
expo:QA 24
MADRID
May 28th,
29th, 30th
2024
sogeti
expo:QA
expoqa.com

New trends in Quality Assurance

Leveraging **AI** and **Blockchain** for Advanced Software Testing

expoQA MADRID
28th, 29th, 30th May
2024

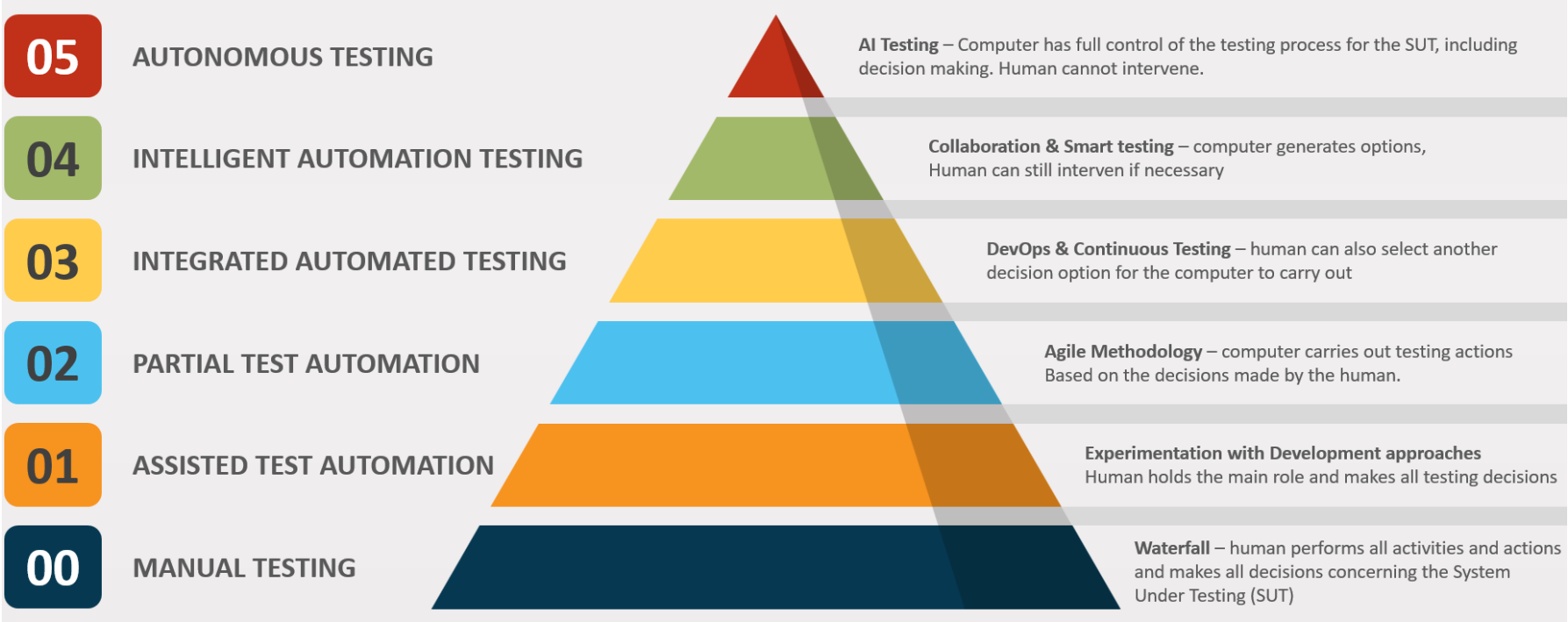
Diogo Goncalves Candeias

QA & Automated Solutions Service Manager
Head of Quality Assurance @Exceltic

AI & QA

Maturity Assessment

The traditional testing method is often riddled with bottlenecks, including a high volume of tests, siloed automation, a shortage of end-to-end visibility of requirements, etc. Many have emerged as frontrunners in next-gen software testing adoption by leveraging **artificial intelligence (AI)-based approach** that harnesses the power of **machine learning (ML)** to orchestrate quality across the testing process.

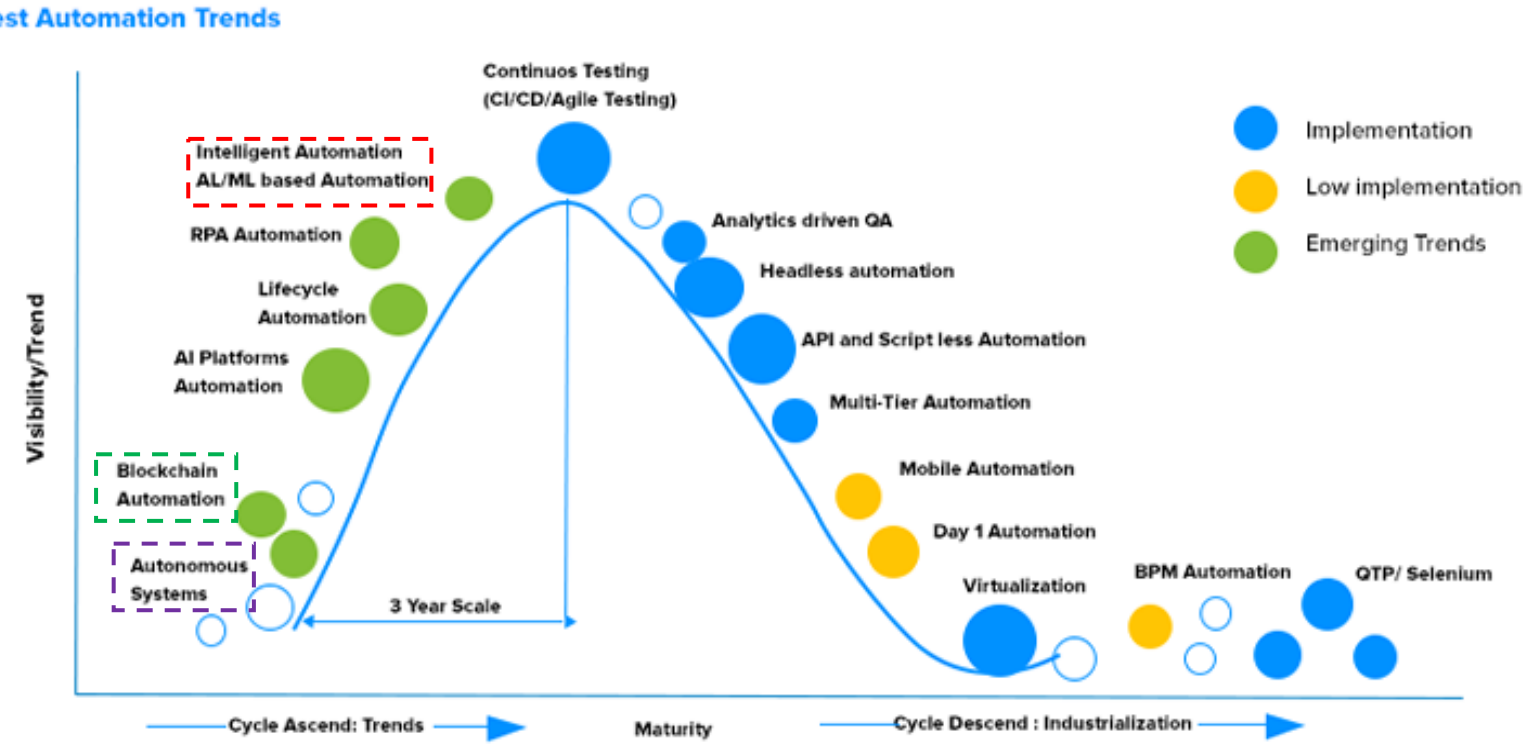


Achieving testing maturity by automating your testing practices can thoroughly transform your product development and upkeep process.

But how do you get there, and what does growth look like?

Automation Trends

For many organizations, it's difficult to find the correct test automation approach and, furthermore, it is often cumbersome to implement a test automation infrastructure. The real weakness of automated testing is that, for the most part, the test authoring, maintenance and results analysis are still manual. **This is where AI and ML** (pattern-recognition technology, uses algorithms to predict trends).



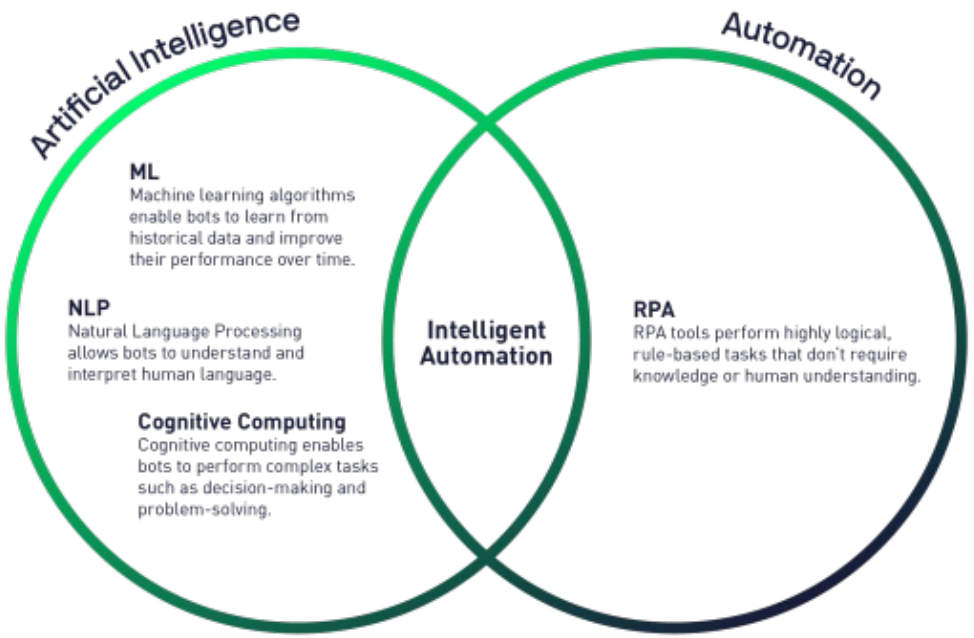
When **AI and ML** started to weave themselves into test automation practices, IT leaders could see the potential of automation in testing infrastructures.

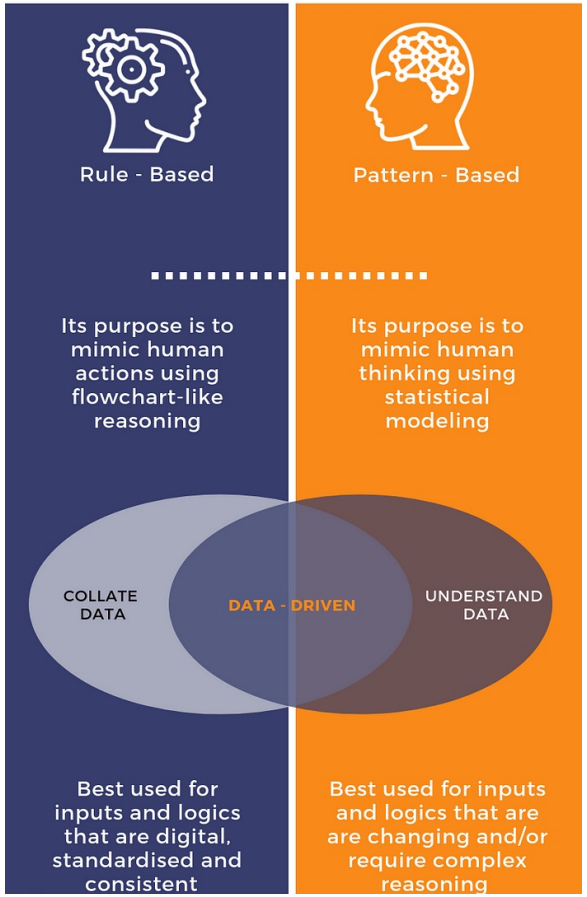
While testing creates an excess of information and logs, **AI and ML tools** cut through the noise and spot irregularities within apps.

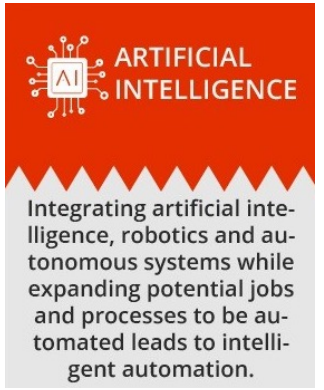
AI vs Automation

Automation executes predefined tasks, reducing manual intervention and enhancing efficiency.

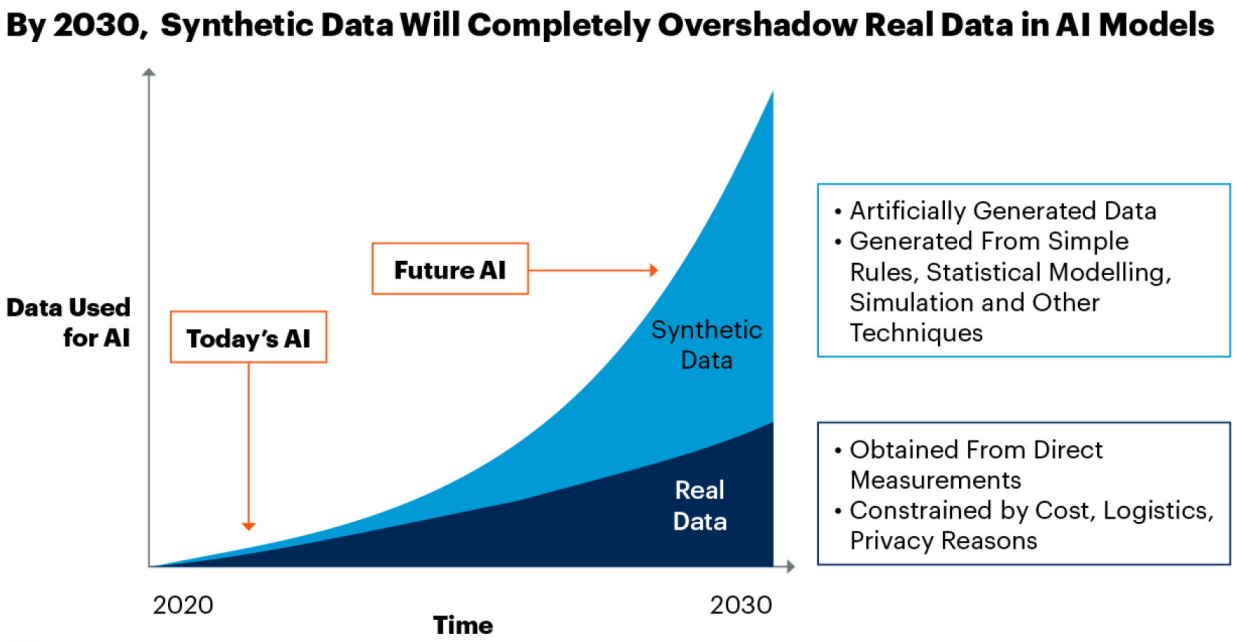
AI, incorporating machine learning and advanced algorithms, learns from data, adapts, and makes decisions without explicit programming.

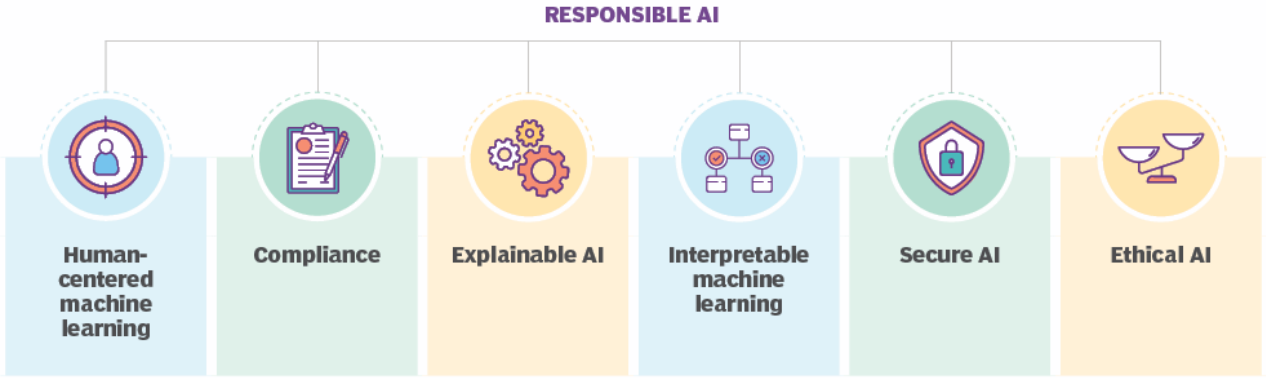






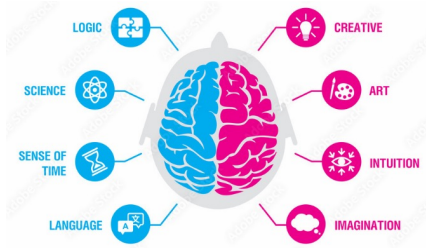
Challenges on the Path to Autonomous





Complex Test scenarios

Not all test scenarios are good candidates.
Achieving autonomous exploratory testing can be important milestone for the AI field.



Test Data Management

Ensuring availability of realistic and diverse test data can be difficult
Good prompt engineering is needed
Data privacy and data masking

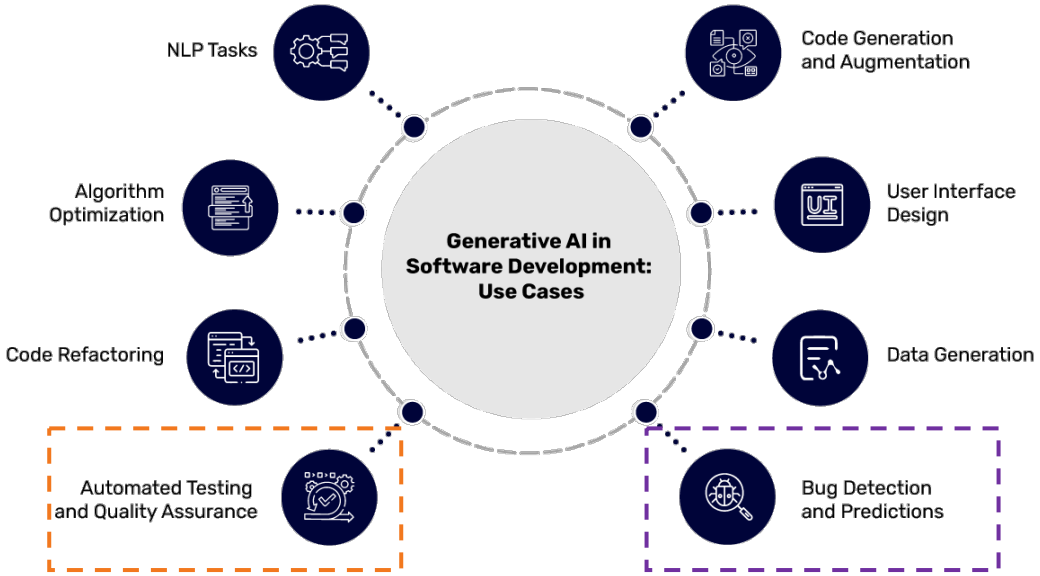
AI Model Training and Integration

AI models need training and fine-tuning
Continuous effort and investment
Integration with CI/CD pipelines can be complex

Bias and Accuracy

AI algorithms may introduce biases or inaccuracies in test case Generation, execution and defect detection

Gen AI Trends – from buzz to business value



Gen AI is at the forefront of testing trends in 2024.

This *shift* towards intelligent testing ensures quicker and more accurate testing processes, enhancing overall software quality.

Regression testing automation

AI-powered regression testing is an intelligent way to automatically generate test cases, identify relevant test scenarios, and prioritize tests based on their severity.

Defect Prediction

AI in software testing uses historical testing results to predict defect metrics for future releases. It can identify gaps in quality using predictive analytics and minimize redundancy.

Adaptive test case generation & smart test script maintenance

AI can generate test cases dynamically based on evolving requirements. It also helps maintain and update scripts automatically when a change occurs in the application under test.

Self-healing automation

can identify script breakages that may arise from any object or other changes during the development and testing process and resolve them quickly. Automatic bug triage is another benefit of AI-powered software testing.

Impact Radar for Generative AI

Gartner

Potential Gen AI Impact in Soft. Dev. Lifecycle

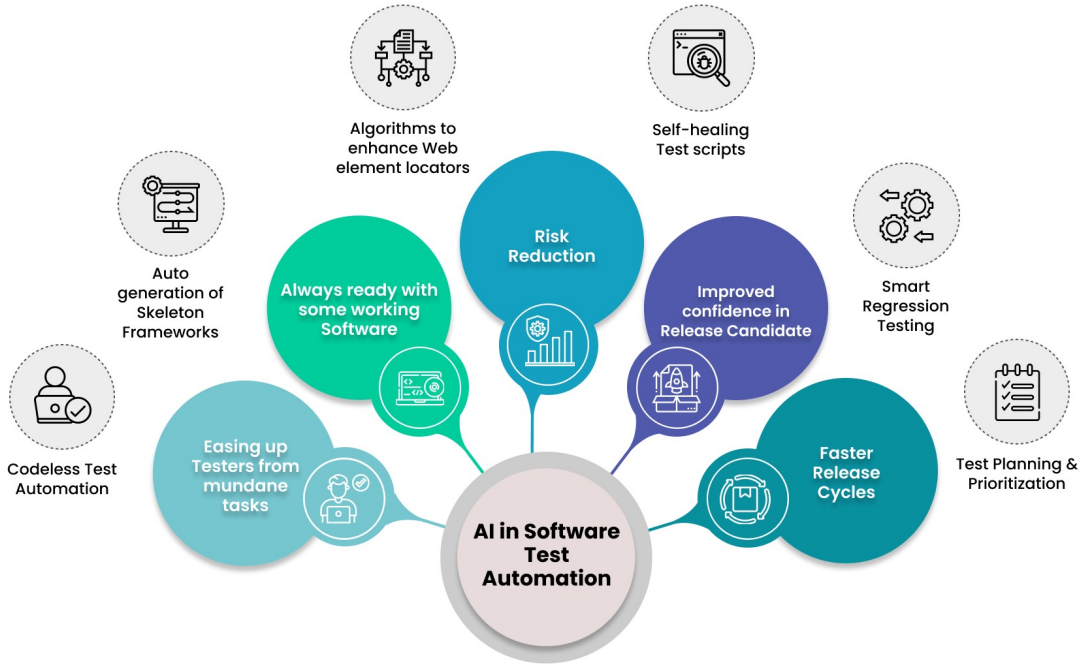
Test data generation: Generate synthetic data based on usage patterns and other domain insights by using GenAI. Deep learning models, such as variational autoencoder **(VAE)** and Generative Adversarial Network **(GAN)** models can improve data utility by feeding models with more data.

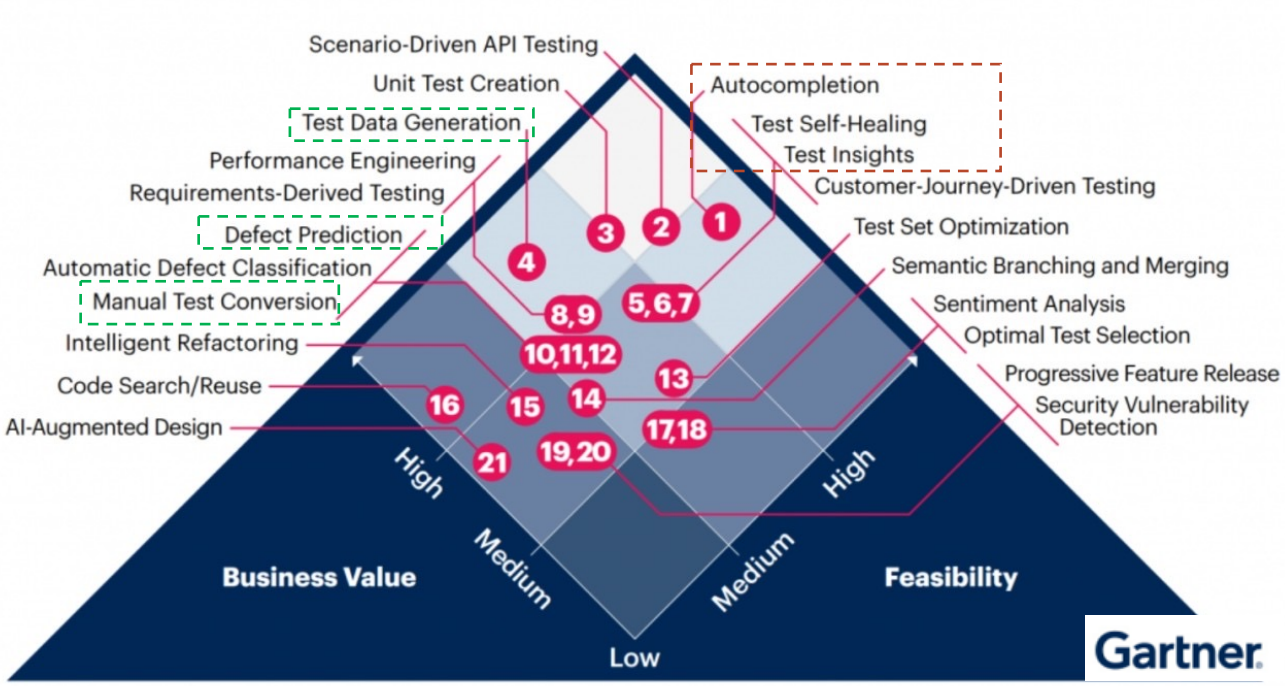
Test self-healing: Automatically update test scripts by using GenAI to identify changes in the application under test, such as updates to the UI or API, changes to the workflow or changes to the configuration.

Requirement-derived testing: Generate test scenarios by using natural language techniques such as **NLU and NLP** to automatically analyze requirements, user stories

Defect prediction: Identify gaps in quality and defect targets, minimize redundancy and improve the effectiveness and efficiency of testing processes by using GenAI and natural language techniques such as **NLP** to detect patterns

Test set optimization: Identify redundancies and similarities in test-case inventories; using GenAI to optimize execution sequencing, using NLP to identifying test coverage gaps





AI impacted Automation Test Areas

Test Data Generation	Test Case Generation	Test Scripts Generation	Self-healing scripts	Visual Comparison
Generate new test data based on previously used ones	Generate test cases based on acceptance criteria	Generate test scripts based on test cases	Low maintenance	Compare visual checkpoints against baselines

Create Business AI Model
Train the model
Generate & filter the data

Synthetic Data

gretel™ MOSTLY·AI

Statice SYNTHO

Manual test cases creation
Scope and Test Data used
Maintenance

taskade

Manual creation
Maintenance

Scripts Generation
Coding Assistance
Scriptless Automation Tests generation

TestCraft

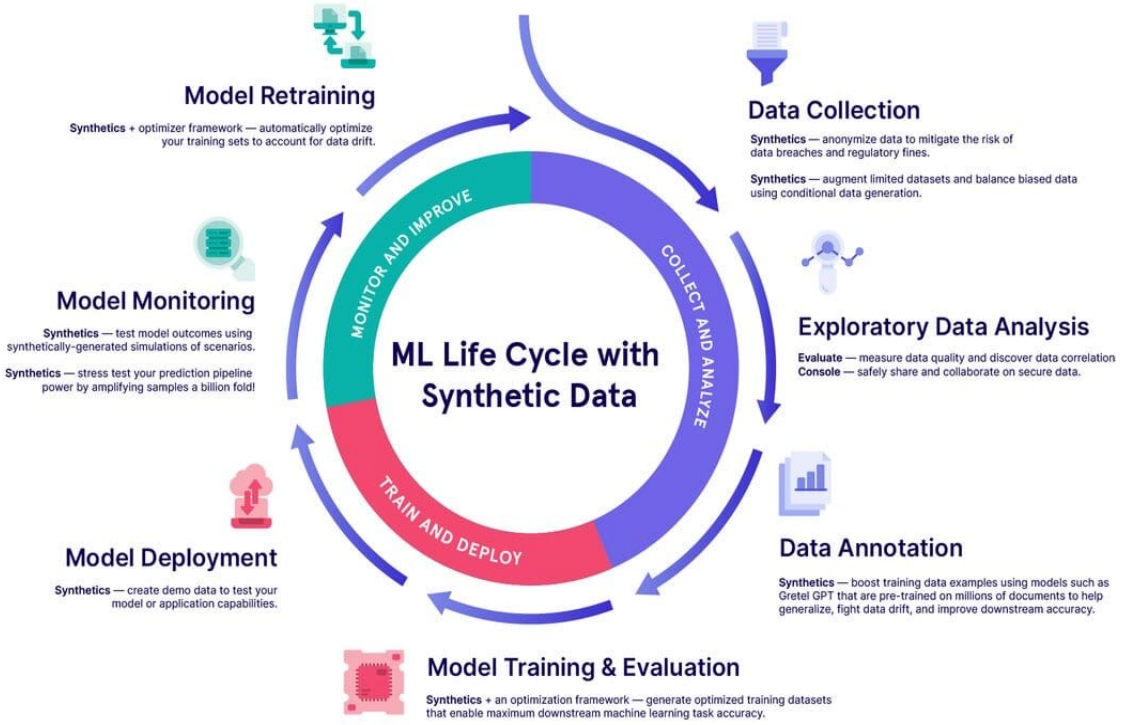
UI Locators maintenance
Changed layout/position
Changed tag names and attributes

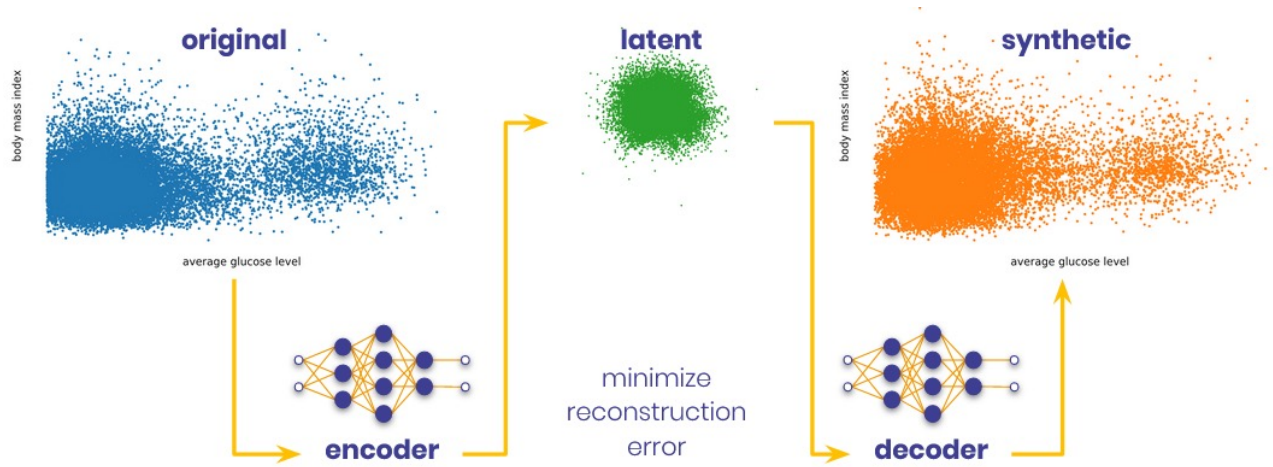
Healenium
Cypress Test Gold
Playwright
Applitools
TestRigor

Pixel perfect comparison
Mark changed areas
Ignore screen áreas
Support matrix

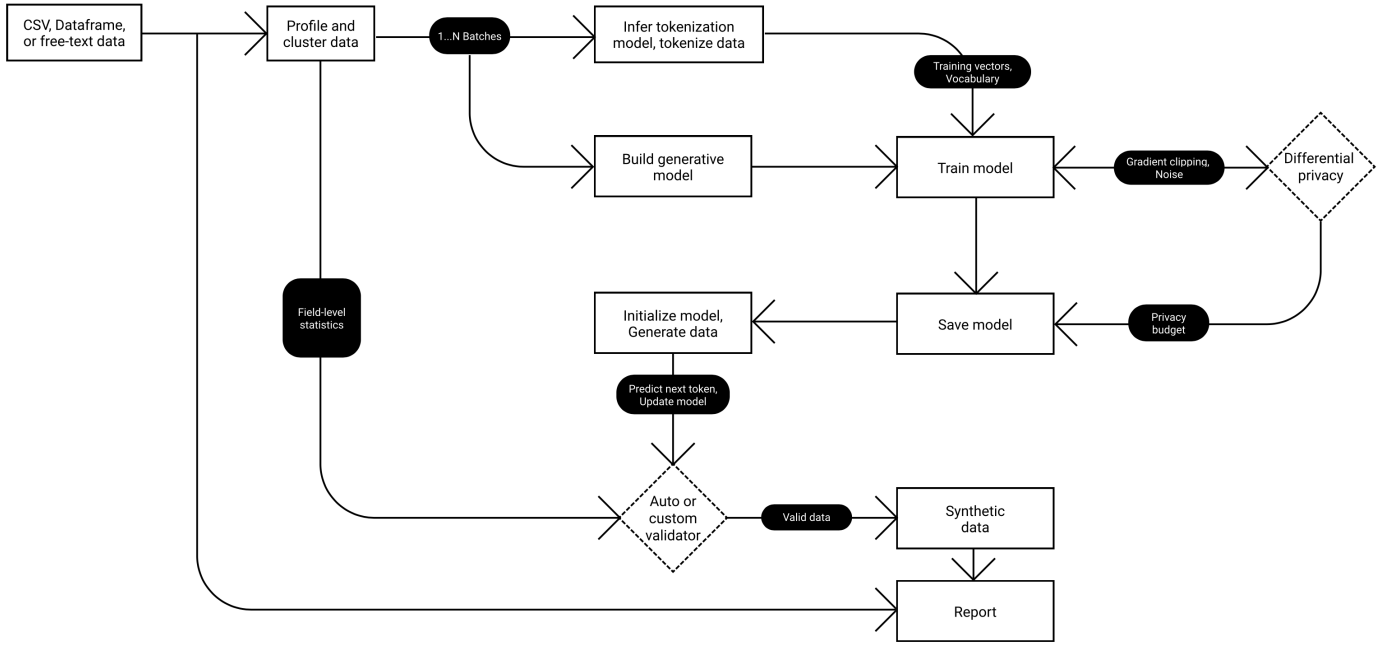
Applitools
pCloudy
App Percy
(BrowserStack)

Synthetic Data



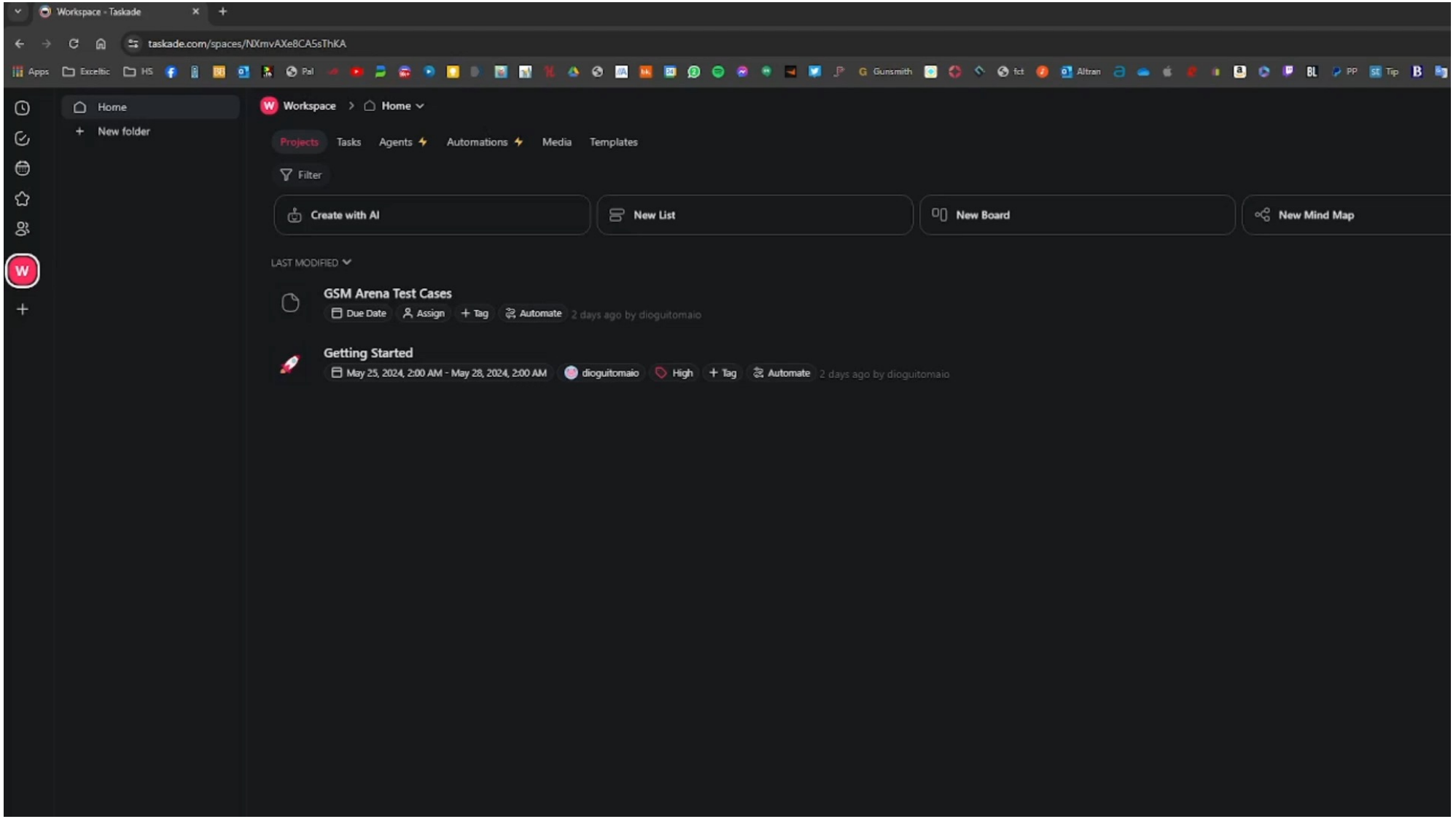


Gretel-synthetics application flow



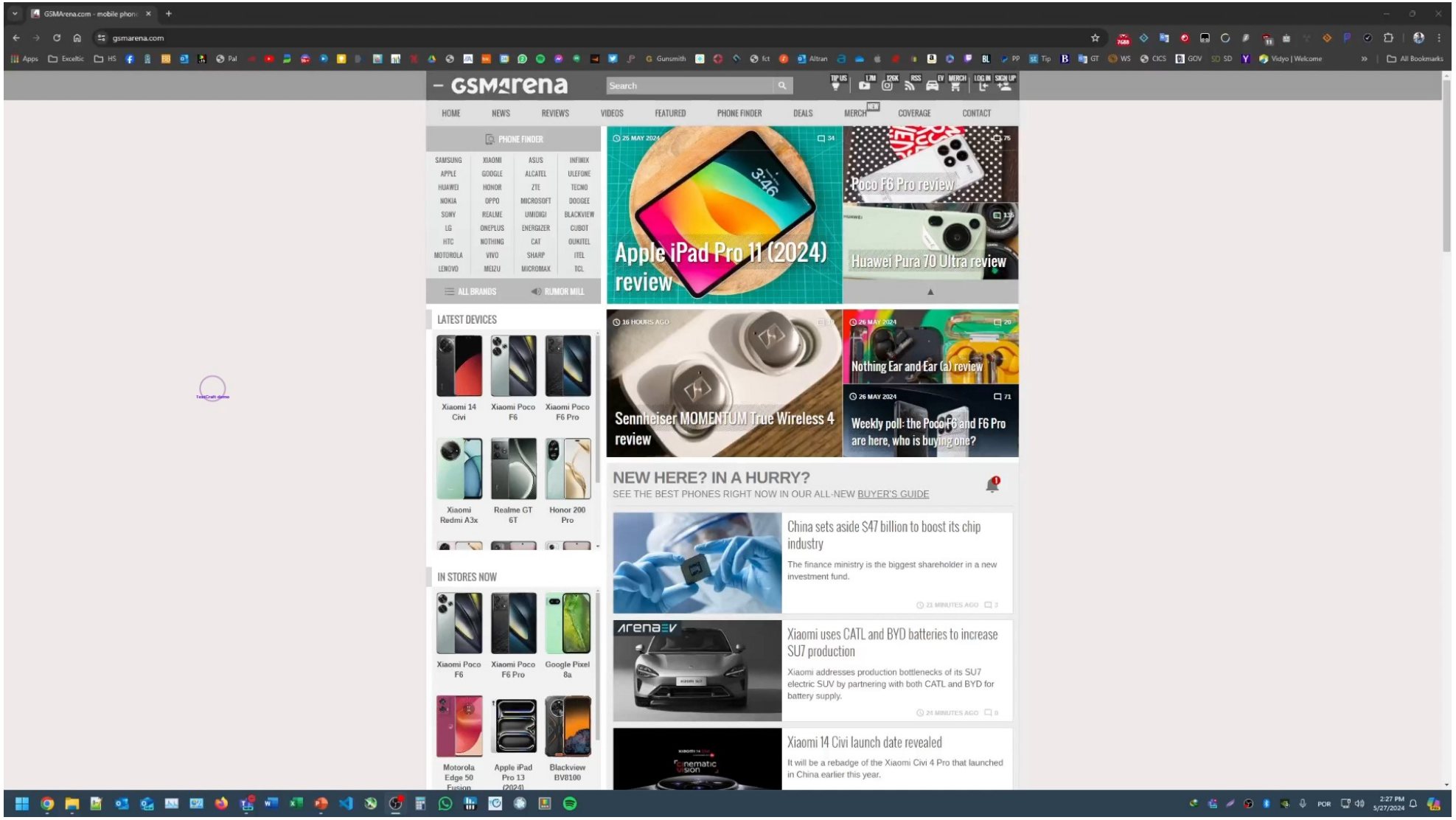
Test Case Generation

taskade

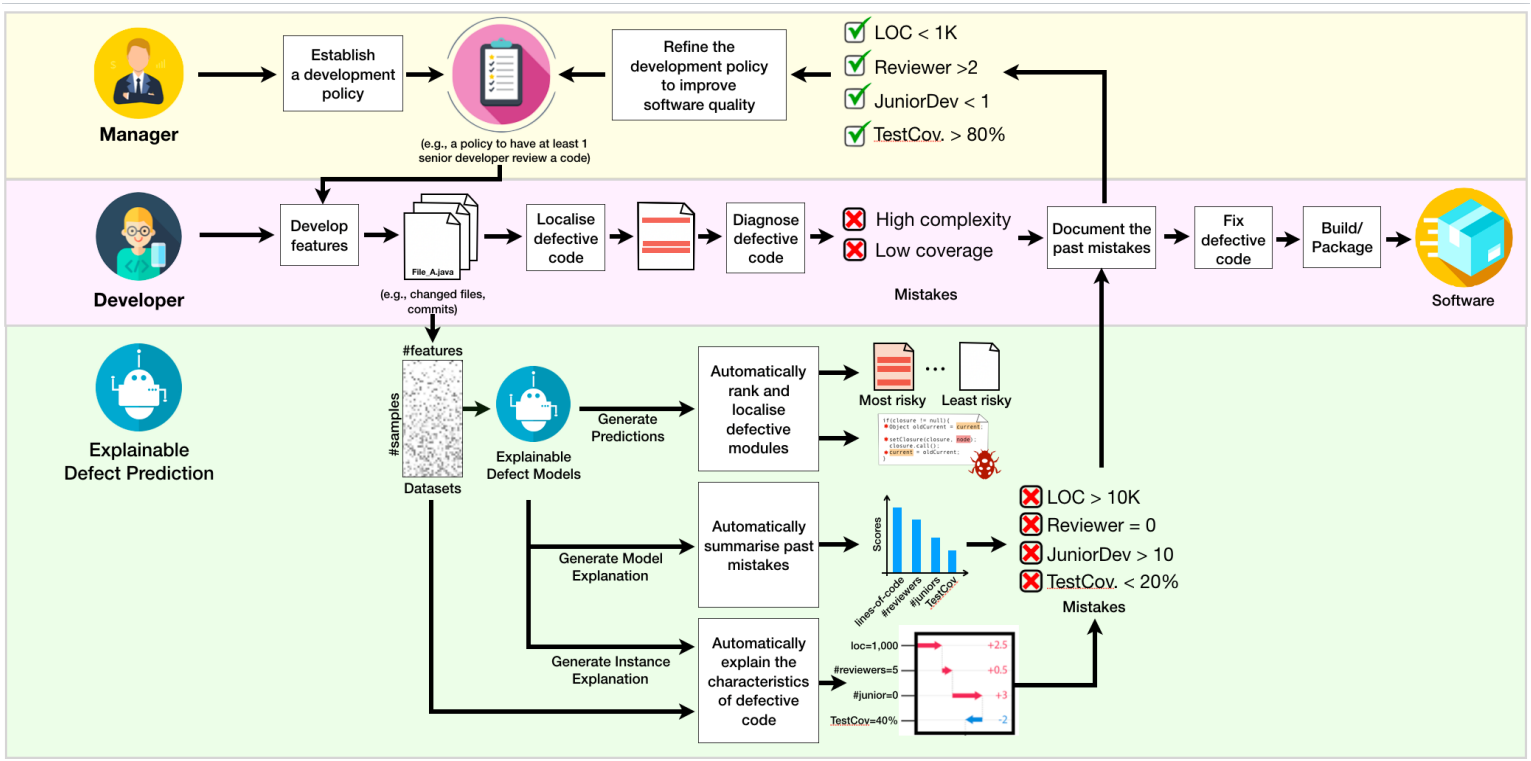


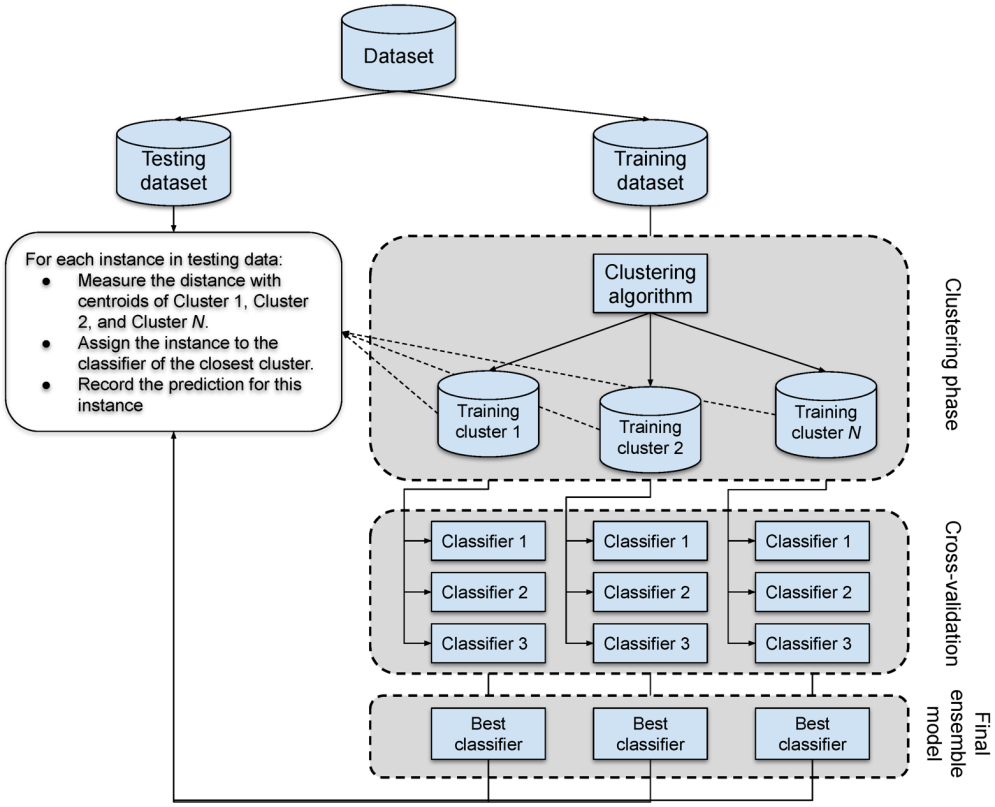
Test Scripts Generation

TestCraft



Defect Prediction and Test Case generation



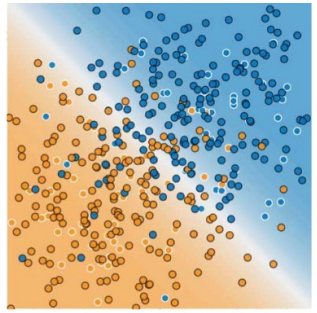


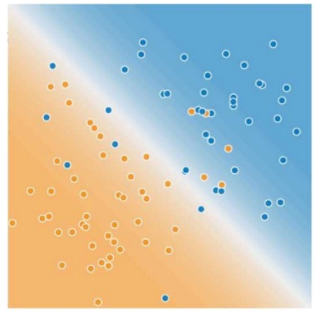
AI builds test cases for prioritized and common user journeys, then explores permutations around them to hunt bugs.

AI explores paths a human would never think of and intelligently refines exploration to maximize coverage.

AI churned code modules and create test cases to test quirks of particular developers and historical bug patterns.

AI predicts how the characteristics of this release will impact customer satisfaction.


Training Data


Test Data

Risks

Models "hallucinate": GenAI needs to be carefully validated and all search data verified

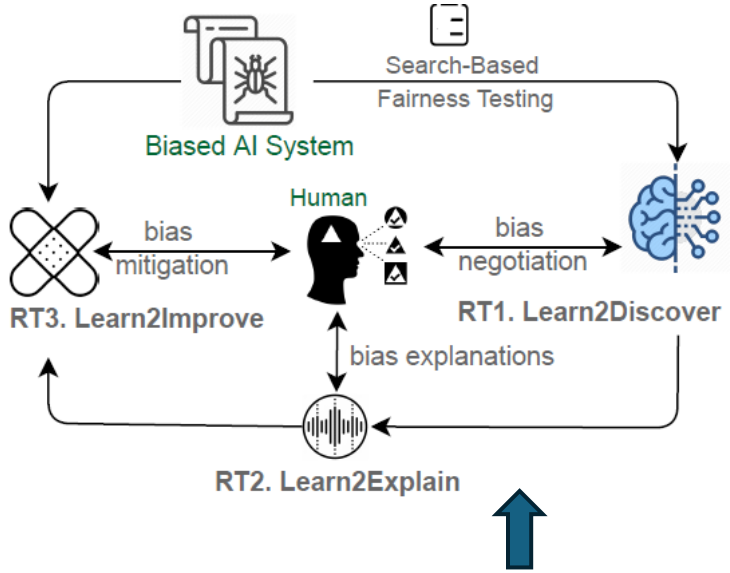
Speed: Growing number of models built with diverse stakeholders and complex pipelines
Exponential increase in data, storage and computing
Shortage of AI / ML talent

Exploitable loopholes: Once AI learns from historic data, understand patterns, it is highly possible that the model can identify loopholes and exploit those. So, the output generated by them might be in conformance to the rules but invalid.

Risk of unknown: It's not clear how the models treat and process inputs to generate the outputs. There are security risk while handling sensitive data.

Models can't unlearn: If there is a need to untrain a model this can be highly expensive and less efficient. The need can appear due to various reasons such as information security. Two option in this case are:

- ✓ Deletion of the specific data point from the training data and retrain the model (**very expensive**)
- ✓ **SISA** (Sharded, Isolated, Sliced and Aggregated) approach – data is isolated and processed in small parts and when there is the need to delete the data points it's more manageable.

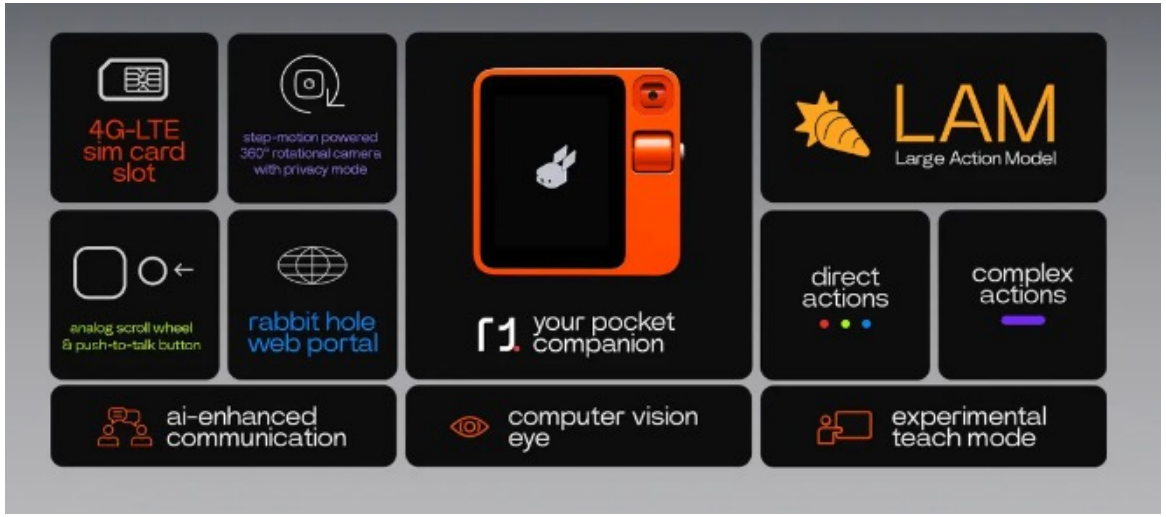


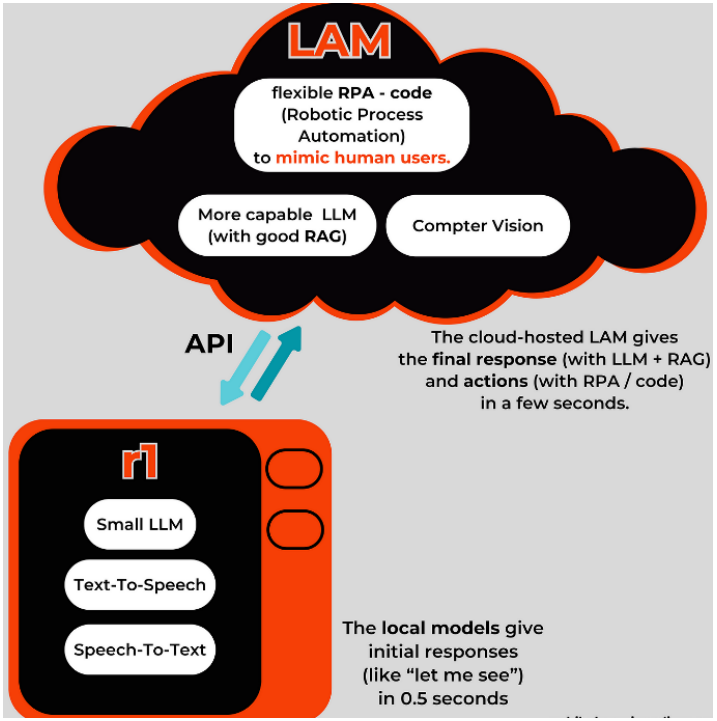
Rabbit R1 Case

“LAM stands for “Large Action Model.”

Model that learns how to use any software it runs into, takes actions, and gets better over time. It learns by studying how people use online interfaces, and then it can operate those interfaces in the same way that a human would.

Importantly, it also understands natural-language inputs. You ask it to do something, and it takes care of it for you.”



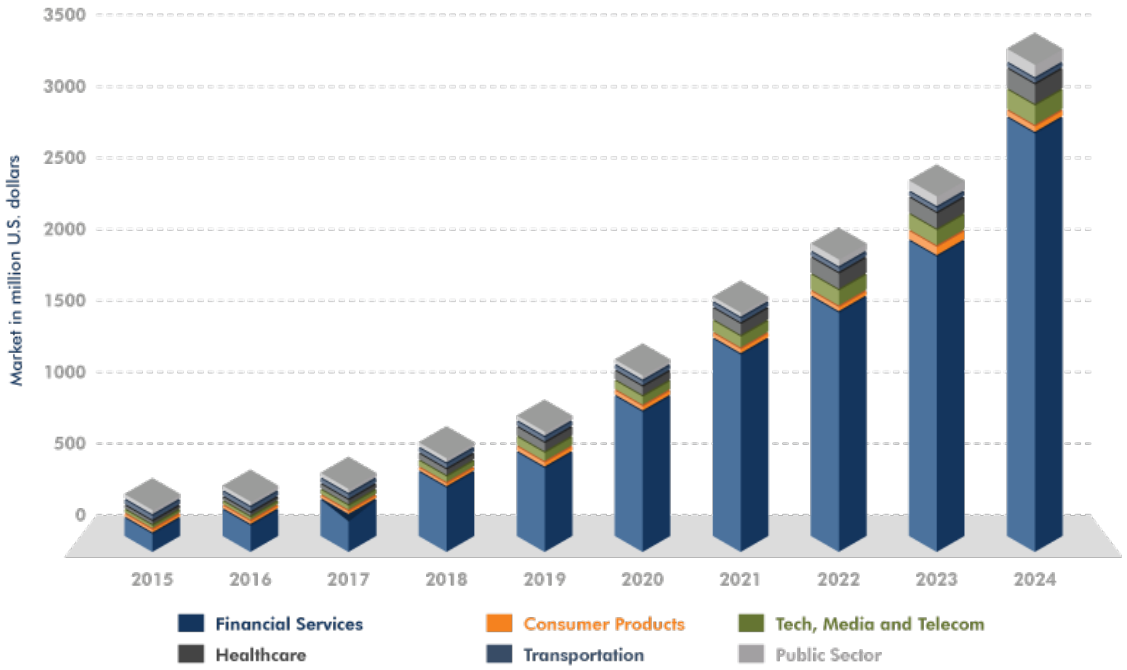


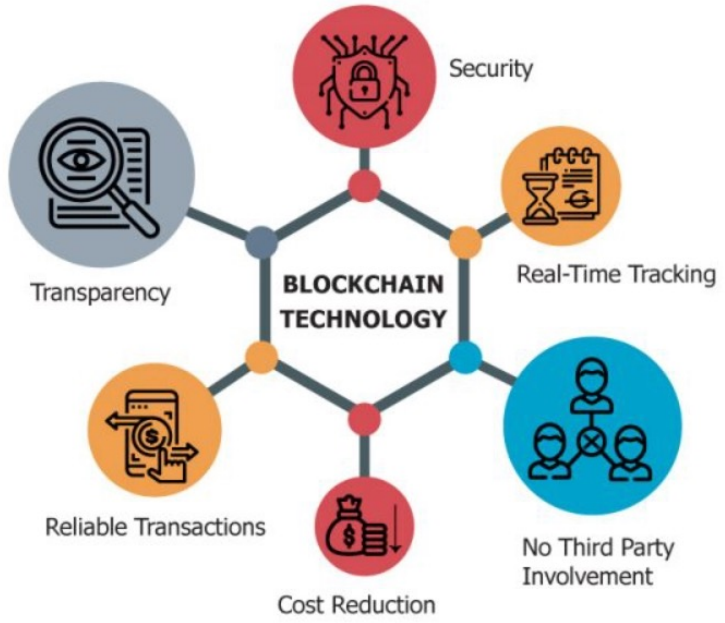
Blockchain Technology

Blockchain Impact

In 2024, blockchain technology is transforming the landscape of software testing. By leveraging the decentralized and immutable nature of blockchain, software testing processes are becoming more **transparent, secure, and efficient** than ever before.

Blockchain enables the creation of tamper-proof records of test results, ensuring the integrity of data throughout the testing lifecycle. Moreover, **smart contracts** built on blockchain facilitate automated test execution and validation, reducing manual intervention.



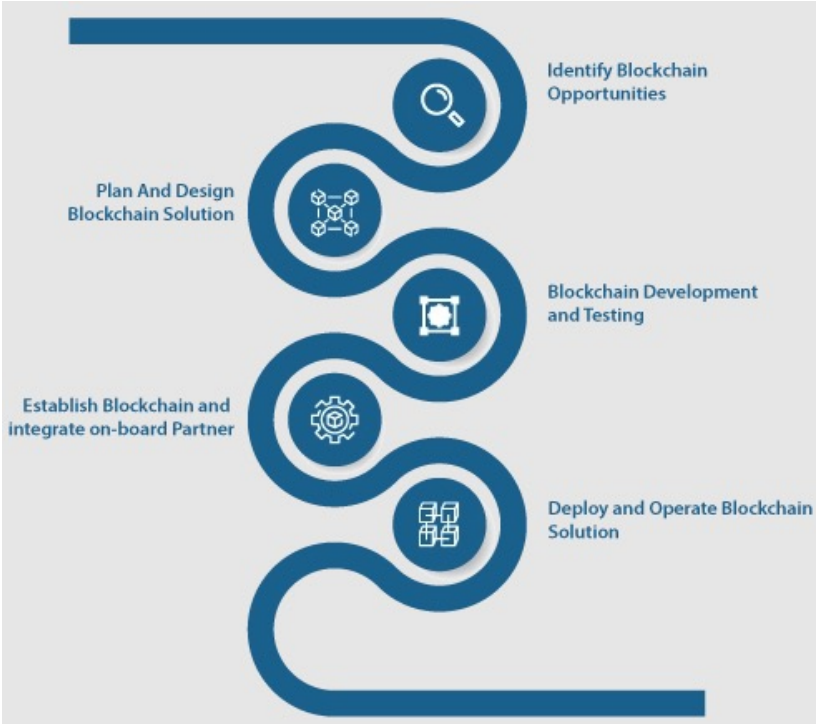


Why Should Businesses Adopt Blockchain Testing?

Businesses across all industries are increasingly realizing the value of blockchain technology and investing heavily in blockchain-based applications.

It is thus **crucial for QA engineers to learn how to assess blockchain-based applications** with expanding deployment and integration capabilities. This will help –

- ❑ Ensure a secure infrastructure for your business.
- ❑ Eliminate flaws in a decentralized ledger.
- ❑ Validate all the entities of the blockchain system.
- ❑ Develop a blockchain ecosystem that functions as expected.
- ❑ Secure the blockchain technology and connected infrastructure.
- ❑ Reduce the risks of adding new applications and facilitate smooth revalidation.



Accelerate Adoption +

- "Shift Left" approach with early test automation
- **Performance, Security** testing
- **Service Virtualization** with digital assurance services
- **API, Functional/Non-Functional t**esting

Process Driven Development

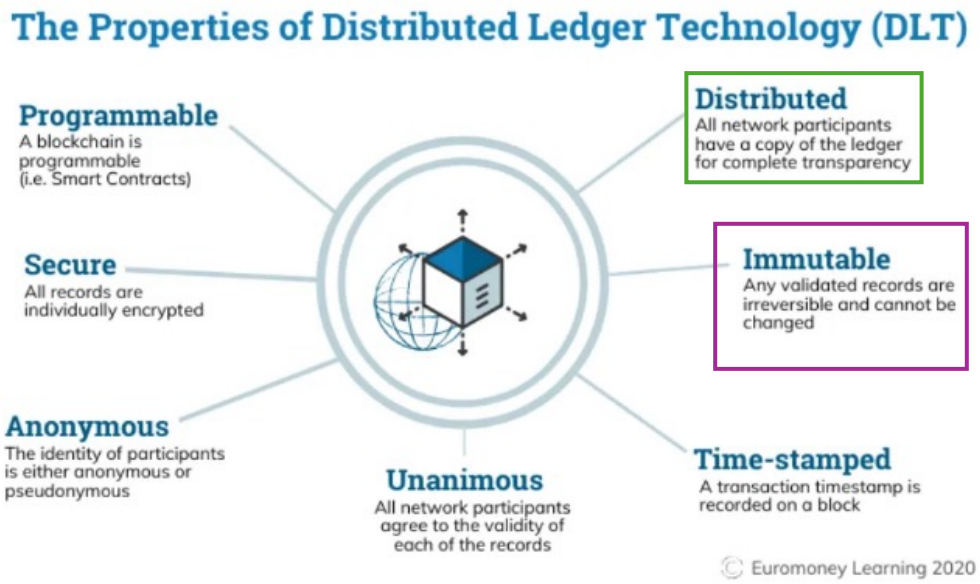
Risk based testing using design failure modes, effects & criticality analysis

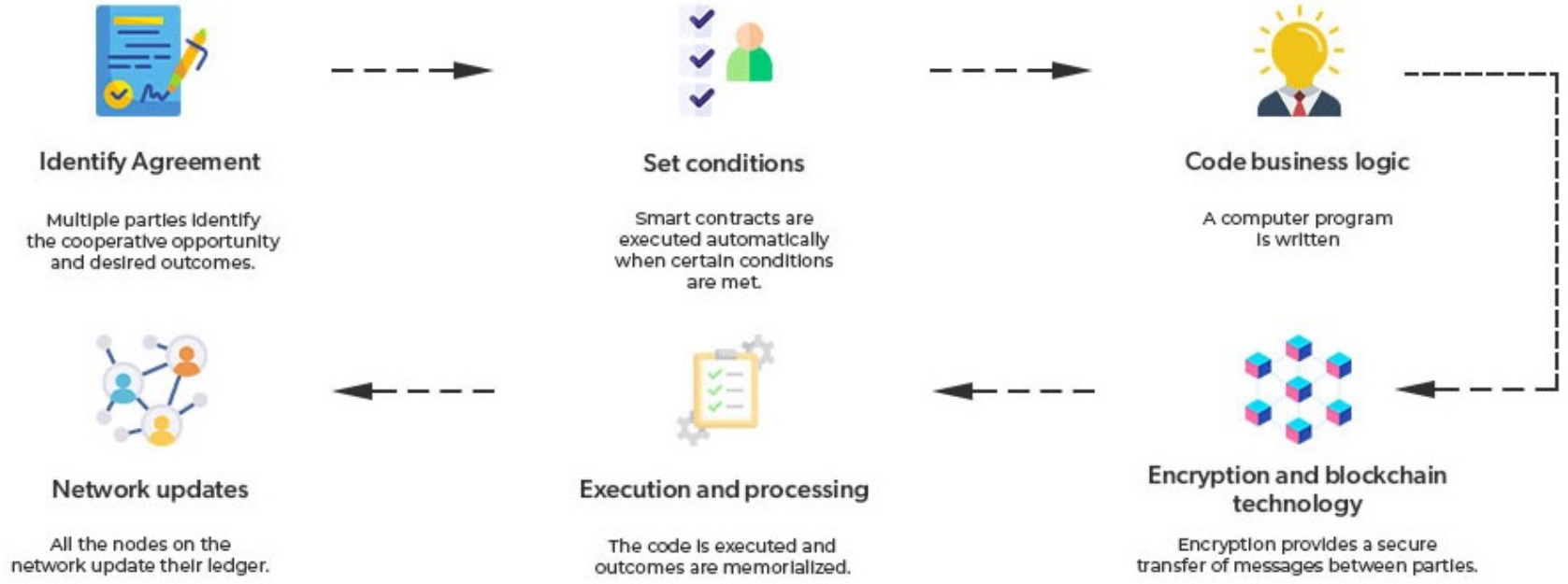
+ **Tool agnostic set up**

Blockchain Components

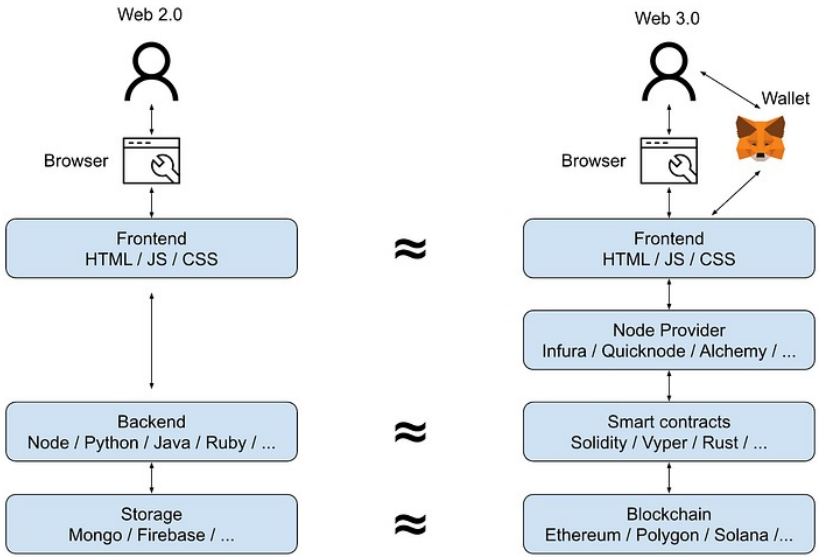
Smart Contracts

Business rules coded on blockchain
Stores rules for negotiating the terms of an agreement, automatically verifies fulfillment
Executes the agreed terms
Fraud resistant
Immutability – can never be changed or tampered
Distributed – outcome is validated by everyone in the network, just like any transaction in a blockchain

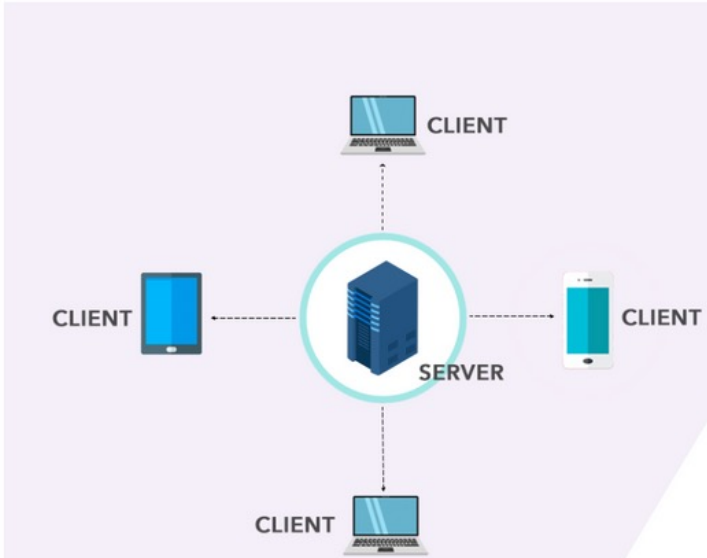




Web2 to Web3 journey



Shift to web3 mentality

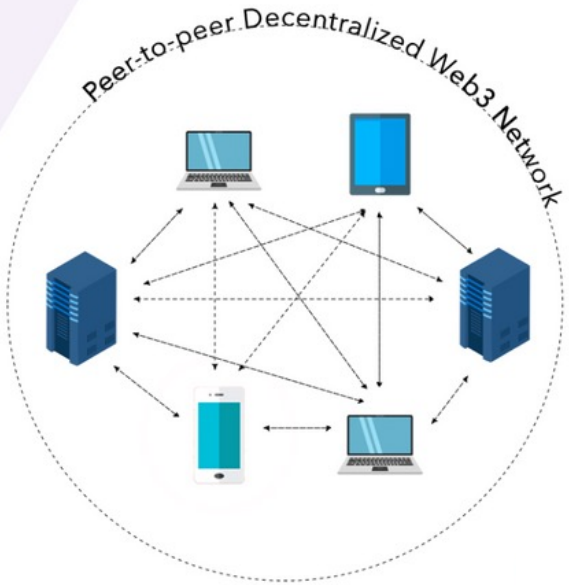


WEB2

Data & apps stored on servers owned and controlled by companies and regulations

WEB3

Data & apps owned by many and stored on servers throughout the Web3 network



Consensus mechanism

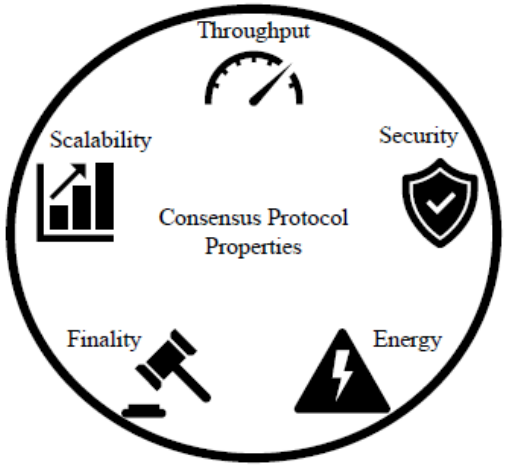
This means that every transaction requires all relevant parties to be in agreement, **consensus**, for the transaction to be valid.

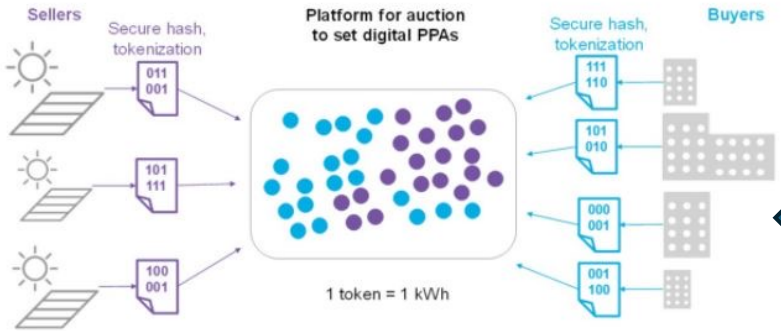
PROOF OF WORK

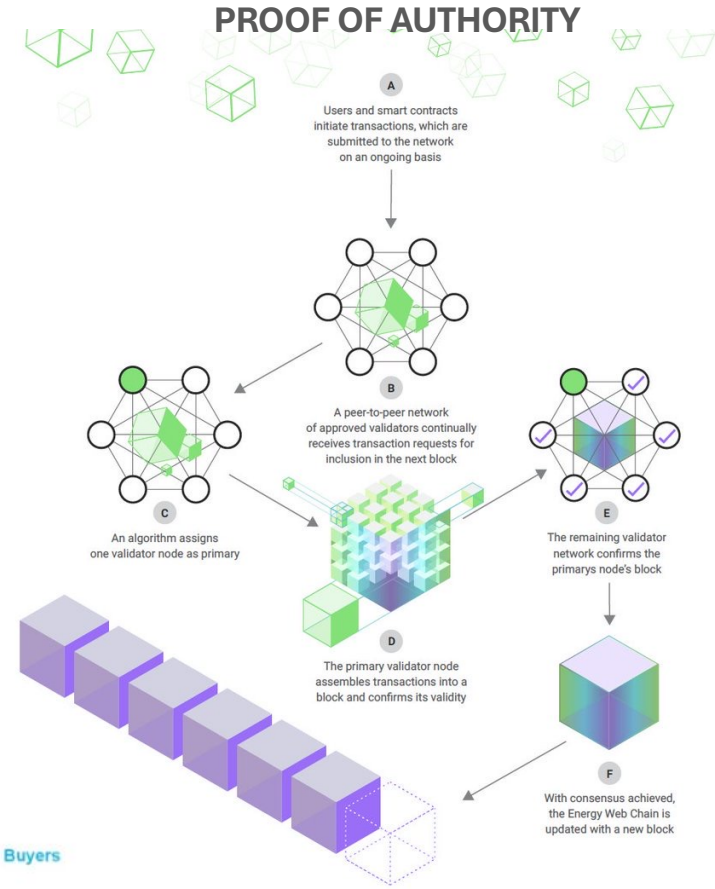
- Block reward given to **first** miner
- More **computing** power = more mining power
- **Hight** energy cost
- Miners pool and mining **bacomes centralized**
- Must provide **proof** to solve block
- Miner receivers block **reward**

PROOF OF STAKE

- Chance of solving block **proportionate** to staked wealth
- More **wealth** = more mining power
- **Low** energy cost
- Mining is **decentralized**
- Must **stake** wealth to solve block
- Validator receives block **transaction fees**





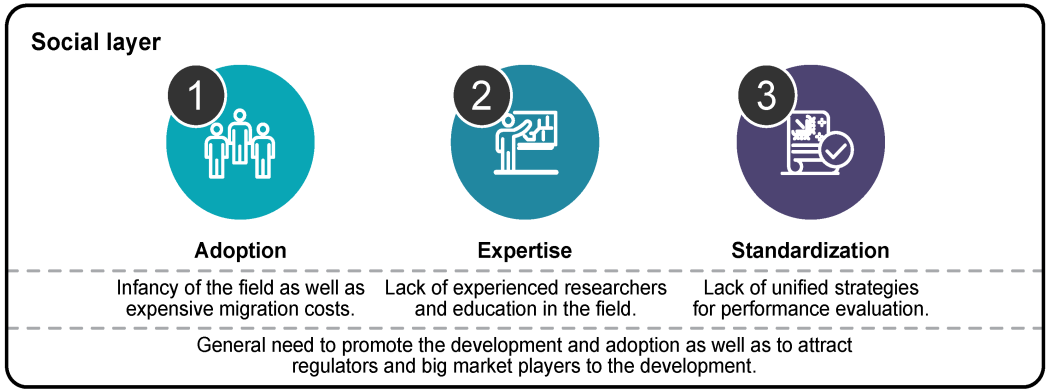


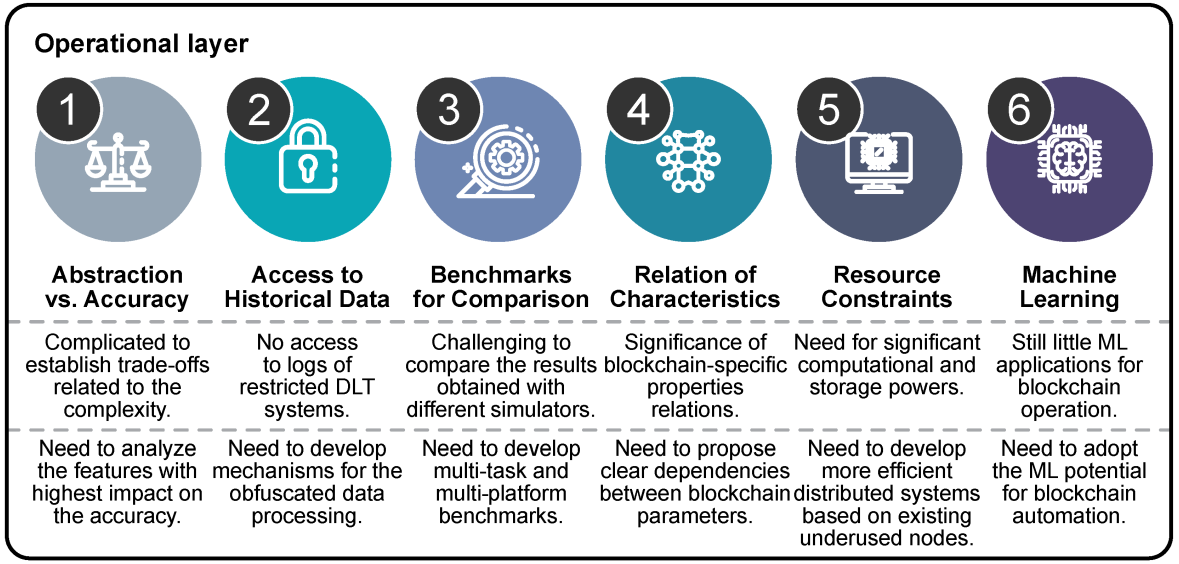
- ❑ Based on identity nodes
- ❑ Private blockchain
- ❑ No reward
- ❑ High Performance
- ❑ Eliminates the possibility of an attack

Source: BloombergNEF, WePower.

Operational and Social Challenges

- **Understanding** the Technology
- **Lack of blockchain tools**
- **Lack of best practices**
- **Defining the test strategy** requires a deep understanding of the application
- **Addition of blocks.** Validate all the blocks that get added to the chain post authentication of every transaction
- **Block and Chain size**
  - A block contains real-time ledger record with encryption and timestamp
  - The chain sizer can be many blocks as the chain lengthens
- **Transmission of Crypto-Graphical data**





An exploit generated 184 Billion Bitcoin

Bitcoin Value Overflow

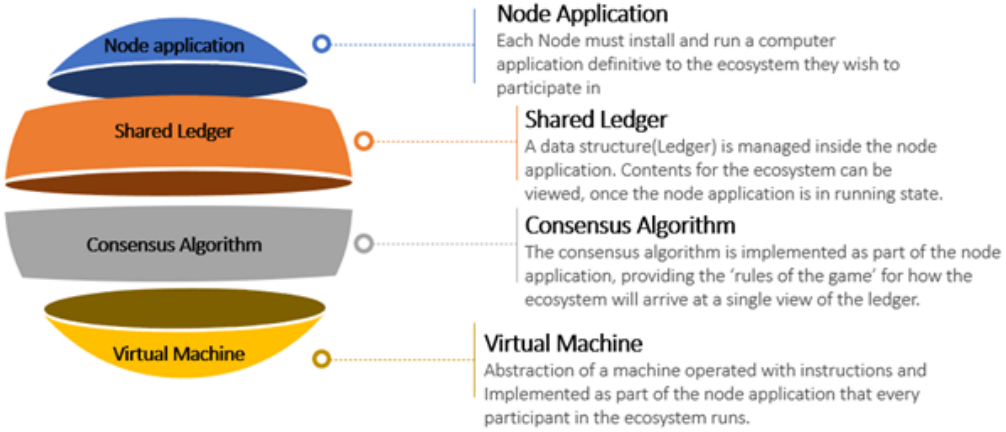
SushiSwap

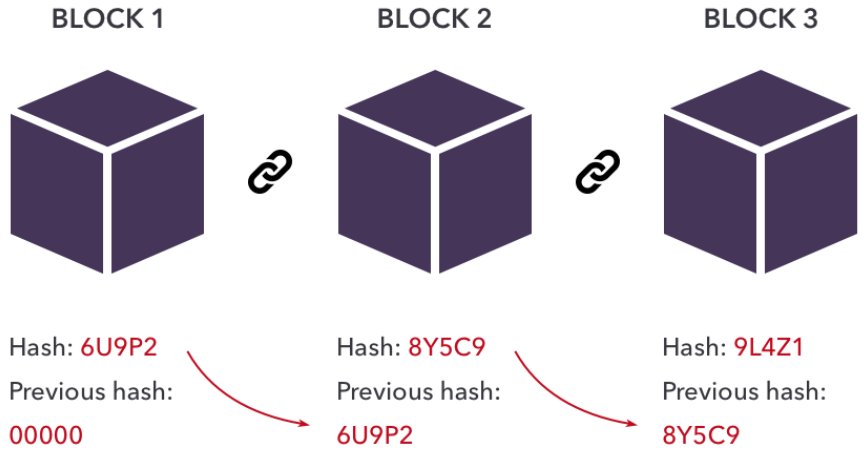
1 user lost 1,800 ETH ~3.3 Million $

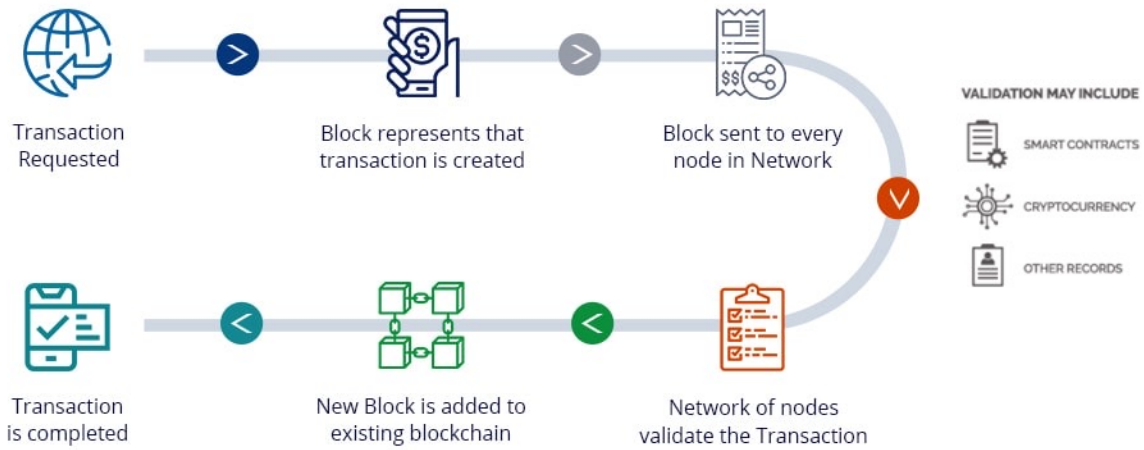
RouteProcess02 contract **exploited** and then distributed across various blockchain networks

Blockchain Testing Scope

As a result of its new method to process transactions securely, blockchain technology has the potential to increase the **reliability, efficiency, and effectiveness** of the processes.



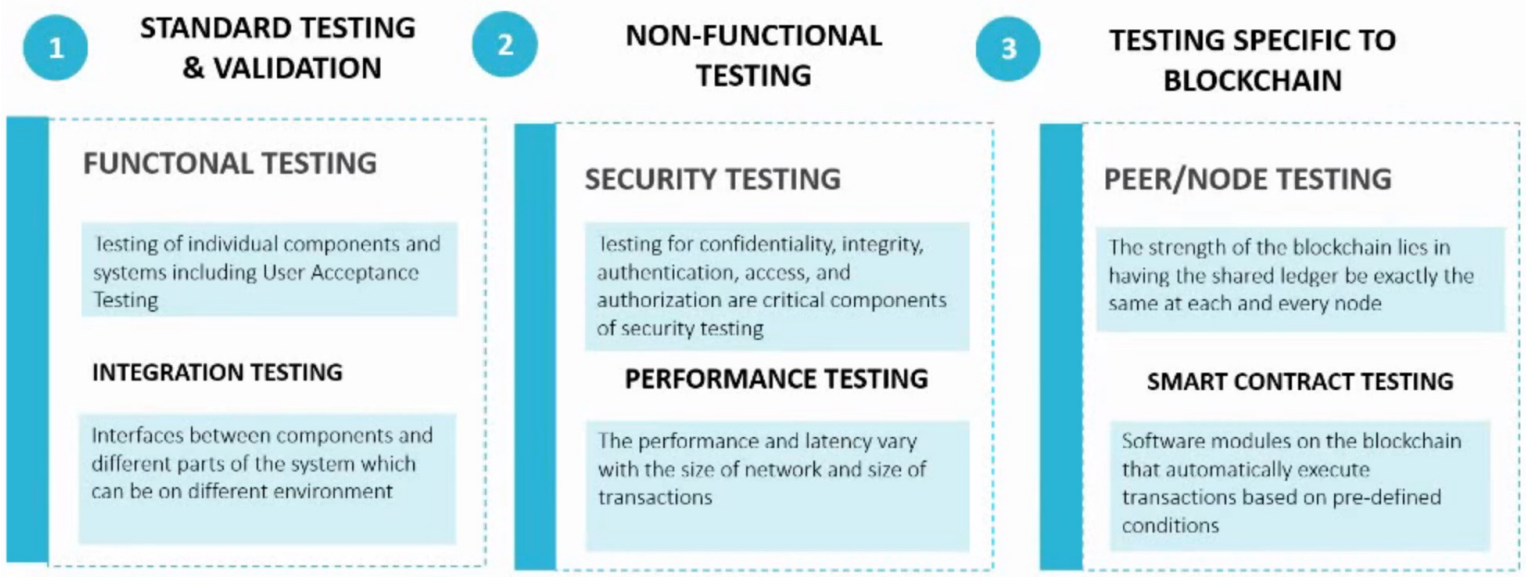


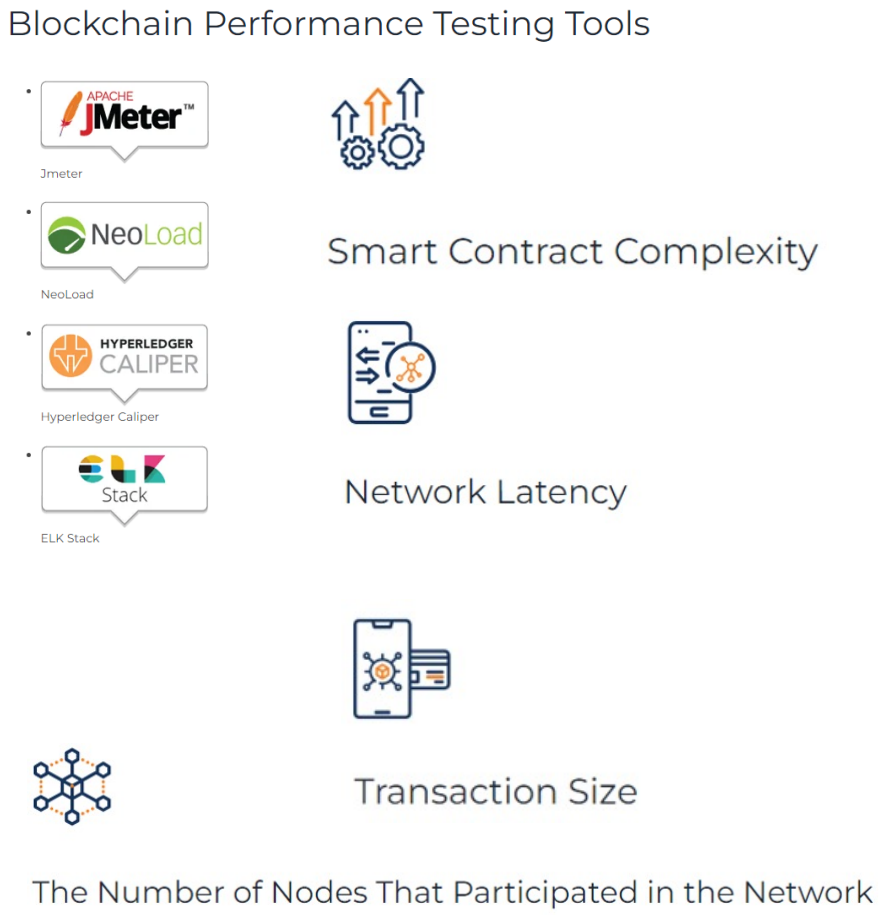


Blockchain Testing Scope

Blockchain technology has revolutionized various industries with its promise of decentralized, secure, and transparent systems. To ensure the reliability and performance of blockchain applications, thorough **testing is crucial**.

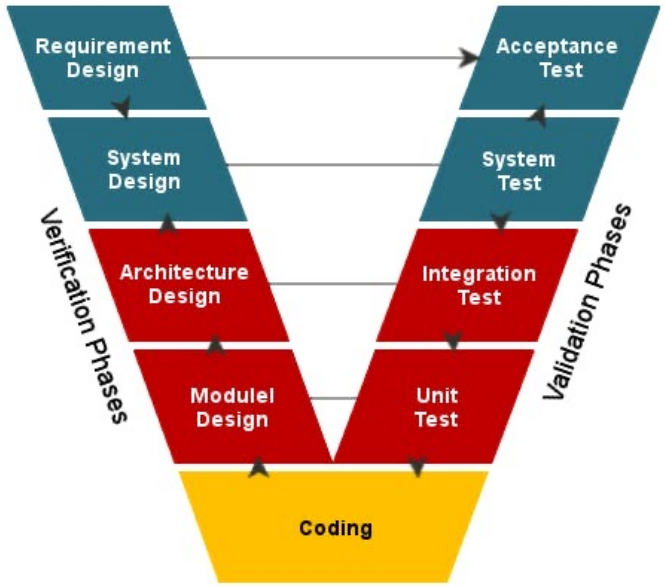
Understanding and implementing these testing types is essential for the successful deployment and maintenance of robust blockchain solutions.

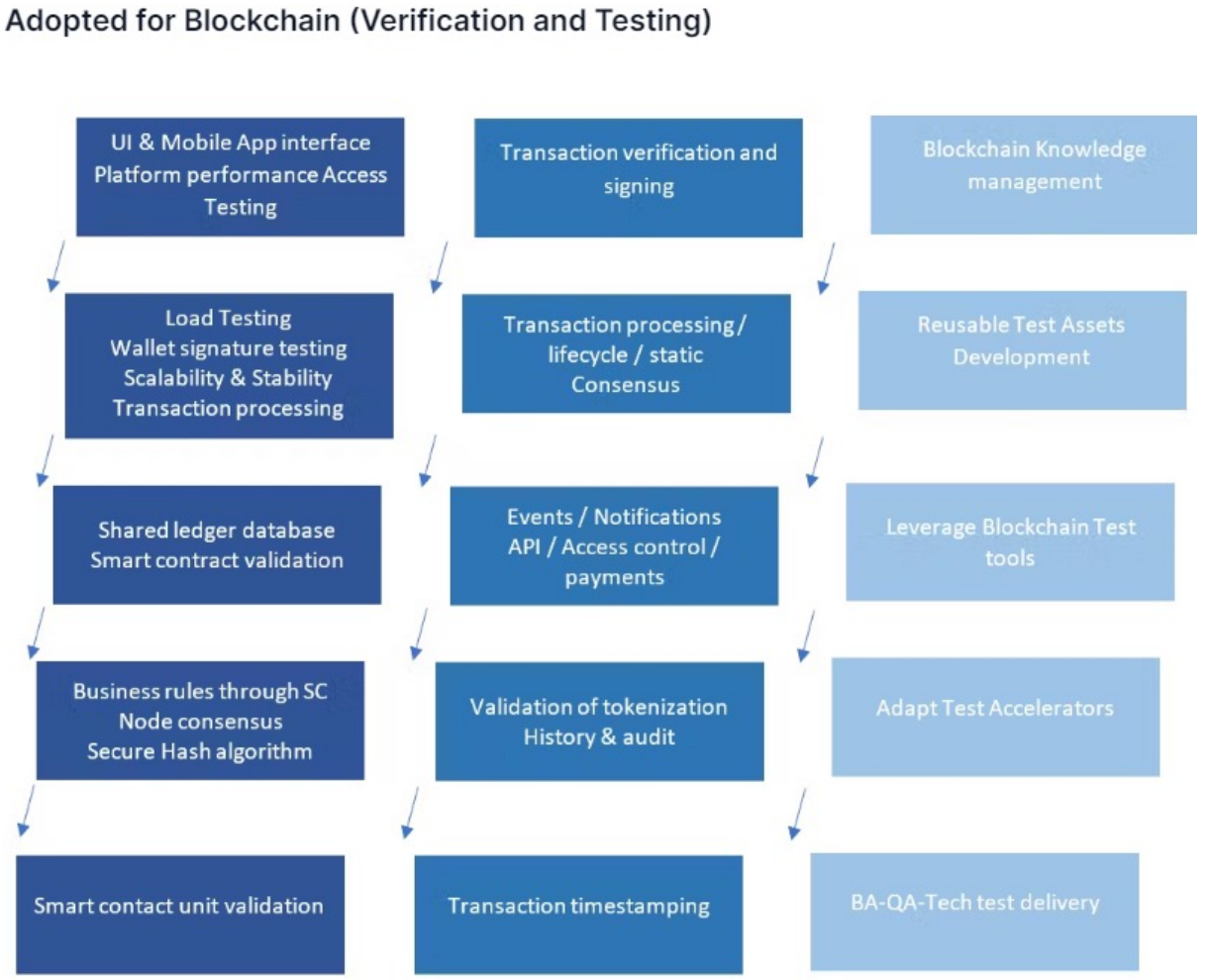




Blockchain Testing Scope

The process of Verification and Testing needs to changed and we can migrate from a **simple V-Model to an adapted version** considering the most relevant features of a Blockchain system.





Blockchain Testing Scope

SPECIFIC IN THE BLOCKCHAIN SOLUTION TESTING TO BE CONSIDERED:

BLOCKCHAIN TRANSACTIONS

TRANSACTION PROCESSING

Validation of transaction lifecycle through Apps / Vaults / Wallets

TRANSACTION STATE/EVENT

Proof of distribution
-Basic data model for blockchain
-Validation of event notifications

VAULT/ WALLET

- Validation of Transactions message verification (signing)
- Common trusted database
- Storage algorithm
- Extensive testing of wallets or hashing algorithms in enterprise blockchain would be more required on 100% open source/ permissionless solutions

BLOCKCHAIN GOVERNANCE

HISTORY & AUDIT

-Validation of secure registry and audit of transaction
- Immutable record
- Secure Identification of assets

CONSENSUS MECHANISM

- Consolidated consistent dataset
- Various consensus models (PoW, PoS, Byzantine FT)
- Transaction timestamping

SMART CONTRACT

Validation of business rules through Smart Contracts:
1. Self enforcing contracts
2. Digitized analog contracts
3. Real-time auditing of transactions

BLOCKCHAIN PLATFORM

INTERACTION WITH APIS

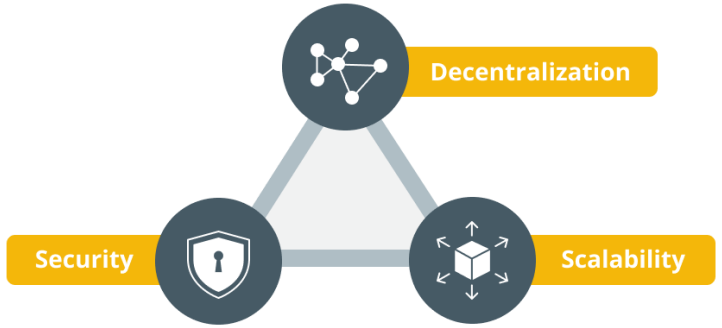
Validation of interface with API's for access control payments, track and trace and balances.

KEY & IDENTIFY SERVICES

- Consortium or private blockchains
- Public blockchains
- Validation of tokenization of assets

BLOCKCHAIN APPLICATIONS

Web Applications
Native applications
Wallet/Vault Applications



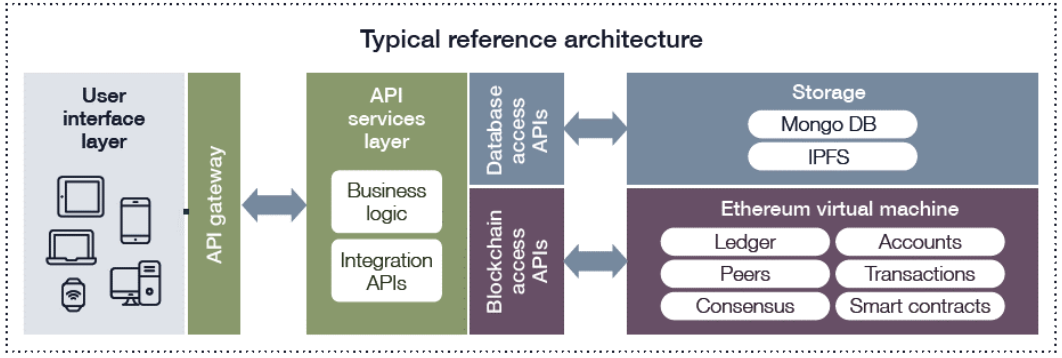
TEST AUTOMATION PLATFORM

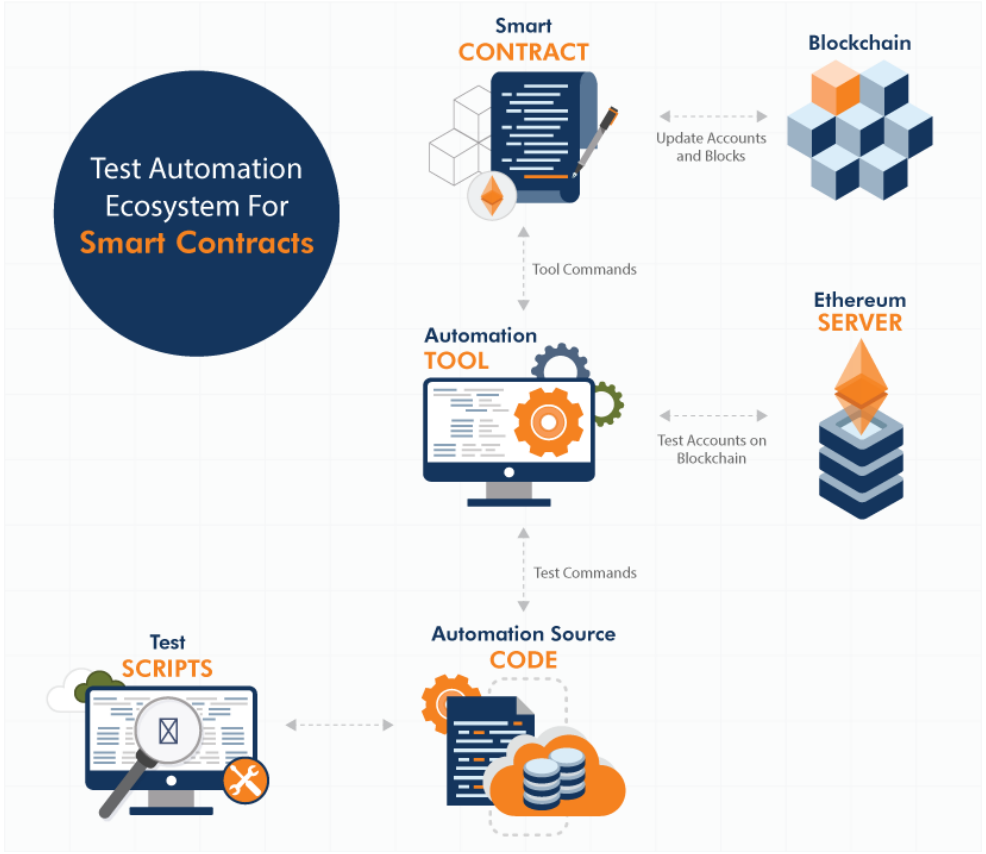
AI-DRIVEN SCENARIO EXTRACTION

INDUSTRY SOLUTIONS

VISUALISATIONS

Architecture and Automation Focus





Programming language

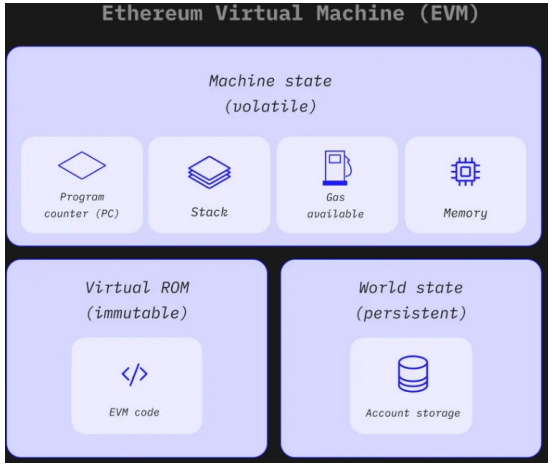
SOLIDITY

Bitcoin testing tools

bitcoinj

Ethereum Testing Tools

docker



TRUFFLE

Ganache

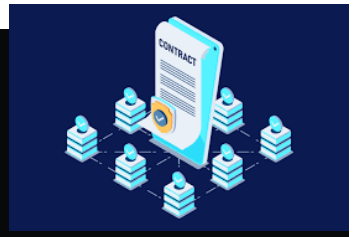
Brownie

Hardhat

(Embedded Local Node available)

Test your smart contract with Truffle

Create Smart contract




```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Fundraiser {
    address public owner;
    address[] public backers;

    constructor() {
        owner = msg.sender;
    }

    function sendMoney() public payable {
        require(msg.value > 0, "No Ether were sent.");
        backers.push(msg.sender);
    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }

    function endFundraising() public {
        require(msg.sender == owner, "Only the owner is allowed to end the fundraising.");
        payable(owner).transfer(address(this).balance);
    }
}
```

Creating a test file

```
const Fundraiser = artifacts.require("./Fundraiser.sol");

contract("Fundraiser", (accounts) => {
    it("some test", async () => {
        // ...
    });
    it("some other test", async () => {
        // ...
    });
});
```

Interact with contract

While testing a contract you will need to interact with it. The first thing you would want is to access an instance. From the artifact, you can call the .deployed().

```
const contract = await Fundraiser.deployed();
```

To access a public variable from your contract, we will use the call() function.

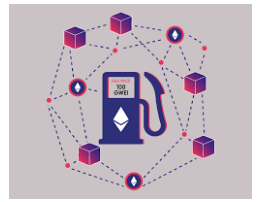
```
const owner = await contract.owner.call();
```

Finally, to call a public function just call the function like you would to on any object.

```
let balance = await contract.getBalance();
```

```
await contract.sendMoney({from: '0x0...', value: 1});
```

value is in Gwei (1 Gwei = 0,00000001 Ether), to send Ether



Test your smart contract with Truffle

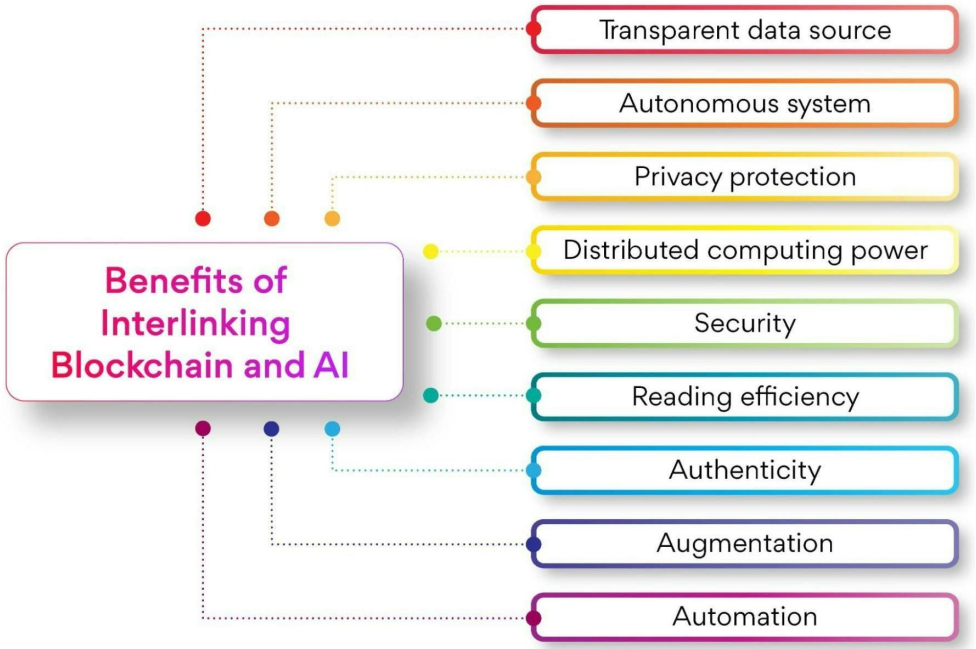
Writing tests

```
it("check the participants array after participation", async () => {
    // Get the first account
    let account = accounts[0];
    // Retreive the deployed contract
    const contract = await Fundraiser.deployed();
    // Make the account send 14 gwei
    await contract.sendMoney({from: account, value: 14});
    // Get the particpant at index 0
    let backer = await contract.backers.call(0);
    // Check if that participant is the same as the addess we used
    assert.equal(backer, account);
});
```

```
it("check the balance increase", async () => {
    // Get the first account
    let account = accounts[0];
    // Retreive the deployed contract
    const contract = await Fundraiser.deployed();
    // Get the balance before the operation
    let balanceBefore = await contract.getBalance();
    // Send 1 gwei
    await contract.sendMoney({from: account, value: 1});
    // Get the balance after the gwei was sent
    let balanceAfter = await contract.getBalance();
    // Check if the balance after is equal to the balance before plus 1
    assert.equal(balanceAfter.toString(), parseInt(balanceBefore.toString())+1);
});
```

```
it("check the balance after the fundraising ends", async () => {
    // Retreive the deployed contract
    const contract = await Fundraiser.deployed();
    // Get the contract owner by accessing the owner attribute
    const contractOwner = contract.owner.call();
    let contractsBalanceBefore = await contract.getBalance();
    let ownersBalanceBefore = await web3.eth.getBalance(contractOwner);
    // End the fund raising
    await contract.endFundraising({from: contractOwner});
    let contractsBalanceAfter = await contract.getBalance();
    let ownersBalanceAfter = await web3.eth.getBalance(contractOwner);
    // Check if the contract balance is now zero
    assert.equal(contractsBalanceAfter, 0);
    // Check if the owner account received the contract's balance
    assert.equal(ownersBalanceBefore >= ownersBalanceAfter+contractsBalanceBefore, true);
});
```

Blockchain and AI



Artificial Intelligence

AI is a dynamic technology that examines its surroundings and generates solutions based on its past experiences. On the other hand, blockchain is a passive technology that secures blocks cryptographically and is indifferent to the data written into the network.

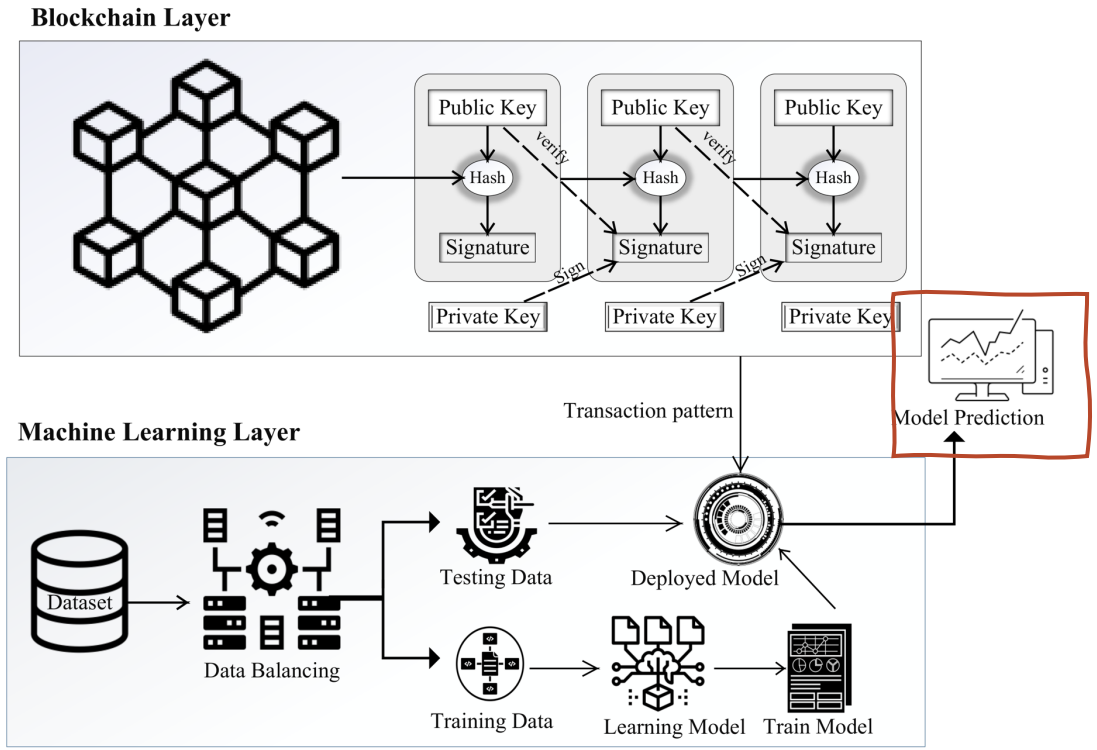
Blockchain

Blockchain is a type of data structure that enables the creation of a secure, decentralized, peer-to-peer system of ledgers. These ledgers consist of blocks of data that are immutable, time-stamped, and cryptographically linked, making them tamper-proof.

AI + Blockchain, Combined

AI refers to a machine's ability to imitate human behavior, particularly in problem-solving, language, and identification. Machine learning, a subfield of AI, is a data analysis technique that allows machines to learn from data, recognize patterns, and draw conclusions without explicit programming.

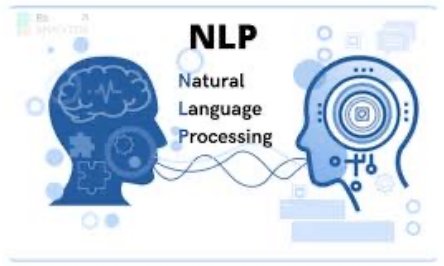
Blockchain and ML



Blockchain-based transactions are verified using a **machine learning model**, and the prediction result shows that the transaction is legitimate or malicious.

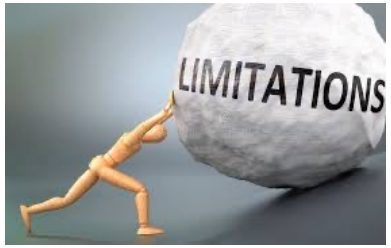
The prediction of the machine learning model is based on the training and testing of a transaction-based dataset.

AI can streamline blockchain operations in various ways.
It can enhance scalability by compressing transaction data, refining consensus mechanism design, and optimizing network resource allocation.



Review Smart contracts

Security



- ❑ Newer blockchain networks **lack historical data**.
- ❑ AI models trained with on one type of protocol
- ❑ **Inability to generate new content or adapt** to novel scenarios **limits the ability to detect new attacks** or **automating smart contract creation**
- ❑ **Scalability**
- ❑ **Privacy**
- ❑ **Interoperability**

Structural Convergence of GenAI and Blockchain

“Generative AI Needs Blockchain to Thrive in 2024”

GenAI has recently emerged as a promising solution to address critical challenges of blockchain technology, including **scalability, security, privacy, and interoperability.**

Decentralized Data Sources

Data is king.
Web3’s decentralized architecture offers a new approach to data collection and storage – **more robust AI models**

Collaborative AI Model

By introducing clear and attractive tokenized incentives for AI development, blockchain can help ensure that the value generated by new AI models is **equitably** distributed among developers and contributors

Crowdsourced Computing Power

AI models, with their extensive computational requirements, can benefit from blockchain’s decentralized infrastructure, leveraging crypto mining resources and other independent GPUs

Effectively democratize access to computational resources, turbocharging AI model training and execution, high-fidelity VR experiences, and other computation-heavy forms of entertainment

Generate Smart Contracts

GenAI techniques, **such as GANs and LLMs (Large Language models**), can be applied for smart contract generation by learning and simulating the patterns and logic found in existing smart contracts.

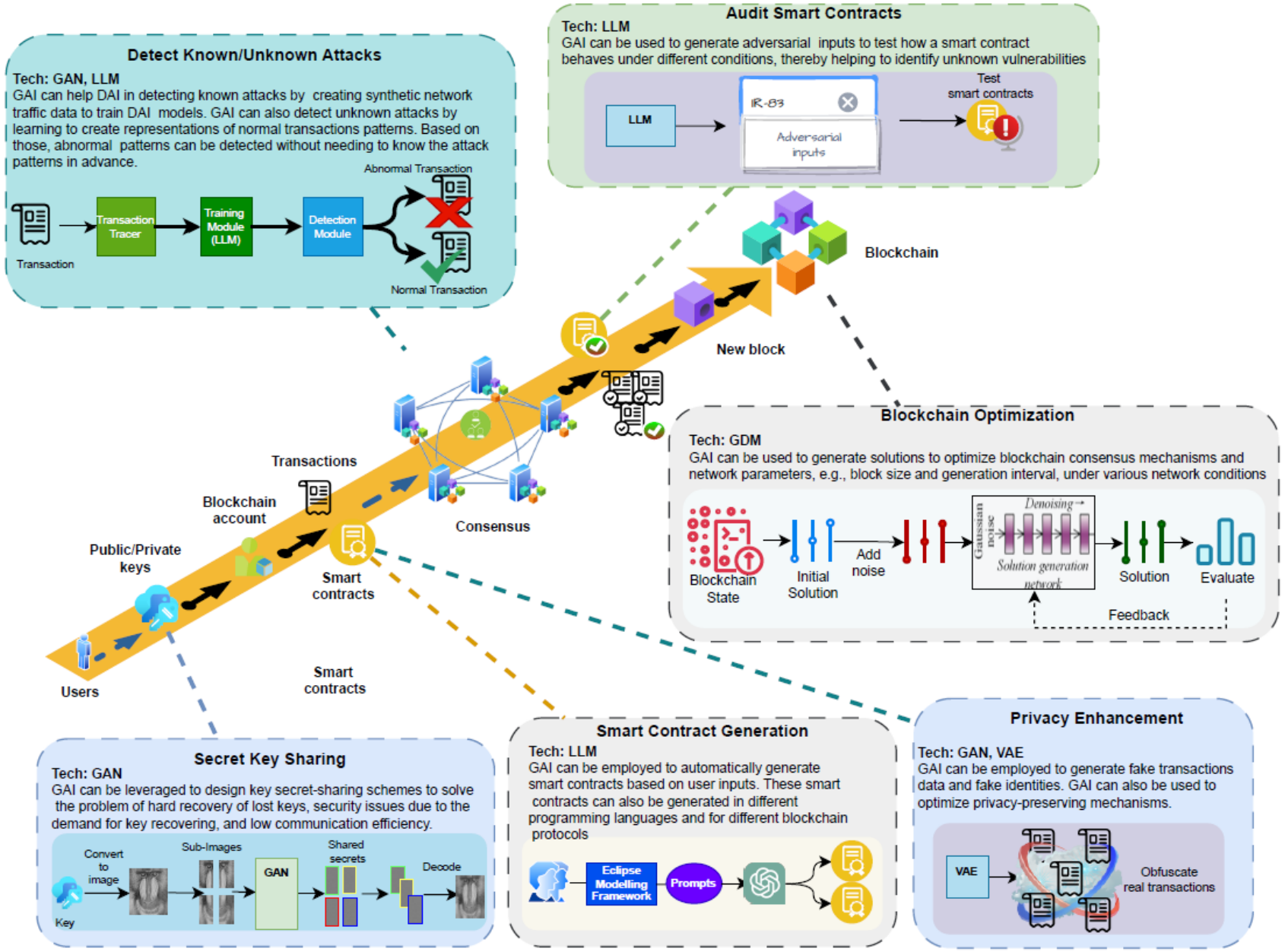
For example, GenAI can **generate code** that adheres to the syntax and semantics of smart contract programming languages such as **Solidity** (used in Ethereum).

Optimize blockchain network designs

GenAI can be applied to optimize blockchain designs by creating synthetic workloads and transaction patterns that mimic real-world usage scenarios.

Based on those, blockchain developers and network administrators can **simulate** different resource allocation strategies and optimize them for high efficiency and performance.

Generative AI-enabled Blockchain Networks



- A user generates a public/private key pair to **join a blockchain network**. GenAI can aid in key generation and sharing processes.
- Once joined, the user can create transactions and smart contracts. GenAI can **automatically generate smart contracts**.
- Transactions and smart contracts are validated by the **consensus mechanism**. GenAI can **audit smart contracts** and detect attacks from transactions. GenAI can also be leveraged to **optimize blockchain network** parameters and consensus mechanisms.
- Once validated, transactions and smart contracts are collected to create a new block to add to the chain. GenAI also can generate **fake transactions to obfuscate real transactions to improve privacy**.

diogoguerramaio

diogo.goncalves@exceltic.com

Thank you!


expoQA 24
MADRID
May 28th, 29th, 30th 2024
Thank you for attending
expoqa.com