

# Software Engineering Assignment

Natalija Glisovic

June 2025

## 1. Introduction

My research area is recommender systems. Recommender systems are models that provide personalized suggestions to users, helping them discover relevant content or products. These systems are essential in modern digital environments, helping users navigate overwhelming amounts of information by tailoring content based on their preferences, behaviour, and history. I am an industrial PhD student at an e-commerce, so my focus is specifically on furniture as items.

Within this broader field, my focus is on context-aware recommender systems. Traditional recommendation models often rely solely on user-item interactions, i.e. what a user has rated, purchased, or viewed in the past. However, these models may overlook important external factors that influence user preferences, such as time of day, location, device used, or social context. Additionally, item information such as product category, colour, item dimensions are also external factors to consider. Context-aware recommender systems aim to incorporate such auxiliary information to refine personalization and make recommendations more relevant in specific situations.

A central challenge in building these systems lies in how to integrate contextual data into the learning process. My research explores two key dimensions of this problem: (1) data augmentation, i.e. expanding or enriching the training data to include more varied and informative contextual signals; and (2) representation learning, i.e. embedding users, items, and contextual factors into shared latent spaces that allow the model to reason more effectively about their interactions. The ultimate goal of the research is to design recommender systems that are not just personalized, but situationally aware so that they are capable of adapting to the dynamic nature of human preferences across different environments.

## 2. Lecture principles

### 2.1 Property-Based Testing

The lecture on Quality Assurance in SE covered the concept of property-based testing. This is an approach where tests are based on general properties, i.e. expected behaviours of the systems. This is combined with randomized test data generation to uncover unexpected behaviours. Unlike traditional example based tests, PBT explores a wide range of inputs, making it effective at revealing subtle bugs and violations of expected "laws.". It is also easy to do with no complex tools or algorithms required.

This aligns with challenges in context-aware recommender systems. In these systems, outputs are personalized and depend not only of user history, but also contextual factors such as time. However, the personalization aspect often makes these systems difficult to test, since expected outputs vary by user and context. Using property based testing we can define properties such as stability under similar contexts where small changes in context should not lead to very different recommendations. We can also test for context sensitivity where a meaningful change in context, e.g. bedroom furniture vs. kitchen furniture, should result in observable changes in recommendation. This can help uncover edge cases such as over personalization, confirmation bias feedback loops, or context features being ignored. This allows testing general behavioural expectations rather than specific recommendation outcomes, which offers a way to validate robustness and adaptability.

### 2.2 Behavioural Software Engineering (BSE)

One of the topics we learned about was Behavioural Software Engineering (BSE), which is the study of cognitive, behavioural, and social aspects of software engineering. Specifically, I will focus on the

cognitive bias part of this lecture. Confirmation bias refers to the tendency of individuals to seek out and interpret information that confirms their pre-existing beliefs, while disregarding conflicting evidence. The lecture also covered availability bias, i.e. the tendency to be influenced by information that is recent, easy to recall, or widely publicized, and anchoring bias, which is the tendency to rely too heavily on an initial piece of information when making decisions, even if it is irrelevant.

In my research area of recommender systems, cognitive biases can be unintentionally reinforced through algorithmic personalization. Specifically, in context-aware recommender systems, we tailor recommendations not just based on users' historical behaviour, but also on contextual signals such as time or social environment. While this enhances relevance, it increases the risk of creating a feedback loop where users are continually exposed to content that aligns with their prior preferences and situational patterns, which can deepen the effects of confirmation bias.

For example, a user who typically listens to upbeat music at the gym might consistently be recommended similar playlists in that context, preventing exposure to different genres or moods. Over time, this filtering can lead to not capturing or encouraging evolving user preferences. Similarly, availability bias may cause recommender systems to overemphasize recently consumed or trending content, simply because it is more prominent or easier to retrieve. A popular song might dominate recommendations, even if it doesn't align with user interests. Meanwhile, anchoring bias can occur when early interactions in a new context set a strong precedent, for instance, if a user watches one romantic movie late at night, the system may anchor on that behavior and continue recommending similar content at that time, regardless of their broader preferences.

These biases have broader implications not only for user experience but also for long term engagement and diversity of exposure. Understanding these behavioural patterns is essential for designing recommender systems that not only personalize effectively but also support exploration and cognitive diversity.

### **3. Guest lecture principles**

#### **3.1 Goal modelling**

From the guest lecture on requirements engineering we talked about how to actually apply it. It includes stakeholder elicitation, goal modelling, system vision and requirements elicitation. I will focus on the goal modelling part. Goal modelling focuses on why a system is being developed, before we go into what and/or how it should do it. Goals represent desired outcomes or intentions of stakeholders, and goal models help in structuring these goals and understanding their relationships. We covered three goal types: usage goals, i.e. relevance for end users, system goal, i.e. system properties, and business goals, i.e. strategic goals set by organization.

Goal modelling is directly relevant to my research on context-aware recommender systems, as it helps structure and analyze the different layers of objectives of the system design. At the business level, organizations may aim to increase user engagement, drive content consumption through personalization. At the system level, goals involve adapting recommendations based on contextual factors such as time, location, or social setting, while ensuring performance, diversity, and privacy. Finally, at the usage level, the focus is on what users want to achieve, i.e. receiving relevant suggestions in specific contexts, discovering new content, and understanding why recommendations are made. Modeling these goals supports clearer trade-off analysis (e.g., between relevance and diversity).

#### **3.2 System black box. vs glass box**

The guest lecture on requirements engineering talked about system black box view vs. system glass box view. This is two different ways to understand and evaluate a system, where black box refers to solely system's external behaviours and glass box looks more at internal logic and data flows within the system.

In my research on context-aware recommender systems, this distinction is relevant as we should adopt both views. From a black box perspective, we evaluate whether the system delivers relevant and timely

recommendations based on contextual input. However, adopting a glass box view is essential for addressing other concerns such as user trust, transparency, and fairness. Understanding how user behaviour, contextual signals, and model logic interact can help users make sense of the system’s behavior and reduce the risks of cognitive bias and over personalization.

#### **4. Data Scientists versus Software Engineers**

Overall, I agree with the distinctions drawn between data scientists and software engineers, particularly the idea that data scientists tend to focus more on modeling and data analysis, while software engineers are more concerned with delivering robust, scalable products under time and budget constraints. However, as the chapter rightly points out, this is a simplification. The boundaries between these roles are often blurry. For example, it’s not necessarily true that data scientists always come from academic backgrounds with PhDs in statistics or machine learning, nor that software engineers lack the same level of formal education. In practice, I’ve seen software engineers with PhDs in mathematics and data scientists with master’s degrees in physics. These examples show that academic background doesn’t strictly define the role. Additionally, the focus on modeling as a defining feature of data science doesn’t capture the full picture. Many data scientists are just as involved in exploratory data analysis, understanding user behaviour, and even participating in deployment and pipeline processes, areas traditionally associated with software engineering.

I believe we’re moving toward a hybrid model where both data scientists and software engineers will increasingly need to learn skills from the other side. As the demand for deploying and maintaining machine learning models grows, software engineers may need a better understanding of modeling techniques. Similarly, data scientists will benefit from learning more about production level systems, version control, and other engineering practices. That said, I don’t think the roles will fully merge. The core skillsets and mindsets still differ, and that distinction is valuable. Data scientists will likely continue to lead on modeling and data interpretation, while software engineers will remain experts in building and scaling applications.

As the chapter notes, interdisciplinary teams are becoming more common, and also necessary. Rather than merging into a single role, I think the future lies in stronger collaboration, where professionals from each domain share enough overlapping knowledge to work effectively together while still having different skillsets.

#### **5. Paper analysis**

##### **5.1 ImageBiTe: A Framework for Evaluating Representational Harms in Text-to-Image Models**

This paper introduces ImageBiTe, which presents a systematic framework for testing representational bias in text-to-image (T2I) models, addressing gaps in AI system engineering [1]. The core of the framework lies in transforming ad-hoc bias detection into a repeatable, scalable engineering process through three key contributions: automated test case generation from user-defined ethical requirements, distribution-based assessment of sensitive community representation, and seamless integration into AI development workflows. This approach is crucial for AI system engineering because it operationalizes ethical requirements into testable specifications, enabling continuous bias monitoring throughout the development lifecycle. Beyond bias detection, this framework also provides a structured method for translating abstract fairness concerns into concrete engineering practices, which is essential for AI regulations in the EU.

My research on context-aware recommender systems shares similar concerns with ImageBiTe regarding fairness and representation in AI generated content. While ImageBiTe focuses on visual bias in generated images, context-aware recommenders face similar challenges in ensuring good representation across different user contexts and demographics. Both domains struggle with the challenge of defining and measuring fairness in personalized AI outputs. ImageBiTe’s systematic approach to specifying ethical requirements and automated bias assessment could inform how we evaluate fairness in recommenda-

tion systems, particularly in how different contexts might amplify or mitigate representational biases. The framework also includes user defined sensitive communities, which could be used in recommender systems as they vary across user demographics.

For a fictional AI intensive software program we can consider a content creation platform that uses AI for multiple modalities, i.e. generating images, writing content, and providing personalized recommendations for creators. This platform serves global users creating content for education, marketing, and entertainment. ImageBiTe’s framework would be integrated into the platform’s T2I generation pipeline, continuously monitoring generated images for representational bias across different cultural contexts and use cases. The platform would implement ImageBiTe’s ethical requirements specification to adapt bias detection to different markets and environments. My research would complement this by ensuring that the content recommendation engine considers cultural context when suggesting templates, styles, or content themes, preventing biased recommendations. The system would maintain consistency between visual and recommendation fairness, creating content generation where both images and recommendations avoid harmful biases such as stereotypes.

To better support ImageBiTe’s vision of systematic bias assessment in AI systems, I would expand my research to include explicit fairness-aware contextual modeling. This would involve developing context representations that capture not just situational factors but also demographic and cultural dimensions that could influence fair recommendation outcomes. I would integrate ImageBiTe’s concept of user defined ethical requirements into recommendation contexts, allowing stakeholders to specify fairness expectations for different contextual scenarios. Additionally, I would develop complementary assessment frameworks for recommendation bias that mirror ImageBiTe’s distribution based evaluation approach, creating standardized metrics for measuring representation across different contexts and user groups. Instead of just detecting bias like ImageBiTe does, my research would actively try to reduce bias by understanding the situation and making better, more balanced recommendations that don’t reinforce harmful stereotypes.

## **5.2 Rule-Based Assessment of Reinforcement Learning Practices Using Large Language Models**

This paper presents a rule-based framework that leverages Large Language Models (LLMs) and heuristic based code detectors to automatically assess compliance with best practices in Reinforcement Learning (RL) training pipelines [2]. The main concept is defining 31 architectural rules targeting critical areas such as checkpoints, hyperparameter tuning, and agent configuration, then implementing both traditional pattern matching detectors and LLM based analyzers to validate these practices in source code. This approach is crucial for AI systems engineering because it addresses the complexity of RL systems where manual oversight of training practices becomes impractical. The framework has the goal to ensure consistency and reliability of RL implementation by automatically detecting any violations of established best practices, to reduce the risk of training failures, improve reproducibility and enable better quality assurance in the AI development.

Although not common, RL can be used in my research area of recommender systems. However, the main connection between my research and this paper is that both domains deal with complex AI pipelines that require a more systematic approach to ensure reliability. My research is done at a large e-commerce where data handling, model training and hyperparameter optimization and performance monitoring are crucial, especially as we are dealing with large scales of data. The paper’s emphasis on automated compliance checking resonates with challenges in recommender systems where proper feature engineering, context integration, and evaluation metrics must be consistently applied. The LLM based code analysis approach could be adapted to validate best practices in recommender system implementations, such as ensuring proper context feature extraction, appropriate evaluation protocols, and correct handling of user privacy considerations.

For a fictional AI scenario we can consider an e-commerce platform that uses AI extensively for personal-

ized shopping experiences. This system employs multiple AI components: context-aware recommender systems for product suggestions, RL agents for dynamic pricing and inventory management, computer vision for product categorization. The platform processes millions of user interactions daily, requiring robust MLOps pipelines with continuous training, validation, and deployment across different AI models. The paper's rule based assessment framework could be integrated as a quality gate in the pipeline, automatically validating that all RL components (pricing agents, recommendation policy optimization) follow established best practices before deployment. My context-aware recommender research would contribute the recommendation engine component, ensuring that user context (location, time, device, purchase history, browsing behaviour) is properly integrated to provide personalized product suggestions. The system would need to handle real time context updates while maintaining recommendation quality and user privacy.

To better support the paper's AI engineering vision, I would expand my context-aware recommender research to include automated validation frameworks specifically designed for recommender systems. This could involve developing rule sets similar to the paper's 31 RL rules but focused on recommender system best practices such as proper context feature engineering, appropriate evaluation metrics (beyond accuracy to include diversity, novelty, and fairness) and privacy preserving techniques. I would integrate LLM based code analysis to automatically detect strange patterns in recommender implementations, such as data leakage in temporal splitting or inappropriate evaluation protocols. Long term, this research could evolve into a comprehensive framework for automated quality assurance in recommender systems, addressing challenges like bias detection, fairness validation, and ensuring that context-aware features are properly implemented and maintained across the system lifecycle, ultimately contributing to more reliable and ethical AI systems in production environments.

## 6. Research Ethics & Synthesis Reflection

In order to find the papers I went to the CAIN conference link we got given and checked the 2025 accepted papers. From there, I focused on any title that sounded interesting to me and also that I felt fit to answer the questions we got given. This included LLM, AI. I recognize that this was a very biased view to find papers. A more proper approach would be to look at papers from maybe 2-3 more years to not have such limited scope and to read more abstracts and focus less on titles, as they can be misleading. Additionally, someone else could provide additional keywords to make the search less biased.

I did not encounter any misleading titles or abstracts, but I adjusted my search a bit since some of the titles and abstracts that stood out to me I could not find a pdf on the paper on the website or google scholar so I disregarded those and found other ones. This can also lead to some relevant work not being discussed as I did not have access to the papers. In the future, I can first see which papers are accessible before doing my search.

To ensure originality so I do not copy anything from LLM or sources directly I took my own notes on the papers and used those to make my summary.

## References

- [1] S. Morales, R. Clariso, and J. Cabot, "Imagebite: A framework for evaluating representational harms in text-to-image models," *arXiv preprint*, 2024.
- [2] E. Ntontos, S. J. Warnett, and U. Zdun, "Rule-based assessment of reinforcement learning practices using large language models," in *Proceedings of the 4th International Conference on AI Engineering - Software Engineering for AI (CAIN)*, ACM, 2025.