Agnes Koschmider, Judith Michael (eds.): EMISA Workshop 2021
CEUR-WS.org Proceedings 30

Software Design Patterns for AI-Systems

Marius Take[1], Sascha Alpers[1], Christoph Becker[1], Clemens Schreiber[2]
and Andreas Oberweis[1,2]

Abstract: Well-established design patterns offer the possibility of standardized construction of software systems and can be used in various ways. The systematic use of design patterns in the field of Artificial Intelligence (AI) Systems however, has received little attention so far, despite AI being a popular research area in recent years. AI systems can be used for a wide variety of applications and play an increasingly important role in business and everyday life. AI systems are becoming more complex however, the actual machine learning (ML) task comprises only a small part of the total source code of a system. In order to maintain a clear and structured architecture for such systems and to allow easy maintenance, standardized elements should be reused in the design. This paper describes possible applications of well-known design patterns in AI systems to improve traceability of the system design.

Keywords: Artificial Intelligence, Machine Learning, Design Patterns

1 Introduction

Artificial Intelligence Systems are composed of many different components. The source code for the actual machine learning task comprises only a small part of the total source code [Sc15], this is illustrated in Figure 1. Regarding the nomenclature, this paper distinguishes between AI systems and ML algorithms to illustrate the different levels. The ML-algorithm represents solely the code for learning and prediction, whereas the AI-system comprises additional components required to embed ML algorithms into a system. One way to support the integration of ML algorithms in AI systems is the use of design patterns. This approach is further investigated within this paper. Design patterns can facilitate the integration of ML algorithms and ensure certain properties of the system behavior, such as data quality. AI systems in particular have all the maintenance problems of traditional systems plus an additional set of ML-specific problems, such as a significant lack of strong abstractions [Sc15]. Design patterns are a promising approach for this challenge. They support the standardized design of software systems, while increasing the clarity of the implemented system architecture.

In the following, Section 2 describes how design patterns are currently used in relation to AI systems. In Section 3 we show application scenarios for two selected and adapted

[1] FZI Forschungszentrum Informatik, Haid- und Neu-Straße 10-14, 76131 Karlsruhe, {alpers, christoph.becker, oberweis, take}@fzi.de
[2] Karlsruhe Institute of Technology, AIFB, Kaiserstraße 12, 76131 Karlsruhe, {andreas.oberweis, clemens.schreiber}@kit.edu

design patterns. Further design patterns, as well as a more detailed description of the AI system components and related work can be found in the extended version of the paper [Ta21]. The paper ends with a conclusion and an outlook on possible further steps.

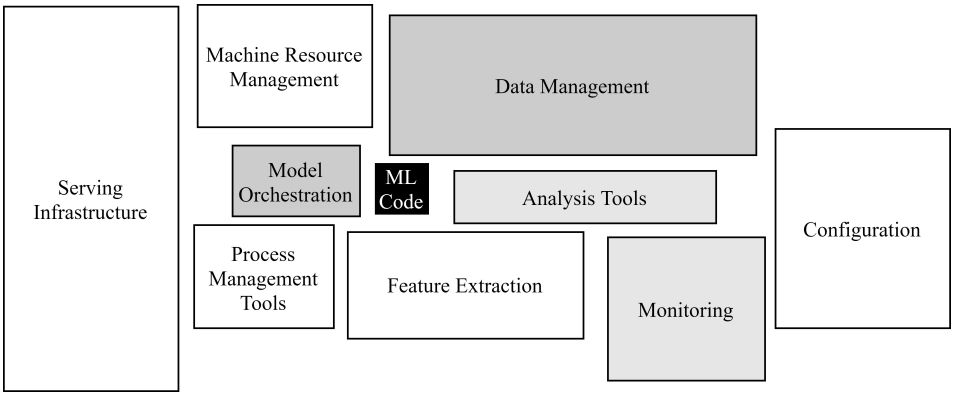

Figure 1: Overview of the components of an AI system, based on [Sc15]

2 Related work

Washizaki et al. [Wa19] observed that at the time of their publication (October 2019) no papers exist that list, classify, and discuss design patterns for AI systems. Research conducted in advance of this paper did not reveal any further approaches. However, the authors also postulate, backed by a survey of a smaller group of developers, that such patterns are needed. Ozkaya furthermore shows in her article [Oz20] that engineering AI-enabled systems is different from other systems. In addition, there would be a research gap regarding development strategies for AI systems or the transferability of existing general approaches.

Design patterns in general enjoy a high level of attention in software development [CW16]. At the same time, AI systems are becoming increasingly important in software development, as evidenced by the high annual market growth. For this reason, there are also approaches to combine both fields. Machine learning methods can be used to identify design patterns within program code [e.g. DTR16], and design patterns can be used to support the development of AI systems [e.g. Sc15, Wa19]. Other works use design patterns for a unified representation of theories and systems, as in [HT19] where a set of patterns is presented that can be used to describe techniques for combining learning and reasoning systems in a unified way (used for example in [FH20]). In this paper, we examine the development and structured design of AI systems. The existing work on this approach differs from this paper, mainly in the type of patterns considered. In the following section, we will focus on the GoF-design patterns (Gang of Four) [Ga94] and demonstrate how they can be used for the development of AI systems. These patterns represent individual

elements of object-oriented programming such as classes, methods, attributes and their relationships in contrast to the patterns presented in [Sc15], which merely provides textual pattern description. Furthermore, this paper does not consider concrete systems in order to make architectural recommendations in this respect, but rather demonstrates how patterns can be used to improve particular aspects of AI systems, such as data quality (management) and ML model selection.

3 Software design patterns for AI systems

This section presents exemplary applications and adaptations of individual GoF patterns for the field of AI systems. The selection of the patterns *adapter*, *factory method*, *observer*, *strategy* and *state* in [Ta21] is based on their popularity [CW16], applicability and successful use in existing AI projects. However, the adaptation of design patterns with regard to AI requirements is not limited to the aforementioned patterns but could also be extended to further patterns. For reasons of limited space, only two design patterns are presented below, a more detailed description of these, as well as the presentation of the adapted patterns a*dapter*, o*bserver* and *state* can be found in [Ta21].

3.1 Factory method

The design pattern *factory method* belongs to the class of creational patterns. As the name suggests, creational patterns are used to create new objects. This is done by calling a method `factory_method()` (see Figure 2). The `factory_method()` returns a new *product object* with each call [Ga94]. In the context of AI systems, the design pattern *factory method* can be used to create training data objects. For machine learning models from the field of supervised learning, training elements require both a training object (for example, a picture or a natural language sentence) as well as a label that indicates the class which the object belongs to. Both attributes are mandatory. To achieve similar predictive accuracy for each existing class and each desired feature, it is necessary to ensure that each class and each feature is represented equally often in the training data. Features here represent subclasses between which no distinction is made in terms of classification. However, differences in the features must not lead to different prediction accuracies. Adjustments to the well-known *factory method* design pattern provide the means to address this requirement at an architectural level (see Figure 2).

The pattern displayed in Figure 2 shows the creation of N*M training objects, where N is the number of existing classes and M the number of different features. To ensure a balanced set of training data one object is created for every possible class and feature combination. With the adjusted *factory method* pattern, there will always be the same amount of training data for all classes and features, thereby postulating the fundamentals for achieving the same prediction accuracy. This provides leverage towards addressing the recently more frequently discussed ethical requirement of *equity* at the architectural level.

Publications such as [ZS18] address this issue, which appears in many current AI applications. As described in [ZS18], accuracies regarding the classification of images of people can vary, for example, with respect to skin color or gender due to unevenly distributed training data. The potential equality of the prediction accuracy with regard to the individual classes and features is usually still dependent on the actual training data content. However, by using the adjusted *factory method* pattern, the equal distribution of training data can already be ensured at the architecture level. In general, this pattern is suitable for any use case where a fixed distribution of the frequency of certain classes in the training data is to be enforced.

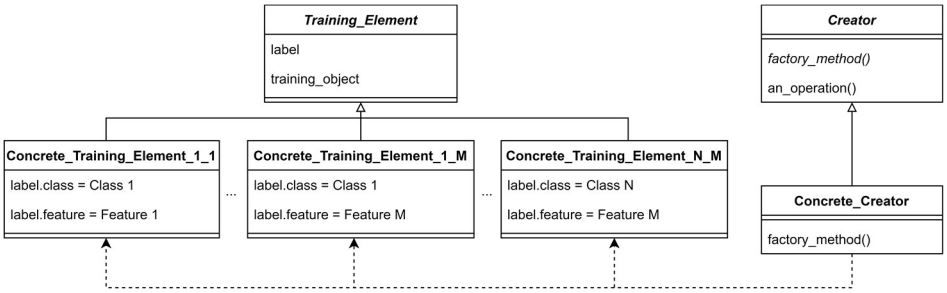

Figure 2: Adapted design pattern factory method for the AI domain

3.2 Strategy

The *strategy* design pattern is categorized as a behavioral pattern and allows the context (the AI system components) to be implemented separately and independently from the actual ML algorithm. The context class holds different strategy objects [Ga94]. Transferred to the AI context, the individual strategies can represent different ML models. Each ML model must implement fixed necessary methods. The context class does not need any information about the concrete implementation or the type of ML models. Functionalities are abstracted from the rest of the system. In addition to the integration of in-house developed ML models, the integration of external models is of interest in many use cases. The abstraction is implemented by the design pattern *strategy*. The actual external model integration is implemented using the *adapter* pattern [Ga94]. The newly created design pattern is visualized in Figure 3.

Using different ML models in AI systems can be useful in many areas, especially in decision-critical application domains. If an AI system is used either to check the creditworthiness of potential borrowers or to diagnose serious illnesses, a minimum degree of certainty regarding the predictions is required. By using several different ML models for the same request, statements can be made about the reliability of the individual results. If the different models provide different predictions, a high degree of uncertainty is assumed, but if all models produce the same result, the statement is more reliable. The combination of several neural networks (*ensembles of NNs* or *deep ensembles*) is used for

example by Lakshminarayanan et al. in [LPB17] for uncertainty estimation. The design pattern *strategy* provides a flexible way of implementing such *deep ensembles*. Developers of the AI system do not need knowledge about the concrete processes in the model but can easily integrate different ML models in a system by using the methods `train()` and `predict()`.

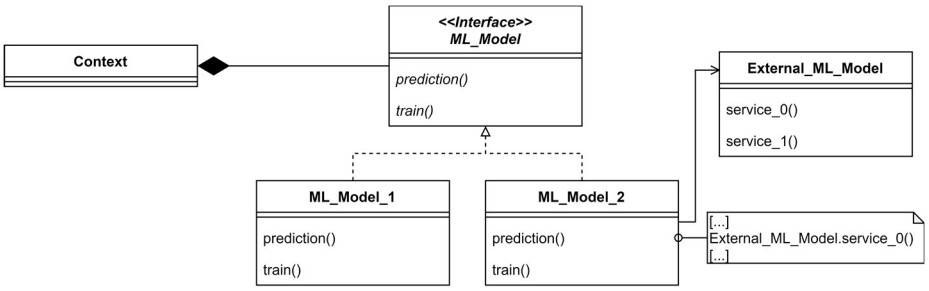

Figure 3: Use of the design pattern strategy in the AI context

4 Conclusion

This paper investigates the use of design patterns for the development of AI systems. The patterns discussed in this paper serve as means for a structured and clear design of AI systems. However, assumptions and conclusions made in this paper should be further investigated in subsequent work. As part of our further research activities, we plan to extensively investigate the application as well as the advantages and disadvantages of the adapted patterns. However, with regard to the use cases described in this paper a general benefit is expected. For example, a variant of the adapted design pattern *strategy* presented here was successfully used in the "Intelligente-Diagnostik"[3] project to design an AI model management platform.

As shown in [Ta21] further patterns have been adjusted to serve a neat and clean design of AI Systems. Nevertheless, there remains a large number of patterns that should be investigated to determine their suitability for the development of AI systems. In later work, the scenarios and adaptations presented here could thus be supplemented with further design patterns. We plan to further explore the use of design patterns to ensure ethical principles (especially in the context of medical diagnostic systems) at an architectural level. With the help of design patterns, ethical requirements could be implemented in a more standardized way. Ethical conditions and possible solutions based on design patterns could thus also be examined in future work.

[3] The Intelligent Diagnostics project deals with AI based skin cancer detection based on images from a special optics. The joint project of five research institutions is funded by the state of Baden-Württemberg. Further information: https://www.intelligente-diagnostik.de

References

[CW16] Czyczyn-Egird, D.; Wojszczyk, R.: Determining the Popularity of Design Patterns Used by Programmers Based on the Analysis of Questions and Answers on Stackoverflow.com Social Network. In Computer Networks: 23rd International Conference: Brunów, Poland, p. 421–433, 2016.

[DTR16] Dwivedi, A. K.; Tirkey, A.; Ray, S. K.: Software design pattern recognition using machine learning techniques. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON): Singapore, p. 222–227, 2016.

[FH20] Fill, H.-G.; Härer, F.: Supporting Trust in Hybrid Intelligence Systems Using Blockchains. In AAAI Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice (AAAI-MAKE), Vol. 1, 2020.

[Ga94] Gamma, E. et.al.: Design Patterns: Elements of Reusable Object-Oriented Software, 1st ed., Addison-Wesley, 1994.

[HT19] Harmelen, F. v.; Teije, A. t.: A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems. In Journal of Web Engineering, Vol. 18 1-3, p. 97-124, 2019.

[LPB17] Lakshminarayanan, B.; Pritzel, A.; Blundell, C.: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In Advances in Neural Information Processing Systems 30, p. 6402–6413, 2017.

[Oz20] Ozkaya, I.: What Is Really Different in Engineering AI-Enabled Systems?. In IEEE Software, Vol. 37, No. 4, p. 3-6, 2020.

[Sc15] Sculley, D. et.al.: Hidden Technical Debt in Machine Learning Systems. In Advances in Neural Information Processing Systems 28, p. 2503–2511, 2015.

[Ta21] Take, M. et.al.: AI-system development with design patterns, Extended Version of this paper. https://url.fzi.de/wp-ai-pattern, 2021.

[Wa19] Washizaki, H. et.al.: Studying Software Engineering Patterns for Designing Machine Learning Systems. In 10th International Workshop on Empirical Software Engineering in Practice (IWESEP), 2019.

[ZS18] Zou, J.; Schiebinger, L.: AI can be sexist and racist - it's time to make it fair. In Nature 559, pp. 324-326, 2018.