Systems Engineering-Based Design & Development of AI/ML Capabilities

James Llinas
University at Buffalo
Ranjeev Mittu
Naval Research Laboratory

Abstract
This discussion paper addresses the inherent technical nature of AI and ML technologies as they affect the challenge of realizing reliable and understandable advanced functional capabilities. Coupled with that challenge, and also discussed herein, is the challenge for the AI/ML communities to apply sound systems engineering methodological techniques that will aid in the construction of not only technically-sound capabilities but systems that are also cost-effective.

1. Challenges to Realizing the Potential of AI and ML Technologies

It should be obvious to anyone who monitors the advancement of technology that there has been (somewhat implicitly) a new "Spring" in the evolution of Artificial Intelligence (AI) and Machine Learning (ML) technologies. In the United States, these advances have been promulgated in no small part as a result of the Defense Advanced Research Projects Agency (DARPA) AI Next program that will invest some US $2B over the next five years in a wide variety of AI/ML initiatives. While some of this investment will be studying how to better evaluate AI/ML processes, very little is directed toward improved engineering methods to develop such capabilities. If formal systems engineering (SE) methodologies had been employed in the (long) history of AI and ML, this shortfall would not be a problem, as these communities would have already inculcated an SE discipline in prototyping and development; in that case, the question would have been the modernization of these SE techniques in keeping with advances in the science of SE. But this counterfactual is not the case. AI and ML have historically been involved more in the research domain rather than the development and integration domains; the accomplishments in these fields have largely been in the achievement of research prototypes with limited deployments to the commercial or defense/military sectors that would have required disciplined engineering development along with considerably more rigorous testing and evaluation. Another factor is the issue of definition—any review of the suggested definitions of AI and ML even today struggle with showing consistency;[1] so what is it that needs to built? A further complication is the multiplicity of specific core techniques both in AI and in ML but especially in ML. A KDNuggets poll [1] shows some 17 different ML methods being employed in the 2018/19-time frame (see Fig.1). Each of these methods have their own nuances in terms of the underlying algorithmic structure and thus each can require specialized engineering methods and test/evaluation/metric techniques for effective development. Note that the larger subgroup of these methods are non-deterministic/probabilistic, the character of which always demands, among other needs, for rigor, statistically-based testing and the methods of experimental design.

Among the many challenges then for AI and ML to be robustly integrated into wide ranges of application, there is the essential challenge with AI that there is not a currently *universally accepted approach* when it comes to implementation [2]. The far larger proportion of AI and ML implementations

[1] There are over 50 hits on the Internet for example if one searches on "Differences between AI, ML, and DL"

in use today are highly specialized [3]. Further, because of the underlying complexities and non-deterministic foundations of AI and ML technologies, there is the issue of unintended consequences and the assurances of ethical and legal behavior as generated by, or as supported by, AI and ML technologies. In the Department of Defense at least, these issues have given rise to new policies providing guidance for the ethical use of AI technologies [4], which in turn will have impacts on how such AI is developed and tested.

In this discussion paper, we present several viewpoints about the need for engineering methods that yield designed AI/ML processes with both predictable and understandable performance as well as guarantees regarding well-defined boundaries of behavior.

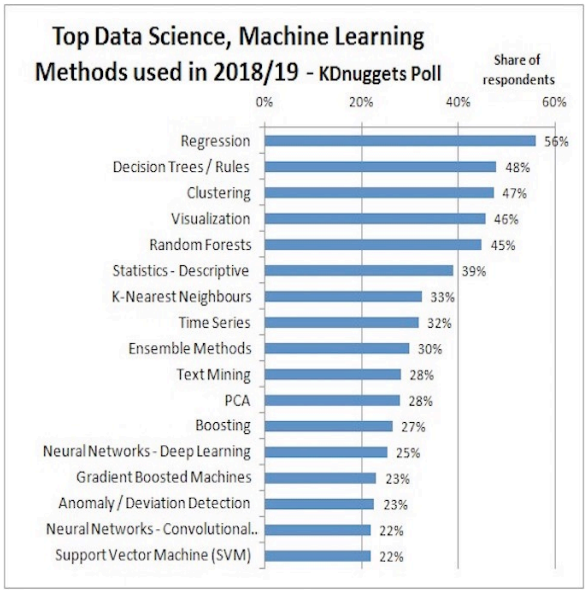

Figure 1. Range of ML Methods Employed [1]

2. Characteristics of Current-Day AI and ML that Challenge the Use of Systems Engineering

There are several other characteristics of AI/Ml that argue for a disciplined engineering approach for development and validation. Emergence is one such characteristic. Here we prefer the definition of emergence given by as a *property* of a complex system: "a property of a complex system is said to be 'emergent' and *it arises out of the properties and relations characterizing the system's simpler constituents*, but it is neither predictable from, nor reducible to, these lower-level characteristics" [REF: https://www.sebokwiki.org/wiki/Emergence]. There are many other definitions and taxonomies of emergence [5,6,7], but our focus is on the effects of emergence, not emergence per se. Chalmers identifies "strong" and "weak" emergence where strong emergence is not deducible even in principle from the laws of a lower-level domain, while weak emergence is only unexpected given the properties and principles of the lower-level domain. Neace and Chipkevich define weak emergent behavior as attributable to behavior of its constituents [8]; they have developed an engineering methodology designed to realize weak emergence as a desired property of a designed system. Desirable weak-emergent properties include self-healing, self-management, self-monitoring, etc., i.e., the desirable degrees of autonomous self-management. They introduce the ideas of network synchronization, functional coherence, and network entrainment as necessary mechanisms for weak emergence in a manufactured Complex Adaptive System (CAS), along with the software agents needed to intend and

achieve weak emergence in the CAS. In [8], Neace and Chipkevich intentionally differentiate between a natural and manufactured CAS's, and further divide manufactured CAS into those with and without agents. In this context, "agent" is used to distinguish an *autonomic software component* capable of self-management by having some degree of artificial intelligence [8]. This demarcation provides them with a framework for describing a methodology for requirements engineering of weak emergent goals in a manufactured agent-based CAS. Neace and Chipkevich [8] suggest an engineering scheme to manage the achievement of desirable weak emergence, and hopefully this approach would work. But the AI literature has many entries about the surprising and often undesirable behaviors of AI processes. Yampolskiy [9] addresses a position that discusses the unpredictability of AI in broad terms. In this paper, Yampolskiy surveys a number of works that discuss the related aspects for SE of AI Safety that addresses concepts of Unknowability [10] and Cognitive Uncontainability [11]. In [12], a lengthy review of specific problems in AI Safety are reviewed; the bulk of these problems are not dealing with concepts of unpredictability and emergent behaviors per se but issues that result from failures in systems engineering and design rigor. For example, in "reward hacking," the objective function that the designer writes down admits of a clever, "easy" solution that formally maximizes the function but perverts the spirit of the designer's intent (i.e., the objective function can be "gamed"). Specifically, Belani, et al. [13] assert that "there is still no widely used and specifically tailored process in place to effectively and efficiently deal with requirements suitable for specifying a software solution that uses machine learning." Assuming that this claim is true, then the systems engineering for ML starts without a framework for an evaluation, since requirements are the foundation for testing.

A further reflection of such concerns is given in Darpa's recent release of the call for the AIMEE program--Artificial Intelligence Mitigations of Emergent Execution [14]. Among the goals for the program is to learn how to prevent the propensity for emergent execution directly at the design stage when the system's programming abstractions and intended behaviors at a particular layer are translated into the more granular states and logic of the next computing substrate layer; by and large, this call by Darpa is a systems engineering challenge. The program goal is stated as: "The Artificial Intelligence Mitigation of Emergent Execution (AIMEE) will address the problem of anticipating, at a system's design stage, the models of emergent execution inherent in its design, and thus mitigate its propensities for exploitability before they lead to actual vulnerabilities in complete deployed systems."

3. Systems Engineering and Software Development Approaches for AI/ML Capabilities

For specification-based engineering and software development, methods and algorithms are nominated as ways to achieve or satisfy the specifications and a software engineer gives a computer the rules for how to make decisions and take actions. When developing software to achieve an AI/ML goal, the software engineer (more likely an AI/ML domain expert) curates and prepares the domain-specific data which is fed into the learning algorithms which are iteratively trained and continuously improved. A machine learning model can deduce from the data what features and patterns are important to the solution, without a human explicitly encoding this knowledge. In the first case, we have a specify-build-validate approach that has a number of positive attributes for a project management team. The major weakness of this process is that it forces a premature commitment to a specification before the needs of the system procurers are fully understood. Historically (i.e., before the current "AI Spring"), the AI/ML community recognized this weakness and developed a process that formulated ideas—to build a prototype—and assessed the ideas [15]. The development of AI capability has thus induced a shift in design thinking from deductively-based approaches to inductively-based ones. In a sense, this shift shifts the perspective of the engineer from a software engineer to a data scientist, as shown in Figure 2.

The inductive approach is focused on innovation and rapid prototyping in the cycle of iterative ideas as new approaches are rapidly prototyped and tested. This rapid prototyping process converges to a solution considered to be adequate at the algorithmic level, but it may not be best in the systemic sense for the system into which this AI/ML capability is integrated. This approach then can lead to a hidden "technical debt," as discussed in Sculley et al. [16]; the notion of technical debt is related to the idea of hidden long-term maintenance cost for software. A key point is that, as regards lifecycle costs, the development phase is a relatively small part. Further, as we are still in the "AI Spring," the reality of lifecycle costs for systems involving AI/ML technologies has not yet been experienced. While the distribution of the degree of AI/ML functionalities in real-world systems is not very well known, Sculley et al. assert that even though ML typically comprises a small part of a larger system, the within-ML and ML-system complexities and their pathologies can lead to high lifecycle costs.

- Developing AI processes has imputed a Shift in Design Thinking
  - From Deductive reasoning to Inductive reasoning
  - From clear Specifications to Goals
  - From tested guarantees to Best Effort

Data Scientist	Software Engineer
Fixed datasets for training and evaluation	Product-oriented; concerned about cost, performance, stability
Focus on accuracy	Quality related to customer satisfaction
Prototyping operations	Solutions often need to be scaled, adapted to changing data and size
Modeling expertise; feature engineering	Software maintainability crucial aspect of efficiency and life-cycle cost
Model size, updating ability, implementation stability not a focus	Issues of security, safety, ethics typically involved

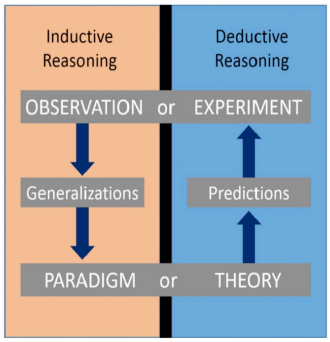

Figure 2. Shifts in Design Thinking and Software Development

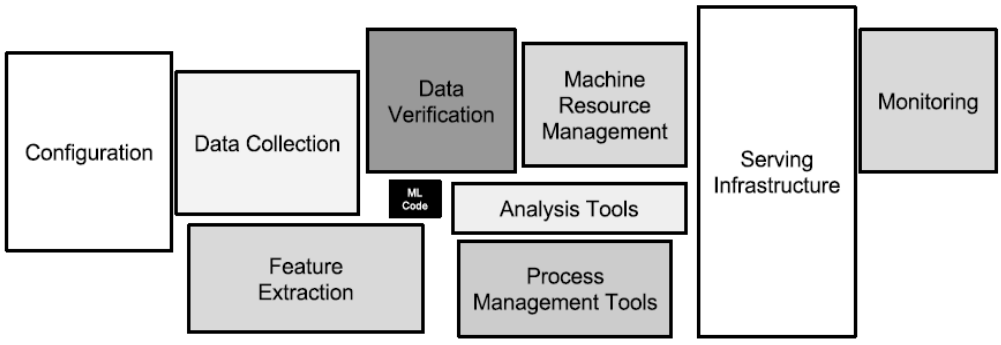

Figure 3. Small Proportionality of ML in Systems [16]

Sculley et al. [16] further argue that ML systems have a special capacity for incurring technical debts, because they have all of the maintenance problems of traditional code plus an additional set of ML-specific issues. Quoting from [16]:

> "Traditional software engineering practice has shown that strong abstraction boundaries using encapsulation and modular design help create maintainable code in which it is easy to make isolated changes and improvements. Unfortunately, it is difficult to enforce strict abstraction boundaries for machine learning systems by prescribing specific intended behavior. Indeed, ML is

required in exactly those cases when *the desired behavior cannot be effectively expressed in software logic without dependency on external data*."

Their paper goes on to elaborate a number of other categories of issues that add to this debt, in part related to the fact that in many systems the ML code is a small proportion of a system's code, and that realizing the relatively small benefits from the ML component at the cost of massive increases in system complexity are rarely a wise practice. Even the addition of one or two seemingly innocuous data dependencies can slow further progress. They assert that "Paying down ML-related technical debt requires a specific commitment, which can often only be achieved by a shift in team culture."

4. Issues Related to Test and Evaluation

The major focus of the AI and ML communities has historically been on the development and testing of a specific AI/Ml capability that we call an "agent." There are subtleties in the testing of AI/ML processes, as they do not "crash" as many other types of software do, but instead they "behave oddly or badly". Similarly, notions of "bugs" are unclear, especially for ML as bad features are notionally similar to bugs. Such testing has also been focused on performance testing, as regards the intended functionality or capability. A review of papers related to test and evaluation of AI/ML capabilities shows that, at least at present, the focus has been on measures of performance (MOPs). For example, in the Deep Learning (DL) domain, the focus has been on timeliness: computational efficiency, accuracy, operational robustness, and confidence [17]. Across many DL and ML studies, for example, the standard pattern recognition metrics are the basis of many comparative studies to determine the best-of-breed techniques (as noted previously in Fig. 1, many such methods exist). Many papers and testing approaches focus on particular performance-centric metrics. For example, classification performance can be measured by a range of evaluation measures including accuracy, true and false positive rate, precision and recall, F-scores, Areas Under (ROC) Curves, and Brier scores. However, as Flasch [18] points out, even in the simplest scenarios, a single aggregated measurement is insufficient to accurately reflect the performance of a machine learning algorithm. Predictive accuracy has been often used as the major metric for ML evaluation, but Provost, Fawcett, and Kohavi [19] pointed out the limitations of using predictive accuracy as the gold standard in classification; they were early proponents of ROC analyses, especially for changing class and cost distributions. Flasch [18] makes a very interesting point about many papers using *both* F-scores and Areas Under the ROC Curve (AUC) together whereas these metrics compute different things from different viewpoints. So, even if the focus is on the performance of the AI/ML system component, there are issues about selecting the *set of metrics* that best apply for the intended application.

Another level of testing is the class of model-based black-box testing techniques that aim to assess the correctness of a reactive system, i.e., the implementation under test (IUT) with respect to a given specification (assuming a specification has been properly constructed). The IUT is viewed as a black-box with an interface that accepts inputs and produces outputs. The goal of model-based black-box testing is to check if the observable behavior of the IUT "conforms" to a specification with respect to a particular conformance relation. In the case of Machine Learning models, there are no expected values beforehand in that ML models output a prediction. Given that the outcome of Machine Learning models is a prediction, it is not easy to compare or verify the prediction against an expected value that is not known beforehand. But for non-deterministic operations within an ML agent, there is no easy way to provide an expectation. This void has given rise to the idea of pseudo-oracles and "metamorphic" testing. Metamorphic relations represent a set of properties that relate multiple pairs of inputs and outputs of the target program/application such that proportional results due to changes in design parameters can be estimated [20].

Finally, we discuss the issue of "DevOps" and "MLOps" for the overarching processes of development and testing of AI and ML capabilities in the context of deployed systems (see [21] for an overview of MLOps). These terms and concepts both relate to the idea of engineering culture and the synthesis of work teams that historically have often been isolated/siloed—the development team and the team responsible for the deployed operational system. This separation of course finds that the teams remain unified on the issue of testing related to the deployed system, but not unified for a developmental prototype. Testing in DevOps spans the whole software development and delivery lifecycle. Testers are no longer just focusing on functional testing and feature verification. These philosophies or codes of practice are still in their early stages as regards defining proven new approaches to testing, but it seems that these "Ops" ideas for AI and ML are closely linked to the Agile Development model where testing is done in every iteration of development; the idea is to bring such processes to play at the systems level. Doing so will also impact the design of workflow processes and models. Defining optimal workflows is complicated; one example of such a workflow is shown in Figure 4 [22].

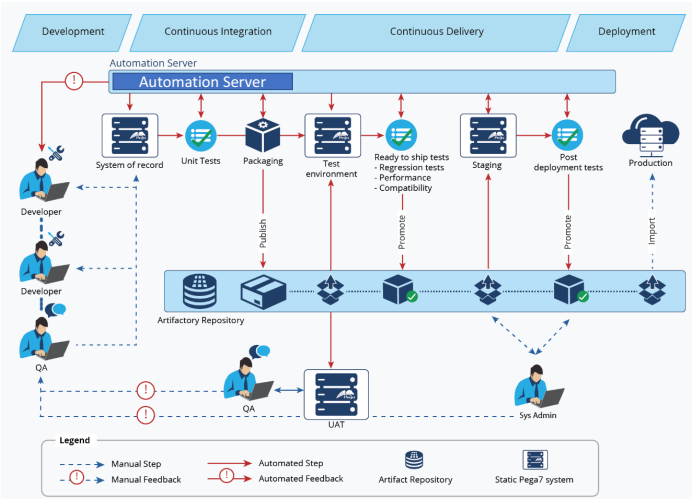

Figure 4. Example of Workflow Complexity Toward Realizing MLOps Type Processes [22]

5. Summary

In this discussion paper, we have offered views on several topics related to the intersection of Systems Engineering and AI/ML technologies and capability development. There are many issues to be discussed that bear on the nature of this intersection, but of course, not all such issues were raised here. Included here is the nature of the history of AI and ML, having been born in their earliest days as revolutionary technologies, and then going through their so-called Winters and Springs, but within those cycles and up to this moment in time, they are still considered revolutionary and thus not mature. The concern for the SE/AI-ML intersection is that the current-day capabilities are being expedited too quickly into deployed systems. For some applications, this rapid deployment is of moderate concern, but for many applications, there are very critical aspects derived from a too-rapid deployment, in the worst cases bearing on life and death situations. It is in society's best interest that this intersection be studied and implemented in the best possible way, with assurances not only with regard to technical and functional aspects, but for ethical, legal, and safety aspects as well.

6. References

[1] https://www.kdnuggets.com/2019/04/top-data-science-machine-learning-methods-2018-2019.html)
[2] https://www.cuelogic.com/blog/design-thinking-for-ai
[3] https://www.forbes.com/sites/bernardmarr/2017/07/13/the-biggest-challenges-facing-artificial-intelligence-ai-in-business-and-society/#33ea75ef2aec
[4] Defense Innovation Board, AI Principles: Recommendations on the Ethical Use of Artificial Intelligence by the Department of Defense, from https://admin.govexec.com/media/dib_ai_principles_-_supporting_document_-_embargoed_copy_(oct_2019).pdf; Nov. 2019
[5] Holland, O. T. 2007. "Taxonomy for the Modeling and Simulation of Emergent Behavior Systems". In Proceedings of the 2007 Spring Simulation Multiconference, 28–35.
[6] Jochen Fromm, Types and Forms of Emergence; http://arxiv.org/abs/nlin.AO/0506028
[7] Chalmers, D. J. (2006) Strong and Weak Emergence, in: P. Clayton & P. Davies (eds), The Re-emergence of Emergence, Oxford, UK; Oxford University Press, pp. 244–256.
[8] Kerrin S. Neace, Mary Beth A. Chipkevich, Designed Complex Adaptive Systems Exhibiting Weak Emergence, NAECON 2018 - IEEE National Aerospace and Electronics Conference, 23-26 July 2018, Dayton, OH.
[9] Roman V. Yampolskiy, Unexplainability and Incomprehensibility of Artificial Intelligence; https://arxiv.org/ftp/arxiv/papers/1907/1907.03869.pdf
[10] Vinge, V. Technological singularity, in VISION-21 Symposium sponsored by NASA Lewis Research Center and the Ohio Aerospace Institute. 1993.
[11] Cognitive Uncontainability, in Arbital. Retrieved May 19, 2019 from https://arbital.com/p/uncontainability/
[12] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Man´e. Concrete problems in AI safety, from arXiv preprint arXiv:1606.06565, 2016.
[13] Hrvoje Belani, Marin Vuković, and Željka Car. 2019. Requirements Engineering Challenges in Building AI-Based Complex Systems, from arXiv preprint arXiv:1908.11791 (2019).
[14] https://intelligencecommunitynews.com/darpa-releases-aimee-call-for-proposals/
[15] I. Sommerville, "Artificial intelligence and systems engineering," Prospects for Artificial Intelligence: Proceedings of AISB'93, 29 March-2 April 1993, Birmingham, UK, vol. 17, p. 48, 1993
[16] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, and J.-F. Crespo. Hidden technical debt in machine learning systems. In Neural Information Processing Systems (NIPS). 2015
[17] Blasch, E., et al., Deep Learning Measures of Effectiveness, NAECON 2018 - IEEE National Aerospace and Electronics Conference, Dayton, OH, July 2018
[18] Flasch, P., Performance evaluation in machine learning: The good, the bad, the ugly and the way forward. In: Proceedings of the 33th AAAI Conference on Artificial Intelligence (2019)
[19] Provost, F. J.; Fawcett, T., and Kohavi, R. 1998. The case against accuracy estimation for comparing induction algorithms. In Int Conf on Machine Learning, 445–453.
[20] https://vitalflux.com/qa-metamorphic-testing-machine-learning/
[21] Sato, Kaz. "What is ML Ops? Solutions and best practices for applying DevOps to production ML services". Artificial Intelligence Conference. O'Reilly. Retrieved 10 October 2018.
[22] https://community.pega.com/knowledgebase/articles/devops-release-pipeline-overview/devops-release-pipeline-overview