2018 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
SYSTEMS ENGINEERING TECHNICAL SESSION
AUGUST 7-9, 2018 - NOVI, MICHIGAN

IMPLEMENTING AUGMENTED INTELLIGENCE IN SYSTEMS ENGINEERING

Mark Petrotta
System Strategy Inc.
Sterling Heights, MI

Troy Peterson
System Strategy Inc.
Sterling Heights, MI

ABSTRACT

This paper will explore the opportunities for artificial intelligence (AI) in the system engineering domain, particularly in ways that unite the unique capabilities of the systems engineer with the AI. This collaboration of human and machine intelligence is known as Augmented Intelligence (AuI). There is little doubt that systems engineering productivity could be improved with effective utilization of well-established AI techniques, such as machine learning, natural language processing, and statistical models. However, human engineers excel at many tasks that remain difficult for AIs, such as visual interpretation, abstract pattern matching, and drawing broad inferences based on experience. Combining the best of AI and human capabilities, along with effective human/machine interactions and data visualization, offers the potential for orders-of-magnitude improvements in the speed and quality of delivered.

INTRODUCTION

Augmented Intelligence (AuI), an approach that promotes "team play" of human and machine intelligence, is a modern refinement of established AI approaches. By effectively joining the human skills in pattern matching, unstructured data, and intuition with computational approaches that excel in domain search, systematic trade space exploration, and statistical evaluation, the combined "team" has been proven to be more effective than either in isolation. For instance, machine learning algorithms can process past system designs, learn significant design characteristics, and visually present outcomes and the various tradeoffs. The human team can evaluate the domain space quickly, and watch for exceptional cases that might not be accurately handled by the machine. This paper will explore the potential, challenges, and requirements of implementing AuI in the engineering of systems.

AuI has been enabled by the adoption of Model Based Systems Engineering (MBSE), and particularly the use of formal modeling languages such as the Systems Modeling Language (SysML), UML, Architecture Analysis and Design Language (AADL), etc. Previous document-centric approaches to systems engineering resulted in less well-defined systems that, while generally intelligible to human readers, were too unstructured for algorithmic approaches. The model-centric approach, using SysML, is ideal for AuI, since SysML was designed to be both human and machine readable. For the human, there is a concrete visual

representation easily interpreted by a human (e.g. the familiar SysML block diagram, containing "boxes and arrows"). For the machine, SysML has a precise semantic representation that allows for simulation and model execution. Together, these two representations make SysML well-suited as a language for enabling AuI.

The innovation of systems is a prime opportunity for the application AuI, given the rapid increase in product complexity and time constraints. As an example, the relationship between the requirements of a system and the functional performance (e.g. SWAP-C: space weight, power, and cooling) is fundamentally mapping, traceability, and parametric relationships between requirements and design parameters. In other words, if a requirement is changed, what is the impact on functionality and performance? Even with MBSE, this is still a manual process of evaluating the models, collecting the inputs of domain experts, and determining the impact. In contrast, AuI offers the ability to use every past system as inputs to machine learning algorithms. The design team can quickly visualize the opportunities and the challenges of those design decisions. Not just one course of action can be evaluated, but every course of action could be visualized, evaluated, and communicated to the design team, subject more to computational constraints rather than time constraints.

MODEL-BASED SYSTEMS ENGINEERING (MBSE)

The growing complexity of systems necessitates a systems engineering approach. It requires a systems paradigm which is interdisciplinary, leverages principals common to all complex systems, and applies the requisite physics-based and mathematical models to represent them. INCOSE defines Model-Based Systems Engineering (MBSE) as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases…" [1] The Object Management Group's MBSE wiki notes that "Modeling has always been an important part of systems engineering to support functional, performance, and other types of engineering analysis."[2]

The application of MBSE has increased dramatically in recent years and is becoming a standard practice. This has been enabled by the continued maturity of modeling languages such as SysML and significant advancements made by tool vendors. These advancements are improving communications and providing a foundation to integrate diverse models. MBSE is often discussed as being composed of three fundamental elements – tool, language and method. The third element, method, has not always been given proper consideration. Because the language and tool are relatively method independent, it is methodology which further differentiates the effectiveness of any MBSE approach and its ability to help manage the complex and interrelated functionality of today's systems. For the approach discussed in this paper, the "methodology" includes the application of Artificial Intelligence to augments the application of Systems Engineering activities.

ARTIFICIAL INTELLIGENCE

Artificial Intelligence (AI) is the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

Since the term AI is widely used for a variety of different algorithms and approaches, most definitions are functional rather than technical. For instance, according to Stanford AI researcher Jon McCarthy, "Any

program can be considered AI if it does something that we would normally think of as intelligent in humans" [3].

There are two primary categories of AI:

- Rules Based: Rules based AI uses a set of defined rules to derive and manipulate data. This represents explicit knowledge that can be provided to the AI system. For instance, in lexical parsing of the English language for natural language processing, grammatical rules can be defined (e.g. proper nouns are capitalized, as is the first word in a sentence). Rules based is also used for solvers, such as navigation algorithms that minimize travel time while following road rules (e.g. turns are permitted at certain intersections).
- Patterns Based: Usually known as Machine Learning (ML), patterns based approaches seek to capture tacit knowledge - knowledge which is difficult or impractical to explicitly define - through statistical approaches. For instance, it is difficult to completely list rules for email spam filtering. However, with a set of emails categorized as either spam and not spam, an algorithm can infer certain words or phrases that are effective predictors of spam for classification of new emails. Similarly, autonomous systems can learn appropriate driving techniques by observing a human driver.

Many modern systems are hybrids of the rules and pattern approaches. For instance, chess AI systems use rules (e.g. 6 piece types, each with a small set of moves) for traversing the game tree, along with learned patterns to preemptively eliminate certain branches as poor moves.

HUMAN / AI INTERACTIONS

Competition

Chess was one of the first significant applications of AI. It was an ideal game to test the capabilities of AI, since it is a zero-sum, perfect information two-person game with a small set of rules. Zero-sum refers to the fact that an advantageous move by player A is disadvantageous to player B. Perfect information means that all positions for each player are perfectly visible (in contrast to other games like poker, where some information is hidden, and there is a role for "bluffing"). Chess is also a Markov process, where each arrangement of chess pieces can be evaluated independently of the moves that created that arrangement. These characteristics allowed for effective implementations of state space search algorithms, and comparisons of performance of human player versus machine [4].

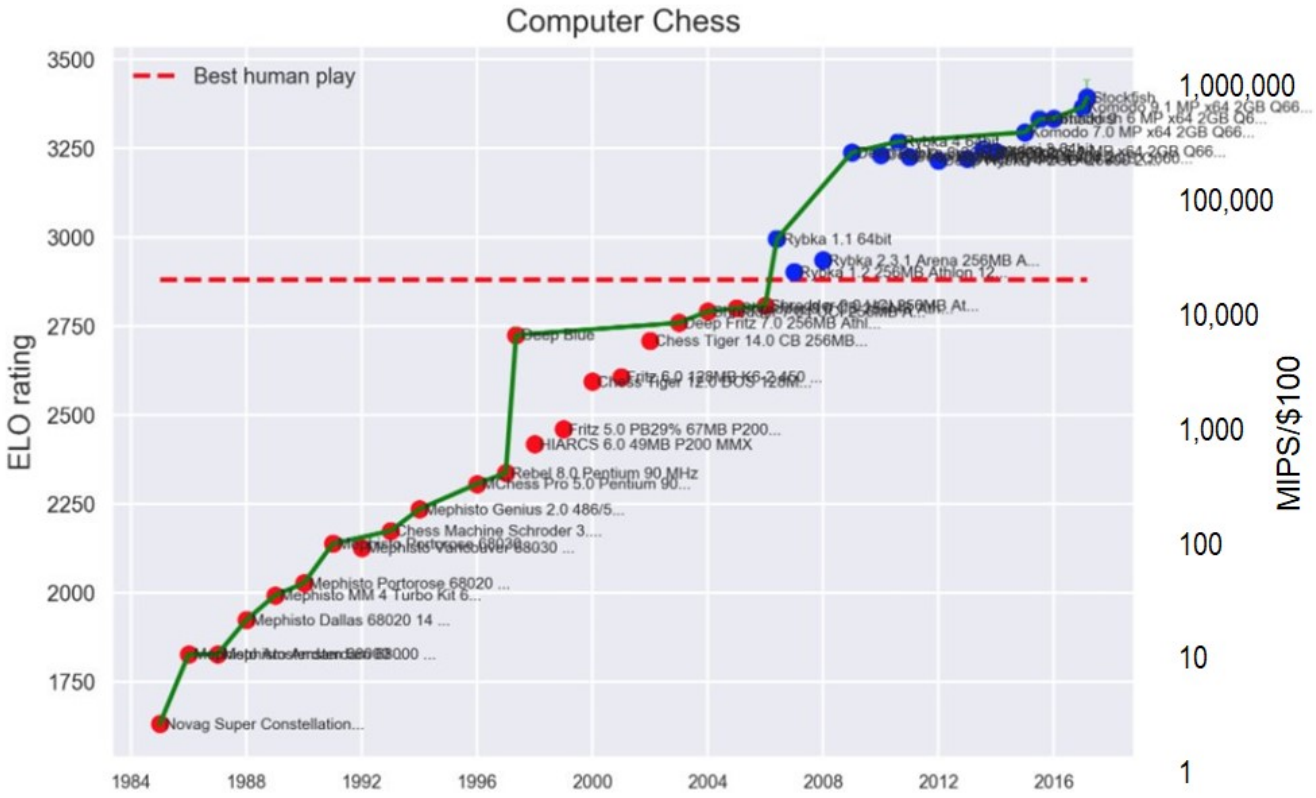

Figure 1 - Growth of chess AI capabilities as measured on the ELO ranking scale of zero-sum games. Human grandmaster performance is shown in dashed-red. The line is approximately horizontal, as human performance changes very little on the scale required to show AI growth.

Figure 1 shows the evolution of chess AIs as ranked on the ELO scale, a method for calculating the relative skill levels of players in zero-sum games such as chess (named after its creator, physicist Arpad Elo) from 1984 to 2016. On the second Y axis is MIPS (millions of instructions per second) per $100, the affordability of computational power shown in logarithmic units. As computational power grows exponentially, the chess

AI performance grew linearly, characteristic of tree search algorithms. Shown in red is the best human grandmaster performance, which changes very little during this timeframe.

In 1996, IBM's supercomputer Deep Blue played world chess champion Garry Kasparov in a pair of six-game chess matches. The first match was won by Kasparov, but the second was won by Deep Blue. This match was the first defeat of a reigning world chess champion by a computer under tournament conditions.

Collaboration

Given the increase in AI performance over time, few expected long-term competitive human-machine chess. Kasparov, reflecting on the state of chess, developed a new style of competitive chess known as "freestyle chess", "team play", or "advanced chess". What would happen if, instead of competing against one another, humans and computers collaborated? In this style of play, human players can use any resource at their disposal, including AIs, books, or other human team members. This style of play is also known as "centaur" play (i.e. human head on a horse body).

In 2005, a Freestyle Chess Tournament ended with a shock win by the team "ZachS" - two amateur human players using three computers - defeating other grandmaster-led teams utilizing their own strong AI systems. To the chess world, that was as much as a shock as Kasparov's loss to Deep Blue. This led Kasparov to reflect on the nature of man/machine teaming:

> *What makes a good freestyle player? Someone who can work out the most effective combination, bringing together human and machine skills. I reached the formulation that a weak human player plus machine plus a better process is superior, not only to a very powerful machine, but most remarkably, to a strong human player plus machine plus an inferior process. [5]*

By process, Kasparov is referring to the working relationship between the human and AI. Does the human understand how the AI recommendation was made, and correspondingly - conditions where the AI recommendation may be weak? This has been restated as Kasparov's Law:

> *A weaker player with an effective working relationship with AI will outperform a stronger player with a weak relationship with AI.*

AUGMENTED INTELLIGENCE

While AI performance has increased significantly, improved by better algorithms and dramatically increased computational power, there is still no artificial general intelligence (AGI) – known as "strong AI" – outside of science fiction. Strong AI would be able to out-perform a human on any intellectual task. All existing forms of AI are "weak AI" - artificial intelligence that is focused on one narrow task. Therefore, it is useful to think of AI as a tool or prosthetic that enables better human cognition. Augmented Intelligence (abbreviated AuI to differentiate it from AI), is the cognitive unit consisting of a human and AI in a teaming arrangement.

Augmented Intelligence is a complement, not a replacement, to human intelligence, by helping humans become faster and smarter at the tasks they're performing. Augmented teams should measurably improve on performance scales, as in the ELO score in chess. In an effective AuI team, the team should outperform either the human or AI in isolation:

$$p(AuI) > p(Human)$$
$$p(Aui) > p(AI)$$

Figure 2 - Assertion that the team performance will exceed the performance of the individual components with Kasparov's conditions are met. p is an arbitrary performance measure.

GUIDELINES FOR COOPERATIVE ENVIRONMENT

AuI systems must reflect realistic understandings of user needs, human psychology, and methods of interaction. A poorly designed system will make the combined decision worse, not better. The corollary to Kasparov's Law, with application to design is:

> *A human-centered design environment is necessary to both the creation and deployment of algorithms intended to improve expert judgment.*

The corollary to Kasparov's law states that a well-designed decision environment involving both the algorithm and the human decision-makers is required to improve human judgement. Therefore, we offer the following guidelines for algorithm and human-machine environment:

Guidelines for Algorithm Development

The algorithm should have:

1. Agency: Reflect the information, goals, and constraints that the decision-maker tends to weigh when arriving at a decision
2. Perspective: Analyze from a position of domain and institutional knowledge, and an understanding of the process that generated it – provide context.
3. Relevancy: Anticipate the realities of the environment in which it is to be used
4. Objectivity: Avoid biased predictors
5. Transparency: Be transparent, peer-reviewed or audited to ensure that unwanted biases have not inadvertently crept in
6. Candor: Effectively present measures of confidence and "why" messages (ideally expressed in intuitive language) explaining why a certain algorithmic indication is what it is

Figure 3 – Guidelines for Algorithm Development (Adapted from [6])

Guidelines for the Human Machine Interface

The user environment should have:

1. Clarity: The algorithm's assumptions, limitations, and data features should be clearly communicated

2. Intelligibility: Algorithm end users should have a sufficiently detailed understanding of their tool to use it effectively
3. Methods: Guidelines and business rules should be established to convert predictions into prescriptions
4. Responsibility: Suggest when and how the end user might either override the algorithm

Figure 4 - Guidelines for Human Machine Interface (adapted from [6])

Example - NLP Reconciliation of Model Elements

In Figure 5, a natural language processing (NLP) algorithm was used to suggest merge options between two SysML models. In other words, merge model A with model B, combining elements that are identical. Model A is shown in the first column, and the best fit from B in the second column. Note that some matches are verbatim, while others are very poor quality matches (e.g. survey vs service plan).

Model A	Model B
Verification Procedures	Verification Procedure
Quality Assurance Evaluation Results	Qualification Test Report
Quality Assurance Plan	Quality Management Plan
Configuration Record	Configuration Status Report
Survey	Service Plan
Complaints	Service Plan
Portfolio Evaluation Report	Evaluation Report
Needs Assessment	Product Need Assessment
Measurement Procedures	Measurement Procedure
Service Level Agreement	Service Management Plan
Use Case	Reuse Plan
Knowledge Management Records	Knowledge Management Record
Organizational Procedure	Operational Test Procedure
Project Life-Cycle Policies and Procedure	Life-Cycle Policy and Procedure
Other Plans	Service Plan
Operational Strategy	Operational Test Procedure
Test Procedure	Audit Procedure
Analysis of Metrics and Variations	Risk Management Policy and Plan
Test Procedures	Audit Procedure
Complaint	Contract
Release Records	Release Record
Organizational Procedure	Operational Test Procedure
Business Action Plan	Installation Plan
Quality Assurance Plan	Quality Management Plan
Disposal Records	Disposal Record
Measurement Results	Measurement Plan
Measurement Strategy	Measurement Procedure
Design Description	Database Design Description

Figure 5 – Poor HMI for Model Merge. Algorithm NLP uses simple Hamming distance measurement of model duplicates.

In Figure 6, the same match algorithm displays the confidence level to the user, and sorts the assessment by confidence. From the algorithm, the "Candor" rule is applied, presenting measures of confidence. For the HMI, there is "Clarity" for the algorithms assumptions, and "Responsibility" for selecting a threshold for when to approve the algorithm selections (shown as darker green).

Model A	Score	Model B
Knowledge Management Records	0.96	Knowledge Management Record
Verification Procedures	0.96	Verification Procedure
Measurement Procedures	0.95	Measurement Procedure
Measurement Procedures	0.95	Measurement Procedure
Measurement Procedures	0.95	Measurement Procedure
Maintenance Procedures	0.95	Maintenance Procedure
Validation Procedures	0.95	Validation Procedure
Disposal Records	0.94	Disposal Record
Release Records	0.93	Release Record
Assessment Report	0.88	Assessment Record
Assessment Report	0.88	Assessment Record
Release Report	0.86	Release Record
Customer Satisfaction Report	0.86	Customer Satisfaction Survey
Customer Satisfaction Report	0.86	Customer Satisfaction Survey
Customer Satisfaction Report	0.86	Customer Satisfaction Survey
Information Security Plan	0.85	Information Security Policy
Other System Requirements Specification	0.85	System Requirements Specification
System Design Descriptions	0.81	Software Design Description
System Design Description	0.81	Software Design Description
Knowledge Management Policy	0.81	Knowledge Management Record
Measurement Data	0.81	Measurement Plan
Measurement Data	0.81	Measurement Plan
...		
Survey	0.33	Service Plan
Budget	0.33	Audit Report
Standard	0.30	Audit Plan
Skills	0.26	Personnel Skills Record

Figure 6 – Better HMI for Model Merge. Quality of match is included in the center column, and the results are sorted by certainty.

Note that the result set is identical for both examples. The only difference is the inclusion of a quality metric, sorting by the quality metric, and visual indication of high-quality matches. In practice, the second example produced much more trust in the team, and better utilization of suggestions.

IMPLEMENTATION CONSIDERATIONS

Augmented Intelligence for SE Conceptual Model

Augmented Intelligence for Systems Engineering (AuISE) involves teaming arrangements that can be used to enhance the engineer's decision making in a transparent fashion, leverage expert knowledge, and apply solutions rapidly. The systems engineer and algorithm can coordinate to develop transparent, traceable, and understandable system designs that are better than either human or algorithmic approaches could develop alone.

AuISE teaming requires a framework for analyzing and applying AuI principles to SE processes. The INCOSE Agile Working Group has developed a conceptual framework for learning and applying patterns in SE, called the Agile Systems Engineering Lifecycle MBSE (ASELCM) domain model [7][8]. The ASELCM domain model itself references the ISO/IEC/IEEE 15288:2015 framework for engineering processes. Together, these can be effective in describing a cooperative SE framework.

Agile Systems Engineering Lifecycle MBSE (ASELCM) Domain Model

The ASELCM establishes a set of system reference boundaries. This ASELCM Pattern particularly refers to three major system reference boundaries, and within those, six subsystem reference boundaries. These are

all logical boundaries (defined by the behavior, not the identity, of systems), and are depicted by the iconic diagram of Figure 7. Figure 8 presents a more detailed version of the iconic representation.

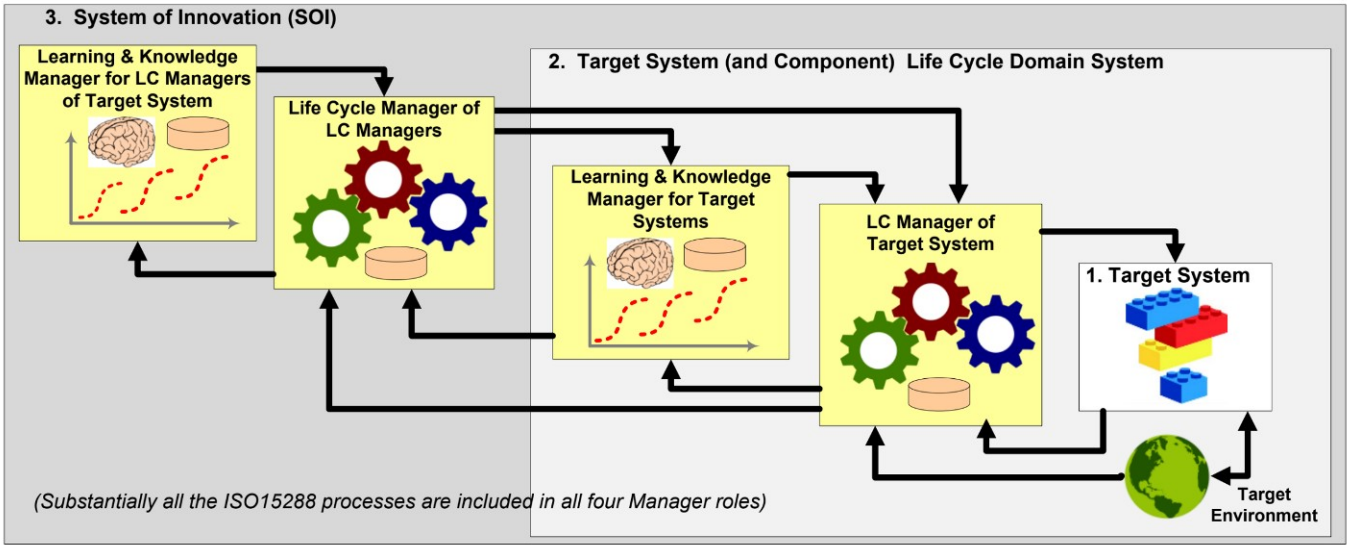

Figure 7 - Iconic view of the Agile Systems Engineering Lifecycle MBSE (ASELM) model

System 1 is the system under development, or target system. System 2 reflects the product lifecycle domain of the target system, including its operational environment. System 3 is the system of innovation, or the "system of systems" that reflects on the design process itself. This analysis focuses on System 1 and System 2.

The manager roles, shown in yellow, contain all processes described by ISO 15288. The Lifecycle Manager of Target System is responsible for applying known information about the target environment and target system. The Learning and Knowledge Manager is responsible for learning new information about the Target System.

In a traditional SE environment, these would be largely human roles. For instance, an automotive headlight engineer could consider options for LED lighting over halogen, given the known reliability, cost, space, weight, etc., considerations. This reflects the System 2 Lifecycle Manager (LCM) role. The same engineer could use product data to learn new facts about the target system, such as that high ambient temperatures in hot environments reduce LED lighting longevity. In this case, the engineer is operating in the System 2 Learning and Knowledge Manager (LKM) role.

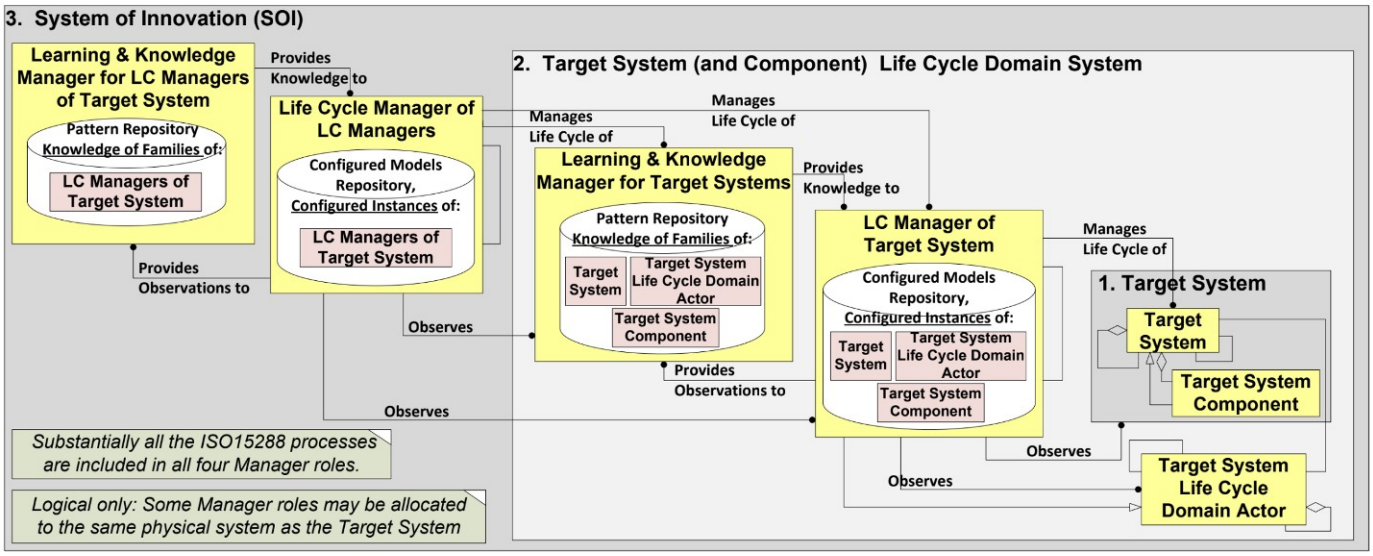

Figure 8 – Detailed view of the Agile Systems Engineering Lifecycle MBSE (ASELM) model

The Lifecycle Manager, in applying known information, can be compared with the AI rules based approach. Similarly, the Learning and Knowledge Manager uses the AI patterns based (machine learning) approach to learn new information.

ISO 15288 Process Framework

ISO/IEC/IEEE 15288:2015 establishes a common framework of process descriptions for describing the life cycle of systems created by humans. It defines a set of processes and associated terminology from an engineering viewpoint. These processes can be applied at any level in the hierarchy of a system's structure [9].

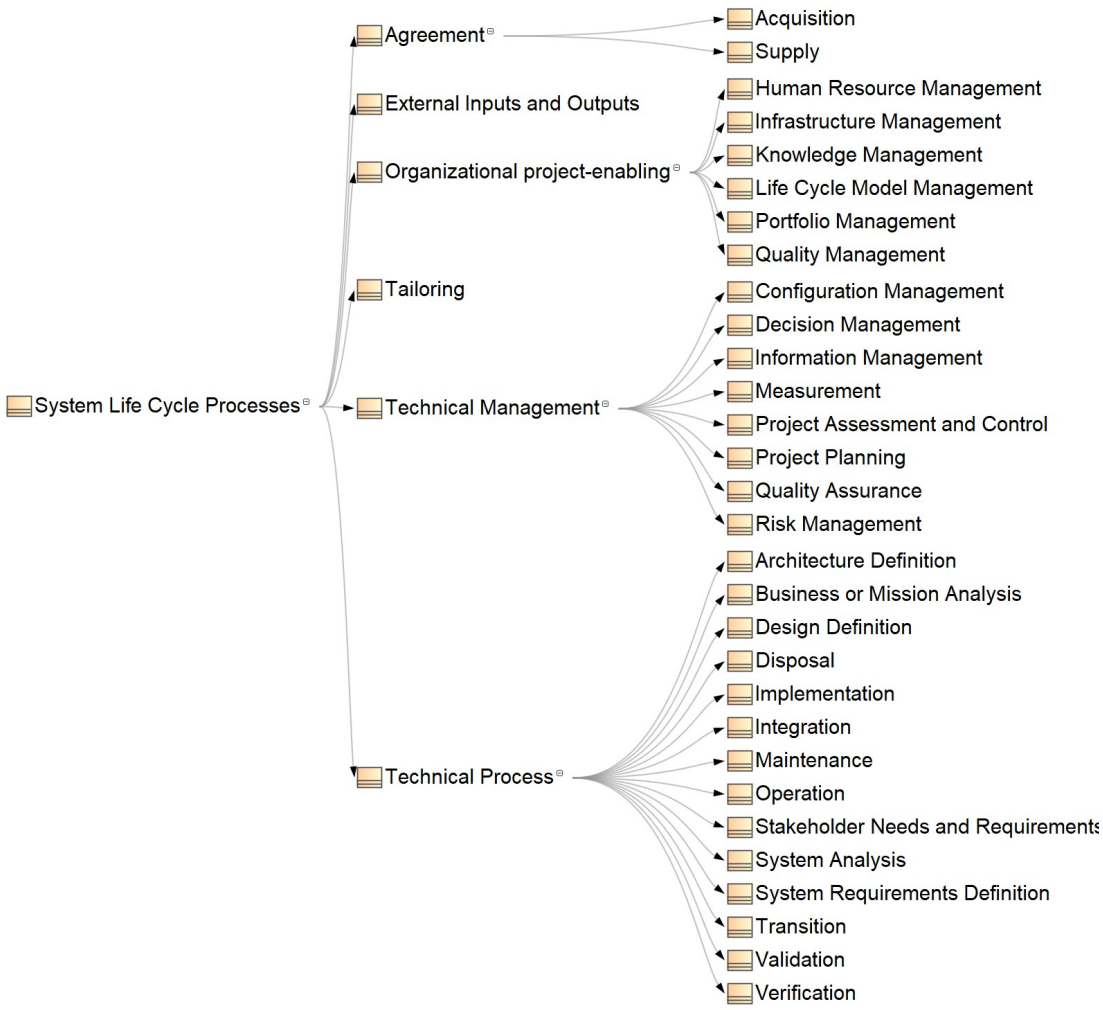

Figure 9 - ISO/IEC/IEEE 15288: "Systems and Software Engineering–System Life Cycle Processes"

CASE STUDY

Consider an example fault diagnostics system for a hydraulic actuator (this example inspired by Siemens MADe reliability analysis tool). This falls under the ISO15288 System Life Cycle Processes → Technical Processes → Maintenance and Design Definition. Using the ASELCM System 2, a Lifecycle Manager (LCM) and Learning & Knowledge Manager (LKM) can be defined.

The LCM will apply what is known about hydraulic actuators to the target system. In this case assume that there is a performance requirement to achieve > 90% fault isolation, and constraints that sensors can be placed on hydraulic or mechanical connectors (see Figure 10).

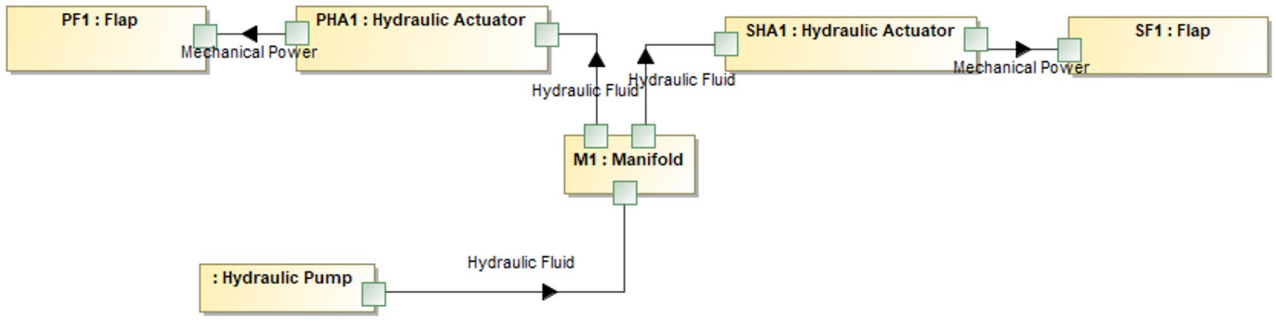

Figure 10 – Simple hydraulic system with 0% fault isolation

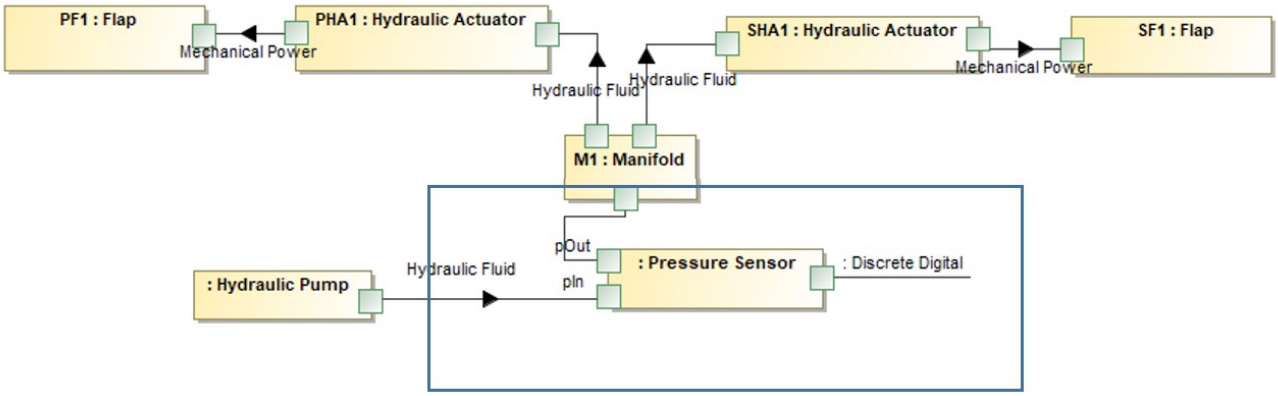

Figure 11 - Simple hydraulic system after one sensor added (14% isolation of all possible failure modes)

As shown in Figure 10, there is 0% fault isolation, since no sensors are included in the design definition. By applying the LCM, the AuISE system can apply what is known about the problem space, and select placement of sensors on hydraulic or mechanical connectors to achieve the fault isolation. For instance, shows the system after one pressure sensor is added. This increases the fault isolation capability to 14%.

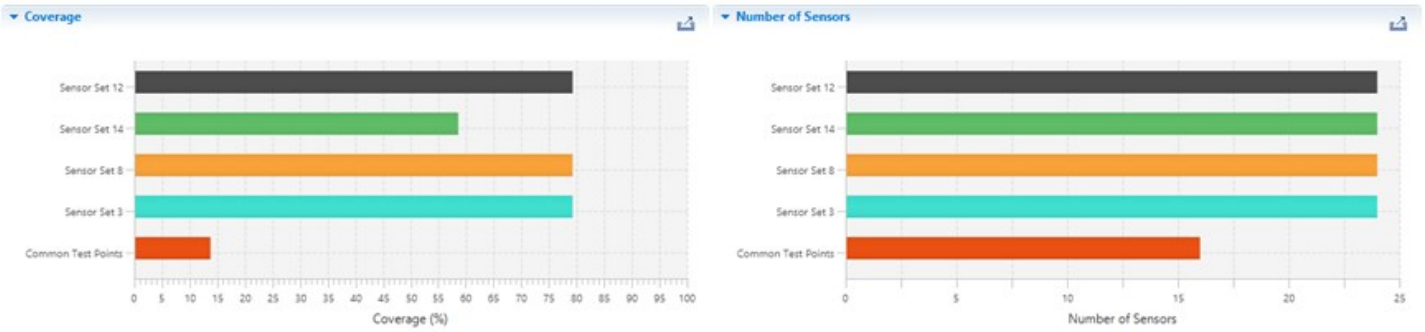

Figure 12 – System 2 Lifecycle Manager applied to options for sensor placement to maximize isolation coverage. (genetic algorithm example from Siemens MADe tool)

The LCM can continue to suggest options for optimal sensor sets. Figure 12 shows a genetic algorithm with options for coverage as a function of number of sensors. Each row reflects one design option selected by the LCM.

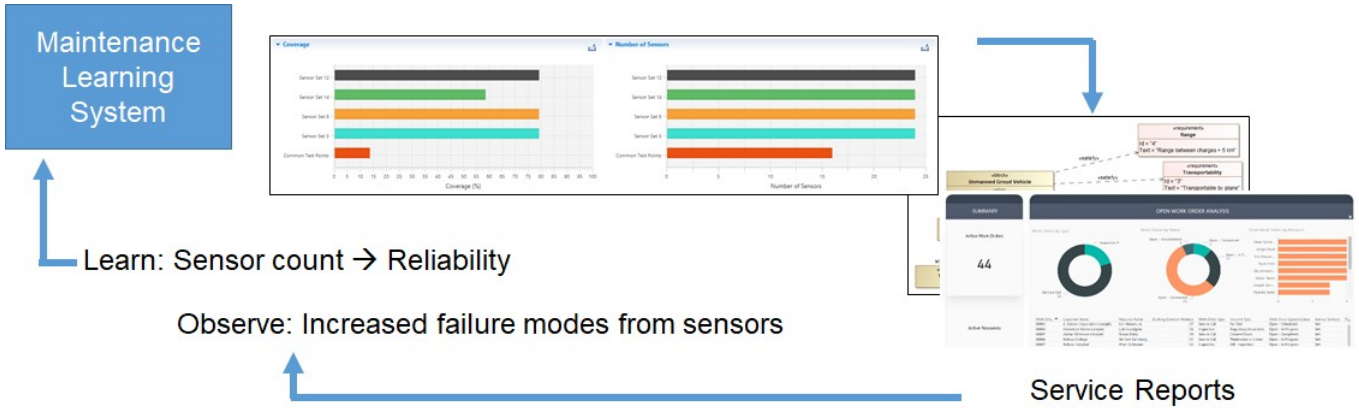

Figure 13 – Learning new aspects about a system - System 2 Learning and Knowledge Manager applied to the fault isolation example.

The optimal design can be selected by the AuISE team, and applied. Note that only isolation coverage was considered. In a more realistic example, cost, space, weight (i.e. SWAP-C) and other factors would be considered.

Figure 13 considers that use of the Learning and Knowledge Manager (LKM). In this example, the LKM can learn new aspects of the system from available data. The service reports are used to observe that the

sensors themselves fail, therefore – increasing sensors can decrease reliability while simultaneously increasing fault isolation. This knowledge is captured, and can be considered during the application of knowledge in the System 2 LCM. Therefore, the LCM, applying was is already known, can consider real-world reliability as a consideration during design selection.

In summary, the LCM applies what is known. The LCM can be improved by the LKM. The LCM can be a "rules based" AI informing the team to make an effective decision. The LKM utilizes "patterns based" AI to learn new rules, and inform the LCM.

CONCLUSION

Artificial intelligence has moved from conceptual and theoretical applications to core industrial functions in the last few years. Virtually all the progress in AI is in exceeding human intelligence on narrow tasks. Rather than replacing human intelligence, a common trope of science fiction, AI has augmented human intelligence.

The motivation for augmenting human intelligence can be expressed by Kasparov's Law – the performance of human/AI teams dominate both the human and AI individually. Kasparov's law has held in all domains, virtually without exception. In 2009, Garry Kasparov concluded that "weak human + machine + better process was superior to a strong computer alone and, even more remarkable, is that its superior to a strong human + machine + inferior process. Therefore, Augmented Intelligence (AuI) became more relevant than AI alone.

However, the performance of team is predicated on a good working relationship between the AI and human. We have presented guideline criteria for cooperative environments, for both the AI algorithm and the human machine interface. Humans are not designed or easily modifiable, so the responsibility falls on algorithm and human machine interface design to support teaming.

To apply AuI to SE, Augmented intelligence for Systems Engineering (AuISE) requires a conceptual framework to lay out the tasks done in systems engineering and how design is accomplished. ISO15288 is an effective framework for describing SE tasks, and the Agile Systems Engineering Lifecycle Model (ASELM) describes how known information can be applied, and new information learned about the target system. AuISE may change the nature of design, involving less direct modeling of the system, and more problem space and constraint definition.

Applications of AI in SE are still in early stages, lagging AI that is used in the product itself. Changes in the product space, such as autonomy and robotics, will only increase the need for AI in SE, as the amount of available information grows. By providing a conceptual framework and SE process hierarchy, allows for discussion of how to effectively use AI in systems engineering.

REFERENCES

[1] INCOSE SE Vision 2020

[2] OMG Wiki, http://www.omgwiki.org/MBSE/doku.php?id=start

[3]"What is AI", Jon McCarthy, http://jmc.stanford.edu/artificial-intelligence/what-is-ai/index.html

[4] A Brief History of Computer Chess, Erik J. Larson, https://thebestschools.org/magazine/brief-history-of-computer-chess/

[5] Garry Kasparov on AI, Chess, and the Future of Creativity (Ep. 22), https://medium.com/conversations-with-tyler/garry-kasparov-tyler-cowen-chess-iq-ai-putin-3bf28baf4dba

[6] Why artificial intelligence needs human-centered design - Deloitte Review, issue 22

[7] Introduction to the Agile Systems Engineering, Life Cycle MBSE Pattern, Bill Schindel, Rick Dove

[8] INCOSE Agile Working Group, 941.

[9] ISO/IEC/IEEE 15288:2015 Systems and software engineering -- System life cycle processes, https://www.iso.org/standard/63711.html

"Many jobs will continue to be lost to intelligent automation. But if you're looking for a field that will be booming for many years, get into human-machine collaboration."
-Kasparov, Deep Thinking