2014 IEEE International Conference on Software Testing, Verification, and Validation Workshops

Practical Challenges in Test Environment Management

Rudolf Ramler and Johannes Gmeiner
Software Competence Center Hagenberg GmbH
Softwarepark 21, A-4232 Hagenberg, Austria
{rudolf.ramler, johannes.gmeiner}@scch.at

***Abstract*—Test environments are a critical prerequisite for successful test automation. The definition, setup and maintenance of a test environment are a common source for pitfalls and a major cost driver in test automation. This paper describes the practical challenges involved in managing test environments and discusses solutions proposed in the literature and by academic research. The identified challenges involve the four areas classical test management tasks, development and evolution, configuration issues as well as automation for test environments.**

Keywords—test environment; testbed; test automation; test management

I. INTRODUCTION

Test environments are a critical part of a project's software test infrastructure. Appropriate test environments are a prerequisite for effective test automation and can show a substantial impact on testability, defect detection, and overall test costs. Evidence about the practical importance of test environments is provided in numerous real-life accounts and lessons learned collected from the field.

In *Experiences of Test Automation* [1], Allen and Newman summarize the insights from a successful test automation project: "Developing a sound automation framework was one of the keys to our success, but ... What really clinched the success of our automation project was the creation of a test environment that was automated, robust, predictable, and synchronized with the automation framework." Page et al. describe different approaches for organizing the test environment in *How We Test Software at Microsoft* [2] suggesting fully automated deployments. In this context, "getting the test environment just right is a critical element".

A comprehensive list of common pitfall in automated testing has been compiled by Persson and Yilmaztürk in their report on *Establishing Automated Regression Testing at ABB* [3]. This list includes, for example, the pitfall "insufficient configuration management for test environment and testware". Common problems related to the design of the test environment have also been reported by Berner et al. in *Observations and Lessons Learned from Automated Testing* [4]: "(1) manual installation and configuration procedures for the system under test, (2) lack of access to essential infrastructure like the configuration management system, and (3) an automation where the test execution is not observable (enough) to the tester." In *Technical Debt in Test Automation* [5], Wiklund et al. studied test automation issue in a telecommunication subsystem. They found that the "test facility infrastructure is not transparent" and, thus, moving the test environment from local workstations to remote servers altered the timing, which in turn caused the test execution system to become instable and unpredictable.

In test automation projects for machinery software we made similar observations. The tested system consists of embedded software for controlling the machine hardware as well as a high-level software system providing a graphical user interface (GUI) for human machine interaction and integration with manufacturing execution systems (MES). For test automation, repeatedly setting up the test environment to resemble the numerous variants of hardware and software configurations became a limiting factor [6]. Furthermore, test environments including physical hardware such as machinery, robots and auxiliary devices are a major cost driver; the price tag of a fully-equipped machine is in the range of several hundred thousand Euros. Thus, a considerable capital is tied-up in the test environments.

This paper describes the practical challenges involved in managing test environments and discusses solutions proposed in the literature and by academic research.

II. TEST ENVIRONMENT

The term *test environment* (*testbed*) is defined as "an environment containing hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test" [7]. It is a part of the test infrastructure, which includes all organizational artefacts needed for testing, and it is closely related to the development of testware, which includes the development of setup procedures for the test environment etc.

We distinguish following core parts of the environment for testing a software system: (a) The *Operational Environment* forming the basis for executing the system under test. It contains the software stack not part of the system under test (e.g., application server, operating system) and the underlying hardware (e.g., IPC, I/O controller). (b) *Neighboring Hardware Systems* required for testing (e.g., machinery, robots) either in form of the real hardware or as simulators. (c) *Neighboring Software Systems* interacting with the system

978-0-7695-5194-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ICSTW.2014.41

under test (e.g., MIS, CAD). Due to their complexity and availability, dummies and simulators are used for testing instead of the real instance of these systems.

III. Challenges Experienced

This section describes the challenges related to test environments that we have experienced in conducting and automating testing for machinery software.

A. Management Challenge

The classical challenge is making the right test environments available on time and on budget. Thus, it is considered mainly a task of test management. It starts with defining the requirements for the test environment. "A test environment that is too different from ... production can miss many bugs. A test environment that is so exactly like production that it is production can cost too much to build" [2]. When test environments are developed or operated by third parties, management tasks also include procurement, acceptance testing and supervising maintenance. Test process improvement (TPI) frameworks [8] are typically suggested to address the related challenges.

B. Development and Evolution Challenge

In our case, the machinery and its hardware components are developed by the same company in parallel to the software system. Due to this co-engineering approach, the test environment used in the software life-cycle has to closely reflect the evolution of the machinery. Development and testing start on hardware prototypes and simulators for machine components and proceed until acceptance on a number of fully-fledged machinery setups in different configurations. Finally, adaptations and extensions necessary to meet individual customer requirements are tested as part of the commissioning process. Test environments for each of the hardware configuration have to be maintained throughout the life-span of the delivered machinery. Hardware-in-the-Loop (HiL) simulation [9] and Software-in-the-Loop testing based on models [10] are typical solutions.

C. Configuration Challenge

In our previous work on automated regression testing for a software product line [6], we discussed the issue of providing suitable test cases for multiple product variants. The high number of potential product variants triggers a further challenge for managing test environments. In our case, software product variants often depend on specific hardware properties. For example, software modules for controlling robots can only be executed (and tested) on machines including a robot. As a response to this challenge, Magro et al. [11] suggest using the software product line approach to define domain specific validation environments also for testing.

D. Automation Challenge

Even when a high number of appropriate definitions for test environments can be produced, effective test automation requires that the test environment can be setup automatically. In addition, due to the risk of test cases altering the state of the test environment, means to reset the test environment to its initial state have to be provided. Otherwise side effects of one test case may influence the result of others when the execution order is changed. With the advent of the DevOps movement [12], new paradigms have also been proposed for the administrative tasks that concern operations more than development. An example is the idea to consider infrastructure as code [13], which is supported by tools like Puppet (puppetlabs.com) and Chef (www.getchef.com/chef).

IV. Conclusions and Future Work

Test environments are a critical prerequisite for successful test automation. However, several challenges affect the involved tasks in management, development and evolution, configuration and automation of test environments. What makes these challenges hard to tackle – but at the same time interesting – is that they are intermingled with other activities and roles in testing and development, and that they extend into different application areas and research topics.

Thus, promising solutions are available from research or have been suggested by new movements in software engineering and quality management. As part of our future work, we plan a case study on multiple industry projects to explore the improvement potential of these solutions related to the identified challenges.

References

[1] D. Graham, and M. Fewster, "Experiences of Test Automation: Case Studies of Software Test Automation," Addison-Wesley, 2012.

[2] A.C. Page, K. Johnston, and B.J. Rollison, "How We Test Software at Microsoft," Microsoft Press, 2009.

[3] C. Persson, and N. Yilmazturk, "Establishment of Automated Regression Testing at ABB: Industrial Experience Report on 'Avoiding the Pitfalls'," Proc. 19th IEEE Conf. on Automated Software Engineering (ASE '04), 2004.

[4] S. Berner, R. Weber, and R. Keller, "Observations and Lessons Learned from Automated Testing," Proc. 27th Int. Conf. on Software Engineering (ICSE '05), 2005.

[5] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Technical Debt in Test Automation," Proc IEEE Fifth Int. Conf. on Software Testing, Verification and Validation (ICST '12), 2012.

[6] R. Ramler, and W. Putschögl, "Reusing Automated Regression Tests for Multiple Variants of a Software Product Line," TAIC-PART 2013, Proc., IEEE Sixth Int. Conf. on Software Testing, Verification and Validation Workshops (ICSTW '13), 2013.

[7] IEEE Standards Board, Standard Glossary of Software Engineering Terminology, IEEE Std 610.12, 1990.

[8] H. Heiskanen, M. Maunumaa, and M. Katara, "A Test Process Improvement Model for Automated Test Generation," Proc 13th Product-Focused Software Process Improvement (PROFES'12), 2012.

[9] M. Schlager, "Hardware-In-The-Loop Simulation," VDM Verlag, 2008.

[10] A. Bayha, F. Grüneis, and B. Schätz, "Model-based software in-the-loop-test of autonomous systems," Proc. Symposium on Theory of Modeling and Simulation - DEVS Integrative M&S Symposium (TMS/DEVS '12), 2012.

[11] B. Magro, J. Garbajosa, and J, Perez, "A Software Product Line Definition for Validation Environments," Proc. 12th Int. Software Product Line Conference (SPLC '08), 2008.

[12] J. Roche. "Adopting DevOps Practices in Quality Assurance," Commun. ACM 56, 11 (November 2013), pp. 38-43.

[13] D. Spinellis, "Don't Install Software by Hand," IEEE Softw. 29, 4 (July 2012), pp. 86-87.