From Cool Demos to Production-Ready FMware: Core Challenges and a Technology Roadmap

Gopi Krishnan Rajbahadur

arXiv

How to cite this session?

```
@misc{Rajbahadur2024AIwareTutorial,
author = {Gopi Krishnan Rajbahadur and Gustavo Oliva and Dayi Lin and Ahmed E. Hassan and Ben Rombaut},
title = {From Cool Demos to Production-Ready FMware: Core Challenges and a Technology Roadmap},
howpublished = {Tutorial presented at the AIware Leadership Bootcamp 2024},
month = {November},
year = {2024},
address = {Toronto, Canada},
note = {Part of the AIware Leadership Bootcamp series.},
url = {https://aiwarebootcamp.io/slides/2024_aiwarebootcamp_rajbahadur_fromcooldemostoproduction-readyfmwarecorechallengesandatechnologyroadmap.pdf }}
```

Check this paper for more information about this session

```
@article{rajbahadur2024cool,
  title={From Cool Demos to Production-Ready FMware: Core Challenges and a Technology Roadmap},
  author={Rajbahadur, Gopi Krishnan and Oliva, Gustavo A and Lin, Dayi and Hassan, Ahmed E},
  journal={arXiv preprint arXiv:2410.20791},
  year={2024}
}
```

Also, check this paper for more information about this session

```
@inproceedings{Hassan2024FSE,

author = {Hassan, Ahmed E. and Lin, Dayi and Rajbahadur, Gopi Krishnan and Gallaba, Keheliya and Cogo, Filipe
Roseiro and Chen, Boyuan and Zhang, Haoxiang and Thangarajah, Kishanthan and Oliva, Gustavo and Lin, Jiahuei
(Justina) and Abdullah, Wali Mohammad and Jiang, Zhen Ming (Jack)},

title = {Rethinking Software Engineering in the Era of Foundation Models: A Curated Catalogue of Challenges in the
Development of Trustworthy FMware},

year = {2024},

publisher = {Association for Computing Machinery},

address = {New York, NY, USA},

url = {https://doi.org/10.1145/3663529.3663849},

booktitle = {Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software
Engineering},

pages = {294-305},

numpages = {12},

series = {FSE 2024}

}
```

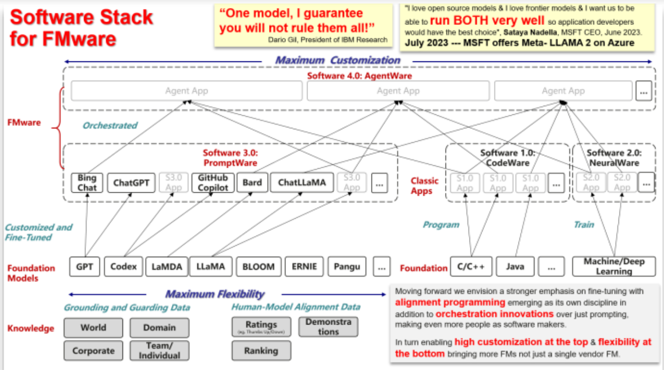
But my organization/BU is not doing AI/FM/LLMware!

Look again! **If anyone in your org is using an LLM for something they are developing FMware!!**

Goldman Sachs projects

- FMware to raise World GDP by 7%
- FMware market to be 22% of Software Market (150 of 685 Billion USD$)

AI/FMware is a systems problem not an FM/LLM/AI problem



January 2023

Hassan et al.

FMware is more than just a single model instead it is multi-generational/model/component

The Shift from Models to Compound AI Systems

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, Ali Ghodsi Feb 18, 2024

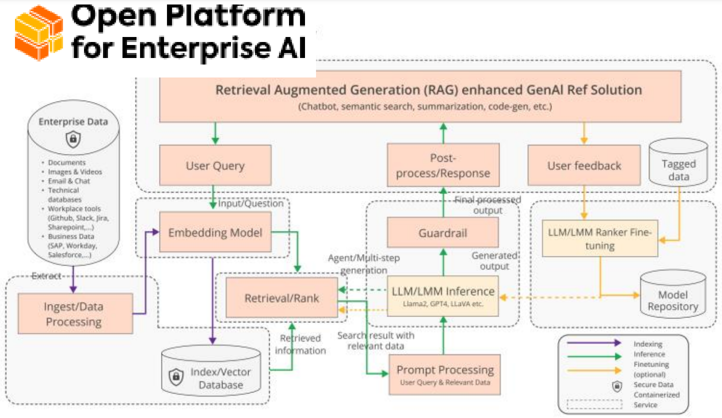
BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Berkley AI Research (BAIR) -- BAIR brings 50 faculty members and 300 researchers in computer vision, machine learning, natural language processing, planning, control, and robotics.

February 2024

Berkley AI Research (BAIR):

State-of-the-art AI results are increasingly obtained by ***compound systems*** with multiple components, not just monolithic models.



May 2024

OPEA alliance (50 Companies including Intel, RedHat, SAS)

working to define the architecture blueprint to enable enterprise ready AIware (focusing on RAGware for now)

Echoing many views across industry… "One Model will not rule them all", VP of IBM Research late 2023.

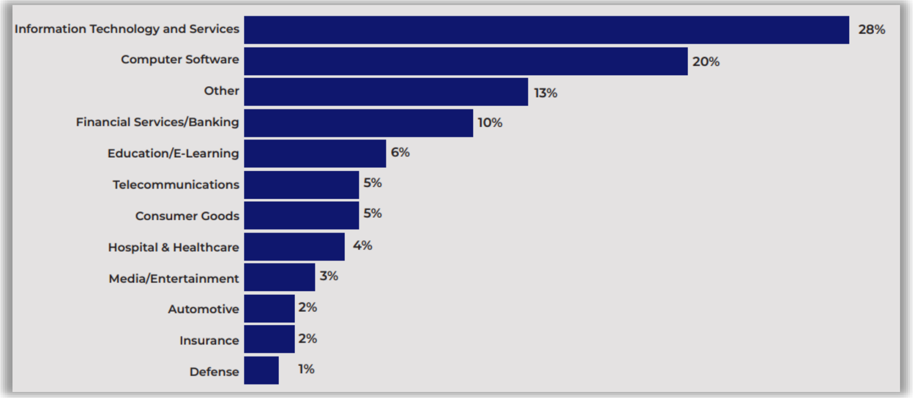
Numerous reasons:

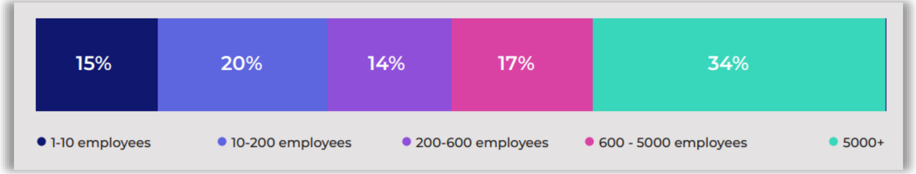
1. Systems are *dynamic/quick to update*, models are static and will always lag
2. Systems provide *easier control and trust* over black box *monolithic* models.
3. Systems offer *cost and SLA flexibility* which vary widely for each context/deployment

2023 State of FMware in Industry

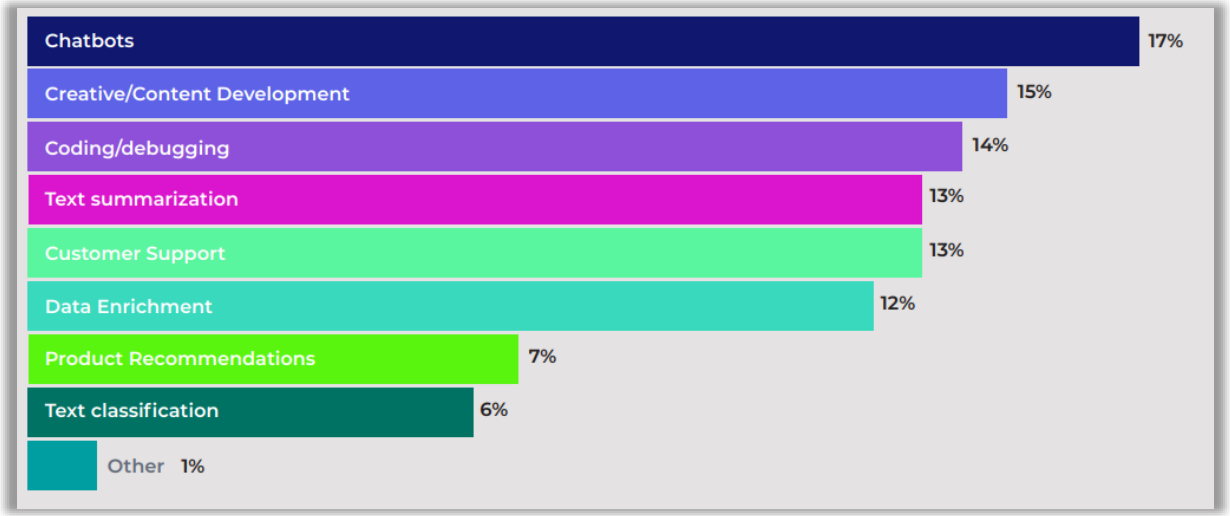
Intel Survey of 434 companies half of which have over 600 employees

Demographics

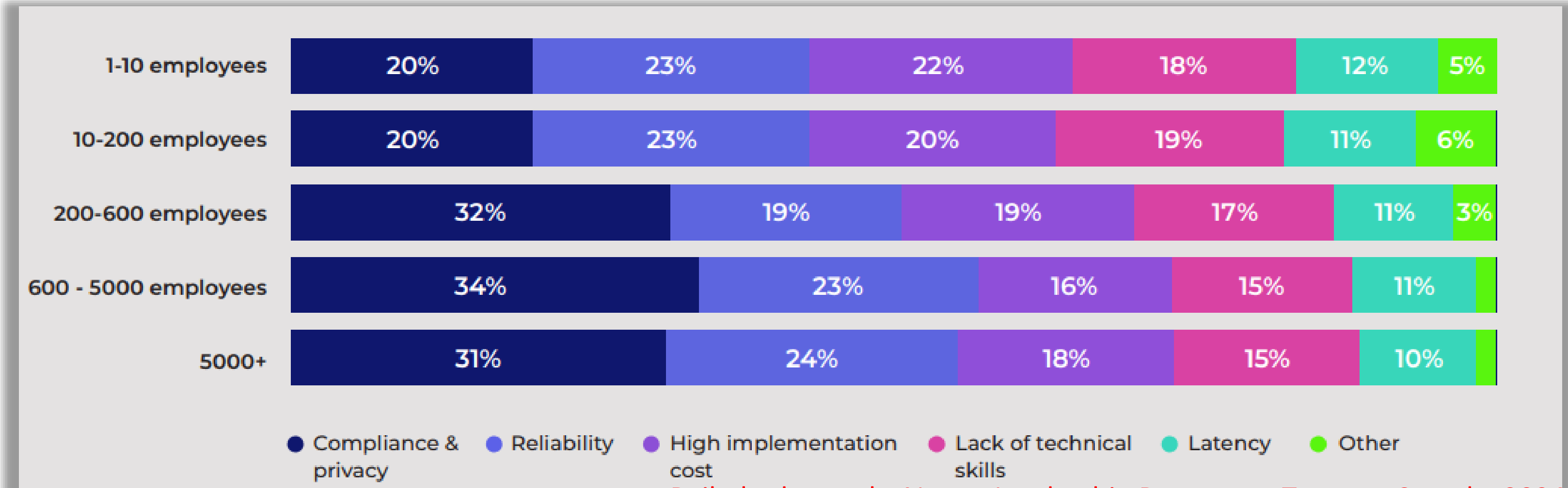




Use Cases



Greatest concerns around FMware



CSE Focus

2022-2024: Implementation Costs + Tech Skill Gap + Governance

2025-: Reliability + Latency + Governance

Rajbahadur et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Going from cool-demos to production-ready FMware is extremely hard!

LinkedIn Engineering Blog

Consistent quality… The team achieved 80% of the basic experience we were aiming to provide within the first month and then spent an additional four months attempting to surpass 95% completion of our full experience - …., the initial pace created a false sense of 'almost there,' which became **discouraging as the rate of improvement slowed significantly for each subsequent 1% gain.**

" **The complexity of these systems surpases anything that we have seen before, Neuralware was hard this stuff is REALLY HARD!! Very few people trained and they are too pricey!!**"

Red Hat

"most of these, like each of these tests, would **probably cost 1-2 cents to run,** but once you end up with a lot of them, that will start adding up anyway". P4 attempted to automate testing but was asked to stop their efforts because of costs in running benchmarks, and instead would only run a small set of them manually after large changes

Microsoft

GitHub

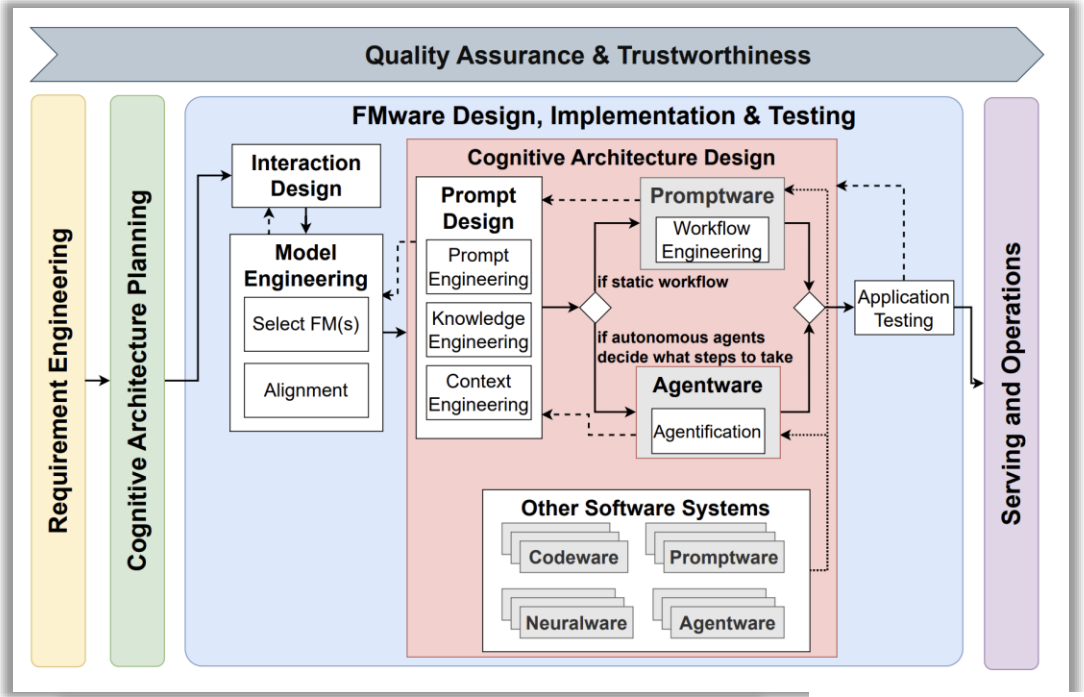
"There is a large class of problems that are easy to imagine and build demos for, but extremely hard to make products out of. For example, self-driving: **It's easy to demo a car self-driving around a block, but making it into a product takes a decade."** Karpathy

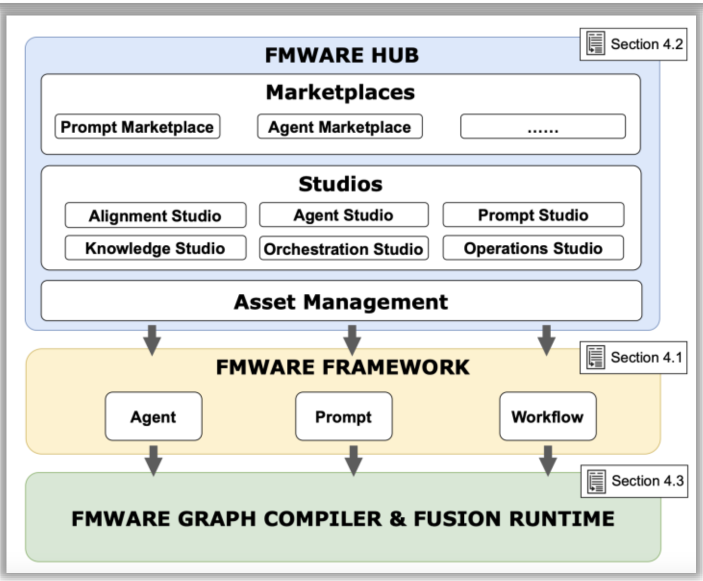
TESLA

Rethinking Software and Software Engineering in the Foundation Model Era

SE Challenges:

1. Managing alignment data
2. Crafting effective prompts (Intent Alignment)
3. Multi-generational software components
4. Degree of controllability
5. Compliance & regulations
6. Limited collaboration support
7. Operation & semantic observability
8. Performance engineering
9. Quality under non-determinism
10. Siloed tooling & lack of processes





Rajbahadur et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Thematic analysis to identify challenges that prevent productionizing FMware

Step 1: Data collection – 3 years of attending and organizing conferences in the AIware space and meticulously documenting FMware related interactions

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

FSE 24

AI_dev Open Source GenAI & ML Summit

AIWARE 2024 BRAZIL

SEMLA Trustworthy Engineering of AI Software

PORTUGAL LISBON ICSE 24 INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING CCB

RAY SUMMIT Presented by Anyscale

FM+SE SUMMIT 2024 | 2024 AIware Latam | 2024 AIware Bootcamp | 2024 FM+SE Summit | 2023 FM+SE Vision | 2023 FM+SE School

Thematic analysis to identify challenges that prevent productionizing FMware

Step 1: Data collection – Open source working groups, working with customers, developing FMArts and in-depth literature review

Open Platform for Enterprise AI

Leading the research working group in the OPEA initiative and participation in the other Working Groups

SPDX

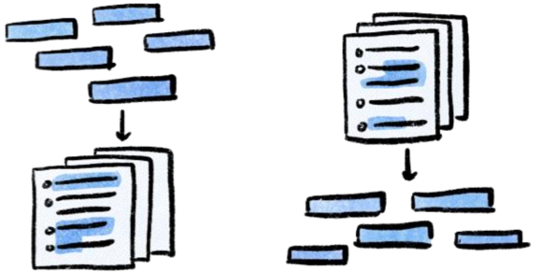
Leading AI and Dataset profile Working Group

Discussions with product teams and customers in the Industry. Experience developing FMArts and deploying it in production and building production-ready FMware for a large e-commerce customer

Conduct an in-depth literature review of academic and grey literature

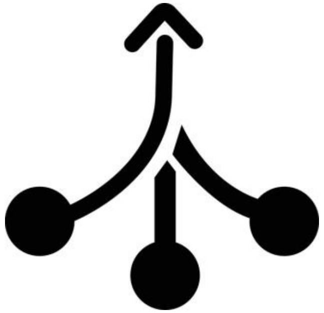
Thematic analysis to identify challenges that prevent productionizing FMware

Step 2: Data coding and initial thematic grouping



Step 3: Collaborative discussion to identify overarching themes

Step 4: Thematic consolidation



Recurrent issues in productionizing FMware

- Lack of assertion-based unit tests
- Text-based evaluation leads to overestimation of quality
- Ineffective and inefficient AI-as-judge technologies
- Difficulty navigating the regulatory and legal compliance minefield
- Lack of automated testing capabilities

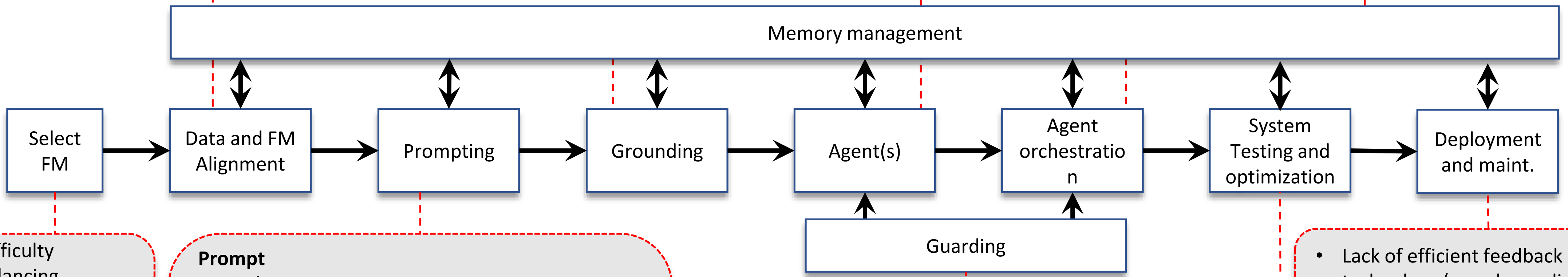
Ineffective prompt-level context management
- Low information density grounding data
- Irrelevant grounding data
- Leveraging advanced techniques where simpler ones (keyword search) would suffice

- Overreliance on FM planning capability instead of step-wise or multi-agent planning and validation

- Low data efficiency
- Low domain coverage
- Low data quality

- God Agents
- Too low-level tool usage abstraction
- Capability centric instead of use-case centric documentation for tools

- Inefficient knowledge representation
- Cumbersome and error-prone in-memory and across-memories knowledge management



- Difficulty balancing functional requirements with performance and costs

Prompt
- God prompts
- Lack of Built-in QA checks for prompts (Semantic and structured checking)
- Lack of portability across FMs
- Lack of FM-specific optimizations
- Complexity of determining the rationale of failure (FM vs prompt limitations)

In-context learning
- Insufficient Examples
- Non-representative Examples

- Simple keyword based guarding is ineffective
- Lack of Hallucination guardrails

- Lack of efficient feedback technology (seamless solicitation and integration)
- Lack of FMware-native observability

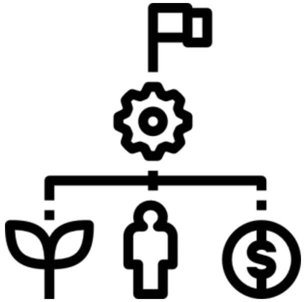
- Lack of latency handling mechanisms
- Lack of retry optimizations
- High cost of regression testing
- No Software Performance Engineering practices in place (model swapping)
- Lack of controlled execution mechanisms makes verification of fixes very hard

Challenges for Production-Ready FMware

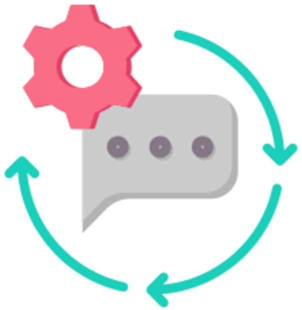
Testing

Observability

Controlled Execution



Resource-aware QA



Feedback Integration

Built-in Quality

Rajbahadur et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

Lack of assertion based unit tests

- Traditional tests rely on fixed outcomes, but FMware's varying outcomes complicate this process.

Text-based evaluation leads to overestimation of quality

- They often fail to capture deeper issues like relevance, accuracy or alignment with business rules

Lack of automated testing capabilities

- Existing tools often miss nuanced complexities in FMware making them inadequate for production systems

Challenges for Production-Ready FMware – SOTA Overview

Testing | Observability | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

Benchmarks ≠ Model evaluation ≠ Software testing

Benchmarks

Assessing model's general capabilities

Model evaluation

Evaluating model's fit on a specific task

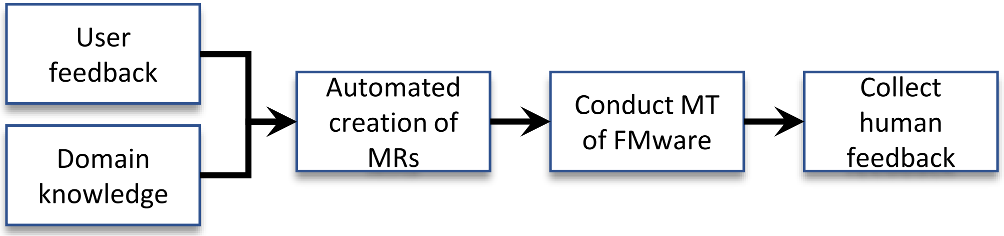
Software testing

Testing the expected behavior of the whole system end-to-end

Challenges for Production-Ready FMware – Proposed Tech

Testing | Observability | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

Automated test generation



Need an automated Metamorphic Testing suite that automatically extracts MRs leveraging domain knowledge, user feedback in the wild and human guidance

Next generation AI-as-judge framework

Tailored Prompts: Align with business logic and domain constraints for reliable evaluations.
Deep Evaluation: Focus on accuracy, consistency, and compliance, not just format.
Efficient Training: Use curriculum engineering for cost-effective, lightweight models.

Challenges for Production-Ready FMware – Recurrent issues

Testing | **Observability** | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

Complexity in determining failure rationale

- Diagnosing whether failures result from prompt issues or FM shortcomings is difficult, often hindering the identification of the root cause and delaying effective resolution.

Lack of FMware-native observability

- Traditional observability tools focus primarily on functional observability metrics, such as latency, execution races, and throughput, which are insufficient for capturing the internal reasoning and decision-making processes of FMs

Challenges for Production-Ready FMware – SOTA Overview

Framework	Trace or Request-Response (RR) Monitoring	Volume		Latency (PX)						Tokens			Resources	
		Trace Count/Status	LLM Call Count/Status	Trace Latency	LLM Latency	LLM Calls per Trace	Tokens / sec	Trace Time-to-First-Token	LLM Time-to-First-Token	Total Tokens	Tokens per Trace	Tokens per LLM Call	Cost	HW Util.
LangSmith	Trace	Yes		Yes						Yes			Yes	
HumanLoop	RR	Yes		Yes						No			No	
Qwak	Trace	Yes		Yes						Yes			Yes	
Helicone	RR	Yes		Yes						Yes			Yes	
Langfuse	Trace	Yes		Yes						Yes			Yes	
Dynatrace	Trace	Yes		Yes (very simple)						Yes			Yes	
Pheonix (Arize)	Trace	Yes		Yes						Yes			Yes (cost – Arize)	
Lunary	Trace	Yes		Yes						Yes			Yes (cost)	
LangKit (WhyLabs)	Trace	Yes		No (?)						No (?)			No (?)	
Laminar	Trace	Yes		Yes						Yes			No	
TraceLoop	Trace	Yes		Yes						Yes			No	
DataDog	RR	Yes		Yes						Yes			Yes (cost)	

Focusing on low level details, primarily collecting traces and runs consisting of LLM calls, Vector DB calls, user prompts, and associated metrics (e.g., volume, latency, token counts, resource utilization, etc.).

OpenLLMetry https://github.com/traceloop/openllmetry

- Use existing standard OpenTelemetry instrumentations for LLM providers and Vector DBs
- Support some new LLM-specific extensions

for example OpenAI, Anthropic API calls

Challenges for Production-Ready FMware – Recurrent issues

Testing | **Observability** | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

General Observability Framework

Captures functional metrics and FM/agent cognitive processes, enabling traceability across decision-making stages.

Plane Flight Recorder for Agents

Selectively logs reasoning steps and communications to trace decision-making pathways.

Observability Analytics Engine

Visualizes events at high abstraction levels, aiding root cause analysis in complex workflows.

Surrogate Agent for Debugging

Mitigates observer effects by reasoning verbosely, offering transparent insights without altering original behavior.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | **Controlled Execution** | Resource-aware QA | Feedback Integration | Built-in quality

Lack of controlled-execution mechanisms

- Without tools like feature flags, staging environments, or canary releases, testing changes before full deployment becomes risky.

Ineffective FM update mechanisms

- Fixes can take months, and updates are non-deterministic—providing more training data does not ensure specific issues are resolved.

Challenges for Production-Ready FMware – SOTA Overview

Challenges of Non-Deterministic Outputs

FMware's unpredictability leads to varying execution paths, making traditional deterministic testing methods ineffective.

Need for Controlled Execution Frameworks

The lack of repeatable execution and exploratory testing tools hinders efficient failure identification and performance optimization.

Testing and Verification Limitations

Limited control over cloud-hosted models and insufficient release documentation impede verifying updates and building user trust.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | **Controlled Execution** | Resource-aware QA | Feedback Integration | Built-in quality

Controlled Execution Framework

Balances repeatability and variability in execution paths for thorough testing and validation.

Repeatable Execution Framework

Ensures consistent execution flows via snapshots and monosemantic units, enabling precise debugging and fix verification.

Guided Exploration Engine

Systematically varies inputs and decisions to explore alternative paths, uncover hidden bugs, and enhance system robustness.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | **Resource-aware QA** | Feedback Integration | Built-in quality

Low data efficiency

- Accurately estimating inbound and outbound data volume is crucial for production-ready FMware, impacting both budget and performance.

High cost of regression testing

- As FM APIs evolve, performance can degrade silently, making it essential to frequently re-evaluate across numerous scenarios to ensure system reliability.

Ineffective and inefficient AI-as-judge technologies

- Existing tools often miss nuanced complexities in FMware making them inadequate for production systems

Challenges for Production-Ready FMware – SOTA Overview

Lack of FMware native caching

Traditional caching fails to handle FMware's dynamic nature

Test Optimization Issues

Lack of dependency-based test execution leads to redundant testing, wasting computational resources and increasing costs.

Lack of Retry Mechanisms

Current QA frameworks lack intelligent retry systems, requiring advanced re-prompting to handle FM failures efficiently.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | **Resource-aware QA** | Feedback Integration | Built-in quality

Resource-Aware QA Framework

Reduces FM calls and computational overhead with intelligent caching and dynamic test optimization.

FMware-Native Caching

Stores and reuses query results, ensuring cache integrity and efficiency during large-scale testing.

Test Execution Optimization

Dynamically prioritizes and groups tests to minimize resource use and detect defects early.

Retry Optimization System

Adapts prompts and validates responses to handle FM unpredictability, preventing repeated failures.

Rajbahadur et al., AIware Leadership Bootcamp, Toronto, Canada, 2024

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | Resource-aware QA | **Feedback Integration** | Built-in quality

Lack of efficient feedback technology

- Production environments, unlike demos, require automated, iterative feedback systems to enable continuous learning, performance optimization, and user engagement.

Error prone in-memory knowledge management

- Conflicting knowledge within memory systems can lead to inconsistencies and errors when FMs must navigate between contradictory pieces of information.

Challenges for Production-Ready FMware – SOTA Overview

Inadequate Feedback Mechanisms

Reliance on explicit, ad-hoc methods fails to enable passive, scalable feedback collection essential for real-time FMware refinement in production.

Feedback Differentiation

Lack of a framework to manage universal outer knowledge vs. user-specific inner knowledge feedback leads to overlooked insights and inconsistent performance.

Fragmented Feedback Integration

Poor memory and context management across FMware systems hinder synchronization and consistency in leveraging feedback effectively

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | Resource-aware QA | **Feedback Integration** | Built-in quality

Feedback Integration Framework

Captures, processes, and acts on user feedback for continuous FMware improvement.

Automatic Feedback

Passively collects implicit cues (e.g., hesitations, corrections) during user interactions for real-time analysis.

Guided Exploration Engine

Differentiates "outer" and "inner" knowledge, enabling scalable, user-specific, and global optimizations through federated learning.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | Resource-aware QA | Feedback Integration | Built-in quality

God prompts

- God prompts” that try to handle multiple tasks at once lead to unpredictable outputs and complicate debugging and maintenance

God agents

- Monolithic “God agents” handling many tasks create complexity and maintenance challenges in production.

Overreliance on FM’s planning capability

- Relying solely on an FM’s planning capabilities introduces risks like unpredictability and lack of transparency, often leading to errors and unreliable outcomes

Challenges for Production-Ready FMware – SOTA Overview

Poor development practices

"God prompts" and "God agents" hinder modularity, scalability, and debugging, complicating FMware development.

Unstructured Knowledge Management

Reliance on raw data in vector databases degrades performance; improved tools and formalized curricula are needed for efficient compositional skill-building.

Immature QA and Compliance Mechanisms

Lack of built-in prompt validation and incomplete compliance frameworks leave FMware prone to unpredictable behavior and legal risks.

Challenges for Production-Ready FMware – Recurrent issues

Testing | Observability | Controlled Execution | Resource-aware QA | Feedback Integration | **Built-in quality**

Knowledge Graphs and Curriculum Engineering

Shift from vector databases to knowledge graphs and use structured curriculum for compositional skill-building through collaborative AI-assisted co-creation.

Curriculum Optimization Tools

Automate QA to prune outdated data, remove redundancies, and ensure high-quality inputs for reliable FMware performance.

FMware Bill of Materials (FMwareBOM)

Extend SPDX 3.0 to track all components and licenses, ensuring legal compliance with automatic generation and formal verification.

Production-Ready FMware–Challenges and Core Technologies

	Challenge 1: Testing	Challenge 2: Observability	Challenge 3: Controlled Execution	Challenge 4: Resource-Aware QA	Challenge 5: Feedback Integration	Challenge 6: Built-in Quality
Recurrent issues	• Lack of assertion-based unit tests • Lack of automated testing capabilities • Text-based evaluation leads to overestimation of quality	• Complexity of determining the rationale of failure (FM vs prompt limitations) • Lack of FMware-native observability	• Lack of controlled execution mechanisms makes verification of fixes very hard	• Low data efficiency • Lack of latency handling mechanisms • Lack of retry optimizations • High cost of regression testing • No Software Performance Engineering practices in place • Ineffective and inefficient AI-as-judge technologies	• Lack of efficient feedback technology (seamless solicitation and integration) • Cumbersome and error-prone in-memory and across-memories knowledge management	• Low domain coverage, Low data quality • Lack of Built-in QA checks for prompts (Semantic and structured checking) • Non-representative or insufficient Examples • God prompts, God agents • Low information density or irrelevant grounding data • Overreliance on FM planning capability instead of step-wise or multi-agent planning and validation • Lack of Hallucination guardrails
Core Technologies	• **Automated test generation** • **Next generation AI-as-judge framework**	• **General Observability Framework** • **Plane Flight Recorder for Agents** • **Observability Analytics Engine** • **Surrogate Agent for Debugging**	• **Controlled Execution Framework** • **Repeatable Execution Framework** • **Guided Exploration Engine**	• **Resource-Aware QA Framework** • **FMware-Native Caching** • **Test Execution Optimization** • **Retry Optimization System**	• **Feedback Integration Framework** • **Automatic Feedback** • **Guided Exploration Engine**	• **Knowledge Graphs and** • **Curriculum Engineering** • **Curriculum Optimization Tools** • **FMware Bill of Materials (FMwareBOM)**

Production-Ready FMware–Challenges and Core Technologies

	Challenge 1: Testing	Challenge 2: Observability	Challenge 3: Controlled Execution	Challenge 4: Resource-Aware QA	Challenge 5: Feedback Integration	Challenge 6: Built-in Quality
Recurrent issues	• Lack of assertion-based unit tests • Lack of automated testing capabilities • Text-based evaluation leads to overestimation of quality	• Complexity of determining the rationale of failure (FM vs prompt limitations) • Lack of FMware-native observability	• Lack of controlled execution mechanisms makes verification of fixes very hard	• Low data efficiency • Lack of latency handling mechanisms • Lack of retry optimizations • High cost of regression testing • No Software Performance Engineering practices in place • Ineffective and inefficient AI-as-judge technologies	• Lack of efficient feedback technology (seamless solicitation and integration) • Cumbersome and error-prone in-memory and across-memories knowledge management	• Low domain coverage, Low data quality • Lack of Built-in QA checks for prompts (Semantic and structured checking) • Non-representative or insufficient Examples • God prompts, God agents • Low information density or irrelevant grounding data • Overreliance on FM planning capability instead of step-wise or multi-agent planning and validation • Lack of Hallucination guardrails
		bility ork ght for bility s	• **Controlled Execution Framework** • **Repeatable Execution Framework** • **Guided Exploration Engine**	• **Resource-Aware QA Framework** • **FMware-Native Caching** • **Test Execution Optimization** • **Retry Optimization System**	• **Feedback Integration Framework** • **Automatic Feedback** • **Guided Exploration Engine**	• **Knowledge Graphs and** • **Curriculum Engineering** • **Curriculum Optimization Tools** • **FMware Bill of Materials (FMwareBOM)**

arXiv