Command and Control Architecture Using KEEL® Technology for Loosely Coupled Systems

Tom Keeley
Compsim LLC
June 2004

KEYWORDS

Command and Control Systems, Decision-Making, Loosely Coupled Systems, Agent Based Systems, Cognitive Technology, KEEL Technology, Neural Nets, Fuzzy Logic, Autonomous Operation

ABSTRACT

Loosely coupled system architectures have been suggested to respond to systems that can degrade without completely shutting down when segments of the system become disabled. This has led to research in autonomous and semi-autonomous system behaviors. There has been considerable research in the development of cooperative agent techniques based on belief networks and social paradigms. This presentation focuses on how a control hierarchy of loosely coupled agents might be accomplished by controlling the beliefs and sensitivity of the subservient agents. The suggestion is that while there is a value in a system composed of loosely coupled agents in its ability to withstand failures of individual components, the command authority still needs a mechanism for directing the subservient agents as the goals of the overall system change. This presentation introduces a model for describing rules for agents that can be tuned by a command authority. This will allow the command authority to tune the tactical resources according to a strategy, while still allowing the agents to react to tactical situations.

INTRODUCTION

Command and control (C2) systems have been articulated in the government arena as those that "can coordinate widely dispersed units, receive accurate feedback, and execute more demanding, higher precision requirements in fast moving operations"[1]. These same requirements can be applied to commercial business and operational functions as these entities attempt to operate in dynamic business environments. In government and military applications these systems are hybrid human and mechanical systems that are distributed across a battlefield and around the globe. Humans operate as both sensors and actuators in the system. They are supplemented with physical mechanical and electrical sensors and actuators. Humans, when they are participating as part of the system are operating as autonomous or semiautonomous engines. The hierarchy of command and the networks of communications create the conduit through which control must be exerted. The disjointed human element of these systems provides a loose coupling. The untethered operation of humans and their ability for independent thought and action are examples of loosely coupled systems. When humans work together they are operating as parts of a system. This loose coupling provides both an advantage and a disadvantage for command and control systems. It has long been recognized that loosely coupled systems are more flexible and can withstand component failure better than a tightly coupled or large monolithic system. The loss of an individual component will not (normally) cause the entire system to fail. This is especially important in complex systems where it has been suggested that the system will be operating most of the time in some kind of failure mode[2]. The issue addressed in this paper is how to control the loosely coupled systems.

There has been a vast amount of work done that contributes to the bank of knowledge on how to support the distribution of command and control across large loosely, coupled systems. This continues to be an area of research and development by a large number of entities. Organizations like MIMOSA (Machinery Information Management Open Systems Alliance) focus on shared information with a maintenance focus. FIPA (Foundation for Intelligent Physical Agents) addresses interaction protocols. OPC (OLE for Process Control) is an industry driven activity using Microsoft's object linking and embedding model. UPNP (Universal Plug & Play) / JINI are initiatives driven by Microsoft and Sun respectively, that focus on ease of interoperation. All of these groups are trying to develop de facto standards that will allow components of a system to interoperate. (In this paper we will refer to this domain as "infrastructure".) The responsibility of the infrastructure is to support the objective of the system. The sensors and actuators are the "tools" of the system used to meet its objective. Humans can also be considered "tools" of the overall system. These tools can also be described as "actors" or "agents" in the system. The remaining components of the system (in terms of this paper) are the rules that define how the system (collection of agents) will accomplish its objective. In some, more human, systems, these rules are based on "policies", while more specific actions may be defined by specific commands. In manufacturing systems, the rules have historically been defined as control logic.

The remainder of this document focuses on a mechanism for describing the rules and delivering commands throughout a loosely coupled system. It also discusses a mechanism for processing the rules that takes advantage of the autonomous or semi-autonomous behavior of distributed agents.

COMMAND AND CONTROL HIERARCHY OF HUMAN SYSTEMS

The objective of the command and control architecture of a system is to provide a mechanism for distributing commands to the appropriate actors. This hierarchy allows supervision to be distributed to lower level entities. The hierarchy also allows for peer actors to work together to accomplish tasks. In human systems one expects to make use of human intelligence when accomplishing tasks. An organization expects to make use of levels of authority and organizational policy to set the objectives of lower level actors. In human systems, employees are "trained" so they understand the "rules" they are to use to accomplish tasks. Organizational policies describe ethical processes to restrict the performance of the actors to acceptable levels. When humans are recruited into an organization, they may be queried for their background and interests as another mechanism to insure that they will conform to organizational expectations. One objective of the owner of the human system is to establish that the employees will fit in the organization and respond to direction in an expected way. In human systems, the employee is expected to operate according to the rules. When an employee fails to perform as expected that employee is often sent for specialized training. That training is intended to correct performance problems. In some cases the employee will be replaced or reassigned. In human systems the employees are not treated as pieces of equipment. They are not operated completely by remote control. An organization needs to take advantage of the human's ability to make judgmental decisions. The training process is intended to help the employee make the right judgmental decisions in the course of performing job activities.

COMMAND AND CONTROL OF DEVICES

In an industrial automation system, most components of the system are driven directly by remote control. Commands are given to devices (actuators) and devices are expected to perform them. In many cases, there is a feedback signal that indicates that the command was accepted and/or that the function was completed. These control functions tend to be very simplistic. Other functions tend to be a little more complex when processes are adjusted. PID algorithms are often used to change values in a dynamic system where response times of control and feedback signals have a time constant associated. Still, these tend to be very small focused situations. Humans still play a large part in most automation systems, integrating maintenance schedules, managing inventories, loading parts, processing orders, etc to direct the operation of the entire system. When enterprise systems are utilized, they are commonly operated in a completely remote control model. The components of the system play a very small role in determining how the system functions. They just perform as directed. The primary benefits from the distributed industrial automation architecture are primarily derived from the fact that the components tend to be simpler and that component failure can sometimes be isolated easier than if there was one monolithic system. Judgmental decisions are never allocated to the devices. Performance is left to the supervisors on the factory floor. This means that the components don't use their autonomy to the benefit of the complete system.

LOOSE COUPLING

Human systems are loosely coupled by their very nature. Military systems do not normally rely on an individual soldier. Military organizations are constructed to respond to failure of individual soldiers (or components), and while performance may degrade, the "system" will continue to perform. Military systems will "adapt" to the circumstances.

In distributed Industrial Automation Systems individual production lines, or in some cases, individual machines, may be stopped when a component fails or an emergency stop is executed. Adaptation may only occur at the MIS or ERP layer when production must be rescheduled and rerouted. So, for the most part, Industrial Automation Systems tend to be tightly coupled systems that operate in a remote control mode today.

COMMANDS IN LOOSELY COUPLED SYSTEMS

When commands are delivered in human systems, they are delivered with the understanding that the human receptor will use some judgment in accepting the command. For example, if a factory worker was told to move a box from one location to another by a supervisor, the supervisor is assuming that the factory worker will recognize that if removing the box from its present position in a stack of boxes would cause the remaining boxes to collapse in a pile. If the factory worker was an industrial pick and place robot and the only robot logic was to pick up and place, the "system" may end up with a pile of rubble. By adding the capability to "adapt" to the environment, the industrial robot could make a judgmental decision of whether to move the box or not, OR, how to accomplish the task of moving the box safely. Thus one feature that would be an advantage for a loosely coupled system would be to give that device the ability to make judgmental decisions.

In the previous example, the robot was given a discrete order. A more complex scenario might be that the robot has an active directive to move boxes from one point to another. Consider the scenario where the order situation was increasing and the requirement was to move the boxes from one point to another more rapidly. In a completely remote control application, it may be possible to command the robot to move the boxes faster. However, by moving the boxes faster, there may be some degradation in the equipment. This might increase the risk associated with completing the task at all. There might also be a safety issue involved, where risk to human life or to other equipment might be accelerated. And, while this may not be an immediate problem, it might be something that would contribute to a failure sometime in the future.

DYNAMIC RULES

An alternative to directly commanding the robot to move more quickly would be to ask the robot to move faster by changing its reward mechanism. It is not normally considered that a robot will value any kind of reward mechanism. It is just performing according to a fixed set of rules. If, however, the rules were dynamic and were based on a reward mechanism, then it could balance risk and reward and make the decision to move more rapidly on its own. Thus, if the value of moving faster was justified, the robot would move faster; if not, the command would be rejected (and explained). Should the supervising entity feel it was appropriate (based on its broader view of the system) it could continue to escalate the value until it was accepted.

This process has several values: First, it benefits from the lower level device understanding its own limitations that could be changing over time. Second, it allows the senior level device to get a better understanding of its distributed resources.

It may be obvious from the previous discussion that we are not dealing with linear systems. We are dealing with multi-variable, non-linear systems. If one was to document these relationships with conventional source code (IF, THEN, ELSE) code it would be a difficult and probably impractical task.

Judgmental decisions, made by humans, are commonly attributed to right brain activity. Where left brain functions operate on language, numbers, formulas and discrete logic, the right brain decisions have to do with images and subjective functions.[3] When we attribute expertise to humans, we don't make that judgment just by their ability to calculate formulas, but by their ability to interpret what the answers to the formulas mean and what one should do with the answers.

The factory worker that has been asked to move the boxes from one location to another is balancing the importance of the directive from the supervisor against his/her risk tolerance and maybe his/her work ethic. If the worker decides that the stack of boxes may not fall when the desired box is removed from the stack, but the stack does eventually collapse; the worker's judgment may be held into question. He or she may be disciplined for faulty judgment.

A more rigorous mechanism for describing the rules might be used in a control system, where devices (robots, etc…) might be assigned the task of moving the boxes. There is the same potential for judgmental error, because the same judgmental decision is required. The requirement is to be able to adjust the judgmental process, rather than sending the robot somewhere for disciplinary actions. This adjustment could be to reduce its risk tolerance. It could also be to modify the supervisory command that reduced the value of the task, or both.

One advantage of tuning the judgmental decision-making of the robot, compared to sending the human for disciplinary action, is that the tuned robot can be duplicated. When a human for sent for re-tuning, it will not necessarily create a deterministic response. Every human may react differently.

MODELING JUDGMENTAL DECISIONS

Several technologies are currently applied to respond to the needs for judgmental decision-making for devices. Neural net based solutions are used to create repeatable responses to variable inputs. Known patterns are used to train the neural nets. Answers are determined when the taught patterns are compared with real-world data. Because neural nets are pattern matching solutions, there is no explanation of why the answer was created. It is difficult to "tune" the system when new inputs to the system need to be taken into consideration. Fuzzy logic is also used to address this type of problem. Fuzzy logic uses a process of linguistic uncertainty. Human terms are used to assign values that are assigned to geometric domains (fuzzy sets). The geometric values are combined to create an answer through a defuzzification process (center of gravity, for example). This approach blurs the data and allows answers to be created from soft (judgmental) terms. The results are repeatable, but not necessarily explainable in human terms. We define neural nets as a "mechanism" type of solution. The design of neural nets attempts to mimic the structure of the human brain. We define fuzzy logic as a "process" type of solution. Fuzzy logic has an objective of addressing solutions in a human-like way but is not restricted to the structure of the human brain. Other Artificial Intelligence techniques like forward and reverse chaining approaches have been used. These also fall in the "process" type of solutions.

This paper proposes another approach to the modeling of judgmental decisions. We call it KEEL (Knowledge Enhanced Electronic Logic). It is a "process" type of solution. It is based on the concept that judgmental decisions are accomplished by evaluating the importance of information and by defining relationships between information items. It is also a type of “expert system”, in that it requires a human expert to define the rules. It is also based on the assumption that complex judgmental decisions are not just addressing single isolated issues. Judgmental decisions are commonly evaluating multiple inter-related problems where one solution impacts other solutions. It also provides a method that gives completely explainable decisions and actions. This is important in solutions that must be audited and extended over their lifetime.

The factory worker is balancing all of these items: the directive from the supervisor, the potential disciplinary action that might take place if the command is rejected or the task fails, the difficulty of removing the box from the stack, the safety aspect associated with removing the box from the stack, the effort required to move all the boxes above the one of interest, potentially the time left in the work day (hoping the problem could be left to the next shift), etc. The factory worker is also basing his/her actions on how well he/she has been trained to understand the risks. The training is intended to establish values for risk, etc. The final course of action is based on how the different data items are valued and how the entire dataset is balanced when taking into account the relationships between data items.

This type of problem is not normally addressed with conventional logic. We believe it is not practical to develop algorithms of this complexity with a text based language. An approach is required that allows the rules to be defined as relationships between data items that allows immediate feedback to the decision-making process during the development process.[4]

Figure 1 shows a screen capture of the KEEL Toolkit development environment. The graphics on the screen represent KEEL source code. The bars at the top of the window indicate three decisions or actions. The height of each of these bars is an indication of its importance (the importance of the decision or action). The importance of a decision or action can be set manually, or it can be set automatically as the result of another decision or action (as shown in Figure 2). The dark gradient in these bars is an indication of the level of support accumulated for each of the decisions or actions. The slider bars at the bottom of the window are indications of inputs to the system. They are oriented so the inputs to each of the decisions or actions are grouped below that decision or action. The indicators above the sliders provide an indication of whether the respective input is supporting (unfilled circle) or blocking (filled circle). The wires (lines) on the screen provide an indication of how one decision or action impacts another. In this example, the accumulated support for the left most position or action is controlling the importance of the middle position or action. The accumulated support for the middle position or action is providing a level of blocking for the third position or action.

Figure 2 shows how KEEL source code provides immediate feedback to the developer during the development process. As the scroll bars (inputs) are manipulated, the entire system rebalances as all decisions and actions are re-evaluated. Relationships are defined by dragging wires between elements. When new wires are added, the system again re-balances itself and the decisions and actions are re-evaluated. Figure 2 show the impact of one analysis of inputs on the left and a re-balanced analysis of inputs on the right. By tracing the wires between elements, one can see the cognitive logic in graphical terms.

When considering complex relationships it is common to define non-linear relationships. The simple coding elements shown in Figure 1 can be combined to create these complex relationships. To assist the user visualize the logic, it is common to graph the relationships. Figure 3 is a 3-dimensional graph that is representative of the types of relationships that might be created.

CONTROLLING JUDGMENTAL DECISIONS

The hierarchy of control in this type of Command and Control system is exerted by supplying a driving signal to the decision-making process from the superior element. Thus to increase the incentive for an action, a bias is used to adjust the rule. Figure 4 shows this type of command authority. As stated earlier, the bars at the top represent the accumulation of the inputs which are shown as vertical sliders below the positions. The left most bar at the top, in this example, represents the local evaluation of a situation. The middle bar at the top represents the input from the command authority. The right-most bar shows an accumulation of both the local interpretation and the command authority. The "Threshold" value, which is a trigger for an action, is set at the midpoint (50%) level of the right-most bar. Note that in both the left and right screen shots, the command authority has supplied a value less than the Threshold. Note that the internal perceived value by the agent is less than the Threshold in both cases. Thus, left to its own, the agent would not trigger an action in either case. The right screen shot, however, shows that with a slightly higher internal perceived value and with the same command modification, the accumulated value will be sufficient to trigger the action.

Significant advantages can be achieved with this type of command structure. It allows a granular level of control. This can be established when the command authority supplies a modifier that is less than the threshold. The lower the external command bias, the more latitude is left to the agent. It can also demand direct control by supplying a modifier that is larger than the threshold.

The development process for this type of system is much more efficient due to the graphical source code which allows rapid definition of complex relationships. The dynamic nature of the graphical code allows the design to be evaluated as it is being developed. The designer can "see the system think" as inputs are balanced and all decisions and actions are re-evaluated. The designer can *see* the impact of the external commands into the decision-making process.[5]

Cognitive work analysis (CWA)[6] emphasizes the importance of the environment for understanding constraints of both the domain and actor and shows how the cognitive process operates at different levels. Figure 5 shows a more complex command and control architecture. In this case, command adjustments are coming from multiple sources. At the agent level, each device will have its embedded intelligence that tells it how to interpret the value of information items and how to include its internal strategy in its actions. Figure 5 shows how each of these rule sets can be tuned by multiple sources. The internal strategy can be tuned by commands from higher levels given that authority. The internal tactics can be modified by local observations as well as commands provided by other external information sources that may have access to information not available locally. This figure also shows how information from a peer level device can also be used to modify performance.

SUMMARY

KEEL technology provides an alternative to neural nets, fuzzy logic, and text based programming languages when judgmental decisions need to be incorporated into systems of loosely coupled devices. The system can benefit from localized knowledge that exists at the agent level by allowing commands from senior level devices to tune the decision-making rules of the agents rather than just exerting direct control. This is a preferred approach to alternatives that fail to utilize localized knowledge that is available at the agent. It is also preferred over approaches that extract all field data into a centralized host where all decisions for the entire system are made. Industry has broadly adopted the decentralized distributed model for system design which benefits from simplified components and the ability to withstand individual component failure. These systems can become more adaptive and responsive and make better use of the information at the point of action and still retain the ability to exert management control over the processes by allowing the components (agents) to take on more subjective decisions using the type of soft control defined here.

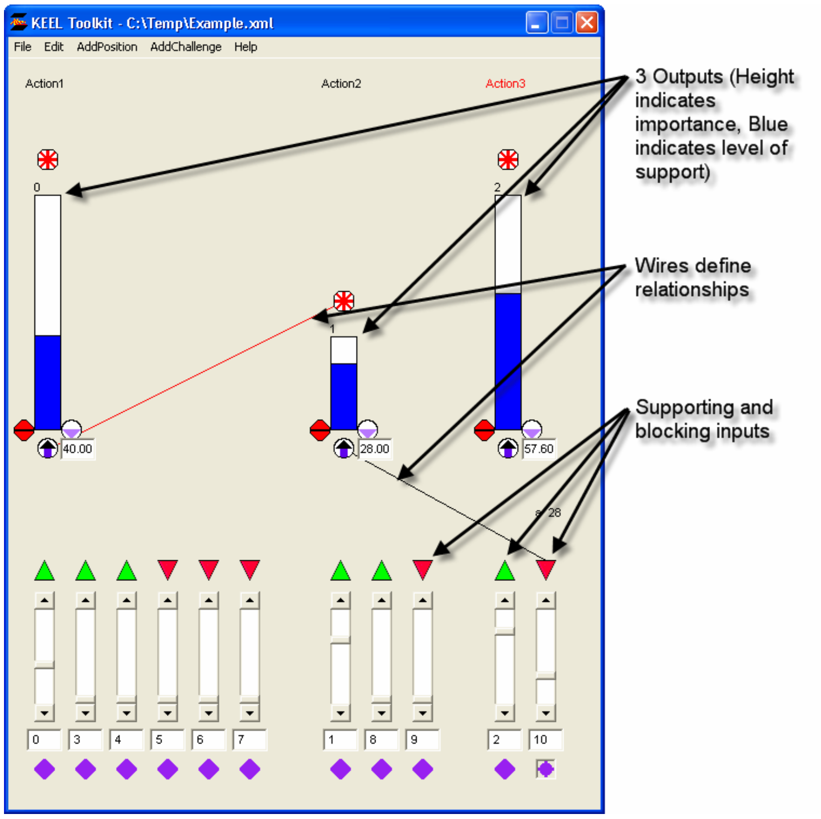
KEEL Toolkit - C:\Temp\Example.xml
File Edit AddPosition AddChallenge Help
Action1
Action2
Action3
0
1
2
40.00
28.00
57.60
28
0
3
4
5
6
7
1
8
9
2
10
3 Outputs (Height indicates importance, Blue indicates level of support)
Wires define relationships
Supporting and blocking inputs

FIG. 1 – GRAPHICAL PROGRAMMING LANGUAGE

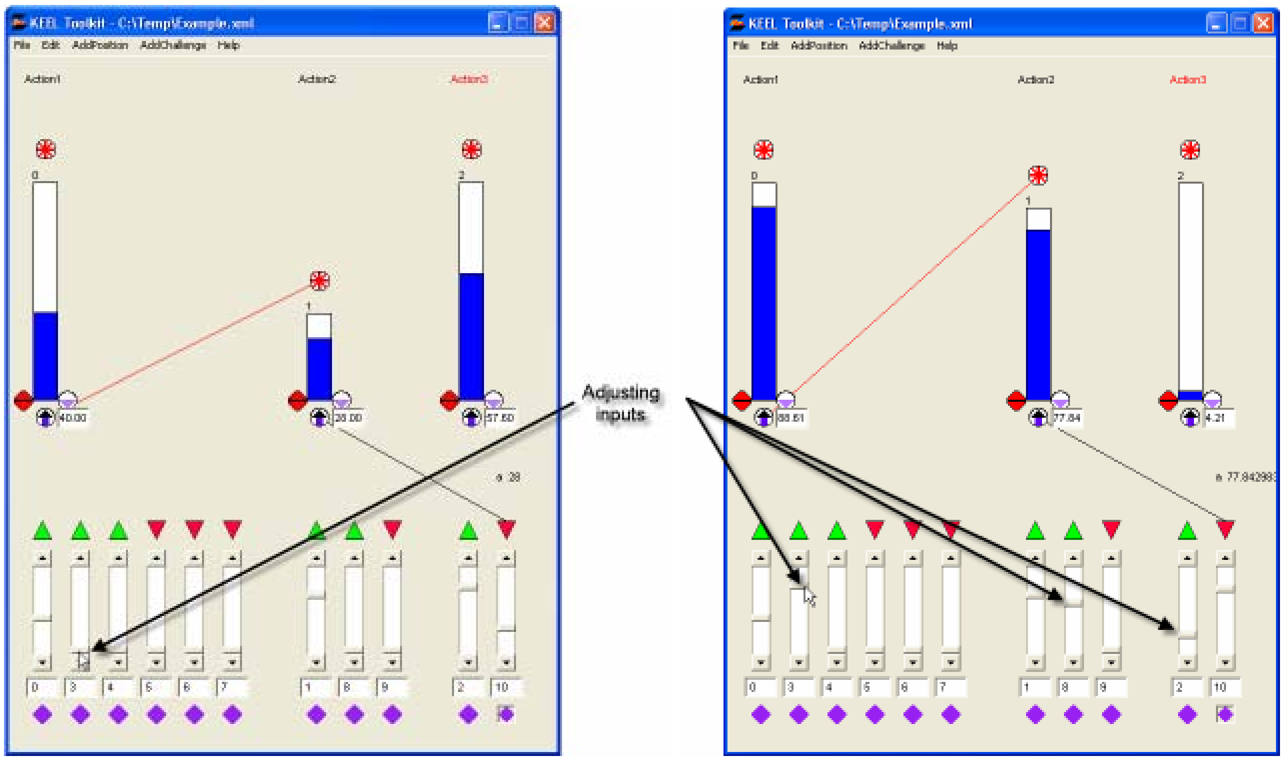

FIG. 2 – DYNAMIC CODE

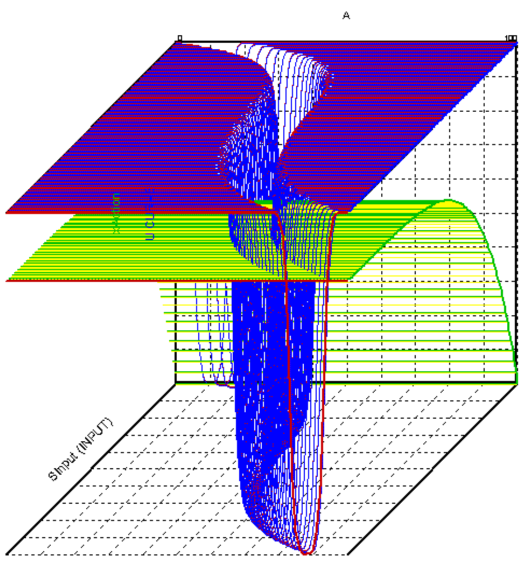

FIG. 3 – THINKING IN CURVES

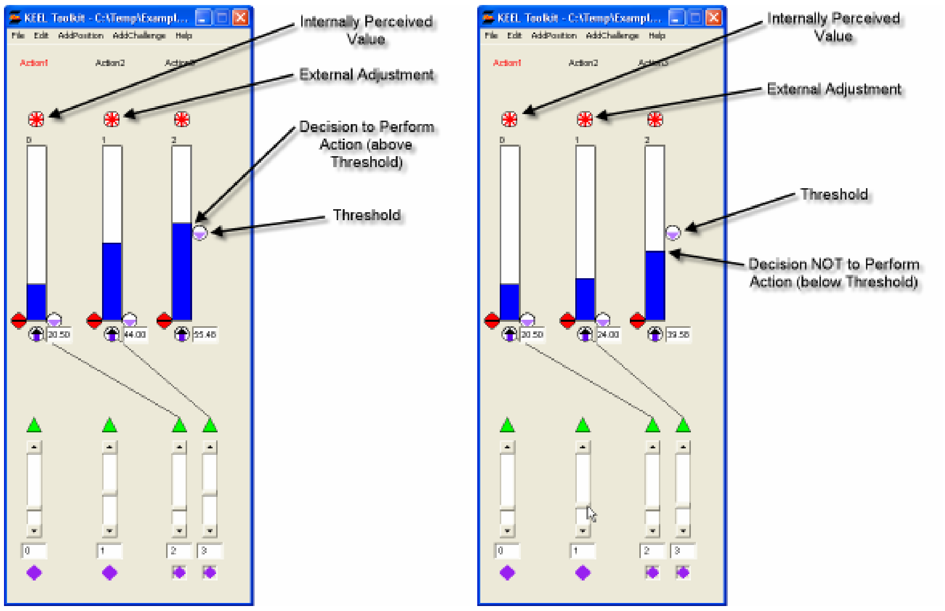
KEEL Toolkit - C:\Temp\Exampl...
File Edit AddPosition AddChallenge Help
Action1
Action2
Action3
Internally Perceived Value
External Adjustment
Decision to Perform Action (above Threshold)
Threshold
Decision NOT to Perform Action (below Threshold)
20.50
44.00
55.48
24.00
39.58

FIG. 4 – IMPACT OF EXTERNAL COMMANDS

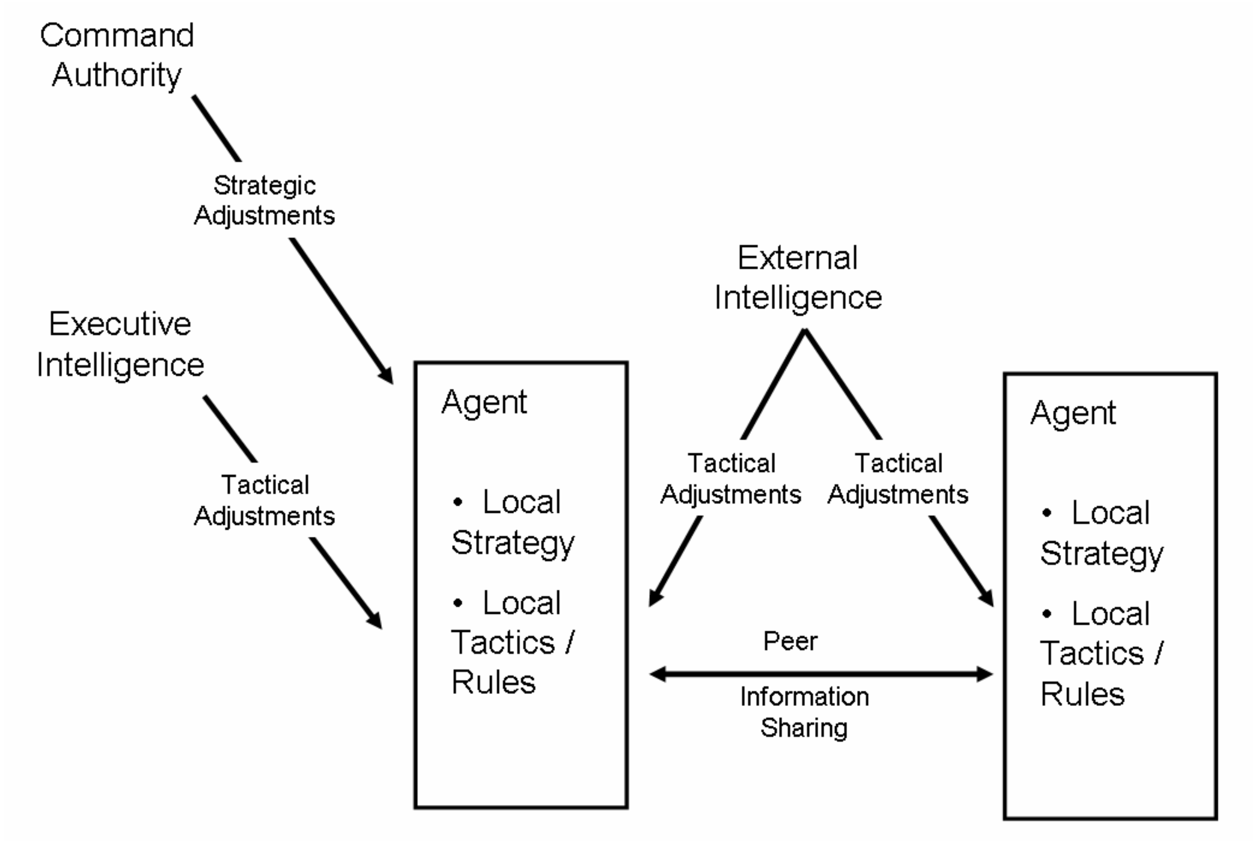

FIG. 5 – MULTIPLE INFORMATION SOURCES FOR COMMANDS

REFERENCES

[1] Defense Information Systems Agency, "What is the Joint Global Command & Control Systems", http://gccs.disa.mil/gccs/, last updated May 19, 2003

[2] Gall, John, "Systems-Failure (Theory of Errors)", Systemantics, How Systems Work and Especially How They Fail, Published by Quadrangle / The New York Times Book Company, Inc., 1977, pg. 61.

[3] McCrone, John, "'Right Brain' or 'Left Brain' Myth or Reality?", The New Scientist, http://www.rense.com/general2/rb.htm

[4] Keeley, Tom, "Right Brain Programming – Case for a New Programming Paradigm", http://www.compsim.com

[5] Keeley, Tom, "KEEL Technology Applied to Highly Distributed Loosely Coupled Systems", Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering, Tampa, Florida, 25-26 March 2004.

[6] Cummings, Mary, "Designing Decision Support Systems for Revolutionary Command and Control Domains", A Dissertation Presented to the faculty of the School of Engineering and Applied Science, University of Virginia, January 2004.