# WASP Software Engineering Course Module 2025*
# Module 2 - Final Essay

Qingwen Zhang
qingwen@kth.se

August 24, 2025

## 1 Introduction

Our research focuses on self-supervised learning for understanding dynamic scenes. This is a key technology for autonomous systems, especially for self-driving cars. The main goal is to train a model to understand and predict motion in complex environments using raw data, avoiding the need for large, hand-labeled training set. The process of creating these manual labels is expensive, slow, and often leads to errors. This has become a major bottleneck for scaling up in developing a better autonomous perception system.

Our work is guided by the principle of *Let Data Be the Teacher*. We are exploring methods that allow systems to learn directly from sensor data, like 3D point clouds. By developing self-supervised methods, we aim to build perception systems that are easier to scale, more reliable, and can adapt to new situations. Our work has produced several new methods, such as DUFOMap [1] and SeFlow [2]. These methods use the geometry and timing of sensor data to tell the difference between static and dynamic objects and to predict how dynamic objects will move. For example, SeFlow combines dynamic object classification with rules about how objects should move consistently, which leads to better scene flow estimates. This approach allows a system to constantly learn and improve its understanding of the world as it gathers more data. The long-term goal of this research is to build a complete framework for self-supervised dynamic scene understanding that can be used in many robotics applications to make them safer and more efficient.

## 2 Lecture Principles

The lectures on Software Engineering introduced two concepts that are especially relevant to our research: the difference between Verification and Validation and the principles of Property-Based Testing (PBT). These ideas give us a useful way to think about and check the quality of the self-supervised models we are developing.

Verification ('Are we building the product right?') and validation ('Are we building the right product?') are basic concepts in software engineering that apply directly to our work. In our research, verification relates to checking that our model, like SeFlow, is implemented correctly based on its design. This includes things like code reviews and unit tests to make sure the algorithms are working as intended. However, validation is the bigger challenge. It asks whether the model correctly understands a dynamic scene in a way that is useful and safe for a self-driving car. A model can be technically

*We have used LLM for editing and polishing author-written text.

correct (verified) but still fail in the real world if it hasn't learned the right things. The example of a mislabeled trolley in the Argoverse 2 dataset shows the critical need for validation. The example of 'Hacking Wood' from the lecture (being efficient but not effective) perfectly captures this. Our self-supervised approach is naturally heavy on validation, as it constantly refines the model's understanding based on real-world data to ensure it builds the right product.
Property-based testing (PBT) is another key technique from the lectures that applies directly to our research. Instead of testing single examples, PBT checks that general rules, or properties, are always true for the system. This is very useful for deep learning models, where we can't possibly test every input. For our scene flow estimation models, we can define properties that should always be true. For example, a 'temporal consistency' property would check that an object's predicted motion is smooth over time. Another property, 'object rigidity', would ensure all points on a rigid object like a car are predicted to move together. Using PBT, we can automatically generate many different test cases from real or simulated data to see if our model breaks these rules. This is a much better way to find bugs and edge cases than creating tests by hand. As the lecture noted, PBT is great at finding 'strange' test data that causes problems, which is exactly what we need to build robust and reliable perception systems.

## 3 Guest-Lecture Principles

'Requirements Engineering' provides useful ideas that connect well with our research on self-supervised dynamic scene understanding. Two particularly helpful concepts are the difference between the 'Problem-Space' and 'Solution-Space' and the method of 'Stakeholder Elicitation'. These ideas provide a clear way to think about our research goals and the different factors that can affect our success.
The difference between the problem-space ('why' and 'what') and the solution-space ('how') is a key idea that helps clarify the focus of our work. Our research is mainly in the solution-space; we are developing specific methods like SeFlow and DUFOMap as the 'how'. However, the lecture stressed that we must first understand the problem space. For us, the problem space is about defining what it means for a self-driving car to 'understand' a dynamic scene and why it's so important. The 'why' is safety and efficiency. The 'what' is about specific abilities, like correctly finding moving objects and predicting their motion. The main message, 'Understand the Problem Before You Build the Solution', is an important reminder that our technical work must solve a real-world problem. The high cost and poor quality of manual annotations, a major motivation for our work, is a key aspect of this problem space. By clearly defining this, we can better judge if our solutions are solving the right problems.
'Stakeholder Elicitation' is another important concept. While a PhD project can seem like an individual effort, it exists within a larger context of stakeholders who care about the outcome. As the lecture explained, a stakeholder is anyone affected by the system or who can influence it. In our case, stakeholders include our supervisors, the research community, potential industry partners, government regulators setting safety standards, and even the general public. Each of these groups has different goals. The research community might value new ideas, while an industry partner will care more about efficiency and reliability. Regulators will focus on safety. By identifying these stakeholders and their needs, like in the lecture's TechStack Inc. example, we get a better picture of our research's context. This helps us form better research questions and explain our work to different groups. Thinking of the public as a stakeholder, for instance, reminds us of the need for explainable AI, so people can trust the decisions made by the autonomous systems using our models.

## 4 Data Scientists v.s. Software Engineers

The lectures and course materials highlight key differences between the roles of data scientists and software engineers. We agree with the essential distinction presented: data scientists are primarily focused on exploration and discovery, while software engineers are focused on building reliable and maintainable systems. These roles are not opposed but are complementary parts of creating successful AI-intensive products.
In our work, this distinction is very clear. The development of the core SeFlow model feels like a data science task. It involves experimenting with different neural network architectures, designing new loss functions, and running tests to see how accurately the model can estimate motion. The goal is to discover the best way to represent and learn from the 3D point cloud data. This is an iterative, research-driven process.
On the other hand, making our methods practical involves software engineering. For example, our work on DUFOMap, which creates an efficient map for real-time use, requires thinking about system performance, memory usage, and how the component will integrate into a larger autonomous driving stack. A data scientist might create a model that is highly accurate but too slow to run on a car; a software engineer's job is to either optimize it or build the infrastructure around it to make it work reliably in the real world.
Looking forward, we believe these roles will both specialize further and, in some ways, merge. The idea that one side will simply absorb the other is unlikely. Instead, we expect to see more specialized roles like "ML Engineer" emerge, whose job is to bridge the gap between data science and software engineering. These individuals will be experts in taking models from the prototype stage to production. At the same time, the boundaries will blur. Software engineers working on AI systems will need to understand the basics of deep learning, such as the concept of model drift or the importance of data quality, to build effective systems. Similarly, data scientists will need to adopt more software engineering best practices, like using version control for their models and data, writing code that is easy to test, and creating models that are designed for deployment from the start. The growth of MLOps (Machine Learning Operations) as a field is strong evidence of this trend. It is all about applying engineering principles to the machine learning lifecycle. For our research to have a real-world impact, we must consider both sides: we need the data science to create innovative models and the software engineering to ensure they can be deployed and maintained safely and efficiently.

## 5 Paper Analysis

To connect our research to the current challenges and practices in AI engineering, we analyzed two papers from the CAIN conference (2025-2022). The first paper from CAIN 2023 explores the foundational problems of data and annotation in the automotive industry, which sets the stage for our work. The second paper from CAIN 2025 examines the end goal of safety assurance for autonomous systems, showing where our work must ultimately fit.

### 5.1 Automotive Perception Software Development [3]

**Core Idea and SE Importance** The main idea of this paper is that building perception systems for cars is difficult because automotive companies and their suppliers struggle to agree on what 'good' data and 'good' annotations are. The authors interviewed people in the Swedish automotive industry and found that there is a 'lack of effective metrics for data quality' and 'unclear definitions of annotation quality'. This is a major software engineering problem because if you cannot clearly define your requirements—in this case, for data and its labels—you cannot build a reliable system.

It creates ambiguity, makes it hard to test the system, and undermines accountability. For safety-critical systems, this is a huge risk. The paper shows that data is not just an input but a core part of the system that needs its own rigorous engineering process.

**Relation to Our Research** This paper perfectly sets up the problem that our research is trying to solve. The challenges it describes, the high cost of annotation, the difficulty in defining quality, and the inconsistencies in manual labeling, are the exact reasons we are developing self-supervised methods. Our work aims to reduce the dependency on this complex and often flawed annotation process for training. By letting the data be the teacher, our methods learn directly from raw sensor input. This approach addresses the paper's finding that unclear specifications for annotations are a major source of problems for large-scale training.
However, it is important to acknowledge that self-supervision does not eliminate the need for human-labeled data entirely. To properly *evaluate and validate our models*, we still rely on a smaller, high-quality "gold standard" dataset to serve as ground truth. The key engineering trade-off is shifting the burden of annotation away from massive, expensive training sets towards smaller, more manageable, and meticulously curated test sets. This allows human effort to be focused where it is most critical: on verifying the system's performance and ensuring it meets safety requirements. This makes the challenge of defining annotation quality, as highlighted by the paper, more tractable because the scope is much smaller and the purpose (evaluation, not training) is much clearer.

**Integration into a Larger AI-intensive Project** Let's imagine a fictional project at a ride-sharing company (called Company A) that aims to continuously improve the performance of its entire fleet of autonomous vehicles. The managers of the company would read this paper and understand that simply outsourcing data annotation to the lowest bidder is a risky strategy that could lead to low-quality models. Instead, they would see the value of our self-supervised SeFlow model. The company would build its strategy around collecting large amounts of raw sensor data from its vehicles and using our methods to continuously train and improve the perception system. This would reduce their reliance on external annotation suppliers and give them more control over the quality and consistency of their AI models, directly mitigating the 'ecosystem challenges' described in the paper.

**Adaptation of Our Research** To better address the issues raised in this paper, we could adapt our research to include developing new metrics for data quality. The paper highlights that the industry lacks good metrics. Our self-supervised models could be used not only for perception but also as a tool to evaluate the data itself. For example, we could design our system to automatically flag new sensor data that is unusual or 'out-of-distribution' compared to what it has seen before. This would provide an automated signal to the engineering team that the data might represent a new edge case that needs attention. This would adapt our research from simply being a consumer of data to also being a validator of data, providing a direct solution to one of the key problems identified by the paper.

## 5.2 LLM-Based Saftey Cases Generation [4]

**Core Idea and SE Importance** A safety case is a formal document that provides a structured argument, supported by evidence, to prove that a system is safe enough for its intended use. The core idea of this paper is to explore whether Large Language Models (LLMs) can help automate the creation of these safety cases for autonomous driving systems like Baidu Apollo. Manually creating these documents is described as a 'tedious, and error-prone process' that requires a lot of expert knowledge. This is extremely important for software engineering because it addresses the final and most critical step in building trustworthy AI systems. A safety case is the ultimate form of validation, providing the necessary assurance to regulators and the public that a system's non-functional requirements (like safety) have been met.

**Relation to Our Research** Our research on perception models like SeFlow provides the critical technology that a safety case like this would need to analyze. The safety case would make high-level claims, such as 'the vehicle will not collide with pedestrians'. The evidence to support such a claim would come directly from testing and validating our model's performance. The accuracy, robustness, and reliability of our SeFlow model's predictions about the movement of objects in a scene would be a key piece of evidence. This paper shows us that our work doesn't end with publishing a model with high accuracy on a benchmark; for it to be used in the real world, its performance must be documented and integrated into a formal safety argument.

**Integration into a Larger AI-intensive Project** The Company A project would use the methods described in this paper in its final stages, before deploying its autonomous vehicles on public roads. After developing and testing our SeFlow model, the safety engineering team at company A would be responsible for building a comprehensive safety case. They could use an LLM-assisted approach to draft arguments about how SeFlow's performance contributes to overall vehicle safety. For example, they would feed the LLM the test results, validation reports, and operational data from our model, and the LLM would help structure this evidence into a formal safety argument, following industry standards like Goal Structuring Notation [5]. Our model's outputs would be the foundation of the safety argument's evidence.

**Adaptation of Our Research** This paper shows that for our research to be useful in a safety-critical product, it's not enough for our model to just be accurate. We also need to provide clear, verifiable evidence of its performance in a way that can be used in a safety case. To support this, we could adapt our research to make our models more 'explainable' and 'monitorable'. For example, we could design SeFlow to not only predict motion but also to output a 'confidence score' for its predictions. When the model is uncertain, it would report a low confidence score. This score would be a valuable piece of evidence for the safety case, as it would help define the operational design domain where the system can be trusted to operate safely. This would directly align our research with the end goal of formal safety certification.

## 6 Research Ethics & Synthesis Reflection

**Search and Screening Process** Our literature search was targeted specifically at the list of long papers from the CAIN conferences (2022-2025) provided for the assignment. We began with a manual review of all the paper titles in the list. The goal of this initial pass was to create a shortlist of papers that appeared most relevant to our research on self-supervised perception for autonomous systems. We filtered for keywords and concepts such as 'automotive', 'data', 'annotation', 'perception', 'safety', and 'requirements'.
From this shortlist of approximately 5 papers, we moved to a second screening phase where we carefully read the abstract of each paper. This step was important for confirming the paper's true focus. We selected the final two papers: a) Heyn *et al.* on automotive perception challenges and b) Odu *et al.* on safety cases for Baidu Apollo—based on their direct and complementary relevance to our work. One paper defined the core problem-space that motivates our research (data and annotation), while the other

addressed the end-goal of engineering assurance (safety cases), providing a complete narrative for our analysis.

**Pitfalls and Mitigations** The main challenge we encountered was the potential for misleading titles. Some titles seemed broadly relevant to "AI Engineering" but, upon reading the abstract, were found to be highly specific algorithmic studies with little discussion of broader engineering lifecycle concerns. Our mitigation for this was the rigorous two-step screening process. Relying on abstracts rather than just titles ensured our final selections were genuinely aligned with the course's focus on engineering practice. Another potential pitfall was selecting two papers that were too similar. We mitigated this by deliberately choosing papers that addressed different stages of the engineering lifecycle, one focused on the initial data challenges and the other on the final safety validation, to ensure a richer and more comprehensive analysis.

**Ethical Considerations** The core analysis, synthesis, and intellectual contributions in this essay are our own original work. In the spirit of transparency and modern research practices, we want to clarify the role of Large Language Models (LLMs) in our writing process. We utilized an LLM as a sounding board during the initial brainstorming phase to help explore different ways of connecting lecture concepts to our research. However, the core ideas and the structure of the arguments were developed independently. Furthermore, after drafting the content ourselves, we used an LLM as an assistive tool for language polishing, including grammar correction, proofreading, and improving the overall clarity and flow of the text.

## References

[1] Daniel Duberg, Qingwen Zhang, Mingkai Jia, and Patric Jensfelt. DUFOMap: Efficient dynamic awareness mapping. *IEEE Robotics and Automation Letters*, 9(6):5038–5045, 2024.

[2] Qingwen Zhang, Yi Yang, Peizheng Li, Olov Andersson, and Patric Jensfelt. SeFlow: A self-supervised scene flow method in autonomous driving. In *European Conference on Computer Vision (ECCV)*, page 353–369. Springer, 2024.

[3] Hans-Martin Heyn, Khan Mohammad Habibullah, Eric Knauss, Jennifer Horkoff, Markus Borg, Alessia Knauss, and Polly Jing Li. Automotive perception software development: An empirical investigation into data, annotation, and ecosystem challenges. *arXiv preprint arXiv:2303.05947*, 2023.

[4] Oluwafemi Odu, Alvine Boaye Belle, and Song Wang. Llm-based safety case generation for baidu apollo: Are we there yet? In *2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 222–233, 2025.

[5] Goal structuring notation (gsn) standard.