Ascea

EXPERIENCE MATTERS

Viewpoint

Testing, Testing 123 …

An introduction to the importance of undertaking rigorous testing, prior to systems implementation and understanding the consequences, impacts and subsequent risks of not doing so.

Introduction

I have been involved with a number of large, complex business change programmes over the years, mainly in the critical national infrastructure space at various organisations, but the lessons learned on those programmes apply to other IT / business change programmes, especially those implementations at the core of the business. My role has been primarily around Test and Assurance Management, responsible for defining and writing the testing strategy, establishing the approach for each phase, so when this all-important phase of the project comes around everyone is aware of the timescales required, task in hand, their role and what constitutes a pass or fail for implementation.

Implementing any large business change programme is a complex and significant undertaking. The process typically involves several phases, with testing being one of the most critical aspects. Testing ensures that the system meets the business requirements, performs as expected, and integrates as much as possible seamlessly into the organisation's existing IT infrastructure. However, if testing activities like Unit Testing, Integration Testing, System Testing, Regression Testing, User Acceptance Testing (UAT) and Operational Acceptance Testing (OAT including Performance Testing) are not conducted comprehensively and with rigor, the consequences can be severe. It never ceases to amaze me that when time pressures are on and they will be, some Programme Managers consider shortening and/or missing particular testing phase(s). This can be done, and may be lucky, but the risks rise exponentially.

Each type of testing serves a unique purpose, skipping or inadequately performing any of them can have far-reaching impacts on system functionality, user satisfaction, business operations and reputation. This article outlines the purpose of each type of test and the potential consequences and impacts of neglecting each testing phase in the context of a large business change programme implementation.

1. Unit Testing

Purpose:

Unit Testing focuses on testing individual components or modules of the solution. It ensures that each module functions correctly on its own, without being integrated with other parts of the system. This element covers full parameter testing and both positive and negative testing.

Consequences of Not Conducting Unit Testing:

Undetected Coding Errors: This is the phase where individual modules are tested in isolation, but all the parameters are tested, positive and negative testing is completed, and error coding is verified. If this is not rigorously tested, coding errors or defects may go unnoticed. These defects can propagate into larger, more complex issues when the system is fully integrated.

Faulty Business Logic: Inadequate unit testing may result in incorrect implementation of business logic. As some systems are highly customisable, failing to verify that each custom component functions correctly at a granular level can lead to incorrect data processing, calculations, or reporting errors.

Increased Cost and Time: Identifying issues at later stages, such as during System or User Acceptance Testing, becomes more expensive and time-consuming. Fixing issues after integration takes more effort because it involves additional debugging and rework.

Impact: The risk of discovering critical errors late in the project increases massively, which can delay the entire project timeline and inflate costs when errors are detected further through the testing process. Also, if business users encounter unexpected functional defects during later testing phases this will tend to lead reduced confidence and increase frustration in the system. If defects are identified once in production, the costs and timescales of rectification rise exponentially.

2. Integration Testing

Purpose:

Integration Testing is a software testing phase that focuses on verifying the interactions between integrated units or modules. It ensures that these components work together as intended, detecting interface errors and issues related to data flow, communication, and compatibility before moving on to system testing

Consequences of Not Conducting System Testing:

Integration Failures: Without comprehensive integration testing, there's a high risk that modules within the system won't interact correctly or may exchange incorrect data, leading to wrong calculations, errors in reporting and ultimately flawed decision making.

System Failures: Lack of integration testing can cause critical system components to fail when they interact with other components, leading to crashed, poor or incorrect data flows and ultimately down time.

Security vulnerabilities: Unchecked and/or incomplete interactions between modules can introduce security loopholes, making the system an easier target for attacks and therefore increased vulnerabilities exist.

Increased costs and project delays: Errors or bugs which are identified further down the testing process once all the systems are integrated (or worst still, in production), will cost significantly more in both cost and time to rectify, impacting budgets, delivery schedules and customer confidence.

Impact: Integration efficiency issues between the individual modules can lead to significant increase in time taken and computing power used to complete the designed functionality therefore having a major impact on overall system performance. Also, integration issues within the system will lead to breakdowns, making critical business processes, such as inventory management or financial reporting, unreliable. Without detecting and resolving performance issues early on, the system may fail to meet operational demands, resulting in downtime or slow responses, which can significantly impact productivity and customer service.

3. System Testing

Purpose:

System Testing involves verifying that all components within the entire solution works as a cohesive system. It checks that all integrated components function together properly, including interfaces, data flows, and system-wide processes.

Consequences of Not Conducting System Testing:

Integration Failures: Without comprehensive end to end system testing, there's a high risk that different modules, components and systems won't interact correctly. For instance, integration between the asset management module, finance module and supply chain management module could fail, resulting in data synchronisation issues, loss of revenue, etc.

Unidentified System-Wide Defects: Problems that only occur when different systems and modules are working together—such as data being lost or miscommunicated between systems—will go unnoticed. This can cause critical failures in real-time transaction processing.

Bespoke Functionality Failures: System testing can also identify business process or customised functionalities that are not operating as intended, impacting critical business processes Skipping this phase may mean that the system may not operate as the business requires, potentially impacting timescales, costs increases and loss of business confidence due to development rework.

Impact: This is the first opportunity for the complete system / solution to be tested as an integrated system. This phase allows issues between the business process, systems and/or components to be identified early on. Without this phase been conducted it can lead to system integration issues, process bottleneck, data inconsistencies and system breakdowns, potentially impacting critical business processes, such as inventory management or financial reporting, etc.

4. Regression Testing

Purpose:

Regression Testing ensures that new systems, changes, configurations, and/or updates to the system don't negatively affect existing functionalities, and all the systems work in harmony. Every time new code or configurations are added, regression testing checks that the rest of the system and associated interfaces remains stable.

Consequences of Not Conducting Regression Testing:

Unintended Consequences of Changes: Failing to perform regression testing after system updates or configuration changes can result in previously working functionality not working as expected or breaking. For example, a change made in the payroll module might unexpectedly affect other areas like human resources or accounting, causing significant errors and disruptions.

Data Integrity Issues: Inadequate regression testing can lead to data integrity problems, where changes in one module affect data accuracy in another. For instance, a change in tax calculation logic might alter values in financial reporting without any warning having significant financial implications.

Increased Risk of Downtime: If regression testing is skipped, there's a greater risk that unexpected issues will arise after the system goes live. This could result in system downtime as developers scramble to fix problems that should have been caught earlier.

Impact: Not conducting regression testing can result in the loss of trust in the new system's reliability, as new bugs may appear unexpectedly. This can lead to frequent patches or emergency fixes, disrupting business operations and eroding user confidence. Worse still, issues that affect core business functions can have a direct impact on the company's financial performance.

5. User Acceptance Testing (UAT)

Purpose:

UAT ensures that the system meets the business requirements and is fit for use by end-users. It's the phase where business users validate the system's functionality based on real-world scenarios and business processes and business confidence is gained.

Consequences of Not Conducting UAT:

Critical Business Process Failures: Since UAT mimics actual user scenarios, failing to perform it thoroughly can result in critical business processes not being adequately tested. This may lead to operational disruptions once the system is in production, loss of credibility and Business confidence within the system implementation and changes.

Business Requirements Not Met: Without rigorous UAT, the system may not align with the business processes or expectations. Customisations that were intended to improve efficiency or usability might not work as intended, leading to user dissatisfaction.

Low User Adoption: If end-users aren't involved in validating the system, they may find it difficult to use or unsuitable for their needs when it goes live. This can result in resistance to adopting the new system, as users will prefer to stick with legacy processes.

Impact: Without thorough UAT, the new system is likely to face significant usability issues, code errors, and therefore may not meet the needs of the end-users, leading to operational inefficiencies and financial impacts on the project and business as a whole. Business users might resort to manual workarounds, diminishing the value of the new system and delaying the realisation of return on investment. Additionally, poor user adoption can stall the broader digital transformation efforts of the organisation.

6. Operational Acceptance Testing (OAT)

Purpose: OAT ensures that the system is ready for production from an operational perspective, but it is not intended to test system functionality. It tests the new system under real-world conditions, focusing on performance, stability, security, backup, disaster recovery, and support processes.

Note – dependant of the size and scale of the implementation, one or all of the OAT elements may have a dedicated phase e.g. – performance testing

Consequences of Not Conducting OAT:

Infrastructure and Environment Failures: Without OAT, there's no guarantee that the new system will function properly in the live environment. Hardware compatibility, network performance, or even cloud integration may not be adequately tested, leading to system crashes or performance issues in production.

No Backup and Recovery Validation: OAT includes testing of backup and disaster recovery procedures. Skipping this step could mean that the business won't be able to recover critical data in the event of a failure, putting the organisation at risk of data loss or extended downtime.

Operational Support Gaps: If operational readiness isn't tested, support teams may be unprepared to handle issues that arise once the system is live. This can lead to delays in resolving incidents and prolong system outages.

Impact: The failure to perform OAT can result in a shaky go-live, with the new system prone to performance issues, security vulnerabilities, and inadequate operational support. The organisation could face prolonged outages, financial losses due to downtime, and reputational damage if the system fails to deliver during critical periods, such as month-end financial closing or a major promotional event.

My final thoughts; Testing phases typically come towards the end of a phase for a programme and/or element within the programme. Therefore, any slippage of the previous phases tends to put pressure on subsequent phases which inevitably flows into testing timescales. As outlined above, it is essential that testing timescales should be protected. It should also be noted that cost to fix and implement any errors within systems goes up significantly as we go through the testing process, but more importantly those costs go up exponentially if problem is not identified until it goes in production systems. Therefore, it is imperative that an organisation implementing a large business change programme, should ensure that these testing phases are carried out with rigor and comprehensiveness, essential to avoid costly rework, minimise downtime and guarantee a smooth and successful go live. Investing in a thorough testing strategy, associated approach and plans helps mitigate risks and lays the foundation for long-term operational success and user adoption.

I hope you have found this article useful, an effective Testing approach starts with a 'fit for purpose' Testing Strategy, Ascea can help you develop that. If you would like to get in touch, please do so at experiencematters@ascea.co.uk

Terry Jefferson – Joint Founder, Ascea