An Open and Efficient Architecture For Real-time Decision Making in Complex Systems

Karl F. Hirsch

TinMan Systems, Inc.
karl.hirsch@tinmansystems.com

1. ABSTRACT

Enabling a new wave of power-dense, energy efficient and dexterous mechanical systems, while at the same time, reducing complexity in design, operation and maintenance, requires not only significant improvements in methods and materials of actuation, but also the integration of pervasive intelligence within these systems. Intelligent, high performance, low maintenance machines are of enormous value in terms of human quality of life. With advances in mechanical engineering and materials science, manufacturing can resolve the hardware requirement. And now, with recent advances in computer processing power, what is needed is an efficient, technology-based, open architecture solution to design, deploy, utilize and update this pervasive intelligence. This paper presents a brief review of existing thought on intelligent actuation, the requirements of achieving real-time intelligence, and presents a technology based solution.

2. INTRODUCTION

2.1. Overview

Pervasive intelligence within an electro-mechanical system must rapidly integrate and process numerous inputs, from internal and external sensing, present recommended operational parameters to the operator and convert high-level operator commands to coordinated finely tuned actuator manipulation. In effect, the intelligence must compute all decisions necessary to optimally meet operator-level commands and objectives. Within milliseconds, and continuously, this intelligence must present the operator with the best choices, given all the data, and translate operator commands and system-level task management steps into multiple simultaneous mutually supportive decisions (desired actuation) to achieve the operator performance objective (move fast, move quiet, move efficiently…). In order to provide for this capability, the intelligence must exist within and among the actuators comprising the system and as an interface to the human operator (if not an autonomous system) – functioning as a single fabric of coordinated collaborative decision making. This coordination must also dynamically provide for conflict resolution and for performance deficits due to system degradation.

2.2. Background

A significant body of research at University of Texas has been conducted by D. Tesar *et al* for structured decision making in open architecture electro-mechanical systems to enable a new wave of technology based on machine intelligence (robotics, manufacturing cells, robot surgery, orthotics/prosthetics, more-electric vehicles, aircraft, etc.) [1]. Central to Tesar's formal and well-documented framework is the "intelligent" electro-mechanical actuator [hereinafter referred to as "iEMA"] as its basic building block. In this framework, intelligent mechanical systems are comprised of a set of iEMAs. Each iEMA is physically armed with multiple sensors, and highly certified with a finite number of performance (capability) maps and a set of combinations of performance maps (decision surfaces for optimal performance).

While this "knowledge" exists at the iEMA level, the operational intelligence is largely derived from utilization of additional knowledge at the system level where a human priority - based selection of decision surfaces is made, and presented visually to help the human operator make the best choice. In so doing, the system level intelligence also provides for the real-time fusion, integration and non-linear computation of real-time sensor data which includes actual output performance, to determine exactly where in this performance domain the system is currently operating. This circular and continuous computational framework allows for rapid and specific determinations (decisions) to be made of exactly how best to operate the system of iEMAs to achieve the given objective.

Tesar has shown that with the iEMA as the basic building block of this new wave of technology, and assuming that each iEMA is highly certified with performance data, enormous efficiencies in manufacturing and supply can be attained as well as on-demand system assembly to rapidly meet changing requirements in target theaters of operation.

This requires pervasive intelligence – decision making in complex systems - at the actuator and at the system level.

The purpose of this paper is to present the key requirements for designing and deploying pervasive intelligence within a complex electro-mechanical system, and present a solution that meets these requirements.

3. REQUIREMENTS FOR REAL-TIME PERVASIVE MACHINE INTELLIGENCE

How do you design and integrate this pervasive intelligence? How does it function in a real-time dynamic environment and make multiple simultaneous decisions from multiple inputs? How do you embed and utilize expert knowledge and performance capabilities at the iEMA level (and system level) efficiently? How do we update this embedded knowledge?

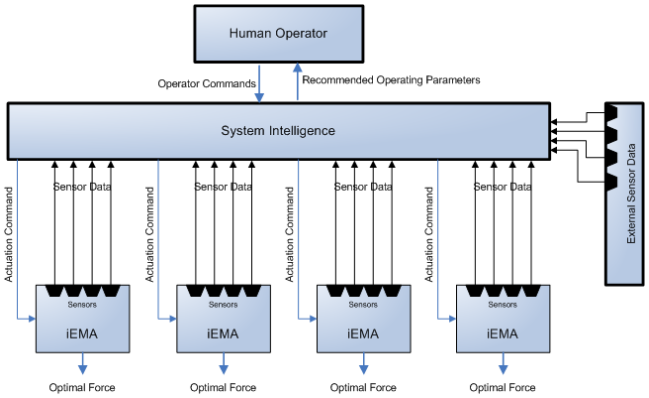

Figure 3-1: Example Intelligence layer within a Complex [MIMO] System of intelligent electromechanical actuators. Shows Human Operator, 4 Actuators and External Sensor Array

Real-time decision making in Multi-Input Multi-Output [MIMO] systems represents an immense challenge to design and implement. Fundamentally, this is a non-linear, unpredictable, parametric problem that requires a fusion of expert knowledge on system operation, mechanical systems capabilities and effective human operator interface.

3.1. MIMO Systems - Significant Number of Possible Circumstances to Account For

Each actuator within a system has multiple sensors (e.g. a multi speed vehicle drive wheel actuator may have 10 sensors) and contributes actuation relative to its performance capabilities and to the higher order operator commands. The intelligence-based system of actuators must translate the operator commands into a coordinated joint-contribution of forces across actuators. This system-level intelligence must also present to the human operator a set of recommended higher level operating parameters based on the current aggregate state and performance potential of each actuator (actuator-level sensor data), current system-level sensor data (GPS, collision avoidance, external environment conditions, etc..,) and current / most recent operator objectives.

In determining the total domain of possible input vectors to the system level intelligence, we can use permutation analysis. Given a quantity of k components/actuators of a system, each with a count of n sensors, each sensor with a count of g discretized ranges, we know that:

$$possible\ unique\ circumstances = \prod_{k=1}^{a} \prod_{n=1}^{s_k} g_{n_k} \quad (1)$$

where:

a = number actuators

s = number of sensors specific to an actuator

g = sensor granularity (quantity of discrete value ranges of interest between min and max)

Applying equation (1) in section 3.1 shows us that even for a simple mechanical system, with only 4 actuators, each with only 4 sensors with an average granularity of 3 discrete ranges of interest, and a set of 5 operator objectives, there are (3^20)*5 or 17,433,922,005 possible operating scenarios to account for in designing a systems-level decision making process.

3.2. MIMO Systems – Require Simultaneous Decisions

On a real-time basis, each cycle of logic execution within an intelligent MIMO system must provide for simultaneous decisions to be made. For instance, to maximize traction in a wheeled vehicle, all wheels must receive independent instruction on the production of torque, to adequately respond to feedback on loss of traction and feed-forward sensing of approaching terrain. This requires that within the single cycle of logic, for each of the wheel hub actuators, the system-level intelligence has both processed/computed unique and relevant information (sensor data and operating objectives) to determine power delivery (the output). The intelligence layer must also determine which recommendations to make to the operator in preparation for the next cycle of logic.

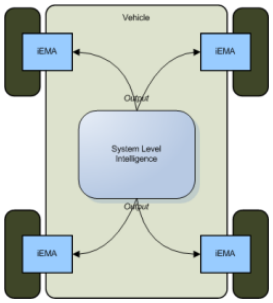

Figure 3-2: Four Wheeled Intelligent Actuation - Simultaneous Output

The system level intelligence must also account for conflict resolution between and among various operational states and/or decisions.

3.3. Current Tools / Approach:

Statistical decision tools, inferencing and custom programming algorithms, database referencing and mathematical optimization cannot adequately address real-time decision making in complex systems with an infinite domain of decision scenarios. Why? As shown in section 3.1, real-time MIMO decision making in dynamic environments presents a near infinite number of possible circumstances to anticipate. Given that typical MIMO systems may be comprised of parallel, series or hybrid decision structures – some or all utilizing output

feedback as input – the level of uncertainty to account for, by simply programming/coding a collection of algorithms for each domain-specific intelligence is unattainable – thereby increasing risk to the operator of such systems and the system itself.

It is conceivable that a manufacturer-based programming effort for each system could result in a custom design of a reasonably useful intelligence layer – but this approach would:

- have significant non-diminishing monetary costs,
- be unscaleable to on-demand manufacturing,
- be counter to a competitive supply chain,
- have relatively slow runtime execution and
- relatively high runtime power consumption cost.

3.4. A Decision Design and Deployment Technology is Needed:

What is needed is a standardized, real-time decision technology (decision-making construct and methods to exploit) for complex systems, made available through an open software platform for:

1) **designing** a structured flow of logic visually for complex multi-input multi-output systems,
2) **embedding and updating** knowledge within that structure, and
3) **integrating and utilizing** that structure within the target system

The platform and the runtime operation of the intelligence must provide for an "open architecture" to support a standard component-level intelligence interface to foster a competitive iEMA manufacturing supply field, and to efficiently utilize existing best-in-class software algorithms (such as biometrics, target tracking, LADAR, GPS/navigation systems, performance maps, sensor processor fault, etc..,) and operating drivers specific to various system hardware.

4. SOLUTION

4.1. The Decision Making Construct: Hierarchical Modular Role-Specific Non-Linear Computational Platform

The decision making technology that applies universally to both autonomous as well as human oversight-based operation in MIMO systems, is a hierarchical modular role-specific non-linear computational flow of logic.

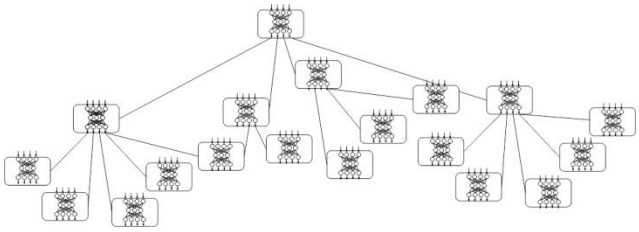

Figure 4-1: High-level View of Hierarchical Decision Structure

A hierarchical approach combining multiple forms of logic along with multi-action decisions at any level affords four major computational advantages:

1) **Significant reduction in input permutations**
   - reducing risk and uncertainty
2) **Simultaneous processing and decision making**
   - allowing complex system-level management across many iEMAs and decision structures
3) **Significant computation efficiency** due to context-specific computation (not all models within the structure must be computed at each logic cycle)
   - enabling very fast computation (real-time responsive) and
   - low power consumption (less processing)
4) A **single run-time function** results in all actions
   - providing an open architecture for synchronous and asynchronous external code or actuation execution

The models each have inherent rules of connectivity with other models, along with a sequencing and priority of computation. Each of the models represents a specific form of logic computation such as Boolean logic, low-order mathematics computation, fuzzy logic, optimized and constrained neural network models, timers and discretization templates. Simple to highly complex structures can be intuitively constructed with these computational building blocks.

Feedback through internal decision and response loops enables state management and internal multi-purpose short term memory. State awareness is provided for by specialized watch structures on the primary entry node of the hierarchy.

4.2. Visual design of a flow of logic and knowledge utilization within and among standard modules and custom sub-systems

We utilize this approach by visually and algorithmically abstracting a set of linear and non-linear computational models into objects and allowing an intelligence architect (domain expert) to then visually construct a flow of logic much like a human would visualize and approach a problem these models based on a set of context specific connectivity rules, in sequence, parallel or in feedback loops

Expert knowledge from a domain expert is embedded within the structure through various optimized training algorithms within the design environment and then updated without modification to the structure itself pre and post deployment.

4.3. Embed and Update Knowledge

System level and actuator level intelligence is comprised of software-based decision making structures and embedded knowledge. Collectively, this intelligence is a complex, non-linear decision making structure comprised of a flow of logic that utilizes requisite embedded knowledge about how best to perform given the input data at that time. The embedded knowledge is critical to determining the best possible actions because it provides the set of optimal operating parameters in a

dynamic environment (changing operator objectives/demands, external physical environment changes, system and sensor degradation/failure, etc..,). The knowledge itself is likely to be a combination of pre-deployment performance tests/data and human domain expertise on optimal actuator/system operation.

4.3.1. Embedding Knowledge is done through Tuning Non-Linear Models to Produce Desired Decisions

Embedding knowledge comes in two forms: simply tuning [training] plastic state models within the design environment to mathematically compute a given output decision under a set of known circumstances. This is done through the design-time application of a modified batch-based back-propagation error-reduction algorithm on all anticipated sets of circumstances (input set permutations) for the target model. We can further reduce uncertainty associated with ranged variables by quantizing them to "discrete segments". The other form of knowledge comes from the decision-based logic flow within the hierarchical structure. This is done by directing decisions within the framework to result in further processing, or a transfer of processing flow based on computed outcome of the preceding logic. With run-time learning (post-deployment acquisition of knowledge), this intelligence layer can then increasingly abstract the minutia of operating instructions necessary for each operator command. *(Runtime learning not discussed in this paper).*

4.3.2. Update knowledge without loss to existing knowledge – pre and post deployment

Knowledge related to optimal performance of a system or actuator must be updated when elements of the system or actuator change – whether through an intentional manufacturing (pre-deployment) design modification/improvement or during operation (post-deployment) as a result of damage or degradation, or even replacement. The process of adding or modifying performance knowledge must not present risk to the existing knowledge.

4.4. Efficient rules based open design process

A standard set of non-proprietary software tools that can be used by multiple manufacturers to design intelligence with a standard interface at the actuator level – thus ensuring an efficient actuator supplier market.

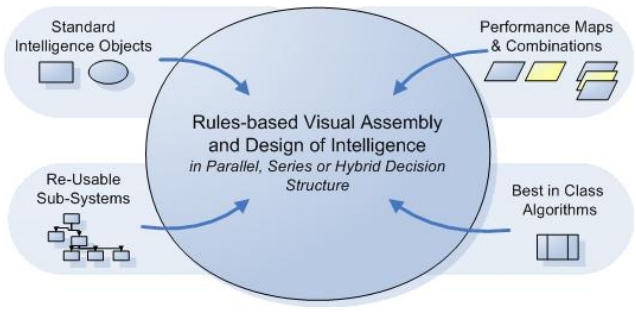

Figure 4-2: Open, Rules-based Visual Assembly of Intelligence Components

The design tools must allow reasonable visual abstraction of complex logic for real-time decision making in multi-input multi-output systems. For maximum efficiency, the design process and tools framework should provide for reusable, shareable and upgradeable intelligence objects.

4.5. Efficient integration and runtime execution

Supporting an open architecture for rapid assembly, configuration and refresh of actuators and sensor components requires a standard interface and sensor framework protocol that can gain familiarity.

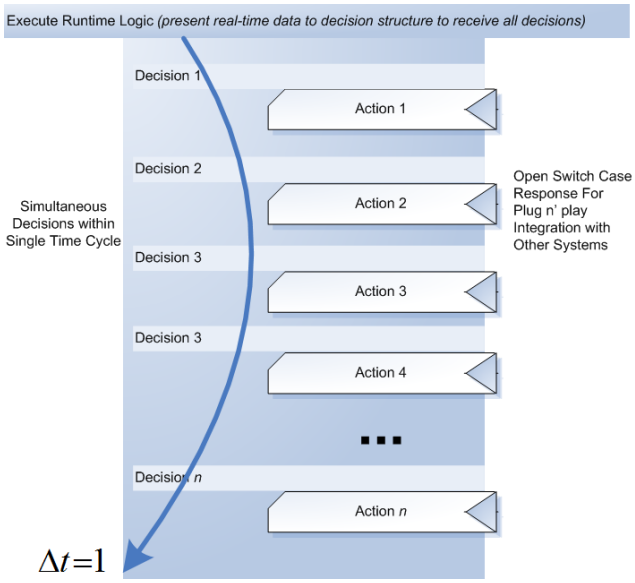

Figure 4-3: Runtime Execution for Simultaneous Outputs and Plug-n-Play Actions

5. CONCLUSION / RECOMMENDATIONS

Machine intelligence is achievable. It is enabled by a new approach to designing and deploying a logic framework and embedding, utilizing and updating expert knowledge within that framework. Real-time decision making in complex systems functioning in a dynamic environment is inherently parallel, non-linear and extremely complex to codify programmatically. TinMan Systems has recently released a first generation patent-pending platform that allows the development, deployment and runtime management of fully autonomous artificial intelligence. This platform is the result of nearly 9 years of research and the last 5 years of software development. The framework presented in this document is the foundation for TinMan technology.

This technology foundation can be used for *both* autonomous systems as well as for the man-machine intelligence projects contemplated in recent US ARMY Science and Technology Strategic Initiatives as well as the envisioned Intelligent Actuator Program.

It is recommended that a single hardware platform with one or more iEMAs be designated as a target prototype for an in-lab

custom development test integration of rudimentary machine intelligence given the existing TinMan design and deployment platform.

In parallel, or perhaps preceding the above work, it is recommended that exploratory work is done to customize the TinMan AI Platform to support contemplated initiatives in both soldier instrumentation (no actuation) and intelligent actuation in a vehicle (as above):

Project related work with the TinMan Platform:

- Runtime Integration of Current Learning Algorithms
- Streamline and Port the Runtime for Embedded Systems
- Runtime Decision Visualization Layer
- Import translation layer for Causal Bayesian test data
- Investigate and determine options for interface to visual decision presentation display for HDM
- Develop in-IDE interface for and runtime support for manufacturer specific and best-in-class algorithm integration.
- Investigate integration of best-class sensor-fault algorithms to distinguish sensor failure from low data integrity.

Rough timelines for platform customization, prototype and system integration are dependent on further discussions.

About TinMan Systems AI Software Platform: Most AI tools and AI middleware applications today are impractical for development of an AI system that can function autonomously in a dynamic environment. The TinMan IDE allows the construction, training and simulation of an AI system, along with the packaging and export of that system for use in the runtime environment. The runtime environment provides two dynamic link libraries which load, reconstruct and then execute all AI systems and coordinates the communication between and among the AI entities and their human controllers, if any.

AI systems developed in TinMan do not have to be autonomous, nor function in a continuous mode. An AI system can be designed to receive an input data set and compute one or more answers – from a single manual user selection event in the host application. However, TinMan AI systems are extremely well-suited for continuous processing of input data in a dynamic environment, allowing for the continuous potentially perfect execution of all desired outputs.

The process of constructing an AI system involves the declaration of a set of variables, a set of outputs and the selection and connection of a set of modular neural networks. Over 22 types of neural model templates are available for drag and drop addition to the AI system, in familiar logic flow model. States of decision making can either be made simultaneous or exclusive, as well as the end actions from the system.

As variables and actions are defined and then connected to the models, and as models are connected to each other, appropriate structural modifications are automatically and transparently made by the TinMan IDE. Although an AI system can be viewed at the neural level for a model or the entire system, the composition and connectivity of atomic units of the system completely abstracted. With the logical design of a system complete and connections and appropriate interpretations of data set, thorough training is performed. The resulting trained system is then packaged and made ready for integration into the host application. At runtime, the host application loads the packaged AI system via the TinMan runtime engine, feeds the AI system the set of input variable values, and responds to the returned computed action identifiers

6. ACKNOWLEDGMENT

I wish to thank Dr. Delbert Tesar at University of Texas for the numerous conversations, and the opportunity to discuss and share our respective work with each other. I also would like to thank industrial participants for insightful dialog and discussion in the recent DARPA EMA/EMS Workshop in Austin, TX in Nov. I also wish to thank Dr. Todd Hylton at DARPA for the introduction to Dr. Tesar and Todd's foresight to the connection between TinMan's autonomous artificial intelligence technology and the need for human operator based intelligent machines.

7. REFERENCES

[1] G. Krishnamoorthy, P. Ashok, D. Tesar, *Structured Decision Making for Enhanced Design and Operation of Ever Increasing Complexity in EMS,* (compendium of 5 papers) Robotics Research Group, University of Texas, Feb. 27th, 2011.

[2] D Tesar, *Next Wave of Technology*, University of Texas, July 2009.

[3] D Tesar, *Multi-Speed Hub Drive Wheels*, University of Texas, November 10th, 2011.

Karl F. Hirsch is founder and CEO of TinMan Systems, an early stage technology company based in Seattle. TinMan provides software and services for the design and deployment of autonomous artificial intelligence. Karl is the author of company's commercial platform, and has spent the last 5 years in its development. Karl received a B.S degree in Pre-Med Kinesiology at UCLA in 1988. He has 23+ years experience in software development (c, c++, java, php, actionscript 3.0, HTML and DirectX), and leadership roles in strategy, operations and business development at rapid growth, high-technology companies. He founded three technology companies – Preview Systems (IPO Dec. '99), OneChannel.net (M&A – Dec. '02) and TinMan Systems (current). As a 3-time software CEO and founder he has significant experience in equity financing from venture capital firms and public companies, raising ~$73M dollars, across 13 private equity financings over last 15 years. Karl also authored the earliest digital licensing and copy-protection/DRM technology (Timelock SDK and runtime) still in use today by software publishers. (1st DRM tools used in Microsoft and Symantec products ~ 1995). He has a background in evaluation, structuring and monetization of software/tech patents - written and filed 5 US patents – spanning 250+ claims in digital rights management [DRM] and artificial intelligence [AI], 2 patents have been awarded. Karl has remained active in software and technology industry organizations and consortiums – he served 2 years as Chairman – Software and Information Industry Association (SIIA) Internet & Electronic Licensing / Digital Distribution Division and a year as CompTIA member and 2 years contributing as Architect to Ingram Micro's RosettaNet Prg

