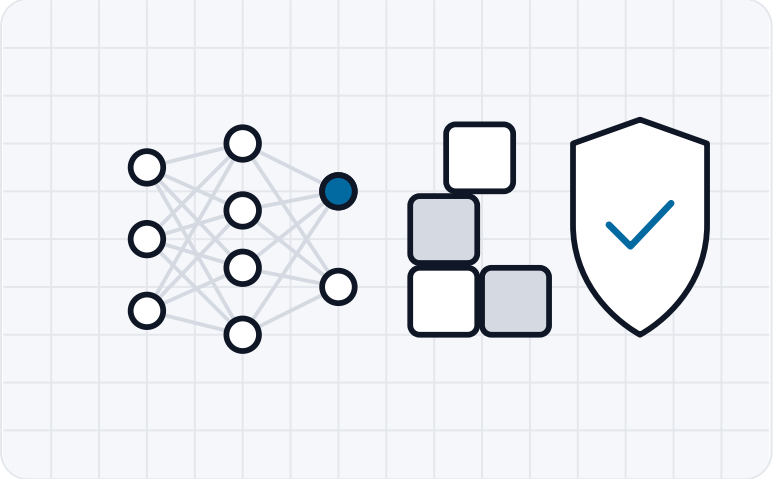
A-SIT Technology

E-Government Innovation Center

29 April 2023 Overview · AI · Privacy & Crypto · 16 min read

Theoretical analysis of the approach to the development of AI-based services

Bianca Danczul



Authors	Bianca Danczul
Version	2023-04-25
Source	https://technology2.a-sit.at/en/publications/theoretical-analysis-of-the-approach-to-the-development-of-ai-based-services/
Note	Machine translation.

Key findings

- Security measures are well established in classical software development, but not yet in AI development.
- No/low-code tools carry risks: vulnerabilities, privacy concerns, compliance risks, lack of transparency and dependence on tool providers.
- The black-box nature of no/low code makes the models hard to understand and offers few testing options.
- High code offers more control and flexibility, but requires more technical know-how and is more time-consuming.

- The choice between low/no code and high code depends on know-how, resources and use case; security must never be neglected.

Artificial intelligence is continuously used in both the public and private sectors with an increasing tendency. This is further driven in particular by the availability of so-called low/no code development frameworks. However, increased use combined with sometimes little understanding due to the black-box nature results in increased requirements, especially for security and privacy. Measures that address the aforementioned requirements and contribute to the development of secure software are already well established in traditional software development, but not in AI development.

Complicating matters is the fact that development options such as no/low and high code in turn harbor quite different security requirements. The goal is to address the different development variants and to highlight special security requirements, as well as to discuss their differences.

1. Introduction

The growing use of artificial intelligence across industries and sectors poses a growing challenge for security and privacy. It is therefore of great importance to take measures that ensure secure AI development. While traditional software development already has established measures to ensure security and privacy, these are not yet as widespread in AI development. To complicate matters, AI development has different approaches such as no code, low code and high code, each of which brings different security requirements [1].

No code here denotes a user-friendly approach to AI development that requires no programming skills or technical expertise and thus allows users to apply pre-built AI models easily. However, it also carries strong risks regarding understanding, which can lead to security implications. Low code is more complex than no code and requires the use of simple scripting languages, but still little technical know-how and background knowledge about the models used. Finally, high code is the classic variant of AI development and offers the highest degree of flexibility and adaptability, but requires a high level of technical expertise.

Because of the large differences between the various AI development variants, it is important to address the specific security requirements. This article therefore explains the different approaches to AI development, highlights the specific security requirements and compares the approaches in a discussion.

2. No and Low Code

No-/low-code AI development is an approach to creating AI models and applications that requires neither programming skills nor a high level of technical expertise [2], and the two terms are often used together in the literature. This approach is intended to enable laypeople to create AI models quickly and easily, without having to learn complex programming languages or hire a team of experts. The following briefly describes the general procedure in low-/no-code development and discusses advantages, disadvantages and security risks.

2.1. Areas of application and overview

Low/no-code AI development tools offer drag-and-drop interfaces, pre-built modules and automated workflows that allow users to create AI models without having to write code [3]. These tools typically offer a range of AI functions, such as natural language processing, image recognition and predictive analytics. The most popular low-/no-code AI development tools include Google AutoML[1], IBM Watson Studio[2] and Microsoft Azure Machine Learning[3]. Specific areas of application include the following:

- Chatbots and virtual assistants : Low-/no-code AI development tools such as Chatfuel[4] based on GPT-4 or ManyChat[5], as well as ChatGPT[6], offer the possibility to create chatbots and virtual assistants without programming skills. These tools use pre-built modules for natural language processing and conversation management and can be integrated into popular messaging platforms or websites.

- Business intelligence and data analysis : Microsoft Power BI[7] and Tableau[8] allow users to create data visualizations and dashboards without requiring programming skills. These tools use pre-built connectors and data models and allow data fields to be created via drag and drop in order to build interactive visualizations.

- E-commerce : Tools such as Zyro[9] and Wix[10] allow the creation of websites and online shops with AI-supported product recommendations, personalized content and automated order processing. Membrace[11] enables AI-supported content moderation. These tools use pre-built modules for managing product catalogues, payment processing and shipping logistics.

- Image recognition and object detection : Low-/no-code AI development tools such as Google Cloud Vision[12] and Clarifai[13] allow users to create models for image recognition and object detection without programming skills. These tools use pre-built models and APIs for image analysis and can be integrated into various platforms and applications.

- Language translation : Translation services such as Google Translate[14] or DeepL[15] offer the possibility to use AI-supported language translation. These tools use pre-built models and APIs for natural language processing and can be integrated into various platforms and applications.

As has been shown, there are many areas of application for low/no code. This is also reflected in the fact that it is already used in various domains, as shown for example by Rokis and Kirikova [2]. In addition to the e-commerce and business intelligence areas already mentioned, these include customer relationship management, entertainment, content management systems, robotic process automation and medicine. Further applications are business process automation in manufacturing, applications in public healthcare, recommender systems and the monitoring of machine learning models themselves [2]. They are also already used in smart cities [4, 5], in the field of function block diagram programs [6] and in supply chains [7].

2.2. Advantages and disadvantages

The following briefly lists some general advantages and disadvantages of low-/no-code AI development before the generic development process and the security implications are addressed.

The advantages are [8]:

- Ease of use: Even users without technical expertise can easily create AI models and applications with the help of low-/no-code AI development frameworks, without specialist knowledge. This makes AI accessible to a wider group of users and allows companies to use AI technology without having to hire specialists.
- Speed and efficiency : Low-/no-code AI development tools offer pre-built modules and automated workflows that allow users to create AI models quickly and efficiently.
- Scalability : Low-/no-code AI development tools are designed to process large amounts of data and can easily be scaled up or down depending on project needs. This makes it easier to create AI models and applications that can be used on different platforms and devices without additional development work.
- Cost efficiency : Low-/no-code AI development tools can be more cost-efficient than hiring a development team, as they eliminate the need for specialised technical staff and reduce the time and resources required for development work.

Of course, low-/no-code AI development frameworks have not only advantages but also disadvantages, which are briefly listed below:

- Limited customisation options [2, 3] : Such tools rely heavily on pre-built modules and automated workflows, which greatly limits the degree of control and customisation available to users. This can make it difficult or impossible to create complex AI models and applications.
- Limited functionality [3] : Compared with low-code or high-code AI development tools, low-/no-code AI development tools offer only very limited functionality, which can likewise make it difficult to create complex AI models and applications.
- Security aspects [3] : Low-/no-code AI development tools can, among other things, have security vulnerabilities that inexperienced users cannot detect because of a lack of background knowledge, and can lead to many risks, which are listed separately later.
- Dependence on tool providers : Low-/no-code AI development tools are often provided by third parties, which means that users are heavily dependent on ongoing support and updates. This in turn can lead to uncertainty and risks if, for example, the provider discontinues support for the tool.
- Lack of understanding [2] : Understanding the data basis and the models is a basic prerequisite for a successful machine learning model. Because of the black-box nature of low-/no-code frameworks, however, users often cannot comprehend the result, which can lead to incorrect results and problems in troubleshooting.
- Lack of testing options [2]: There are currently insufficient options for actually testing low-/no-code frameworks comprehensively, which can lead to undesired side effects.

As has just been shown, low-/no-code AI development tools offer accessibility, speed, efficiency, scalability and cost efficiency, but also have limitations regarding customisation, security concerns, dependence on tool providers and limited functionality.

2.3. Development process for low code / no code

According to Rokis and Kirikova [2], the development process of low-code/no-code applications comprises several phases, which they identified in the course of a literature review:

- Requirements analysis : This is the first and continuous step throughout the entire development process and essential for the success of the application, as this step influences platform selection, validation methods and user requirements. In this phase, one is strongly dependent on the offerings of the tool providers, which can be a significant challenge. In addition, changing requirements later is difficult, as the tools may not meet the new requirements. On the positive side, creating a first prototype with low-/no-code development frameworks is often quick, which simplifies close coordination with customers.

- Planning: The feasibility study, including the selection of a suitable platform, falls into this phase. Because of the large number of available platforms and development frameworks, this phase is difficult, which is why various comparison criteria can be considered, such as graphical user interface, security, interoperability, reusability, scalability, application support or deployment support. Further criteria mentioned are the specification of roles and users, support for requirements specification and verification, support for testing and validation, and the availability of artificial intelligence components. In this phase it is also essential to decide whether the dependence on the platform provider is to be accepted.

- Application design : In this phase the design is defined. This includes, for example, configuring the data schema for the application, such as creating entities and defining relationships, constraints and dependencies. Further content includes defining the user interface and defining user access and security management, that is, broadly speaking, the architecture, scalability and modularity. Challenges arise in particular in extensibility, which can be difficult to impossible, and in interoperability owing to the lack of standards. Scalability can also lead to problems.

- Development : This phase comprises the implementation of various operations for interface elements and the implementation of business logic rules and workflows. It also includes the integration of external services, which allows external services to be used directly in low-/no-code software development via application programming interfaces (APIs) provided by third parties. Problems arise here in particular from incomplete or incorrect documentation, the lack of learning resources or non-intuitive interfaces. In addition, in some areas knowledge of software development is sooner or later needed after all, which can overwhelm users with little technical expertise. Finally, the limited flexibility of functionalities and user interfaces poses further challenges for users, alongside limited debugging options.

- Testing : Various tests are carried out in this phase. Challenges arise in this phase in particular because of missing documentation. Many development platforms do offer test tools, but these are often limited in their functionality and offer few possibilities for carrying out non-functional tests.

- Deployment : This phase covers the provision of the application. In practice, problems arise in particular with the configuration and accessibility of the application, which are made worse by incomplete platform documentation. Version control is also described as a problem.

- Maintenance : The last, continuous phase is maintenance. Even though no-/low-code platforms are generally regarded as easy to maintain, debugging, alongside other functions such as monitoring or anomaly detection, can lead to challenges.

2.4. Security implications

Low-/no-code AI development tools have specific security implications that should be taken into account when using these tools. The most important ones are listed below:

- Vulnerability to cyber attacks [1, 8] : Low-/no-code AI development tools can have security vulnerabilities that can be exploited by cyber attackers. These vulnerabilities can be present in the pre-built modules and workflows used to create AI models and applications, or in the tool itself. Users without technical expertise may not be aware of these vulnerabilities or may lack the knowledge needed to detect and fix them.

- Privacy concerns [8] : Low-/no-code AI development tools may require access to sensitive data such as personal information, financial data or confidential business information in order to create AI models and applications. If the tool is not properly secured, there is a risk that this data will be disclosed to unauthorised parties.

- Bias [1, 8] : Low-/no-code applications can promote bias and disinformation.

- Compliance risks : Low-/no-code AI development tools may not comply with the relevant data protection laws and regulations. For companies using these tools, there is a risk of non-compliance, and legal and financial penalties may follow.

- Lack of transparency and control [8] : Low-/no-code AI development tools may not offer transparency about how AI models and applications are created, which can make it harder to identify and remedy security and privacy risks. Inexperienced users may also have no control over the AI models and applications created, which can make it difficult to ensure that these models and applications are secure and comply with the relevant regulations.

- Dependence on tool providers [8] : Low-/no-code AI development tools are often provided by third parties, which means that users depend on the tool provider for security updates and patches. If the tool provider does not offer regular updates or support, the AI models and applications created with the tool may be endangered by security vulnerabilities.

In summary, low-/no-code AI development tools have security implications that should be taken into account when using them. These include vulnerability to cyber attacks, privacy concerns, compliance risks, lack of transparency and control, and dependence on tool providers. It is important to ensure that low-/no-code AI development tools are properly secured and comply with the relevant regulations in order to mitigate these risks, although in practice this can be a problem because of the lack of understanding. With proper use, however, low-/no-code development frameworks can even help to minimise security risks [8].

3. High Code

High code denotes the development of AI applications with classic machine learning (ML) applications. As this topic has already been covered in several of the author's studies (including [9]), only a brief overview is given in the following in order to be able to follow the subsequent discussion.

In AI-based software development, a general software development cycle should be followed, which usually includes phases such as requirements gathering, design, implementation, testing and maintenance, but phases such as initiation, concept development, requirements analysis, technical design, development and testing, implementation, deployment and optimisation are also common. It is worth noting that in development the phases of training, modelling, application and inference are essential and require a deep understanding of statistical modelling, data analysis and machine learning concepts. Risks and security concerns arise in particular during the collection, analysis and processing of data, and because of AI decisions. The risks can be eliminated by comprehensive testing and the use of special techniques such as data augmentation or model regularisation.

4. Conclusion and discussion

The development of AI systems is primarily about creating and training models in order to make predictions or decisions on the basis of input data. Traditionally, this process was very technical and required special knowledge of programming and data science. With the emergence of the division into low/no-code and high-code development, however, this process has also become more accessible to non-technical users.

Low/no-code development for machine learning refers to the use of platforms and tools that require little or no programming to create ML models. These platforms often use drag-and-drop interfaces and pre-built components to make it easier for users to create models without having to write code themselves. Some popular examples of low/no-code ML development platforms are Google AutoML or IBM Watson Studio.

High-code development for machine learning, by contrast, refers to the traditional process of creating ML models with programming languages such as Python or R. In this approach, code is written to implement algorithms, process data and fine-tune models. In high-code ML development, developers have more control over the model creation process and can customise it better, which however also requires more technical expertise.

In general, low/no-code development for machine learning is often faster and more accessible than high-code development. It is therefore ideal for users without technical expertise or for people with limited resources. These platforms also often contain built-in functions for data preprocessing, model selection and evaluation, so that it is easier to create accurate models without extensive knowledge of data science.

However, low-/no-code platforms also have their limits. They may not offer the same degree of flexibility, customisation and security as high-code development, and have strong restrictions in functional scope, data volume or the type of models. In addition, understanding the models is very difficult because of their black-box nature, which can lead to additional security risks and few testing options.

High-code development for machine learning, by contrast, offers developers more control and flexibility in model creation. Developers can write specific code to implement advanced algorithms or to preprocess data in a way tailored specifically to the needs of their organisation. This degree of customisation enables

more accurate models, but requires a higher level of technical know-how and can by its nature be more time-consuming and resource-intensive.

In summary, both low/no-code and high-code development for machine learning have their advantages and disadvantages, and the decision for one of the two methods depends on factors such as the users' technical know-how, the available resources and the specific use case. For simple projects or those with limited resources, low/no-code development may be the best option, while high-code development may be necessary for more complex projects or those that require advanced algorithms or customisation. Security must never be neglected, however.

References

[1] G. Hurlburt, „Low-Code, No-Code, What's Under the Hood?", IT Prof., Jg. 23, Nr. 6, S. 4–7, 2021, doi: 10.1109/MITP.2021.3123415.

[2] K. Rokis und M. Kirikova, „Challenges of Low-Code/No-Code Software Development: A Literature Review" in Lecture Notes in Business Information Processing, PERSPECTIVES IN BUSINESS INFORMATICS RESEARCH: 21st international conference, Ē. Nazaruka, K. Sandkuhl und U. Seigerroth, Hg., [S.l.]: SPRINGER INTERNATIONAL PU, 2022, S. 3–17, doi: 10.1007/978-3-031-16947-2_1.

[3] M. Woo, „The Rise of No/Low Code Software Development-No Experience Needed?" (eng), Engineering (Beijing, China), Jg. 6, Nr. 9, S. 960–961, 2020, doi: 10.1016/j.eng.2020.07.007.

[4] Fitsilis und P. Fitsilis, Hg., Building on Smart Cities Skills and Competences: Human Factors Affecting Smart Cities Development, 1. Aufl. Cham: Springer, 2022.

[5] A. ElBatanony und G. Succi, „No-Code for Smart Cities" in Internet of Things, Building on Smart Cities Skills and Competences: Human Factors Affecting Smart Cities Development, Fitsilis und P. Fitsilis, Hg., 1 Aufl., Cham: Springer, 2022, S. 247–256, doi: 10.1007/978-3-030-97818-1_15.

[6] O. Ogundare, G. Q. Araya und Y. Qamsane, „No Code AI: Automatic Generation of Function Block Diagrams from Documentation and Associated Heuristic for Context-Aware ML Algorithm Training" in 2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR), Krakow, Poland, 2022, S. 191–195, doi: 10.1109/ICMERR56497.2022.10097820.

[7] S. K. Jauhar, S. M. Jani, S. S. Kamble, S. Pratap, A. Belhadi und S. Gupta, „How to use no-code artificial intelligence to predict and minimize the inventory distortions for resilient supply chains", International Journal of Production Research, S. 1–25, 2023, doi: 10.1080/00207543.2023.2166139.

[8] Z. Yan, „The Impacts of Low/No-Code Development on Digital Transformation and Software Development", 28. Dez. 2021. [Online]. Verfügbar unter: https://arxiv.org/pdf/2112.14073.

[9] Bianca Danczul, Sichere Entwicklung von KI-Diensten. [Online]. Verfügbar unter: https://technology.asit.at/sichere-entwicklung-von-ki-diensten/.

1. https://cloud.google.com/automl ↩

2. https://www.ibm.com/cloud/watson-studio ↩
3. https://azure.microsoft.com/en-us/products/machine-learning/ ↩
4. https://chatfuel.com/ ↩
5. https://manychat.com/ ↩
6. https://chat.openai.com/ ↩
7. https://powerbi.microsoft.com/en-us/ ↩
8. https://www.tableau.com/ ↩
9. https://zyro.com/de/ki ↩
10. https://de.wix.com/ ↩
11. https://membrace.ai/ ↩
12. https://cloud.google.com/vision?hl=de ↩
13. https://www.clarifai.com/ ↩
14. https://translate.google.com/?hl=de ↩
15. https://www.deepl.com/de/translator ↩