International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

Designing Modular AI Architectures for Enterprise Scale: From Monolithic Models to Composable AI Systems

Hemant Soni, Priyadharsini Ramamurthy

Senior Solution Architect (Telecom and AI Platforms), Capgemini America Inc., Atlanta, Georgia, USA

Department of Computer Science, Oklahoma State University, Stillwater, OK, USA

Hemantsoni2015@gmail.com

pramamu@ostatemail.okstate.edu

ABSTRACT: Enterprises adopting machine learning have moved beyond isolated proof of concept models towards production systems that must integrate with established platforms such as customer relationship management (CRM), enterprise resource planning (ERP) and telecom operations and business support systems (OSS/BSS). This paper argues that the dominant obstacle to value realisation is architectural rather than algorithmic. We review the limitations of the monolithic model pattern, in which a single large model is trained, deployed and maintained as one indivisible unit, and we set out a reference architecture for modular, composable AI systems. The proposed architecture organises capability into independently deployable model services, a shared data foundation with a feature store and lineage tracking, and an operations control plane that automates integration, testing, monitoring and governance. We map established software engineering practice, namely microservices, continuous delivery and design patterns for machine learning, onto the constraints of enterprise AI, and we discuss integration with CRM, ERP and telecom stacks through an API and event driven serving layer. A qualitative evaluation compares the monolithic and modular approaches across maintainability, scalability, governance and time to change. The analysis indicates that modularisation reduces coupling and shortens the path from experimentation to a governed, observable production system, at the cost of an increased operational surface area that the control plane is designed to manage.

KEYWORDS: Modular AI architecture; composable AI systems; MLOps; microservices; feature store; enterprise integration; OSS/BSS; machine learning deployment.

I. INTRODUCTION

The central difficulty in enterprise artificial intelligence is no longer whether a model can be trained to acceptable accuracy. Public foundation models and mature open source frameworks have made capable models widely available. The difficulty is that a large share of machine learning initiatives never reach dependable production, and of those that do, many are expensive to operate and difficult to change. Studies of real deployments repeatedly attribute this gap to engineering and organisational factors rather than to modelling accuracy, including accumulated technical debt, brittle data dependencies, and the absence of disciplined operations.

A recurring root cause is the monolithic model pattern. In this pattern a single, large model is trained, deployed and maintained as one indivisible artefact, tightly coupled to a specific dataset, a specific serving stack, and frequently to a single team. The pattern is attractive during experimentation because it minimises moving parts. At enterprise scale it becomes a liability. Any change, whether to a feature, a data source, or a downstream consumer, forces revalidation of the whole, retraining cycles are slow, and the model cannot be reused across the customer relationship management (CRM), enterprise resource planning (ERP) and telecom operations and business support (OSS/BSS) systems that need it.

This paper takes the position that the unit of design for enterprise AI should be the system, not the model. We review the limitations of the monolithic pattern and propose a reference architecture for modular, composable AI systems in which capability is delivered as independently deployable services over a shared data foundation, coordinated by an operations control plane. The contributions are: (i) a synthesis of established software engineering practice, namely microservices, continuous delivery and machine learning operations, applied to the constraints of enterprise AI; (ii) a

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

layered reference architecture that spans data sources to enterprise consumption; and (iii) a qualitative evaluation that contrasts the monolithic and modular approaches and offers guidance on when each is appropriate.

II. LITERATURE SURVEY

The study of production machine learning as an engineering discipline predates the current interest in large models. Sculley and colleagues described how machine learning systems incur hidden technical debt through entanglement, undeclared consumers and unstable data dependencies [1]. Breck and colleagues proposed a rubric for production readiness that treats data, model and infrastructure testing as first class concerns [2]. Amershi and colleagues documented the software engineering practices of teams building machine learning products and the friction introduced by data management and model evolution [3].

The operations perspective has since been consolidated under the term machine learning operations. Kreuzberger and colleagues offer an aggregated definition of MLOps, identifying the principles, components and roles that recur across tools and case studies [4]. John and colleagues propose a maturity model that frames adoption as a progression rather than a single state [5]. Earlier, Baylor and colleagues described a production scale platform that integrates data validation, training and serving into a coherent pipeline [6]. Lakshmanan and colleagues catalogue reusable design patterns for the data and model lifecycle [7].

On the architectural side, the microservices style provides the most directly relevant precedent. Newman describes the decomposition of systems into small, independently deployable services aligned to business capability [8], and Dragoni and colleagues survey the principles and open problems of the style [9]. In parallel, the rapid growth in model scale and the emergence of reusable foundation models have made composition more attractive: the transformer architecture [10], few shot capable language models [11], bidirectional pretraining [12] and the broader notion of foundation models [13] all point towards systems that combine general components rather than training a bespoke monolith for every task. Retrieval augmented generation, which couples a parametric model with an external knowledge index [14], is an early and influential example of treating a model as one component within a larger system.

What the literature does not yet provide is consolidated guidance on assembling these practices into a single reference architecture for the enterprise, one that spans the data foundation, the operations control plane and integration with CRM, ERP and telecom stacks. This paper addresses that gap.

III. PROPOSED METHODOLOGY AND DISCUSSION

A. Design Principles

The proposed architecture rests on five principles. Separation of concerns assigns data preparation, model logic, serving and governance to distinct components with explicit contracts. Independent deployability allows each model service to be released, scaled and rolled back without coordinating a release of the whole. Contract first interfaces decouple producers from consumers, so that an enterprise application depends on a stable interface rather than on a model internal. Observability by default treats logging, metric capture and drift detection as part of every component rather than an afterthought. Governance by default embeds access control, lineage and approval into the pipeline so that compliance is a property of the system, not a manual checkpoint.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

B. Reference Architecture

Fig.1. Layered reference architecture for modular, composable enterprise AI

Figure 1 presents the architecture as a stack of layers. At the base, heterogeneous data sources include network telemetry, billing records, customer interactions and signals from connected devices. The data foundation standardises ingestion, exposes curated features through a feature store, and records lineage, quality and versioning so that any prediction can be traced to the data that produced it. Above the foundation, the operations control plane provides continuous integration and delivery, automated testing, monitoring and governance for every model service.

Model capability is expressed as a set of composable pipelines registered in a model registry, and exposed as independently deployable model services. The serving and integration layer presents these services through an application programming interface (API) gateway for synchronous use and an event bus for asynchronous use, supporting both batch and online inference. At the top, enterprise consumption systems, namely CRM, ERP and telecom OSS/BSS together with customer channels, consume predictions through stable interfaces without knowledge of the underlying model implementation.

C. From Monolith to Modules

Decomposition follows business capability rather than algorithm. A monolithic churn model that ingests raw billing, network and interaction data and emits a single score is refactored into a feature pipeline that publishes reusable features, a scoring service that consumes those features, and an explanation service that can be evolved independently. The model service becomes the unit of deployment, with its own version, interface and monitoring. Not every workload warrants decomposition; a small, stable model with a single consumer may reasonably remain monolithic. The architecture therefore treats modularisation as a decision driven by rate of change, reuse and the number of consumers, rather than as an absolute rule.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

D. The Data Foundation

A shared feature store is the component that most directly enables reuse and consistency. By computing features once and serving them to both training and inference, it removes a common source of training and serving skew and allows several model services to share curated signals such as customer tenure, recent network experience or billing volatility. Lineage and versioning make experiments reproducible and give governance a factual basis.

E. The Operations Control Plane

The control plane automates the path from a registered pipeline to a governed, observable service. Continuous integration validates code, data schemas and model behaviour; continuous delivery promotes artefacts through environments with approvals; monitoring tracks input distributions, prediction quality and operational health, raising alerts on drift; and governance enforces access, captures audit trails and links each deployment to its lineage. The control plane is the mechanism by which the additional operational surface area of a modular system remains manageable.

F. Enterprise Integration

Integration uses two complementary patterns. API first integration suits synchronous decisions, such as a CRM agent requesting a next best action while a customer is on the line. Event driven integration suits asynchronous flows, such as reacting to a network event published by an OSS platform by recomputing a risk score and emitting a downstream event. In telecom settings the same model services can support churn prediction, next best action and assurance use cases across CRM and OSS/BSS, precisely because they are exposed through stable interfaces rather than embedded in any single application.

IV. RESULTS AND DISCUSSION

Because the contribution is architectural, the evaluation is qualitative and comparative. Table I contrasts the monolithic and modular approaches across the dimensions that determine cost of ownership at enterprise scale.

Table I. Qualitative comparison of monolithic and modular approaches

Dimension	Monolithic model	Modular system
Maintainability	Low; change forces full revalidation	Higher; change is localised to a service
Reuse	Limited; bound to one consumer	High; services shared across systems
Scalability	Coarse; scale the whole unit	Fine; scale services independently
Governance	Manual and late	Embedded and continuous
Time to change	Slow; coupled releases	Fast; independent releases
Operational cost	Low surface, high change cost	Higher surface, lower change cost

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

Table II offers guidance on selecting a pattern for a given workload, recognising that modularisation is a means rather than an end.

Table II. Pattern selection guidance

Factor	Favours monolith	Favours modular
Rate of change	Low	High
Number of consumers	One	Many
Reuse potential	Low	High
Governance need	Light	Strict

To make the analysis concrete, consider a telecom operator that wants to reduce churn while improving customer experience. In the monolithic pattern, a single model embedded in the CRM produces a churn score; a parallel effort in the assurance domain duplicates much of the same feature logic for a network experience score, and neither can be reused by the other. In the modular pattern, a feature pipeline publishes shared signals, a churn service and an experience service consume them, and a next best action service composes both. CRM consumes recommendations synchronously through the API gateway, while OSS events trigger asynchronous recomputation through the event bus. The marginal cost of a new use case falls because it composes existing services rather than rebuilding them.

The principal cost of the modular approach is operational surface area. More services mean more interfaces, more deployments and more components to monitor. This cost is real, and it is the reason the operations control plane is treated as a first class layer rather than as tooling. Where workloads are small, stable and singular, the simplicity of a monolith remains defensible. The architecture is therefore best understood as a target for capabilities that are reused, that change often, or that carry significant governance obligations, which describes most enterprise AI of lasting value.

V. CONCLUSION

Enterprises adopting machine learning succeed or fail on architecture more often than on accuracy. The monolithic model pattern that serves experimentation well becomes a barrier to reuse, change and governance at scale. This paper has proposed a layered reference architecture for modular, composable AI systems that organises capability into independently deployable services over a shared data foundation, coordinated by an operations control plane and integrated with CRM, ERP and telecom OSS/BSS through API first and event driven patterns. The qualitative analysis indicates that modularisation shortens the path from experimentation to a governed, observable production system and lowers the cost of change, at the expense of operational surface area that the control plane is designed to absorb. Future work includes extending the control plane towards greater automation and incorporating the emerging class of large generative models as composable services within the same architecture.

REFERENCES

[1] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. F. Crespo, and D. Dennison, "Hidden Technical Debt in Machine Learning Systems," in Advances in Neural Information Processing Systems (NeurIPS), vol. 28, 2015, pp. 2503-2511.

[2] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," in Proc. IEEE Int. Conf. on Big Data, 2017, pp. 1123-1132.

[3] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software Engineering for Machine Learning: A Case Study," in Proc. IEEE/ACM 41st Int. Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019, pp. 291-300.

[4] D. Kreuzberger, N. Kuhl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," arXiv preprint arXiv:2205.02302, 2022.

[5] M. M. John, H. H. Olsson, and J. Bosch, "Towards MLOps: A Framework and Maturity Model," in Proc. 47th Euromicro Conf. on Software Engineering and Advanced Applications (SEAA), 2021, pp. 1-8.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 6, Issue 1, January –February 2023||

DOI:10.15662/IJRPETM.2023.0601013

[6] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, et al., "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform," in Proc. 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD), 2017, pp. 1387-1395.

[7] V. Lakshmanan, S. Robinson, and M. Munn, Machine Learning Design Patterns. Sebastopol, CA, USA: O'Reilly Media, 2020.

[8] S. Newman, Building Microservices: Designing Fine-Grained Systems. Sebastopol, CA, USA: O'Reilly Media, 2015.

[9] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: Yesterday, Today, and Tomorrow," in Present and Ulterior Software Engineering, Springer, 2017, pp. 195-216.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, pp. 5998-6008.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 1877-1901.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.

[13] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, et al., "On the Opportunities and Risks of Foundation Models," arXiv preprint arXiv:2108.07258, 2021.

[14] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459-9474.

[15] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in Deploying Machine Learning: A Survey of Case Studies," arXiv preprint arXiv:2011.09926, 2020.