ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/ramesh-babu-potla/655

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 4 & Iss: 1

Research Article

Reducing Test Cycle Time by 70% in ERP Transformations: A DevTestOps Blueprint for Manufacturing Enterprises

Ramesh Babu Potla*

Citation: Potla RB. Reducing Test Cycle Time by 70% in ERP Transformations: A DevTestOps Blueprint for Manufacturing Enterprises. *J Artif Intell Mach Learn & Data Sci 2021* 4(1), 3242-3249. DOI: doi.org/10.51219/JAIMLD/ramesh-babu-potla/655

Received: 02 March, 2021; **Accepted:** 18 March, 2021; **Published:** 20 March, 2021

***Corresponding author:** Ramesh Babu Potla, ERP IT SAP Delivery Manager/ERP Digital Transformation COE/ BTP COE / Gen AI, USA

ABSTRACT

Digital initiatives like Enterprise Resource Planning (ERP) changes in manufacturing enterprises are some of the most complicated and risky undertakings, in part because of the large-scale customization, intricate integrations and business critical processes. The testing processes usually take 30 and 50 percent of the ERP program schedule and in most cases, it is the greatest point of bottlenecking to release agility. The conventional methods of testing, namely, late involvement in tests, manual testing, vulnerable environments and rigid regression packs are not able to meet the speed, quality and predictability requirements of the contemporary manufacturing organizations. In this paper, a DevTestOps blueprint that suits SAP ERP landscapes is proposed and incorporates testing to the software delivering lifecycle. This strategy focuses on shift-left unit testing, automated test data management, service virtualisation, risk-based regression optimisation and end to end (E2E) automation driven by CI/CD pipelines. However, in contrast to generic DevOps models, the proposed DevTestops model clearly addresses SAP-related limitations, including transport management, intricate master data relationships and heterogeneous system integrations, which are typical within a manufacturing ecosystem. A quantitative assessment of the proposed solution on the basis of multi-year ERP transformation programs shows that the suggested solution results in the achievement of up to 70 percent decrease of test cycle time, 45 60 percent decrease of escaped defects and considerable increase in release predictability and business confidence. The results confirm that DevTestOps is an important enabler of scalable and high-quality ERP delivery in manufacturing firms.

Keywords: DevTestOps, SAP ERP, Manufacturing Systems, Test Automation, CI/CD, Risk-Based Testing, Service Virtualization, Test Data Management, ERP Transformation.

1. Introduction

1.1. Background

The manufacturing businesses are also more and more relying on ERP solutions like SAP ECC and SAP S/4HANA that provide more closely integrated processes in production planning, supply chain management, finance and quality management[1,2]. With organizations undertaking digital transformation programs such as system upgrades, brownfield conversions and greenfield implementation, modernisation of the ERP has become a strategic priority to operational efficiency, scalability and long-term digital resiliency. In spite of their significance, these transformation programs are often met with a lot of challenges such as cost overruns, prolonged schedules and post go-live instabilities. The poor and inefficient testing practices have been repeatedly cited in the previous industry and academic research as a leading cause of such failures, especially in large scale manufacturing settings where business processes are much interdependent. The conventional ERP testing methods are mostly phase-gated with most of the test execution

being postponed to system testing, regression testing and user acceptance testing phases towards the end of the lifecycle of delivery. This model leads to the situation where defects are detected at the stage when the costs of the remediation are the greatest and the time flexibility is the lowest and organizations frequently have to decide between late go-lives and tolerance of known defects. The problem is also aggravated when it comes to production of ERP landscapes which entail complicated integrations with systems like Manufacturing Execution Systems (MES), Product Lifecycle Management (PLM), Warehouse Management Systems (WMS) and third-party logistics providers. These environments have a scale, intensity of data and complexity of integrating such overwhelming systems that manual, end-to-end testing approaches are becoming less and less sustainable. These constraints underline the necessity of the more proactive, automated and risk-based testing paradigm that can be applied to ensure the faster delivery at the same time being able to provide the stability needed in the mission-critical manufacturing processes.

1.2. Limitations of conventional ERP testing

The traditional ERP testing methods have a number of systemic constraints which considerably restrict their efficacy in large scale manufacturing settings. Late validation of business logic and custom code is one of the most important weaknesses and meaningful testing is often postponed until system or user acceptance testing phases. This delay causes the defects to be detected at the most expensive and disruptive stage of remediation, which often causes schedule slips or reduced quality of release. Problems found at an early stage during a custom development or configuration tend to spread throughout integrated modules amplifying the level of defect severity and complexity of solutions. The other significant constraint is that it heavily depends on manual functional testing, which is ineffective in the ERP settings with large business process coverage and high turnover. Manual execution is laborious, prone to errors and cannot be consistently repeated in different releases, decreasing confidence in test results. It is also complicated by lack of availability of test data as the traditional methods rely on ad hoc or outdated data sets that cannot mirror real business situations. Long data refresh times and inconsistent master data can lead to test failures which do not actually reflect defects in the system and divert the testing energy to this non-meaningful task. Moreover, traditional ERP testing does not have good isolation and validation mechanisms to dependent systems. ERP environments are manufactured to interoperate with a greater number of external systems including MES, WMS and third-party logistics that are frequently unavailable or unstable when under test. The lack of the ability to test integrations independently makes teams experience scheduling delays and incomplete coverage of the exception cases. Lastly, classical regression testing uses fixed test suites, which are steadily increasing in size as the functionality increases. With time, these suites become unmanageable, resulting in high execution time or biased dispensation of tests, all of which raise the chances of escaped defects. All these constraints collectively highlight the importance of increasingly automated, data-based, risk-oriented testing paradigm to the intricacy of ERP manufacturing systems.

1.3. Emergence of DevTestOps

The need to increase software delivery speed and reliability has motivated the transformation of time-tested methods of development and testing into the automation-based integrated models[3,4]. DevTestOps is thus DevOps arriving at the logical next stage of making the concept of testing a first-class, ongoing process as well, not a checkpoint. DevTestOps focuses on the initial validation, feedback and collective responsibility of quality among the development, testing and operations teams.

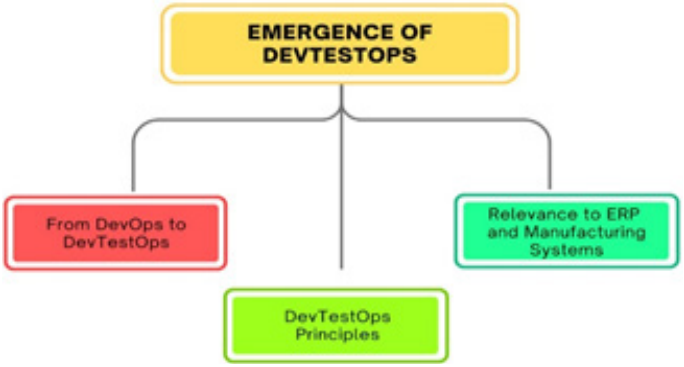

Figure 1: Emergence of DevTestOps.

- **From DevOps to DevTestOps:** DevOps was created in response to the lack of efficiency in siloed development and operations teams and concentrated on continuous integration, continuous delivery and the automation of infrastructure. Although the deployment speed and operational stability were greatly enhanced with the help of DevOps, testing was still to a large extent manual and phase-based in most organizations. DevTestOps fills this void by integrating automated testing and quality engineering activities into DevOps pipelines. This is because this evolution makes quality ratified throughout the delivery cycle since the code, configuration and integrations change and is not tested at the conclusion of the delivery cycle.
- **DevTestOps principles:** DevTestOps is based on the principle of shift-left testing, multiple-level test automation, environment and data preparedness and quality measurement continuity, at the most fundamental. The testing activities are also synchronized with the development workflows and automatically triggered as part of CI/CD pipelines. Prevention of defects is given more importance than defect detection and the early feedback mechanisms make it possible to rectify the problems promptly. Together, all these principles make the system reduce the cycle time, better the defect containment and increase the overall system reliability.
- **Relevance to ERP and manufacturing systems:** The introduction of DevTestOps is especially important in the case of ERP systems within any manufacturing setup, where any problem during the release may interfere with production, inventory and financial processes. ERP landscapes are a predisposed setup, multifaceted unifications and great data reliance, which complicates late-stage testing and renders it dangerous and ineffective. DevTestOps provides a methodical way to overcome these issues as it provides the process of constant validation of custom code, configuration change and interfaces. Making necessary adjustments to the requirements of packaged ERP software, manufacturing enterprises will be able to attain faster releases, higher quality performance and increased trust in system changes.

2. Literature Survey

2.1. ERP Testing challenges in manufacturing

Testing Enterprise resource planning (ERP) in the manufacturing settings is generally accepted to be a bottleneck undertaking in terms of the presence of strained nature of the end-to-end business process, heavy reliance on master data and elevated system configuration[5-7]. And unlike standalone applications, manufacturing ERP systems combine modules, i.e. the manufacturing planning, materials, quality and financial management modules, i.e. the defects added in one part of the system can propagate across to other down-stream systems. The previous research evidence indicates that faults in ERP systems have an inequitable impact on major manufacturing metrics, such as the precision of production timing, inventory worth and the success of order completion, which increases operational and financial risk. This has been further supported by empirical data that over 60 percent of ERP errors are due to configuration errors and ad hoc ABAP code and logic, as opposed to standard code, but unit testing and automated validation of such elements are still minimal in a significant number of real-world implementations. This discrepancy in defect locations and test concentration also adds to the late defect identification and expensive defect repair processes.

2.2. DevOps adoption in enterprise systems

The implementation of DevOps has been fast in individual ones based on cloud-natives and micro-Services, whereas the uptake has been rather slow in enterprise ERP. In the literature this lag can be attributed to perceived rigidity of packaged software, the upgrade cycle imposed by vendors and corporate concerns associated with stability and compliance of the system. However, recent research shows that the concepts of continuous integration and continuous delivery can be applied to the ERP situation in a pragmatic way through reinterpretations of the concepts. In particular, transport management automation, consistency checks on configuration, interface validation and providing test data have been demonstrated to minimize manual effort and deployment risk. This finding indicates that although ERP systems cannot enhance DevOps in its purest form, a hybrid model, sometimes called ERP DevOps, also known as DevTestops, can provide significant improvements in the deployment rate and defect rate with little loss to the governance requirements.

2.3. Risk-based and regression optimization techniques

The use of risk-based testing has been introduced in literature of an efficient method of handling the size and complexity of ERP regression testing. Instead of approaching all the test cases in the same manner, risk-based models are used to allocate priority of execution of tests according to business criticality, functional change impact, past defect density and frequency of use. Various empirical investigations find that with systematic use of this prioritization; regression suite optimization can achieve an execution effort two to four times less without significant loss of functional coverage. Such benefits are especially notable in the ERP landscapes as entire regression cycles are both time consuming and resource intensive. Nevertheless, other studies also warn the risk-based methods rely on the incessant improvement of the risk models, that is, based on the experience of production accidents and the history of changes to prevent blind spots in the course of time.

2.4. Gaps in existing research

Although there is an accumulating scholarly and professional attention devoted to the optimization of ERP testing and implementing DevOps, gaps are observed in the current literature. The vast majority of research considers specific areas of automation of tests, transport management or risk-based prioritization, but not many research suggests an integrated DevTestOps model focused specifically on the manufacture environment of SAP ERP. Moreover, there is little and somewhat disjointed quantitative information that reflects these practices to measurable results which include; cycle-time compression, less regression effort and low rates of escaped defects in production. The benefits that have been reported are mostly anecdotal or situation-specific and it may be hard to extrapolate the results of research to large-scale manufacturing firms. Such gaps highlight the necessity of the holistic, empirically verified strategy that incorporates the principles of DevOps, risk-driven testing and ERP-specific limitations, which can, in turn, spur the development of research presented in the given paper.

3. Methodology

3.1. Overview of the DevTestOps blueprint

The suggested DevTestOps blueprint introduces a framework of quality engineering tool to fit the ERP-based manufacturing systems through the integration of testing to the whole delivery lifecycle. [8,9] Instead of using the test to be a downstream validation step, the blueprint entails incorporating quality controls into workflow processes of development, configuration, deployment and operations. The framework is designed with regards to five complementary dimensions, which altogether serve the issues of ERP complexity, regression scale and release velocity **(Figure 2)**.

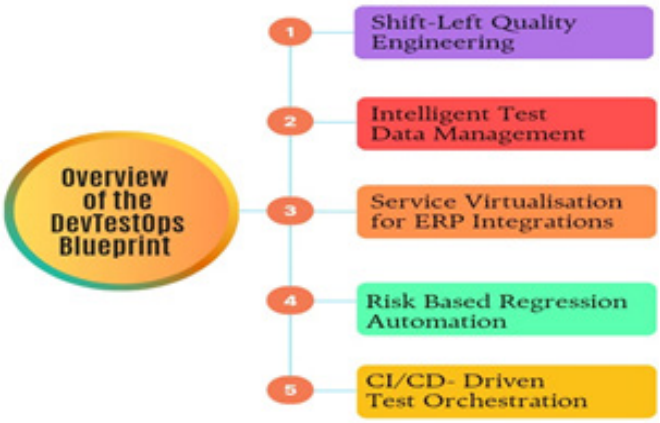

Figure 2: Overview of the DevTestOps Blueprint.

- **Shift-left quality engineering:** Shift-left quality engineering focuses on initial validation of ERP settings, custom ABAP code and integration touchpoints by design and development stages. Defects are identified at an earlier stage, as they are found by the introduction of static code analysis, checking configuration consistency and early unit level validation of defects. This will minimize the defect spreading to subsequent test stages, decrease defect remediation and enhance long-term system stability before integration and user acceptance.
- **Intelligent test data management:** The smart test data management is concentrated on making sure that proper,

compliant and business-representative data is available through the testing life cycle. Considering the data-sensitive characteristics of the ERP manufacturing processes, the blueprint will use automated data extraction, masking, refresh and synchronization methods. These features allow the ability to execute tests with repeatability and reliability and to reduce test failures caused by data and dependency on common or volatile test environments.

- **Service virtualization for ERP integrations:** Service virtualization is used to resolve the complexity of ERP integrations with other systems, including the MES, SCM and third-party logistics platform. Virtual services provide the ability to continue test services regardless of external system readiness because of simulated unavailable, expensive or volatile interfaces. The capability greatly enhances the coverage of tests in integration cases with minimal delays in the schedules and environmental dependence of multi-system ERP landscapes.
- **Risk-based regression automation:** Risk based regression automation gives priority to the running of tests in accordance with business importance, change impact and past defective patterns. Rather than running full regression suites whenever a release occurs, the blueprint determines, dynamically and in sequence, test cases which involve the greatest risk to the manufacturing operation. Such an approach to automation saves and minimizes the time cycle of regression, without losing confidence in the quality and readiness to produce.
- **CI/CD-driven test orchestration:** CI/CD-based test orchestration consists of automated testing as a part of continuous integration pipelines and continuous deployment pipelines specific to ERP environments. The automated triggers are the ones that start the right test suites after changes in codes, configuration transport or a refresh of a system. The sites achieve the faster feedback, consistent quality gates and controlled release cadence through the orchestration of the tests through environments and aligning them with deployment workflows without imposing on the requirements of ERP governance.

3.2. Shift-left testing for SAP landscapes

ABAP unit testing aims at testing custom developments with a way of validating the new development using automated ABAP Unit test cases that are run during transport creation[10,11]. These are tests that ensure that business logic, data handling and exception situations are tested independently, such that defects are discovered prior to code being advanced to higher landscapes. By taking ABAP Unit execution into the development process, it will ensure a consistent set of quality standards, a higher level of code reliability and the minimization of bringing faulty custom logic in the system integration and regression testing processes.

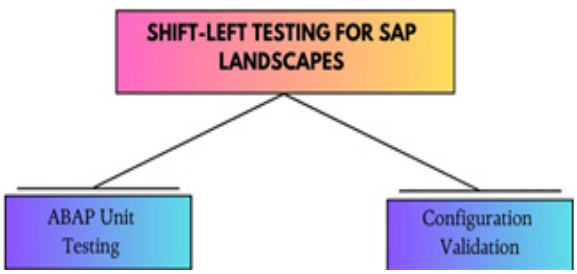

Figure 3: Shift-Left Testing for SAP Landscapes.

- **ABAP unit testing:** Configuration validation helps deal with one of the core causes of ERP defects by automatically assessing SAP configuration alterations concerning pre-set business regulations and conformity constraints. Detection Automated checks are activated when in configuration transport or preparing a release to identify any discrepancy, dependency or rule violation. This proactive validation saves the use of late phase functional validation, minimizes feedback loop and eliminates the effects of configuration defects on important manufacturing processes.
- **Configuration validation:** Configuration validation is a solution to one of the major sources of ERP errors as it will verify SAP configuration changes against a set of business rules and compliance constraints. Automated checks are activated when configuring transport or preparing the release to ensure that incompatibility or missing dependency is identified at an early stage of the lifecycle. This active validation ensures reduced dependency on post-code testing to identify the functionality of the code, reduces the number of feedback loops and limits the possibility of configuration flaws affecting key features of the manufacturing procedure.

3.3. Test Data Management (TDM)

The importance of the Test Data Management (TDM) in manufacturing ERP testing is related to a high dependence of the business process on precise master and transactional data[12,13]. Meno cases like manufacturing planning, batch operations, inspection and inventory valuation involve highly efficient operations in the real sense thus need data that is closely applicable in real time operations as well as to have meaningful results. Poor or erratic test data may tend to cause erroneous faults, malfunctioning of tests and decreasing confidence in the quality of the system. The TDM approach that is proposed to deal with those challenges is to set up an organized, mechanized and convenient mechanism of delivering dependable test information in the test life cycle of the ERP. The implementation will create automated data provisioning pipelines to allow on-demand provisioning of test data, refreshing and synchronization of test data in the development, quality and pre-production environments. Such pipelines save on the amount of manual work and make sure that test data is kept up to date with system configuration change and code evolutions. As a measure to solve the problem of data privacy and regulatory issues, the framework uses masked production-like datasets, in which sensitive business and personal data is masked without altering the referential integrity and business relevance. This will allow realistic testing without putting confidential information at risk and breaking the rules on compliance. Synthetic data generation is also used to facilitate edge-case and negative scenario testing which is perhaps hard or unsafe to obtain with production systems. Synthetic data sets provide a means to control the generation of rare state like capacity overloads, material shortages or exception-driven process flows and enhance the test coverage and system resilience. Automated provisioning, data masking and synthetic data generation combined will form a scalable and reproducible TDM capability. This combined solution reduces the number of test failures that are caused by data, increases the speed at which the test can be completed and expands the reliability level of manufacturing ERP validation processes as part of the larger DevTestOps toolbox.

3.4. Service virtualization

Virtualization of services is one of the enabling factors towards successful testing in the complex manufacturing ERP landscapes[14,15], where business processes highly rely on integrations with external and downstream system Manufacturing Execution Systems (MES), third-party logistics providers, tax calculation engines and supplier portals. Dependencies of this nature are not readily available, tend to be volatile, expensive to promote or shared by more than one program, therefore end-to-end testing is not easily structured to be scheduled and repeated on a regular basis. The proposal service virtualization solution will resolve this issue by separating the activities of ERP testing programs with the state of availability and readiness of adjacent systems and by using this method, one can conduct an independent, repeatable and continuous test. In this framework, virtual services are developed to behave like a real endpoint of integration by simulating a request-response pattern, data formats and communication protocols. The services are meant to propagate normal operational feedbacks as well as extraordinary conditions such as error notifications, timeouts and rejecting any message. Service virtualization allows the realistic testing of ERP error-handling logic as well as recovery mechanisms and performance thresholds under controlled conditions by including configurable latency and failure mode. This stage of simulation is especially helpful in the manufacturing case when master data exchange or delivery is delayed, incomplete or incorrect and can be extremely disruptive to the production and fulfilment process. Service virtualization also enables parallel development and testing as teams can test the interfaces of the SAP early even in cases where the external systems are in development or changing. Virtual services are version able and can be used across test cycles to ensure consistency and traceability and to decrease integration related bottlenecks. Service virtualization is the feature of the DevTestOps blueprint that is aimed at increasing the coverage of tests, reducing feedback loops and making the system more resilient. Dependency-induced delays can be minimized in such a way and more thorough integration testing can be undertaken, hence, producing more predictable releases and lowering the risk of integration failure when carrying out production manufacturing environments.

3.5. Risk-based regression testing

The risk-based regression testing is intended to meet the time and size limitations of the ERP regression cycle; the emphasis of testing will thus be on business-critical and change sensitive parts of the system[16,17]. This method uses structured risk assessment to determine the degree of test selection and test execution sequence instead of performing an exhaustive and fixed set of regressions on each release. It has been demonstrated that prioritizing the remunerated initiatives of regression testing in line with the operational risk will allow organizations to gain faster response but maintained assurance of systems stability, especially in manufacturing settings where system-related defects may have a domino business effect.

- **Risk scoring model:** The risk scoring model proposed provides a quantitative risk value to every test case obtained by integrating three important dimensions, namely business criticality, change impact and the historical density of defects. Business criticality indicates the value of the process driving that to the core manufacturing activities, like production planning, inventory management or financial posting. The change impact factor depicts the degree key of the most recent code, configuration or integration alterations to the functionality that the test case covers. Historical defect density contains the occurrence and harshness of defects discerned previously in the identical functional field. The model takes these three factors multiplied together to yield a composite risk score that provides objective differences between high-risk and low-risk situations. The cases that score higher are given precedence in implementation of the regression, such that the areas that are most likely to create a disruption to the business are proved first.
- **Regression pack optimization:** Regression pack optimization is a variation of a risk scoring model that automatically collects regression tests to create a regression pack of each release. Instead of having a fixed combination of test cases, the regression pack is compiled depending on the recent changes in the system, risk rating and the release scope. The test cases that are at risk are always presented and the less risky cases can be run selectively or on a schedule, depending on time and resource availability. This adaptive strategy saves a lot of time on the overall regression running time as well as proper enough functional coverage and functional quality assurance. Since the risk scores are continually improvements of production feedback and defect trends, the optimized regression strategy is maintained in line with the changing system behaviour and business priorities.

3.6. CI/CD pipeline integration

The integration of CI/CD pipelines has been identified as the execution infrastructure of the DevTestOps blueprint that automates and orchestrates testing processes throughout the ERP delivery life cycle[18,19]. With pipelines in SAP contexts, they are modified to handle transports, configuration adjustments and governance restrictions but remain susceptible to continuous feedback on transports. The quality assurance is also a process that is continuous and repeatable, since the automated validation steps are already part of the pipeline and no longer confined to the Release-based activities **(Figure 4)**.

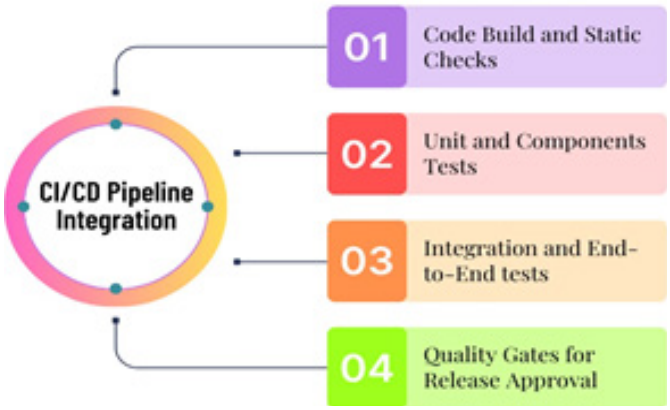

Figure 4: CI/CD Pipeline Integration.

- **Code build and static checks:** The pipeline is initiated by automated code build and statical analysis processes that checks custom developments prior to its packaging into transports. The codes of standard are enforced by checking the codes in terms of sizing, syntax aspects for coding, performance risk and security weaknesses and as well

compliance with the guidelines of SAP development. Earlier detection of problems on the code level means that fewer defective transports are going to proceed to downstream testing and enhances the quality of code in general.

- **Unit and component tests:** After successful static validation, unit and component tests are automatically run so as to check the functional correctness of individual objects and tightly related components. Unit tests involve single logic whereas component tests are used to test interaction among related SAP objects like classes, function modules and configurations. Pipeline automation of these tests gives fast feedback to developers and prevents incremental regressions.
- **Integration and end-to-end tests:** End-to-end (E2E) tests and integration are then initiated when a change is promoted to common testing environments. These tests authenticate entire business processes flows between SAP modules and integrated outside systems and in numerous situations, have recourse to service virtualization where immediate reliance's do not exist. Carrying out integration and E2E in the pipeline also means that cross-functional and cross-system effects are identified at an early stage to mitigate potential failures in the user acceptance testing or the production system.
- **Quality Gates for release approval:** Quality gates are formal points of control in the CI/CD pipeline and can be used to check the release readiness. Before promotion to higher environments metrics are looked at including test pass, defect severity levels, code quality levels and risk-based coverage of any program. Releases are only authorized after meeting specified quality criteria, which means that there will be consistent management between the governance aspect and also facilitate quicker and authoritative ERP deployments.

4. Results and Discussion

4.1. Cycle-time compression

One of the key measurable implications of the offered DevTestOps framework is cycle-time compression, which demonstrates how the system will enable the ERP delivery to deliver faster improvement and quality without reduction **(Figure 5)**. The framework is expected to drastically cut the time taken by main testing stages which are typically a bottleneck in the manufacturing project of SAP, by bringing in the automation and risk-based prioritization and early validation practices **(Table 1)**.

Table 1: Cycle-Time Compression.

Phase	Traditional (Days)	DevTestOps (Days)
System Testing	30	12
Regression	20	6
UAT Support	15	6

- **System testing:** The conventional model of ERP delivery takes around 30 days to be tested because of the magnitude of manual tests execution, environment dependencies and late defects discovery. System testing under the DevTestOps strategy consumes about 12 days which includes automated execution of tests, proving test data and service virtualization. Early defects prior to product development are to be found by performing shift-left testing that reduces the amount of rework at this stage and facilitates quicker stabilization of end-to-end business processes.
- **Regression testing:** The greatest time-cutting deals with regression testing, as it goes through an estimated time of 20 days in the classic framework to approximately 6 days in the DevTestOps. This has been enhanced majorly due to the risk-based automation of regressions where high impact test cases are given priority and unnecessary execution of tests are removed. Dynamic regression pack optimization provides the optimization of testing efforts on areas that are most influenced by changes with a large margin of speed up feedback cycles without losing confidence in the integrity of systems.
- **UAT support:** User Acceptance Testing (UAT) support normally takes up to 15 days with the involvement of testing teams to deal with defects, data complications and the instability of the environment. Under the Scheme of DevTestOps, support effort at UAT is minimized to about 6 days with the systems more mature at entry of UAT, better quality defects and readiness to test data. The effect of this is that business users are less distraught by such disruptions and that the UAT cycles are completed with more efficiency at lower dependency rate in the context of extended testing support.

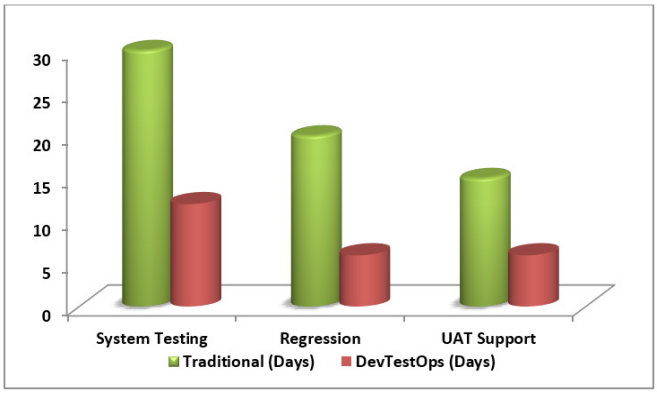

Figure 5: Graph representing Cycle-Time Compression.

4.2. Defect leakage reduction

Less leakage of defects is a vital pointer of effectiveness in the proposed framework of DevTestOps in enhancing the quality of the ERP system and the stability of operations. In the traditional models of ERP delivery, many defects are passed down to production because of late testing, unfavourable system regression and insufficient verification of configuration and integration change. Implementing the DevTestOps methodology has entailed a significant reduction of about 4560% in the number of defects in the production cycle and this is the actual evidence of the practical advantages involved in early validation and risk-driven testing practices. This is mainly due to the shift-left testing activities that are entrenched in the framework that allows faults to be detected earlier in the stages of their introduction. Performing ABAP unit testing and automated configuration validation early before the faulty custom code and incorrectly configured system can survive to system and regression testing ensure that these errors and mistakes do not advance into system and regression testing stages. The framework reduces the propagation of defects since problems are resolved sooner and,

therefore, it decreases the chances of a complex and difficult to diagnose issues emerging in the production environment. Risk-based regression testing also helps reduce defect leakage since the most business-critical and change-sensitive business ones are always verified with each release cycle. The regression process brings more attention to areas that are most likely to fail by ranking the test cases by their business impact, scope of change and past defects patterns. This focused validation has an enormous effect in reducing the efficiency of detecting defects in comparison to the conventional, cumbersome regression methods that usually dissipate the testing effort in high-risk regions. Furthermore, test coordination and service mocking through CI/CD enhances a range of integration testing and ensures uniformity through leveraging service virtualization and set of tests to validate end-to-end processes and exception handling conditions. Taken together, each of these practices forms a series of quality gates in the life cycle of delivering products and this eliminates the use of defect detection at a late stage. The reduction in escaped defects as observed but not only improves the reliability of a system but also helps reduce the cost of post-production support, increases user confidence and reduces operational disruptions during manufacturing in the ERP environment.

4.3. Release predictability

The importance of the release predictability is the key success factor of manufacturing organizations using ERP systems as the means of facilitating time-sensitive and well-integrated business functions. Finally In classic ERP delivery models, the late detection of defects, bad test environment and poor regression coverage of defects before release is common and results in late fancy schedule slips and last-minute fixes. In such a case, the success rates of releases usually fall at about 65, which kills business confidence in IT delivery. The proposed DevTestOps framework greatly enhanced the predictability of the release with success release going well above 90 and much more controlled and reliable delivery procedure. This has much been due to the fact that structured automation and continuous validation throughout the delivery lifecycle have been introduced. CI/CD pipelines enable reliability and repeatability in the execution of builds, test and quality checks and minimize variability with manual processes. Quality gates are deployed at every step of the process to promote only those changes that have passed pre-established stability and coverage criteria and allow the minimization of failures or rollbacks occurring at the last moment. Consequently, release schedules are less unpredictable and vulnerable to unforeseen inconveniences. Risk-based regression testing is also important in improving predictability by fixing the test coverage to business priorities and impact of change. The framework minimizes the uncertainty of high-risk manufacturing processes and integrations by making sure that before everything is release, the scenario is always checked up. Moreover, better data management of the tests and virtualization of the service will contribute to the stability of the environment, thereby eliminating any further delays in the case of data problems or absence of external systems. Combined, these practices would create a clear and metrics-focused delivery model which gives the stakeholders quick and consistent indicators of release readiness. The subsequent increase in the rates of successful releases builds more business confidence in IT, allows planning manufacturing operations more reliably and promotes the transition to more regular and reliable ERP releases.

5. Conclusion

As the paper illustrates, in a deliberate implementation, tailored to the SAP ERP manufacturing landscape, DevTestOps will change quality control, which predominantly is a downstream, reactive and reactive measure, to a value-adding and continuous capability that manifests itself throughout the delivery lifecycle. Critical ERP testing methodologies that extensively use late-stage manual testing and lengthy regression testing are found to be showing increased disparity with the requirement of speed, stability and predictability in the new manufacturing business ventures. To overcome these limitations, the suggested DevTestOps infrastructure proposes to implement the principles of quality engineering into development, configuration, integration and deployment in a systematic manner. The framework allows ensuring a high level of defects being detected at the early stages of its development and during the testing of configuration alterations by the practice of shift-left testing, which minimizes the spread of defects to subsequent testing. Intelligent test data management also guarantees that realistic, compliant and business representative data is always available and this is one of the greatest causes of test failure and delays in the ERP systems. Service virtualization also separates testing behaviour and behavioural dependencies on outward systems, which makes it possible to validate integration behaviours and exceptions with confidence when operating in controlled and repeatable conditions. The combination of these abilities forms a consistent and scalable testing platform that can be used to provide continuous delivery without affecting the governance or compliance mandates. The automation of risk-based regression is a significant development in the regression testing involving the size and intricacy of ERP. The framework prioritizes executing the tests by the level of business criticality, change impact and historical defect trends so as to concentrate its effort where it is giving the highest risk reduction. As part of CI/CD pipelines, these practices made it possible to have automated coordination of tests and objective quality gates to filter out only stable and promotion-ready changes. Empirical findings that have been reported in this study such as; a significant compression of time in the cycle, a significant reduction of the defects that were not eliminated and a significant improvement in predictability of releases have been quantitative evidence in the effectiveness of the framework. In addition to operational efficiency, the implementation of DevTestOps promotes closer relationships between the development, testing and operation teams making quality goals in line with business results. Enhanced release reliability builds stronger business faith of IT delivery into the manufacturing organizations so that they can plan their operations with increased confidence and agility. Finally, this study validates the claim that a customized DevTestOps strategy is not just viable in the context SAP ERP manufacturing systems but it is an indispensable approach to generating quality sustainably, speedy operations in a competitive manufacturing ecosystem.

6. References

1. Aloini D, Dulmin R, Mininno V. Risk management in ERP project introduction: Review of the literature. Information & management, 2007;44: 547-567.

2. Bradford M, Florin J. Examining the role of innovation diffusion factors on the implementation success of enterprise resource planning systems. International journal of accounting information systems, 2003;4: 205-225.

3. Doom C, Milis K, Poelmans S, et al. Critical success factors for ERP implementations in Belgian SMEs. Journal of Enterprise Information Management, 2010;23: 378-406.

4. Kim Y, Chen YS, Linderman K. Supply network disruption and resilience: A network structural perspective. Journal of operations Management, 2015;33: 43-59.

5. Lwakatare LE, Kuvaja P, Oivo M. Relationship of Devops to agile, lean and continuous deployment: A multivocal literature review study. In International conference on product-focused software process improvement. Cham: Springer International Publishing 2016: 399-415.

6. Sarker S, Lee AS. Using a case study to test the role of three key social enablers in ERP implementation. Information & Management, 2003;40: 813-829.

7. Staats BR, Upton DM. Lean principles, learning and software production: Evidence from Indian software services. Harvard Business School Technology & Operations Mgt. Unit Working Paper, 209.

8. Yoo Y, Henfridsson O, Lyytinen K. Research commentary-the new organizing logic of digital innovation: an agenda for information systems research. Information systems research, 2010;21: 724-735.

9. Ioannou G, Papadoyiannis C. Theory of constraints-based methodology for effective ERP implementations. International Journal of Production Research, 2004;42: 4927-4954.

10. Rajagopal P. An innovation-diffusion view of implementation of enterprise resource planning (ERP) systems and development of a research model. Information & Management, 2002;40: 87-114.

11. Radanliev P, De Roure DC, Nicolescu R, et al. Future developments in cyber risk assessment for the internet of things. Computers in industry, 2018;102: 14-22.

12. Lee JD, Seppelt BD. Human factors and ergonomics in automation design. Handbook of human factors and ergonomics, 2012: 1615-1642.

13. Ndoulou AO. The role of enterprise resource planning in entrenching business processes in a selected organisation in the Western Cape, South Africa (Doctoral dissertation, Cape Peninsula University of Technology), 2019.

14. Ganesh K, Mohapatra S, Anbuudayasankar SP, et al. Enterprise resource planning: fundamentals of design and implementation. Springer, 2014.

15. Garg VK, Venkitakrishnan NK. Enterprise Resource Planning: concepts and practice. PHI Learning Pvt. Ltd, 2003.

16. Denton P. Business strategy driven IT systems for engineer-to-order and make-to-order manufacturing enterprises (Doctoral dissertation, Loughborough University), 2009.

17. Felstaine E, Hermoni O. Machine Learning, Containers, Cloud Natives and Microservices. In Artificial Intelligence for Autonomous Networks. Chapman and Hall/CRC, 2018: 145-164.

18. Felderer M, Ramler R. A multiple case study on risk-based testing in industry. International Journal on Software Tools for Technology Transfer, 2014;16: 609-625.

19. Lwakatare LE. DevOps adoption and implementation in software development practice: concept, practices, benefits and challenges, 2017.

20. Amer Hamzah AF. The Development of Transport Request System in SAP using ABAP Language, 2006.

21. Nguyen DM, Huynh QT, Ha NH, et al. Automated test input generation via model inference based on user story and acceptance criteria for mobile application development. International Journal of Software Engineering and Knowledge Engineering, 2020;30: 399-425.