WASP Software Engineering and Cloud Computing (2025)
Software engineering module assignment

Michele Di Sabato - michele.di.sabato@umu.se

August 23, 2025

1. Introduction

My research area pertains to mathematical methods for clustering and statistical hypothesis testing in functional data analysis. In ordinary multivariate statistics and data analysis, each datum is a vector of features and datasets are tabular datasets. On the other hand, in functional data analysis each datum is a map from one space to another. For example, time series can be modeled as functions of time and images can be viewed as function of the pixel's locations/indices. The central task of hypothesis testing in this field is to assess whether evidence in the data (e.g. time series or images) contradicts a null hypothesis H_0 in favor of an alternative H_1. Practically, we quantify how inconsistent the observed data are with H_0. If strong inconsistencies are detected, then this is counted as evidence in favor of the alternative hypothesis H_1. The goal is not to prove that a statement is true, but to supply reliable procedures that let scientists (domain experts) draw such conclusions from arbitrary datasets. The methodologies developed in this field of statistics are designed to fulfill certain theoretical guarantees, such as a desired control over false discoveries. These occur if the null hypothesis is wrongly rejected. Papers in this field often pair theoretical proofs with simulations based on synthetic datasets, that illustrate the proposed testing method, highlight its performance, and expose computational bottlenecks. Using SE terminology, these simulations are the software "product" and the "users" coincide with the members of the research group, who would like to draw conclusions based on the output of the simulations. Working with time series poses significant computational challenges, as testing methodologies often require high-resolution data. The theoretical proofs need to come to terms with the actual practical implementation of these methodologies which introduces numerical errors and sometimes compromises with theoretical assumptions. Moreover, it may sometimes be necessary to use packages written in a different programming language than the one used for the simulations (usually R). In such cases, workarounds are often required to ensure compatibility.

A concrete illustration comes from gait analysis: comparing knee-flexion curves during a one-leg hop for distance between previously injured patients and healthy controls. Here one would test $H_0 : \mu_{\text{injured}}(t) = \mu_{\text{healthy}}(t)$ for all times t (the knee flexion dynamics of injured and healthy patients are the same) against the opposite $H_1 : \mu_{\text{injured}}(t) \neq \mu_{\text{healthy}}(t)$ for some time t. Multiple methods rooted in functional data analysis might, for example, highlight differences at the take-off and landing phases of the jump. This example shows that the output of this field often benefits from the opinion of domain experts who aren't familiar with mathematics or machine learning.

2. Lecture Principles

1. Methods for verification: While verification consists in checking if the software/product works correctly and meets its technical specifications, the goal of validation is making sure that the product meets the customer's/user's needs. As said in the introduction, in the context of my research, the "product" usually consists in the output of simulations aimed at showcasing the practical aspects of a proposed methodology. These simulations are computationally intensive. Re-running them because of slight mistakes in the code, such as mishandling in the fault-detection and fault-tolerance of one or more jobs in the distributed system used to run the code, leads to significant delays. To avoid these, it could be useful to implement a more systematic control of "whether we are building the product right", as stated in the lecture. This not only consists in

the use of static techniques (such as code walkthroughs), but also dynamic techniques. For example, if part of a simulation is written in R but requires a function that exists only in Python, one needs to re-implement that function in R. In such cases, it is useful to verify that the R version produces the same output as the original Python implementation. Alternatively, when multiple parallelization schemes are possible, it's useful to verify on smaller-scale simulations that all schemes produce identical results before selecting the fastest implementation. What I gained from the lectures is firstly an understanding of the research field of software testing and secondly the importance of performing code checks and tests systematically, methodically, and with a user-oriented mindset.

2. Version control and pragmatic software engineering "rules": In my research, version control is essential because simulations are often run under different settings, for example by varying how synthetic data are generated or how the null and alternative hypotheses are formulated (H_0). Since research is not a linear process, it is common to explore one path, realize later that another design choice was better, and then need to revert to an earlier setup. With version control, I can always revert back to a previous version of the code and simulation parameters. This avoids repeating costly runs from scratch and keeps a clear history of how the project evolved. GitHub is particularly useful in ensuring this: even if the project does not involve a big research group, keeping a GitHub repository is useful to track code changes. Coding standards are equally important in this context. For example, when submitting newly developed R packages to CRAN[1], one should avoid duplicating the same functionality in multiple places. In my simulations, several simulation settings rely on very similar dataset creation steps. Instead of writing slightly different functions that all do almost the same thing, I restructure the code so that one well-written, standardized function handles this step for every setting. This reduces redundancy, makes the code easier to maintain, and ensures that if I need to update or fix something, I only need to do it in one place.

3. Guest-Lecture Principles

The two concepts from the guest lecture that resonated with me and my research broadly relate to the idea that planning steps systematically and in a disciplined manner before starting a project is beneficial. Although my research area is not closely linked to software engineering, I found that requirements engineering plays a role even in my work, particularly regarding the following two concepts:

1. Identifying stakeholders: The stakeholders in my research are heterogeneous. For example, when building simulations to compare different methods of doing statistical hypothesis testing, the list of people who have a certain interest in the "system" (simulations) to be developed includes other PhD students, supervisors I work with regularly, professors unfamiliar with the system's structure, and domain experts who lack mathematical backgrounds but are proficient in the scientific field where my research applies (e.g. gait analysis, as stated in the introduction). Listing all the stakeholders for a given project allows to reduce the risk of missing important needs or constraints and helps prioritize certain requirements over others. For example, imagine that the output of the simulations is displayed through an interactive dashboard. Through this dashboard, a user/stakeholder may change some model settings (the variability of the synthetic data, the metric used to check if the data are coherent with the null hypothesis H_0, the number of functions/time series in the dataset, etc.). Since there are many settings that could be changed, it is desirable to keep the dashboard as simple and readable as possible by excluding unnecessary elements. The interest of the stakeholders can clearly help decide which simulation settings to include in the dashboard. A domain expert might not be interested in changing the number of time series in the dataset, while other stakeholders with a mathematical background might want to use this simulation setting to see the asymptotic properties of the proposed model.

2. Cost of defect removal, goal modelling and refinement: In my research, the cost of defect removal is directly linked to the fact that simulations are computationally expensive: the generated time series are sampled at many points, producing very long vectors. If the setup of the simulations turns out to be wrong (for example, how synthetic data are generated, or how null and alternative hypotheses are formulated), fixing that mistake means re-running everything. This is costly both in terms of computation and time, similar to how software engineering defects become more expensive to fix the later they are discovered

[1] the official online repository where R packages are published and shared, see [1]

in the development pipeline. By carefully checking and refining requirements early (e.g., making sure the simulation setup is correct before running large-scale experiments), it is possible to decrease the waste in resources and time. A systematic goal refinement step prior to the implementation of the simulations' pipeline also helps make the implementation process smoother. This also removes redundancy: if the goal is precisely formulated (e.g. "we want to understand how the variability in the original dataset influences the model"), then it is possible to write more concise code that could, possibly, be updated and expanded if/when new goals are formulated.

4. Data Scientists versus Software Engineers

Overall, I agree with the distinction between data scientists and software engineers proposed in Chapter 1 of the book. Specifically, I agree on the fact that these two roles are very distinct and rely on different (academic) backgrounds. Because of this, in my opinion, it is challenging to be proficient in both "data science" and "software engineering", since both fields come with a broad collection of challenges. At the same time, the definition of the role of data scientists provided in the book seems to be simplified and generalized. Data scientists seem to be depicted as workers who are only focused on maximizing the accuracy of their model and who are almost unaware of the existence of a broader system that comes with other requirements. Instead, software engineers are described as individuals who work on almost all the rest of the software product. Nowadays, every data scientist most likely knows that the performance of their ML model is only one piece of a bigger picture, together with training costs and automation in the training/re-training pipeline. At the same time, it is clear that the author of the book uses the term "data scientist" to purposefully describe an outdated role, which is being replaced by ML engineers.

There is a common view of the data science pipeline that goes from data collection to model deployment. The author of the book argues that this is unrealistic, since data scientists must also deal with deployment and monitoring in an efficient, scalable, and robust way. These components inevitably influence the core data science tasks such as model training, validation, and feature engineering, and therefore cannot be considered in isolation. It would be unrealistic for a company to expect candidates ("unicorns", using the book's terminology) who are both proficient in building models for their needs, and also highly skilled in deploying these models efficiently and at scale. Tools that help integrate these steps without requiring infrastructure to be built from scratch already exist, such as Hopsworks ([2]). However, every company's needs are unique, and the MLOps part of the pipeline should therefore be tailored to the organization. This means that, in my opinion, the extended model pipeline[2] should be handled by two roles at the same time. Both roles should fall in the broader category of ML engineers but with different specifications. One role should have a strong background in building models, the other should also be an ML engineer but with a solid background in software engineering tailored to ML operations. Throughout the book, the authors describe a T-shaped employee, who is proficient in one specific topic in their field, but who also has broad general knowledge of other topics that pertain to the field. Both roles described previously should be T-shaped employees, being proficient in either model building, feature engineering, and data science, or MLOps. Their broad knowledge should allow them to communicate easily between each other regarding possible model choices or architecture choices to make models deployable. The separation of these two ML engineering roles allows for a more flexible interaction between the various teams who work on the product. Indeed, as mentioned in the book, the extended model pipeline is just one component of a broader system, which consists of the actual software architecture of the product and the interface with the external world (i.e., the users). This means that if MLOps are dealt with by someone who has a strong SE background, then it would be easier for them to communicate with the software developers who deal with this broader infrastructure.

[2] This is the pipeline described in Chapter 2, going from "model requirements" to "model deployment"

5. Paper analysis

5.1. Paper 1: Developer Experiences with a Contextualized AI Coding Assistant: Usability, Expectations, and Outcomes

The authors of this paper ([3]) discuss contextualized coding assistants. When making use of general purpose AI chatbots to write code, some issues might be encountered such as a lack of specificity of the generated code. On the contrary, contextualized coding assistants can be used to generate text and code that better fits some specific requirements. These "specialized" chatbots are based on the Retrieval-Augmented-Generation (RAG) methodology: when prompting a contextualized LLM, the RAG system allows the model to "look for" the answer to the prompt by retrieving the relevant information from a given database. Essentially, contextualized AI assistants work by firstly retrieving information from the database and secondly by generating text or code in response to a given prompt. The database contains the "context" that the AI assistant uses to better taylor the answer to the prompt to the needs of the company, the research group or any other type of user. The topics discussed in this paper closely relate to the topics of the lectures and, more broadly, to the engineering of AI systems, where code needs to follow specific legal requirements, numerous standards or rules and is checked both statically and dynamically. Using AI chatbots can reduce human errors in the code and using specialized AI chatbots helps developers navigate the (sometimes) overwhelming number of rules and constrictions that need to be followed when writing code, many of which can be difficult to manage by humans alone.
The authors of the paper conducted a study to qualitatively understand if the use of specialized chatbots in a company can be beneficial to software developers. Moreover, they investigated whether such AI tools can easily be integrated in the workflow of the company's teams. The contextualized chatbot analyzed in the paper is called "StackSpot AI" and its influence in the workflow was tested on a group of employees of a software company. In particular the database accessed by the RAG contained exemplary code snippets, guidelines regarding repository commits, a list of software requirements, and other information.

The study proposed by the authors of the paper mainly consisted with a demonstration of the capabilities of StackSpot AI to the developers who participated in the study. Then, the participants were allowed to explore StackSpot AI by solving some tasks and interacting with the database accessed by the retrieval system of the AI assistant. The study ended with a group discussion. The results of the survey demonstrated some benefits of contextualized AI coding and a general perception of increased productivity. This was due to the ability of the chatbot to quickly generate precise and relevant code snippets, that satisfied the company's code standards and requirements. Moreover, StackSpot AI works as a chatbot, i.e. it allows users to interact and iteratively refine the provided code. On the other hand, some challenges were highlighted: some users found it challenging to understand which information to include in the database accessed by the chatbot's RAG system. Indeed, it appeared that a higher amount of excessively specific information contained in the database, would affect negatively the quality of the answers, which would require much more refinement and adjusting.

It is challenging to describe a larger AI-intensive software project where the paper's ideas could be of "practical" use. This is because the paper does not provide a potential solution to a practical problem (as, instead, in Paper 2, see later). Rather, the paper gives an overview of the benefits/challenges of integrating contextualized AI coding assistants in a software developer team, that might instead be used to working with general-purpose AI coding assistants. At the same time, in any large AI-centered software product, the use of chatbots to generate code is becoming more and more frequent. Each project's needs and requirements are unique, hence the use of ad-hoc coding could be highly beneficial. In my own research, it is desirable that the R code written to test hypothesis or cluster functional data adheres to certain standards that could make the code eligible to be potentially published on CRAN ([1], which collects publicly-available R packages) in the future. At the same time, it is often the case that new research builds up on top of previous results. In these cases, it is desirable that code design choices implemented by previous researches are maintained, to facilitate future researchers who will approach the same topic. To find a trade-off between all these (possibly conflicting) requirements, it could be beneficial to adapt a contextualized AI coding assistant by including in its database exemplary code snippets and all the necessary requirements. This is one way I could change my research project so that, over time, it better supports the paper's AI idea.

5.2. Paper 2: Generating and Verifying Synthetic Datasets with Requirements Engineering

The problem addressed in this paper ([4]) concerns data augmentation in training ML models, specifically for object detection in images. Object detection involves classifying and locating instances of specific classes in images. For example, given a picture of a cat, the model must recognize the presence of a cat and locate all instances by surrounding them with bounding boxes. For such models to perform well, the training dataset must contain sufficient examples of each class. When this is not the case, a common solution is to artificially augment the dataset with images of under-represented classes. These synthetic images can be generated by suitable generative models. The issues tackled in the paper relate to my research project because performing simulations based on synthetic dataset is often a good way to showcase the capabilities of a mathematical framework aimed at performing hypothesis testing for functional data (as described in the Introduction).
What makes a generated image "good" or valid enough to be included in the augmented dataset? Is it possible to check if a procedure consisting in generating images is reliable enough to be used to augmented a dataset which contains under-represented classes? The authors of this paper propose a procedure to check the validity of the augmented images by embedding requirements engineering into the data augmentation procedure. As indicated during the lectures, requirement engineering is a crucial step when developing a software product, especially when the core of the project is a ML model and pipeline.
The authors of the paper assume that the task at hand is object detection for images. They assume that the dataset lacks sufficient instances of certain classes. Therefore, the training dataset needs to be reliably augmented in a suitable way. Assume that the under-represented class is the class "person". The augmentation procedure detailed in the paper is as follows: a set of requirements (also called "specifications") are first formulated, for example: "the object detector shall detect a person if the person is standing close to the image" or "the object detector shall detect a person if the person is sitting down in the bottom-right corner of the image". Note that these requirements are both requirements for the generated images and requirements for the so-called "downstream model", i.e. the object detector. Then a prompt is formulated, for example: "generate the picture of a person close-up". This prompt is fed to a generative model (Flux). The generated image is then manually checked to see if it follows the rules stated in the prompt and the given requirements. Finally, an "auxiliary", pre-trained object detector (YOLO) is fed the generated image to see if it can spot the person in the image. The idea is that if a pre-trained model works well with the generated synthetic image, then (hopefully) the downstream model (which will be trained with the augmented dataset) will have a chance to be correct as well. If the image passes both the manual check and the object-detector actually identifies what it should identify, then the image is included in the augmented dataset.

The paper's ideas and my own WASP research can fit together in the following AI-intensive software project. Assume that some stakeholders would like a product that, given brain scans (as images) of children, detects anomalous portions that could be linked with cancer. This means that the ML/AI model is the core of the software product to be developed. Assume that this problem is mathematically framed as described in the Introduction, i.e. as a statistical problem where the goal is to test if each image contains anomalous portions that could be linked with cancer. In this case H_0 can be a template image of a cancerous mass. Since, luckily, brain cancer doesn't present itself often in children, the ML model wouldn't have enough images to reach a satisfactory level of accuracy. Reliable data augmentation could be a solution to this problem. In this case, it could be beneficial to adapt the data augmentation procedure described in the paper. This would require the use of an "image segmenter" as downstream model. As a result, the final model would be guaranteed to have been trained on "realistic" images of brain cancer.

Finally, it could be possible to tweak my research project to support the paper's AI-engineering idea and the challenges it addresses. The main point of the paper is that it is important to make sure that synthetic data are valid and realistic. The idea proposed by the authors tackles this by integrating requirements engineering in the data generation step. Essentially, each generated image is accepted in the augmented dataset if it is deemed "realistic" enough. This would make the model pipeline more trustworthy to the project's stakeholders. This implementation would require some changes to the methodology provided in the paper, however it is an important point to keep in mind when performing simulations, even in the context of functional data analysis.

6. Research Ethics and Synthesis Reflection

My research project is mostly based on mathematical statistics, rather than software engineering. For this reason, I chose papers that did not require very specific background knowledge.
Throughout my search, I did not find any misleading titles/abstracts. My search was limited to the accepted papers from any CAIN conference that were classified as ”long papers”.
To ensure originality, no LLM was used to create the content of this report. All ideas contained in this report are expressed in my own words.

References

[1] The comprehensive r archive network (cran). https://cran.r-project.org. last accessed: 23/08/2025.

[2] Hopsworks. https://www.hopsworks.ai. last accessed: 23/08/2025.

[3] Gustavo Pinto, Cleidson De Souza, Thayssa Rocha, Igor Steinmacher, Alberto Souza, and Edward Monteiro. Developer experiences with a contextualized ai coding assistant: Usability, expectations, and outcomes. In *Proceedings of the 2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN 2024)*, pages 81–91. Association for Computing Machinery, 2024. Discovered via CAIN 2024 conference page at: https://conf.researchr.org/details/cain-2024/cain-2024-call-for-papers/9/Developer-Experiences-with-a-Contextualized-AI-Coding-Assistant-Usability-Expectati (last accessed: 23/08/2025).

[4] Lynn Vonderhaar, Timothy Elvira, and Omar Ochoa. Generating and verifying synthetic datasets with requirements engineering. In *Proceedings of the 2025 IEEE/ACM 4th International Conference on AI Engineering – Software Engineering for AI (CAIN 2025)*, pages 212–221. IEEE/ACM, 2025. Discovered via CAIN 2025 conference page at: https://conf.researchr.org/details/cain-2025/cain-2025-call-for-papers/11/Generating-and-Verifying-Synthetic-Datasets-with-Requirements-Engineering (last accessed: 23/08/2025).