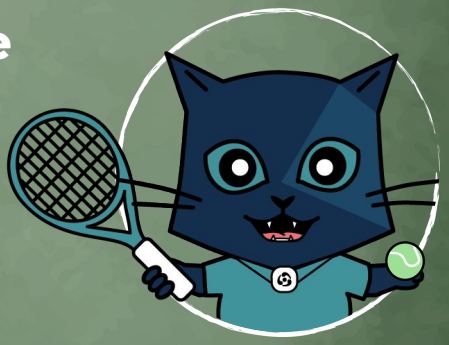
exactpro

ISTQB® AI Testing course

Chapter 7. Testing AI-Based Systems. Overview



Iosif Itkin, Iuliia Emelianova, Dmitrii Degtiarenko

Contents

7.1 Specification of AI-Based Systems **3**
7.2 Test Levels for AI-Based Systems **13**
7.2.1 Input Data Testing **15**
7.2.2 ML Model Testing **19**
7.2.3 Component Testing **21**
7.2.4 Component Integration Testing **23**
7.2.5 System Testing **25**
7.2.5.1 System Testing **26**
7.2.5.2 System Integration Testing **28**
7.2.6 Acceptance Testing **30**
7.3 Test Data for Testing AI-Based Systems **32**
7.4 Testing for Automation Bias in AI-Based Systems **37**
7.5 Documenting an AI Component **42**
7.6 Testing for Concept Drift **45**
7.7 Selecting a Test Approach for an ML System **49**

7.1 Specification of AI-Based Systems

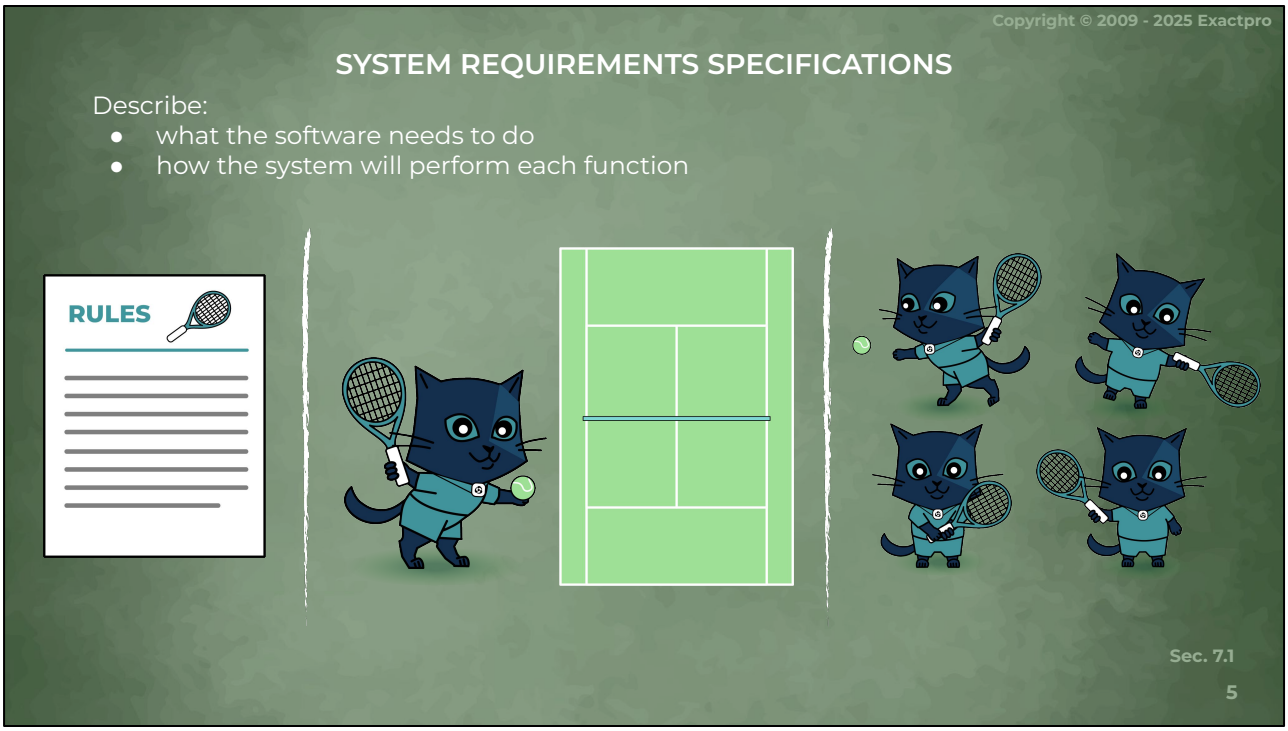
We have dived deep into the process of developing AI-based systems and have learned that we can use metrics to assess the model's quality, but what about evaluating other components of the system. What else can we look at when testing such complex systems? We have to keep in mind that the testing process starts with requirements and specifications.

DESIGN SPECIFICATIONS

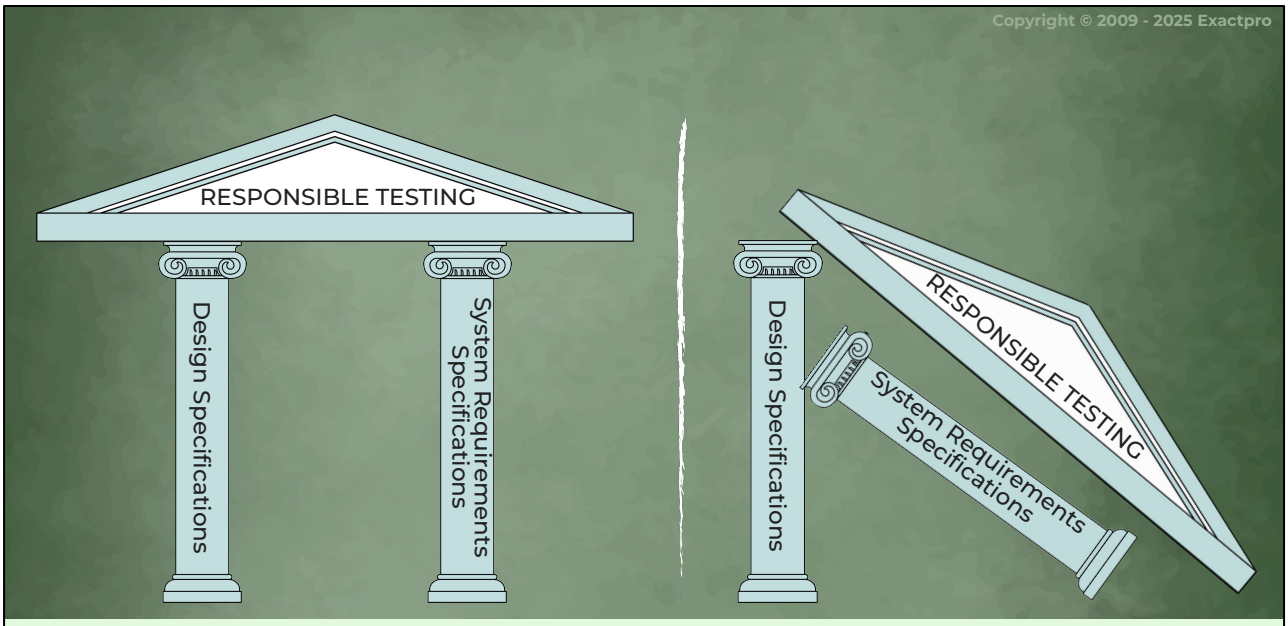
Describe:

- the software
- performance of the software
- user-software interactions

System requirements and design specifications are equally important for both conventional systems and AI-based systems. Design specifications describe your software and its performance as well as how the user should interact with it.

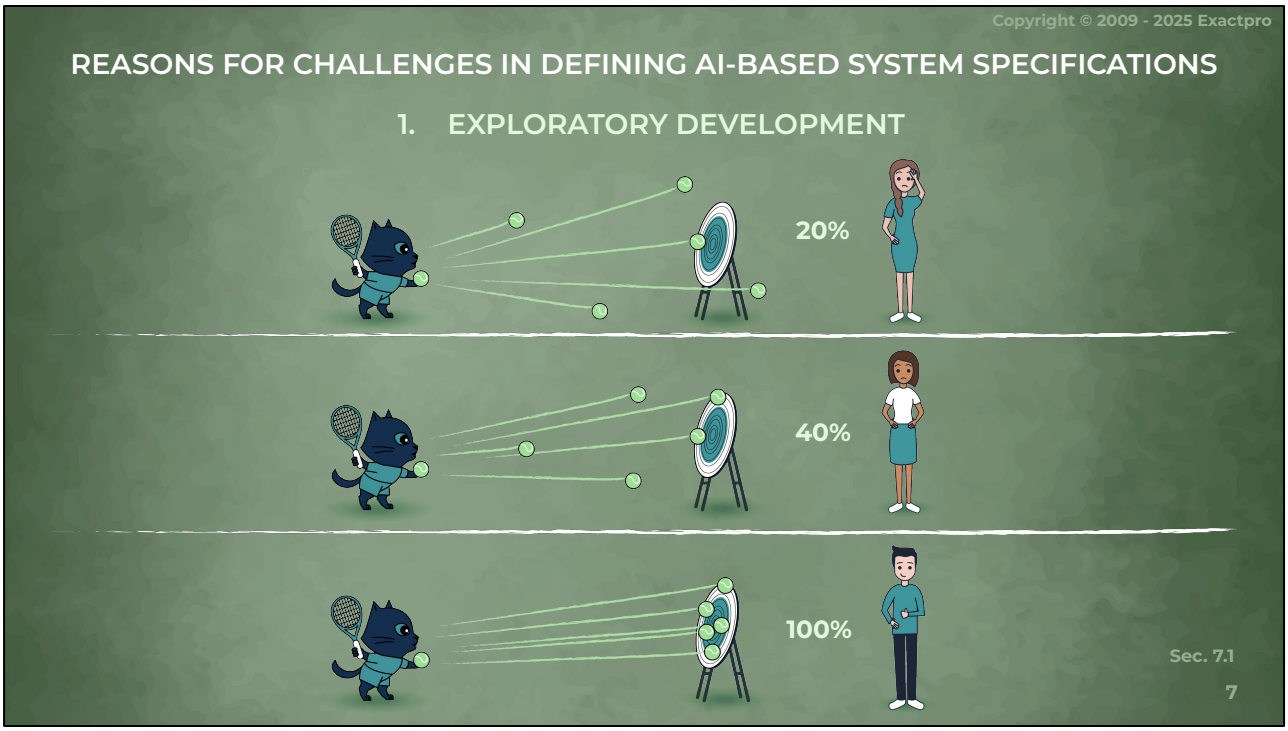


System requirements specifications on the other hand specify what the software needs to do, and how the system will perform each function. In a nutshell, system requirements provide the "what" of the software project, outlining its goals and functionalities, while design specifications provide the "how," offering detailed technical instructions on how to build the system to meet those requirements.



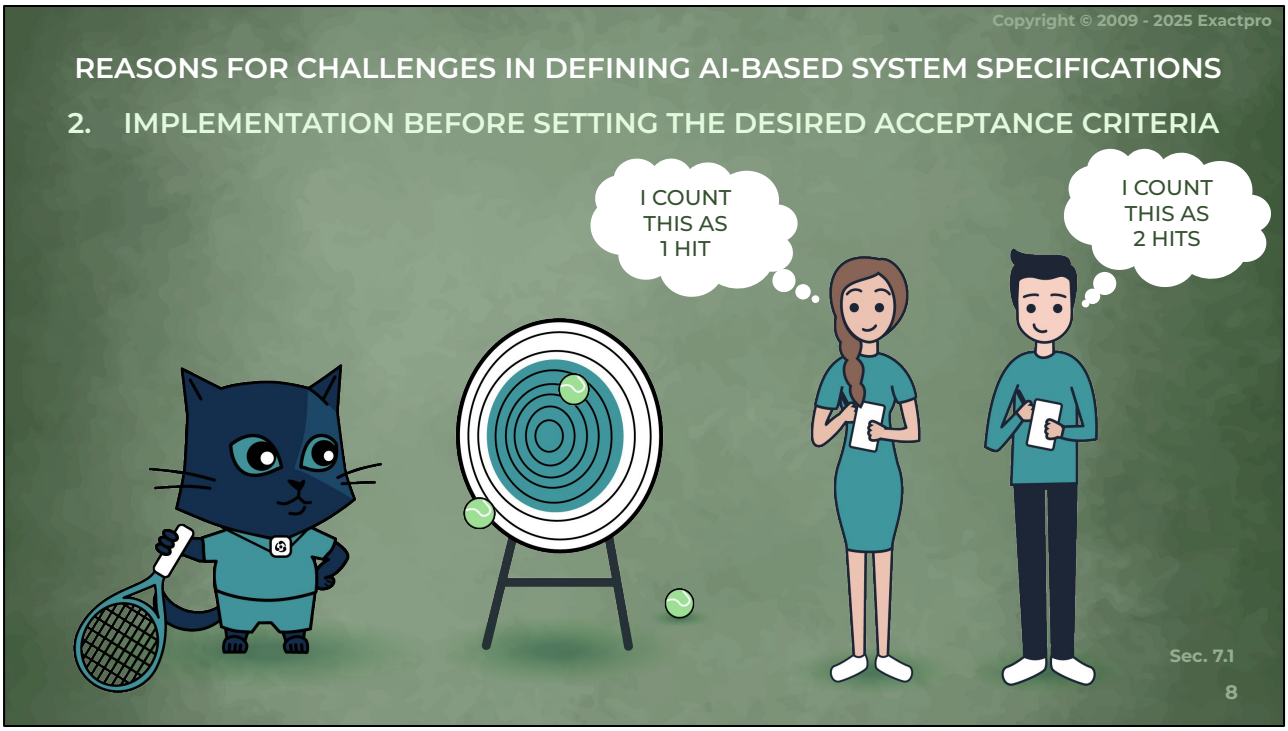
Test oracle problem: The challenge of determining whether a test has passed or failed for a given set of test inputs and state

The specifications provide the baseline for testers to check against in order to verify if system behaviour aligns with what's expected. However, if the specifications are incomplete and lack testability, this introduces a test oracle problem when we need to determine the correct output for a given input.

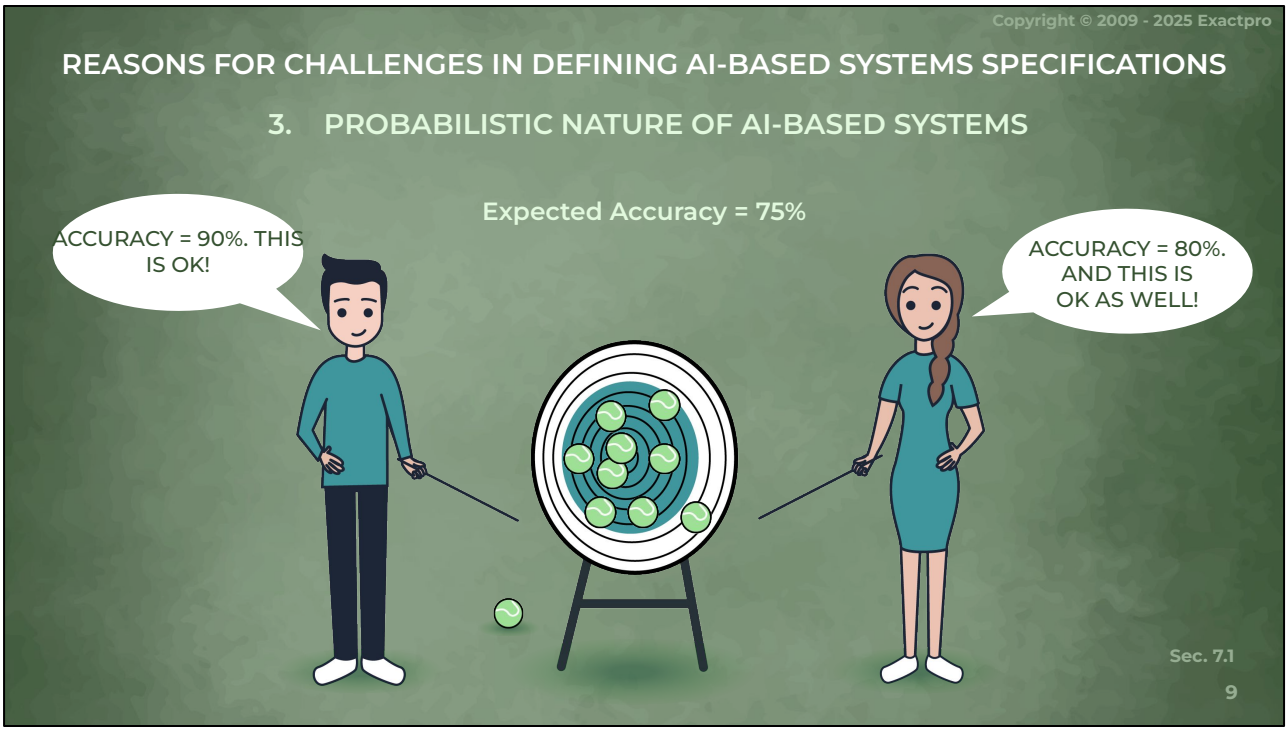


Let’s look at the reasons why defining AI-based system specifications can be challenging:

- In many AI-based projects, requirements are specified as high-level business objectives and predictions. Often, AI-based system projects start with a dataset, with the aim to establish which predictions can be obtained from that data. So development of these types of systems is exploratory. This approach, of course, is totally different from the conventional project when we determine the underlying logic from the get-go.



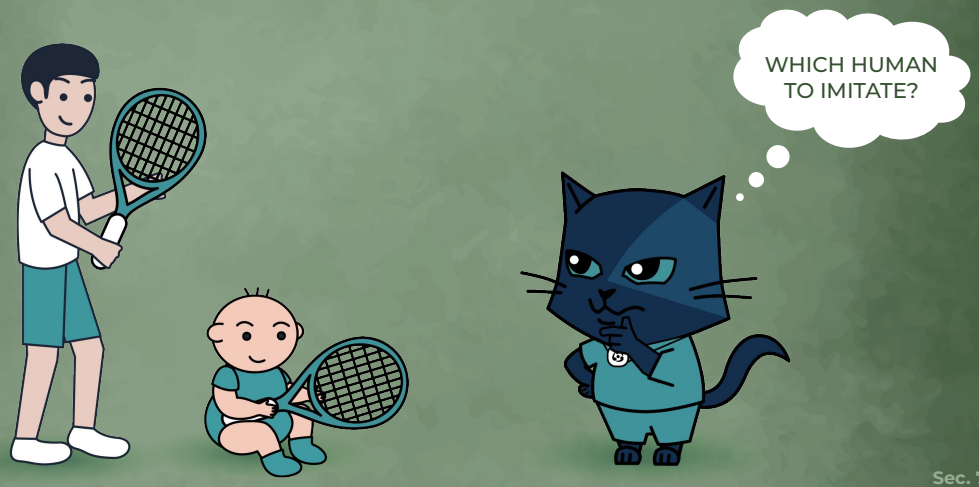
- The accuracy of the AI-based system is reviewed after we get independent test results. This often leads to inadequate specifications as implementation starts before we establish the desired acceptance criteria.



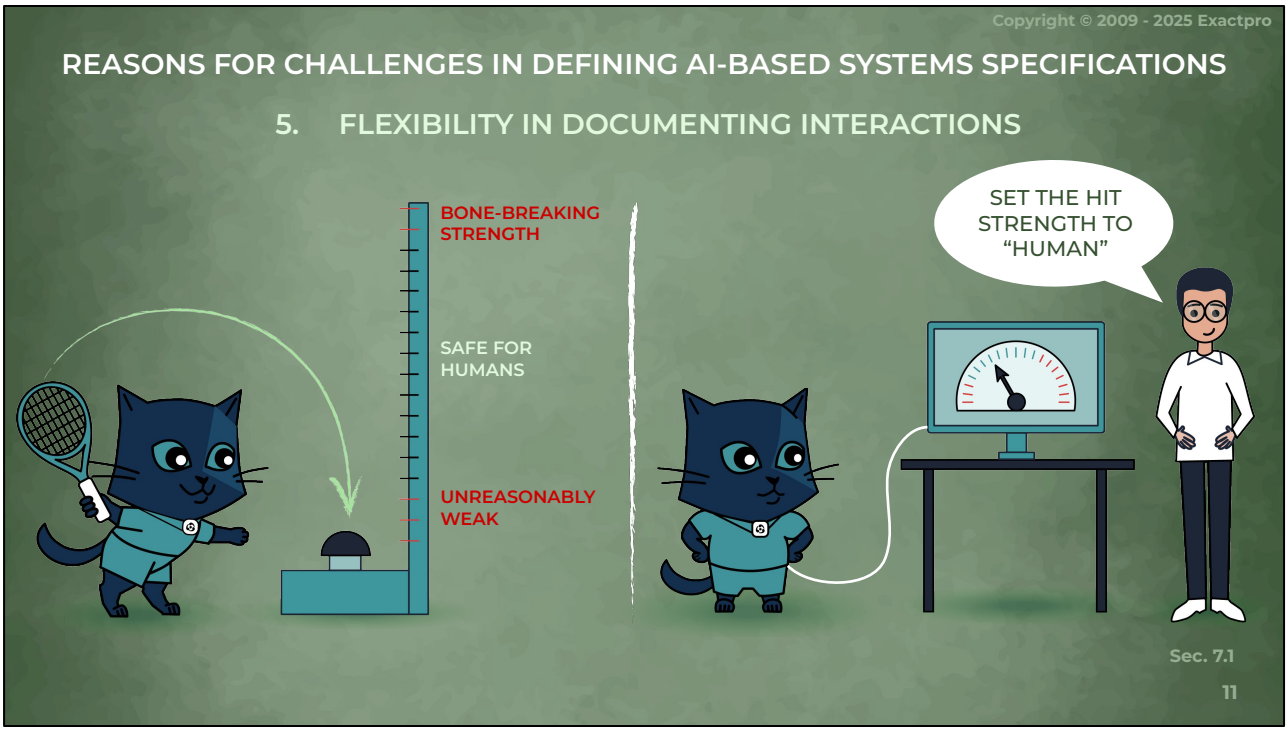
- The probabilistic nature of many AI-based systems can make it necessary to specify tolerances for some of the expected quality requirements, such as the accuracy of predictions.

REASONS FOR CHALLENGES IN DEFINING AI-BASED SYSTEMS SPECIFICATIONS

4. POORLY SPECIFIED BEHAVIOUR REQUIREMENTS



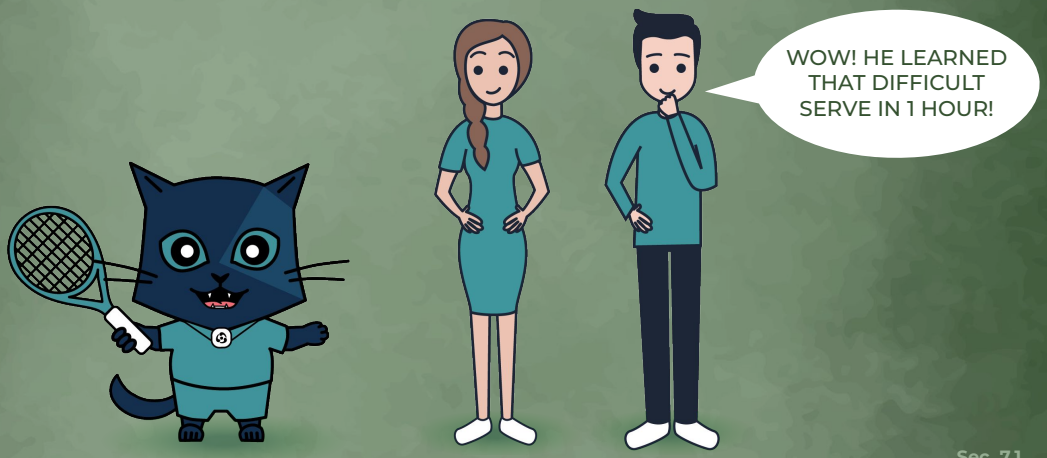
- Imitating human behaviour can lead to poorly specified behaviour requirements because all people are different and have different capabilities which exacerbates the process of defining a test oracle.



- Where AI is used to implement user interfaces through natural language recognition, computer vision or physical interaction with humans, systems need to show even more flexibility in documenting such interactions.

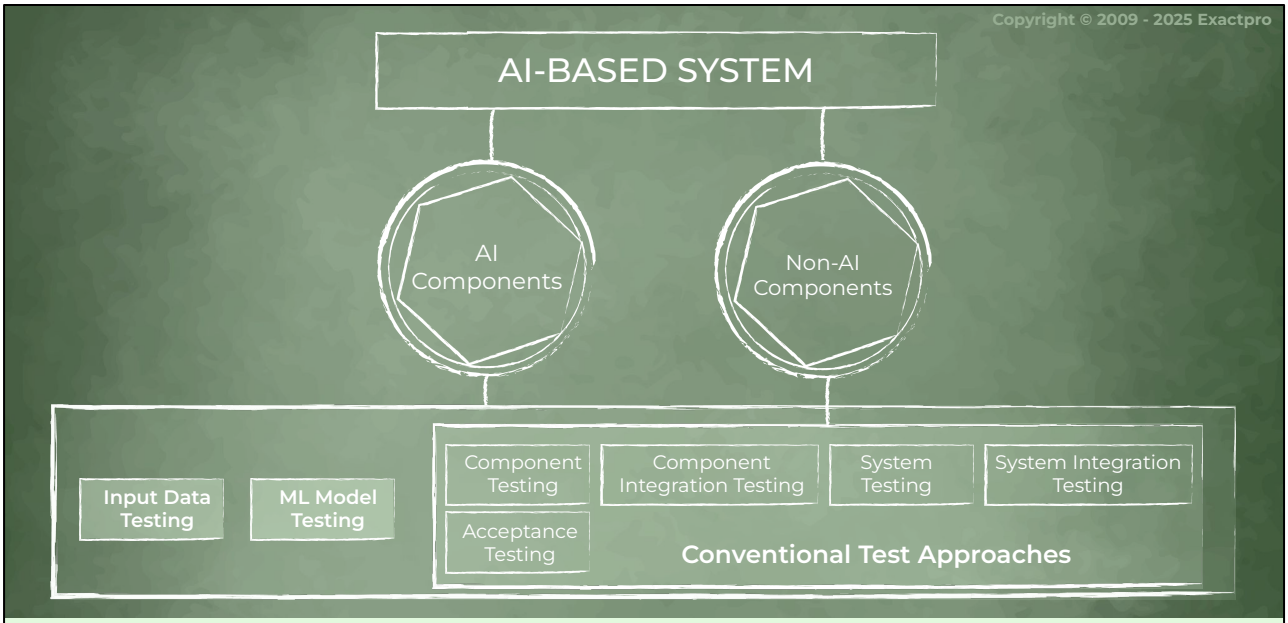
REASONS FOR CHALLENGES IN DEFINING AI-BASED SYSTEMS SPECIFICATIONS

6. DIFFICULTIES TO DEFINE AND TEST QUALITY CHARACTERISTICS



- Quality characteristics specific to AI-based systems, such as adaptability, flexibility, evolution, and autonomy, need to be included in requirements specification. But since these characteristics are relatively new, this can make it difficult to define and test them.

7.2 Test Levels for AI-Based Systems



AI component: A component that provides AI functionality

AI-based systems usually have AI and non-AI components. While we test non-AI components using conventional approaches, AI components may require something different. The biggest difference here is the inclusion of two new test levels to handle the testing of the input data and the models used in AI based systems. All test levels that tackle AI component testing need to be closely supported by data scientists and domain experts.

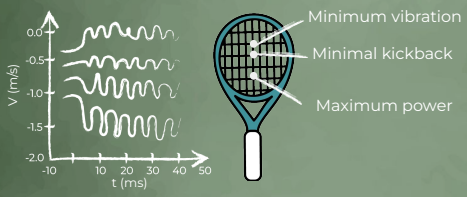
7.2.1 Input Data Testing

INPUT DATA TESTING includes:

REVIEWS

STATISTICAL TECHNIQUES

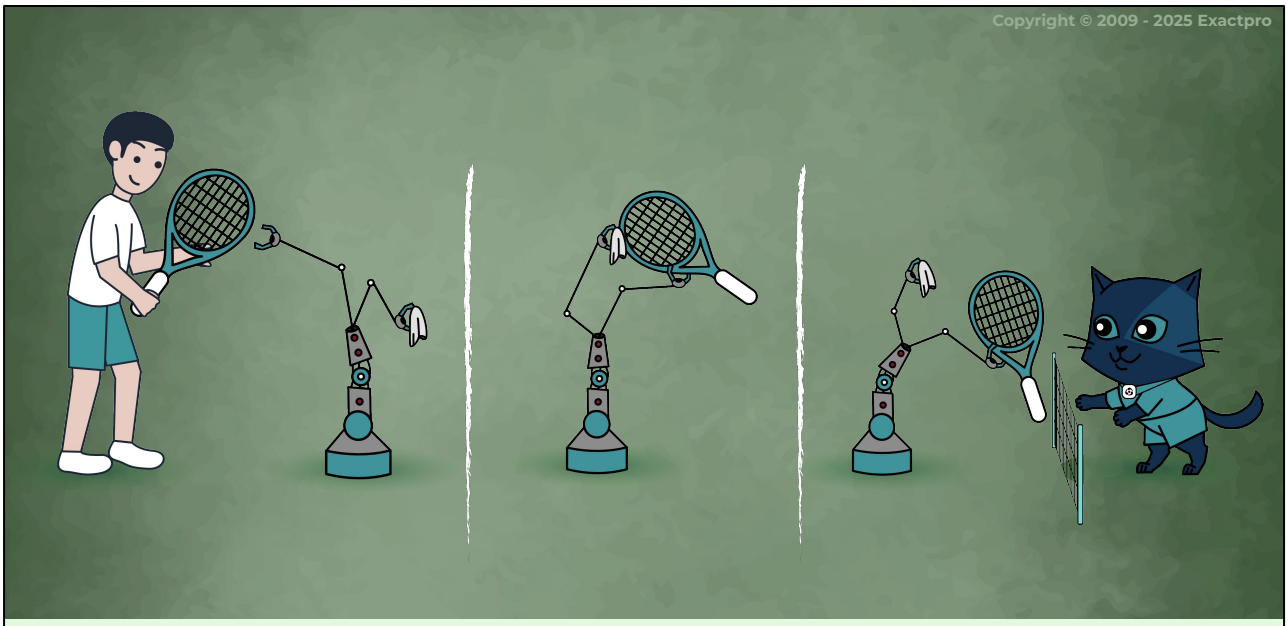
EXPLORATORY DATA ANALYSIS



STATIC AND DYNAMIC TESTING

Input data testing: A test level that focuses on the quality of the data used for training and prediction by ML models

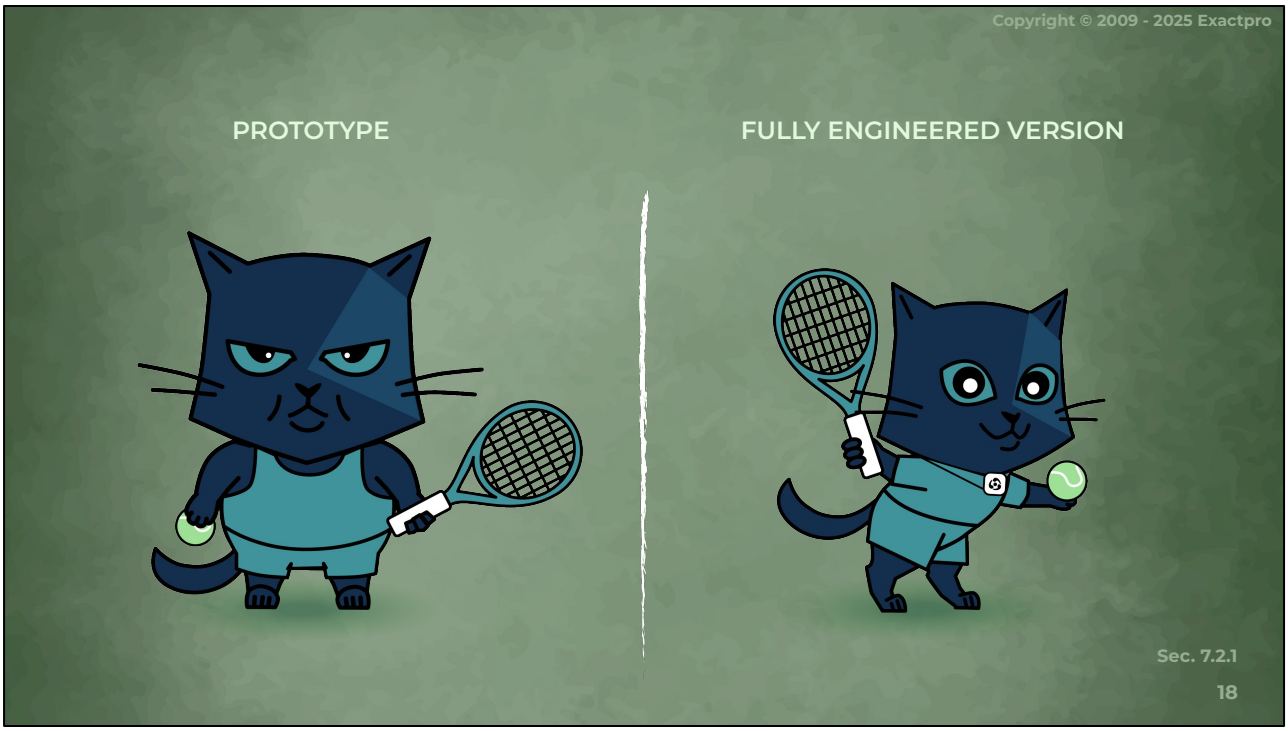
First comes **input data testing**. Its objective is to ensure the highest quality of training and prediction data. This includes reviews, statistical techniques, exploratory data analysis and static and dynamic testing of the data pipeline (by the way, the pipeline is what we call the software responsible for data supply).



Data pipeline: The implementation of data preparation activities to provide input data to support training by an ML algorithm or prediction by an ML model

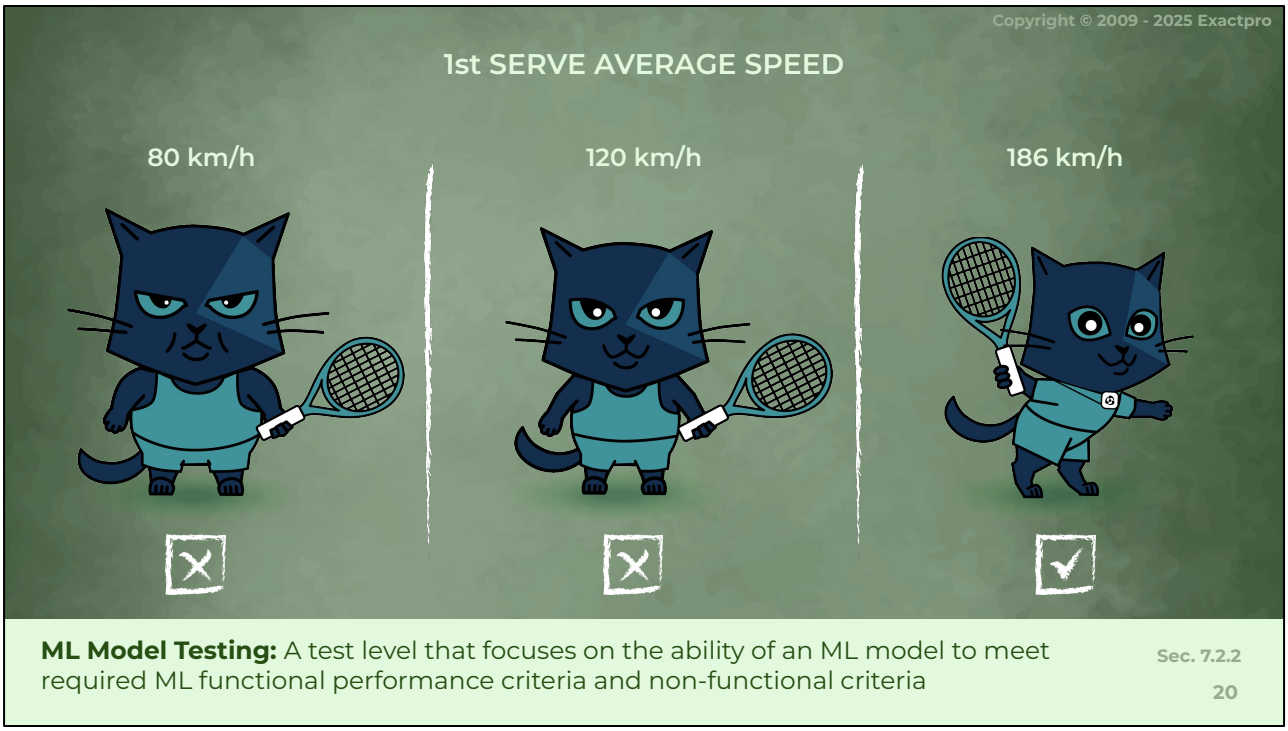
The **pipeline** takes data from production, cleans it, and sends it to the environments where this data is being analysed and the models are being trained.

For example, it may be historical data for the exchange rates or stock prices, or data describing financial transactions.



Usually, the data pipeline is made up of several components performing data preparation, and the testing of these components includes both component testing and integration testing. The training pipeline may be quite different from the data pipeline used to support operational prediction. For training, the data pipeline can be considered a prototype, compared to the fully engineered, automated version used operationally. Testing of these two versions may differ and should include testing the functional equivalence.

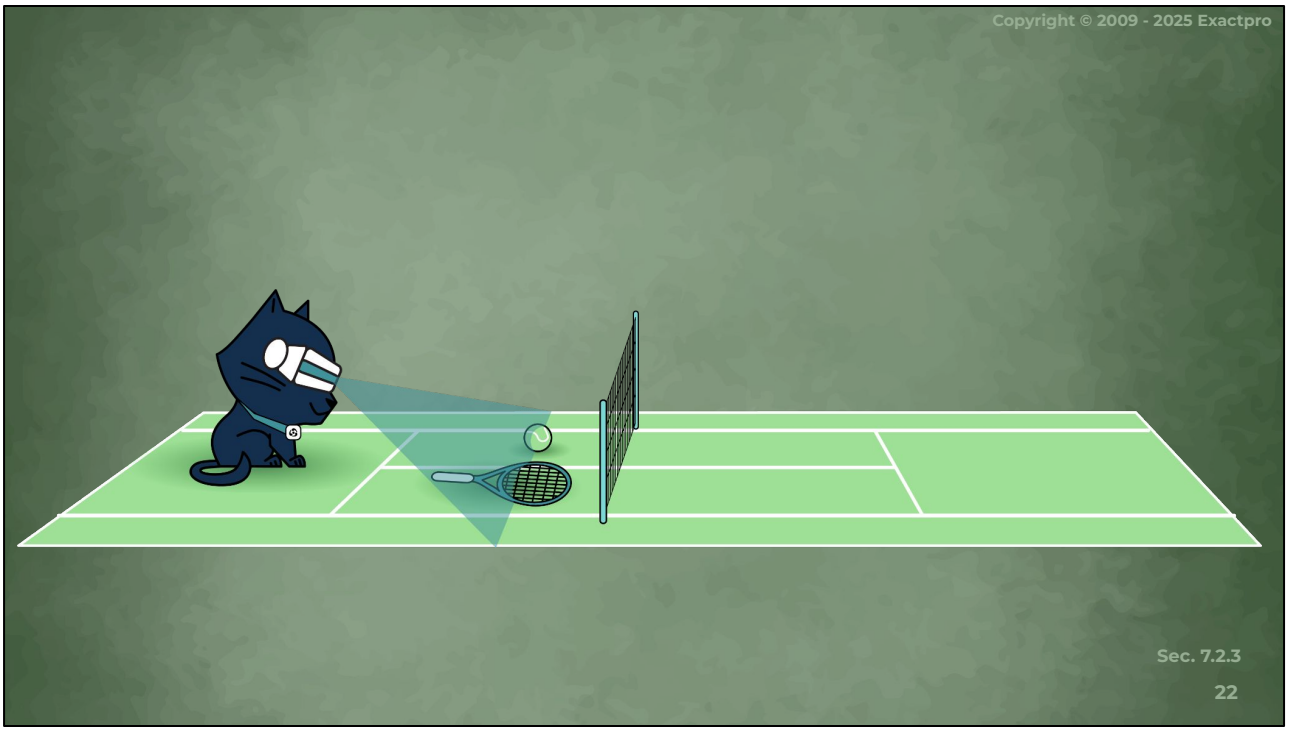
7.2.2 ML Model Testing



ML Model Testing: A test level that focuses on the ability of an ML model to meet required ML functional performance criteria and non-functional criteria

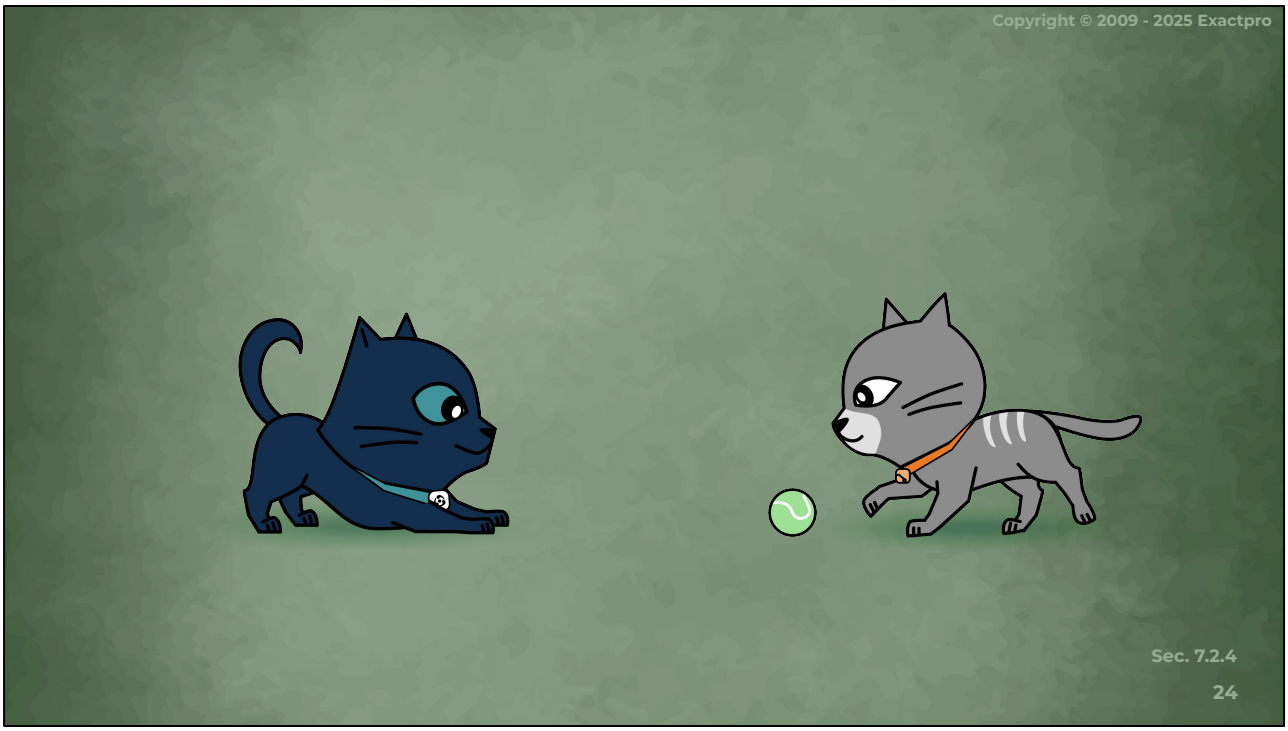
Then we have **ML model testing**. Its objective is to confirm meeting the specified performance criteria, including functional and non-functional ones. This step also has to make sure that the choices of ML framework, algorithm, model, model settings and hyperparameters are as optimal as possible. If applicable, ML model testing may also include testing to achieve white-box coverage criteria. The selected model is later integrated with other AI and non-AI components.

7.2.3 Component Testing



After that comes **component testing** for any non-model components, such as user interfaces and communication components.

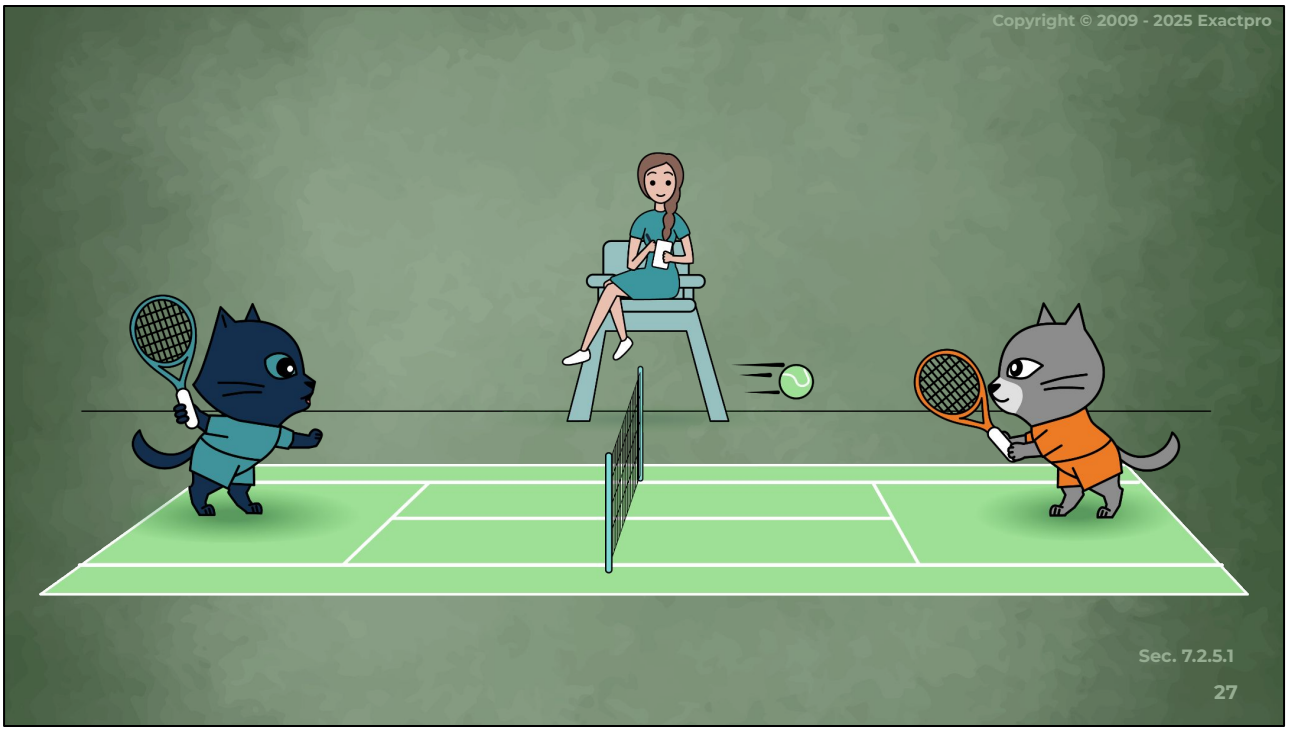
7.2.4 Component Integration Testing



Then there is **component integration testing** that ensures that AI and non-AI system components interact correctly. Here we make sure that the data pipeline inputs function as expected, and that model predictions are exchanged with the relevant system components. Where AI is provided as a service, it is normal to perform API testing of the provided service as part of component integration testing.

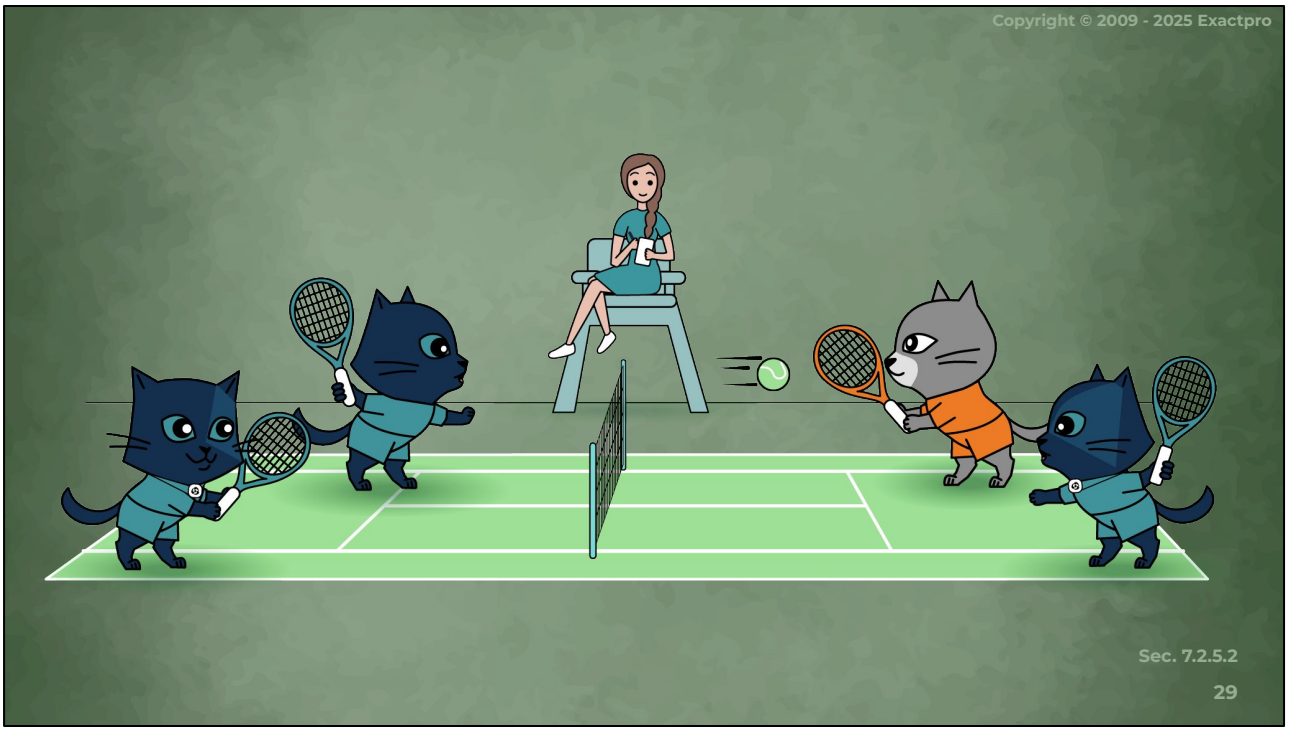
7.2.5 System Testing

7.2.5.1 System Testing



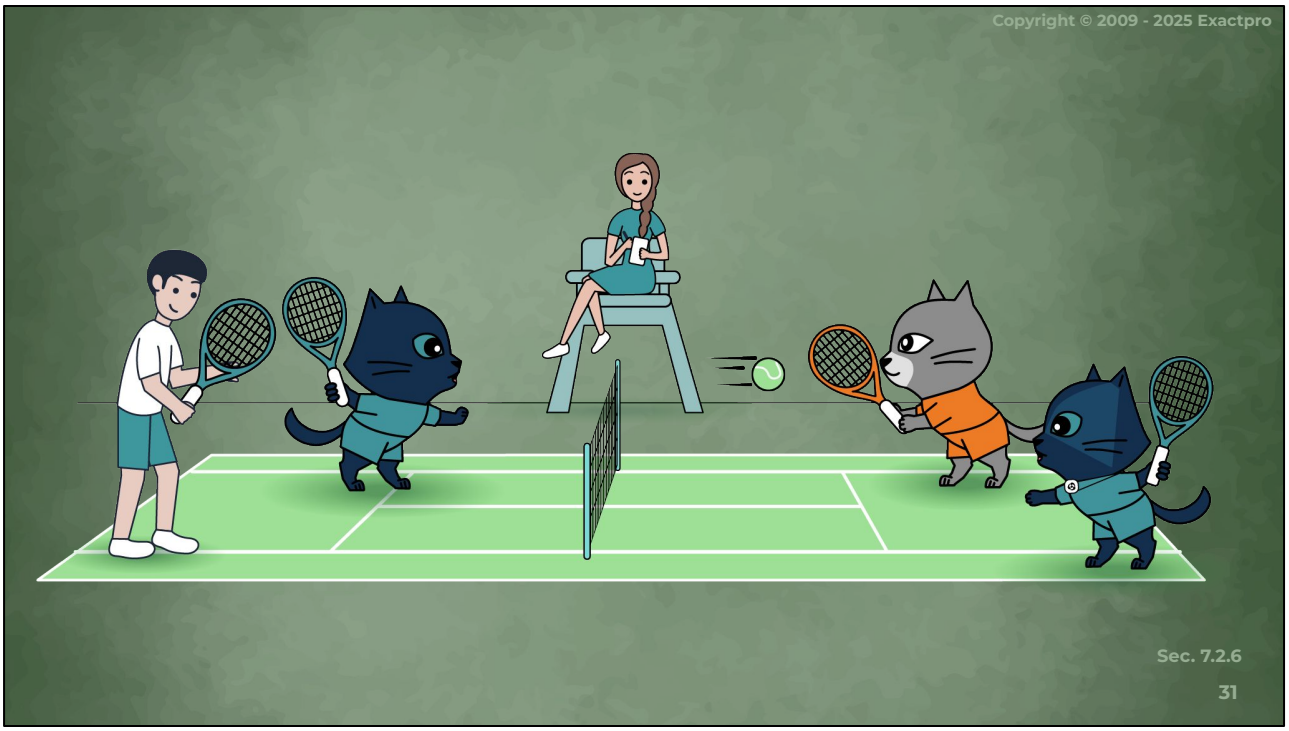
The next step is **system testing** which is conducted to make sure that the complete system (including both AI and non-AI components) performs as expected in a test environment that closely represents the operational environment from both functional and non-functional viewpoints. It can be done as field trials in the expected operational environment or simulator testing if test scenarios are hazardous or difficult to replicate in an operational environment. During system testing, the ML functional performance criteria are re-tested to ensure that the test results from the initial ML model testing are not affected when the model is added to the complete system. Here we also test many of the non-functional requirements as well.

7.2.5.2 System Integration Testing



System integration testing follows the system testing and verifies that multiple subsystems or systems work together as expected in an integrated environment. This testing focuses on ensuring seamless interaction and data flow between complete systems, such as the integration of an AI-driven system with external operational systems, hardware, or APIs. It evaluates both functional and non-functional aspects of the integration. System integration testing helps uncover integration issues that might only appear when all subsystems are connected.

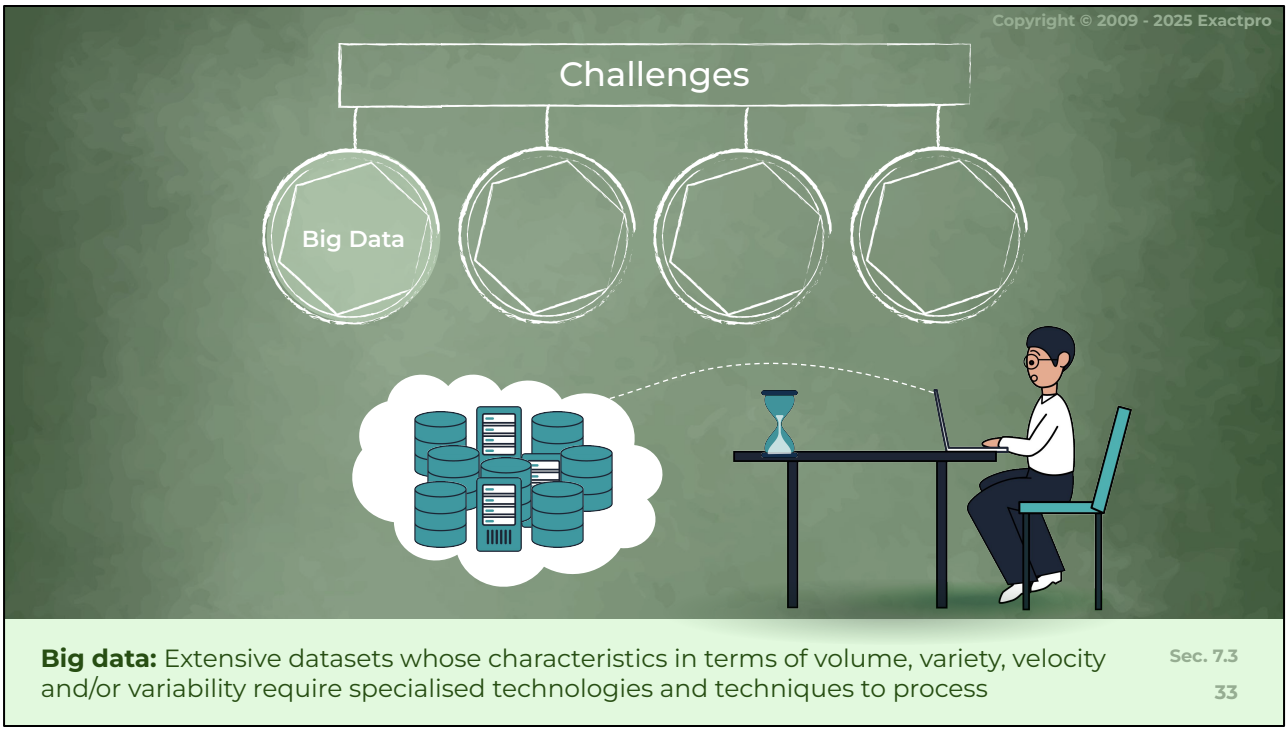
7.2.6 Acceptance Testing



And lastly we have **acceptance testing** that is used to determine whether the complete system is acceptable to the customer, even though this can be very complicated to establish for AI-based systems.

So, we have just looked at 7 levels of AI systems testing. But as with any conventional system, there could come certain risks and potential problems. Let's look at some of them…

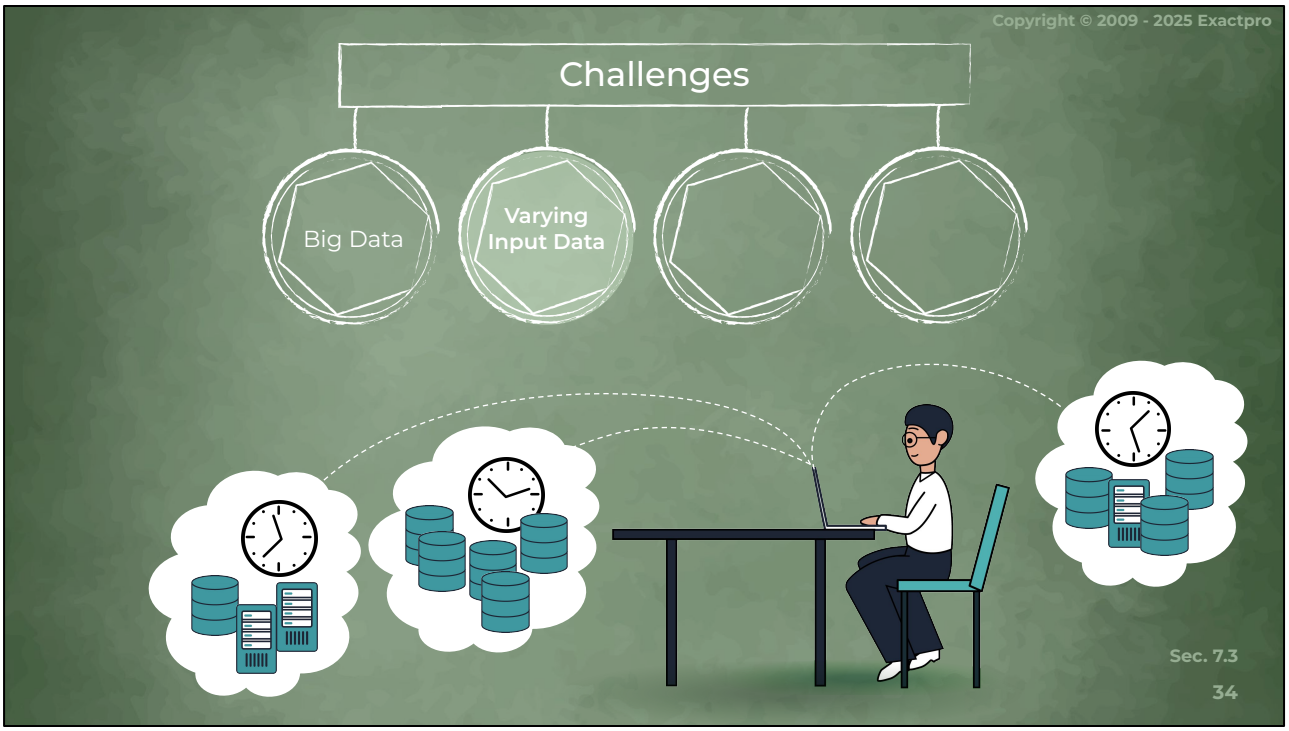
7.3 Test Data for Testing AI-Based Systems



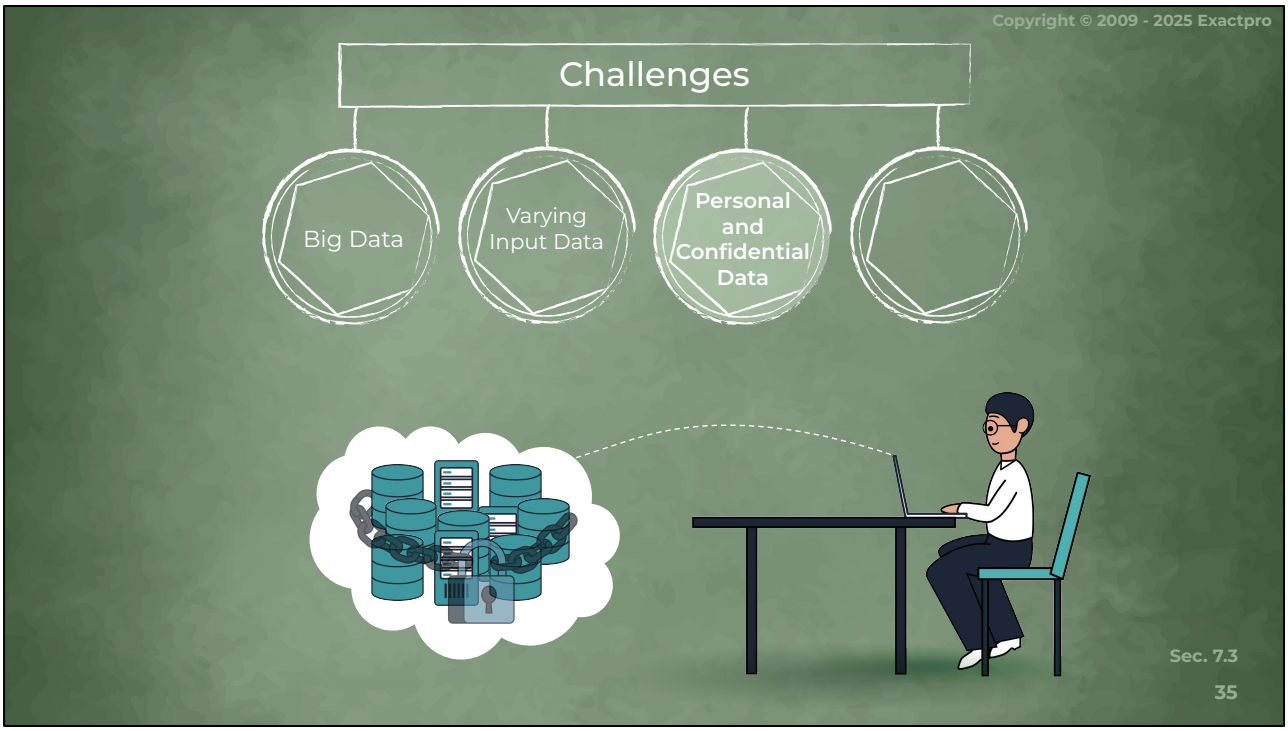
Big data: Extensive datasets whose characteristics in terms of volume, variety, velocity and/or variability require specialised technologies and techniques to process

There are several potential challenges in dealing with acquiring test data for AI-based systems, including the following:

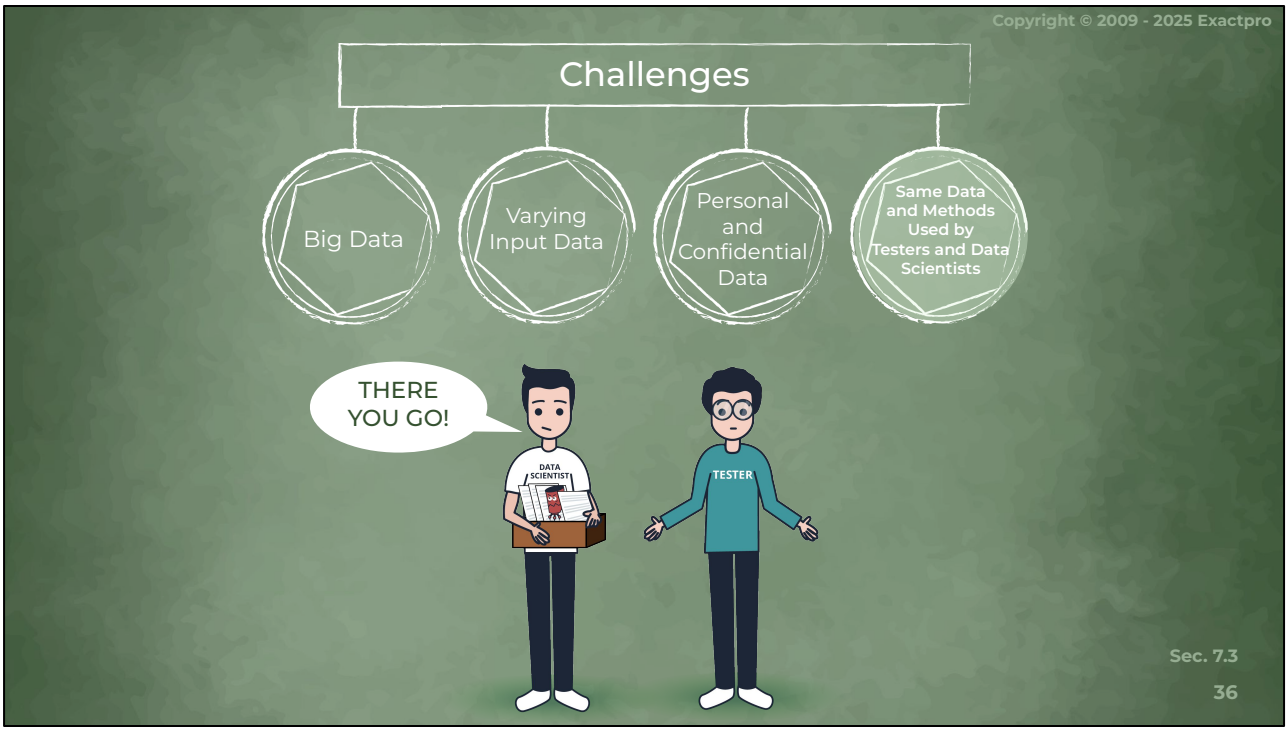
- **Big data** can be difficult to create and manage. For example, it may be difficult to create representative test data for a system that consumes large volumes of images and audio at a high speed.



- **Input data** may need to change over time, particularly if it represents events in the real world. For example, when creating a model that predicts how long it takes to fix a bug, you need to consider that the description of bugs can be updated over time.

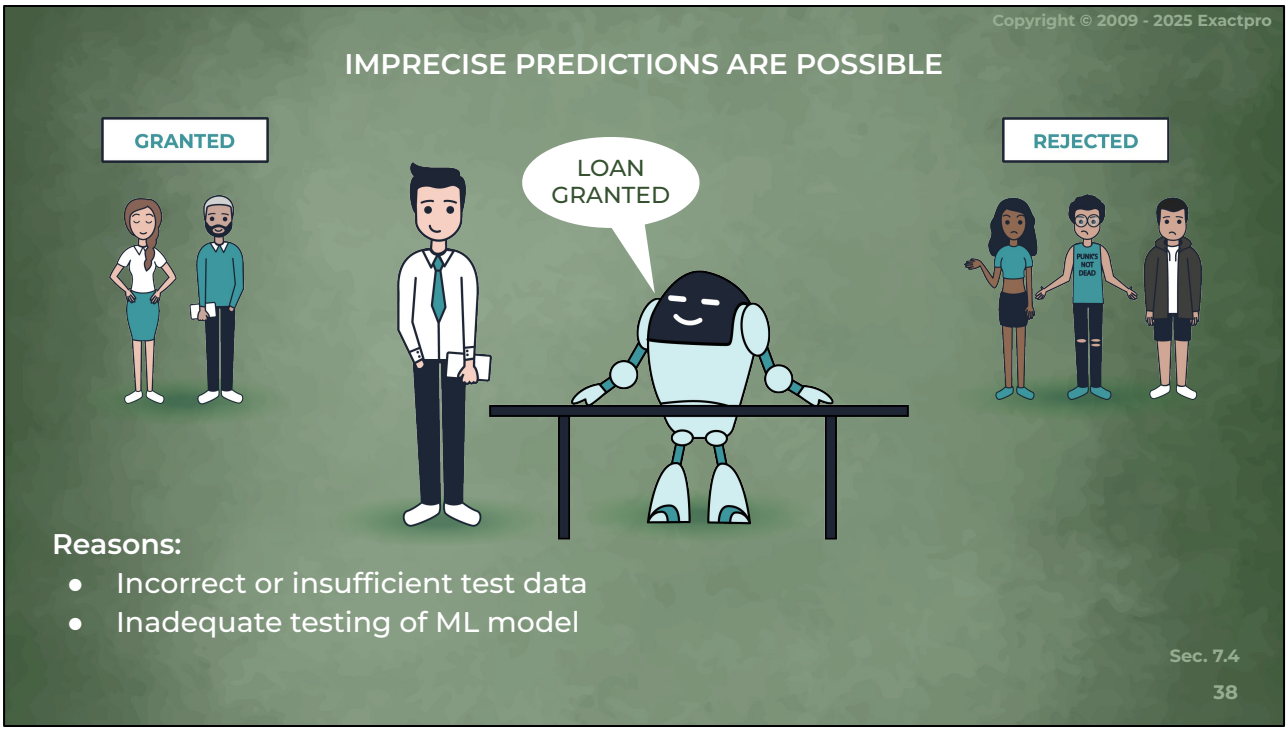


- **Personal and confidential data** may require sanitisation, encryption, or redaction. Legal approval for use may also be required.



- When testers use the **same data acquisition and data pre-processing methods** as the data scientists, then defects in these steps may be masked.

7.4 Testing for Automation Bias in AI-Based Systems



Incorrect or insufficient test data, and therefore inadequate testing of the ML model, can lead to the system providing people with imprecise predictions.

FORMS OF AUTOMATION BIAS:

1. Accepting recommendations provided by the system without even thinking about other sources

Automation bias (complacency bias): A type of bias caused by a person favouring the recommendations of an automated decision-making system over other sources

In addition, there is a tendency for humans to be too trusting of AI-based systems decision-making. This misplaced trust may be called either **automation bias** or **complacency bias**, and takes two forms:

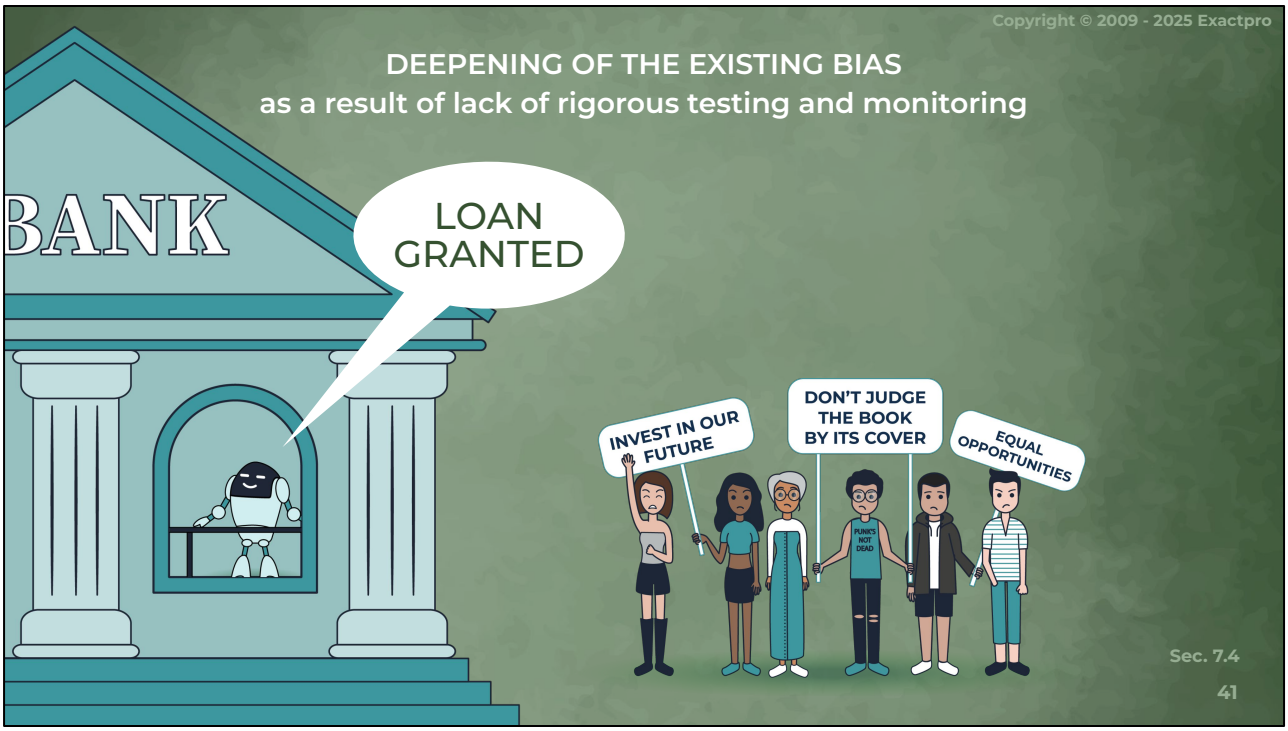
- accepting recommendations provided by the system without even thinking about other sources;

FORMS OF AUTOMATION BIAS:

2. Overlooking failures due to the lack of adequate system monitoring

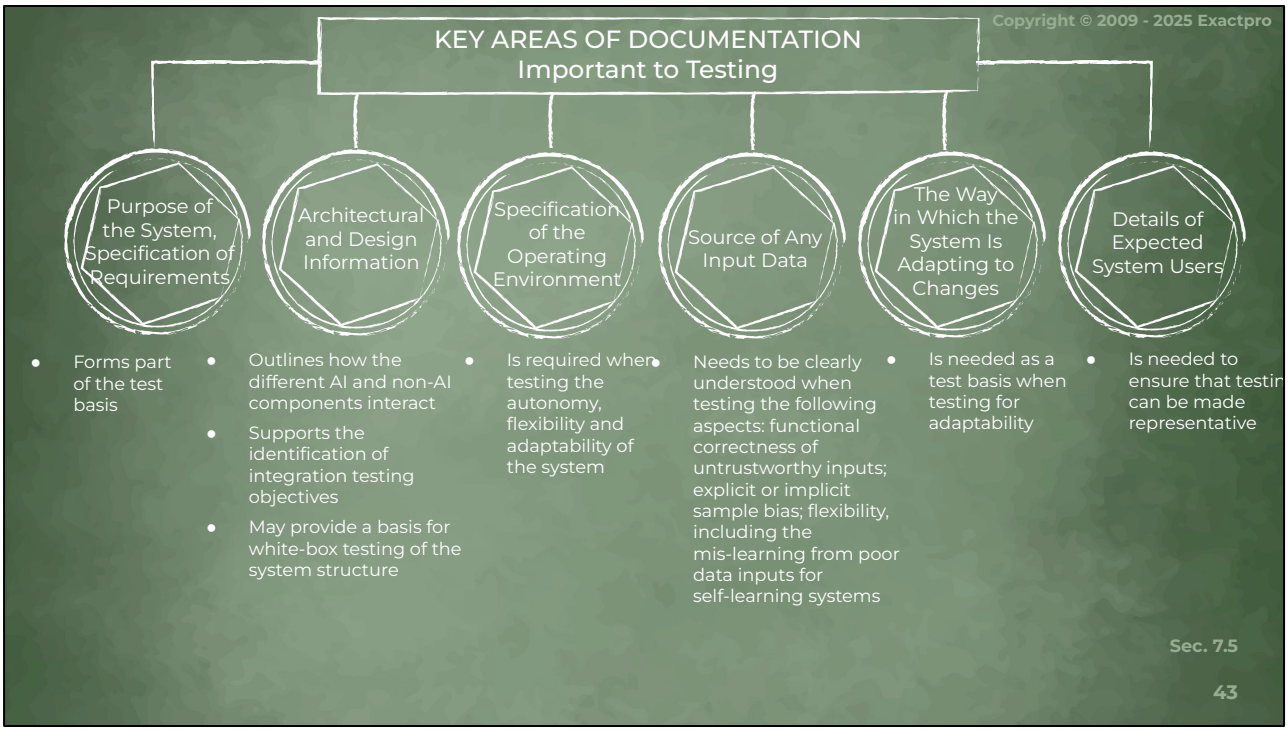
Automation bias (complacency bias): A type of bias caused by a person favouring the recommendations of an automated decision-making system over other sources

- overlooking failures due to the lack of adequate system monitoring.



For example, it is very important in finance and banking. Here, AI presents an opportunity to transform how we deal with credit and risk. Lack of rigorous testing and monitoring can easily lead to deepening of the existing bias, creating vicious cycles that reinforce biased credit allocation while making issues in lending even harder to find.

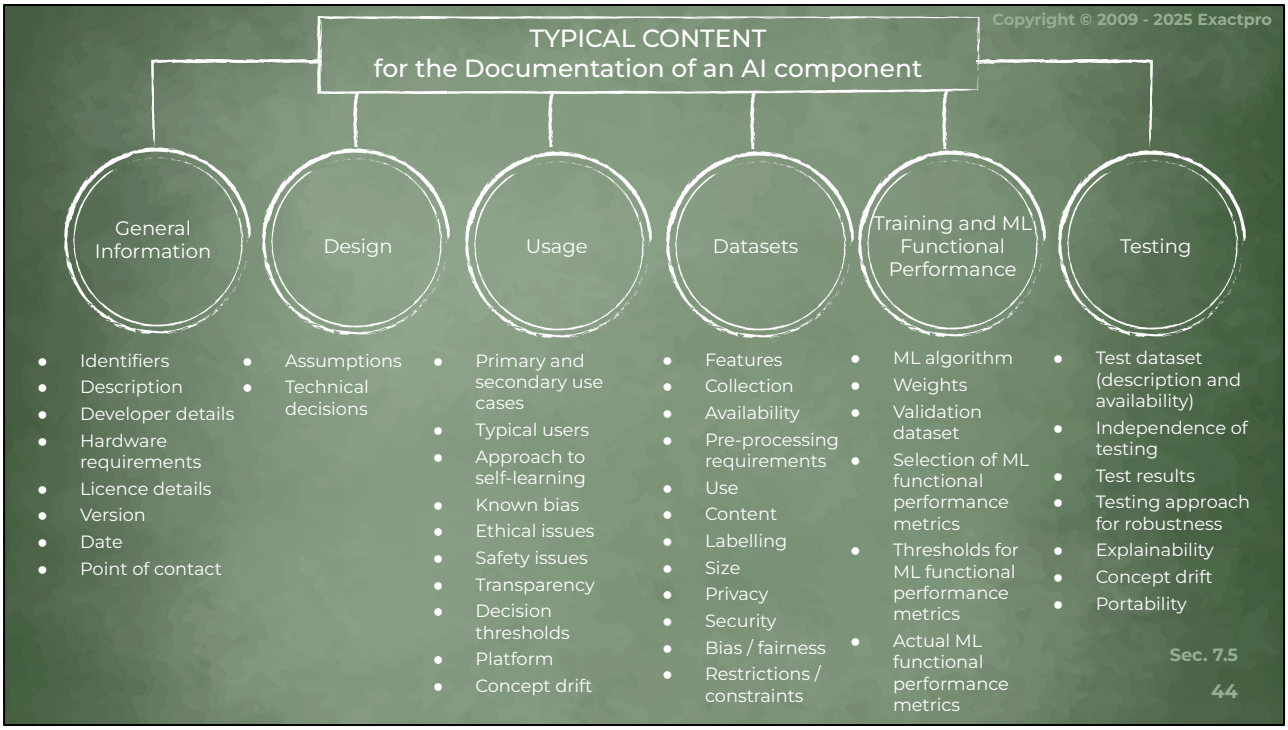
7.5 Documenting an AI Component



In order to avoid that we need to pay attention to documentation. The key areas of documentation that are important to testing are:

- The purpose of the system, and the specification of functional and non-functional requirements. These types of documentation typically form part of the test basis.
- Architectural and design information, outlining how the different AI and non-AI components interact. This supports the identification of integration testing objectives, and may provide a basis for white-box testing of the system structure.
- The specification of the operating environment. This is required when testing the autonomy, flexibility and adaptability of the system.
- The source of any input data, including associated metadata.This needs to be clearly understood when testing the following aspects:
    - functional correctness of untrustworthy inputs;
    - explicit or implicit sample bias;
    - flexibility, including the mis-learning from poor data inputs for self-learning systems.
- The way in which the system is expected to adapt to changes in its operational environment. This is needed as a test basis when testing for

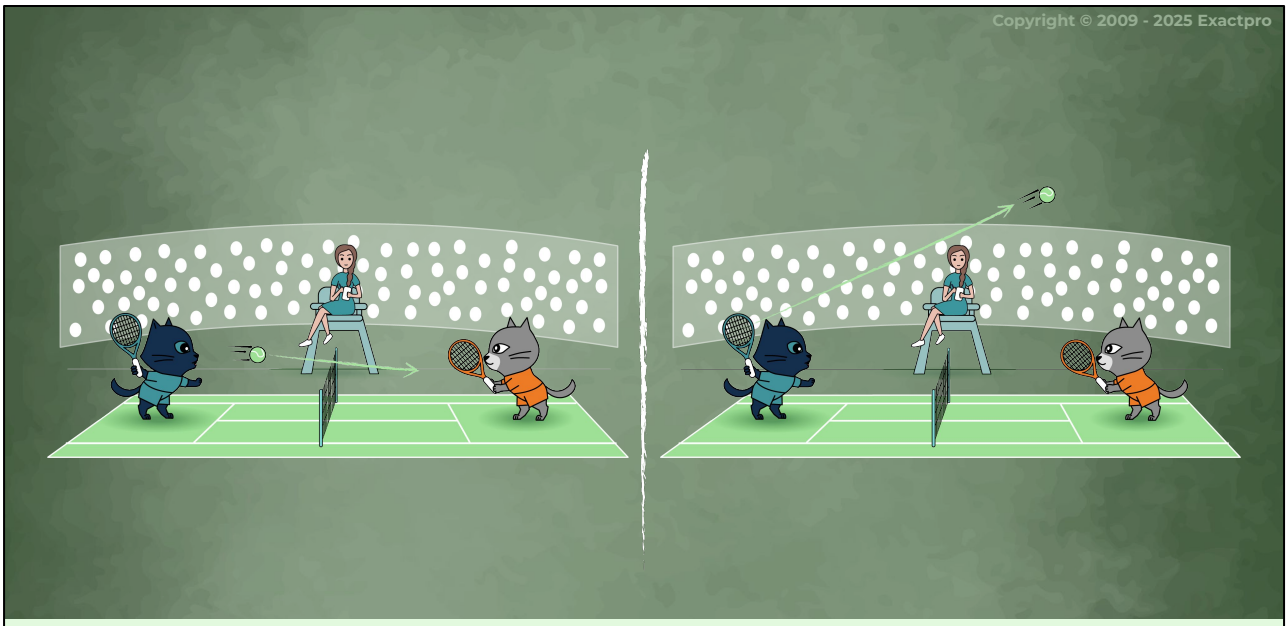
- adaptability.
- Details of expected system users. This is needed to ensure that testing can be made representative.



The typical content for the documentation of an AI component includes:

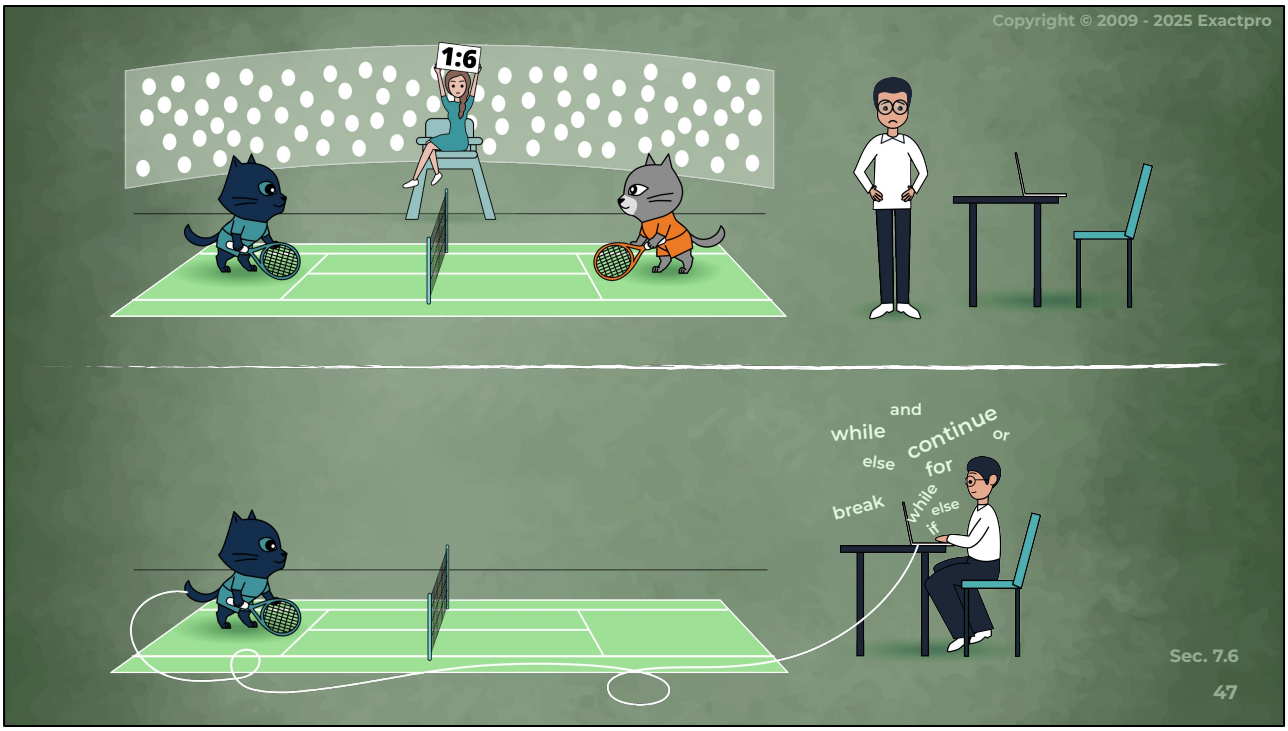
- **general information**: identifiers, description, developer details, hardware requirements, licence details, version, date and point of contact;
- **design**: assumptions and technical decisions;
- **usage**: primary and secondary use cases, typical users, approach to self-learning, known bias, ethical issues, safety issues, transparency, decision thresholds, platform and concept drift;
- **datasets**: features, collection, availability, pre-processing requirements, use, content, labelling, size, privacy, security, bias/fairness and restrictions/constraints;
- **testing**: test dataset (description and availability), independence of testing, test results, testing approach for robustness, explainability, concept drift and portability;
- **training and ML functional performance**: ML algorithm, weights, validation dataset, selection of ML functional performance metrics, thresholds for ML functional performance metrics, and actual ML functional performance metrics.

7.6 Testing for Concept Drift

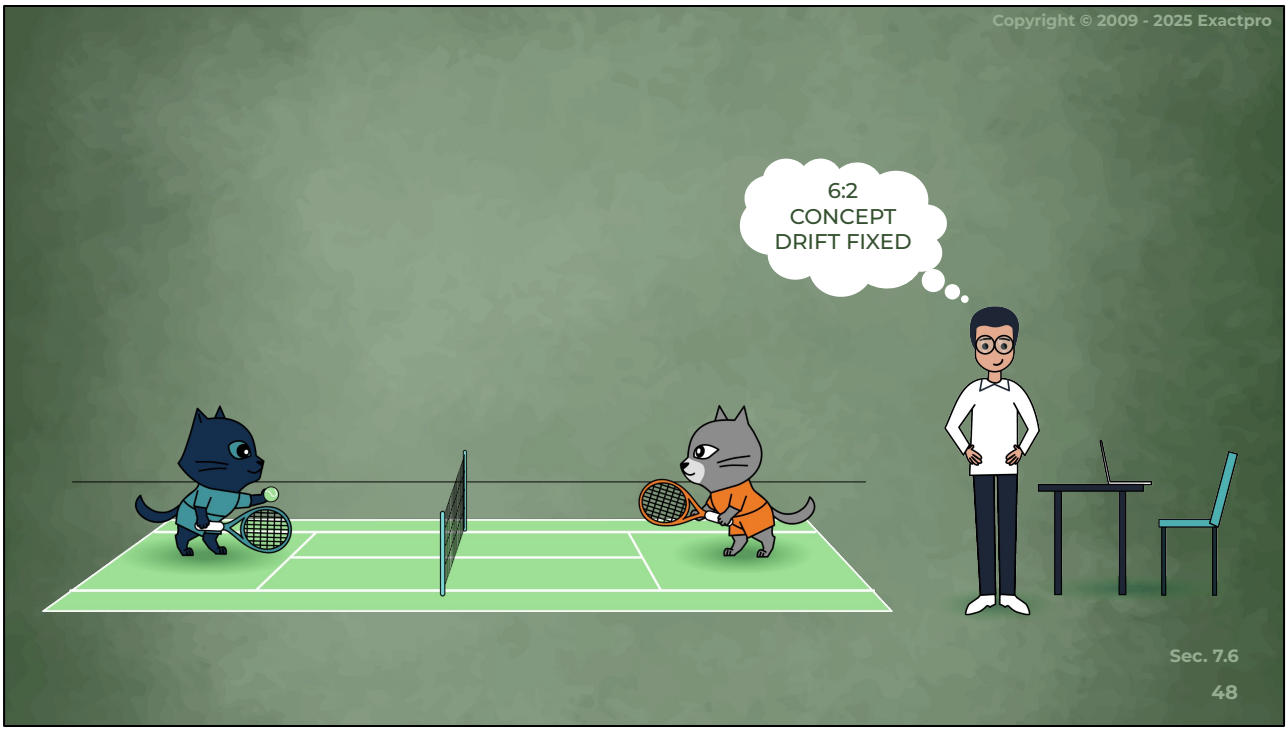


Concept drift: A change in the perceived accuracy of an ML model predictions over time caused by changes in user expectations, behaviour and the operational environment

One more problem that we can face is that the operational environment can change without the trained model being able to adapt. This is called a **concept drift** and it typically causes the outputs of the model to become less accurate and less useful over time.



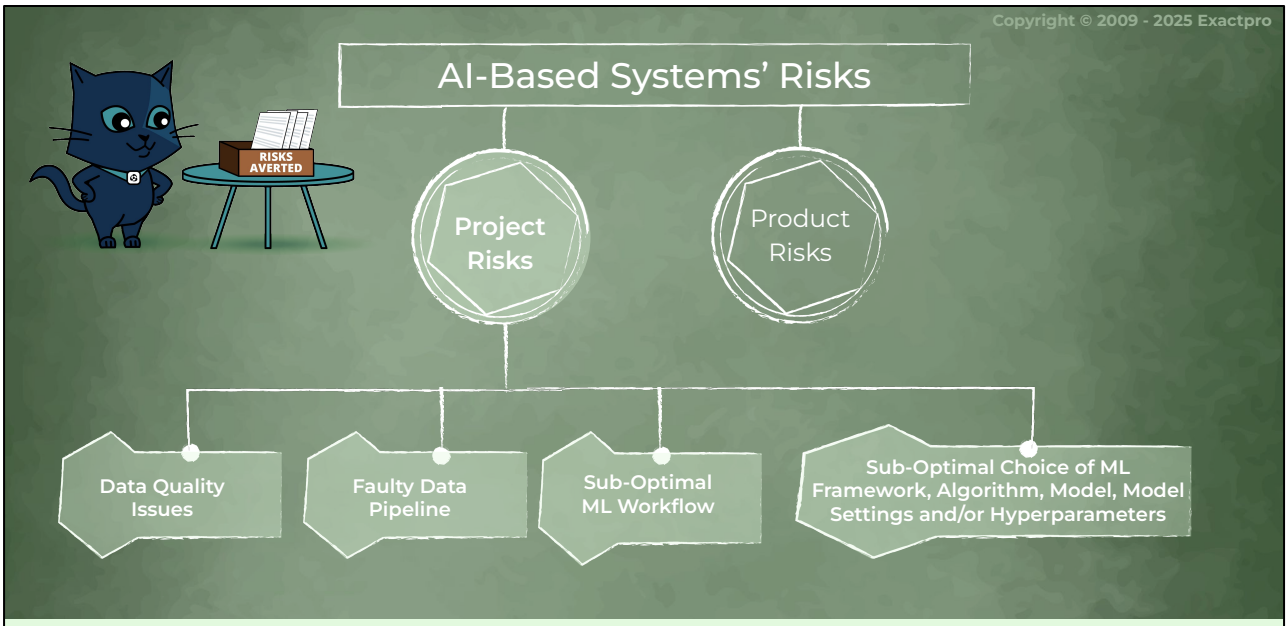
Systems that are susceptible to it should be regularly tested against their agreed functional performance criteria, to ensure that any concept drift is detected quickly.



This can be done through retiring or retraining the system. **Retiring** is the removal of the system. And **retraining** is performed with up-to-date training data and followed by confirmation testing, regression testing, and possibly a form of A/B testing, where the updated B-system is better than the original A-system.

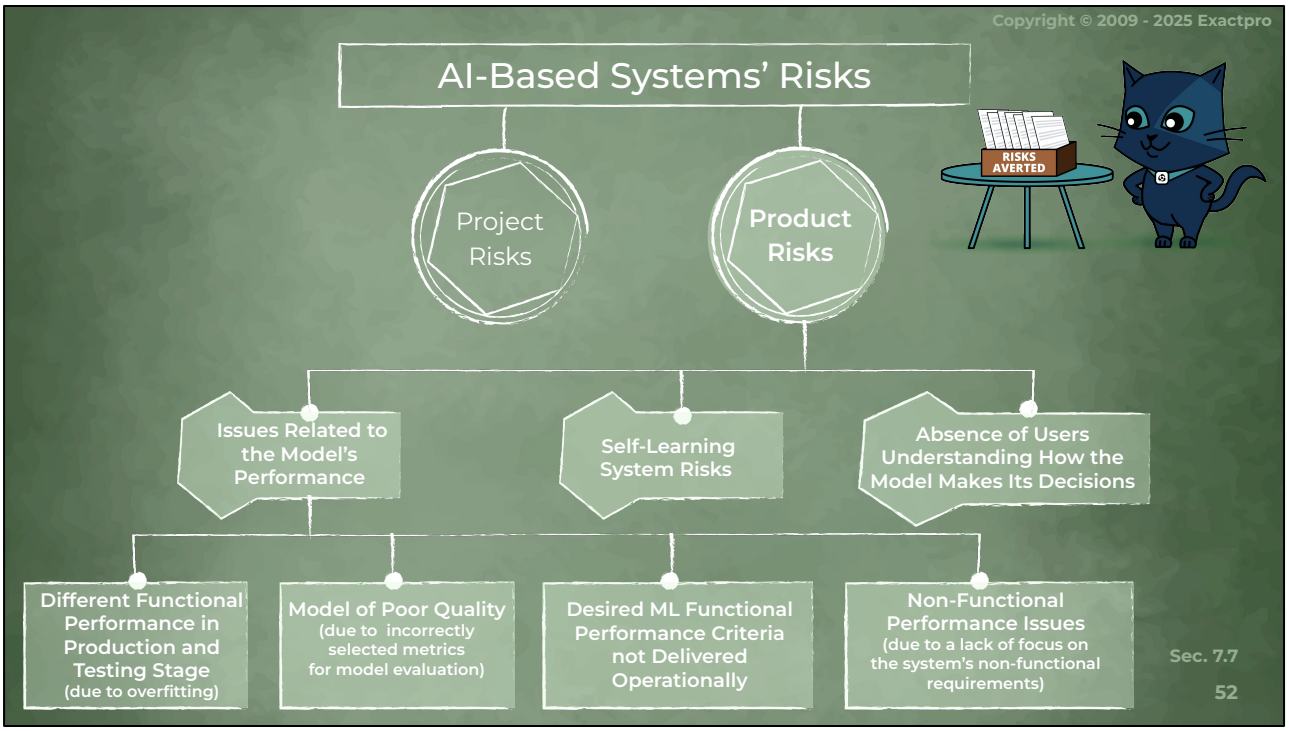
7.7 Selecting a Test Approach for an ML System

An AI-based system generally comprises both AI and non-AI components. The testing approach for such systems is guided by a risk analysis and includes traditional testing methods alongside specialised techniques designed for AI components and AI-driven systems. Let's explore some common risks and their corresponding mitigations for ML systems.



Data pipeline: The implementation of data preparation activities to provide input data to support training by an ML algorithm or prediction by an ML model

AI-based systems' risks can be broken into project and product ones as it is for conventional systems. **Project risks** are connected to the ML development workflow. For instance, data quality issues can happen due to poor assessment during exploratory data analysis or corrupted data pipeline. They impact the next steps of the workflow such as model training, evaluation, and tuning. As a result, the choice of a framework, algorithm/model, and settings/hyperparameters might be sub-optimal. These risks can be mitigated by subject matter expert reviews, EDA, and dynamic testing.



After a model is deployed into production, we start dealing with product risks that stem from the fact the users might come across a number of issues related to the model's performance. If training data is irrelevant it will cause the situation when the functional performance of a model in production differs significantly from the results at the testing stage, this happens frequently due to overfitting. For example, a model may provide excellent predictions when the data is similar to the training data but provides poor results otherwise. If the development team has incorrectly selected metrics for model evaluation, the end-user may consider the model to be of poor quality, even though it shows satisfying results with the selected metrics.

This is because all the metrics used to validate the model are artificial and do not always reflect the logic of the world around us. It also might occur due to concept drift. In addition to that, AI-based systems might have non-functional performance issues when usability of the software doesn't meet users' expectations. Since all of these risks originate from the development phase, they can be resolved by improvements in ML workflow, for example, subject matter expert reviews and relevant NFT activities.

We would also like to highlight self-learning system risks. For example, the data used by the system for self-learning may be irrelevant. In this case,

reviews by experts could identify the problematic data. The system may be failing due to the new self-learned functionality being unacceptable. This could be mitigated by automated regression testing including performance comparison with the previous functionality. The system may be learning in a way that is not expected by the users, which could be detected by experience-based testing.
And even if a model works perfectly, a user might be still unsatisfied with its lack of explainability, interpretability and/or transparency, so it is critical for the stakeholders to understand how a model makes its decisions. This also might be considered a risk, especially in areas such as legal and healthcare.

EXERCISE

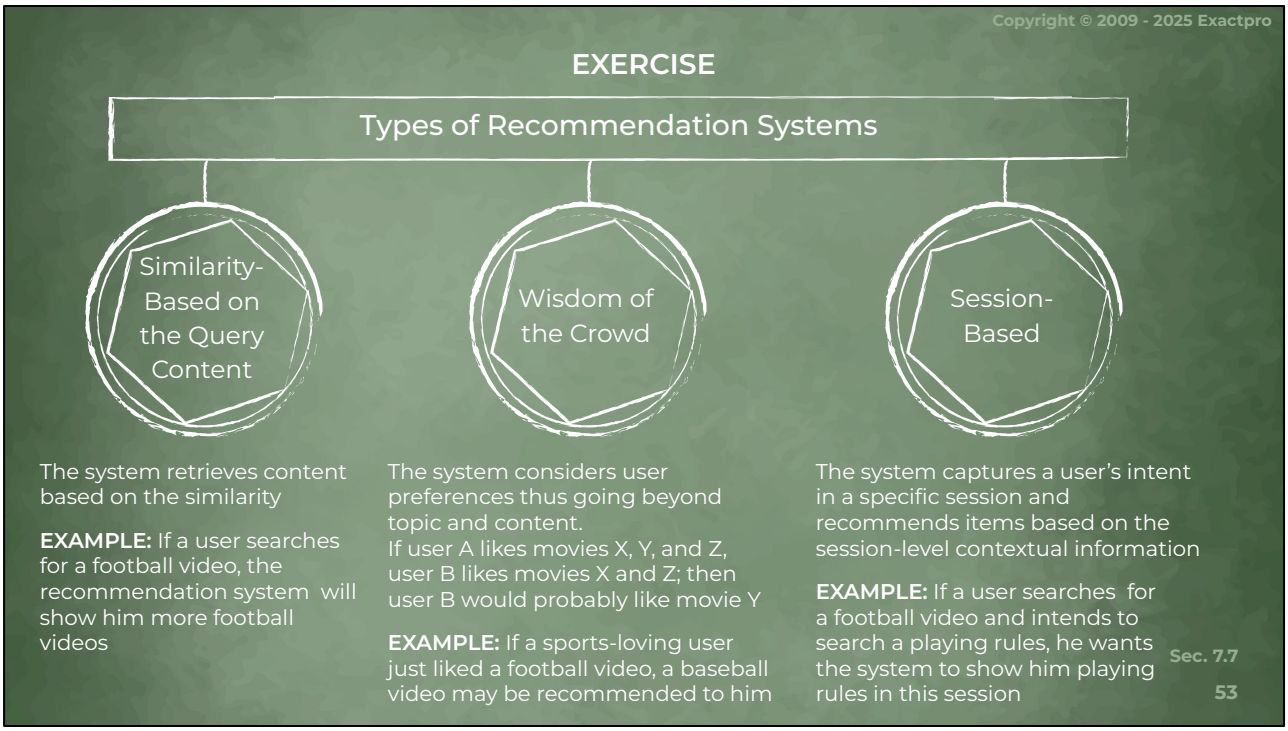
TASK

Determine a **TEST APPROACH** to be followed when developing an ML system for providing **VIDEO RECOMMENDATIONS** on YouTube.

Which **TYPICAL RISK ASPECTS** are most likely to be associated with this system and require mitigation through testing?

#	RISK ASPECTS	TEST APPROACH
1.	Low data quality and data diversity, formatting issues	Reviews, EDA and dynamic testing
2.	Suboptimal choice of ML framework, algorithm, model, model settings and/or hyperparameters	Reviews with experts

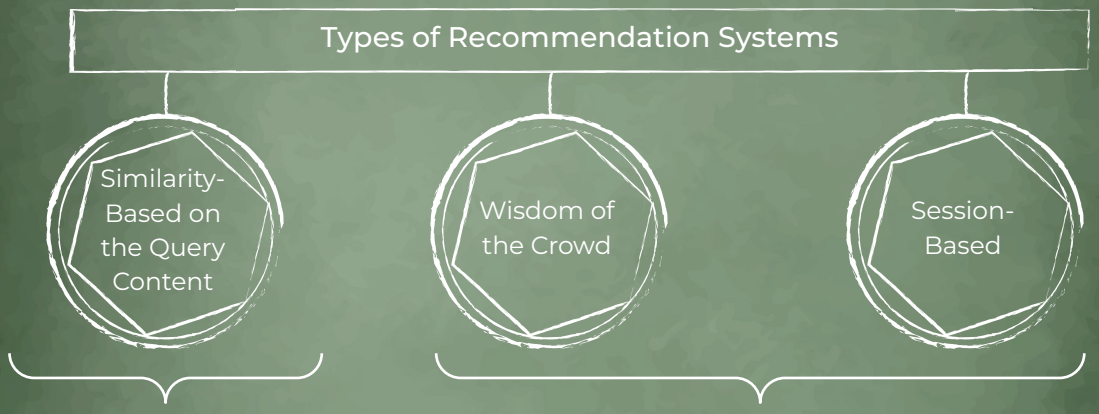
And now do some exercise. Imagine that you have been asked for your opinion on the test approach to be followed when developing an ML system for providing video recommendations on YouTube. Which typical risk aspects do you expect most likely to be associated with this system and require mitigation through testing?
We will take a look at some of the ways through which we can test our entire recommendation system – from the model's behaviour to the health of the pipeline.
The test approach is based on a risk analysis for an ML system. That is why we should start with some typical risks, specific to our video recommendation system, that require mitigation through testing. A good recommendation system should be trained on a large diverse video dataset. Hence, the first group of risks is associated with the low data quality and data diversity, formatting issues. To mitigate it we should use reviews, EDA and dynamic testing. The next possible risk is suboptimal choice of ML framework, algorithm, model, model settings and/or hyperparameters. And it may be mitigated by reviews with experts.



There are three types of recommendation systems:

- **Similarity-based on the query content**: The system retrieves content based on the similarity. For example, if you like a football video, it will show you another one. The match is based on the item content like description, title, etc.
- **Wisdom of the crowd**: Instead of relying on item content, these recommendation models consider user preferences. If user A likes movies X, Y, and Z, user B likes movies X and Z; then user B would probably like movie Y. These models are popular because they go beyond topic and content. They can thoroughly recommend a baseball video to a sports-loving user who just liked a football video.
- **Session-based**: Session-based systems capture a user's intent in a specific session and recommend items based on the session-level contextual information. For instance, If a user searches for a football video and intends to search a playing rules, he wants the system to show him playing rules in this session, even though he might have liked a certain nature video earlier.

EXERCISE



1. When user-item interaction data is insufficient
2. For cold-start problem (when insufficient data is for a new user)

Require a lot of user-item interaction data

The second and third require a lot of user-item interaction data. If that is not available, one might start with the first type of recommendation system. Even when there is a lot of data for existing users, one might not have enough for a new user. This situation is called the cold-start problem in recommendation systems. In such cases, content-based recommendation systems can be a good proxy until one has enough interaction data for new users.

EXERCISE

#	RISK ASPECTS	TEST APPROACH
3.	Incorrect objective design (a minimum watch time, like, share, download, clicks may be used as objectives for training)	Reviews, EDA and dynamic testing
4.	Recommending irrelevant or repetitive content leading to user disengagement	Extensive testing to evaluate the system's ability to provide diverse and relevant recommendations. Use metrics such as click-through rates, watch time, and user feedback to measure the quality of recommendations
5.	Recommending low-quality or misleading content that may harm user's trust	Implement content quality assessment mechanisms. Test the system's ability to filter out low-quality or misleading videos. Check that user's feedback is monitored and the model quality metrics are adjusted

Let's return to the risk aspects. One more risk is related to the incorrect objective design. For example, the model may be trained to predict if a user likes a video on YouTube based on whether or not a user will watch 95% of the said video by video length. If we have a near-perfect model that predicts a probability of >95% watch, we can say that we are recommending videos that the user likes, right?

Here is the catch though – consider a one-minute video (V1) vs a thirty-minute video (V30). It takes 57 seconds to watch the 95% of V1, and it takes 1710 seconds to watch 95% of V30. V1 could also be a clickbait video, while a user can like V30 and still watch just 1600 seconds of it. So does our definition assure that the positive labels represent user preference?

Secondly, most platforms have multiple signals – like, share, download, clicks, etc. Which objective should one use to train the model? Usually, one is not enough. Let us say we train multiple models based on different objectives. We have multiple scores from each model. Then a single number score is created based on an aggregation formula on all scores, which is used to create the final ranking.

The point is that if the training objective is not carefully designed, even a near-perfect model will not give good recommendations.

The next possible risk aspect is recommending irrelevant or repetitive content which may lead to user disengagement. To mitigate it, an extensive testing may be conducted to evaluate the system's ability to provide diverse and

relevant recommendations.
One more risk is recommending low-quality or misleading content that may harm user’s trust. The possible mitigation is to implement content quality assessment mechanisms, to test the system's ability to filter out low-quality or misleading videos, and to check that user’s feedback is monitored and the model based on content quality metrics are adjusted.

EXERCISE

#	RISK ASPECTS	TEST APPROACH
6.	Making mistakes during inference due to the mishandling of model features (user location, age, video length, etc.) For example, model predictions could vary if a feature was scaled in training but not in inference	Reviews, EDA, dynamic testing to evaluate model predictions and metrics
7.	Inconsistencies and inaccurate recommendations due to using older embeddings in training	Check that after each training cycle, the updated embeddings are used to recommend items

One more risk is making mistakes during inference due to the mishandling of features. During model training, we might use many features besides embeddings, like user location, age, video length, etc. It is common to apply transformations like scaling to these features before using them. If you scaled a feature in training but not in inference, your model predictions could vary. To mitigate this kind of risk we may use reviews, EDA and dynamic testing to evaluate model predictions and metrics. The next risk is related to the updated embeddings. Recommendation systems are trained periodically. Using older embeddings can lead to inconsistencies and inaccurate recommendations. Hence, after each training cycle, the updated embeddings should be used to recommend items, and we should check this.

EXERCISE

#	RISK ASPECTS	TEST APPROACH
8.	The desired ML functional performance criteria are met, but the users may be unhappy with the delivered results due to the selection of the wrong performance criteria (e.g., high recall was selected when high precision was needed)	Reviews with experts, experience-based testing, more frequent testing of the operational system (could mitigate the concept drift risk)
9.	Dissatisfied users when recommendations do not align with their preferences or if they find the content uninteresting	User acceptance testing to gather feedback on the relevance and quality of recommendations. Check that user engagement metrics are monitored and retraining mechanisms are implemented. Test the system's ability to adapt to changing trends and shifting user preferences by introducing new data and simulating real-world scenarios

One more possible risk aspect is when the desired ML functional performance criteria are met, but the users may be unhappy with the delivered results. This may happen when the performance criteria are wrong selected, e.g., high recall instead of high precision is chosen as target metric.
In the case of video recommendations on YouTube, false positives are detrimental. YouTube's primary goal is to suggest videos a user will love – and to avoid recommending videos a user will dislike at all costs. However, it's perfectly acceptable if not all videos a user could enjoy are presented to them. In other words, false negatives are not a major concern. For this kind of problem, precision is the right target metric.
Reviews with experts may mitigate the chance of choosing the wrong ML functional performance metrics, or experience-based testing could also identify inappropriate criteria. The risk could also be due to concept drift, in which case more frequent testing of the operational system could mitigate the risk.
Let's proceed to the next risk. Users may be dissatisfied if recommendations do not align with their preferences or if they find the content uninteresting. The way to mitigate this is to conduct user acceptance testing to gather feedback on the relevance and quality of recommendations, to check that user engagement metrics are monitored and retraining mechanisms are implemented to improve user satisfaction, to test the system's ability to adapt to shifting user preferences and changing trends by introducing new data and simulating real-world scenarios.

EXERCISE

#	RISK ASPECTS	TEST APPROACH
10.	Algorithmic biases leading to recommendations that favour popular, highly ranked or clickbait content over a user's actual preferences (*popularity bias*), or fail to understand multiple user interests at the same time, providing only a certain kind of result (*single-interest bias*)	Bias testing to identify and address potential biases, evaluation of recommendation performance across diverse user segments and content genres, usage of fairness-aware metrics to assess balance and equity
11.	Users' privacy concerns related to the de-anonymisation, collection and use of the users' viewing history for recommendations	Rigorous privacy testing, check compliance with privacy regulations, check that users are allowed to opt-out or adjust privacy settings

Another possible risk is algorithmic biases in recommendations. If not trained properly, recommendation and ranking systems may be programmed with a series of algorithmic biases which might impede their effectiveness. These biases can vary, based on whether a recommendation algorithm prioritises popular, highly ranked or clickbait content over a user's actual preferences (**popularity bias**), or fails to understand multiple user interests at the same time, recommending only a certain kind of result (**single-interest bias**).
To mitigate this kind of risk you may implement bias testing to identify and address potential biases, evaluate recommendation performance across diverse user segments and content genres, and use fairness-aware metrics to assess balance and equity. One easy way to measure bias is to see the views distribution. What percentage of views are captured by the top 1%, 5%, 10%,… of videos, and how often are these videos recommended to users vs other videos. This 80-20 effect can be seen across topics (specific topics dominate the app), creators (few popular creators vs niche creators), etc. An ML model learns bias in the dataset, among other things. So if your dataset is biased, then chances are that your recommendation results will reflect it.
One more risk is privacy risk. As recommendation systems leverage large quantities of user data to carry out their functions, they may be prone to privacy risks. Data containing personal identifiers may be collected by such systems without obtaining explicit consent, causing loss of user agency. If not fortified adequately through data protection and cybersecurity mechanisms,

these datasets may run the risk of being de-anonymised and misused by bad actors to granularly-profile users.
The possible way to mitigate privacy risks is to implement rigorous privacy testing, to check compliance with privacy regulations and to check that users are allowed to opt-out or adjust privacy settings.

EXERCISE

#	RISK ASPECTS	TEST APPROACH
12.	Violating copyright laws or ethical standards in content recommendations	Compliance review with relevant regulations, content and ethical guidelines
13.	Problems with scaling of a growing user base and increasing amounts of content	Scalability testing to ensure the recommendation system can handle increasing workloads, testing system's response time under different levels of user activity and content volume
14.	Adversarial attacks with malicious attempts to manipulate the recommendation system	Adversarial testing to identify vulnerabilities, robustness testing to evaluate the system's resilience to adversarial attacks

The next risk aspect is violating copyright laws or ethical standards in content recommendations. Compliance review with relevant regulations, content and ethical guidelines might help to mitigate these legal and ethical risks.
Let’s proceed to the scalability and performance risk when the system may struggle to scale with a growing user base and increasing amounts of content. The possible mitigation is to conduct scalability testing to ensure the recommendation system can handle increasing workloads and to test the system's response time under different levels of user activity and content volume. Finally, the last risk aspect we will discuss is adversarial attacks with malicious attempts to manipulate the recommendation system. To mitigate this risk, adversarial testing should be conducted to identify vulnerabilities, and robustness testing should be implemented to evaluate the system's resilience to adversarial attacks.