Cognizant

Digital Systems & Technology

Continuous Testing Is Key for Enterprises to Adopt AI Platforms

While the introduction of AI-as-a-Service is a paradigm shift that makes AI adoption easier for enterprises, AI assurance is even harder in the SaaS model. Establishing a continuous testing ecosystem helps deploy both rapid upgrades to the AI platform and continuous changes to enterprise IT assets. Here is a practitioner's view of design considerations for establishing such an ecosystem for AI-as-a-Service adoption.

August 2020

Executive Summary

Artificial intelligence (AI) elevates the quality of human lives by solving some of the most important problems that we face today. The adoption of AI in businesses remakes the world of commerce, by making enterprises more creative and competitive. Per the Gartner Hype Cycle,[1] AI is entering a golden age and will attain greater heights over the near and long term.

Another game-changer has been Software-as-a-Service (SaaS), which has transformed software production and consumption models. Built on the principles of cloud computing, the SaaS market is valued at $60.3 billion today and is expected to grow at 9% CAGR from 2019 through 2023.[2] SaaS models have enabled companies to focus increasingly on their core businesses, leaving software development challenges to vendors.

According to Gartner's top AI trends in 2019,[3] most organizations' preference for acquiring AI capabilities is shifting toward having them infused in off-the-shelf enterprise applications. Hence, enterprise application/platform providers such as Salesforce, Microsoft, Amazon, Google, SAP, etc. are embedding AI technologies within their offerings as well as introducing AI platform capabilities and embracing the AI-as-a-Service model. This evolution has made AI adoption easier for enterprises.[4]

Quality assurance (QA) in general (and testing in particular) plays a vital role in AI platform adoption. AI platform testing is complex for the following reasons:

- Testing AI demands intelligent processes, virtualized cloud resources, specialized skills and AI-enabled tools.
- As AI platform vendors typically strive for rapid innovation and automatic updates of their products, the pace of enterprise testing to accept product updates should be equally fast.
- AI platform products usually lack transparency and interpretability, and so they aren't easily explainable.[5] Hence, it is difficult to trust testing.

Figure 1 summarizes the vectors of change that add complexity to testing in an AI platform adoption.

Vectors of change that add complexity to testing in AI-as-a-Service model

Market-led	Technology-led	Domain-led	Human-led
❙ Enterprise modernization. ❙ Deeper verticalization. ❙ Increased digitalization. ❙ Cost optimization.	❙ Rapid releases. ❙ Unexplainable AI. ❙ AI to test AI. ❙ Data complexity.	❙ Regulations. ❙ Internationalization. ❙ Localization. ❙ Hyper-customizations.	❙ Skill gap. ❙ Generational culture. ❙ Diversity. ❙ Access. ❙ Trust.

Figure 1

Modern QA shifts the role of testing from defect detection to prevention. Moreover, the quality of AI is very much dependent on the quality of the training models and the data used for training. Therefore, unlike conventional SaaS testing models that only focus on cloud resources, logic, interfaces and user configurations, AI testing should additionally cover areas such as training, learning, reasoning, perceptions, manipulations, etc., depending on the AI solution context. This white paper presents a perspective view of a testing framework for enterprise AI platform adoption, based on learnings attained through real-world implementations.

AI test design considerations in a SaaS model

Modularity helps solve complex problems, which includes AI testing. In an AI-as-a-Service model, the AI algorithm is provided by a platform vendor and then IT enterprises configure it by developing interfaces and providing data for training to enhance end-customer trust of AI-based intelligent applications. Therefore, AI testing should address the basic components of data, algorithm, integration and user experiences.

Secondly, testing should validate the functional fitment of the solution within enterprise IT. It should certify the configuration of an AI platform product in the business ecosystem, such that the business purpose of AI adoption is met. It should also verify the training model used to raise the solution in an organizational construct.

Thirdly, the approach that the AI algorithm adapts – such as statistical methods, soft computing, etc. — must be addressed in the algorithm validation process. Though the solution is a black box, necessary coverage should be established to certify its validity.

Finally, the tools that AI logic applies – such as search, optimization, probability, etc. — should be covered in the functional validation process.

Figure 2 summarizes the various test elements that the AI testing framework should address.

Typical AI test elements in AI-as-a-Service model

AI Components	AI Problems	AI Approaches	Tools Used in AI
Data. Algorithm. Integration. Experience.	Knowledge. Reasoning. Planning. Learning. NLP. Perception. Manipulation.	Statistical methods. Computational intelligence. Soft computing. Symbolic AI.	Search. Mathematical optimization. Probability. Economics models.

Figure 2

AI test attributes for design considerations

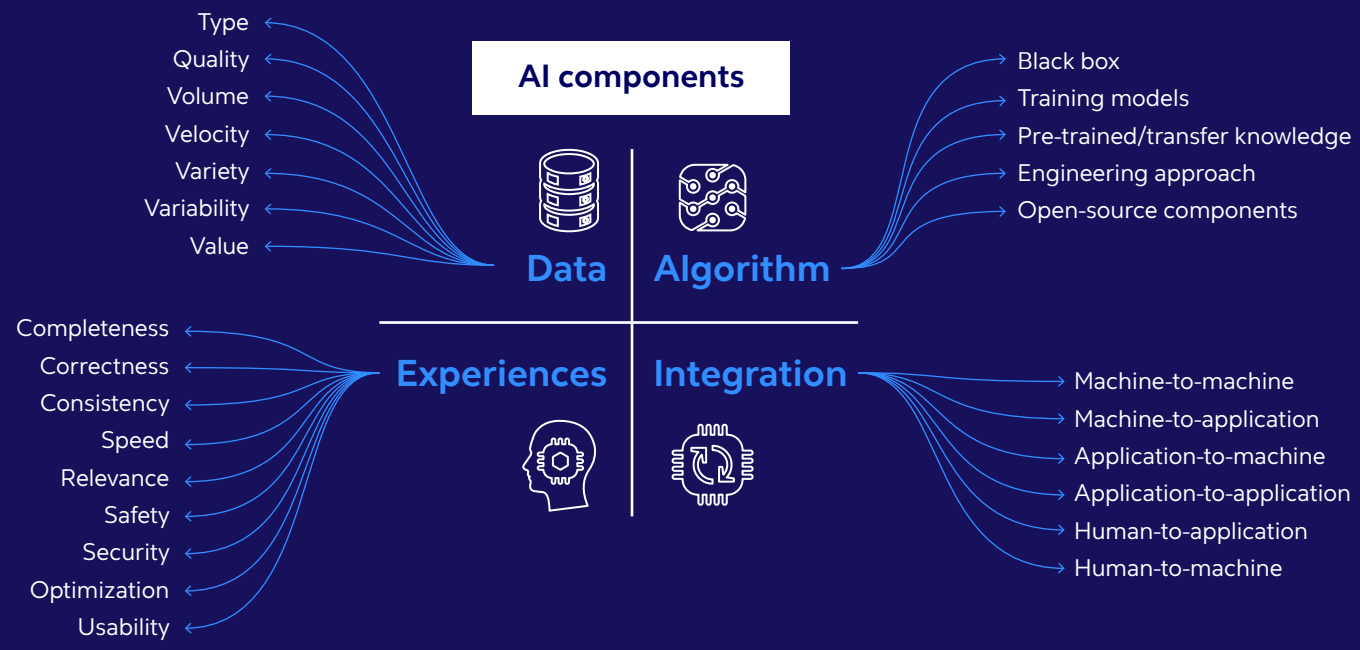


Figure 3

It is critical to apply the nuances of AI to each test element. For example, data assurance strategy should address the complexities that the AI solution would introduce in terms of type, quality, volume, velocity, variability, variety and value of data. Figure 3 presents a practical list of test attributes for test design considerations.

We found that QA maturity is critically important for an organization to choose an AI platform solution. The key to success is a high degree of test automation, without which enterprises cannot manage frequent releases of the AI platform (and the products within) as well as ongoing internal application releases.

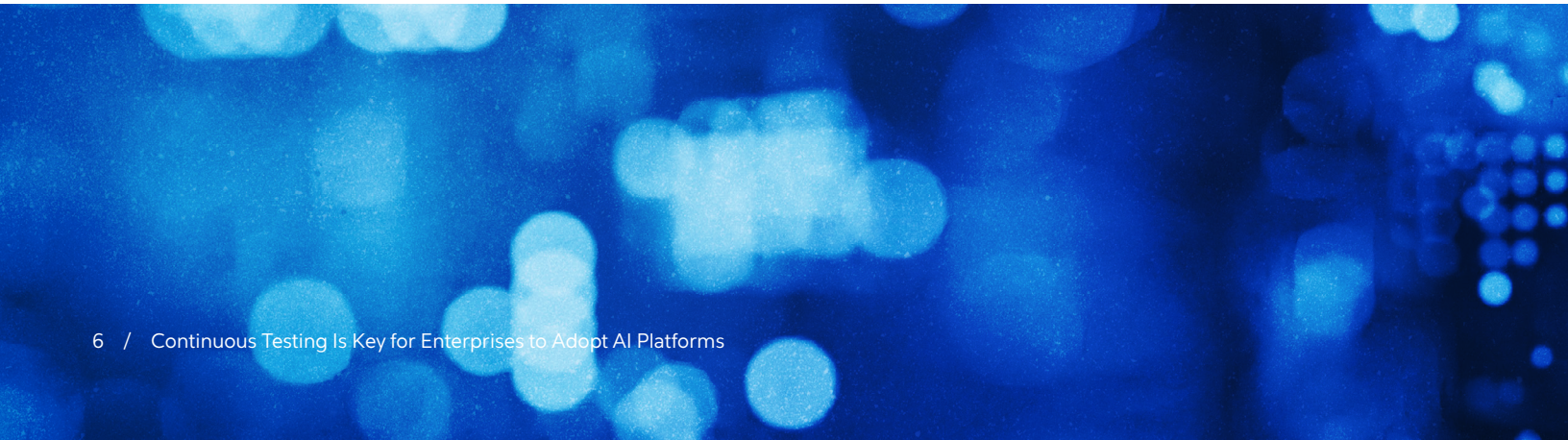
Importance of continuous testing in AI platform adoption

To keep pace with the evolution of AI and SaaS delivery, enterprises must modernize their data and interface infrastructures. Engineering methodologies behind enterprise IT application interface development and data administration practices should support the rapid release cycles that the AI vendor would follow. To make it happen, continuous integration (CI) techniques have proven effective.[6]

While automation is the basis of CI, test automation in a CI construct is fraught with challenges. Each code integration is verified by an automated build, followed by an automated "smoke test," allowing teams to detect problems early in the lifecycle. It is important to understand and overcome these challenges and find a way to automate tests by applying CI techniques to enable seamless and continuous delivery (CD).[7] In summary, CD is driven by CI.

In CI, code is logged-in several times a day and then recompiled, generating multiple QA feedback loops. Thus, development teams must collaborate and make frequent deployments, which mandates greater levels of process automation. Also, to successfully apply CI, automation of the builds and deployment process are critical; this eventually ensures self-testing builds.

We observed that evolution of the Agile and DevOps models has accelerated the feedback loop between development and testing, institutionalizing continuous testing (CT) and continuous development that resulted in CD.[8] In this way, DevOps enterprises are always ready for the faster release cycles inside and outside their four walls.



Challenges of automating QA in continuous delivery

In our efforts to help customers implement SaaS-based AI platforms, our teams have observed and solved a number of challenges. Since the release of AI platform products are more frequent, the stability of the code behind each application interface and data extraction decreases. Additionally, when distributed teams work together, such as in an Agile model, functionalities and AI interfaces change across multiple Sprints. Hence, automating QA is far from an easy task. Key AI adoption challenges that our testing teams have overcome include:

- **Ensuring test coverage:** It is absolutely critical to sync AI product release functionalities with enterprise application functionalities. As the code for interfaces and data extraction are integrated continuously, chances are that critical tests for a particular requirement could be missed. Also, unanticipated code changes could lead to limited test coverage during test automation.
- **Costlier defect fixes:** When test coverage is not complete, defects that belong to an earlier Sprint or product release are detected much later in the development/deployment cycle. Once the data and application interfaces are in production, fixing these defects becomes significantly more expensive.
- **Compromised user performance:** As AI product releases are frequent and software data features are added incrementally, there is a risk that user experiences of the AI solution will be suboptimal. Automated experience assurance at build levels is complex and time-consuming.
- **Multitier model of AI solutions:** In a multilayer model, performance is distributed across layers, and QA must be orchestrated across the ecosystem.
- **AI solution to unlearn to accept the fixes:** The AI solution may have learned to live with the errors/bugs of prior releases. Unlearning them and retraining the algorithms to accept the fixes is a complex activity. To do this, test results and training models should be aligned.
- **Leading with trust:** As AI is making systems autonomous, we as humans fundamentally need assurance so that we can trust these systems. Since AI products are already lacking transparency, building automation on top of it will only add more complexity.
- **Tough to test:** Validating ethics, bias, etc. is difficult to achieve through automated testing.

Continuous testing ecosystem for AI platform adoption

Enterprise IT assets such as data, applications, infrastructure, etc. are constantly changing. At the same time, AI products are continuously upgraded by the vendor in order to improve experiences and efficiencies. Given this dynamism, it is crucial to establish a continuous testing ecosystem that not only automatically confirms the ever-changing enterprise IT landscape, but also validates the changing versions of the AI product within the context of the enterprise adoption.

To establish a CT ecosystem, we executed the following high-level steps in our client engagements:

- **Shifting automation test scripts to an enterprise version control tool** to establish a single source of truth. Instead of storing automation scripts in a test management tool or a shared folder structure, they are checked into a version-control repository.
- **Integrating the automation suite with a code/data build deployment tool** to enable centralized execution and reporting. Test teams align code/data builds with their respective automation test suite. Tool-based auto-deployment during every build was orchestrated to avoid human intervention.
- **Classifying the automation suite in multiple layers of tests** to speed up feedback at each checkpoint. Tests for each data and code build, such as health check and smoke test, were performed to verify if the key system features and individual services are operational and no blocking defects occur.
- **Optimizing testing at code and data level** to improve cost and time-to-market. Conventional requirements-based testing tends to result in considerable over-testing without the guarantee of complete coverage. The impact of missed tests on coverage and quality is significant in CI. This typically results in additional QA costs and extended testing timelines that, in turn, impact project success. There are solutions that perform impact analysis to map the code and data changes to affected test cases. This ensures an optimized testing while maximizing test coverage.[9]
- **Test the training model.** Traditional testing methods only help validate engineering philosophies, not the algorithmic approach of the AI solution. By testing the training model, we could certify if the solution has learned the given instructions — enforced, supervised or unsupervised. It was critical to recreate the same scenarios multiple times to check for correctness and consistency. Similarly, it was also critical to establish a process as part of testing, to train the AI solution to learn from bugs, errors, exceptions and mistakes, etc. Fault/error tolerances were established based on the customer-defined exception handling.

AI platform adoption: Continuous testing ecosystem

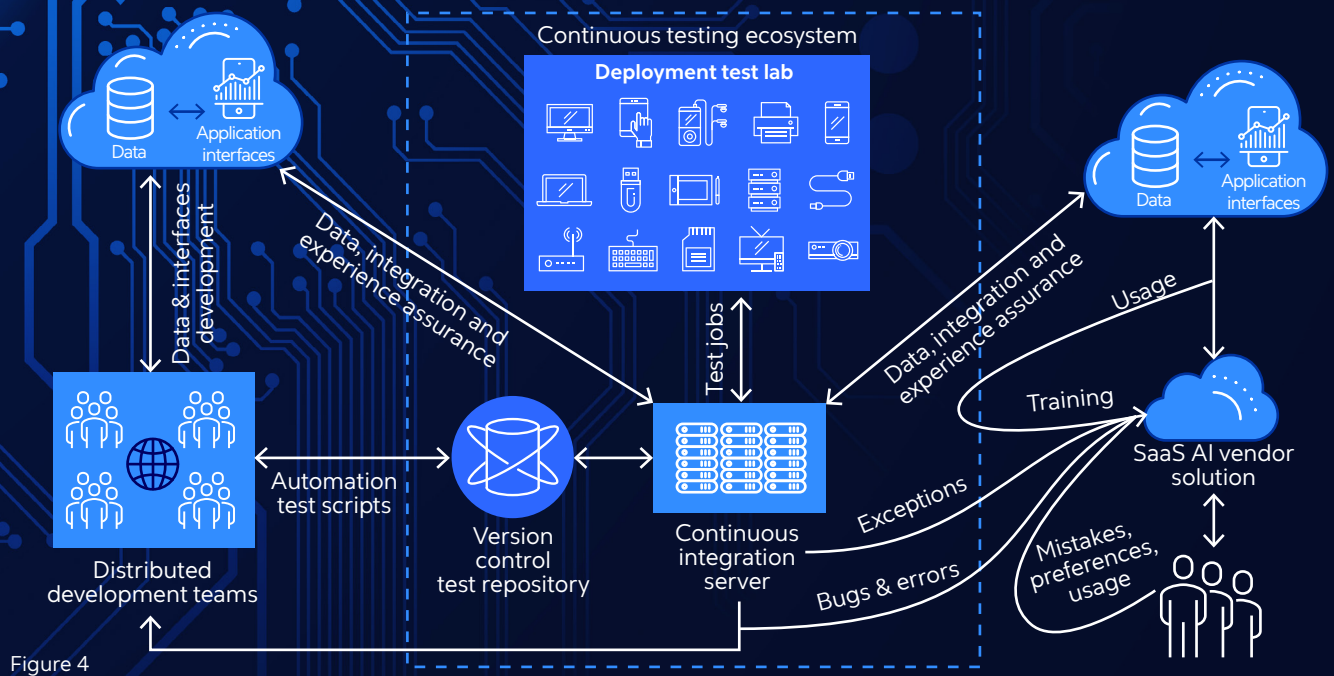


Figure 4

- **Apply a transfer learning model.** AI techniques have challenges carrying their experiences from one set of circumstances to another, leading to more and more testing/training in real-world production data. CT setup helps continue the learning from testing through production rollout with less worry about transfer learning.
- **Embrace intelligent regression.** If execution time for overall regression is high, CT setup becomes less effective due to prolonged feedback cycles. To avoid this, a subset of regression was carved out at run-time based on critically impacted areas. The team applied machine learning (ML) to achieve smart regression. Effective ML algorithms that use a probabilistic model for selecting regression tests[10] help optimize the use of cloud resources efficiently and speed testing.
- **Utilize full regression.** This is done overnight or during the weekend, depending on the alignment with recurring build frequencies. This represents the final feedback from the CT ecosystem. The goal is to minimize feedback time by running parallel execution threads or machines.

Figure 4 presents our continuous testing ecosystem approach for AI platform adoption. When there was no manual intervention for testing, bugs, errors, mistakes and any algorithmic exceptions, all became sources of discoveries for the AI solution. Similarly, the actual usage and user preferences also became the source of training that continued through production.

Data extraction: Test design considerations

Successful modern enterprises are data-driven. Technological progression and the proliferation of devices are spewing an avalanche of data. Every commercial technology, AI especially, has been built on the promise that it empowers leaders with the right data at the right time for better decisions. Quality of data is the most important success criterion for AI adoption.

To this end, useful data resides outside the enterprise as much as inside. The ability to extract every useful bit and byte and make it available to the AI engine is therefore a must. Extract, transform and load (ETL) is a heritage term that refers to a data pipeline that collects data from various sources, transforms the data according to business rules and loads it into a destination data store.

The ETL field has advanced to enterprise information integration (EII), enterprise application integration (EAI) and enterprise cloud integration platforms as a service (iPaaS). Irrespective of the technological advancements that provide interoperability among the multiple disparate systems that make up a typical enterprise architecture, the need for data assurance has only grown in importance.

Our teams used an AI data assurance strategy to address key functional testing activities such as map reduce process validation, transformation logic validation, data validation and data storage validation. We also focused on nonfunctional aspects of performance, fail-over and security. Structured data was easier to administer whereas unstructured data that originated outside the enterprise IT is a major headache. Stream processing principles help prepare data in motion – i.e., processing data as soon as it is produced or received from websites, external applications, mobile devices, sensors and other sources through event-driven processing. Checking the quality through established quality gates is an absolute necessity.

Messaging platforms such as Twitter, Instagram, WhatsApp, etc. are popular sources of data. When used, they connect applications, services and devices across various technologies via a cloud-based messaging framework. Most companies we work with need to process data from these platforms. We apply deep learning technologies to the data loads to extend the insights and foresights contained in their systems of record. Some of this data requires neural network solutions to solve complex signal processing and pattern recognition problems including speech-to-text transcription,[11] handwriting recognition[12] and facial recognition.[13] Necessary quality gates are established to test data that spans these platforms. Figure 5 presents the overall framework that we follow in the data test design for AI platform adoption.

AI/ML-driven data testing: Quality gates

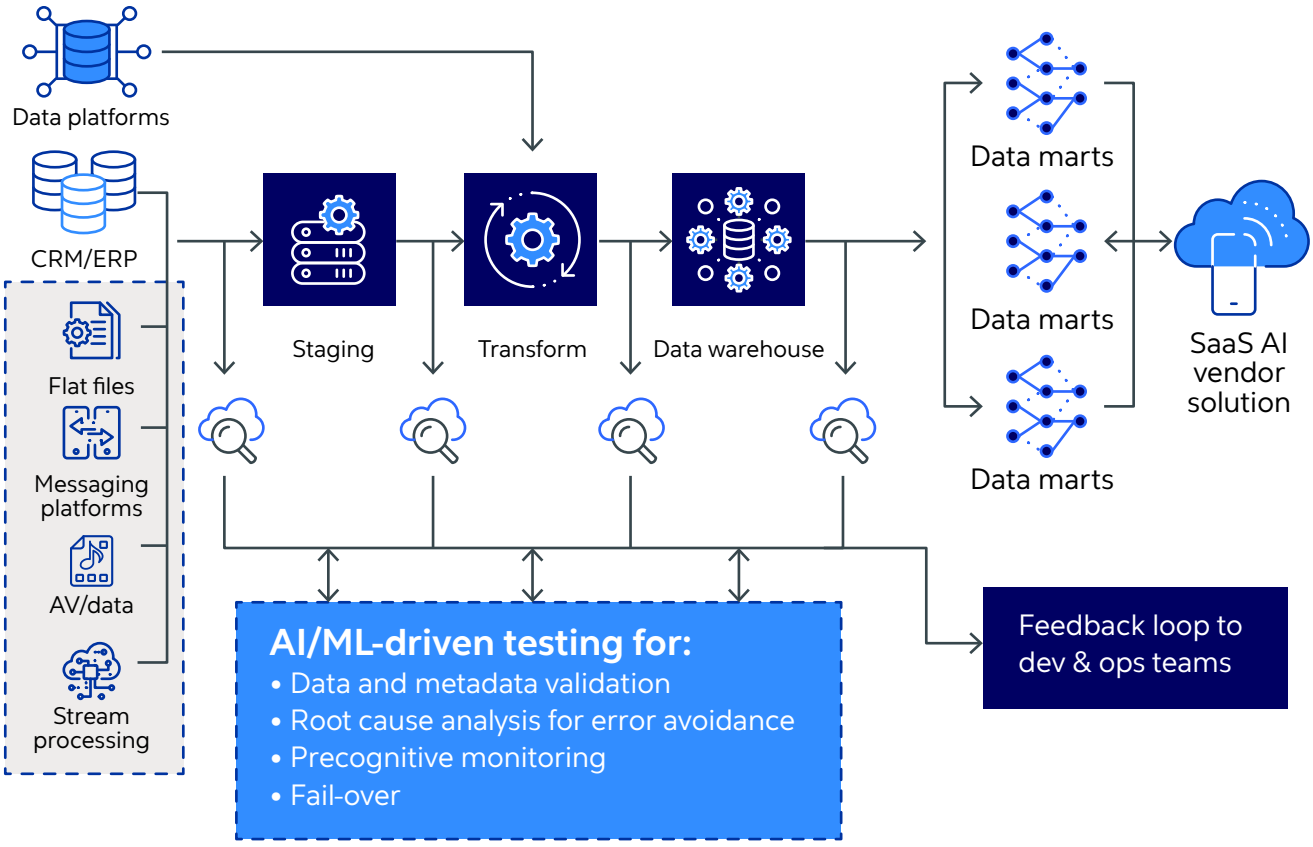


Figure 5

AI-driven design elements that can also be applied to the testing process include:

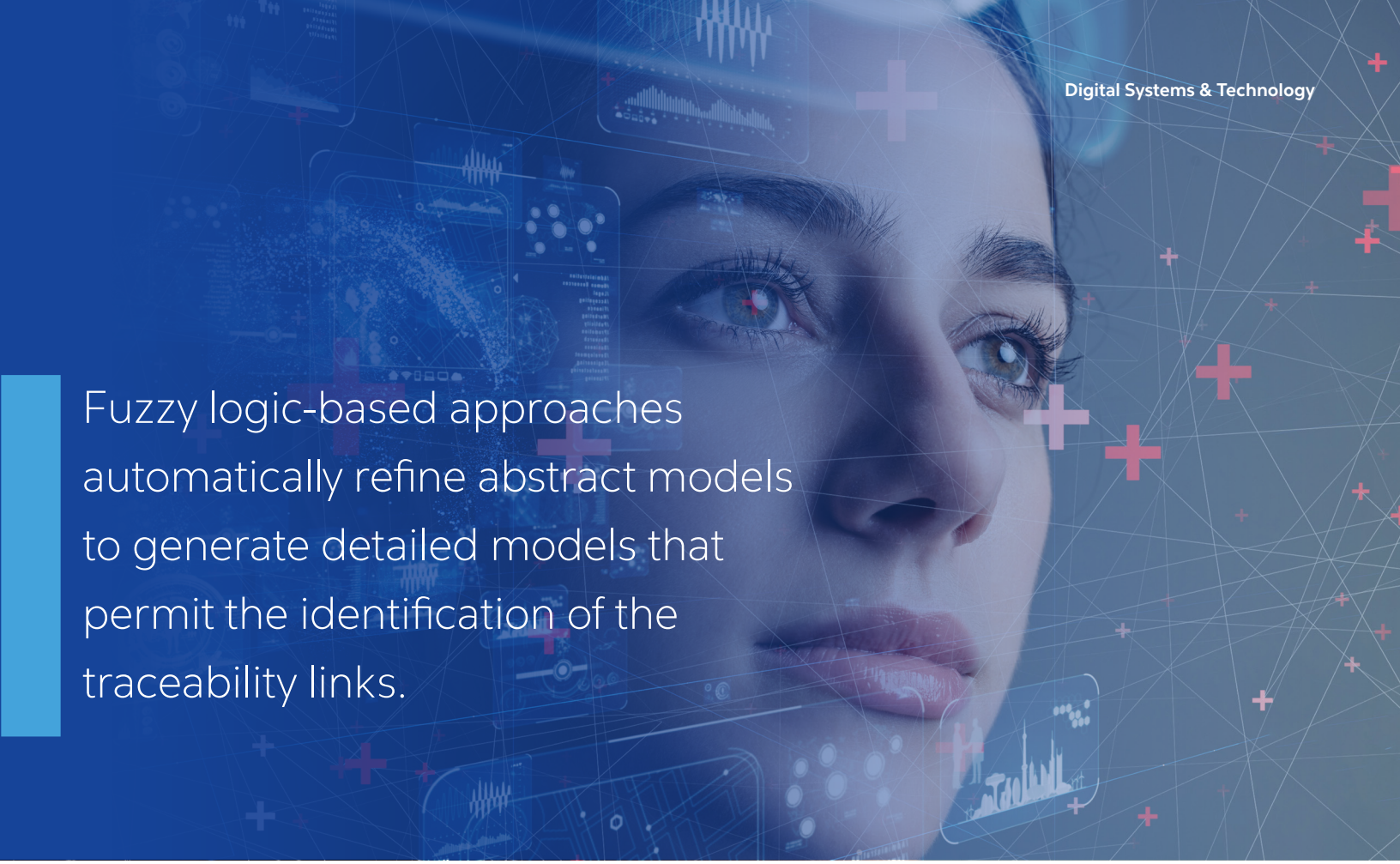
- **Quality gate automation:** ML algorithms can be implemented to determine if the data is a "go" or "no go" based on historical and perceived standards, avoiding inferior customer experiences.
- **Root cause prediction:** Triaging the root cause of a data defect helps continuously improve the data quality. With patterns and correlations, testing teams can implement ML algorithms that trace defects to the root causes.[14] This helps auto-perform remedial tests and fixes before the data progresses to the next stage, leading to self-testing and self-healing.
- **Leveraging precognitive monitoring:** ML algorithms can scout for symptoms in data patterns and associated coding errors and implement corrective steps automatically.
- **Fail-over:** ML algorithms can detect failures and automatically recover to proceed with processing, registering the failure for learning.

Algorithm: Test design considerations

When the internals of a software system are known, developing tests is straightforward. In an AI platform solution, "interpretability"[15] of the AI and ML is low – i.e., input/output mapping is the only known element and the mechanism for the underlying AI function (prediction, for example) cannot be looked at or understood. Although traditional black box testing helps address the input/output mapping, when there is lack of transparency humans will have difficulty trusting the testing model.

AI platform solutions that we helped implement were black boxes. Traditional black box testing techniques help us meet most of the test requirements. However, there are unique AI techniques available that help validate AI functionality, so testing can go beyond pure input and output mapping. Key AI-driven black box testing techniques that we have found useful to AI platform adoption are:

- **Posterior predictive checks simulate replicated data under the fitted model** and then compare these to the observed data.[16] Taking this approach, testing can look for systematic discrepancies between real and simulated data.
- **Genetic algorithms to optimize test cases:**[17] The challenge of generating test cases is to search for a set of data that leads to the highest coverage when used as input to the software being tested. If this problem is solved, the test cases can be optimized. There are adaptive heuristic search algorithms that perform basic acts of natural evolution such as selection, crossover and mutation. In the generation of test cases using heuristic searches, feedback information concerning the tested application is used to determine whether the test data meets the testing requirements. The feedback mechanism gradually adjusts test data until the test requirements are met.



Fuzzy logic-based approaches automatically refine abstract models to generate detailed models that permit the identification of the traceability links.

- **Neural networks for automatic generation of test cases:** These are physical cellular systems that can acquire, store and process experiential knowledge. They mimic the human brain in order to carry out learning tasks. Neural networks learning techniques are used in automatic generation of test cases.[18] In this model, a neural network is trained on a set of test cases applied to the original version of the AI platform product. Network training pivots on inputs and outputs of the system. The trained network can then be used as an artificial oracle for evaluating the correctness of the output produced by new and possibly faulty versions of the AI platform product.
- **Fuzzy logic for model-based regression test selection:** This approach selects test cases on the basis of changes made to the models of a software system. While these approaches are useful in projects that already use model-driven development methodologies, a key obstacle is that the models are generally created at a high level of abstraction. They lack the information needed to build traceability links between the models and coverage-related execution traces from the code level test cases. Fuzzy logic-based approaches[19] automatically refine abstract models to generate detailed models that permit the identification of the traceability links. The process introduces a degree of uncertainty, which is addressed by applying fuzzy logic based on the refinements to allow the classification of the test cases as retestable according to the probabilistic correctness associated with the refinement.

Integration: Test design considerations

All SaaS solutions, AI-as-a-Service included, come with a set of defined web services that enterprise applications and other intelligent sources can interact with to deliver the promised outcome. Additionally, AI platforms come with EAI technologies that use web services to simplify the integration of business processes, workflows and databases across an organization's systems.

Web services have evolved to provide platform independence – i.e., interoperability. The complexity of these interfaces will demand an increased level of testing to ensure that the systems communicate properly. In a CI/CD environment, it is critical to check the compatibility of these interfaces in every build.

In the AI product adoptions that we worked on, the primary challenge was to virtualize the web services and validate the data flow between the AI platform solution and the application or the IoT interfaces. The reasons for complexity include:

- There is no user interface to test, unless it is consumed or integrated with another application/source that may or may not be ready to test.
- All elements of a service need to be validated no matter which application uses them or how often they might be invoked.
- The underlying security parameters of the services must be validated.
- Connection to services is made through different communication protocols.
- Multiple channels calling a service simultaneously leads to performance and scalability issues.

Conventionally, end-to-end processes are tested – as an end user would do when using the application UI. AI products communicate using APIs, web services and messaging middleware via a messaging bus. The UI is generally provided by the AI platform product itself.

Most business rules and functional complexities typically reside in the middle tier. This creates an opportunity to validate business rules and functionality, enhancing coverage at this tier. Since communication between the interacting components or interface layers happens through message transactions, we focused on the following during the AI testing process:

- **Smart virtualization:** In a world of iterative design and development, the application layers, interfaces and data extractions are built over successive iterations through distributed teams. Similarly, enterprise IT teams must be working in parallel preparing for the anticipated changes in the AI product releases. In both cases, a complete interface layer may not always be available for testing. To test the complexities in the interfaces such as human, machine and software, virtualization techniques are useful. Using data, device and server virtualization models, we tested the interfaces by inputting messages directly into a system, thereby simulating end-user action and reducing UI or application dependency.

- **Looking beyond interface design:** Oftentimes, critical business logic resides in various databases, in the form of database objects. Looking beyond, interfaces help simulate any upstream application and data behavior using the communication layer, thus reducing the dependency on upstream applications. Similarly, any downstream check on the data flow can be performed by verifying the status in downstream databases or validating the returning messages. Message-based validation[20] and database object validation techniques are also quite useful.
- **Validating business functionality:** We validated every service or API, parsed and verified messages and queried the status of the service calls in the database.
- **Checking for nonstandard code usage:** Advances in AI are achieved not just due to the availability of large data sets and more computing power, but also due to high-quality open-source libraries, which allow developers to quickly code and test AI models to be used in conjunction with the AI platform. Evolution of AI platforms have also led to proliferation and percolation to the real-world applications that enterprises are starting to use. These practices could bring nonstandard code and data into the enterprise IT environment, and these codes and data need special attention during testing. We established an open-source council with representatives from enterprise IT and business organizations to properly govern the usage of open-source libraries.

Advances in AI are achieved not just due to the availability of large data sets and more computing power, but also due to high-quality open-source libraries, which allow developers to quickly code and test AI models to be used in conjunction with the AI platform.

Experiences: Test design considerations

Today, the digital mandate for enterprises is customer-centricity. Delivering an unmatched customer experience is now imperative for business success. This is an even greater objective in AI adoption.

Examining only attributes such as performance, security, accessibility, etc. is not good enough. Experience assurance should address the complexity that cloud and AI solutions are adding. Functionality of AI systems keeps evolving as systems "learn." Testing performed in the first iteration therefore may be irrelevant in the second iteration, as the system's behavior baseline might have moved. In a dynamic situation like this, the testing mechanism should also be based on constant learning from the solution adoption process. We implemented ML-based algorithms (see Figure 6) that analyze code, data, server, test and usage patterns to develop various insights, which in turn are used to activate learning, self-healing and fail-over routines that boost customer experiences.

ML-driven prediction systems used in AI platform testing

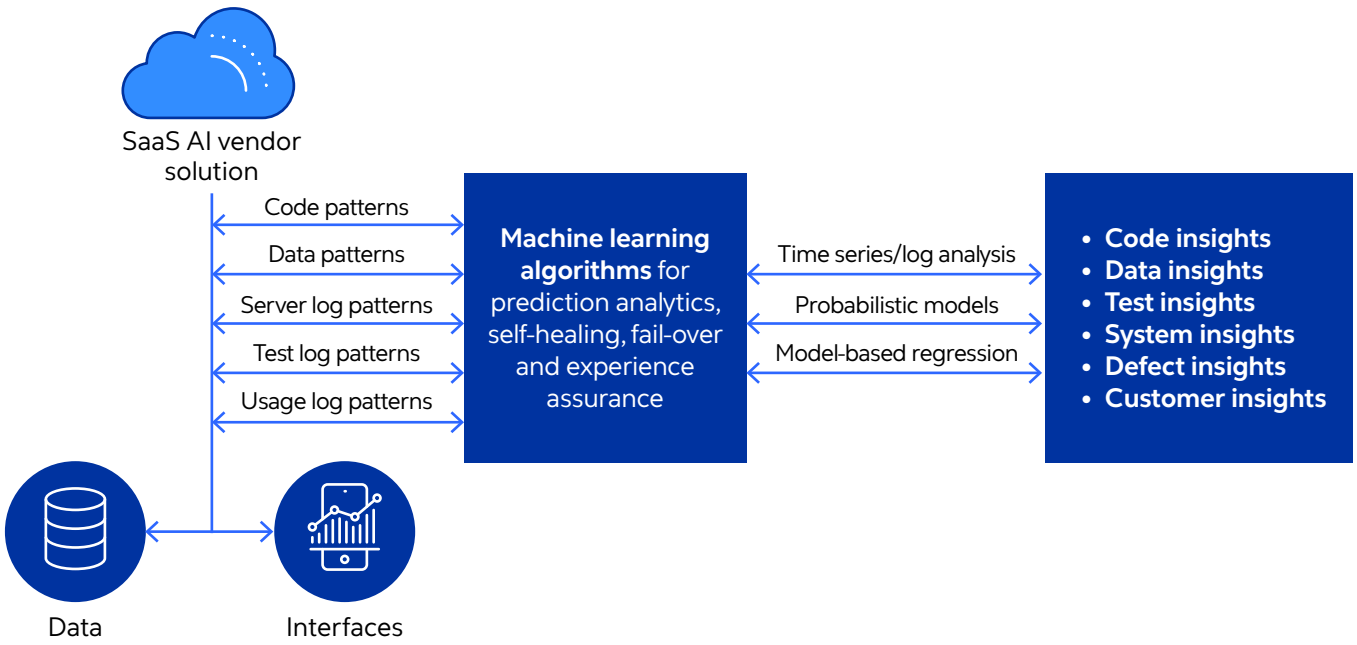


Figure 6

We found the following design considerations to be very effective in the AI experience testing framework:

- **Engineer for experience, then test for it.** Enterprise AI strategy should be derived from an end-user perspective. So, it is important to make sure that the testing team represents actual customers. Early involvement of customers helps not just the design, but also provides early access to earn trust.
- **Agility and automation should be delivered through a build-test-optimize model.** Testing cycles at the Scrum level should have considerations for user experience. Necessary peer reviews were established to achieve it, which eventually helped implement a build-test-optimize cycle.
- **Continuous security with an Agile approach is critical.** We had the corporate security team part of the Agile teams to (1) own and validate the organization's threat model developed on the AI product solution at the Scrum level, and (2) evaluate the structural vulnerabilities (from a hypothetical hacker's point of view) for all the multichannel interfaces that an AI solution architecture may have.
- **Speed is critical.** Attributes of the AI data such as volume, velocity, variety and variability forced preprocessing, parallel/distributed processing and/or stream processing. Testing for performance at all these levels helped optimize the design for the speed that users expect from the system.
- **Nuances of text, voice and video testing are important.** Research suggests that conversational AI remains at the top of corporate agendas, spurred by Amazon Alexa, Google Assistant and others. New technologies such as augmented reality, virtual reality, edge AI, etc. continue to emerge. Testing text, voice, video and NLP systems aligned with potential customer profiles should be deployed. This is especially important when crowd testing may not be a choice.
- **Simulation helps test the limits.** Checking for user scenarios is fundamental in experience assurance. Testing, exceptions, errors and violations helped predict AI system behavior, which in turn helped us validate the error/fault tolerance level of AI applications.
- **Trust, transparency and diversity are the foundations for AI governance.** Critically important activities include verifying the trust that enterprise users develop for the AI outcome, validating the transparency requirements of the data sources and algorithms to reduce the risks and grow confidence in AI, and ensuring diversity in the data sources (by involving testers in checking AI ethics and accuracy). To make all this happen, we made sure to use testers with both increased levels of domain knowledge and the technical know-how of the data, algorithm and integration processes within the larger enterprise IT.

Looking forward

Continuous testing is a fundamental requirement for AI platform adoption. When we adopted the modular approach of perfecting the designs of data, algorithm, integration and experience assurance activities, we could create an ecosystem where enterprise IT was much more prepared to accept the frequent changes in internal and external AI components. As testing processes and the tools are continuously enabled by AI and ML techniques, we foresee AI platforms evolving into self-testing and self-healing systems.

Endnotes

1 "Top Trends on the Gartner Hype Cycle for Artificial Intelligence 2019," Gartner, Sept. 12, 2019; www.gartner.com/smarterwithgartner/top-trends-on-the-gartner-hype-cycle-for-artificial-intelligence-2019/.

2 "Software as a Service (SaaS) Market Worth USD 60.36 Billion, at 9% CAGR During 2019-2023," Technavio, Bloomber.com, June 26, 2019; www.bloomberg.com/press-releases/2019-06-26/software-as-a-service-saas-market-worth-usd-60-36-billion-at-9-cagr-during-2019-2023-technavio.

3 Op. cit., Gartner.

4 "Cloud adoption to accelerate IT modernization," McKinsey Digital, April 12, 2018; www.mckinsey.com/business-functions/mckinsey-digital/our-insights/cloud-adoption-to-accelerate-it-modernization#.

5 Filip Karlo Došilović, Mario Brčić, Nikica Hlupić, "Explainable artificial intelligence: A survey," IEEE, July 2, 2018; https://ieeexplore.ieee.org/document/8400040.

6 Mojtaba Shahin, Muhammad Ali Babar, Liming Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," IEEE, March 22, 2017; https://ieeexplore.ieee.org/document/7884954.

7 S.A.I.B.S. Arachchi, Indika Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," IEEE, July 31 2018; https://ieeexplore.ieee.org/document/8421965.

8 Ibid.

9 Yuecai Zhu, Emad Shihab, Peter C. Rigby, "Test Re-Prioritization in Continuous Testing Environments," IEEE, Nov. 12, 2018; https://ieeexplore.ieee.org/document/8530018.

10 Remo Lachmann, "Machine Learning-Driven Test Case Prioritization Approaches for Black-Box Software Testing," The European Test and Telemetry Conference 2018; www.ama-science.org/proceedings/getFile/ZwtmZt==.

11 Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, Khaled Shaalan, "Speech Recognition Using Deep Neural Networks: A Systematic Review," IEEE, Feb. 1, 2019; https://ieeexplore.ieee.org/document/8632885.

12 Patrick Doetsch, Michal Kozielski, Hermann Ney, "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition," IEEE, Dec. 15, 2014; https://ieeexplore.ieee.org/document/6981033.

13 Ali Mollahosseini, David Chan, Mohammad H. Mahoor, "Going deeper in facial expression recognition using deep neural networks," IEEE, May 26, 2016; https://ieeexplore.ieee.org/document/7477450.

14 Julen Kahles Bastida, "Applying Machine Learning to Root Cause Analysis in Agile CI/CD Software Testing Environments," Aalto University School of Science, Dec. 20, 2018; https://aaltodoc.aalto.fi/handle/123456789/36347.

15 Op. cit. Filip Karlo Došilović, etc.

16 Andrew Gelman, Jennifer Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models*, pp. 158, 169, Dec. 18, 2006; www.stat.columbia.edu/~gelman/arm/.

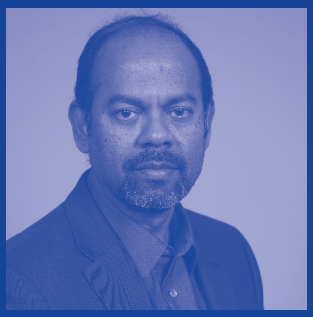
17 Xiaoan Bao, Zijian Xiong, Na Zhang, Junyan Qian, Biao Wu, Wei Zhang, "Path-oriented test cases generation based adaptive genetic algorithm," doi.org, Nov. 14, 2017; https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0187471.

18 Meenakshi Vanmali, Mark Last, Abraham Kandel, "Using a neural network in the software testing process," *International Journal of Intelligent Systems*, Jan. 2002; www.researchgate.net/publication/220063934_Using_a_neural_network_in_the_software_testing_process.

19 Mohammed Al-Refai, Walter Cazzola, Sudipto Ghosh, "A Fuzzy Logic Based Approach for Model-Based Regression Test Selection," IEEE, Sept. 9, 2017; https://ieeexplore.ieee.org/document/8101248.

20 Sylvain Halle, Roger Villemaire, "Runtime Monitoring of Message-Based Workflows with Data," IEEE, Sept. 26, 2008; https://ieeexplore.ieee.org/document/4634758.

About the author



Anbu Ganapathi Muppidathi

Global Markets Leader, Enterprise Application Services (EAS) Practice, Cognizant

Anbu Muppidathi is the Global Markets Leader for Cognizant's EAS Practice and a member of Cognizant's Executive Leadership Team. As the global market leader, Anbu is responsible for the growth and success of the enterprise cloud technology SaaS practices that include Salesforce, SAP, Workday, Oracle, IBM, PEGA, etc. in the fields of supply chain management, finance, sales, human capital management, customer experience, business process management and integration management. Prior to this role, Anbu was the Global Markets Leader for Cognizant's Quality Engineering and Assurance Practice between 2014 and 2018 and was the NA Practice Leader for Cognizant's Banking and Financial Services Practice between 2002 and 2014. Anbu is a member of the Harvard Business Review Advisory Council and Forbes Technology Council, and a winner of IDG, CIO Magazine's "2019 Ones-to-Watch" Award. He obtained his MBA from Duke University and a master's degree in computer applications from Alagappa University in India. Anbu can be reached at Anbu.Muppidathi@cognizant.com | www.linkedin.com/in/AnbuMuppidathi.

About Cognizant

Cognizant (Nasdaq-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 194 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

Cognizant

World Headquarters

500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters

1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters

#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

APAC Headquarters

1 Changi Business Park Crescent,
Plaza 8@CBP # 07-04/05/06,
Tower A, Singapore 486025
Phone: + 65 6812 4051
Fax: + 65 6324 4051

Codex 5832