FM+SE Vision 2030 Summary Report: Challenges and Opportunities in the Road Ahead

Ahmed E. Hassan, Queen's University, Canada
Bram Adams, Queen's University, Canada
Haoxiang Zhang, Queen's University, Canada
Thomas Zimmermann, Microsoft Research, USA
Foutse Khomh, Polytechnique Montreal, Canada
Nachi Nagappan, Meta, USA

Contents

Event Summary and Feedback from Experts 3
Event Highlights 3
Pain Points and Suggestions 4
Recommendations for Future Events 4
FM+SE Vision Opening Session 5
Overview 5
Key Highlights 5
Talk 1: FM+SE 101 Technical Briefing (Part 1) 8
Overview 8
Key Highlights 8
Talk 1: FM+SE 101 Lectures (Part 2) 13
Key Highlights 13
Talk 3: Beyond Code - Helping Software Teams with LLMs (Microsoft) 17
Overview 17
Key Highlights 17
Talk 4: Large Sequence Models for Software Development Activities (Google) 19
Overview 19
Key Highlights 19
Talk 5: Open and Responsible Development of Large Language Models for Code (ServiceNow) 21
Overview 21
Key Highlights 21
Talk 6: CodeCompose - A Large-Scale Industrial Deployment of AI-assisted Code Authoring (Meta) 23
Overview 23
Key Highlights 23
Talk 7: Towards Better Software Quality in the Era of Large Language Models (UIUC) 25
Overview 25
Key Highlights 25
Talk 9: Building Trustworthy Foundation Model Applications (Huawei Canada) 27
Overview 27
Key Highlights 27
Talk 10: The Legal Landscape of ML/FM Systems (University of Victoria) 30

Overview 30
Key Highlights 30
Panel A - Impact of Foundation Models on Software 32
Panelists 32
Discussion Highlights 32
Panel B: Impact of Foundation Models on Developer Productivity 34
Panelists 34
Discussion Highlights 34
Additional Insights and Anecdotes 35
Important Dates 35
Panel C: SE Challenges for Foundation Models 37
Panelists 37
Discussion Highlights 37
Audience Interaction 38
Panel D: Foundation Models for Software Engineering 40
Panelists 40
Discussion Highlights 40
Trustworthiness and Education in Model Use 40
Multimodal Data in Software Engineering 41
Closing Thoughts 41
Panel E: Foundation Models and Open Source 42
Panelists 42
Discussion Highlights 42
Future Perspectives 43
Audience Interaction and Queries 43
Panel F: Foundation Models and Software Trustworthiness 44
Panelists 44
Discussion Highlights 44
Acknowledgement 46

Event Summary and Feedback from Experts

The "FM+SE Vision 2030" meeting (https://fmse.io/vision/) was a gathering of professionals and thought leaders in the fields of Artificial Intelligence/Foundation Model (AI/FM) and Software Engineering (SE). The event aimed to foster understanding and collaboration in integrating these two important domains. The event jumpstarted the AIware community. And the turnout was fantastic! We had 108 attendees (57 from industry and 51 from academia), plus 50+ folks joining in via Zoom. It was a mix of a Dagstuhl with deep discussions and a school with great tutorials and industrial keynotes. The active involvement of industry professionals brought a real-world perspective to the discussions, touching on the genuine pain points faced by practitioners today. An event recap video is at https://www.youtube.com/watch?v=isnj0x8-7sM. Videos for a few of the talks are posted on the AIware YouTube Channel (https://www.youtube.com/@aiware2023). This report gives an in-depth overview of the event. For more information check https://fmse.io/ and https://www.aiwareconf.org/.

Event Highlights

- **The definition and alignment of the Software 4.0 concept and AIware vision by attendees from various companies like Microsoft, Meta, and Google as well as academia.**
- **Futuristic and Constructive Content:** Attendees praised the event's significant contribution to advancing the software engineering (SE) field and software in general. The content was not only seen as futuristic but also as constructive and beneficial for the industry's progression.
- **Diversity and Expertise of Speakers:** The selection of speakers, including a blend of established and emerging professionals from both academia and industry, was a standout feature. Specific appreciation was given to the depth and insightfulness of talks, especially those focusing on industry-specific tools like DIDACT and CodeCompose. This diversity in speakers fostered an environment of engaging and informative discussions, enhancing the overall quality of the event.
- **Organizational Excellence and Interactive Format:** The balance between theoretical discussions and practical demonstrations, particularly in the context of FM technologies, was appreciated. The combination of panel discussions, practical demos, and breakout sessions was seen as conducive to a comprehensive learning and networking experience.
- **Networking and Communication Opportunities:** The event's approach to fostering networking was particularly commended. The strategic seating arrangement during meals was highlighted as an innovative method for promoting interaction and exchange among participants, enhancing the overall networking experience.
- **Confluence of Industry and Academic Perspectives:** The gathering of insights from both industry leaders and academic figures was seen as a key strength. This aspect allowed for a rich exchange of opinions and insights, bridging the gap between theoretical research and practical industry applications. Participants valued the opportunity to gain first-hand insights into the dynamics and practices of leading companies in the industry.
- **International Reach and Achievement of Goals:** The international scope of the event was recognized, with its ability to meet the objectives of an international seminar being particularly emphasized. This feature was important for attendees from various

geographical regions and industry sectors, who found the event to be aligned with their expectations of an international forum.

Pain Points and Suggestions

- **Broader Topic Inclusion:** Attendees expressed a desire to see more discussions around the integration of software engineering with hardware aspects, suggesting that this would provide a more holistic view of the challenges ahead.
- **Detailed Application Discussions:** Requests were made for more in-depth discussions on applying FMs in various areas of software engineering beyond just coding and code review, such as requirement engineering, architecture design, as well as testing and verification.
- **Diversity in Organized Debates and Panels:** A suggestion was made to include organized debates to surface deeper issues, and to have more diverse panels, potentially including more speakers with extensive industry experience.
- **In-Depth Exploration of Emerging Topics:** Attendees showed interest in deeper dives into specific areas, such as the training process of big code models, quality standards of training data, development of domain-specific agents, and the role of FMs in different phases of software development.

Recommendations for Future Events

- **Expanding the Range of Topics:** Inclusion of discussions on system-level software interactions with hardware systems and broader applications of FMs in software engineering would be beneficial.
- **Enhancing Speaker Diversity and Depth:** Future events could benefit from a broader range of speakers, especially those with deep insights into emerging FM+SE topics.
- **Global Accessibility:** Considering hosting similar events in other regions like Asia-Pacific and Europe could cater to a more diverse audience, accommodating the needs of attendees from various regions.
- **Continued Engagement Post-Event:** Establishing platforms like Slack for ongoing discussions and networking post-event would help maintain the momentum and community engagement generated by the event.

This comprehensive synthesis of the feedback provides a roadmap for enhancing the future editions of the event, focusing on expanding its scope, diversifying its content and speakers, and fostering ongoing community engagement.

FM+SE Vision Opening Session

Overview

The "FM+SE Vision 2030" meeting was a gathering of professionals and thought leaders in the fields of Artificial Intelligence/Foundation Model (AI/FM) and Software Engineering (SE). This meeting aimed to foster understanding and collaboration in integrating these two important domains.

Key Highlights

Concept and Significance of Foundation Model

- **Terminology Preference:** Emphasis was placed on using the term "Foundation Model (FM)" instead of "Large Language Models (LLM)" to highlight a broader scope beyond language, encompassing multimodality and foundation aspects.
- **Industry Endorsement:** Quotes from prominent figures like Sam Altman, CEO of OpenAI, and Emad Mostaque, CEO of Stability AI, emphasized the importance of recognizing the foundation nature of these models over their size, underlining their multifaceted potential and implications for various industries.

The Future Vision of Software Engineering

- **Paradigm Shift in Software Engineering:** The meeting highlighted that AIware, i.e., AI-powered software, has the potential to democratize software creation. The definition of software along with many Software Engineering (SE) aspects, processes, tools, platforms and techniques needs to be either reimagined, reformulated or redesigned, enabling individuals of all backgrounds to participate in its creation with higher reliability and quality. Over the past decade, software has evolved from human-driven ***Codeware*** to the first generation of AIware, known as ***Neuralware***, developed by AI experts. Foundation Models (FMs, including Large Language Models or LLMs), ushered in software's next generation, ***Promptware***, led by domain and prompt experts. However, this ***Promptware*** merely scratches the surface of software's future. We are already witnessing the emergence of the next generation of software, ***Agentware***, in which humans and intelligent agents jointly lead the creation of software. With the advent of brain-like World Models and brain-computer interfaces, we anticipate the arrival of ***Mindware***, representing the 5th generation of software. Agentware and Mindware promise greater autonomy and widespread accessibility, with non-expert individuals, known as Software Makers, offering oversight to autonomous agents. Such a progression reflects the increasing sophistication and integration of AI in SE.
- **Focus Areas in Software Engineering:**
  - **Foundation Models for SE:** The concept of intelligence as a service was discussed, highlighting the integration of Foundation Models in SE.
  - **SE for Foundation Models:** Attention was given to the development of sophisticated foundation models and the role of SE in enhancing such sophisticated software systems.

- **Rethinking Software Engineering:** There was a consensus on the need to redefine SE in light of the new challenges and opportunities that are presented by foundation models.

Shared Perspectives from Industry Leaders

- **Yoshua Bengio's (the University of Montreal) Observation:** The enormous potential and rapidly-evolving capabilities of foundation models in the future of software were recently recognized.
- **Yann LeCun's (Meta) Critique:** While recognizing the limitations of generative models, their significant capabilities were also noted.
- **Sal Khan's (Khan Academy) Viewpoint:** The focus was on the positive applications of AI in augmenting human intelligence.
- **Satya Nadella's (Microsoft) Comments:** The unprecedented workloads of these models are unlike anything seen in Microsoft Azure's history.
- **Sam Altman's (OpenAI) Urgency:** The importance of rapid and safe integration of foundation models into products that are accessible to humanity, despite such models not being fully perfected.

Meeting Structure and Activities

- **Division of Events:** The event was structured into two parallel track: Vision track for strategic discussions, and School track for educational purposes. The event included joint sessions in the mornings followed by separate tracks for Vision and School. Breakout sessions facilitated focused-= discussions on specific themes.
- **Networking Opportunities:** The agenda included networking sessions to promote interactions among diverse participants, fostering collaborative discussions and idea exchange.
- **Future Events and Conferences:** Announcements were made about upcoming events and conferences related to the integration of Foundation Models and SE.

Demos and Practical Applications

- **GitHub Copilot Workspaces and the importance of keeping the developer-in-the-loop:** GitHub demonstrated Copilot Workspaces, showcasing its capabilities in AI-powered feature\task realization. The demo highlighted the tool's ability to interpret and implement coding tasks from natural language descriptions while keeping the developer-in-the-loop, underscoring the advancements in AI-assisted coding. The challenges in evaluating and validating AI-generated code were discussed, with an emphasis on the need for tools that allow user steerability and intervention in the AI-powered process.
- **Command line copilot:** The presentation by Diomidis Spinellis showcased the ai-cli-lib library, a breakthrough in integrating AI with command line interfaces, allowing existing CLI tools like Bash, SQLite, and GDB to execute AI-enhanced commands. Spinellis demonstrated the library's utility in automating command execution, querying databases, and more, through a live demo. Highlighting the use of AI to streamline developer workflows without modifying original tools, the initiative marks a significant advancement in AI-assisted programming. The importance of foundational programming knowledge for effectively leveraging such AI capabilities was also emphasized, illustrating the balance

between automation and critical evaluation in the evolving landscape of programming tools.

- **FMArts engineering and productivity platform:** The demo by Dayi Lin, Huawei Canada, showcased a FMArts, which is a long-term effort towards creating a cradle-to-grave platform for the engineering of trustworthy FMware. He pointed some of the unique properties of FMArts that enable the design and development of complex FMware in a timely manner and he briefly discussed some his first hand experiences in developing such complex FMware.
- **Design document reviewing:** The demo by Susumu Tokumoto showcased Fujitsu's vision in leveraging foundation models to automate various software engineering tasks like design document reviewing.

Important Dates and Future Events

- Upcoming related events include an AI+SE Shonan meeting, a Montreal AI+SE school event, and the first ACM international conference on AIware in conjunction with FSE 2024.

This session highlighted the need for a collaborative spirit and forward-thinking discussions around foundation models in software engineering with the event serving as a platform for sharing insights, exploring new ideas, and setting the stage for future advancements in the integration of Foundation Models and Software Engineering.

Talk 1: FM+SE 101 Technical Briefing (Part 1)

Overview

- **Presenters:** The briefing was led by Prof. Ahmed E. Hassan and Prof. Bram Adams from Queen's University as well as Prof. Zhen Ming (Jack) Jiang from York University, with collaboration of several researchers from Huawei Canada. The briefing focused on foundation models in the context of SE and SE in the context of foundation models.
- **Objective:** The briefing provided an overview of foundation models, bridging them with SE perspectives. This approach reflects a shift in industry terminology from "large language models" to "foundation models," as exemplified by IBM's strategy to use foundation models in customer communication.
- **Context:** The speakers discussed a course that they designed and are currently teaching. This course is pioneering in its aim to fundamentally rethink SE and Software with foundation models at the core, suggesting a significant paradigm shift in the field.

Key Highlights

Concept and Capabilities of Foundation Models

- **Defining Foundation Models:** These models are defined as being trained on extensive and diverse data sets, primarily using scalable self-supervision methods. The models are adaptable to a wide range of downstream tasks.
- **Emergent Abilities and Evolution:** The models are noted for their emergent abilities, like reasoning and instruction following, which are their most exciting aspects. There's a progression from traditional coding paradigms (*codeware*, *neuralware*) to more advanced concepts like *promptware*, *agentware*, and *mindware*, suggesting a profound evolution in the definition of software and how we interact with software.

Future of Software Engineering with Foundation Models

- **Emerging Software Trends:** The briefing touched upon new types of software, such as conversational and probabilistic software, reflecting a trend towards accepting imperfection (software that can be sometimes wrong) in the output of the software.
- **Complexity in Building Foundation Models:** Building these sophisticated models involves dealing with complex, distributed software systems. This complexity underscores the urgent need for SE innovations to deliver such sophisticated systems.

Flash Attention and Software Engineering Implications

- **Chris Ray's Influence and Flash Attention:** The speaker introduced Chris Ray's concept of 'Flash attention', underscoring the need for making complex systems accessible to average users. Ray's work highlights the balance between complexity and accessibility.
- **The Frog Analogy and Hype Cycle:** The discussion included the "frog in boiling water" analogy in the context of the hype cycle, where foundation models are at the peak of inflated expectations, bringing along promises but also inconsistencies in areas like trustworthiness, safety, efficiency, and productivity.

Interaction with Foundation Models

- **Different Interaction Levels:** The briefing described various levels of interaction with foundation models, ranging from simple task instructions to complex functionalities where the model itself becomes a function, a software component, or even a complete software system.
- **Limitations and Challenges:** Foundation models, despite their advanced capabilities, face limitations (and emergent limitations as our use of them continues to evolve and expand) like inadequate complex reasoning, tendencies for hallucinations, closed-loop constraints, and limited context understanding.

Foundation Models in Programming

- **Broader Engagement in Programming:** The briefing drew analogies with calculators and Excel, suggesting that foundation models, like these tools, enable a broader range of people to participate in the craft of software creation.
- **Shift from Developers to Makers:** There's a significant shift from traditional software developers to 'software makers', indicating a democratization of software creation and a move towards inclusive, accessible software development.
- **Different Programming Levels:** The briefing classified programming into end-user, enterprise, and system programming, each with different degrees of human involvement and expertise requirements, while highlighting the current status of the integration of foundation models in each of these levels of programming.

Trustworthiness and Software Evolution

- **Critical Role of Trustworthiness:** The importance of trustworthiness in foundation models was emphasized, especially given their increasing integration into various software systems.
- **Changing Software Development Landscape:** The briefing noted shifts in software development roles, given the growing integration of AI and the changing responsibilities of developers.
- **Practical Application - Tesla Example:** An example from Tesla was discussed, showcasing how AI models are used in developing features like auto-wipers where the software is based on data and AI models (aka. *AIware*) instead of traditional coding (aka. *Codeware*).

Lifecycle and Future of FMWare

- **Reimagining the Software Engineering Lifecycle:** The need to rethink the entire software engineering lifecycle in the context of foundation models was highlighted, encompassing aspects like model building, integration, performance engineering, operational management, and governance.
- **From CPUs to Foundation Models as Compute Units:** The briefing projected a future where foundation models would serve as new types of computational units, replacing traditional CPUs, and enabling more flexible and powerful computing paradigms.
- **GPT Evolution and Licensing Models:** The evolution of ChatGPT models like GPT-4 turbo was discussed, alongside changes in licensing models that affect how these technologies can be accessed and leveraged.

Data Pipeline, Model Training, and Challenges

- **Managing Complex Data Pipelines:** The complexity involved in managing data pipelines for foundation models was noted, highlighting the challenges in ensuring data quality and pipeline reliability.
- **Model Training Techniques:** Various methods for fine-tuning models for specific tasks were discussed, including human-in-the-loop approaches and supervised fine-tuning, which are essential for enhancing the performance and relevance of models.

Specialized Fine-Tuning: Costs and Challenges

- **Financial and Resource Challenges:** The high costs and resource-intensive nature of fine-tuning foundation models for specialized applications were discussed, emphasizing the need for significant investment in this area.
- **Anthropic's Constitutional AI:** The emergence of constitutional AI by Anthropic was highlighted as a new development in AI, focusing on aligning AI systems more closely with human values and requirements in an automated and lower cost manner over current human-intensive approaches.

Concept of Constitutional AI and Its Implications

- **Introduction to Constitutional AI:** The idea of training a GPT model with a constitutional framework, akin to a set of ethical or operational guidelines, was discussed. This concept involves leveraging AI models to check the outputs of the being-built model against these guidelines.
- **Reducing Human Involvement:** This approach aims to minimize human involvement in the training feedback loop. By embedding a constitutional framework, one can auto-regulate a model's outputs according to predefined ethical or operational standards, thus automating the rather costly process of ensuring appropriate responses.

Challenges in Foundation Model Development and Industry Practices

- **Complexities in Human Feedback Integration:** The briefing highlighted the associated complexities and costs with integrating human feedback into foundation models. This includes difficulties in modeling rewards and establishing effective policies.
- **Industry Strategies and Model Collapse:** Current industrial practices, such as using larger models to judge the outputs of smaller models, were discussed. However, this strategy risks 'model collapse,' where the quality of the model degrades over successive training iterations.
- **Ad Hoc Nature of Current Practices:** The talk pointed out the ad hoc nature of approaches in software performance engineering within the context of foundation models, indicating a lack of systematic methodologies.

Foundation Model Hallucinations: Types and Mitigation

- **Understanding Hallucinations:** The concept of hallucinations was explained, distinguishing between intrinsic (unacceptable errors) and extrinsic (errors due to missing information) hallucinations.
- **Reducing Hallucinations Through Fine-Tuning:** The briefing emphasized that fine-tuning foundation models for specific tasks could help mitigate hallucinations. However, ensuring the safety and reliability of these systems remains an open challenge.

Rapid Evolution of Foundation Models and Collaborative Efforts

- **Unprecedented Pace of Foundation Model Development:** The speaker noted the extraordinarily rapid development in the field, driven largely by open collaboration and online sharing.
- **Historical Analogies for Safety Mechanisms:** Drawing parallels to historical developments like the implementation of seatbelts in vehicles, the need for similar safety mechanisms in foundation model was underscored, highlighting the importance of developing reliable and safe foundation model systems.

Key Terminologies and Concepts in Foundation Models

- **Exploring Foundation Model Parameters and Prompt Engineering:** A discussion on the misconceptions that larger foundation models are always superior was presented. The role of prompt engineering, (or more suitably called 'prompt hacking'), and techniques like React and Chain of Thought were explained.
- **Temperature Settings and Token Usage in Foundation Models:** The function of temperature settings in AI responses was elaborated, illustrating how higher temperatures can lead to more innovative and varied responses.

Debunking the Myth: Bigger Models Aren't Always Better

- **Challenges with Large Foundation Models:** The speaker discussed the limitations of larger foundation models, including their expensive training costs, increased latency, and potential for reduced effectiveness.
- **Emphasis on Data Quality and Model Efficiency:** A growing focus on improving data quality, efficient fine-tuning, and model optimization was highlighted, shifting the emphasis away from merely increasing model size.

Integrating Foundation Models into Software

- **Building Foundation Model Powered Software Systems:** The session aimed to delve into the challenges of incorporating foundation models into software systems.

Interactive Session: Questions and Discussions

- **Addressing Bias in Foundation Models:** The process of managing bias in foundation models, including the significance of fine-tuning and extensive safety testing, was discussed.
- **Handling Catastrophic Forgetting and Security Issues:** Concerns about foundation models forgetting previously-learned information and the associated security risks with foundation model generated content were raised. The need for robust mechanisms to safeguard data and ensure trustworthy foundation model systems was emphasized.

The comprehensive briefing provided an in-depth look at the current state and future possibilities of foundation models in software. It covered a range of topics from technical aspects of model training and data management to broader implications for the software industry, its practitioners, and stakeholders. The session underscored the transformative potential of foundation models in redefining software and software engineering practices and the role of developers in an increasingly foundation model powered world. The session concluded with an open call for further questions and discussions, signifying the complex interplay of challenges and opportunities

presented by the integration of foundation models in software. The breadth of topics covered reflects the multifaceted nature of foundation model development and its impact on software and software engineering practices.

Talk 1: FM+SE 101 Lectures (Part 2)

Key Highlights

Augmenting Foundation Models with External Knowledge

- **Knowledge-Intensive Reasoning Focus:** Emphasizing the necessity of supplementing foundation models with current or specific knowledge that the model's original training data may not include. This is crucial when dealing with outdated models or proprietary information that cannot be integrated into the model.
- **Retrieval-Augmented Generation (RAG):** A key method introduced was RAG, which involves augmenting a model's responses with external sources such as articles or database content. The method helps a model to access updated or specific information beyond its training data.
- **React Method for Dynamic Reasoning:** In contrast to RAG, React focuses on dynamic problem-solving by breaking tasks into smaller steps and seeking information as required. It avoids rigid, hardcoded reasoning and instead teaches the model to adapt and learn from various situations, akin to teaching it to 'catch anything' rather than just providing direct answers.

Tackling Challenges in Foundation Model Development

- **Overcoming Hallucination and Citation Needs:** The importance of providing accurate citations for information sources and integrating additional data was highlighted as a means to combat hallucination (inaccurate or misleading foundation model responses).
- **Managing Extensive Documents in Prompts:** Addressing the challenges posed by the limited size of prompts, especially when handling large documents. This often requires breaking down the information into several smaller prompts and using methods like self-consistency to identify the most common answers across these prompts.
- **Synchronizing Retrievers and Foundation Models:** There's ongoing work to synchronize foundation models with retrievers more effectively. However, such systems' complexity and cost remain high, posing challenges for their widespread deployment.
- **Innovations in Managing Document Inputs:** Introduction of new methodologies like HyDE to improve the ability to retrieve the most appropriate documents and Chunked-RAG to manage cases where the relevant documents are too large to fit in a single prompt.

Advanced Prompt Engineering and its Future

- **Complexities in Current Prompt Engineering:** The speaker highlighted the intricate and costly nature of current prompt engineering practices, appropriately called 'prompt hacking' as they lack systematic methodologies and require continual re-verification and tweaking.
- **The Emergence of Intent Compilers:** Looking ahead, the speaker emphasized the potential of 'intent compilers' to streamline and automate the process of prompt engineering, moving away from the current labor-intensive and fragile practices.
- **Introduction of New Prompt Engineering Tools:** Recent developments in the field, such as Google's Prompt Breeder and new programming models from Stanford, signal a shift towards more automated, efficient, and optimized prompting methodologies.

- **Relevance of Software Engineering Principles:** The briefing underscored the ongoing necessity for rigorous software engineering principles in the development and maintenance of foundation model systems to ensure their long-term effectiveness and sustainability.

Debugging and Evaluation of Foundation Models

- **Debugging in FMware Development:** Discussed the challenges in debugging prompts, where it's crucial to determine whether errors arise from the foundation model's misunderstanding, the prompt's inadequacies, or deficiencies in external data sources.
- **Comprehensive Evaluation Methods:** Covered various benchmarks for evaluating foundation models' effectiveness in generating code and responses, including HumanEval, CoderEval, and EvalPlus. These benchmarks aim to test foundation models against real-world scenarios.

Insights into Intelligent Software Engineering

- **Foundation Model Integration in Software Engineering Practices:** The briefing pivoted to discuss how foundation models can support various software engineering activities, focusing on foundation models as a collaborative tool rather than a replacement for human developers.
- **Highlighting Critical Challenges:** Identified critical areas such as requirement engineering, semantic telemetry, test-driven development, and performance engineering, emphasizing their importance in the context of FM-augmented software development.
- **Governance and Security in FMware:** Addressed governance and data security concerns, particularly in situations where foundation models handle sensitive or personal data.

Innovations in Software Engineering with Foundation Models

- **Microsoft India's CodePlan:** Integrates classical program analysis with a foundation model to understand software beyond just text. It considers classical SE elements like call graphs to provide a broader and richer context.
- **Copilot Workspace Release:** Marks a shift from traditional software development approaches. Unlike CodePlan, Copilot Workspace employs an agentic system where a foundation model-augmented agent interacts with the developer, seeking clarifications and approvals before execution. This reflects a more collaborative approach, contrasting with the unilateral approach seen in earlier models.

Challenges and Future Directions in FM-Integrated Software Development

- **Multi-agent Systems for Software Engineering:** Current models, such as 'ChatDev', aim to automate the entire software engineering cycle. However, they predominantly focus on greenfield (new) development, regenerating entire software systems for every run, which is impractical in real-world development scenarios.

- **Limited Software Engineering Knowledge in FM-augmented Agents:** Current FM-augmented agents lack comprehensive knowledge of software engineering principles like design patterns and architecture styles. They also don't improve over time, missing out on the maturity and career progress typically experienced by human developers.

- **Concept of Agent Gyms:** Future developments may see the creation of 'agent gyms' where FM-augmented agents, like basketball players practicing in courts, continually train and improve their capabilities.

- **MetaGPT Model:** Targets new developments but lacks support for repeatable results. Its interaction model is more linear (aka. waterfall) and doesn't reflect the flexible, iterative nature of modern software development processes.

Limitations and Risks of Current FM-Augmented Development Tools

- **High Risk for Junior Developers:** Junior developers might uncritically accept suggestions from FM-augmented tools due to lack of experience in turn leading to potential errors.

- **Necessity for Guardrails:** There's a need for mechanisms to guide FM suggestions, as users tend to favor responses from FM-augmented tools like ChatGPT, even when they're incorrect, due to their polished presentation.

- **Vulnerability and Licensing Issues:** FM-augmented tools tend to replicate the same vulnerabilities as human developers and have issues with complying with software licenses, creating legal risks.

- **Complexity in Debugging and Understanding FM-generated Code:** FM-augmented tools like Copilot have the risk of producing code that is difficult for developers to debug, debug and personalize for their use case, leading to inefficiencies.

Opportunities and Future of Foundation Model in Software Engineering

- **Reiteration of Key Messages:** The speaker reiterated the critical messages: larger models are not always superior, prompt engineering is currently a trial-and-error process, and the need for continuous verification and development of best practices in AIware.

- **Reflections on FMware Product Development:** The briefling concluded with thoughts on the challenges of developing trustworthy, commercially viable, and scalable FMware, highlighting the essential role of software engineering in making FM technologies robust and reliable for real-world usage.

- **Integrating Users in the Development Cycle:** Future foundation models could include end users in the software development process, customizing software releases based on user data and feedback.

- **Evolving Software Development Processes:** FM-augmented tools are expected to bring significant changes to the software development lifecycle, emphasizing collaboration between FM-augmented agents and human developers.

- **Incorporating Foundation Models in Software Engineering:** There is potential for using foundation models to enrich various aspects of software engineering, from requirement analysis to data tagging and testing.

- **Reimagining Software Engineering with Foundation Models:** The briefing concluded with the notion that foundation models are reshaping software engineering, requiring a

reevaluation of traditional practices and emphasizing the integration of foundation models' capabilities in various stages of the software development lifecycle.

The session provided an in-depth overview of the current state, challenges, and prospective directions for foundation models in software engineering and the impact of foundation models on software. The session underlined the importance of integrating time-tested software engineering principles with the latest advancements in foundation models, ensuring a robust and practical approach to the development and deployment of FMware.

Talk 3: Beyond Code - Helping Software Teams with LLMs (Microsoft)

Overview

- **Presenters:** Tom Zimmermann and Rob DeLine from Microsoft Research.
- **Objective:** The speakers discussed the integration of Large Language Models (LLMs) in enhancing the dynamics of software teams, focusing on applications beyond code completion.

Key Highlights

Research Themes and Background

- **Focus on Software Teams:** The researchers have been studying software team dynamics in large companies. This includes understanding their information needs and collaboration patterns.
- **Data-Centric Research Approach:** Emphasis on incorporating data science and foundation models in software development. They conducted detailed studies on data scientists at Microsoft and developed analytics systems specifically for software development.
- **Adaptation During COVID-19:** Their research direction adapted to explore remote working dynamics during the pandemic.

FM-Augmented Development Tooling

- **Emergence of FM-augmented Tools:** Highlighted the significant impact of tools like GitHub Copilot on their research, leading to new studies on how these tools reshape software development practices.
- **Development of Trust Framework:** Introduced the PICSE framework, focusing on building trust in software tool recommendations, highlighting the importance of user confidence in particular in AI solutions.

The Dream Team Vision

- **Concept of Human-FM Collaboration:** Proposed a model where FM experts collaborate with human teams, offering specialized knowledge and support in various aspects of software development.
- **Key FM Roles Identified:** Identified three main FM-enhanced roles - consulting experts, data scientists, and tech leads, each catering to specific needs within software teams.

Prototypes Demonstrated

- **FM Consulting Expert Prototype:** Aimed at assisting teams in goal-setting, identifying best practices within the organization, and guiding metric identification.
- **Tech Lead Expert Prototype:** Focused on helping new team members navigate large codebases, acting as an FM mentor within IDEs.
- **Data Science Expert Prototype:** Designed to facilitate replication of research studies on team-specific datasets, providing methodologies and analysis steps from academic papers.

System Design and Challenges

- **Orchestration of Multiple Agents:** Emphasized the interconnectedness of various agents working in unison to process inputs and generate collaborative outputs.
- **Operational Challenges:** Discussed challenges in evaluating foundation model systems, creating robust architectures, handling model inconsistencies, and managing limited context windows in models.

Insights and Best Practices

- **Systematic Prompt Engineering:** Stressed the need for a systematic approach to creating and refining prompts, including rigorous testing beyond initial successes.
- **Selective Model Utilization:** Discussed using different GPT models based on task requirements and strengths.
- **Error Handling:** Focused on anticipating and managing model failures with robust coding strategies.
- **Utility-Centric Evaluation:** Shifted focus from accuracy to the utility and productivity enhancements provided by FM tools.

Research Approach and Future Directions

- **Strategic Problem Selection:** Highlighted selecting unique research problems by leveraging team-specific strengths in expertise, methodology, and context.
- **Ambition in Research:** Emphasized choosing challenging and non-obvious research questions, accepting inherent risks in pursuit of innovative solutions.
- **Adapting Research Methods:** Discussed continuously adapting research methods in response to the rapidly evolving AI and foundation model landscape.

Q&A Insights

- **Management of Leadership Expectations:** Discussed balancing initial enthusiasm with practical, robust solutions in management presentations.
- **Perceptions of AI in the SDLC:** Addressed skepticism about the use of generative AI for non-coding activities in the software development lifecycle.

The session provided an in-depth look into Microsoft Research's efforts in integrating foundation models into various aspects of software development. It underscored the potential of foundation models to not only assist in coding but also to significantly enhance the broader aspects of software teams, fostering a collaborative ecosystem between AI and human expertise.

Talk 4: Large Sequence Models for Software Development Activities (Google)

Overview

- **Presenter:** Petros Maniatis, a Research Scientist at Google DeepMind.
- **Objective:** Discussing the application of large foundation models in software engineering, based on Google's detailed automated tracing of software engineering activities.

Key Highlights

The Concept of Software Engineering as a Holistic Journey

- **Overview:** Software development is portrayed not just as a coding task but as a comprehensive journey with multiple interactions and steps.
- **AI Integration:** Emphasizing the necessity for AI assistants to be integrated into the entire software development process, recognizing their multiple benefits beyond coding.

Leveraging Internal Software Engineering Logs

- **Data Source:** Leveraging Google's in-depth logs of software engineering activities as a rich data source for training models.
- **Collaborative Initiative:** The project involves collective efforts from teams across DeepMind, Google Brain, and Google's core developer division.

Introduction to the DIDACT FMware

- **Model Description:** DIDACT is an FMware that is developed to assist in various tasks like code review/editing and build repair.
- **Training Methodology:** The model uses a unique approach involving the state, intent, and action format for training examples, aligning with text-to-text transformer model architectures.

Observing Emergent Capabilities in DIDACT

- **Predictive Skills:** Showcased examples where DIDACT predicts documentation updates and code modifications in response to code changes.
- **Adaptive Learning:** The model's ability to learn from past code changes and predict future actions, adapting to both code additions and deletions.

Practical Implementation and Applications

- **Tools Based on DIDACT:** Various internal Google tools, such as those for code review resolution and build error fixing, are built by leveraging DIDACT.
- **Development Process:** Details the iterative development and enhancement of these tools, focusing on model quality and usability improvements based on practical user feedback.

Insights into Model Tuning and User Engagement

- **Fine-Tuning Techniques:** Explores how the model was specifically fine-tuned for tasks like code review resolution, including adjustments in precision thresholds based on user feedback, and language-specific model tuning.

- **User Engagement Enhancements:** Discusses the significance of user interface design and proactive suggestions. Details how visibility and accessibility of suggested edits were improved, leading to higher user acceptance and engagement.

Involving Reviewers in FM-Augmented Development

- **Reviewer Involvement:** Highlights the critical role of code reviewers in the FM-augmented process, allowing reviewers to preview and validate FM-generated suggestions.
- **Building Trust:** This approach not only serves as a quality check but also boosts users' trust in FM suggestions, enhancing the tool's overall credibility and acceptance.

Evaluating Productivity Impacts

- **Productivity Measurement:** Discusses Google's strategies for assessing the productivity impacts of AI tools, including comparisons of coding speeds between groups with and without access to the tools.
- **Reported Efficiency Gains:** Significant improvements in coding speed and workflow efficiency were observed, showcasing the effectiveness of AI tools in enhancing software development processes.

The talk emphasized the importance of AI tools learning from and being integrated into the full cycle of software development activities. Maniatis expressed enthusiasm about expanding these methodologies to a wider range of settings, including more programming languages and tasks, highlighting the dynamic and evolving role of AI in software engineering.

Talk 5: Open and Responsible Development of Large Language Models for Code (ServiceNow)

Overview

- **Presenter:** Raymond Li from ServiceNow, representing a collaborative project with numerous contributors.
- **Objective:** Discussing "BigCode" focusing on the development of large language models for code, emphasizing open and responsible methodologies.
- **Context:** The project operates under the ethos of "big science", fostering collaboration across various entities, including universities and industry. Over 1,000 members from 60+ countries are involved, contributing ideas and participating in discussions.

Key Highlights

Non-Technical Considerations and Governance

- **Data Consent and Privacy Risks:** The project addresses the ethical use of third-party copyrighted data, highlighting the need for appropriate licenses and considering privacy implications due to the presence of personal information in public data sources.
- **Software Safety and Security Concerns:** Emphasizes the necessity to consider the potential misuse of models for generating insecure code or malware.
- **Governance Card:** Developed to document various non-technical aspects of the project, ensuring a comprehensive understanding of the implications and responsibilities.

Openness in Research and Development

- **Challenges of Closed Development:** Closed development of large language models hinders scientific reproducibility and limits understanding of model behaviors.
- **Advantages of Open Research:** Advocates for open and transparent research to allow community contributions and enhance user trust.
- **BigCode Project's Approach:** Emphasizes transparency by releasing datasets, models, and detailed documentation, promoting responsible research practices.

The Stack: Dataset Collection

- **Data Size and Collection:** The project gathered 6.4 terabytes of permissively licensed source code, starting with 220 million repository names from GitHub archives.
- **Data Processing Steps:** Involved extensive data processing, including license filtering, deduplication, and language selection, to refine the dataset to a manageable and usable size.
- **Data Inspection and Opt-Out Tools:** Introduced tools for users to check if their repositories are part of the dataset and provided options to opt-out for greater control and transparency.

Model Training Process

- **Training Data Details:** The training data comprised of 800 gigabytes of code across 86 programming languages, supplemented with GitHub issues, commits, and notebooks.
- **Scaling Laws and Model Size Decision:** Leveraged scaling laws to balance model size and training data size, considering both training and inference compute costs.

- **Model Architecture Overview:** Adopted a classic decoder-only transformer architecture, incorporating multi-query attention and flash attention for memory efficiency and increased throughput.
- **Training Setup and Resources:** The model was trained over approximately one month using 500 GPUs, employing tensor, pipeline, and data parallelism for effective distributed training.

StarCoder: The Released Foundation Model

- **Model Variants:** Introduced various versions of StarCoder, including specialized versions focused on Python and enhanced natural language training data.
- **Community Engagement:** Acknowledged significant contributions and enhancements made by the open-source community using the Stack and the StarCoder model.

Evaluation and Future Work

- **Benchmarking and Evaluation:** Leveraged benchmarks such as HumanEval and DS1000 for model evaluation, providing a measure of model performance and capabilities.
- **VSCode Extension Development:** Highlighted the creation of a VSCode extension for StarCoder, featuring a code generation verification tool to spot whether the generated code was part of the training data set.
- **Anticipation of StarCoder 2:** Announced ongoing work on the next iteration of StarCoder, indicating continuous development and improvement.

Interactive Questions and Discussions

- **Concerns on Data Cleaning and Vulnerability:** Addressed inquiries about strategies to filter vulnerable code and the trade-offs between data quantity and quality.
- **Reinforcement Learning in Model Development:** Explored the potential role and impact of reinforcement learning in enhancing foundation model's performance and alignment, particularly in code generation contexts.
- **Data Quality Filtering Strategies:** Discussed approaches for identifying and removing low-quality code data, using heuristics and manual inspection to ensure the training on high-quality datasets.

The talk concluded with an emphasis on the importance of balancing open research with responsible practices in developing foundation models for code. The ongoing efforts to improve models like StarCoder were highlighted, showcasing the project's commitment to continuous development and innovation.

Talk 6: CodeCompose - A Large-Scale Industrial Deployment of AI-assisted Code Authoring (Meta)

Overview

- **Presenters:** Chandra Maddila and Vijayraghavan Murali from Meta.
- **Objective:** Discussion centered on CodeCompose, Meta's AI-assisted code authoring solution. The talk aimed to elucidate the necessity, development, deployment, and integration of CodeCompose in Meta's workflow.

Key Highlights

Necessity of In-House AI-Assisted Code Authoring

- **Custom Frameworks at Meta:** Highlighted the unique programming languages and frameworks developed at Meta.
- **Need for Trust and Data Control:** Emphasized controlling data within the company for maintaining trust and security.
- **Custom Training Objectives and Integration:** Addressed the need to tailor foundation models to specific user experiences across various coding platforms at Meta.
- **Telemetry and Feedback:** Stressed the importance of comprehensive logging for continuous evaluation and improvement of the system.

Overview of CodeCompose

- **Functionality:** Assists in writing code, recommends best practices, and aids developers in navigating unfamiliar codebases.
- **Deployment:** Made available to all developers at Meta, supporting a wide array of languages and integrated into multiple IDEs and platforms, including their notebook solution, Bento.

Model Training: The Data Pyramid Concept

- **Training Approach:** Leveraged a pyramid model focusing on quantity at the base (third-party data) and quality at higher levels (first-party data).
- **Base Layer:** Large-scale third-party data used for initial pre-training.
- **Middle Layer:** Refined training with first-party high-quality data.
- **Top Layer:** Advanced training techniques for custom behavior modeling.

Development and Evolution of the Model

- **Model Progression:** Transitioned from a 1.3 billion parameter model to a 6.7 billion model, eventually adopting a 7 billion parameter model from Code Llama.
- **Custom Training Objective:** Developed a unique language causal masking objective tailored to Meta's specific coding scenarios.
- **Model Selection Criteria:** Picked a 7 billion parameter model for balancing accuracy and response times.

System Design and Deployment

- **Infrastructure and Deployment:** Set up a distributed training system, storing model checkpoints for global access. Deployed an inference service in various data centers.
- **Client-Side Interaction:** Implemented a language server protocol to integrate seamlessly with multiple IDEs.
- **Performance Optimization:** Prioritized achieving rapid response times and maintaining high accuracy in suggestions.

Evaluation and User Feedback

- **Offline Evaluation:** Assessed the model's ability to generate sections of hidden code, showing improvement over baseline models.
- **Online Evaluation Metrics:** Tracked developer acceptance rates and the extent of CodeCompose's contributions to their code.
- **Qualitative Feedback:** Predominantly positive responses focused on the tool's utility in speeding up coding, discovering new methods, and improving code documentation.
- **Constructive Feedback:** Suggestions for improvement included reducing hallucinations and better integration with traditional autocomplete systems.

Challenges and Learnings

- **Trust and Safety Challenges:** Focused on ensuring that the code suggestions were safe, complied with privacy standards, and were free from vulnerabilities.
- **User Experience Balancing:** Tackled the challenge of adjusting model suggestions to be useful without being intrusive and maintaining quick response times.
- **Comprehensive Impact Evaluation:** Employed a range of metrics to gauge effectiveness, avoiding reliance on single metrics.
- **Future Prospects:** Plans to expand beyond coding suggestions to more comprehensive applications in the software development lifecycle.

The talk demonstrated CodeCompose's effectiveness in aiding developers with coding, with future plans for broader applications within software development processes.

Talk 7: Towards Better Software Quality in the Era of Large Language Models (UIUC)

Overview

- **Presenter:** Lingming Zhang from the University of Illinois Urbana-Champaign (UIUC).
- **Objective:** Discusses the application of large language models (LLMs) to enhance software quality, focusing on testing, repair, and synthesis; and discusses how LLMs impact software development tasks, and whether they can significantly improve software quality in practical settings.

Key Highlights

Background on Language Models

- **Core Function:** Assigning probabilities to token sequences.
- **Architecture:** Based on transformer architecture, emphasizing the importance of the attention mechanism.
- **Training and Capabilities:** Trained on large volumes of unsupervised data (like GitHub and Wikipedia), allowing them to perform complex reasoning tasks and store extensive knowledge. Notable examples include models like GPT-4 with over a trillion parameters.

Impact of LLMs on Software Quality

- **Central Question:** Examining the real-world performance of LLMs in improving software quality.
- **Approach:** Prof. Zhang reviewed how traditional tasks in software development could be re-envisioned with the integration of LLMs.

Fuzz Testing with Language Models

- **Concept and Importance of Fuzzing:** Automatic generation of system inputs/tests to uncover bugs and vulnerabilities. Fuzzing's simplicity but high impact in practice is highlighted.
- **Challenges in Deep Learning Library Fuzzing:** Traditional fuzzing approaches struggle due to Python's dynamic nature and the intricate tensor computation constraints in deep learning libraries.
- **Language Model-Based Fuzzing Approach:** Proposes using generative and infilling language models to create valid and diverse programs for testing deep learning libraries like TensorFlow and PyTorch.

Testing Deep Learning Libraries

- **Methodology:** Combines generative and infilling models for generation and mutation-based fuzzing.
- **Results and Impact:** Over 60 bugs found in real-world systems like PyTorch and TensorFlow, with 55 confirmed as genuine. Demonstrates LLMs' effectiveness in domains where traditional techniques struggle.

Automatic Program Repair (APR)

- **Overview of APR:** Extensively studied field with recent focus on neural machine translation techniques.
- **Alpha Repair Concept:** Reformulates the repair problem as an infilling task, employing LLMs to generate correct code based on the naturalness of software.
- **Advantages:** Overcomes the need for bug fixing datasets, improves context capture, supports multilingual repair, and achieves state-of-the-art results on standard benchmarks.

Enhancing LLM Effectiveness in APR

- **Challenges with Rare/Domain-specific Tokens:** Addresses the difficulty in generating patches with rare or domain-specific tokens.
- **Strategies:**
  - **Plastic Surgery Hypothesis:** Utilizes existing tokens in a project for generating patches, reducing the need for external data.
  - **Copilot Models:** Merges LLMs with traditional code completion engines to handle domain-specific elements and rare tokens.

Program Synthesis and Evaluation

- **HumanEval Dataset and Its Limitations:** Identifies shortcomings in popular datasets used for evaluating model-generated solutions.
- **Evolve Plus Approach:** Combines ChatGPT with traditional program analysis techniques to create new tests, leading to a more rigorous evaluation of model solutions.
- **Findings:** Noticeable performance degradation (around 15%) in popular models under more stringent testing.

Future Research Directions

- **Synergy Between Humans and LLMs:** Explores potential collaborations and unifying tasks in software development.
- **Data Leakage and Semantic Equivalence Issues:** Focuses on addressing these challenges in program repair and synthesis.
- **Byte-Level Analysis for Unstructured Data:** Investigates LLM applications in domains like AFL-style fuzzing where inputs are less structured.

Call for Contributions

- **Invitation:** Announces the first international workshop on LLMs for code, co-located with ICSE2024, inviting submissions from the community.

The talk by Lingming Zhang provided a comprehensive overview of how Large Language Models can revolutionize tasks in software development, underlined the potential of LLMs in enhancing software quality, presenting new methodologies, challenges, and opportunities for future research in the field.

Talk 9: Building Trustworthy Foundation Model Applications (Huawei Canada)

Overview

- **Presenter:** Dayi Lin, Huawei Canada
- **Objective:** The development process of foundation model applications, focusing on decision-making points and aspects of trustworthiness. The talk addresses challenges and raises questions, aiming to spark future research in the field.

Key Highlights

Building Application: Traditional vs. Foundation Models

- **Traditional Process:** Idea generation followed by finding a programmer or self-development.
- **Change with Foundation Models:** Individuals without technical skills consider using foundation models (like GPT) for development.
- **Impact:** Shift from a limited number of developers (29 million) to the entirety of humanity (8 billion) potentially creating applications, democratizing software development.

Software Evolution: Codeware to Neuralware

- **Transition:** From Codeware and Neuralware to Promptware and Agentware.
- **Developer Profile Shift:** From software makers to non-technical domain experts or prompt experts.
- **Enterprise Involvement:** Businesses are also exploring application development with different considerations compared to individual users.

Model Selection Process

- **Key Decision:** Choosing the right model (size, hosting method, and API costs).
- **Considerations:** Larger models are costlier and not always better. Smaller, fine-tuned models can match the performance of larger ones.

Fine-Tuning Foundation Models

- **Approaches Range:** From zero-shot prompting to supervised fine-tuning and reinforcement learning.
- **Data Set Quality:** Importance of high-quality datasets for effective fine-tuning.
- **Industry Methods:** Manual labeling, semi-automatic solutions, fully automatic methods, and their limitations.

Prompt Development Challenges

- **Prompt Sensitivity:** Writing effective prompts is non-trivial and sensitive to textual changes.
- **Prompt Optimization:** Emerging as a necessary step due to evolving best practices and model differences.

Context Engineering in Foundation Models

- **Addressing Hallucinations:** Introducing various data types (e.g., interaction history and external data) to enrich responses.
- **Retrieval Augmented Generation Challenges:** Balancing information retrieval with considerations for text truncation and overlap.

Debugging and Explainability in Foundation Models

- **Debugging Challenges:** Identifying whether issues are due to the prompt or model's understanding.
- **Explainability Importance:** Implementing token-level and component-level explainability for better understanding of model responses.

Security and Compliance in Application Development

- **Preventing Prompt Injection:** Implementing pre- and post-generation safeguards.
- **Ensuring Compliance:** Maintaining adherence to regulations and privacy policies.

Orchestration of Multi-Agent Systems

- **Facilitating Agent Interaction:** Overcoming challenges in multi-agent systems.
- **Control Mechanisms:** Developing systems to ensure effective and compliant agent interaction.

Quality Assurance Challenges

- **Difficulties in QA:** Quantifying the quality of generative outputs is challenging.
- **Approaches in QA:** Human evaluation, semi-automatic methods, or using models like GPT-4 for grading.

User Interaction and Integration

- **Interaction Approaches:** Dialogue-based or integration into existing software.
- **Transparency and Human Oversight:** Communicating limitations and incorporating human evaluation in decision-making processes.

Deployment and Observability

- **Monitoring Applications:** Importance of logging, tracing, and monitoring.
- **Model Caching:** Using caching to enhance performance and reduce costs.
- **Live Experiments:** Employing traditional testing methods for continuous application improvement.

Trustworthiness Dimensions

- **Considerations:** Emphasizing transparency, explainability, security, compliance, and quality assurance throughout the development process.

The talk showcased how to successfully build a trustworthy foundation model application involving a comprehensive understanding of architecture and trustworthiness considerations, provided an extensive overview of the complexities and considerations in developing trustworthy

foundation model applications, emphasizing the importance of security, explainability, and quality assurance in the evolving landscape of FM and software development.

Talk 10: The Legal Landscape of ML/FM Systems (University of Victoria)

Overview

- **Presenter:** Daniel Germán, a specialist in intellectual property and its influence on software development, particularly in open-source software.
- **Objective:** Germán is not a lawyer but is informed about the legal aspects of FM systems and software. He focuses on providing insights from original sources rather than secondary reports, highlighting differences in legal frameworks across countries.

Key Highlights

Overview of Intellectual Property

- **Types of Intellectual Property:** The talk outlines four main types of intellectual property protections relevant to FMs and software development:
  1. **Copyright:** Protects original works of authorship, particularly applicable to software. It automatically applies upon creation in most jurisdictions.
  2. **Trademark:** Involves the protection of symbols, logos, and distinctive marks identifying products or services.
  3. **Trade Secrets:** Covers information that is kept confidential within a company or entity, protected as long as secrecy is maintained, usually through legal agreements like NDAs.
  4. **Patents:** Grants exclusive rights over inventions, a contentious area in software where patents have been debated but are now generally accepted.

Data Ownership and Its Importance

- **Data as the Crux of FMs:** Emphasizes that without data, FMs are effectively valueless. Data owners are increasingly aware of their data's worth and seek a share of the profits that are generated from FMs using their data.
- **Ownership Rights:** Discusses how data owners have copyrights or other forms of legal protection over their data, which grants them control over how their data is used and duplicated.
- **Market Impact:** Illustrates concerns about FMs impacting the market of data owners, especially when FMs use existing data to generate new content, potentially reducing the demand for original content.

Legal Challenges and Case Studies

- **Getty Images vs. Stability AI:** A case where Getty Images sues Stability AI for copyright infringement due to the alleged use of its photographs in training AI models.
- **Doe Against GitHub:** Focuses on accusations against GitHub for using copyrighted material in its AI tools, Codex, and Copilot, potentially including code under restrictive licenses.
- **Sarah Silverman vs. Stability AI:** Involves copyright infringement claims related to derivative works created by Stability AI's generative tools.

Ownership of FM-Generated Content

- **Legal and Contractual Perspectives:** The terms of service of AI platforms like OpenAI and Mid-Journey differ significantly in terms of ownership rights over FM-generated content. In some cases, the user retains ownership, while in others, the platform claims rights.
- **Jurisdictional Variations:** The legal treatment of FM-generated content varies by country. For instance, in the U.K., FM-generated content is owned by the operator of the FMs, while the U.S. does not grant copyright to FM-generated creations.

Legal Opinions and Recommendations

- **Complex Legal Landscape:** Germán discusses the intricate legal landscape surrounding FMs, highlighting the uncertainty and the need for adaptive strategies.
- **Anticipating Legal Changes:** Predicts that laws will evolve to accommodate FM's role in society, potentially through mechanisms that balance the interests of data owners and FM developers.
- **Personal Views:** Shares his perspectives on various legal cases, indicating the potential significance of cases like Getty Images vs. Stability AI in shaping the future legal framework for FMs.

Challenges

- **Navigating Legal Challenges:** The talk concludes with a recommendation to stay informed and prepared for the changing legal landscape in FMs. Germán stresses the importance of understanding the terms of use of FM tools and the implications for data owners, trainers, operators, and users of FM systems.

This comprehensive summary of Daniel Germán's talk delves into the intricate legal landscape surrounding FM and software engineering, focusing on intellectual property issues, data ownership rights, and the diverse legal challenges and cases across different jurisdictions. His insights illuminate the complexities and evolving nature of legal interpretations in FMs, underscoring the need for awareness and adaptability in this dynamic legal environment.

Panel A - Impact of Foundation Models on Software

Panelists

- **Zhen Ming (Jack) Jiang:** York University
- **Rob Deline:** Microsoft Research
- **Ipek Ozkaya:** Carnegie Mellon University, Software Engineering Institute
- **Charles Sutton:** Google
- **Prem Devanbu:** UC Davis

Discussion Highlights

- **The Future of Software as a Foundation Model:** The discussion initiated with contemplation about a future where traditional coding may become redundant, replaced by software as a foundation model. The idea of instructing foundation models to behave like specific applications, such as a web browser, was proposed.
- **Rob Deline's Perspectives on Foundation Models Replacing Apps:** Rob Deline expressed doubts about foundation models wholly supplanting apps. He argued that it's easier to choose ready-made software (akin to shopping) than to specify exact requirements for a foundation model. Additionally, he highlighted the social aspect of software, like in gaming and social media, where shared experiences are integral.
- **Comparative Analogy: Radio, TV, and the Internet:** Another panelist drew a parallel between the coexistence of radios, TVs, and the internet to suggest that traditional software and foundation models would likely coexist. They mentioned that while foundation models would be more prevalent in tasks involving exploration or recurrent tasks, traditional software would remain relevant in other areas.
- **Autonomous Driving as an Example Application of Foundation Models:** The panel discussed the application of foundation models in autonomous driving, envisioning each car as an autonomous agent and the collective as a societal system for optimized traffic flow and safety.
- **Challenges in Software Engineering Research and FM Dominance:** Concerns were raised about the overwhelming influence of FMs in software engineering research, leading to the obsolescence of some traditional research topics. The necessity of adhering to foundation software engineering principles in the era of FM was emphasized.
- **Skepticism and Trust Issues with Foundation Models:** The panel expressed skepticism about the trustworthiness and reliability of foundation models, particularly in critical applications like defense systems. Historical examples were cited, like BlackBerry's initial dismissal of the iPhone, to illustrate the unpredictability of technology adoption and the threshold of trust that end-users are willing to adopt.
- **Human-Computer Interaction in Software Development:** The role of human interaction in software development was highlighted, with a focus on conversational software and the potential shift from a directive to a more interactive, conversational model in software engineering.
- **Legacy Systems and Maintenance Challenges:** The issue of maintaining legacy systems, such as those written in COBOL, was discussed. The potential of FM-augmented agents in performing maintenance tasks on these systems was explored, raising questions about the beginning of a transition to FM-powered systems.

- **Balancing Trust and Convenience in Software Usage:** The balance between trust in software and the convenience it offers was debated, with emphasis on the criticality of establishing trust, especially in tools that interface with foundation models.
- **Vision for Software Engineering and Foundation Models:** Ipek Ozkaya emphasized the importance of visionary thinking in software engineering, acknowledging the areas where foundation models might not be suitable. The need for reliable data and continuous updating of these models was stressed.
- **Charles Sutton's Dual Argument on Foundation Models:** Charles Sutton presented a dual argument: he acknowledged that foundation models have already started replacing certain software functionalities, but also argued that the idea of completely replacing traditional software is far-fetched. He outlined challenges in specifying programs, verifying them, and debugging, which remain unsolved in the context of large language models.
- **Compilers in the Future of Software Engineering:** Towards the end, the panel speculated on the future of compilers in an era dominated by foundation models. While there was a consensus that traditional compilers would still exist, they might become less central, akin to the role of assemblers today.

This panel summary encapsulates the diverse viewpoints and insights shared by the panelists, illustrating the multifaceted impact of foundation models on the future of software engineering. The discussion navigated through skepticism, practical challenges, and visionary thoughts, painting a complex yet intriguing picture of what lies ahead in the intersection of FM and software development.

Panel B: Impact of Foundation Models on Developer Productivity

Panelists

- **Edward Aftandillian:** Involved at GitHub Next, Aftandillian's work centers on developing AI-powered tools like Copilot. His role involves exploring and innovating future applications of FM in software development.
- **Foutse Khomh:** Based at Polytechnique Montreal, Khomh's focus is on the engineering aspects of AI-based systems, emphasizing the trustworthiness of AI-powered software.
- **Peggy Storey:** From the University of Victoria, Canada, and a visitor at the University of Zurich, Switzerland. Storey's research interests are the human and social aspects of software engineering, particularly how generative AI influences the way developers work and collaborate.
- **Hironori Washizaki:** A researcher from Waseda University, Tokyo, Japan and the president of the IEEE Computer Society. Hironori's research encompasses AI for software engineering and vice versa. He is instrumental in the evolution of the SWEBOK Guide, focusing on integrating software engineering and AI into the formal SE body of knowledge (aka SWEBOK).
- **Priyanshu Gupta:** As an associate researcher at Microsoft's PROSE team, Priyanshu's work on collaborative AI agents is directed towards boosting developer productivity, with involvement in projects like VS Copilot.

Discussion Highlights

- **Integration and Impact of Foundation Models (FMs):** The panel discussed how FMs have been successfully integrated into Integrated Development Environments (IDEs), notably enhancing productivity in code completion tasks. This integration symbolizes a significant stride in making coding more efficient and accessible.
- **The Role of FMs Beyond Coding:** A critical discussion point was the potential of FMs to extend their utility beyond mere coding. This includes aiding in decision-making processes, planning stages, and comprehending the broader context and requirements of projects, marking a shift from traditional software development practices.
- **Evolving Developer Roles with FM Integration:** The dialogue highlighted how the introduction of FMs in software development is leading to a transformation in the role of developers. There is an observable shift towards tasks such as prompt engineering, managing FM-generated components, and refining FM outputs, indicating a new direction in the required skills for modern software development.
- **Challenges and Considerations in FM Integration:** The panel acknowledged various challenges in integrating FMs into software development. These include issues like inherent biases in foundation models, the need for consistent human oversight to ensure quality and relevance in FM-generated code, and understanding the psychological impacts of FM integration on software developers.
- **Future Skills and Education for Developers:** The conversation touched upon the evolving skill set that is required for future developers in an FM-augmented landscape. Emphasizing the importance of understanding FM outputs, the discussion pointed out the need for foundation software engineering knowledge to remain relevant and effective in an FM-augmented development environment.

- **FM's Role in Early Development Phases:** There was significant interest in discussing how FMs might contribute in the early phases of software development, such as requirements engineering and architectural design. The idea that FM's role could expand beyond mere coding to more strategic and planning-oriented tasks was explored.
- **Measuring FM's Impact on Productivity:** A complex and multi-faceted topic, measuring the impact of FMs on developer productivity, brought forth discussions on the need for diverse and nuanced metrics. The panelists called for in-depth research methods to accurately gauge FMs' contributions and effects in various aspects of software development.
- **Implications of FMs on Software Engineering Education:** The panelists pondered over the implications of FMs rise on software engineering education. There was a consensus on the need for future software engineers to learn how to assess and comprehend FM-generated code, signaling a shift in educational curriculums and teaching methodologies.
- **Management Perspectives in FM-Augmented Development:** From a managerial standpoint, the panel addressed concerns about leveraging FM to enhance team productivity while maintaining healthy team dynamics and individual contributions. This discussion highlighted the need for new management strategies in FM-augmented software development environments.
- **Developer Experience and Team Collaboration:** The panel explored how FM-augmented tools could potentially impact developers' sense of flow, satisfaction, and the nature of their collaboration within teams. There was a focus on the balance between FM-augmentation/assistance and the intrinsic satisfaction that developers derive from coding and problem-solving.
- **Consumer Preference and Software Quality:** A thought-provoking question was raised about whether, in the future, consumers might prefer artisan software that is developed by humans over AI-generated code. This led to discussions about the potential for a niche market for human-crafted software, similar to the preference for handmade products in other industries.

Additional Insights and Anecdotes

- **Edward Aftandillian** shared his personal experience with AI tools, noting how they assist in mundane tasks and contribute to maintaining a flow in development work.
- **Peggy Storey** expressed concerns about the long-term impact of AI on developers' emotional connection to their work and the potential loss of satisfaction from hand-crafted code.
- **Hironori Washizaki** emphasized the importance of understanding the psychological aspects of developers working with AI, including the dynamics of human-AI pair programming.
- **Priyanshu Gupta** talked about the initial skepticism and eventual acceptance of conversational AI systems in development processes.

Important Dates

- The release of the latest SWEBOK Guide, which addresses the integration of SE and AI, is anticipated soon. The exact date was not provided.

The panel concluded with an interactive Q&A session, where the panelists addressed diverse topics related to the future of FM in software development, the evolving role of developers, and the challenges of managing teams in an FM-augmented development landscape.

Panel C: SE Challenges for Foundation Models

Panelists

- **Zhenming (Jack) Jiang:** From York University in Canada, introduced the panel focusing on the complexities of Foundation Models (FMs), particularly Large Language Models (LLMs), and the operational challenges (FMOps or LLMOps) in quickly deploying these models.
- **Raymond Li**: A research engineer at ServiceNow, specializing in large language models for code, particularly the Star Coder project.
- **Petros Maniatis:** A research scientist at Google DeepMind, working on internal models at Google and broader software engineering activities, including early work on BERT for code.
- **Vincent Hellendoorn:** With a background in training and deploying large language models in the open-source space, Hellendoorn is currently at Google, exploring various applications of large language models for coding.
- **Boyuan Chen:** From Huawei Canada, Boyuan's work centers on using heterogeneous resources to enhance the deployment of foundation model applications (FMware).
- **Kensen Shi:** At Google DeepMind, Shi's interests lie in leveraging code language models throughout the software development process and improving these models.

Discussion Highlights

These detailed discussion highlights from Panel C provide a comprehensive overview of the key points and complexities discussed by the panelists regarding software engineering challenges in the context of foundation models, especially large language models.

Data Management in Foundation Model Operations

- **Complexity and Sensitivity**: Panelists emphasized the intricate nature of data management, noting challenges like sensitivity, licensing, and provenance, especially when dealing with diverse data sets.
- **Provenance and Retraining**: Importance was placed on data provenance due to the need to potentially remove certain data in subsequent training iterations.
- **Pipelines and User-Generated Data**: Managing pipelines for these models is complex due to the influx of vast user-generated data, posing unique challenges in data cleaning and processing.

Training Models: Costs and Efficiency

- **Cost Concerns in Training**: A significant challenge mentioned was the high cost of training these models. The panel highlighted the financial implications of errors in the training process, especially when wrong data sets are used.
- **Efficiency in Training**: The panel discussed recent advancements like FlashAttention that can improve training efficiency. This is crucial given the expensive nature of training large models.
- **Evaluating Model Performance**: Evaluation of models was a point of concern, with current benchmarks being seen as possibly not realistic enough for real-world applications. The need for more robust evaluation methods was stressed.

Versioning and Model Compatibility

- **Rapid Version Changes**: Panelists brought up the challenge of managing versions of models, noting the quick pace of updates and changes in model versions.
- **Checkpoint-based Versioning**: The practice of checkpoint-based versioning was discussed, which points to specific checkpoints in a model's training process but poses challenges in terms of consistency and reliability.
- **Model Dependency and Evaluation**: The dependency on extensive evaluations for every new model version was highlighted, considering this as both costly and time-consuming.

Proliferation of Models and Managing Policies

- **Model Lineage and Tracking**: With the increasing number of models, tracking their lineage and understanding their data sources becomes complex.
- **Diverse Training Data and Policies**: The panelists discussed the difficulties in managing different policies for models trained on varied types of data, emphasizing the importance of understanding model-specific policies and guidelines.

Operational and Hardware Challenges

- **Faulty Hardware Detection**: Issues such as detecting faulty chips in the hardware used for training these models were discussed, noting the nondeterministic nature of some hardware faults.
- **Silent Data Corruption**: The problem of silent data corruption in large-scale cloud infrastructures was mentioned, highlighting the challenges it poses for training large models.
- **Optimization and Compilation**: The need for effective optimization and compilation in utilizing hardware resources was discussed, considering the specific challenges posed by the deployment and training of these models.

Quality of Data and Use of Synthetic Data

- **Balancing Quality with Synthetic Data**: Panelists debated the merits and risks of using synthetic data in training, with concerns about model collapse due to poor-quality synthetic data.
- **Training on Expert vs. Novice Code**: The discussion touched on whether models should be trained on expert-level code for higher quality or on novice-level code to cater to a broader range of users.

Benchmark Creation and Future Predictions

- **Need for Realistic Benchmarks**: There was a consensus on the need for more challenging and realistic benchmarks that would test the limits of current foundation models.
- **Contribution of Academic Consortiums**: The idea of academic consortia contributing to the creation of better benchmarks was proposed, emphasizing the need for collaborative efforts to overcome resource limitations.

Audience Interaction

- Questions from the audience centered on data quality and benchmark creation. One question probed into the practical methods for creating cost-effective benchmarks in academia, considering the limited resources of academia compared to industry. Another

inquiry sought clarity on what constitutes a more complex benchmark and how to define it.

- The panelists were asked to project into the future, envisioning the benchmarks, capabilities, and challenges that might arise by 2030. This question stimulated a discussion on potential advancements and shifts in the landscape of foundation models.
- There was a question about whether language models alone will suffice as foundation models or if a more diverse approach is needed.

The panel concluded positively, recognizing the immense progress made in the field of foundation models. They acknowledged the substantial software engineering challenges that have been overcome to develop these advanced models and highlighted the ongoing efforts to address the remaining challenges in deploying, maintaining, and improving these systems.

Panel D: Foundation Models for Software Engineering

Panelists

- **Gustavo Soares:** From Microsoft, investigates leveraging foundation models for complex tasks like large code migrations or cloud app migrations, focusing on broader applications than typical code generation.
- **Andreas Zeller:** From Saarland University, involves in mining data from repositories, version histories, and integrating machine learning techniques with program and process analysis.
- **Alberto Bacchelli:** From University of Zurich, specializes in the social cognitive aspects of software engineering, with a keen interest in code reviewing and its multifaceted impacts.
- **Jie Zhang:** From King's College London, works at the intersection of foundation models and software engineering, focusing on evaluating non-determinism in LLMs in code generation.
- **Yiling Lou:** From Fudan University, concentrates on AI applications in software engineering tasks, particularly in testing, debugging, and code generation.

Discussion Highlights

- **Expanding Beyond Code Generation (Soares)**: He spoke about looking beyond typical code generation tasks. For instance, when migrating software to cloud platforms like Azure, it's not just about code changes. The models should assist developers in understanding Azure resources, evaluating costs and benefits, and choosing the right architecture.
- **Software Bots for Testing and Debugging (Zeller)**: Proposed the idea of software bots to aid in tasks like testing, debugging, and monitoring. He emphasized the challenges in automated testing, especially for programs with complex inputs. Valid, diverse, and uncommon inputs are required for effective testing, along with the ability to interpret semi-formal natural language descriptions.
- **Mentoring and Collaboration Enhancement (Bacchelli)**: Suggested using LLMs to mentor developers and improve collaboration within code reviews. He envisioned LLMs aiding in enhancing communication skills and providing insights during collaborative processes.
- **Optimizing Code for Quality Assurance (Zhang)**: Discussed the potential of foundation models in optimizing code, particularly for addressing coding smells and improving coding practices, especially for junior developers.
- **Complex Code Comprehension Challenges (Lou)**: Pointed out the difficulties foundation models face in comprehending complex, interdependent code, which is common in real-world software development.

Trustworthiness and Education in Model Use

- **Critical Thinking in Model Responses (Soares)**: Noted that altering how models generate responses (e.g., phrasing them as questions) can promote critical thinking in users. It's essential for models to convey confidence levels appropriately and engage in a more interactive, question-based dialogue.

- **Combining Traditional SE Techniques with Models (Zeller)**: Emphasized the need to not completely rely on FMs and instead to combine generative AI with traditional software engineering methods like static analysis to enhance quality assurance.
- **Novice Developer Risks with AI (general panel concern)**: Discussed risks associated with novice developers uncritically accepting model-generated code. There's a need for more effective training mechanisms to ensure developers critically evaluate AI suggestions.
- **Tailoring Models for Education and Contextual Use (Bacchelli and Lou)**: Explored the idea of domain-specific foundation models for educational purposes and the importance of contextual awareness in model responses for effective use.

Multimodal Data in Software Engineering

- **Challenges in Integrating Diverse Data Types (Soares and Zeller)**: Discussed the complexity of incorporating various data types (e.g., diagrams, audio, video) in software engineering. The challenge lies in creating a cohesive workflow that makes sense from a software engineering perspective.
- **Using Multimodal Data for Testing and Design (Lou and Bacchelli)**: Highlighted the potential of multimodal data in tasks like GUI testing and incorporating domain-specific knowledge. The focus was on how this data could inform better decision-making and improve code quality.

Closing Thoughts

- **Future of AI and Human Collaboration in SE (panel consensus)**: The panelists concluded by stressing the importance of balancing AI and human collaboration in software engineering. They emphasized that while AI can augment and assist, human decision-making remains crucial.
- **Virtualization of Customers and Developers (Zeller and Bacchelli)**: Speculated on the future where AI models could virtualize customers and developers, aiding in decision-making and understanding customer needs. However, they cautioned against over-reliance on AI, reminding that human insight is indispensable.

The discussion highlighted the need for foundation models to extend beyond code generation, focusing on broader software engineering tasks like cloud migrations, testing, and debugging. The importance of human-AI collaboration was reiterated, with a focus on how AI can aid in education, mentoring, and enhancing team collaboration. Concerns were raised about the impact of AI on less experienced developers and the need for effective models that can guide and educate. The potential of integrating multimodal data into software engineering processes presents both challenges and opportunities for more comprehensive and effective AI applications. The panel concluded with a vision of AI augmenting human capabilities in software engineering, emphasizing the irreplaceable role of human decision-making and creativity.

Panel E: Foundation Models and Open Source

Panelists

- **Zhenming (Jack) Jiang:** From York University, works in FM-augmented tools development.
- **Diomidis Spinellis:** From Athens University of Economics and Business, focuses on open source and its evolution with FM.
- **Marco Castelluccio:** Engineering Manager from Mozilla, works in the intersection of open source and FM, especially in the context of the Mozilla web browser project.
- **Daniel German:** From University of Victoria.
- **Tapajit Dey:** From Carnegie Mellon University, Software Engineering Institute, is interested in cultural and societal impacts of FMs on open source communities.

Discussion Highlights

- **Impact of Foundation Models on Collaboration (Jiang)**
  - FM tools like GitHub Copilot act like virtual pair programmers, aiding in coding and reviewing.
  - Data quality in FMs is crucial; examples include errors in ChatGPT's responses to chemistry tests.
  - Shift towards data-driven model improvement with platforms like OpenAI and Hugging Face collecting test cases and feedback.
- **Open Source Ecosystem and FMs (Spinellis)**
  - Concerns about the transparency and reusability of FMs that are trained on open-source data but are not open themselves.
  - Potential for knowledge ossification as foundation models absorb open-source knowledge without adequate contribution back.
  - Need for equitable knowledge sharing and advancements through FMs.
- **Challenges and Opportunities with Open Source FMs (Castelluccio)**
  - The potential revolutionary impact of open source FMs on collaboration and contribution.
  - New contributors using ChatGPT and Copilot, leading to mixed-quality contributions.
  - Necessity for balance between FM assistance and maintaining human communication in open source projects.
- **Cultural Dynamics in Open Source Projects (Dey)**
  - FMs as assistants in open source can help with project maintenance but may weaken human connections.
  - Concerns about AI mediation affecting the personal touch in collaborations.
  - The importance of keeping the 'human in the loop' in open source projects.
- **Trust and Openness in Foundation Models (panel discussion)**
  - The debate on whether open sourcing FMs would instill trust.
  - The necessity of openness for innovation, yet not sufficient for establishing trust.

- The role of traditional software engineering principles like regression testing in assessing FM quality.

Future Perspectives

- **Emerging Contribution Dynamics (Castelluccio)**
  - Increase in contributions from a diverse audience, including non-coders.
  - Future challenges in evaluating non-traditional contributions like prompts and data sets.
- **Evolving Quality Assurance and Ethics (Dey)**
  - The challenge of maintaining code quality and ethical standards in AI-assisted contributions.
  - The potential need for new contribution guidelines that include AI collaboration acknowledgment.

Audience Interaction and Queries

- **Wide Range of Topics Addressed**
  - Questions on trust in AI collaborations, automated code reviews, and the future of human communities in an automated landscape.
  - Discussion on balancing automation and human creativity, with ethical implications of FM use in open source.
- **Consensus on Positive Outlook**
  - General agreement on the bright future of open source with FMs.
  - Acknowledgment of challenges during transitional periods and the need for new practices and ethical considerations.
  - Anticipation of technological advancements facilitating more inclusive community participation in open source.

The panel broadly agreed on the transformative potential of FMs in open source, foreseeing more inclusive and diverse participation. Concerns were raised about maintaining the integrity and quality of contributions, the need for new evaluation systems, and the ethical use of FMs. The future of open source with FMs is seen as promising, with a balance between technological advancement and human-centric approaches in software development.

Panel F: Foundation Models and Software Trustworthiness

Panelists

- Liming Zhu: Research Director from CSIRO Australia, focused on cybersecurity, privacy, responsible AI, and trustworthy AI.
- Ipek Özkaya: Technical Director from Carnegie Mellon University, Software Engineering Institute, working at the intersection of engineering AI-enabled systems and AI for software engineering, with a foundation in software architecture.
- Lingming Zhang: Researcher from University of Illinois at Urbana-Champaign, specializing in applying language models to real-world software problems, including software quality assurance.
- Susumu Tokumoto: Researcher from Fujitsu Research Japan, with interests in AI for software engineering, particularly in testing and debugging.
- Dayi Lin: Principal Researcher at the Center for Software Excellence in Huawei Canada, focusing on software engineering for traditional AI and foundation model applications.

Discussion Highlights

Understanding Software Trustworthiness

- **Zhu:** Defines trustworthiness as an objectively measurable aspect of software, correlating with how well it meets set expectations. Differentiates between trustworthiness (objective assessment) and trust (subjective belief). Specifies that trustworthiness can apply to various attributes like reliability and security.
- **Özkaya:** Focuses on risk and user perception in trustworthiness, noting that it is context-dependent and always in flux. Asserts the impossibility of achieving absolute trustworthiness, as system attributes change with the environment.
- **Zhang:** Discusses the necessity for objective evaluation techniques in assessing software trustworthiness. Acknowledges the limitations in current methods to completely guarantee software correctness and underscores the vital role of human developers in quality assurance.
- **Tokumoto:** Breaks down trustworthiness in foundation models into three layers: the model alignment, prompting, and API usage. Each layer contributes uniquely to the overall trustworthiness.
- **Lin:** Observes that definitions of trustworthiness evolve with software advancements. Highlights those new dimensions, such as explainability and transparency, have become crucial in AI-integrated software, changing traditional trustworthiness perspectives.

The Role of Quality Assurance

- **Zhu:** Suggests a black-box approach to testing foundation models but emphasizes the importance of human-understandable guardrails at the system level for extra safety.
- **Özkaya:** Highlights the critical role of data quality in foundation models. Mentions the potential of fine-tuning these models with domain-specific data to enhance their trustworthiness.

- **Zhang:** Advocates for an integrated approach to working with foundation models throughout the software development lifecycle, using traditional and adapted quality assurance techniques to enhance the software's trustworthiness.

Software Engineering's Contribution to Trustworthiness

- **Panel Discussion:** The panelists collectively recognized the need for software engineering expertise in assessing and improving foundation models' trustworthiness. They stressed the importance of blending conventional software engineering methods with AI-specific approaches in an interdisciplinary manner.

Interaction and Queries from the Audience

- The audience raised various topics, including the integration of domain-specific knowledge into foundation models, the contrast between human-to-human and human-AI collaboration, issues surrounding AI accountability, and strategies to develop trust in foundation models.

The panel agreed on the complex and multi-faceted nature of software trustworthiness, particularly in the context of foundation models. An interdisciplinary approach, combining traditional software engineering practices with AI-specific strategies, was emphasized as critical. The panelists acknowledged the evolving nature of trustworthiness with the integration of FMs, underscoring the importance of human participation, risk management, and the continuous adaptation of quality assurance methods. Audience inquiries highlighted concerns about foundation model's involvement in software development and underscored the need for a deeper understanding of FM-powered systems to build trust.

Acknowledgement

We extend our deepest appreciation to all the FM+SE Vision and School 2030 participants who journeyed from diverse corners of the globe to participate in this event. Their unwavering commitment and enthusiasm in sharing their work, perspectives and visions have illuminated this gathering with a brilliance that is truly unparalleled.

The exchange of perspectives and the depth of discussions witnessed during this gathering have undoubtedly marked a significant milestone in the history of Software Engineering. It is through their collective wisdom and generosity in freely sharing insights that we have been able to glimpse the contours of a future brimming with innovation and possibility.

This report stands as a testament to the richness of their deliberations and the profound impact of their contributions. May this report serve as a vessel to encapsulate the essence of the discussions and propel the trajectory of our field towards new horizons.