RDTA – Repository Driven Test Automation

A new look into reuse of test automation artifacts

Dani Almog, Hadas Schwartz Chassidim, and Shlomo Mark
Dept. of Software Engineering
Sami Shamoon College
Beer Sheva, Israel
e-mail: Almog.dani@gmail.com, hadasch@sce.ac.il, marks@sce.ac.il

Yaron Tsubery
R&D Operations
Enghouse Interactive
9th Nehar Prat St., Giva'at-Ze'ev, Israel
yaron.tsubery@gmail.com

Abstract— Repository Driven Test Automation (RDTA) is an approach to the buildup process of test automation infrastructure which proposes reuse of testing artifacts as a fundamental principle for the creation of test automation. Our research was motivated by a two-fold inquiry: Can testing automation artifacts be reused? If so, how? These inquiries led us to a new concept for the formulation of test automation. The term software repository here refers to a storage location from which software packages or artifacts may be retrieved for reuse in other systems or software products, preferably - as is. This conceptual paper explores different aspects of the reuse of software test automation artifacts and elaborates on several practical implications and changes that arise from the implementation of this new paradigm in a software development organization.

Keywords-testing; test automation; software reuse; repository driven automation.

I. INTRODUCTION

Testing is perhaps the most expensive task in a software project. Large portions of testing costs are derived from the need to assure that none of the newly introduced changes in the code have damaged previous quality – testing for regression is a repetitive activity. Regression testing is an expensive activity that can account for a large proportion of the software maintenance budget [1]. Software engineers add tests into test suites as software evolves, and by this increase the test suite size, the revalidation of the software but, also the testing costs. Special techniques to reduce the regression tests costs by selecting, prioritizing and reducing the number of regression tests and costs, have been proposed [1,2]. However, it can be expensive to employ these techniques and therefore it might not reduce the overall regression testing costs. A survey of practitioners [2] shows that the main benefits of test automation are: reusability, repeatability and effort saved in test executions. Automation can be applied to parts of the testing processes by entrusting repetitive tasks to a test automation system. The main motivation of RDTA is to reduce the overall expenses and efforts in the implementation of test automation by addressing test automation artifacts and the creation process itself [3]. Today, many commercial and open source tools are used for test automation. Large portions of these tools are highly specialized solutions for specific aspects of testing, are focused on different technologies, or are based on particular test paradigms. There is a large variety of specialized test tools for test case generation, test management, test execution, and so forth. There is limited support for combining the numerous specialized tools in an integrated solution except for the provision of technical interfaces between single tools.

The objective of our work is the development of test automation infrastructure rooted in the concept of reusing testing artifacts. In Section II, we briefly revisit the general reuse concepts, including some heuristics [4], and elaborating on some needed architectures, and testing artifacts and other dimensions of test automation. RDTA is introduced in Section III, discussing what, where and how to store the different artifacts. We conclude with conceptual insights into the implications of RDTA in today's modern software development arena (e.g., Unit test, agile, integration, Service-Oriented Architecture (SOA)).

II. BACKGROUND: REUSE OF ARTIFACTS

Analyzing our day-to-day testing activities, we may ask: how much of every action, operation, thinking, doing – is actually uniquely new? When attempting to explain the nature of the reusability concept, we may be challenged by the argument that this has all been done before and, therefore, that there is nothing new to contribute in this field. These notions are almost right: most new contribution stems from context and interpretation. For example, when designing a new test case for a certain application, memory and past experience are utilized to rearrange old knowledge into a new pattern to create a new test case that ought to answer the new aspects we are testing. So from a conceptual standpoint, we are reusing. In this paper we will examine how much reuse is done with regard to testing artifacts. In addition, we review the extent to which we are aware of the

 ISBN: 978-1-61208-441-1

reusable nature of our work when designing a testing artifact. Our day-to-day manual testing work flow is built out of |context| –> |concept| –> |build| –> |use|. We will also review how much of a software engineer's attention/awareness is focused on the issue of reuse [5].

The reuse of artifacts is usually derived from the desire to take advantage of previously developed components and capabilities. In previous scholarship [6], a distinction was made between Development with Reuse (DWR), which focuses on benefits gained from the utilization of reusable resources, and Development for Reuse (DFR), which aims at the creation of reusable products for the benefit of future usage. A generalized reuse model for system development was formulated, suggesting a future quantitative evaluation of reuse in a comprehensive manner [6].

A. Reuse Heuristics

Fortue and Valerdi [4] Addressing the topic of reuse from a systems engineering perspective, a generalized framework for the reuse of systems engineering products has been proposed. This approach is based on reuse heuristics (the following is a partial list selected from the original study) [7]:

- Heuristic a: Reuse is not free, upfront investment is required.
- Heuristic b: Reuse should be planed from the conceptualization phase of programs.
- Heuristic c: Most project related products can be reused.
- Heuristic d: Reuse, in large part, is also an organizational issue.
- Heuristic e: Higher reuse opportunities exist when there is a match between the diversity and volatility of a product line and its associated supply chain.
- Heuristic f: Bottom-up (individual elements where make or buy decisions are made) and top-down (where product line reuse is made) reuse require fundamentally different strategies.
- Heuristic g: Reuse applicability is often time dependent.
- Heuristic h: The economic benefits of reuse can be described in terms of either improvement (in quality, risk identification) or reduction (of defects, cost/effort, and time to market).

The ability to recompose reusable parts is an important requirement for reuse [8]. Anticipating future reuse scenarios make reusable parts easier to compose. Khusidman and Bridgeland [9] presented a framework of reuse and cloning techniques in software development. This work analyzed different aspects of reuse and cloning by utilizing a classification framework to define a matrix of reuse scenarios aimed at efficient reuse. A distinction may be made between "formal" reuse of object code that does not require any customization, and the "opportunistic" "cut-and-paste" reuse achieved by using and modifying fragments of existing solutions [9]. In the following sections, we will attempt to generalize a reuse framework and apply its principles to test automation.

B. Systems Reuse Framework

It has been said that "reuse can increase your productivity by nearly half if you avoid the common pitfalls that derail many reuse programs" [10]. This idea was made clear from the analysis of the outcome of trends in Source Lines of Code (SLOC) of Department of Defense (DoD) software and DoD cost in dollars per SLOC between 1950 and 2000 [10].

However, reuse in software development and testing may present some abuse dangers, such as the propagation of errors in subsequent versions of the software [11]. Lengthy research on reuse of a test case in a safety critical system (for a heart pacemaker) [12] concluded that, conceptually, this approach to reuse is simple, but to implement it in a real project with hundreds of thousands of lines of code, recognizing the commonalities among the test cases, and implementing a mechanism for systematic reuse, is a huge task. Applying reuse techniques at the testing stage of a real project that involved the development of a cardiac rhythm management system led to significantly reduced efforts required to test systems. More recent studies relates reusability to Software Product Line Testing (SPLT) [13] [14] [15]. The strategy of reuse of core assets in SPLT can reduce software testing efforts during development, improve software quality, and potentially decrease the time-to-market of products and services.

C. Reuse of Testing Artifacts

Tiwari and Goel [16], the authors of a wide survey of the literature about the reduction of testing effort through reuse have argued that although there are many systematic studies that deal with quality assurance techniques, virtually no literature or survey exists on reuse-oriented testing approaches. RDTA deals with the reuse of testing automation artifacts using a comprehensive multi-level reuse approach.

III. RDTA: A NOVEL APPROACH

In this section, we present our contribution to the reuse classification framework by laying out the organizational structure for the different levels, types and candidates of storage repositories. The classification system presented here is preliminary and may be further expanded and adapted to different technologies and software development infrastructures.

Even before we consider test automation, we are obligated to store and maintain the testing artifacts. In addition to the Gupta test repository classifications [17], previous research and papers provide a useful resource for testing item classifications, for example, verification items [18]. Complimentary to the Gupta classifications, and in accordance to the research of most of experts in the test

automation industry, we propose this preliminary list of subjects be stored:

- Requirements Repository (a)
- Business Story Repository (b)
- Test Cases Storage (c)
- Basic Operational Element Repository (d)
- Unit Test Repository (e)
- Testing Business Element Repository (f)

Other repositories may be introduced as the result of this research project. All these repositories support reuse when transmitted to, and prorogated among, organizations and teams.

A. *Business and Testing Requirements*

The RDTA approach suggests the requirement for a source link for most testing artifacts. It RDTA samples a subset of configurations to be tested based on environment modeling, requirement analysis and systematic traceability.

RDTA distinguishes between higher Business requirements and their breakdown into functional requirements and nonfunctional requirements.

Therefore, it is imperative that there be a depository where the entirety of the testing requirements are stored, maintained and controlled. Additionally, today many test management tools contain their own storage of test requirements and are linked and traceable to the software requirements as well as to the rest of the testing artifacts.

B. *Business Story Repository*

Derived directly from the store of software testing requirements is a depository of testing business stories. These should be as tightly aggregated as possible. Different aggregation levels may be represented in a repository for these testing stories or business story fragments. For example, "The customer should be able to access the application from most popular interfaces (mobile, pc, remote interface etc.) using a login procedure".

C. *Test Cases Storage*

The test cases repository should derive from the test business stories depository. Please note that test cases are very much application/functionality oriented and therefore require storage in different hierarchies that allow for different affiliations or relationships to be exposed and identified. Figure 1 presents a possible traceability matrix that demonstrates the need for documentation as well as management and control at all items during the testing/fixing operation. Each column presents repository categories containing other testing artifacts. The arrows hint to a possible dependency between the elements.

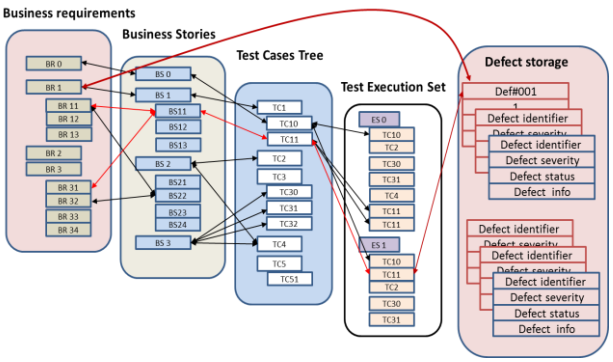

Figure 1. suggested test coverage metrix.

These types of coverage matrices enable tractability [15], and may help may reveal the importance of keeping track of, and documenting, all business and testing artifacts.

D. *Basic Operational Element Repository*

In order to facilitate test automation needs, we must be able to execute and operate all developed applications under conditions of control and isolation. This can be performed during the development phase or in an integration workplace until installation. RDTA divides these repositories into two categories:

1) Operational infrastructure, architectural foundation related storage, and application.

2) Business related storage.

The reusable quality of the items stems from the similarity in the basic application of the actual business behavior in the software.

Each of these artifacts may be used, operated, stored, maintained and manipulated during the testing project. More items may be added and specifically modified. The use of these artifacts is limited by resource constraints and time horizons.

E. *Unit Test Repository*

In order to maintain productive reuse of unit test artifacts, isolated and single purpose (used mostly by developers) unit tests need to be transformed into integrated parts of reusable testing artifacts that are used by all levels of development and quality assurance teams [19].

F. *Testing Business Element Repository*

The need for the reuse of the same generic test case as part of a project scenario that has a different categorical affiliation can be satisfied in most of the existing testing

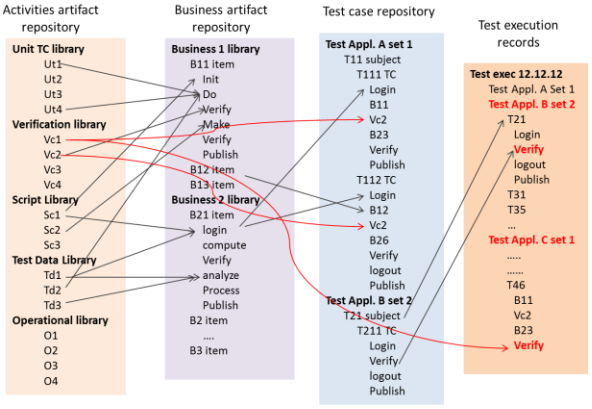

Figure 2. Principal RDTA testing repositories build up

tools by duplicating the same formation and storing it separately. The RDTA approach will store another level of artifacts that relate to the test case business context (see Figure 2).

To facilitate easy access and usage of reusable testing artifacts, the RDTA approach mandates adding another merged level: one that stores, uses and maintains another practical set of testing items. Business artifacts may be related to each element (or object) of the testing artifacts that can be treated as a business portion (as opposed to technical, architectural or other such element).

RDTA will recommend storing and maintaining the actual full context testing scripts so that during subsequent use, the user will have full control of all operational and functional aspects to be tested.

G. Maintaining the Integrity of the Specifications

The RTDA approach suggests a framework where each of the elements is categorized as a service – so it can be recalled and operated independently during the progression of the testing levels. Such a complex, interconnected, and affiliated storage system must be formulated in a very practical manner. Therefore, how and where to store objects are critical issues.

H. How to Store Repositories

Reflecting on the operational practical needs for the storage requirements, the following list of storage requirements has yet to be fully researched and evaluated:

- Easy & efficient storage & retrieval (Ease of use)
- Support for all types of items (from single data items to complex executable modules)
- Support for version control
- Ability to follow complex associations between the items
- Support for dynamic hierarchy relationships
- Discoverable and presentable on multiple layers and dimensions
- Easy to maintain
- Ability to follow security requirements
- Unlimited size.

I. RDTA and the Test Automation Creation Work Process

Adapting the RDTA approach mandates a new four step work process.

1. Analysis of project artifacts and the creation of a project repository.
2. Mapping the affiliations of project artifacts to existing reusable artifacts.
3. Acquiring test artifacts from the common repository for insertion into the project repository.
4. Designing missing test artifacts at the project repository and operation of automatic upload of the new test artifacts to the common repository.

J. Implementing RDTA

Implementing RDTA may prove to be a hard and complicated task in light of the variability and complexity of infrastructure, organizational cultures, standards and new quality measurements. One can foresee two different approaches for implementation:

- Top to bottom – where management dictates, supervises and imposes changes in production.
- Bottom up –where change develops from the bottom through limited experimental trials of one of the test automation teams and subsequently percolates up and spreads gradually through the organization.

IV. CONCLUSION

This paper presents a new conceptual approach to test automation – RDTA. This approach focuses on the reuse principle for test automation artifacts. In order to transition from concept to practice, each subject and proposition presented here should be addressed and developed into an organizational strategy and framework to reduce costs. More broadly, we envision the creation of international sharing schemes for the purpose of resource and performance amplification. Further development of the criteria for the selection of services and the evaluation of RDTA benefits are required.

REFERENCES

[1] , A. G. Malishevsky, G. Rothermel and S. Elbaum, “Modeling the cost-benefits tradeoffs for regression testing techniques”. Software Maintenance. Proc. International Conference on, IEEE, pp 204-213, 2002.

[2] D. M. Rafi,. K. R. K. Moses, K. Petersen, and M.V. Mäntylä,.“Benefits and limitations of automated software testing: Systematic literature review and practitioner survey”. Proc. of the 7th International Workshop on Automation of Software Test, IEEE Press, pp. 36-42, June 2012.

[3] D. Almog and Y. Tsubery, “How the Repository Driven Test Automation (RDTA) will make test automation more efficient, easier & maintainable”. Proceedings of the 8th India Software Engineering Conference. Bangalore, India, ACM: 196-197, Feb 2015.

[4] J. Fortue and R. Valerdi,, “A Framework for Reusing Systems Engineering Products,” Syst. Eng. vol. 16, no. 3, pp 304-312, 2013.

[5] J. Parsons and C. Saunders, “Cognitive Heuristics in Software Engineering Applying and Extending Anchoring and Adjustment to Artifact Reuse,” IEEE Trans. Softw. Eng., vol. 30, no. 12, pp. 873-888, Dec 2012.

[6] G. Wang and J. Rice, “Considerations for a Generalized Reuse Framework for System Development.” Proc. 21st INCOSE Int. Symp., June 2011.

[7] C. E. Cagdas, K. Bhattacharya, J. Su, “Static Analysis of Business Artifact-centric Operational Models,” 2007 IEEE Int. Conf. on Serv. Oriented Comput. and Appl., June 2007, pp. 133-140.

 ISBN: 978-1-61208-441-1

[8] A. H. Bagge, M. Bravenboer, K. T. Kalleberg, K. Muilwijk, E. Visser, "Adaptive Code Reuse by Aspects, Cloning and Renaming," Tech. Rep. UU-CS, issue: 2005-031 (2005).

[9] V. Khusidman and D. M. Bridgeland: "A Classification Framework for Software Reuse", J. of Object Technol., vol. 5, no. 6, pp. 43-61, July - August 2006.

[10] B. Boehm,. "Managing Software Productivity and Reuse," Computer, vol. 32, no. 9, pp. 111-113 1999.

[11] E. J. Weyuker, "Testing Component-Based Software: A Cautionary Tale". IEEE Softw., Vol. 15, No. 5: pp. 54-59, 1998.

[12] M. Poonawala, S. Subramanian, W. T. Tsai, R. Vishnuvajjala, R. Mojdehbakhsh, L. Elliott, "Testing Safety-Critical Systems-A Reuse-Oriented Approach" Proc. 9th Int. Conf. on SEKE, June, 1997, pp. 271-278.

[13] J. McGregor, "Testing a Software Product Line," Testing Techn. in Softw. Eng. Springer Berlin Heidelberg, 2010.

[14] J. Bosch, Design and Use of Software Architecture: Adopting and Evolving a Product-Line Approach, Addison-Wesley, 2000.

[15] Condron. "A Domain Approach to Test Automation of Product Lines," Int. Workshop on Softw. Product Line Testing. p 27, (2004).

[16] R. Tiwari. and N. Goel, "Reuse: Reducing Test Effort ACM," SIGSOFT Softw. Eng. Notes, pp 1-11, March 2013

[17] M. Gupta and M. Prakash, "Possibility of Reuse in Software Testing," 6th Annu. Int. Softw. Testing Conf. in India., 2006.

[18] D. Almog and T. Heart, "Developing the Basic Verification Action (BVA) Structure Towards Test Oracle Automation," IEEE 2010 Conf. on Computational Intell. and Soft. Eng. (CiSE), 2010, pp. 1-4.

[19] D. Almog and Y. Tesubery, "Reuse of Unit Test Artifacts – Allow Us to Dream," Agile Rec. Issue 16 pp. 49 – 52, Nov. 2013.

 ISBN: 978-1-61208-441-1