Continuous Experimentation of Artificial Intelligence Software - A Research Agenda

Anh Nguyen-Duc
University of South Eastern Norway
Bø, Norway
Anh.Nguyen.duc@usn.no

Pekka Abrahamsson
University of Jyvaskylä
Jyväskylä, Finland
pekka.abrahamsson@jyu.fi

ABSTRACT

Moving from experiments to industrial level AI software development requires a shift from understanding AI/ ML model attributes as a standalone experiment to know-how integrating and operating AI models in a large-scale software system. It is a growing demand for adopting state-of-the-art software engineering paradigms into AI development, so that the development efforts can be aligned with business strategies in a lean and fast-paced manner. We describe AI development as an "*unknown unknown*" problem where both business needs and AI models evolve over time. We describe a holistic view of an iterative, continuous approach to develop industrial AI software basing on business goals, requirements and Minimum Viable Products. From this, five areas of challenges are presented with the focus on experimentation. In the end, we propose a research agenda with seven questions for future studies.

CCS CONCEPTS

• **Software and its engineering → Software development methods**.

KEYWORDS

Industrial Artificial Intelligence, AI software, AI system, Continuous Experimentation

ACM Reference Format:
Anh Nguyen-Duc and Pekka Abrahamsson. 2020. Continuous Experimentation of Artificial Intelligence Software - A Research Agenda. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3368089.3417039

1 INTRODUCTION

Recent technological advancements with computing resources, big data, algorithmic maturities and scientific tools have enabled a lot of opportunities with AI for business, industries and societies. Almost all organizations today, from private to public sectors, have some forms of AI initiatives [3]. Despite increased interest in and adoption of AI, 85% of AI projects ultimately fail to deliver on their intended promises to business. One of the most serious risks in software development is building features that no one wants or needs [5, 15].

In a typical AI software development project, AI scientists and software engineers need to do different types of tasks, not only scientific work, but also business-relevant, for instance, defining Key performance indicator (KPI) metrics, hypothesis testing, and applying models to business [8]. Many AI projects employ a polymath scientist, who "do it all", without any specific guidelines. It is a growing interest to systematically address business value at the engineering level of AI software projects. Moreover, in many cases, AI software development is an "*unknown unknown*" problem. Many companies develop different AI models without really knowing how to realize their business value. Another issue is understanding and expectations from an AI project might be vague and slowly elicited [12].

In such an "*unknown unknown*" domain, both applied domain knowledge and algorithmic understanding of AI models are both needed to elicit and validate feasible scenarios of AI software usage. Contemporary Software Engineering research has several frameworks for this type of project, such as continuous experimentation [5], Minimum Viable Product frameworks [4] and Lean startup [15]. The main idea is to perform several experiments about the developing products and conduct hypotheses to test the fitness between the product and the customer or market needs. Such approaches will shape the software's implementation in the way that customers can frequently involve in AI software releases, continuously accept or reject a release according to their business metrics, and make rollback possible before wasted efforts become significant. To contrast with much research currently focusing on verification activities of AI, i.e. model testing and debugging [19], we propose a conceptual model of continuous experimentation of AI software, with the focus on acceptance and suitability with external stakeholders.

The paper is organized as follows, Section 2 presents the research background, Section 3 presents the conceptual framework of continuous experimentation, Section 4 presents the research gaps and Section 5 discusses and concludes the paper.

2 BACKGROUND

2.1 Systematic approaches for development of AI software

We refer to AI software research as studies about engineering methods for software systems that integrate with, or enabled by AI/ Machine Learning (ML) models, data, and its surrounding infrastructure. Although AI/ ML are often modules in larger systems, they require a substantial resource, infrastructure and data to be

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11…$15.00
https://doi.org/10.1145/3368089.3417039

included [16]. AI software research does not only deal with AI/ ML models themselves but also cover processes, infrastructure, data and engineering approaches that could lead to better AI software [11].

Methodological work on AI software development includes a process reported from Microsoft [2], a five-step "*stairway to heaven*" for AI model development [10], or a maturity framework for AI development [1] and continuous delivery for AI software [14]. The common shortage of these models is a vague description of customer needs and the connection between the business and development aspects of AI software. SE research starts to reveal SE challenges for developing AI software, such as challenges of managing datasets, building models, training and evaluating models, and deploying models [10]. AI Engineering specifies particular SE tasks, such as managing multiple AI models, resuse of AI models, model and software integration, monitoring and logging, A/B testings of models and data quality management [3]. All these tasks are relevant in technical space, but they are currently disconnected to business and domain knowledge space.

2.2 Continuous Software Experimentation

Continuous software engineering movement closely resembles the concept of flow found in lean manufacturing and product development [6]. Continuous development is a state-of-the-art methodological approach to developing software products, constantly conducting systematic experiments to validate user needs hypotheses. Continuous development consists mainly of engineering activities, such as integration, delivery, deployment, and evolution. The ideal development process should also cover business and operation aspects. Involving customers and business stakeholders in iterations of software development lead to business-driven experimentation with repeated Build-Measure-Learn [5]. The central artifact of the approach is Minimum Viable Products (MVPs), a version of a new product that realizes a product idea and facilitates the collection of user feedback on it. Techniques such as A/B testing are widely used where features such as text, layouts, images and colors are manipulated systematically and customer reaction is monitored [13]. A continuous experimentation system has been adopted for building, testing and operating software as they are adopting in large companies, such as Ericsson, Google and Microsoft.

3 THE CONCEPTUAL FRAMEWORK OF CONTINUOUS EXPERIMENTATION FOR ARTIFICIAL INTELLIGENCE

In an experimentation system, software engineers will proceed iteratively and propose specifications of both functional and non-functional AI software requirements, develop MVPs, and then communicate to stakeholders to see whether they have gotten the specifications right. Each iteration might provide insights about what was missing in the previous specification and how we might develop with a better one. If the AI/Machine Learning (ML) model is sufficiently specified, its implementation against specified properties is straightforward. However, customers often do not know what they want initially; hence communication of their needs is often an iterative process. Due to the system's complexity, requirements need to go through a lower-level specification and analysis, which is often a part of AI model development. From these ideas, we propose a conceptual model of continuous experimentation of AI software. As shown in Figure 1, there are four pillar artifacts in a continuous validation loop:

- Customer needs represents the expectation of relevant stakeholders, often the project owner (customers or internal demands). Customer needs can be expressed as a wanted business use case.
- Requirement specification is a detailed description of the software system to be developed with its functional and non-functional requirements. The specification of AI software non-functional requirements is a consistent, complete and verifiable semantic representation of customer needs.
- Lower-level specification and analysis is a formal description of non-functional requirements, their constraints and the relationships among them.
- Minimum Viable Products is a viable set of software release, AI models and associated data that can be demonstrated to external stakeholders.

Activities such as model evaluation, deployment, and monitoring can be represented as major blocks in the holistic model too. However, we keep them as sub-tasks inside the MVP for AI software block to highlight the focus on validating business goals and ethical attributes. The process from customers to a release of MVP can occur at different levels. We propose three levels of experiment relating to how an AI model is designed and constructed:

- Experiments with idealizing models: the first level feedback loop when the AI model is planned and designed. Alternative artifacts, i.e. Wizard of Oz or piecemeal MVPs [4, 15] are built to simulate the expected functions of the AI software
- Experiments with training models: the second level feedback loop when the AI model is built with training dataset. Various model development is conducted to understand the model capacity.
- Experiments with operating models: the third level feedback loop when the AI model is under operation. Experimentation-compatible quality metrics are needed to detect abnormalities and trace to model output and input.

4 RESEARCH AGENDA

Five research topics are identified, naming from C1 to C5.

4.1 C1 - Specifying quality and ethical requirements of AI software

Many customer problems solved by AI/ ML are only specified implicitly by their technical characteristics. Business goals are often not specified in a typical Software Engineering manner. Current AI software development processes loosely connect to business context, vaguely mention customer needs as *"business understanding"*, *"model requirement"* or *"understanding of application domain"*. The specifications of these attributes, as quality requirements or in other formats, needs to reflect customer needs, or connecting to business goals. Moreover, they should be detailed and measurable to guide downstream SE activities, such as design, database storage, model building and verification. Applied AI research focuses much

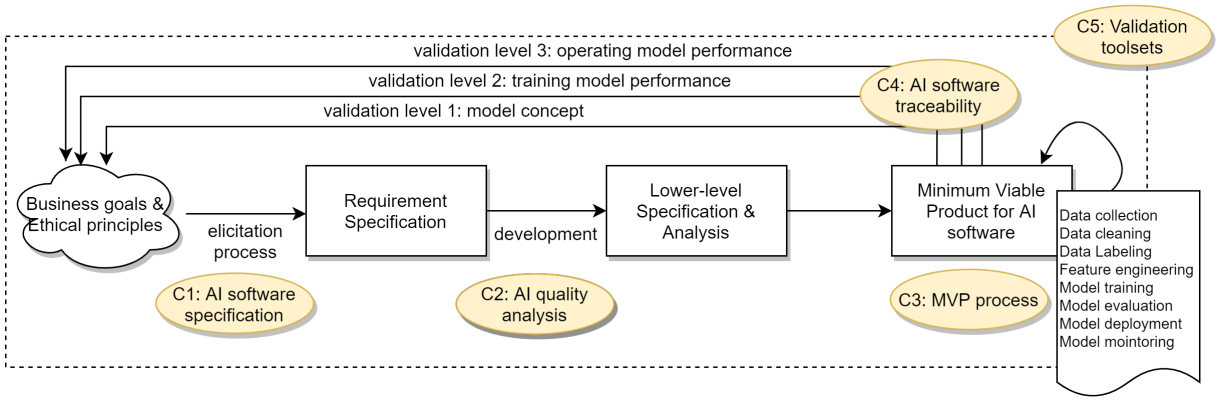

Figure 1: A holistic view of continuous experiment loops for AI software

on testing AI model quality attributes, such as accuracy, explainability, fairness, dependability, or trustworthiness [19]. However, these quality attributes are often taken for granted without questioning why customers need them and to what extent they should be achieved. Furthermore, research starts to look at methodological approach to identify, specify and include ethical requirements into AI software [17].

4.2 C2 - Modelling AI software quality and ethical attributes

Once AI quality attributes are properly specified, it is necessary to understand the relationship among these attributes and to software system contexts. Common SE quality modelling approaches either specify a prescriptive set of quality characteristics or metrics, or follow a method to guide the derivation of quality models [9]. Either approach would require an ontological and conceptual understanding of the quality attributes. The holistic view of such relationships between AI/ML models, their quality attributes (including computational cost) and business goals, constraints, and characteristics of the datasets, as shown in Figure 2. Moreover, AI ethics should aslo be integrated into a quality model as a quality attribute. While such analysis is essential for guiding model development and testing, recent work is only just beginning to explore some of these tradeoffs in AI software. [3, 7].

4.3 C3 – Keep track of model traceability

From a software engineering perspective, traceability supports demonstrating that each requirement has been satisfied and that each system component satisfies a requirement, linking requirements to other development artifacts, such as model and data, understanding their source and rationales, capturing the information necessary to understand the evolution of requirements and associated artifacts. In AI development, changes are made continuously during data processing, hyperparameter optimization and model experimentations. A versioning system for AI needs to keep track of changes for both models and associated datasets [18]. It is not known how these technical changes are traced to requirements, project constraints, and how these links are maintained in the phases of a experimentation process.

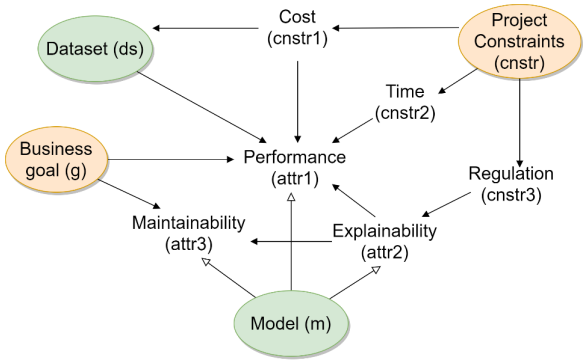

Figure 2: A holistic view of continuous experiment loops for AI software

4.4 C4 –Defining MVPs for an AI model

The key element of MVP methodology is to define the "minimum viable" parts. Besides functional features, MVP for AI software needs to capture the desired business value and ethical requirements in a demonstrable release. This is an open research area as there is currently unknown how to define minimum viable quality, particularly ethical ones. Holistically, MVP for AI captures a version of AI model, associated data and surrounding software components. Not every experimental version of AI model can be seen as an MVP. A set of minimum acceptance criteria should be defined so that an MVP can be accepted or not according to business and ethical requirements. The acceptance criteria can be formulated only when quality specification and analysis is in place.

4.5 C5 – A toolset supporting continuous experimentation

To build a MVP release for AI software, we need to go through a complete development pipeline. Every step in the pipeline, such as data preparation, model building, and model monitoring are essential for achieving the model capacity. The importance of tools support for AI software development, such as versioning control systems, issue

tracking and automated testing frameworks, is recognized [2]. Having a seamless development experience covering all different stages of the software methodology is essential for automation. Compared with traditional tools, the inherent uncertainty of data-driven learning algorithms and complex component entanglement caused by hidden feedback loops impose substantial changes that were previously well understood in SE [16]. Recent work about tooling for AI continuous integration [14, 18] is limited to model performance (accuracy and precision) without considering business and ethical concerns. Automating such a business-to-development pipeline can only be achieved under a specific methodological framework. It is possible to look at existing tools, i.e. Gitlab, Azure DevOps, Jenkins, Azure ML, Cloud AutoML, Kubeflow and other open-source frameworks, and leverage them on implementation of a complete continuous experimentation pipeline.

5 DISCUSSION AND CONCLUSIONS

Adopting continuous experimentation for AI software development highlights a number of Software Engineering challenges that need to be overcome if the concept is to be successful. Finding the solutions requires inter-disciplinary research, which goes beyond SE, data science or AI themselves. The research intersecting SE and AI is currently in an infant stage. Much of research attention now is focusing on verification, i.e. testing of AI software. In comparison to that, research about earlier SE activities, i.e. requirement specification, analysis and evaluation, is overlooked. This research agenda addresses several key scientific challenges that are road-blocking for the integration of business value and ethical principles. Several Research questions can be derived from the proposed research agenda, for examples:

- RQ1: In which way AI software can be specified as software requirements?
- RQ2: How can we improve communication during AI software development among software engineers, data scientists, domain experts and project owners?
- RQ3: How can AI software quality can be modelled in the connection to AI algorithms, data and business goals?
- RQ4: How can the tradeoffs among AI quality can be traced over time?
- RQ5: What are possible approaches to define MVPs for AI software?
- RQ6: How do we capture feedback for AI MVPs?
- RQ7: How can toolsets be developed to support the continuous experimentation of AI software?

Answering our RQs will require a revisit of existing SE knowledge, i.e. processes, models, practices and tools. Current literature shows mostly problem reports, with a few research about the evaluation of methodologies or toolsets. The research in this direction will shed the way for future research by systematic approaches, referencing conceptual models and benchmarking industrial best practices.

REFERENCES

[1] Rama Akkiraju, Vibha Sinha, Anbang Xu, Jalal Mahmud, Pritam Gundecha, Zhe Liu, Xiaotong Liu, and John Schumacher. [n.d.]. Characterizing machine learning process: A maturity framework. ([n. d.]). arXiv:1811.04871 http://arxiv.org/abs/1811.04871

[2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. [n.d.]. Software engineering for machine learning: a case study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice* (2019-05-27) *(ICSE-SEIP '19)*. IEEE Press, 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

[3] Jan Bosch, Ivica Crnkovic, and Helena Holmström Olsson. [n.d.]. Engineering AI Systems: A Research Agenda. ([n. d.]). arXiv:2001.07522 http://arxiv.org/abs/2001.07522

[4] Anh Nguyen Duc and Pekka Abrahamsson. [n.d.]. Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups. In *Agile Processes, in Software Engineering, and Extreme Programming* (2016) *(Lecture Notes in Business Information Processing)*, Helen Sharp and Tracy Hall (Eds.). Springer International Publishing, 118–130.

[5] Fabian Fagerholm, Alejandro Sanchez Guinea, Hanna Mäenpää, and Jürgen Münch. [n.d.]. The RIGHT model for Continuous Experimentation. 123 ([n. d.]), 292–305. https://doi.org/10.1016/j.jss.2016.03.034

[6] Brian Fitzgerald and Klaas Jan Stol. [n.d.]. Continuous software engineering: A roadmap and agenda. 123 ([n. d.]), 176–189. https://doi.org/10.1016/j.jss.2015.06.063

[7] Jennifer Horkoff. [n.d.]. Non-Functional Requirements for Machine Learning: Challenges and New Directions. In *2019 IEEE 27th International Requirements Engineering Conference (RE)* (2019-09). 386–391. https://doi.org/10.1109/RE.2019.00050 ISSN: 2332-6441.

[8] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. [n.d.]. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering* (2016-05-14) *(ICSE '16)*. Association for Computing Machinery, 96–107. https://doi.org/10.1145/2884781.2884783

[9] Michael Klas, Constanza Lampasona, and Jurgen Munch. [n.d.]. Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications* (2011-08). 341–348. https://doi.org/10.1109/SEAA.2011.62 ISSN: 2376-9505.

[10] Lucy Ellen Lwakatare, Aiswarya Raj, Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. [n.d.]. A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In *Agile Processes in Software Engineering and Extreme Programming* (2019), Philippe Kruchten, Steven Fraser, and François Coallier (Eds.). Springer International Publishing, 227–243. https://doi.org/10.1007/978-3-030-19034-7_14

[11] Tim Menzies. [n.d.]. The Five Laws of SE for AI. 37, 1 ([n. d.]), 81–85. https://doi.org/10.1109/MS.2019.2954841 Conference Name: IEEE Software.

[12] Anh Nguyen-Duc, Ingrid Sundbø, Elizamary Nascimento, Tayana Conte, Iftekhar Ahmed, and Pekka Abrahamsson. [n.d.]. A Multiple Case Study of Artificial Intelligent System Development in Industry. In *Proceedings of the Evaluation and Assessment in Software Engineering* (2020-04-15) *(EASE '20)*. Association for Computing Machinery, 1–10. https://doi.org/10.1145/3383219.3383220

[13] Helena Holmstrom Olsson, Hiva Alahyari, and Jan Bosch. [n.d.]. Climbing the "Stairway to Heaven" – A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *Proceedings of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications* (2012-09-05) *(SEAA '12)*. IEEE Computer Society, 392–399. https://doi.org/10.1109/SEAA.2012.54

[14] Cedric Renggli, Bojan Karlaš, Bolin Ding, Feng Liu, Kevin Schawinski, Wentao Wu, and Ce Zhang. [n.d.]. Continuous Integration of Machine Learning Models with ease.ml/ci: Towards a Rigorous Yet Practical Treatment. ([n. d.]). arXiv:1903.00278 http://arxiv.org/abs/1903.00278

[15] Eric Ries. [n.d.]. *The lean startup: how today's entrepreneurs use continuous innovation to create radically successful businesses*. OCLC: 693809631.

[16] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. [n.d.]. Hidden technical debt in Machine learning systems. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (2015-12-07) *(NIPS'15)*. MIT Press, 2503–2511.

[17] Ville Vakkuri, Kai-Kristian Kemell, and Pekka Abrahamsson. [n.d.]. ECCOLA – a Method for Implementing Ethically Aligned AI Systems. ([n. d.]). arXiv:2004.08377 http://arxiv.org/abs/2004.08377

[18] Tom van der Weide, Dimitris Papadopoulos, Oleg Smirnov, Michal Zielinski, and Tim van Kasteren. [n.d.]. Versioning for End-to-End Machine Learning Pipelines. In *Proceedings of the 1st Workshop on Data Management for End-to-End Machine Learning* (2017-05-14) *(DEEM'17)*. Association for Computing Machinery, 1–9. https://doi.org/10.1145/3076246.3076248

[19] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. [n.d.]. Machine Learning Testing: Survey, Landscapes and Horizons. ([n. d.]). arXiv:1906.10742 http://arxiv.org/abs/1906.10742