International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

SCALABLE UAT EXECUTION FRAMEWORK FOR B2B APPLICATIONS WITH DATA-DEPENDENT ENTRY/EXIT CRITERIA USING SQL QUERY HOOKS

Udayan Verma

Denver, USA
udayanverma7@gmail.com

Abstract:
The current report is forward-looking automation architecture with Business-to-Business (B2B) applications being optimized through SQL query hooks as data reliant entry and exit points to the User Acceptance Testing (UAT). Conventional UAT systems are unlikely to work with a dynamic backend data environment and introduce such inefficiencies as premature test execution and premature validation of a database change. The suggested scheme, entails SQL-based gating functions in the automation stream of events and ascertains that the testing procedure stipulates the fulfillment of the preconditioned data circumstances and ceases when the checking of the entirety of the demanded transitions of the backend outlooks has been executed. This data-driven execution model improves the reliability of data testing by excluding the false positives, less human control and gives accurate points of test to go/no-go decision-making. This framework is a strong platform that supports the enhancement of the quality assurance procedures in complex B2B ecosystems because it can support large data volumes and scale tests without issues, besides being compatible with continuous integration and delivery (CI/CD) pipelines.

Keywords: UAT Framework, SQL Query Hooks, B2B Automation, Data-Driven Testing, CI/CD Integration.

I. INTRODUCTION

The final stage of the software delivery cycle is User Acceptance Testing (UAT) which is conducted to confirm that the applications are developed in accordance with the business requirements before production. Good UAT is valuable in B2B where systems require making large volumes of transactions and complex data exchanges between two or more stakeholders. Simple processes are usually not dynamic in validating their processes and this makes the tests to start with incomplete data or early termination of the process before the back-end validation has taken place. These gaps cause flaws into production, slack release, and lack of confidence in the stakeholders.

Such issues are addressed with the SQL query hook in the UAT pipe and a data-driven method of entry and exit criteria. Entry gating is used to verify the presence and the validity of prerequisite data, i.e. customer profiles or batches of transactions, before tests are carried out. Exit gating is a confirmation that all the anticipated updates of the backend are completed before test closure. This model minimises human error and provides a repeatable, auditable, and scalable process that is appropriate to large-scale B2B applications. With the implementation of SQL-based conditional logic as part of CI/CD processes, UAT processes have become more dependable and release cycles have been shortened.

II. BACKGROUND AND RATIONALE

Contemporary B2B systems exist in interconnected ecosystems with numerous subsystems interacting in real time to provide a number of mission-important services, including telecommunications, financial platforms, and supply chain functions. The breakdown of one element can break the entire processes and hence data integrity is necessary in UAT.

Old methods use a lot of fixed scripts and hand checks, and which cannot be used in large-sized systems. They also tend to fail to adjust to the dynamic conditions at the backend thus running tests without preparing the data accordingly or they can end without fully testing the changes [1]. The result is redundant execution process and latent defects that remain unnoticed at the cost of UAT results.

Charter communications came up with a framework to address these issues using SQL query hooks which is a dynamic gating system. The questions are used to determine whether to proceed with tests, suspend or discontinue tests in real time based on live backend information. This type of alignment of the test process and reality system conditions reduces the false positives and increases defects detection to produce a scalable, automated model that is applicable to complex B2B systems and compliance requirements.

III. SYSTEM ARCHITECTURE & PIPELINE DESIGN

The proposed UAT environment can interact with the existing CI/CD pipelines entirely since such tools as Jenkins can be used to organize it, and database servers can be used to validate it. In its simplest form, the system will rely on SQL query hooks to impose entry and exit criteria so that data is ready and complete at every stage of testing lifecycle.

SQL queries used to verify the presence, integrity, and condition of the prerequisite datasets are used as entry criteria. As an example, a framework checks the presence of the required records in the database before executing a scenario with partner transactions, and checks the fulfillment of the predefined conditions. It is after these checks have been passed that the system goes into execution.

The exit criteria are also the ones that are similar and they are based on the SQL queries to ensure that all the updates that were done to the back ends were done as expected [2]. This involves checking transactions that have been recorded, balances checked and status documented. Manual post-tests can also be abolished by the automated checks.

The parameterized SQL hooks that can be used in various situations and settings of partners can be identified as one of the most important features. Modularity helps in ensuring that the maintenance work is reduced and the expansion is improved. Dynamic decision-making of the pipeline is also possible with the help of the conditional logic.

Criteria Type	Purpose	Example Use Case
Entry Criteria	Validate prerequisite datasets before test execution	Verify customer and partner records exist
Exit Criteria	Confirm completion and accuracy of backend updates	Validate transaction reconciliation post-test

Table 1: Entry vs Exit Criteria Overview

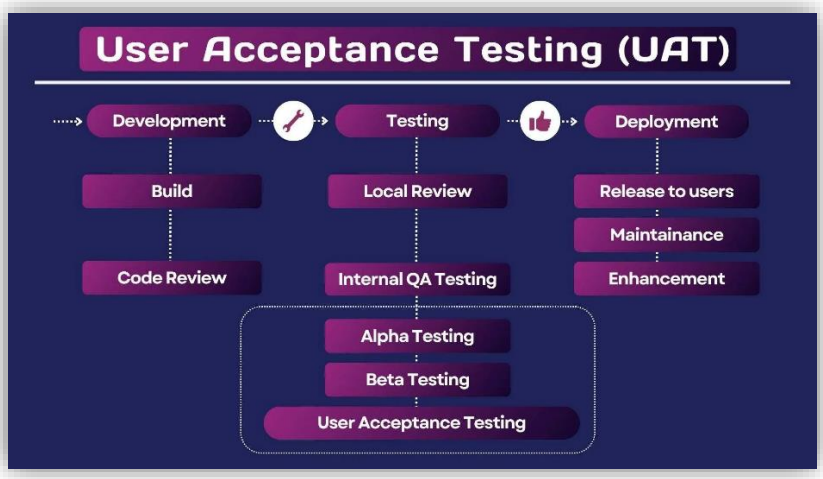

Fig 1: UAT Workflow Architecture

This kind of integration has ensured that the UAT processes are closely integrated with the back-end systems that can be a sure method of managing the complex data dependencies of the enterprise application.

IV. INTEGRATION & MAINTENANCE STRATEGY

The integration and maintenance strategy is a closely related concept to the logistic strategy. The success of UAT framework will be successful in the long run depending on the capability of coping up with the changing business and technical specifications. It is integrated with CI/CD pipelines by directly executing SQL queries within pipeline stages allowing automated gating to occur without human intervention [3]. The Jenkins pipeline is able to interrogate the results of SQL queries dynamically, and decide whether to continue with a test, wait to investigate, or terminate.

Maintenance deals with SQL hook modularity and proactive SQL hook management. By use of parameterized queries, redundancy is minimized and it is easy to make updates in case of change in the database schema or rules [4]. Version control logs script and query modifications, allowing traceability and rollback. A constant check of the database performance ensures that the execution of queries does not slow the testing pipeline.
The structure will be resilient and scalable to meet the continuous delivery goals by integrating automation and structured maintenance.

V. SCALABILITY & OPTIMIZATION

Scalability is essential in a business-to-business setting where thousands of transactions are delivered at the same time. The framework facilitates concurrent execution of SQL queries and test cases on distributed setting. Containerization e.g. Docker guarantees the presence of similar environments and the removal of configuration drift.

Dynamic resource allocation may change capacity with demand [5]. The resources are added during peak periods so that extra volumes are managed, and idle resources can be released, avoiding unnecessary costs to the performance.

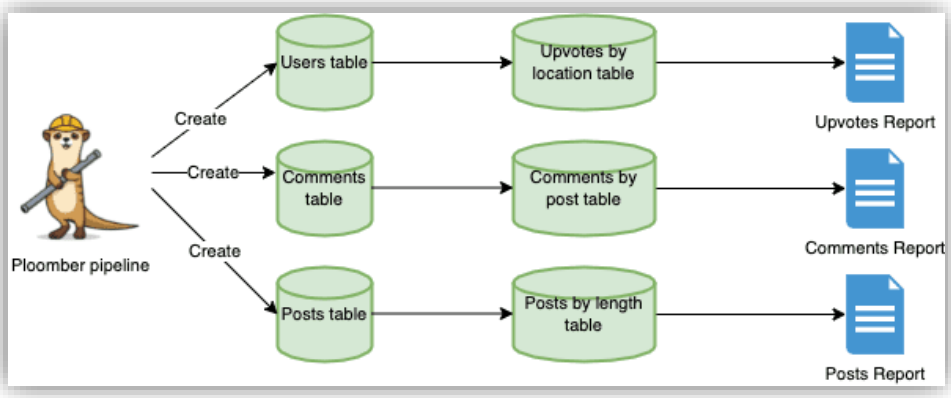

Fig 2: Scalable UAT Pipeline with SQL Gating

VI. REPORTING & MONITORING

The structure is efficient regarding overall reporting and monitoring. The outcomes of the SQL queries are automatically constructed into structured reports that present the auditors and stakeholders with clear evidence of the outcomes of the tests [6]. With live dashboards, it is possible to see a summary of the most meaningful values, including the state of test execution, the pass / fail rate of tests, the health indicators of the system.

The weaknesses will be resolved with time as automated messages will be sent by email or by other tools of use-collaboration like Slack. Reporting tools such as JUnit or Allure can be used with it to provide more detailed visualization such as logs and trend analysis. Hook Name Purpose Sample Output.

Hook Name	Purpose	Sample Output
Check_PreData	Validate required datasets	Pass / Fail
Verify_PostUpdate	Confirm backend updates	Pass / Fail
Transaction_Status	Monitor transaction lifecycle	Pending / Complete

Table 2: Example SQL Query Hooks

These abilities are applied to convert raw execution data into actionable data to make effective decisions and to make improvements.

VII. CONCLUSION

The UAT framework described in this report is scalable and it manages the cracks that are present in the traditional testing framework in the B2B applications. The framework will enable the entry criteria and exit criteria to be SQL queries in which upon meeting the requirements and upon the tests are to be performed. This will result in low false test start, less manual validation work and reliability of the UAT results is increased. The structure is a future-proven architecture created in complicated enterprise settings, as a result of its scaling, ongoing CI/CD combination and modular design. Since data-driven testing methods in organizations remain a practice, the given framework provides a solid basis of quality increment in software and speed of delivery.

REFERENCES:

1. C. Yan, S. Wang, and S. Lu, “Generating Test Databases for Database-Backed Applications,” *Proc. 45th Int. Conf. Software Engineering (ICSE)*, IEEE Press, 2023, pp. 1-12. Available: https://people.cs.uchicago.edu/~shanlu/paper/icse2023.pdf
2. Amazon Web Services, “Testing Stages in Continuous Integration and Continuous Delivery,” *AWS Whitepaper*, 2023. Available: https://docs.aws.amazon.com/whitepapers/latest/practicing-

continuous-integration-continuous-delivery/testing-stages-in-continuous-integration-and-continuous-delivery.html
3. Liquibase, "Guide to Database Continuous Integration," *Liquibase Technical Resource Guide*, 2023.Available: https://www.liquibase.com/resources/guides/database-continuous-integration
4. Jenkins Project, "Using a Jenkinsfile / Pipeline Syntax and Conditional Constructs," *Jenkins Documentation*, The Jenkins Project, 2023. Available: https://www.jenkins.io/doc/book/pipeline/jenkinsfile/ and https://www.jenkins.io/doc/book/pipeline/syntax/
5. R. L'Esteve, *The Azure Data Lakehouse Toolkit: Building and Scaling Data Lakehouses on Azure with Delta Lake, Apache Spark, Databricks, Synapse Analytics, and Snowflake*, Apress, 2022. DOI: 10.1007/978-1-4842-8233-5
6. V. Schubert, S. Kuehner, T. Krauss, M. Trat, and J. Bender, "Towards a B2B Integration Framework for Smart Services in Industry 4.0," *Procedia Computer Science*, vol. 217, pp. 1649–1659, 2023.Available: https://www.sciencedirect.com/science/article/pii/S1877050922024504