AI 시스템의 품질 특성 및 검증 방법

KCSE 2020

Challenges Testing AI-Based Systems

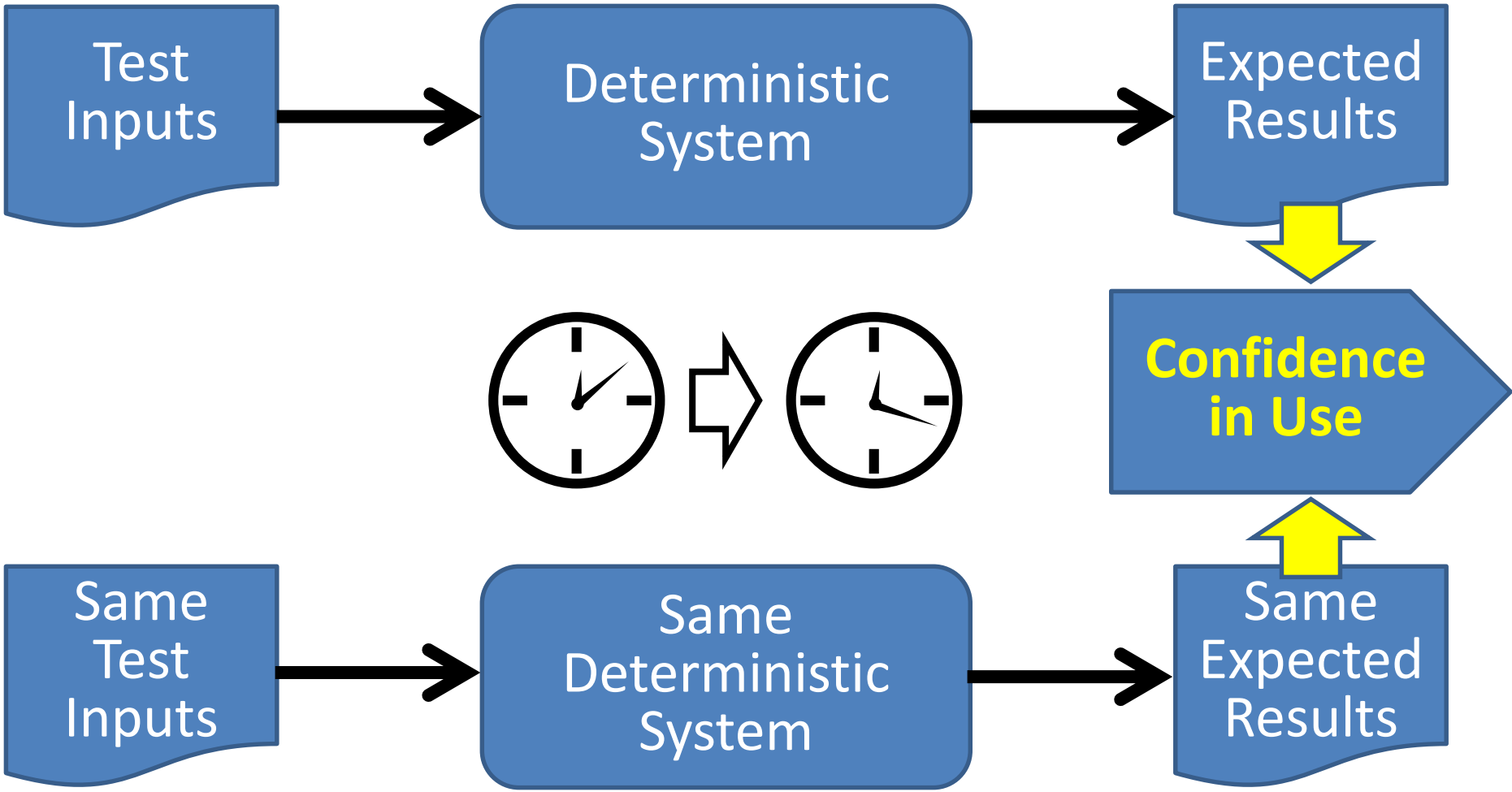
Probabilistic & Non-Deterministic Systems

- **Due to their probabilistic nature, many AI-Based systems do not always generate exact values to use as an expected result**
  - e.g. route calculation based on an initial random seed
  - we accept sub-optimal, but good-enough solutions
    - which makes generating a single exact expected outcome impossible
  - this makes the systems non-deterministic
    - non-deterministic systems make regression test suites more tricky
- **Due to this, we need 'smarter', more complicated, expected results, perhaps including tolerances**
- **Probabilistic AI-Based systems may also require the tester to run the same test multiple times to provide a statistically significant assurance that the system is working correctly**
  - one test that a probabilistic system works is not a guarantee that it will do the same operationally

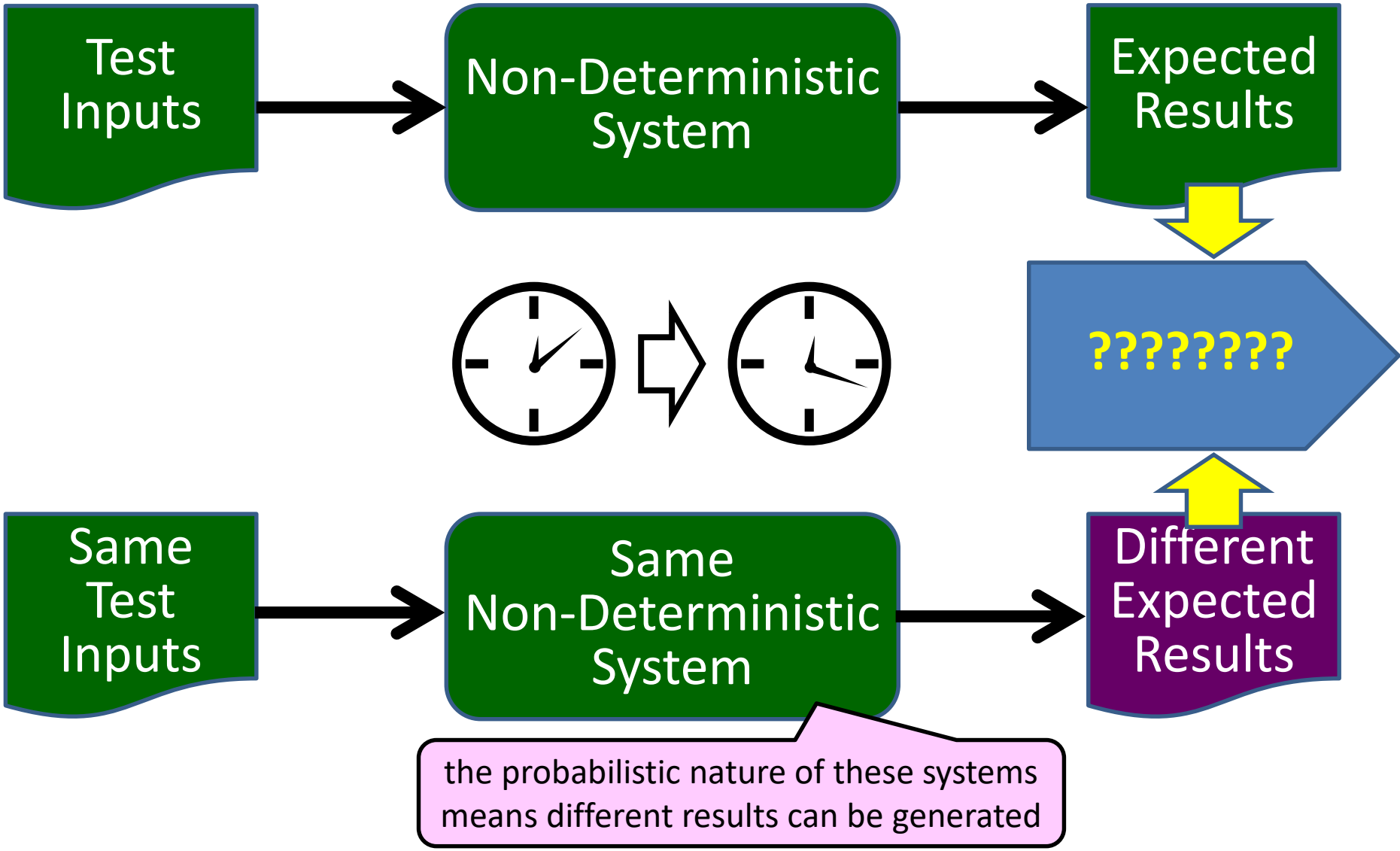
Testing AI-Based Systems

- **AI-Based systems are typically made up of conventional components (e.g. a user interface) and AI components (e.g. a Machine Learning model)**
  - both parts need to be tested
  - and integration tested
  - and both are software with conventional software bugs and so conventional software testing is needed
- **However, AI-Based systems include a number of special attributes that can make additional testing necessary…**
  - AI-specific characteristics, such as flexibility, bias, transparency, etc.

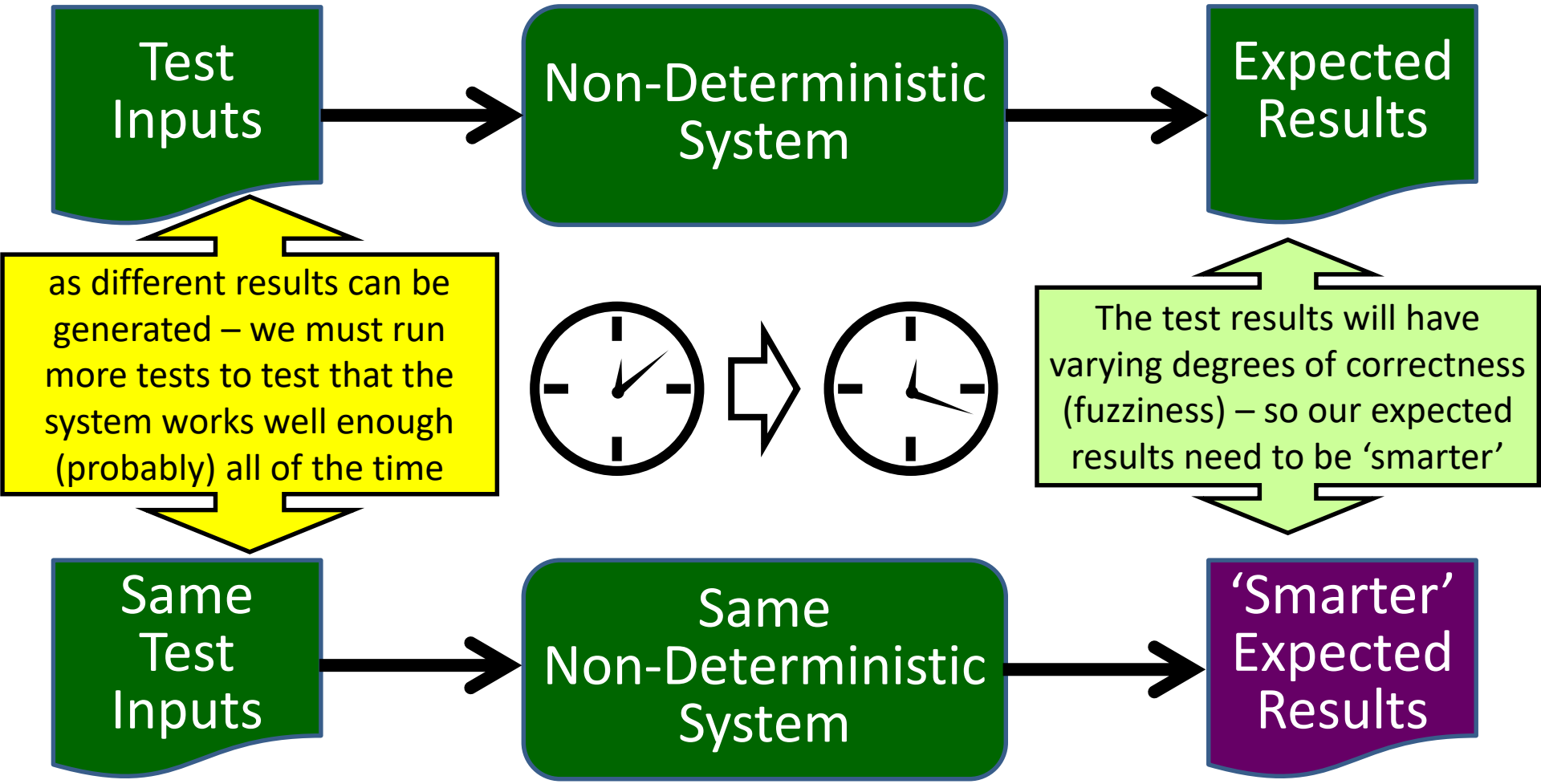
Ideal Deterministic Systems



Testing Non-Deterministic Systems



Testing Non-Deterministic Systems



AI – The Test Oracle Problem

- **To test effectively – we need to be able to specify the expected behaviour**
- **But AI-Based systems can be:**
  - poorly-specified
  - probabilistic
  - non-deterministic
  - complex
  - constantly changing (through self-learning)

Deriving expected results for AI can be very difficult!

Black-Box Testing of AI-Based Systems

Black Box Testing of AI-Based Systems

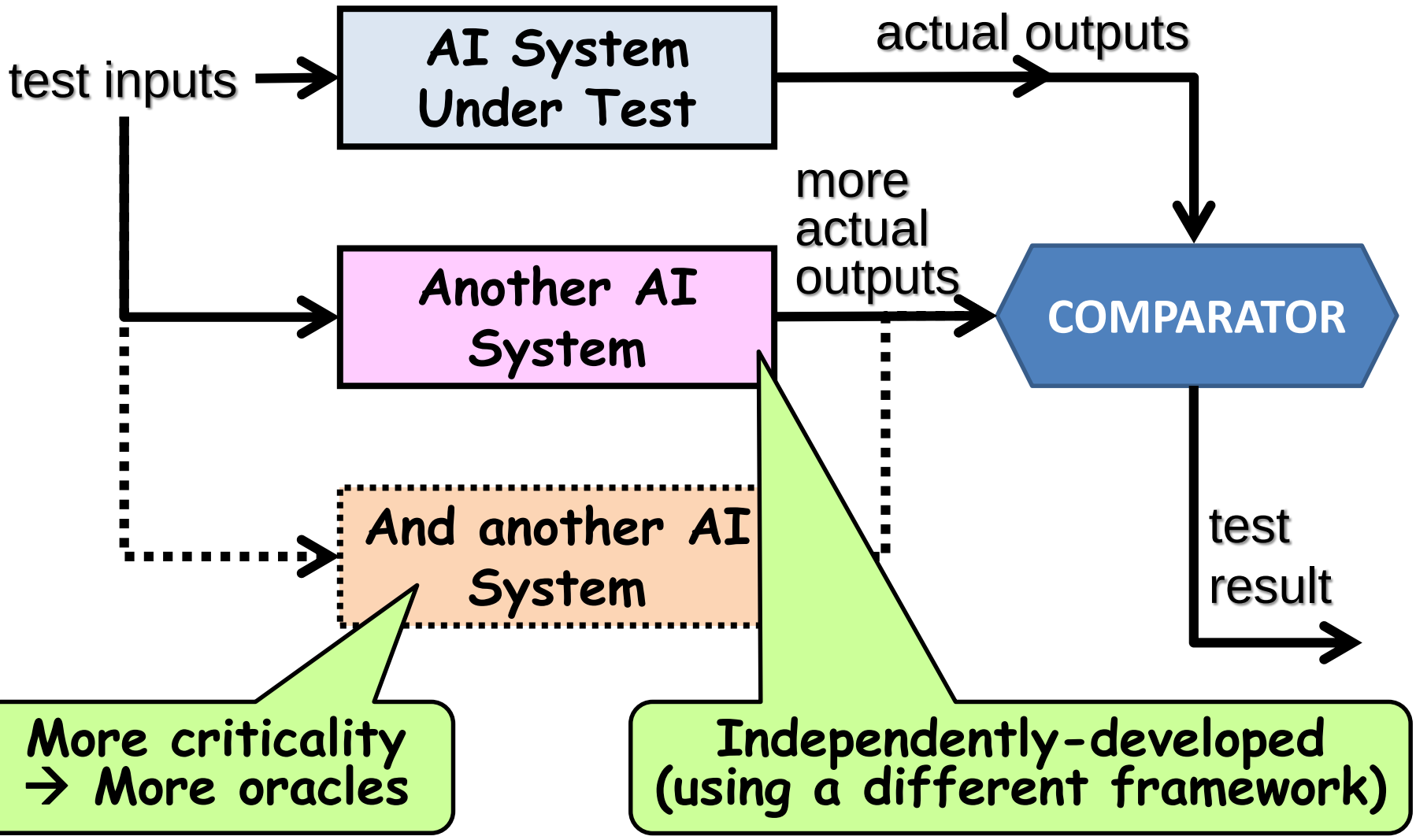
- **Traditional Black Box Techniques**
  - Often reliant on clear specifications…but
  - Testing for Crashes (implicit test oracle)
  - **Combinatorial Testing**
- **A/B Testing**
- **Back-to-Back Testing**
- **Metamorphic Testing**

Back-to-Back Testing Concepts

- **An approach to solving the test oracle problem that uses an alternative version of the program produced independently as a pseudo-oracle**
  - e.g. pre-existing system, written by a different programming team or written in an entirely different programming language
  - e.g. for AI, use different frameworks/algorithms/models (e.g. random forest, SVM, neural network)
- **Not a test case generation technique as test inputs are not generated**
  - expected results are generated automatically by the pseudo oracle
  - when used in partnership with tools for generating test inputs (random or otherwise) it becomes a powerful way to perform high-volume automated testing
- **Also known as Differential Testing**

Back-to-Back Testing

A partial solution to the oracle problem



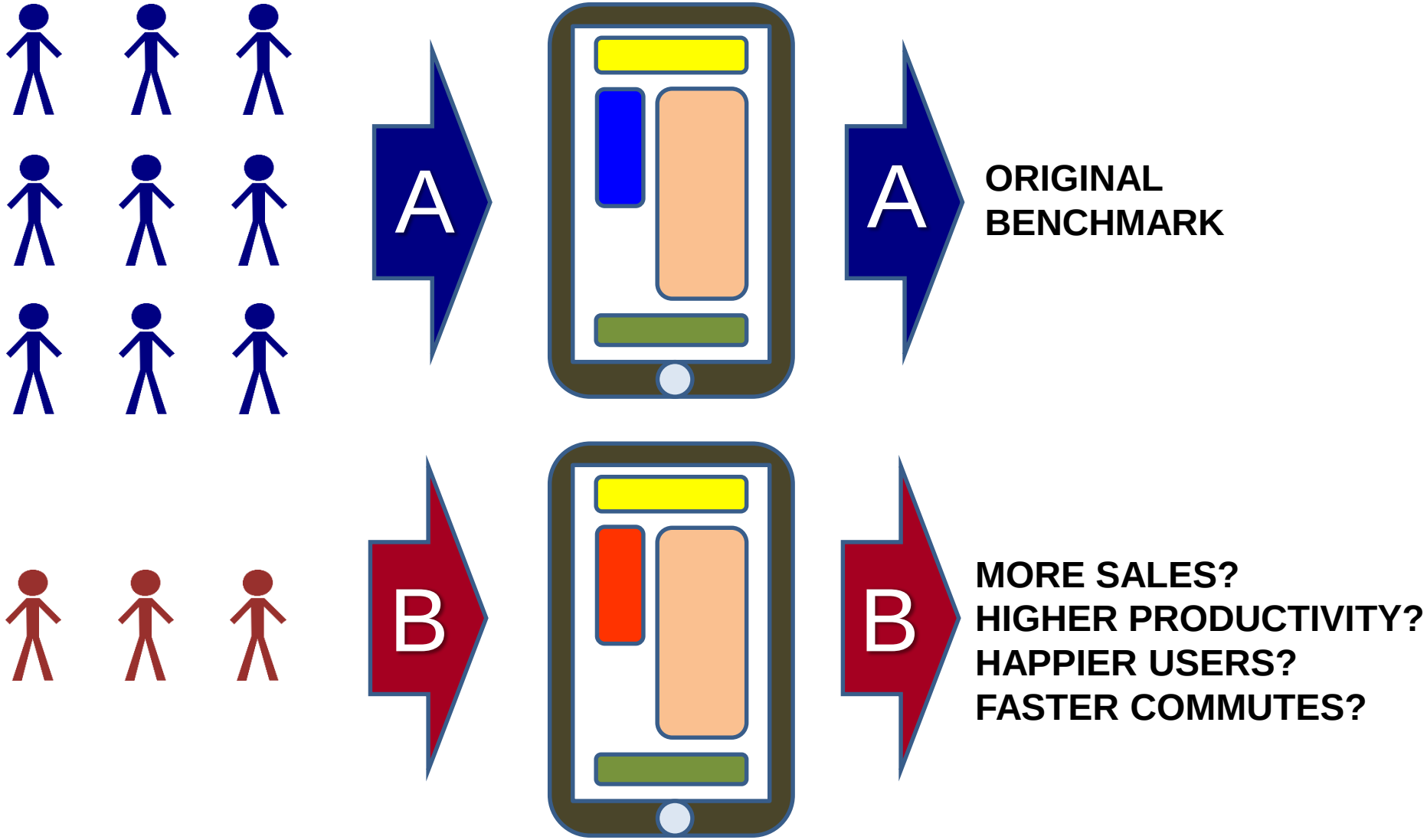
Back-to-Back Testing Practices

- **May do comparisons with:**
  - conventional systems
  - simplified, linear models
  - partial models
  - systems ignoring non-functional constraints, such as speed of prediction
- **Must check for independence between implementations**
  - especially with so much reusable, open source, AI software

A/B Testing - Concepts

- **A statistical testing approach to solving the test oracle problem that allows testers to determine which of two systems performs better**
  - requires a statistically significant number of tests and can be time-consuming, although tools (often using AI) can be used to support it
- **A/B testing is not a test case generation technique as test inputs are not generated**
  - uses the existing system as a partial oracle
  - by comparing the new system with the current system, it is possible to determine if the new system is better in some way
- **Sometimes known as split-run testing**

A/B Testing


A
A
ORIGINAL
BENCHMARK
B
B
MORE SALES?
HIGHER PRODUCTIVITY?
HAPPIER USERS?
FASTER COMMUTES?

A/B Testing - Practices

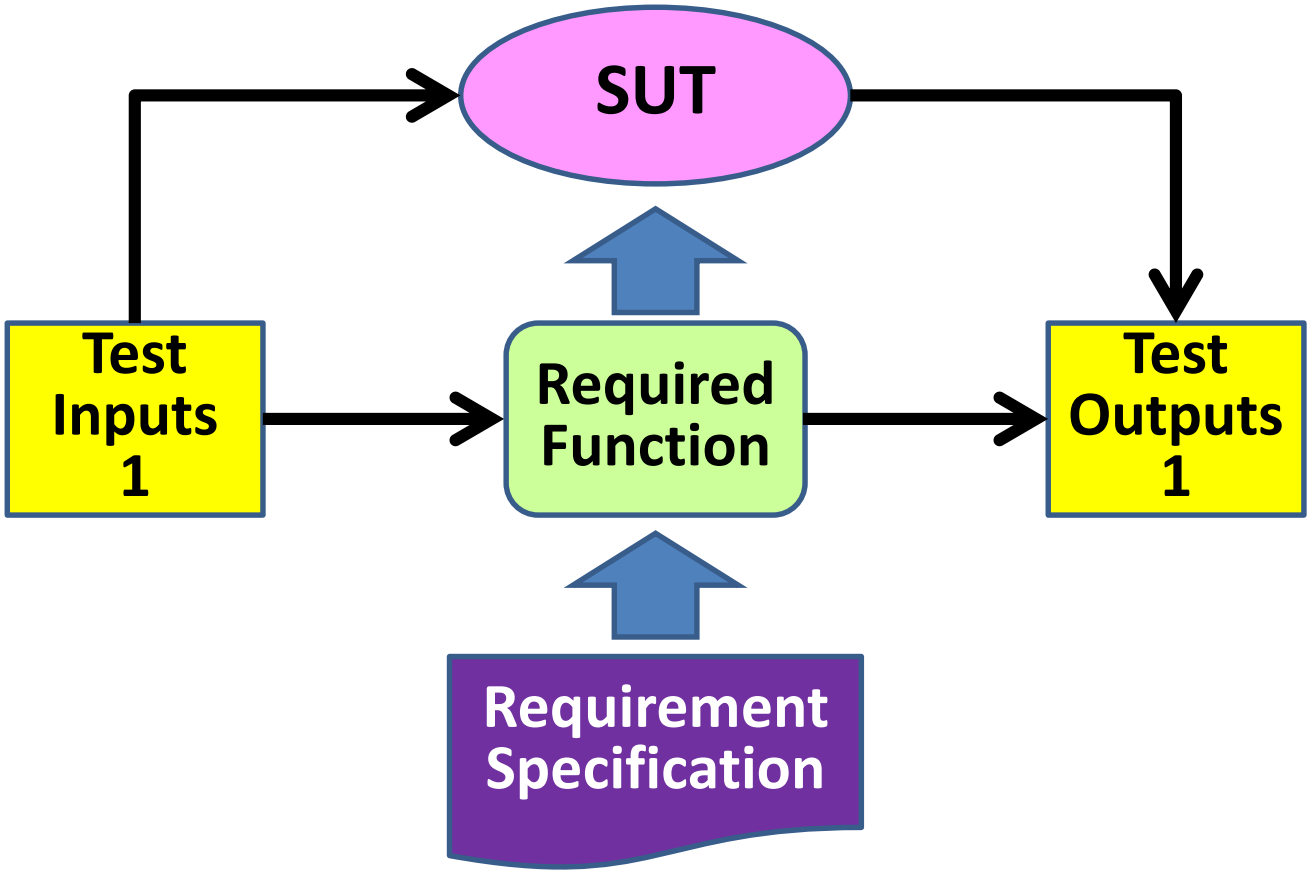
- **Often used for digital marketing in client-facing situations**
  - e.g. finding which variant email message gets the best response
  - the measure of success may by more sales, but for an AI-Based system, such as a ML classifier, performance metrics, such as accuracy, sensitivity and recall, could be used
- **Can be used whenever a component of an AI-Based system is updated, as long as acceptance criteria are defined and agreed**
  - e.g. 'specified performance metrics must improve or stay the same'
- **If automated, it can be used for testing self-learning AI-Based systems, by comparing the new performance of the system with its previous performance and reverting to the previous version if the self-learning has not improved the system performance**
  - but care must be taken to ensure valid acceptance criteria are set

Metamorphic Testing (MT)

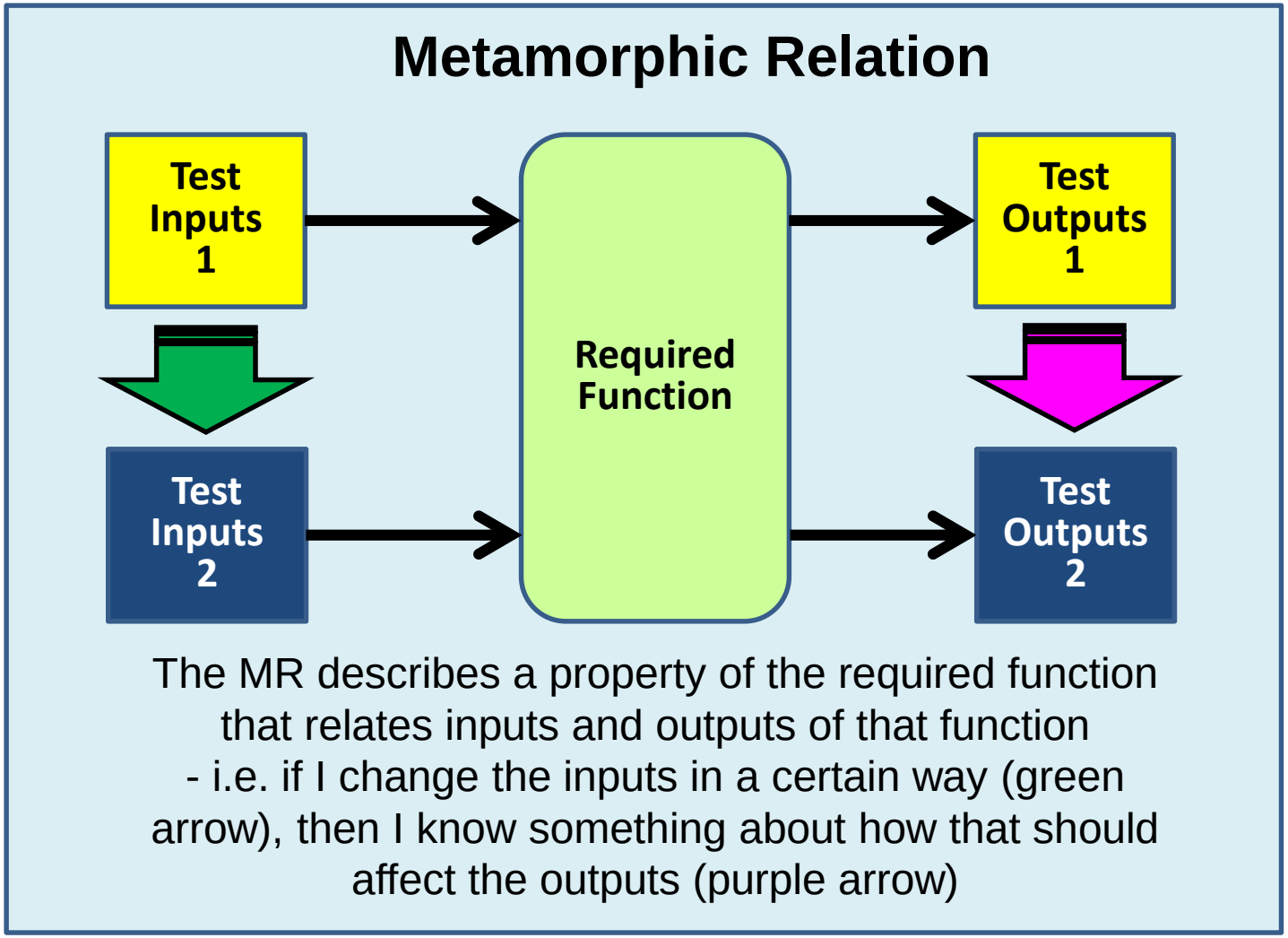
Intro - Metamorphic Testing (MT)

- Invented in 1998 at an Australian university
- Simple, effective, automatable, and language-independent
- Requires an understanding of the application domain
- Black-Box (but, can be Grey-Box)
- An approach to solving the test oracle problem by generating new test cases from existing test cases so that fewer expected results must be generated

MT - Starting Point

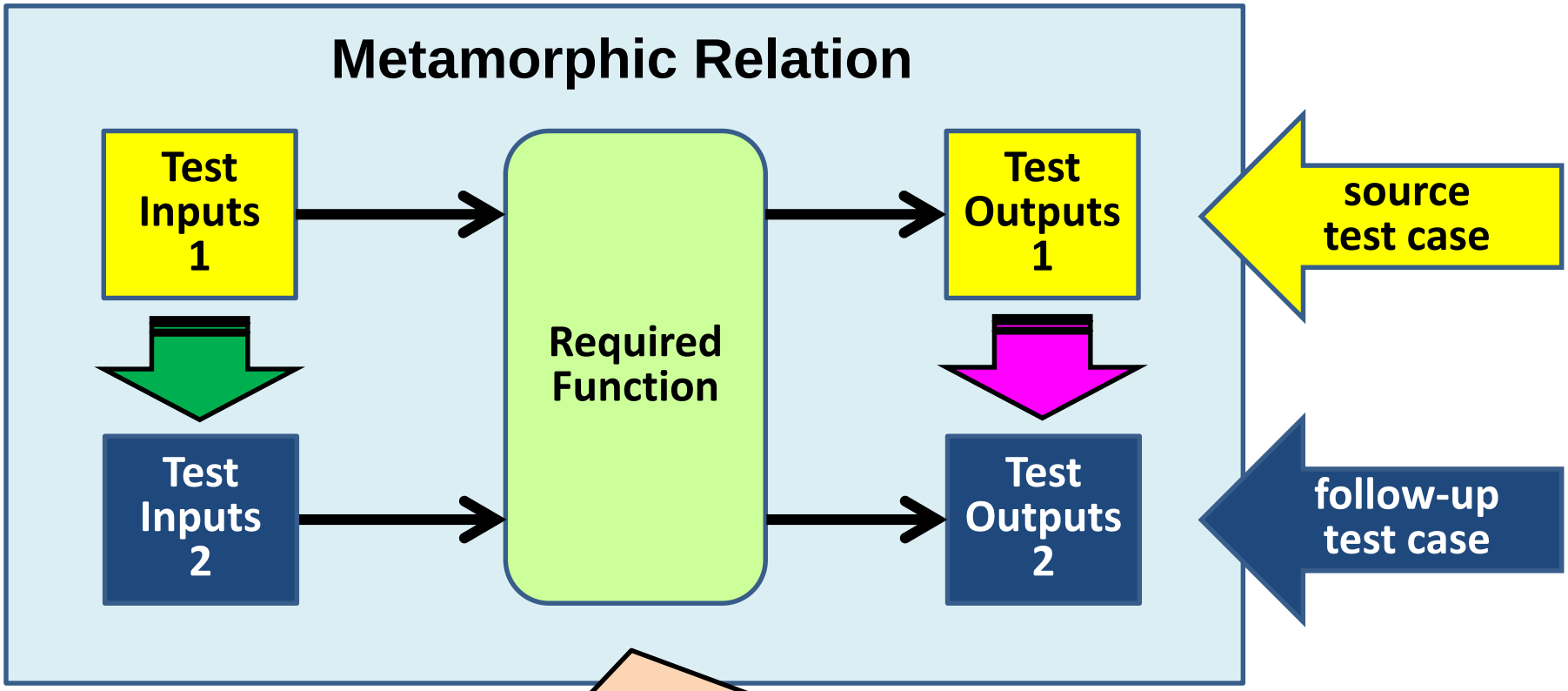


Metamorphic Relation



The MR describes a property of the required function that relates inputs and outputs of that function - i.e. if I change the inputs in a certain way (green arrow), then I know something about how that should affect the outputs (purple arrow)

Metamorphic Testing (MT)



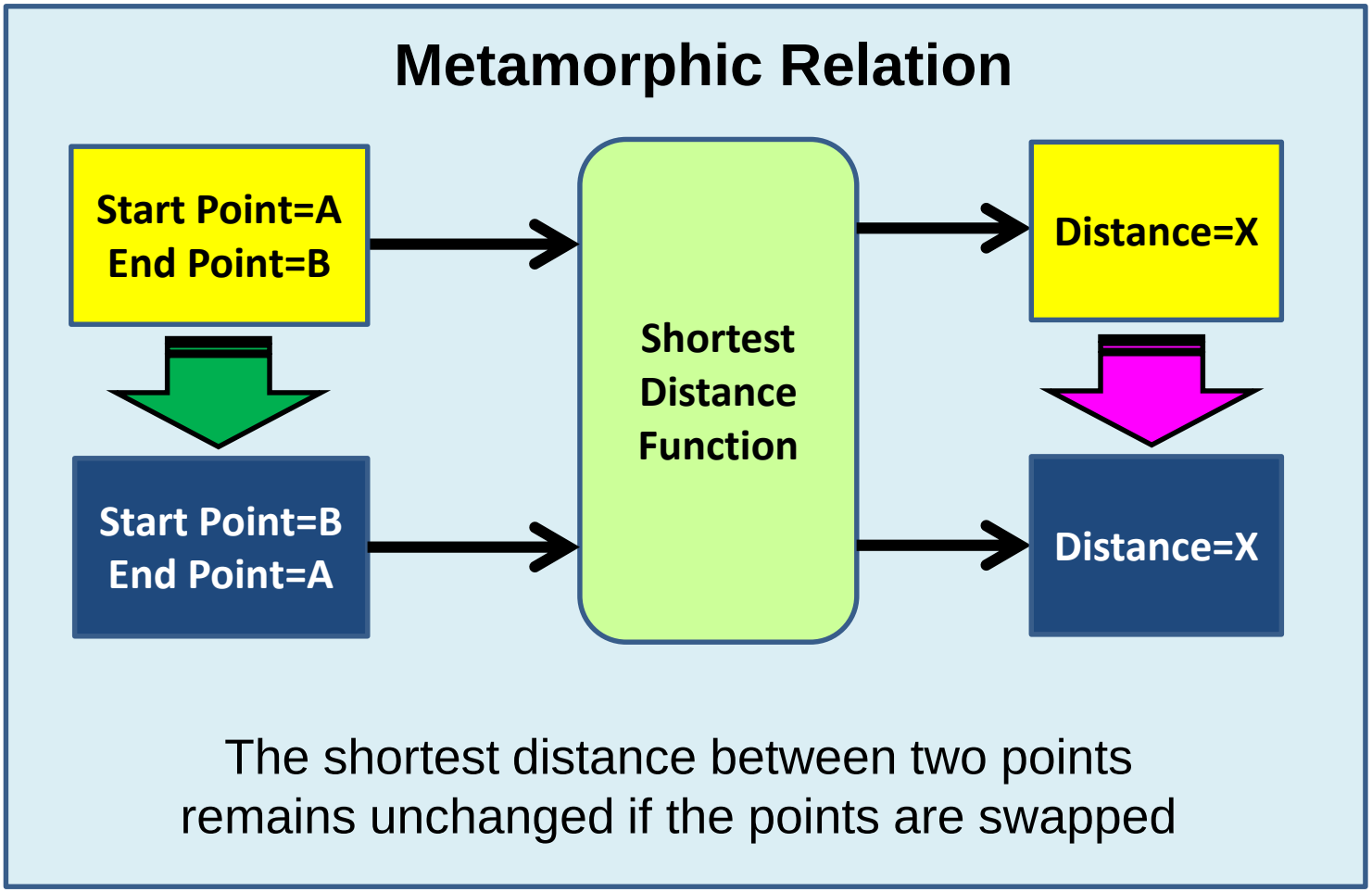
A MR comprises three parts:

- the relation between test inputs (green arrow);
- the required function
- the relation between test outputs (purple arrow);

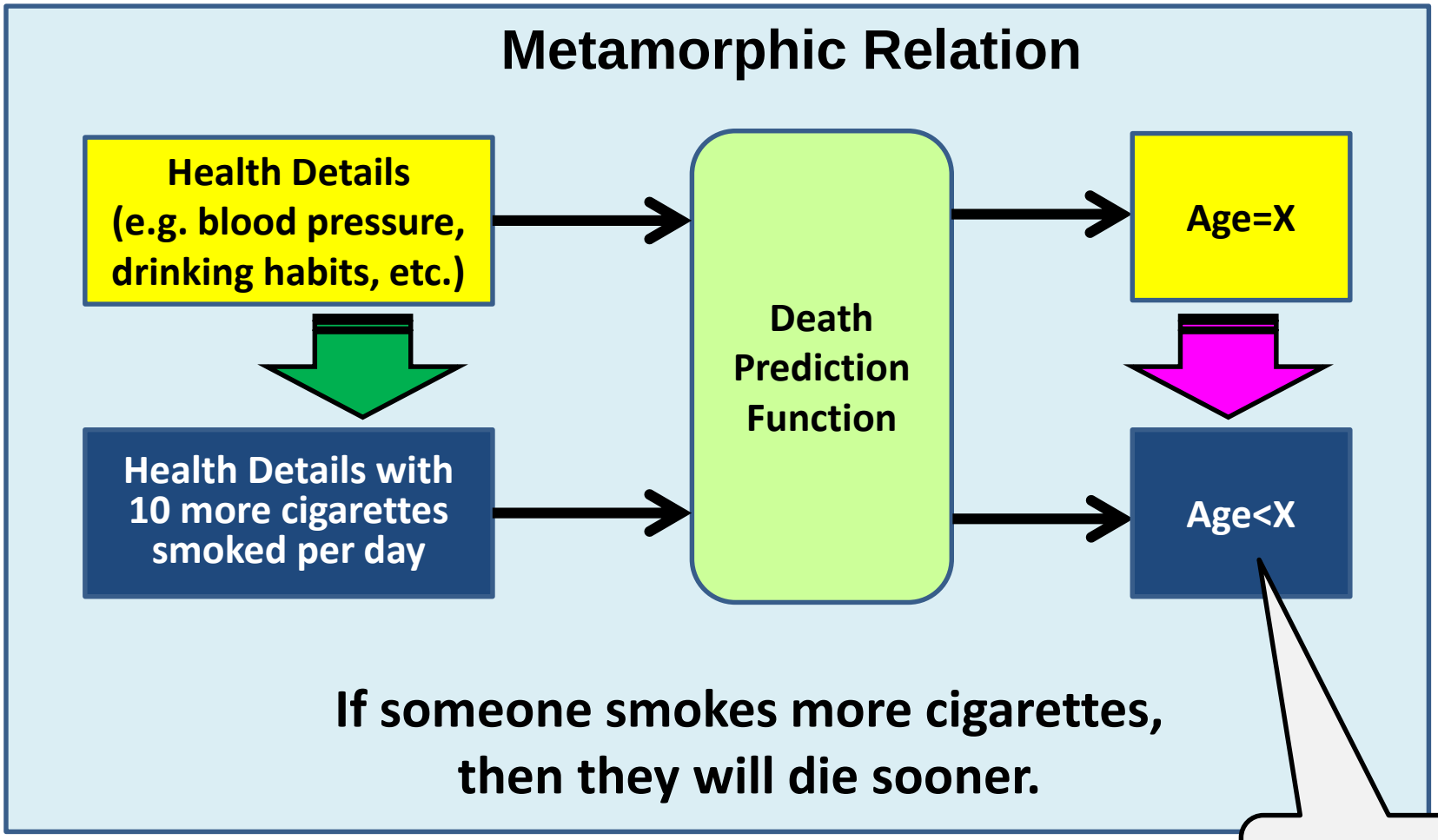
Metamorphic Testing Process

1. **Construct metamorphic relations (MRs)**
   - Identify properties of the program under test and represent them as metamorphic relations between test inputs and expected outputs, together with some method to generate a follow–up test case based on a source test case
2. **Review MRs**
   - Review and confirm MRs with customers and/or users
3. **Generate source test cases**
   - Generate a set of source test cases (using any testing technique or random testing)
4. **Generate follow-up test cases**
   - Use the metamorphic relations to generate follow–up test cases
5. **Execution of metamorphic test cases**
   - Execute both the source and follow–up test cases, and check that the outputs do not violate the metamorphic relation. Otherwise, the metamorphic test case has failed, indicating a bug

MT – Shortest Distance Example



MT – Death Prediction Example



MR – Standard Deviation Example

Metamorphic Relation

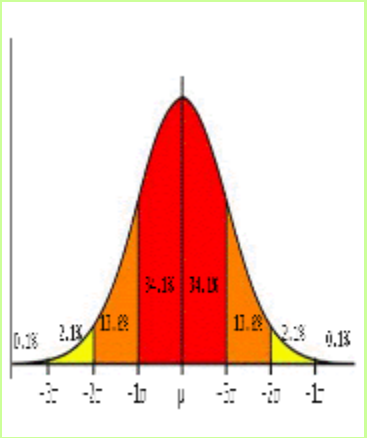
10 integers

same 10 integers in reverse order

same 10 integers in random order

same 10 integers each multiplied by 2

Maths Standard Deviation algorithm



SD

SD

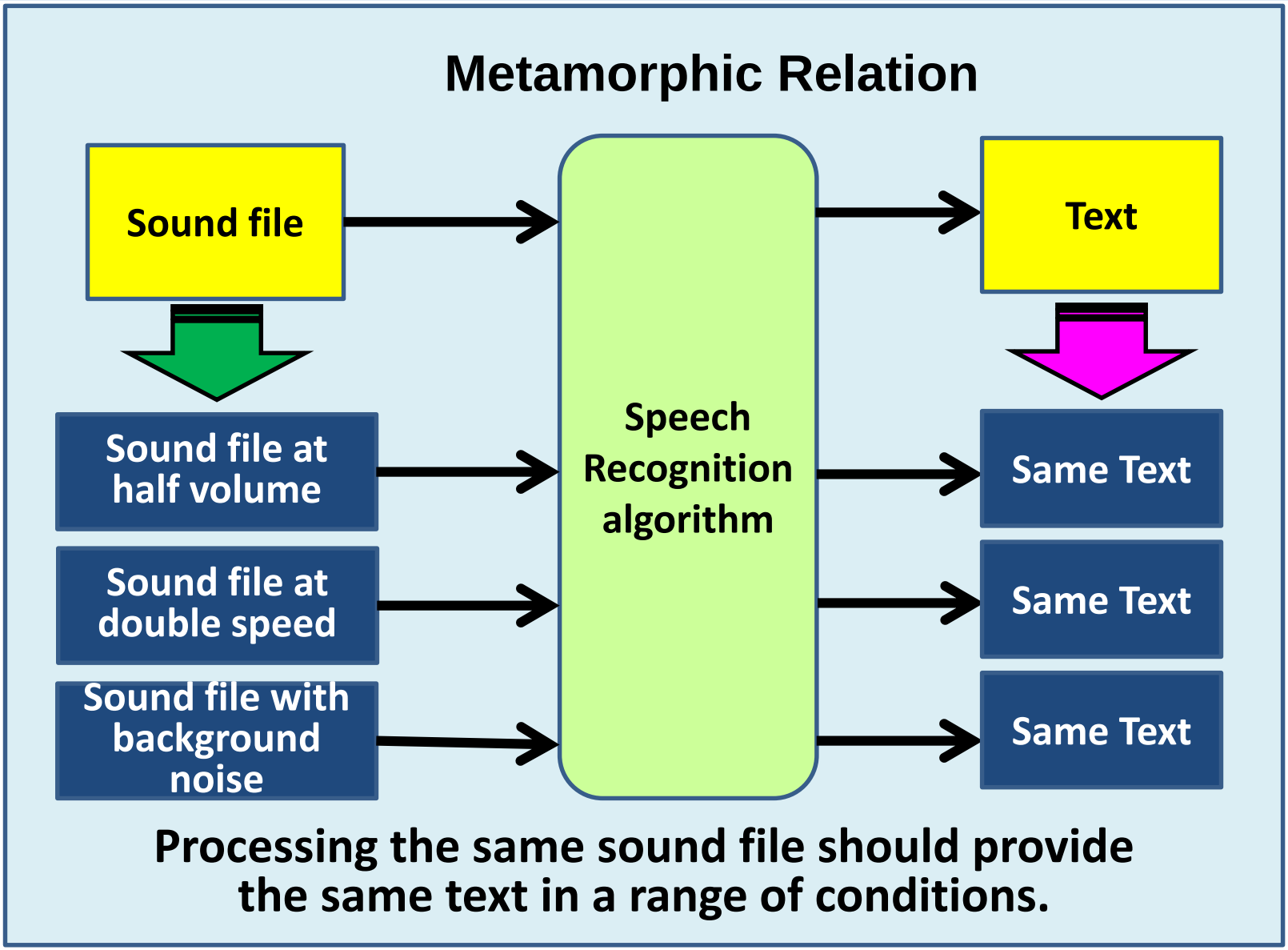
SD

SD * 2

Properties of the standard deviation statistical function (that specifies by how much the members of a group differ from the mean value for the group).

MRs can have more than one pair of test cases

MT – Speech Recognition Example



Example Application of MT

- **Studies of successful use in testing:**
  - bioinformatics
  - web services
  - embedded systems
  - components
  - compilers
  - databases
  - machine learning classifiers
  - online search functions and search engines
  - software product lines
  - security

Advantages of MT

- **Simple concept**
- **Testers without much experience or expertise can learn how to apply MT in a few hours**
- **MR identification is relatively simple, although it cannot be completely automated**
  - the remaining steps, including test case generation, execution, and verification can be easily automated
- **Similar cost as traditional testing techniques**

Summary - Metamorphic Testing

- **Needs good MRs to be effective**
- **MT & Requirements Engineering**
  - we need to investigate new specification practices to support the identification of Metamorphic Relations
  - even without a complete specification, testers can still identify useful MRs and generate effective test cases
    - the best MRs create quite different test cases from the source test case (e.g. covers different paths through the code)
- **Research shows that only 3 to 6 diverse MRs can reveal over 90% of the faults that could be detected using a traditional test oracle**

AI System Characteristics and Acceptance Criteria

Functional & Non-Functional Characteristics

ISO 25010 Product Quality Model

Functional Suitability	Performance Efficiency	Compatibility	Usability	Reliability	Security	Maintain-ability	Portability
Functional completeness	Time behaviour	Co-existence	Appropriateness recognizability	Maturity	Confidentiality	Modularity	Adaptability
Functional correctness	Resource utilisation	Interoperability	Learnability	Availability	Integrity	Reusability	Installability
Functional appropriateness	Capacity		Operability	Fault tolerance	Non-repudiation	Analysability	Replaceability
			User error protection	Recoverability	Accountability	Modifiability	
			User interface aesthetics		Authenticity	Testability	
			Accessibility				

Functional Testing — **what** the system does

Non-Functional Testing — **how** the system does it

AI – Characteristics & Testing

- **Nine AI-specific characteristics have been identified (see next slide)**
- **Any testing also needs to consider specified vs non-specified requirements**
  - although this is not specific to AI
- **In the near future, energy efficiency will become more relevant and we will need to start testing this**

AI-Specific Characteristics

- Adaptability
- Autonomy
- Evolution / Degradation
- Flexibility
- Fairness / Bias
- Performance
- Transparency / Explainability
- Complexity
- Non-Determinism

AI-Specific Characteristics – Adaptability

- **Ability to react to changes in the environment in order to continue to meet both functional and non-functional requirements**
  - characterized by self-configuration, self-healing, self-protection and self-learning
- **Adaptability requires a system to actively or passively gather information about its environment**
  - exploration (active gathering) can be dangerous (e.g. pushing the boundaries of a flight envelope) especially for safety-related situations
- **Adaptability requirements should specify**
  - environment changes to which the system should be able to adapt
  - requirements on the adaptation process
- **Adaptability Testing**
  - would need to be based on environment modification (or mutation)
  - functional and non-functional characteristics would need to be tested
    - regression testing, ideally automated, seems the obvious approach
  - should also test the adaptation process itself, e.g.
    - how quickly does the system adapt?
    - what resources does it use to adapt?

AI-Specific Characteristics – Autonomy

- **Ability of the system to work for sustained periods without human intervention**
- **Level of human intervention should be specified for the system – so part of its functional requirements**
  - e.g. system will maintain cruise condition until one of the following occurs
- **Autonomy could also be considered in combination with adaptability or flexibility**
  - e.g. adaptability or flexibility without human intervention
- **Autonomy Testing**
  - Negative testing could be used to try and force the system out of its autonomous behaviour and request intervention in unspecified circumstances
  - Testing could also be used to 'fool' the system into thinking it was in control when it should request intervention
    - e.g. scenarios at the boundary of its operational envelope – suggesting the application of boundary value concepts to scenario testing

AI-Specific Characteristics – Evolution

- **Concerned with two situations:**
  - user requirements change
    - this could be for many reasons, even based on interaction with the system itself
  - system behaviour changes
    - this could be that the system learns new behaviour as it is used (e.g. self-learning)
- **Negative changes are known as degradation, drift and staleness**
- **Evolution Testing**
  - a form of maintenance testing, run on a frequent basis
  - typically needs to monitor specified system goals, such as
    - performance goals (e.g. accuracy, precision, sensitivity, etc.)
    - data bias (e.g. Microsoft Tay chatbot).
- **Expected response is for the system to be re-trained, perhaps with new data**

AI-Specific Characteristics – Flexibility

- **Ability to work in contexts outside the initial specification**
  - i.e. change system behaviour according to its actual situation to satisfy its objectives
- **Flexibility should be explicitly specified in the requirements**
  - e.g. use of ‘must’, ‘may’, close to’, etc., or use of probabilities and possibilities in specifications
  - e.g. RELAX requirements language
- **Flexibility can be achieved using several technical mechanisms, such as reactivity, pro-activity, interaction, adaptation or self-learning**
- **Flexibility Testing**
  - requires tests that extend the system’s original behaviour
  - metamorphic testing includes the use of metamorphic relations which can be used to extend the initial specification (within limits)

AI-Specific Characteristics – Bias

- **ML is based on discovering and generalizing patterns in the training data**
  - these patterns are then used for future predictions and classifications
- **Fairness can be compromised when bias in the training data is reflected in the system model**
- **The solution is to remove undesirable bias**
  - we need to recognize both explicit and implicit bias
  - comprehensive, specific and objective goals can prevent bias
  - if we know it, we can remove the source of the bias (e.g. sex data) from the training data – or we can fix the data to ignore this feature
- **Or accept the bias – publishing the training data is one way of being transparent about any possible bias**
- **Bias Testing**
  - ensure training data is free from bias through reviews
    - needs expert reviewers who can identify possible features that create bias
  - ensure the system is free from bias through independent testing
    - by creating bias-free testing sets

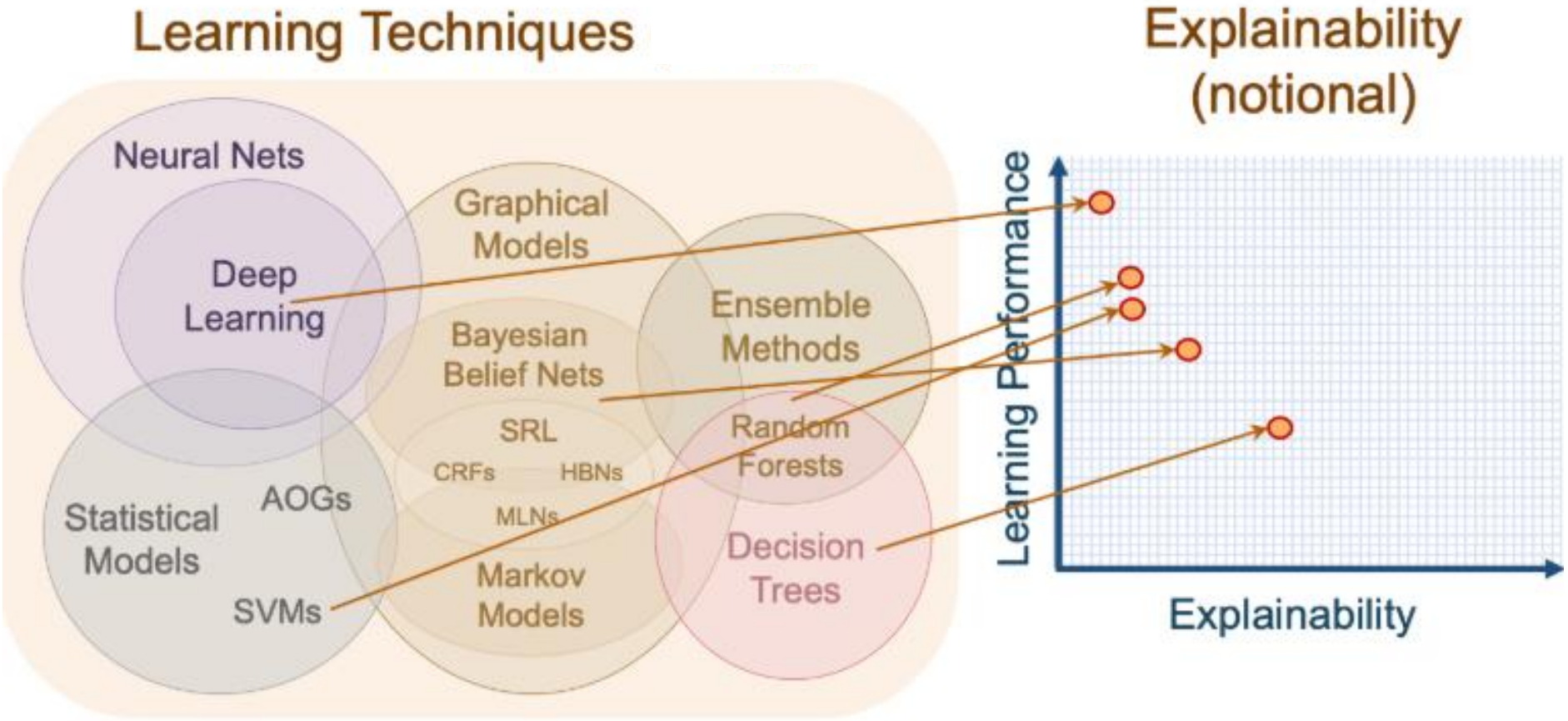
AI-Specific Characteristics – Performance

- **For ML Models**
- **Performance metrics**
  - e.g. accuracy, precision and recall
- **These metrics should be agreed and defined as part of the system requirements**
- **Performance Testing**
  - ML Frameworks (e.g. TensorFlow) or an open source library (e.g. TensorFlow Model Analysis) will provide these measures for a given data set

AI-Specific Characteristics – Transparency

- **Measure of how easy it is to see how an AI-Based system came up with its result (also known as explainability)**
- **The required level of transparency changes from system to system**
  - e.g. marketing campaign vs jail term setting
  - regulated areas are more likely to need transparency (e.g. GDPR, safety-related)
- **The required transparency should inform the AI framework choice**
- **Achieving transparency may be traded off against required accuracy**
- **May address transparency by publishing details of the chosen framework, training algorithm and training data**
- **Explainable AI (XAI) covers ways to make AI-Based systems more explainable**
- **Transparency Testing**
  - qualitative activity ideally performed by users to determine if:
    - the system workings are understandable
    - the provided explanation is satisfactory

Example Levels of Explainability



https://www.darpa.mil/attachments/XAIProgramUpdate.pdf

AI-Specific Characteristics – Complexity

- **AI-Based systems can be extremely complex**
  - it is not unusual for a deep neural network to have more than 100 million parameters
- **AI-Based systems are often used when there is no alternative, due to the complex nature of the problem**
  - e.g. making decisions based on big data
- **The complexity of such systems creates a test oracle problem**
  - it may require several experts much time to agree a single test case
- **Testing complex systems**
  - several test techniques can be used to address the test oracle problem, including
    - A/B testing
    - back-to-back testing
    - metamorphic testing

AI-Specific Characteristics – Non-Determinism

- **Non-Deterministic systems are not guaranteed to produce the same outputs from the same inputs (as for a deterministic system)**
- **Sub-characteristics that cause non-determinism:**
  - probabilistic nature of AI
  - complexity of AI-Based systems
- **Leads to the test oracle problem**
- **Testing non-deterministic systems**
  - several test techniques can be used to address the test oracle problem, including
    - A/B testing
    - back-to-back testing
    - metamorphic testing

Thanks for Listening…