SOME PRACTICAL CONSIDERATIONS AND A METHODOLOGY FOR TESTING AUTONOMOUS SYSTEM INTEGRATIONS

Safiullah Faizullah

Rutgers University, Piscataway, NJ, USA
safi.research@gmail.com

ABSTRACT

With interconnectivity between IT Service Providers and their customers and partners growing, fueled by proliferation of IT Services Outsourcing, with some providers gaining leading positions in marketplace today, challenges are faced by teams who are tasked to deliver integration projects with much desired efficiencies both in cost and schedule. Such integrations are growing both in volume and complexity. Integrations between different autonomous systems such as workflow systems of the providers and their customers are an important element of this emerging paradigm. In this paper we present an efficient model to implement such interfaces between autonomous workflow systems with close attention given to major phases of these projects, from requirement gathering/analysis, to configuration/coding, to validation/verification, several levels of testing and finally deployment. By deploying a comprehensive strategy and implementing it in a real corporate environment, a 10%-20% reduction in cost and schedule year over year was achieved for past several years primarily by improving testing techniques and detecting bugs earlier in the development life-cycle. Some practical considerations are outlined in addition to detailing the strategy for testing the autonomous system integrations domain.

KEY WORDS

Software testing, software validation/verification, unit testing, integration testing, test oracles, empirical studies, regression testing.

1. INTRODUCTION AND BACKGROUND

Large corporations and governmental bodies rely on IT Services Providers to manage their infrastructures, data centers, applications, workflow system, and sometimes their entre IT organization. Distributed nature of IT infrastructure management is emerging as complex and challenging service where it presents providers with unique opportunities and huge financial rewards if such management is executed efficiently. This model necessitates, among others activities, the need for connecting customers' and their partners' autonomous systems to those that Service Providers deploy to manage their infrastructure. With major Service Providers organizations gaining significant market share, such interconnectivity is part of most outsourcing contracts. In particular, interconnecting Service Providers' workflow systems such as Remedy, Service Manager, Service Now (Workflow Systems), with customers' workflow systems via Case/Service Exchanges (interfaces), see Figure 1, is a growing segment and often a central piece of larger contracts. These interfaces are highly visible, due to dependencies of other projects, thereby enabling Service Providers to service the overall contracts and as such it could affect customer satisfaction. Another important point is that with many of these projects on the ever-growing pipeline, reusability is both financially desirable as well as customer satisfaction could be positively enforced with reduced schedule. As such the deployment of these projects should be handled with care from the project initiation to release to production. Such projects go through multiple phases (refer to Figure 2):

1. Start requirement analysis phase, including fields mapping, data translation, protocol, detailing the product specification, defining validation and verification mechanism, test/use cases;
2. Followed by technical steps where protocols definition and fields/data mappings of Service Providers' work flow system such as Workflow System to those of the customer or their partners is conducted;
3. Followed by configuration of interface with coding and deployment;
4. Next test scripts definition and agreement, test oracles, defining adequacy and coverage criteria;
5. Multiple level of testing, utilizing agreed scripts, both by Service Providers and their customers/partners in their respective environments as well as those done together to test the integration of the systems; Please note that integration testing as a step in the overall testing strategy in comparisons to testing autonomous system integrations which must cover overall aspects of testing for this domain. Please keep this distinction in mind when reading this paper.
6. Followed by User Acceptance Testing where the solution is scrutinized by the business teams to ensure the interfaces are behaving per process and business cases that were agreed during requirement phase;
7. Move to production and go-live;
8. Under maintenance and active management.

Loosely speaking, requirement analysis, configuration/coding and testing form the major steps in such interconnections. Therefore, it is important to conduct these steps in standard ways as much as possible. Innovative ways to reuse and reduce waste during the requirement analysis phase are required which are achieved by implementing requirements/definition documents that describe standards and incorporate use cases.

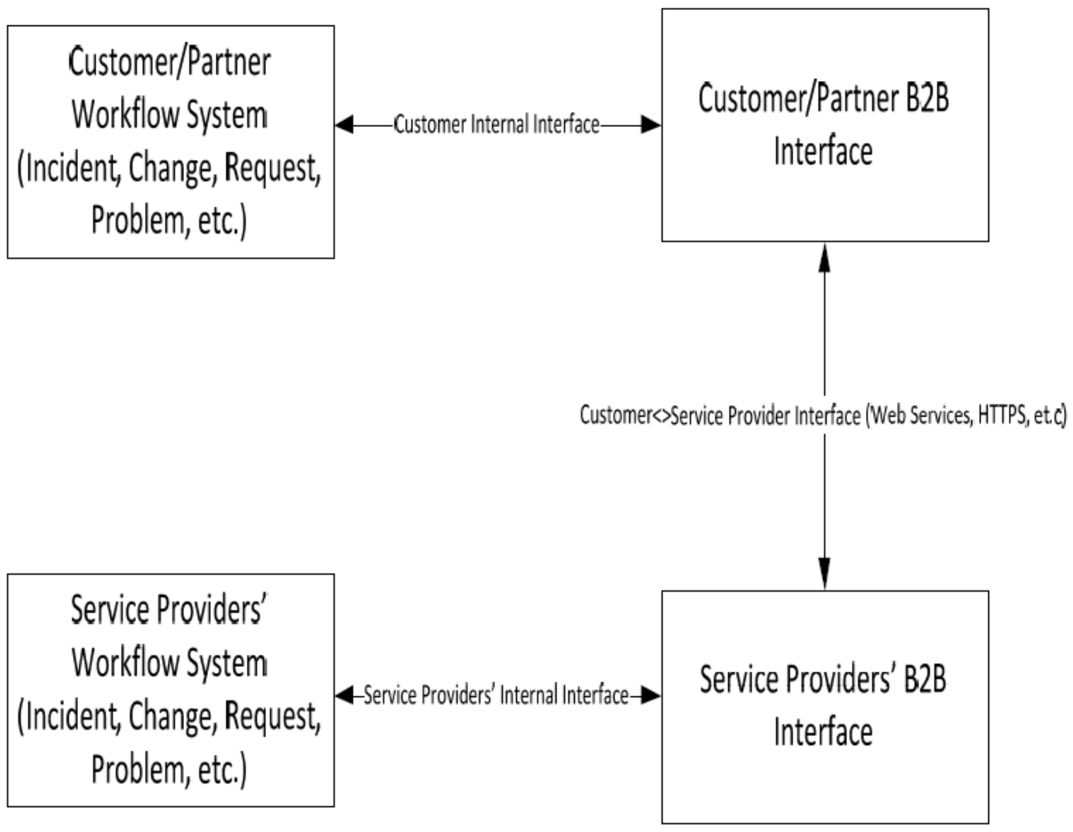
Customer/Partner Workflow System (Incident, Change, Request, Problem, etc.)
Customer Internal Interface
Customer/Partner B2B Interface
Customer<>Service Provider Interface (Web Services, HTTPS, et.c)
Service Providers' Workflow System (Incident, Change, Request, Problem, etc.)
Service Providers' Internal Interface
Service Providers' B2B Interface

Figure 1: Workflow-to-Workflow Interconnectivity Capability for Different ITIL Modules

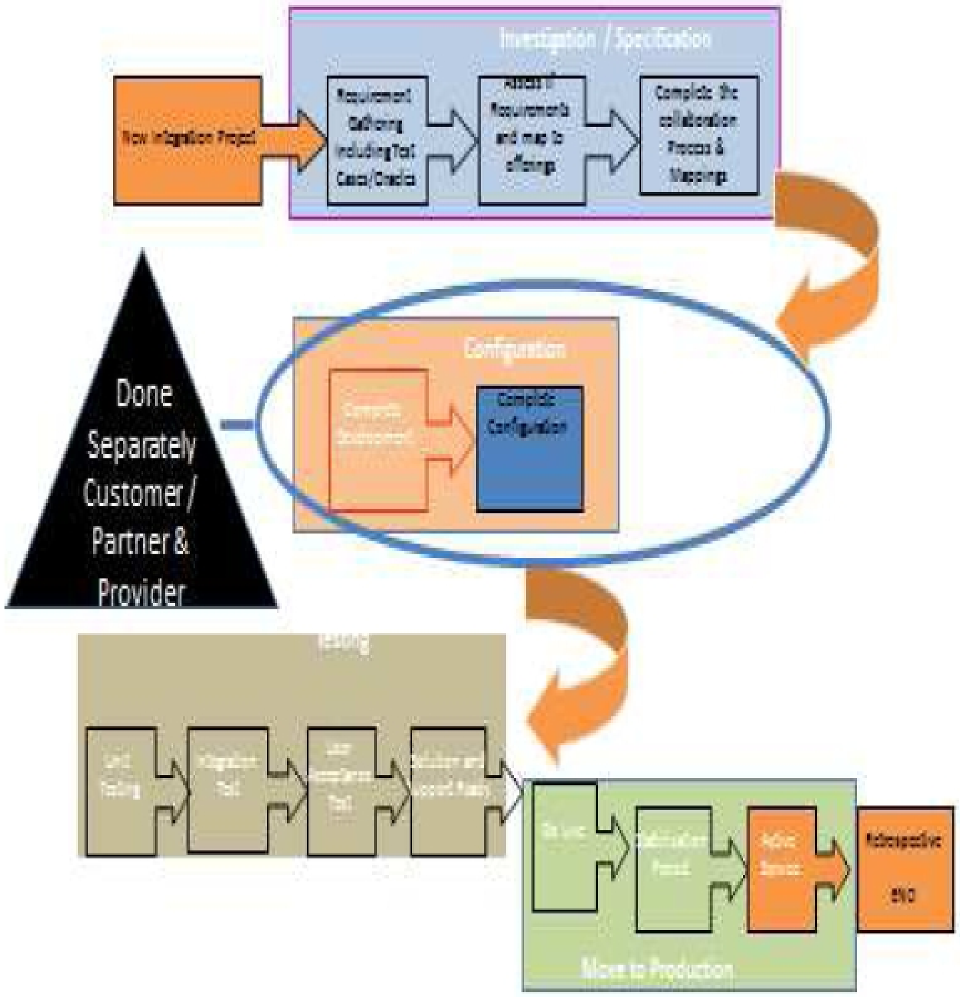
Investigation / Specification
New Integration Project
Requirement Gathering including Test Cases/Oracles
Assess of Requirements and map to offerings
Complete the collaboration Process & Mappings
Done Separately Customer / Partner & Provider
Configuration
Complete Configuration
Testing
Unit Testing
Integration Test
User Acceptance Test
Move to Production
Retrospective
END

Figure 2: Autonomously System Integration Project Flow

Aligning new projects to the definitions and use cases of standards enable reuse and improve efficiencies and hence lower cost and shorten projects schedule. Configurations/coding based on standard mappings/data translation can be streamlined and hence done efficiently. Utilizing an enhanced approach to conduct configurations per standards and in close observation of use cases can be cost effective, and in case the requirements are deemed achievable by the standard definitions and mappings, the reduction in cost and schedule will be significant.

The testing techniques needed for these projects require a significant amount of manual, non-execution based, efforts which typically result in inadequately tested scenarios unless creative techniques are employed. The multiple iterative testing strategies, involving both non-execution based as well as execution based testing, are meant to reduce the required resources. We deploy both testing to specification (black-box testing) and testing to code (white box testing) [1]. Moreover, during iterative development, if testing is not incorporated in the overall software life cycle, developers' waste time fixing bugs that they encounter in later development cycles; these bugs could have been detected earlier if the code had been tested iteratively. The nature of these developments requires testing techniques to quickly test each increment of the configuration during code development. As such a closely coupled approach is needed whereby requirements are tied to use cases which are tested thoroughly and issues are discovered early in the testing cycle. The discovery of bugs, product defects or the need for enhancements at later stages are exponentially more expensive and thereby a point of focus to be avoided or reduced significantly. The iterative nature of the development can make regression testing more efficient if conducted carefully. Another advantage of such test strategy is that developers (or testers) can use capture/replay tools such as WinRunner, to perform stress/performance testing when needed for very heavy and active interfaces [1-8].

2. TESTING STRATEGY

As the most significant and the dominant success criterion for any software projects/products in the industry is the quality [9], this against the ever intense demand for fast turnover with much wanted cost efficiencies, requires enhanced development paradigms with testing taking center stage. It is not surprising then to note that the most time consuming and very crucial step in any software development, including integrations, is testing. In general, software testing is a process of providing inputs to software that is being tested and evaluating the produced results. The mechanism used to generate expected results is called an oracle. There are several approaches that we can adopt to generate, capture, and compare test results. Some of the common ways are:

A. Manual testing/verification of results by human- manual oracle (non-execution based);
B. Testing/Verifying specific values for known responses;
C. Testing/Verifying the consistency of generated values and end points;
D. Interface simulator to produce results;
E. Utilizing sampling of values to compare with independently generated expected results;
F. Automating the testing/verification for regression.

For testing autonomous system integrations, most of the above steps are utilized. These steps enhance the quality of the integrations and improve both schedule (Figure 3) and cost (Figure 4.) All of the tests utilized A through C, and by adding steps, D, E, F, DE, DF, EF, DEF we see improvements in both schedule and cost.

Step A is an essential part of the autonomous systems integrations as each end workflow system has different fields and data structures that need to be mapped to its counterpart and as such it requires human investigation during unit, integration, and UAT. On the other hand, this step is very laborious and hence contributes the most to both schedule duration and cost (see Figure 5 and Figure 6.) Even though human testers in step A will certainly utilize some tools to enhance schedule/cost, currently manual interaction is its dominant aspect as is evident by comparing Figure 3 and Figure 5 for schedule durations as well as Figure 4 and Figure 6 for cost comparisons. Automating some aspects of this step is highly desirable [6, 10-11].

The computing speed has grown very rapidly and that coupled with low cost of memory, test cases can generate very large amounts of data. This makes the oracle data also massive which is needed for comparison. Integration work requires data comparison which must be incorporated into test cases. This requires that we add to test cases the error handling, capturing error results, as well as reporting differences. The nature of integration necessitates dependency on human oracles to verify test results, this can only be fruitful and efficient where the testers know the details of the code, and they are expected to know when the application misbehaves. Manual testing with tester as oracles has disadvantages that include increased costs and longer schedule durations and as such certain cycles of testing (in particular integration and regression testing) are automated as much as possible. New techniques and strategies are needed to reduce the test suites as well as detection of bugs/defects to be shifted to early test cycles (unit testing) in autonomous system integration projects. We have had some successes in this area by bringing the learning from previous work and adjusting our overall testing strategy, see Fig. 3-Fig. 6 for this trend.

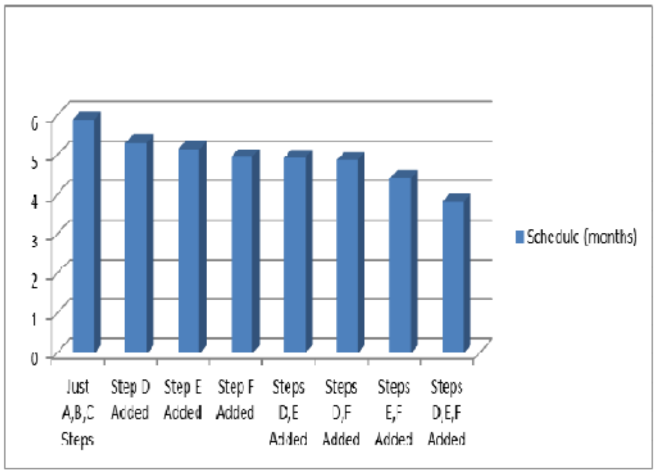

Figure 3: Averaged Schedule for 5 Integrations

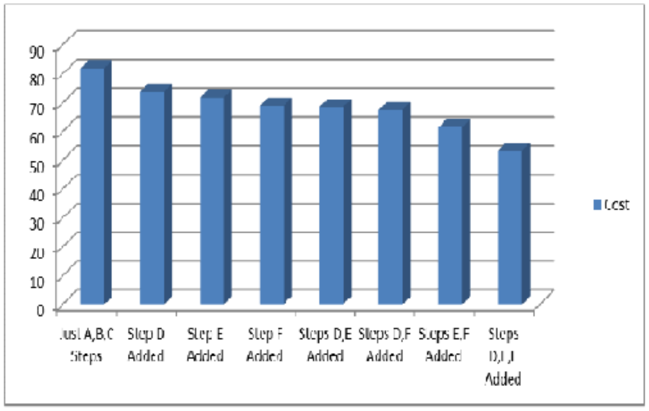

Figure 4: Averaged Cost for 5 Integrations

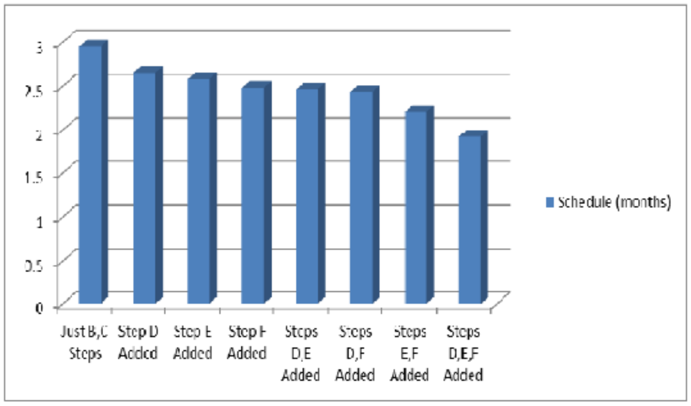

Figure 5: Averaged Schedule for 5 Integrations without Step A

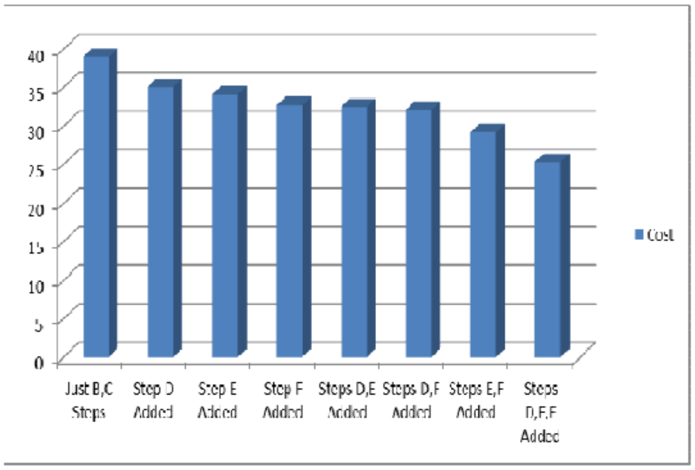

Figure 6: Averaged Cost for 5 Integrations without Step A

3. CONCLUSION AND FUTURE WORK

Interconnecting Service Providers' workflow systems such as Remedy, Service Manager, Service Now (Workflow Systems), with customers' workflow systems via integration interfaces is a growing segment and often a central piece of larger contracts. These interfaces are highly visible, due to dependencies of other projects to work smoothly, enabling Service Providers to service the contract and as such it could affect customer satisfaction positively (or unfortunately negatively if not done efficiently.) These projects need to be done in cost effective way with reasonable schedules. Testing is an essential step in these interconnections and efficient methods need to be deployed to achieve the desired results. We have shown a strategy that is deployed by a major Service Provider where significant savings (~17%-27.5% schedule and ~20%-25% cost reductions) were achieved by employing refined steps (see Figure 3 and Figure 4.)

Another factor for these savings was that by ensuring very few bugs/defects were undetected in early cycles of these projects and thereby no costly retesting as a result of discovering bugs/defects and it removal. As future work, we need to consider automating the testing of certain commonly used use cases [6], particularly during early stages (step A) of our iterative testing strategy, so that we can further reduce the testing cycle and eliminate human related errors and costs. Deploying capture/replay tools such as WinRunner [8] to perform regression testing for very heavy and highly active interfaces. integration of autonomous systems involve many products (workflow systems, network, business processes, etc.) and as such any changes in any of these components will necessitate changes in these interfaces, conduction crucial regression testing is crucial. In addition, keeping an active eye on the environment capacity is very crucial as production outages are very expensive. With growth of autonomous system integrations implementations, this is even more urgent aspect than individual integration projects.

REFERENCES

[1] B. Beizer, Software Testing Techniques, (Van Nostrand Reinhold, New York, 2nd edition, 1990).

[2] M. Bozkurt, M. Harman, and Y. Hassoun., “Testing web services: a survey”, Technical Report TR-10-01, Department of Computer Science, King’s College London, April 2010

[3] W. T. Tsai, et al, “Extending WSDL to facilitate Web services testing”, Proceedings of 7th IEEE HASE, pp. 171- 172, 2002.

[4] A.K. Onoma, W.T. Tsai, M. Poonawala, and H. Suganuma, “Regression Testing in an Industrial Environment”, Communications of the ACM, Vol. 41, No. 5, May 1998, pp. 81-86.

[5] R. Paul, “End-to-End Integration Testing: Evaluating Software Quality in a Complex System”, Proc. of Assurance System Conference, Tokyo, Japan, 2001, pp. 1-12.

[6] J. Robbins, Debugging Applications. Microsoft Press, 2000.

[7] B. Marick, “When Should a Test Be Automated?” Proc. 11th Int’l Software/Internet Quality Week, May 1998.

[8] “Mercury Interactive WinRunner,” 2003, http://www.mercuryinteractive.com/products/winrunner.

[9] L. Osterweil et al., “Strategic directions in software Quality”, ACM Computing Surveys, (4):738-750, December 1996.

[10] S. Faizullah, “Rapidly Evolving Research Area- Testing Software Testing”, Invited Talk at COMSATS Institute of Information Technology (CIIT), Abbottabad, Pakistan, 2009.

[11] S. Faizullah, “Trends in Testing Software Testing- A Case in Focus”, Talk at Frontiers in Technology (FIT) 2010, Islamabad, Pakistan, Dec 21-23, 2010.

AUTHORS

Safi Faizullah received his Ph.D. in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2002. He also received MS and M. Phil. Degrees in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2000 and 2001, respectively. Dr. Faizullah also earned his BS and MS degrees in Information and Computer Science from KFUPM, Dhahran, KSA in 1991 and 1994, respectively. His research interests are in computer networks, mobile computing, wireless networks, distributed and enterprise systems. He has authored over twenty refereed journals and conference papers. Dr. Faizullah works for Hewlett-Packard and he is a Visiting Scholar/Adjunct Professor of Computer Science at Rutgers University. He is a member of IEEE, SCIEI, PMI and ACM