An Agile Software Development Life Cycle Model for Machine Learning Application Development

Romesh Ranawana
Enterprise Machine Learning (Pvt) ltd
Colombo, Sri Lanka
romesh@enterpriseml.ai

Asoka S. Karunananda
Department of Computational Mathematics
Moratuwa, Sri Lanka
asokakaru@uom.lk

2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI) | 978-1-6654-0655-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/SLAAI-ICAI54477.2021.9664736

Abstract— **Software development teams are often hampered when aligning machine learning production with standard software development processes. Iterative experimentation is needed to address the inherent complexities of data collection and preparation, model entanglement, and the technical debt of machine learning. The complexity of this process is compounded due to dependencies on the production environment and real-time data. We propose a unified framework which facilitates the planning, development, and deployment of a machine learning application through parallel processes for software and machine learning engineering. This allows for the risk of both the project and machine learning development to be significantly reduced through continuous integration, evaluation, and production. The framework, named MLASDLC, unifies concepts from standard software development life cycle methodologies (SDLC), development operations (DevOps) and machine learning operations (MLOps) to present a framework for the development of machine learning applications.**

*Keywords— **Machine learning, MLOps, DevOps, agile, SDLC, software development life cycle, experimentation, data-centric***

I. INTRODUCTION

Machine learning is now widely accepted as a key tool by the information technology sector [1] [2]. Despite significant investment, the number of machine learning based applications which progress to production is modest. A recent report by VentureBeat [3] reports that at least 87% of machine learning projects in industry do not make it into production. [4] reports that most projects take at least 6 months to build a working production grade prototype. The main reasons for the low success rate have been identified as the challenges involved with deployment, scaling, and versioning a production ready machine learning system [4]. Industrial demands for quick turn-around, continuous development and update, and continuing training do not blend well with machine learning model development. Machine learning model development is non-linear and is therefore difficult to align with sequential development processes. Iterative experimentation and data engineering is needed to develop and fine-tune a machine learning model. Data collection and processing is also time consuming and costly. Machine learning development poses other challenges due to data dependencies, model entanglement, and other hidden technical debts. The development and refinement of machine learning operations (MLOPs) are responses to some of these challenges. Data-centric model development is also being increasingly used as a method to reduce the risk and uncertainty of machine learning production. These processes are, however, notoriously difficult to align with standard software development practices and has therefore resulted in many companies developing their own customized processes for machine learning application development [5]. Several reports [5] [6] have shown that more discussion is needed on integrated software development models for machine learning application development.

. In section II of this document, we provide a overview of the history of software development life cycle methodologies, the key principles of modern software development and governing principles. The key feature of the proposed framework is the ability to transition from coding centric development cycles to data-centric engineering. Then, in section III, we discuss the steps and challenges of machine learning production and the methodologies which have developed to integrate machine learning systems with end-user systems. In section IV of this paper, we introduce a framework, the machine learning application software development lifecycle (MLASDLC), which closer aligns widely used software development methodologies with machine learning production. This framework is designed to facilitate the continuous agile development, training, and deployment of all three machine learning application components - the machine learning models, the machine learning support system, and the user application - in parallel, and thereby significantly reduce the risk of machine learning application development.

II. SOFTWARE DEVELOPMENT LIFE CYCLES

A SDLC is a collection of methodologies for structuring, planning, and executing tasks related to software development [7] [8]. SDLCs have developed and evolved over the past six decades in reaction to the industry demands for greater control over cost, duration and risk management. Software development started in the 1950s with the development of the first computers. However, the ad-hoc approach to software engineering following during those early days did not translate well when used for larger and more complex software products [8]. The need for greater structure and process for software development led to the development of the Waterfall model by Royce [9] in 1970. This and later sequential models were based on the phased program planning (PPP) model developed by NASA [10]. The next evolution, iterative software development [11], was a result of growing software complexity, and the need to better manage risk in commercial projects. Instead of a single sequence of phases, development was conducted over several short iterative phases. The advent of object-oriented programming allowed large software systems to be better modularized, and thereby facilitating easier iterative development methodologies. The spiral model [12] allowed for greater iteration through the subsequent development phases and introduced better methods for managing risk [13].

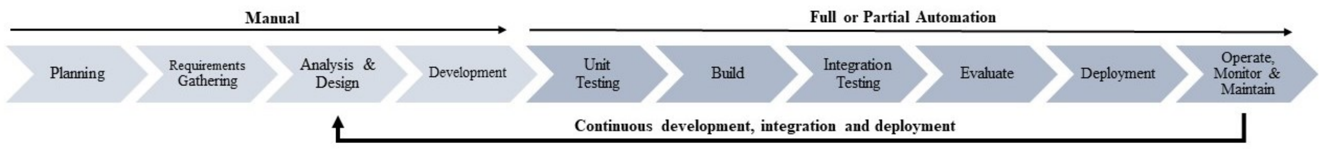

Fig. 1. Agile Continuous Delivery SDL

Fig. 2. Continuous development, testing, deployment, and monitoring

The unified process model [13] built upon these SDLCs and piggy-backed on the widespread adoption of object-oriented programming to significantly reduce the time between development iterations.

Agile methodologies were introduced due to the need for rapid feedback-based software development. They are based on the unified process model and are designed to allow flexibility and adaptability to change through continuous planning, improvement, development, and deployment of the system. Multi-disciplinary agile teams share collective responsibility for all phases of development. This is a significant shift from previous models which used experts for each phase of the development cycle. Takeuchi [10] in 1986, while introducing this concept, compared this to a game of rugby where all the members of the team work together from start to finish. Greater comprehension of the project and its goals allow each member of the team to be better informed and make better decisions. Agile teams are typically cross-functional and more efficient as team members share greater responsibility. This method also encourages trial and error, essential for most technical development processes. The agile unified process (AUP) [14] was introduced in 2010 and is now the dominant methodology used for most software development. Continuous Development and Delivery

Commercial demands for new functionality and the ability to react quickly to market changes has led to the development of methodologies which support continuous evaluation, development, and delivery of software systems. The key metric is how quickly a new version can be designed, development, tested and deployed. The explosion of the software as a service (SaaS) industry has also led to increased interest in processes which allow rapid change [15]. These methodologies, called development operations (DevOps) [16], combine SDLCs with IT operations to facilitate the continuous update of software. DevOps engineers collaborate with the agile team through the entire lifecycle to automate many of the phases of the development process.

Fig.1 shows a summary of the common steps used by commercial software development teams [10]. Development typically follows an iterative model where these steps are repeated in short iterations (Fig.2). Automation of the key steps allows the teams to reduce the time between iterations and forms the backbone of CI/CD based development methodologies.

III. Machine Learning Production

The complexity of building and integrating machine learning applications is challenging to software engineering teams. The inherent differences between software engineering and machine learning do not allow software engineering methodologies to be applied uniformly. Whereas software engineering is dependent on software design, development and testing, machine learning model development is based on data and model design, training, evaluation, deployment, and monitoring. Machine learning systems are non-deterministic and are therefore difficult to build using sequential development methods [17]. Amerce [18], also recognizes the need for a diversity of skills to take a machine learning model to production as one of the major hurdles in machine learning application development. Data, hidden technical debt and the need for iterative experimentation are the main technical challenges of machine learning development.

TABLE I. MACHINE LEARNING APPLICATION ARCHITECTURE

User Application	Machine Learning System	Machine Learning Support System
Front-end and back-end code	Models and model code	Configuration management
User management and security	Custom training code	Machine resource management
Device integration	Evaluation and metric code	Data collection
Version and update control	Data and pipelines	Analysis and monitoring tools
Platform and OS compatibility	Experiment management system	Data verification

A. *Data*

Data collection and preparation is by far the greatest challenge in machine learning model development [18] [2]. The quality, structure and relevance of data is key to the performance of a machine learning model. Data can come from different sources, different levels of quality and in different formats [19]. This data needs to be collected, cleaned, augmented, and labelled for machine learning model development. It can also take time for engineers to understand the data available to them. This comprehension is often key to successful data engineering and model debugging. Data consistency is also a key element of machine learning and is challenging for systems which require large amounts of manually labelled data. The influence of data collection on model development can be decreased by operating data collection and labelling as a parallel process to model development and training. The standard production strategy is to begin with enough data to converge machine learning training. This allows the teams to build the modelling pipeline and evaluation infrastructure without waiting for data collection to complete. Machine learning processes have largely developed based on workflows developed for data science and data mining, and therefore reflects the data dependency of machine learning model development.

B. *Hidden Technical Debt*

Hidden technical debt refers to the machine learning support system (Table 1) needed to test, monitor, update and

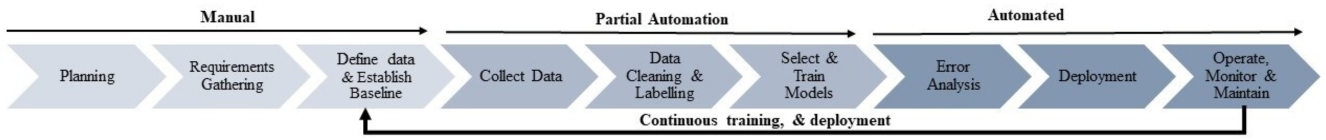

Fig. 3. Production machine learning architecture and workflows

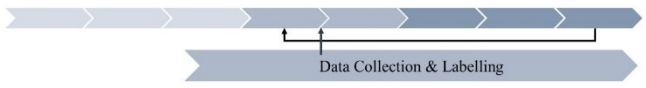

Fig. 4. Parallel MLOps and data preparation workflows

maintain a production machine learning system [20] [21]. Some of the modules in this system are related to 'glue code' which integrate the different machine learning libraries and pipelines. The configuration of machine learning systems also requires complex software. This is referred to in [21] as '*configuration debt*'. The lines of code for configuration management can easily exceed the code for model generation and training. Troubleshooting machine learning systems can be difficult due to dependencies between different models [22]. Maintaining strict boundaries between models during development is difficult as the models can interact in unexpected ways [18]. The strong abstraction boundaries preserved in software development are therefore difficult to maintain with machine learning model development. This is known as 'component entanglement' and can lead to non-monotonic error propagation [23] [20]. Hidden feedback loops due to dependencies between data or between models are difficult to detect and debug. Software modules to continuously monitor the performance of a model in deployment is necessary to detect model degradation and data changes. The modules which facilitate all the above support and monitoring systems constitute the 'technical debt' of a machine learning application.

C. *Experimentation*

Experimentation is a key aspect of machine learning model development. As machine learning models are tuned using several parameters, trial and error is needed to determine the optimal permutation and features. Engineers spend several development rounds determining data dependencies and the optimal set of features to maximize model performance. Model refinement and tuning also requires further rounds of experimentation through data and feature engineering.

D. *Data-Centric Machine Learning*

The attention of the machine learning community has been drawn to data-centric machine learning by Andrew Ng [24] as a solution to reducing the risk, uncertainly and complexity of machine learning projects. Experimentation on both model parameters and data at the same time is difficult. Data-centric machine learning negates this difficulty by freezing the model parameters early in the process and experimenting through data. Data-centric methods also facilitate easier comparison of the performance of the system to user expectations [25]. Automation of model training, evaluation, integration, deployment, and monitoring is key to facilitating data-centric experimentation.

E. *Machine Learning Operations (MLOps)*

MLOps are a collection of methodologies to deploy and maintain machine learning models in productions [26]. The term MLOps is used when DevOps processes are used for machine learning [27]. Like DevOps, MLOps facilitates the continuous training, evaluation, deployment, and monitoring of machine learning models. Fig. 3 shows primary phases of the MLOps process. The first three phases are usually done manually as domain knowledge and system analysis is needed. Data collection, data cleaning and model design is primarily done manually. However, new techniques are increasingly becoming available for automated feature selection, labelling and model generation. A common practice is to run data collection and labelling as a parallel process (Fig. 4). MLOps workflows typically automate the final four phases - model training, evaluation, deployment, and monitoring [28]. Development of the MLOps workflow can be accelerated by starting model development as soon as there is enough data to allow a model to converge. This allows the entire MLOps workflow to be built and tested before rest of the data is available. Many researchers have highlighted the importance of data quality and label consistency to the success of MLOps outcomes beyond this phase [27] [25] [24]. An MLOps framework is an essential feature for data-centric machine learning. MLOps lets engineers focus on improving performance through data engineering and experimentation. An MLOps framework allows engineers to define experiments through the data features and to rapidly analyze results, thereby allowing multiple experiments can be managed in parallel by a small team. A MLOps system is developed using software development principles.

F. *Machine Learning Applications*

The performance of a machine learning application in production is tightly coupled to the user application and the deployment environment. Most user applications are complex software on their own with a front-end, back end, database, input devices, user management and security modules. These modules are typically developed using agile continuous delivery methodologies. The success of any machine learning project is dependent on how soon the application can be integrated to facilitate training and development under real-world conditions. The integration of the MLOps process with DevOps used for the software modules is therefore a very important, but difficult part of any machine learning application development.

IV. AGILE SDLC FOR MACHINE LEARNING PRODUCTION

We present a framework for an agile SDLC for machine learning application development. The framework was founded on the following design goals:
(1) Agile

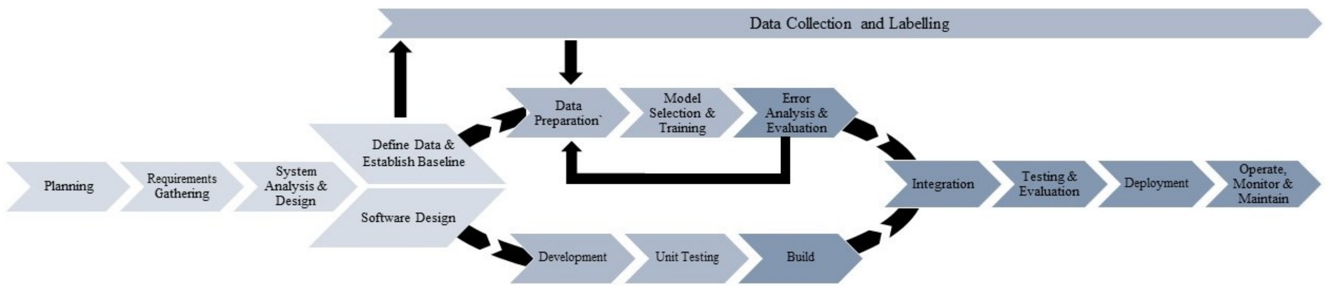

Fig. 5. Agile SDLC for Machine Learning Application Development

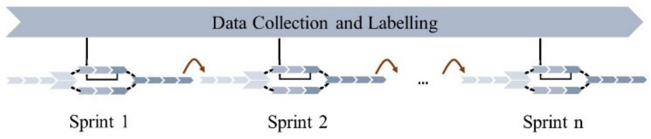

Fig. 6. Agile continuous development, integration, and deployment

Support for agile software development practices of rapid continuous development and evaluation, adaptability, flexibility, collaboration, and cross-functional teams.

(2) Integration

Model development, evaluation and fine-tuning to be conducted on the fully integrated deployment environment.

(3) Experimentation

Simplify experimentation and discovery within an agile development workflow.

(4) Data-centric

Facilitate data-centric experimentation through automation of modelling, evaluation, and deployment.

(5) Collaborative

Encourage greater team participation and collaboration through the simplification of the experimentation process.

Fig. 5 shows the outline of the proposed machine learning application software development life cycle (MLASDLC). The primary feature of this framework lies in how the development of the machine learning model, the machine learning support system, and the user application are combined. We propose that data collection and labelling be conducted as a separate parallel process to allow model development to progress uninterrupted. This model ties in well with the common practice of using separate or out-sourced teams for data labelling. Fig. 6 shows how the MLASDLC can be used through iterative agile sprints alongside a parallel data collection and labelling pipeline. The MLASDLC also advocates clearly defined feedback loops to the data team to facilitate rapid feature engineering and experimentation. The key benefit for commercial teams is the ability to rapidly train, deploy and monitor machine learning applications while they are being developed.

A. *Planning and design*

The first phases of the MLASDLC are planning and requirement gathering. It is important to start with the specification of the end user application, and the objectives to achieve minimum viability. This phase typically includes a feasibility study where the viability of the application, the data sources, access to data and data quality are assessed. The system needs to be defined in terms of the use-cases and expected performance. The system analysis and design phase follows the requirement gathering phase. The system architecture, the software modules and interfaces will be specified during this phase.

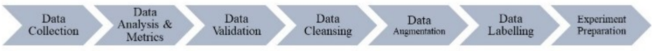

Fig. 7. sub-phases of the model selection and training phase

The system architecture is expanded into a detailed design of each module in the software design phase. The internal software architecture of each module was designed during this phase. This phase forks into two parallel processes. The software design sub-process will focus on the design of user application, the machine learning support framework and the modules needed by for the MLOps. The data definition and baselining sub-phase was used to design the inputs and outputs for the machine learning modules. The input specification for the machine learning modules was initially estimated based on knowledge of the domain and data. These definitions will be changed regularly during data engineering and experimentation. Model output specifications are less prone to change, and the early determination of what each model would output allows for the rest of the MLOps pipeline to be designed. This allows the support system and MLOps pipelines to be built before model development. The development workflow is then then branched into two parallel workflows which operate independently until integration at a later phase.

B. *Software Development*

The software development branch of the MLASDLC follows the standard agile development process of develop, unit test and build. The processes are iterated within a sprint depending on the length of the sprint and the level of DevOps automation. These phases will be used to build and test the user application, and configuration and delivery system, MLOps pipelines and the DevOps pipelines. MLOps are used to integrate the machine learning systems with DevOps to enable CI/CD based experimentation. The model serving and monitoring framework would also be engineered and tested during this phase.

C. *Machine Learning Model Development*

The machine learning branch of the MLASDLC follows the standard methods used for machine learning production – data preparation, model selection and training, and evaluation. The data preparation phase is broken into several sub-phases (Fig. 7) – data collection, data analysis and metrics, data validation, data cleansing, data augmentation, data labelling and experiment preparation. The agile team will focus on on understanding, validating, cleaning, and enriching the data during the first few sprints. Their attention will shift to experimentation and iterative feature engineering

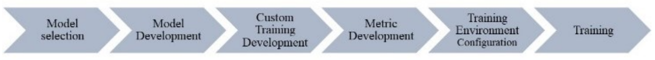
Model selection
Model Development
Custom Training Development
Metric Development
Training Environment Configuration
Training

Fig. 8. Sub-phases of the model selection and training phase

once they develop their data pipelines. Data metric generation, validation, cleansing, and augmentation are often automated, thereby allows for faster experimentation. The data validation modules created during this phase can be used for monitoring the performance of the model and data during serving.

The model selection and training phase is broken into several sub-phases (Fig. 8) – model selection, model development, custom training development, metric development, training environment configuration and training Each of these sub-phases are typically automated through MLOps. As model selection can often be a tedious and time-consuming process, engineers typically begin development with a simple model and deploy it onto the environment once they have enough data for model to converge. This is done so that the MLOps pipeline can be built and tested in preparation for experimentation. The model is then incrementally improved and fine-tuned over subsequent sprints as more data becomes available. Model refinement and tuning is done through feature engineering and model enhancement. The integration with the final deployment system as early in the development process as possible is a key step to allow experiments to be conducted using real-world conditions. User-friendly, no-code interfaces for the management and configuration of these pipelines enables faster experimentation and allows other members of the team to assist with experimentation. UI/UX design principles can significantly assist with reducing the turnaround time between machine learning experiments.

D. *Integration, deployment, and monitoring*

The application software and machine learning modules are integrated into a single system when the two sub-processes converge. Automation of this phase, and the subsequent testing and deployment phases is key to enabling the experimentation environment needed for data-centric development. Model debugging and evaluation, including sensitivity analysis and adversarial attack need to be integrated into the testing and evaluation phase. Systems to automate analytics to system performance would also be integrated.

E. *Workflow*

The MLASDLC is designed to facilitate data-centric model development, experimentation, and agile collaboration. Fig. 9 shows how the focus of the agile team would shift over the sprint cycles. Whereas the first few sprints of the development process would focus on software development of the application, the DevOps systems, the MLOps pipelines, and the machine learning support system, later stages would focus more and on data-centric experimentation. This allows these projects to be conducted using an agile process whilst maintaining the flexibility to tune the machine learning system using iterative experimentation. The MLASDLC is a framework to operationalize machine learning application development within standard software engineering processes.

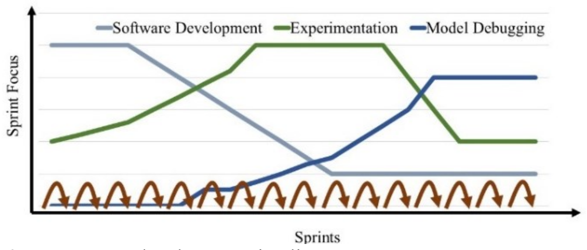

Fig. 9. MLASDLC development timeline

V. Evaluation

This framework is the result of combining our knowledge and experience of software development with the practices we developed for machine learning application production. We have tailored the breakdown of the phases to reflect the processes developed over more than a decade by the authors at SimCentric Technologies, Tengri UAV, and other organizations. The central principle of focusing on the final application from the inception of the project allowed us to develop the applications faster and at lower risk. We were able to evaluate the system in terms of the business requirements at regular intervals, and thereby fine tune development to ensure that the final objectives were achieved.

It is difficult to quantify the time and cost saved due to the technical differences between projects and the small sample size. However, we will quantify our evaluation by categorizing the machine learning applications by development process we used and the outcomes of the project. The projects we conducted can be classified into 2 categories based on the separation between machine learning model development and application development - 1. initial focus on machine learning model development and shift the application development and integration once model development is complete, and 2. the final application is designed and developed alongside machine learning model development. All category 1 projects used sample data or real-world data extracts for model development. Some of these projects not successful and were abandoned. However, those which attained acceptable performance metrics were moved onto application development and integration. In each of these projects, the performance of the model using real-world data was significantly below what was achieved while using extracted training data. In each case, the project teams had to conduct further feature engineering, hyperparameter tuning and model training before comparable performance was obtained. From a sample size of 3 comparable completed projects each from category 1 and 2, we estimate that category 1 projects took at least 30% to 40% longer to deploy. This extra time was mainly consumed by data engineering, feature engineering, integration issues and dependencies.

It is significant to note that one project which used MLASDLC was not successful and abandoned. Herein lies the drawback of using the MLASDLC for application development – the risk of developing the support application before the machine learning system is conceptualized, designed, validated, and tested. It is therefore prudent to validate the use-case, the quantity and quality of available data, and the feasibility of the project before moving onto the primary phases of the MLASDLC.

VI. Discussion

The MLASDLC is an operationalization of the processes we have established for developing production grade machine learning applications. The key feature of this methodology is the design of both the machine learning system and the deployment application at the start of the project. This is a shift from many machine learning projects where the focus is at first on the data and the model. MLASDLC advocates for the deployment application to be rapidly built early in the project, and thereby allows the machine learning systems to be designed, experimented upon and tested using the real-world environment. This methodology significantly reduces the risk of machine learning development by facilitating the use of real-world data for development and testing. This is achieved by extending the MLOps framework into the larger DevOps framework, thereby allows for CI/CD to be used during experimentation. The framework encapsulates these operational processes within standard software engineering principles, and thereby allows projects to be managed and tracked.

Open-ended machine learning projects which are difficult to track have been hampering the smooth operation of software companies. The MLASDLC is intended to provide a blueprint for these companies to manage machine learning projects in the same manner they manage their other software development projects. MLASDLC advocates wide-spread automation, thereby allowing for rapid iteration and testing using data-centric model development, which is fast gaining popularity within the machine learning community. The main objective of this framework is to align machine learning system development with the primary objectives of the application, business needs, customer needs and other KPIs.

Acknowledgement

The authors would like to thank all those who have supported the work which lead to this paper.

References

[1] S. Vaidya, P. Ambad and S. Bhosle, "Industry 4.0 – A Glimpse" *Procedia Manufacturing,* vol. 20, pp. 233-238, 2018.

[2] N. Soni, E. K. Sharma, N. Singh and A. Kapoor, "Artificial Intelligence in Business: From Research and Innovation to Market Deployment," *Procedia Computer Science,* vol. 167, pp. 2200-2210, 2020.

[3] VB Staff, "Why do 87% of data science projects never make it into production?," 19 July 2019. [Online]. Available: https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/.

[4] Algorithmia, "2021 enterprise trends in machine learning," Algorithmia, 2021.

[5] E. d. S. Nascimento, I. Ahmed, E. Oliveira, M. P. Palheta, I. Steinmacher and T. Conte, "Understanding Development Process of Machine Learning Systems: Challenges and Solutions," *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM),* vol. 1, pp. 1-6, 2019.

[6] B. Qian, J. Su, Z. Wen, D. N. Jha, Y. Li, Y. Guan, D. Puthal, P. James, R. Yang, A. Zomaya, O. Rana, L. Wang and R. Ranjan, "Orchestrating the Development Lifecycle of Machine Learning-based IoT Applications: A Taxonomy and Survey," *ACM Computing Surveys,* vol. 53, no. 4, p. 82, 2020.

[7] M. A. Rather and V. Bhatnagar, "A Comparative Study of Software Development Life Cycle Models," *International Journal of Application or Innovation in Engineering & Management (IJAIEM),* vol. 4, no. 10, 2015.

[8] P. Matkovic and P. Tumbas, "A Comparative Overview of the Evolution of Software Development Models," *Journal of Industrial Engineering and Management,* vol. 1, no. 4, pp. 163-172, 2010.

[9] W. Royce, "Managing the Development of Large Software Systems," *Technical Papers of Western Electronic Show and Convention,* pp. 1-9, 1970.

[10] H. Takeuchi and I. Nonaka, "The New New Product Development Game," *Harvard Business Review,* vol. 64, pp. 137-146, 1986.

[11] I. Sommerville, Software Engineering, Addison-Wesley Publishing Company, 2010.

[12] B. Boehm, "A spiral model of software development and enhancement," *Computer,* vol. 21, no. 5, pp. 61-72, 1988.

[13] G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language user guide, Addison Wesley Longman Publishing Co., 1999.

[14] I. T. Christou, S. T. Ponis and E. Palaiologou, "Using the Agile Unified Process in Banking," *IEEE Software,* vol. 27, no. 3, pp. 72-79, 2010.

[15] F. Erich, C. Amrit and M. Daneva, "A Qualitative Study of DevOps Usage in Practice," *Journal of Software: Evolution and Process,* vol. 0, no. 6, 2017.

[16] R. Jabbari, N. B. Ali, B. Tanveer and K. Peterson, "What is DevOps?: A Systematic Mapping Study on Definitions and Practices," in *The Scientific Workshop XP2016*, 2016.

[17] G. Giray, "A Software Engineering Perspective on Engineering Machine Learning Systems: State of the Art and Challenges," *CoRR,* vol. abs/2012.07919, 2020.

[18] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar and N. Nagappan, "Software Engineering for Machine Learning: A Case Study," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019.

[19] R. Ashmore, R. Calinescu and C. Paterson, "Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges," *ACM Comput. Surv.,* vol. 54, no. 5, p. Article 111, 2021.

[20] M. Godfried, "Until Technial Debt In Machine Learning Tear Us Apart," 29 Sep 2020. [Online]. Available: https://towardsdatascience.com/until-technical-debt-in-machine-learning-tear-us-apart-2de7d54dd0ea. [Accessed 31 Aug 2021].

[21] D. Scully, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young and D. Dennison, "Hidden Technical Debt in Machine Learning System," *Advances in Neural Information Processing Systems,* pp. 2494-2502, 2015.

[22] M. Lê and A. Fokkens, "Tackling Error Propagation through Reinforcement Learning:A Case of Greedy Dependency Parsing," in Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Valencia, Spain, 2017.

[23] B. Nushi, E. Kamar, E. Horvitz and D. Kossmann, "On Human Intellect and Machine Failures: Troubleshooting Integrative Machine Learning Systems," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[24] A. Ng, "MLOps: From Model-centric to Data-centric AI," 06 2021. [Online]. Available: https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf. [Accessed 31 Aug 2021].

[25] A. I. Anik and A. Bunt, "Data-Centric Explanations: Explaining Training Data of Machine Learning Systems to Promote Transparency," in *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021.

[26] S. Alla and S. K. Adari, "What Is MLOps?," in *Beginning MLOps with MLFlow*, Berkeley, CA, Apress, 2020, pp. 79-124.

[27] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlaš, W. Wu and C. Zhang, "A Data Quality-Driven View of MLOps," *CoRR,* vol. abs/2102.07750, 2021.

[28] O. Spjuth, J. Frid and A. Hellander, "The machine learning life cycle and the cloud: implications for drug discovery," *Expert Opin Drug Discov.,* pp. 1-9, 2021.