. Environment changes – go back to step 1

Policy Development process with KEEL

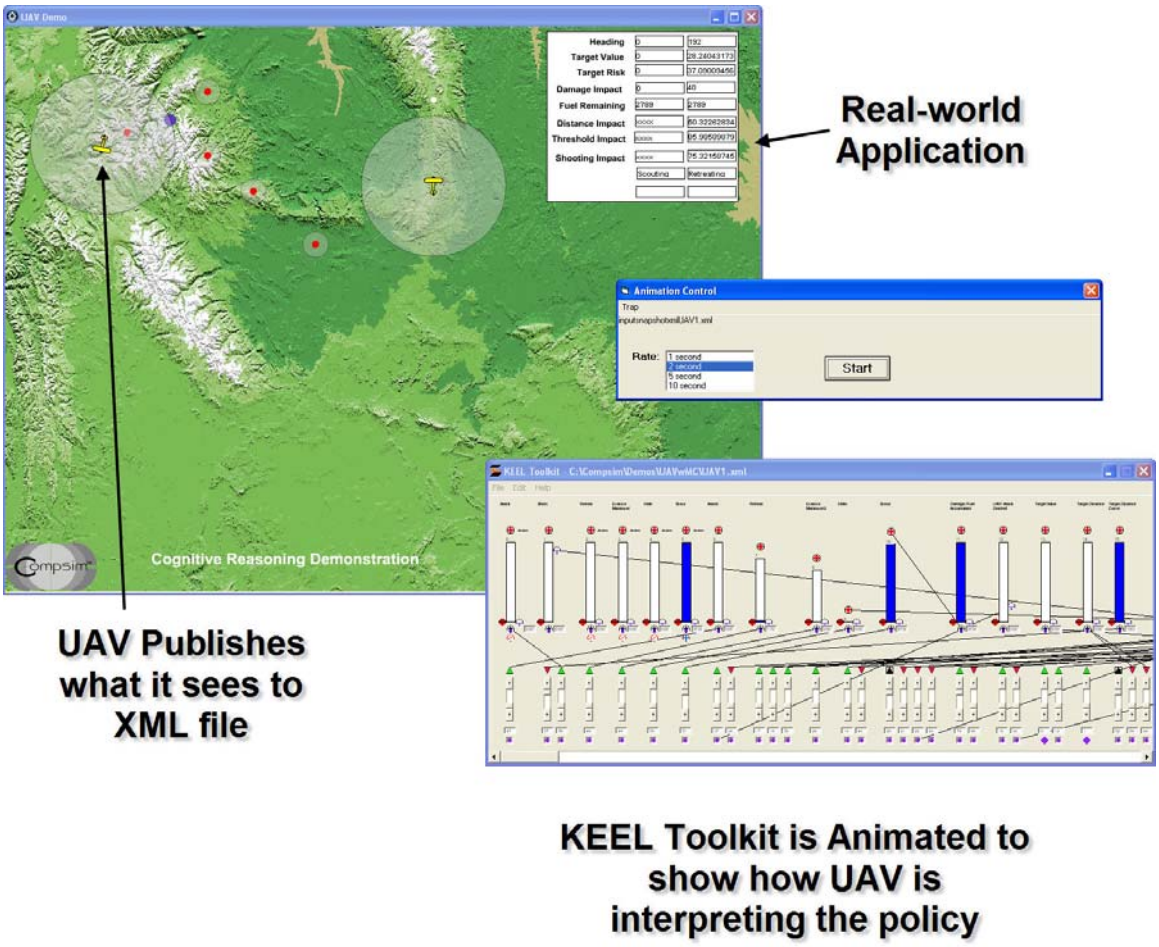
1. Domain expert models and tests a solution to the dynamic problem from within the KEEL Toolkit using the dynamic graphical language
2. When testing is complete, the domain expert gives automatically created code (conventional computer language: C, C++, Java, C#, VB, Flash…) to software engineer for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

Auditing Real World Policy Interpretation

Whether the policy is being evaluated in a simulation before deployment into the real world, or after deployment, it is likely that performance according to policy will need to be audited. This is because of the complex situations that will be addressed by the robot. It will likely be that in a simulated environment all situations will not be completely tested. Also, simulations may not take into effect all of the modes of failure of all of the hardware components. The "black box" equipment in conventional aircraft is intended to provide an audit trail for equipment failure and pilot error when there are problems with the aircraft. A similar type of situation will be required to support robotic operations.

The following image shows a UAV that is publishing its view of the environment while it operates in a simulation. The inputs are published to an XML file that is subsequently read by the KEEL Toolkit that is running the same policy model that exists in the UAV. The domain expert can "watch" the UAV process the policy in real time. Alternatively, if the UAV information is saved to a "black box" file, it can be reviewed at a later time.



Summary

The following is a list of the requirements previously mentioned, along with an explanation of how KEEL supports the needed functionality:

1. A methodology must be provided that allows the domain expert to define the policy with sufficient granularity so that it can be exactly translated into a form that can be explicitly executed by a robot.

    - The KEEL dynamic graphical language allows policies to be described as continuous curves that can be interpreted by the robotic equipment as if they were discrete formulas.

2. The methodology for describing the policy must support the efficient development of policies to address complex, non-linear scenarios.

    - The KEEL dynamic graphical language provides an easy way to document how information is to be processed without resorting to complex formulas.

3. The execution engine for the robot that will execute the policy must be suitable for embedded real-time operation.

    - KEEL Engines that represent the domain expert's design are high-performance, small-footprint functions.

4. The methodology must be completely understandable so it can be efficiently tested before deployment.

    - All functionality is displayed graphically. By tracing the wires one can see instantaneously how different and potentially conflicting data items are interpreted.

5. Robot performance needs to be audited after deployment.

    - Services are provided as part of the KEEL Toolkit such that real world data can animate the graphical language so that decisions and actions can be traced to their cause and justification.

6. The efficiency of the entire policy life cycle must be considered (design, test, deploy, audit, extend).

    - The design is interactive, while it is being developed. This allows it to be tested during the development process. Conventional code is created automatically, thus avoiding human typing errors. Deployment is handled by providing text files in the source code language of choice for easy integration into any IDE (integrated development environment). Auditing is provided with the ability to easily review the real-world interpretation using black box file reviews. Visually observing the importance of information and relationships allows complex scenarios to be reviewed with relative ease. Data is absolute so there is no human interpretation of the results required. Designs can easily be extended, by inserting new items and linking them into existing policies. There is no need to start over every time.

7. The methodology must be architecture independent so it can be deployed on a variety of platforms and in a variety of situations.

    - KEEL Engines are architecture neutral. They are simply cognitive engines created in the programming language of choice. The system architect has

complete control over the marshaling of information and the scheduling of policy interpretation and execution.

In summary, KEEL Technology provides a unique solution that satisfies the requirements of robots that are given the responsibility to seek solutions to problems on their own. While this paper focuses specifically on autonomous or semi-autonomous robots, it can be applied to any type of next generation application where information needs to be interpreted and acted upon, especially when the applications are dealing with complex inter-relationships, when it is absolutely necessary that humans are in ultimate control of the situation, and where the mass production of inappropriate decisions and actions can have major human, economic, and political impact.

KEEL Technology is only available from Compsim LLC.

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. 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.

Contact:

Tom Keeley
President & Technical Visionary
Compsim LLC
1040 Thornridge Ct
Brookfield, Wisconsin 53045-4522
(262) 797-0418
tmkeeley@compsim.com
http://www.compsim.com