

A Tutorial on Software Engineering for FMware

Filipe Roseiro Cogo

Centre for Software Excellence, Huawei Canada
Kingston, Ontario, Canada
filipe.roseiro.cogo1@huawei.com

Dayi Lin

Centre for Software Excellence, Huawei Canada
Markham, Ontario, Canada
dayi.lin@huawei.com

Gopi Krishnan Rajbahadur

Centre for Software Excellence, Huawei Canada
Kingston, Ontario, Canada
gopi.krishnan.rajbahadur1@huawei.com

Ahmed E. Hassan

School of Computing, Queen's University
Kingston, Ontario, Canada
ahmed@cs.queensu.ca

ABSTRACT

Foundation Models (FMs) like GPT-4 have given rise to FMware, FM-powered applications representing a new generation of software that is developed with new roles, assets, and paradigms. FMware has been widely adopted in both software engineering (SE) research (e.g., test generation) and industrial products (e.g., GitHub copilot), despite the numerous challenges introduced by the stochastic nature of FMs. In our tutorial, we will present the latest research and industrial practices in engineering FMware, along with a hands-on session to acquaint attendees with core tools and techniques to build FMware. Our tutorial's perspective is firmly rooted in SE rather than artificial intelligence (AI), ensuring that participants are spared from delving into mathematical and AI-related intricacies unless they are crucial for introducing SE challenges and opportunities.

CCS CONCEPTS

• **Software and its engineering** → **Software creation and management.**

KEYWORDS

Foundation Model, FMware, Software engineering for FMware

ACM Reference Format:

Filipe Roseiro Cogo, Gopi Krishnan Rajbahadur, Dayi Lin, and Ahmed E. Hassan. 2024. A Tutorial on Software Engineering for FMware. In *Proceedings of the 32nd ACM Symposium on the Foundations of Software Engineering (FSE '24)*, November 15–19, 2024, Porto de Galinhas, Brazil. ACM, New York, NY, USA, 3 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

The advent of Foundation Models (FMs), such as GPT-4, marked by their billions of parameters and self-supervised training on extensive datasets, has given rise to a new class of software called FMware, such as Bing Chat and GitHub Copilot. FMs exhibit novel

functionalities not found in smaller models, such as advanced reasoning over multiple steps and adeptly following instructions. This empowers the development of intelligent applications across various domains, capable of being programmed and operated through natural language prompts.

In contrast to traditional “software developers”, the aforementioned attributes pave the way for a novel category of “software makers”, which possess limited computer science training but extensive domain expertise. Software makers have a central participation in the software creation process, consequently pushing the boundaries of Software Engineering (SE) to involve end-users and other stakeholders more actively. However, recent research highlights various challenges in engineering FMware, such as instability of natural language-driven prompts [6], difficult-to-control hallucinations, and the substantial costs and low productivity associated with FMware development and operations. Therefore, in the current era of FMware, it is crucial to adapt existing and emerging SE practices to construct FMware applications that are trustworthy, reliable, resilient, and adaptable.

Tutorial aims and objectives. This tutorial explores the diverse facets of designing and implementing FMware, offering attendees the most recent research findings in this field and the accompanying SE challenges. The goal is to aid practitioners and SE researchers in incorporating FMs across different industry domains and SE subjects, employing established methodologies to guarantee the resulting FMware is trustworthy. Additionally, the tutorial will explore prospective research paths, guiding SE researchers in advancing FMware development practices. What sets our tutorial apart is its perspective, which is firmly rooted in SE rather than AI. This distinction ensures that participants are spared from delving into mathematical and AI-related intricacies unless these details are crucial for introducing the contextual SE challenges and opportunities.

Intended audience and required background. The tutorial is intended to accommodate a broad range of attendees with a diversified background in technology-related domains, being of special interest to SE students, researchers, and practitioners. Attendees with basic understanding of SE concepts should be able to follow the presented material and benefit from our tutorial.

Relevance to Software Engineering researcher and practitioners. FM are experiencing heightened enthusiasm in diverse SE communities, underscoring our tutorial's timeliness and broad

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

FSE '24, November 15–19, 2024, Porto de Galinhas, Brazil

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/18/06

<https://doi.org/XXXXXXX.XXXXXXX>

appeal to academics and industry practitioners. The escalating volume of publications in this domain reflects a growing trend, with attempts to incorporate FMs across various activities in the SE lifecycle. For researchers, the tutorial offers valuable insights to enhance their utilization of FMs in addressing research challenges, along with an understanding of the prevalent engineering challenges in FMware. This contributes to the evolution of engineering practices for this emerging software paradigm. Meanwhile, industry practitioners stand to gain practical knowledge in constructing FMware and insights into the obstacles associated with integrating FMs into their software systems.

2 TUTORIAL FORMAT AND INTENDED DURATION

The tutorial will be composed of **two sessions**: a *lecture session* (120 minutes), where we will present the theoretical fundamentals and key challenges around SE for FMware, and a *hands-on session* (60 minutes), where we demonstrate how to load open-source FMs, how to fine-tune them using parameter-efficient (PEFT) methods, and how to implement Retrieval-Augmented Generation (RAG). The two sessions will form a comprehensive **180-minute tutorial**.

The lecture session will cover two major phases in the lifecycle of FMware, namely *engineering FMs* and *engineering FMware*. Each phase will be covered in a 60-minute session. The lecture session will explain up-to-date literature in FMware using visual resources in the form of slides. Our hands-on session will use free Google Collaboratory notebooks and open-source libraries to implement a case study on the application of FMware to a specific SE task (automated code performance refactoring). The combination of lectures and hands-on sessions will give attendees a practical perspective into the necessary knowledge to build FMware, fostering new applications and research developments.

3 OUTLINE OF THE COVERED TOPICS

Below, we provide a detailed description of the topics covered in the tutorial.

3.1 Session 1: Engineering FMs

3.1.1 Introduction to FMs and the future of Software. The session commences with a broad introduction to the functioning of FMs and outlines the specific challenges that accompany the opportunities arising from the ascendancy of FMs. These challenges encompass *limitations in handling intricate tasks and multi-step reasoning*, the manifestation of *uncontrollable behavior in multi-round interactions*, and the *occurrence of hallucination*, i.e. generating factually incorrect responses with elevated confidence. These challenges have significant implications for the architecture, design, development, quality assurance, and operations of FMware.

3.1.2 Model Alignment. Model alignment involves employing a set of techniques to adjust the parameters of the FM, creating FMware that aligns with particular requirements and contexts. In addition to examining the primary Software Engineering (SE) challenges associated with model alignment, our tutorial will address automated techniques for generating alignment data.

Fine-tuning. Utilizing alignment data, fine-tuning can be applied to the FM to ensure alignment with the desired operational logic. Achieving effective alignment necessitates high-quality labeled data, often posing challenges due to the manual effort and cost involved in its acquisition. Furthermore, fine-tuning presents notable challenges to the hardware infrastructure, particularly with large FMs. The tutorial will address methods to carry out the fine-tuning of FMs efficiently.

RAG for knowledge-intensive applications. Alignment data also finds application in RAG, a technique commonly employed in knowledge-intensive applications [3]. In RAG systems, documents similar to a given prompt are initially retrieved and then utilized as grounding context. This aids the model in generating more relevant and accurate responses, potentially mitigating the hallucination problem. RAG-based systems have recently gained considerable popularity as a cost-effective approach among FMware practitioners [2]. This tutorial will present up-to-date techniques and tools for designing and implementing RAG systems.

3.1.3 Architectures of FMs and Performance Optimization. The choice of an appropriate FM for developing FMware applications is influenced by factors such as the model's parameter size and architecture. Larger FMs incur substantial costs for training, fine-tuning, and deployment, and they often pose challenges related to inference latency. Recent studies suggest that meticulously curated data can enable smaller FMs to outperform their larger counterparts, either learning independently or undergoing fine-tuning with larger models like GPT-4. This tutorial investigates the efficacy of utilizing smaller FMs as alternatives to larger ones without compromising performance. Additionally, it explores various strategies for performance optimization that FMware developers can employ, providing insights into cost-effective FMware development.

3.2 Session 2: Engineering FMware

3.2.1 Prompt Engineering Hacking. Prompts are natural language excerpts that serve as fundamental elements in instructing FMs to perform specific tasks and are considered essential building blocks of FMware. They offer a means to enrich FMs with domain knowledge, for instance, through the application of few-shot learning techniques [1], thereby enhancing the quality and reliability of the generated responses. Our tutorial will cover various prompt-related factors and components crucial for achieving optimal performance in downstream tasks. These factors include instruction clarity, text referencing, task breakdown, and reasoning [5]. Furthermore, we will illustrate best practices for structuring prompts, emphasizing the use of distinct components such as instructions (for guiding model generation), constraints (to set content boundaries), knowledge (providing background information), persona (imparting a specific personality to the model), and output formatting.

3.2.2 Static & AI-driven orchestration. Real-world applications of FMware tend to be more intricate than the single inference of an FM. They frequently entail engaging in multi-step interactions with diverse data sources and other software systems within the environment. Additionally, they might involve the inference of several distinct models to address various sub-tasks. The orchestration of

such workflows is typically carried out either statically by developers or autonomously by agents. Recent research has demonstrated success in employing multiple agents to tackle more challenging tasks that cannot be effectively addressed by a single agent or FM alone. However, autonomous agents and multi-agent systems introduce uncontrollable interactions and may pose risks when granted access to sensitive data and external systems. This tutorial will present an overview of the progress and challenges associated with developing intricate FMware, considering static orchestration, autonomous single-agent, and multi-agent architectures.

3.2.3 Quality assurance and trustworthiness. Quality assurance for FMware is a complex task due to the stochastic nature of FMs. Explaining and pinpointing issues in the FM's behavior is challenging, making it difficult to address problems effectively. Also, reproducing undesired inference results becomes a challenging task. Finally, the generative nature of applications and the adoption of new development paradigms, such as prompting, bring additional risks to the trustworthiness of FMware, encompassing concerns related to security, safety, and robustness.

4 KEY LEARNING OBJECTIVES FOR THE PARTICIPANTS

Upon completion of our tutorial, participants should:

- Understand the technical landscape and the latest SE advances and challenges on FMware.
- Apply the best practices to systematically and safely develop, deploy, and operate FMware.
- Acquire basic hands-on experience using different tools to build FMware, particularly to solve SE problems.

5 SPEAKER INFORMATION AND BIOGRAPHIES

Filipe R. Cogo is a Software Engineering Researcher at the Centre for Software Excellence, Huawei Canada, working on Foundation Models and Software Engineering. His research work has been published in flagship SE venues such as TSE, EMSE, TOSEM, and FSE. Filipe received his Ph.D. from the School of Computing at Queen's University, Canada and has more than ten years of experience teaching Computer Science. More info at <https://www.filpecogo.pro>.

Gopi Krishnan Rajbahadur is a Staff Researcher at Huawei's Centre for Software Excellence in Canada. He is currently working on SE for Large Language Models and the governance of AI datasets. His research interests encompass SE for AI, utilizing AI for SE, and the development of AI software that can be regulated. He is also an active contributor to the field of software engineering and AI standards, serving as the co-lead for the AI and datasets profile in the ISO SPDX standard. He also co-founded the open-source initiative OpenDataology and frequently presents at Open Source Summits. His research has been featured in prominent software engineering publications such as TSE, TOSEM, EMSE, and ICSE.

Dayi Lin is a Staff Researcher at the Centre for Software Excellence, Huawei Canada, working on the topics of SE for AI and FMware (i.e., FM-powered software). His research interests include SE for AI, AI for SE, game engineering, and mining software repositories. His work has been published at several top-tier software engineering

venues, such as TSE, TOSEM, EMSE, and ICSE. His work on SE and games has also attracted wide media coverage including Kotaku, PC Gamers, Gamasutra, and national newspapers. Dr. Lin is the program co-chair of The 1st International Conference on AIware (co-located with FSE'24).

Ahmed E. Hassan is a Full Professor and a Canada Research Chair at Queen's University. He is a fellow of IEEE, ACM, AAIA, and NSERC-Steacie. He is a laureate of the Mustafa Prize, recognized as equivalent in prestige to a Nobel, for founding the field of AI-augmented SE and its associated conference, MSR.

6 TUTORIAL HISTORY

A similar version of the lecture session (Section 2) of this tutorial was presented at the SE+FM School (<https://fmse.io/school/>) as part of the SE+FM Vision 2030 workshop from November 16 to 17 of 2023, in Mexico City, for an audience of approximately 80 attendees. In addition, a similar version of the lecture session is accepted in the ICSE'24 Technical Briefing Track [4], planned to be presented between April 14 and 20, 2024 (TBD). **The hands-on session (Section 2) is unique to this tutorial proposal**, as it has never been presented before.

7 AUDIO-VISUAL AND TECHNICAL REQUIREMENTS

Our tutorial will require internet access and a room with a projector. The tutorial will also require participants to bring their laptops to the room. Alternatively, access to a computer should be provided to participants who cannot bring their own laptop.

REFERENCES

- [1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165*
- [2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv:2312.10997*
- [3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv:2005.11401*
- [4] Dayi Lin, Filipe Cogo, Gopi Krishnan Rajbahadur, and Ahmed E. Hassan. 2024. Technical Brief on Software Engineering for FMware. <https://conf.researchr.org/details/icse-2024/icse-2024-technical-briefings/5/Technical-Brief-on-Software-Engineering-for-FMware>. [Online; accessed 09-February-2024].
- [5] OpenAI. 2023. Prompt Engineering Guide. <https://platform.openai.com/docs/guides/prompt-engineering/>. [Online; accessed 08-January-2024].
- [6] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. 2023. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1146–1156. <https://doi.org/10.1109/TVCG.2022.3209479>