

# PROCESS-ORIENTED TEST AUTOMATION OF CONFIGURABLE BUSINESS SOLUTIONS

Benjamin Blau, Hong Tuan Kiet Vo  
Chair for Information Management and Systems  
University of Karlsruhe (TH)  
Englerstr. 14  
D- 76131 Karlsruhe

blau@iism.uni-karlsruhe.de  
vo@iism.uni-karlsruhe.de

**Abstract:** Along with rapidly changing market requirements for service-oriented business software new challenges for quality management, quality assurance, and testing arise. Customers must be able to react flexibly on changing business environments. As a consequence business solutions tend to become more and more configurable and customizable in order to be adaptable to business processes - not vice versa. However, this new level of flexibility has a large impact on testing and test automation in particular.

In this paper we propose a test automation framework and concept which facilitates process-oriented testing of highly configurable business solutions including automated test case generation and risk-based ranking in order to detect erroneous components and system inconstancy.

## 1 Introduction

The rapid change of markets and business requirements results in a high impact on the development of software solutions. To guarantee flexibility and low costs for change management, business solutions are based on loosely coupled services and processes which are part of service-oriented architectures. Modeling approaches for business processes that are based on independent software components which interact through flexible services, significantly shape innovative business solutions ([WM06], [HS05]). These business solutions are based on a component-based architecture that offers the required level of flexibility in order to adapt the actual business needs. However, as a consequence, it is very difficult to predict the state of the software and business proceedings at a certain point of time. This leads to new challenges regarding quality management and quality assurance, especially in the field of test automation.

Test suites have to adapt to changed requirements and configurations automatically in order to reveal erroneous processes and inconsistent systems states. In particular performing integration test of individually composed business scenarios is a challenging task for quality assurance teams. Additionally the increasing speed of software

development requires that test suites are developed and applied earlier in the software development life-cycle. Studies have shown that the larger the distance between error causation and error detection, the larger the costs for error correction [Gr03].

In this paper we propose and elaborate a framework for the process-oriented test automation of configurable business solutions. In the following section we discuss the test requirements of configurable business solutions in general. In section 3 we describe a generic test automation framework that is fundamental to the process oriented test automation approach. Finally section 4 concludes this work with a summary of the lessons learned and an outlook on further research steps.

## 2 Test Requirements of Configurable Business Solutions

Current business solutions enable companies to adopt, align, and execute their business processes according to the actual business needs. Consequently, these systems no longer implement a number of predefined business processes, but follow a component-based approach that allows for a highly configurable approach to model and implement business processes. Companies can select from a number of process components belonging to a specific business domain and arrange these components according to their business needs. Yet, this level of flexibility poses distinct system test requirements.

Tests are necessary in order to reduce the uncertainty of quality provided by business solutions [Jo02]. In general, one can distinguish between different categories of tests including module-, integration-, acceptance-, functional-, compatibility-, performance-, load-, and stress-tests [My04]. With regard to configurable business solutions, *integration tests* are of particular importance. Integration tests ensure the interoperability of components (i.e. modules, objects) and validate the correctness of the actual business process instance in terms of process flow and input-output requirements.

For traditional business solutions which implement a fixed set of business processes, integration tests are usually conducted manually by business experts. However, this is not a viable option for configurable business solutions due to the number of possible business process configurations that have to be tested. One approach is the application of *automated test procedures* that execute a sequence of actions without the necessity of human intervention [GF99]. Yet, while automated tests can help to keep the testing time and labour costs within acceptable limits, the concept of automated tests requires for a detailed set up of the test procedures and selection of the test sets. In particular the test validation steps have to be carefully modelled in order to make up for the implicit validation of manual testing. Hence, implementing automated-test procedures may result in a complex and costly endeavour that requires for a structured “engineering” approach as outlined by the automated testing life-cycle methodology ATLM [DRP 99]. With regard to configurable business solutions, automated tests per se do not address the challenge that respective test automation procedures have to be implemented for each potential business process instance.

Consequently, for effective integration tests of configurable business solutions, the test automation requires for a process oriented approach which offers means for a modularization and automatic creation of test sets. Moreover, in order to address the trade-off between test-quality and time constraints test set risk ranking methods are required. In the following sections we propose and discuss a process-oriented test automation framework that can be applied within the context of configurable business solutions.

### **3 Process-oriented Test Automation**

In this Chapter we present the main steps of our approach for process-oriented test automation. First we describe how to develop modularized test scripts in order to meet requirements for building a flexible test automation framework. Furthermore we provide ways to formalize reasonable process flows as a basis for automated test set generation. To deal with high scalability and test set complexity we introduce methods for risk elicitation and risk-based test set ranking to guarantee that more critical process flows are tested earlier.

#### **3.1 Modularization of Test Sets**

A common approach for generating test sets for test automation is the *capture/replay* approach ([Du03], [CTC05]). The test developer performs a test sequence manually and records her interaction. This process results in a monolithic test script which entails a hard coded recording of each step in a sequence. Hence, the test script generated by the capture/replay approach includes static values which leads to high effort in case the software and its requirements change. Even though many steps in a test sequence occur multiple times in other sequences, each test set has to be recorded from scratch. Especially in a non-deterministic environment as presented in this paper where the sequence of process steps changes, composability and adaptability are main requirements for an efficient test development. Modularization and encapsulation are key concepts which are necessary to meet these requirements and are essential to build test sets for automated testing of configurable processes. Each process step is tested by a modularized and parameterized test script which has a well-defined interface for composition. Test building blocks minimize the impact of changed requirements on the test script development which consequently leads to a reduction of maintenance effort. Furthermore this approach guarantees a high degree of flexibility which is inevitable for dynamically testing configurable business solutions.

## 3.2 Formalization of Configurable Process Flows

Recalling, the concept of configurable business solutions implies that the system state is not deterministic meaning that it is not predictable in which way a customer configures her processes. Therefore, it is necessary to formalize all possible process sequences in design time. This has to be done in an efficient way in order to extract reasonable paths which can be tested automatically. To make the concept of the Process Flow Matrix understandable we firstly introduce the concept of the Process Flow Graph which is a more intuitive formalization.

### 3.2.1 Process Flow Graph

All reasonable processes and process transitions from a business perspective can be represented by a *Process Flow Graph (PFG)* [MP01]:

**Definition:** A Process Flow Graph *PFG* is a four-tuple  $\langle V, E, I, F \rangle$  with

- $V$  is a set of nodes  $\{v_1, \dots, v_i, \dots, v_j, \dots, v_n\}$  representing processes
- $E \subseteq V \times V$  is a set of directed edges representing process transitions. An edge  $(v_i, v_j) \in E$  if process  $i$  is a predecessor of process  $j$  in a sequence
- $I \subseteq V$  is the set of processes with no predecessor (initial processes)
- $F \subseteq V$  is the set of processes with no successor (final processes)

### 3.2.2 Process Flow Matrix

A *Process Flow Matrix (PFM)* is an extended adjacency matrix representing a Process Flow Graph. Additionally weights describing risk values are added to the edges of the graph. Risk values are determined by frequency of occurrence of process transition in a process sequences. The more often a process transition is part of process sequences the more critical is this particular process for the flawless functioning of the whole system [AJ04]. Risk values are a central concept for the risk rank approach explained in Section 3.4.

**Definition:** A Process Flow Matrix  $PFM_{ij}(n \times n)$  is a special adjacency matrix with weights  $rv_{ij}$  representing the riskiness of the process transition from process  $i$  to process  $j$ . It is  $rv_{ij} \neq 0 \Rightarrow (v_i, v_j) \in E$  in the corresponding *PFG*.

- $\exists j \in [1, n]$  such that  $\forall i \in [1, n]: rv_{ij} = 0$  ( $j$  is a initial process)
- $\exists i \in [1, n]$  such that  $\forall j \in [1, n]: rv_{ij} = 0$  ( $i$  is a final process)
- $\nexists i, j$  such that  $rv_{ij} \neq 0 \wedge rv_{ji} \neq 0$  (no recursive edges)
- $\forall i, j$  with  $i = j$  is  $rv_{ij} = 0$  (no reflexive edges)

### 3.3 Automatic Test Set Generation

Automatically generating test sets to test reasonable process flows can be done by extracting all paths from initial processes to final processes in a Process Flow Matrix. The problem of path extraction is a search problem in a graph with directed edges. Many search types such as breadth-first-search, depth-first-search or leaf-root-search can be applied to extract all process paths. A lot of research has been done in the area of designing efficient algorithms and data structures for graph search ([IA86], [ZH03]). Nevertheless many script programming languages are very simple and not capable of recursive program code. Therefore we exemplarily introduce a non-recursive variant of the depth-first-search algorithm which guarantees the extraction of all paths from initial to final processes and yields a sufficient complexity of  $O(|V| + |E|)$ .

---

**Algorithm 1** *NonRecursiveDepthFirstSearch*

---

*Necessary*  $x$ : node,  $s$ : stack

```
1:   visited[ $x$ ]  $\leftarrow$  true
2:    $s \leftarrow x$ 
3:   while stack not empty
4:        $x \leftarrow$  getTopElement( $s$ )
5:       forall  $y$  such that  $\exists (v_x, v_y) \in E$  do
6:           if visited[ $y$ ] = false then
7:               visited[ $y$ ]  $\leftarrow$  true
8:               stack  $\leftarrow$   $y$ 
9:           endif
10:      removeElementFromStack( $x, s$ )
11:   endforall
12: endwhile
```

---

### 3.4 Risk Ranking of Test Sets

The diversity of reasonable process configurations leads to a large amount of test sets which results in a time-consuming test execution phase. Testing each path within a graph of reasonable process flows leads to an exponential problem comparable to the complexity of GUI tests [MPS01]. Therefore the test sets have to be ordered in a sense that test sets evaluating more critical process steps have to be tested earlier respectively ranked higher. In order to generate a ranking with respect to riskiness of process flows it is inevitable to identify critical process transitions. For this purpose weights are added to the transition from one process component to another in order to represent a certain risk value of this transition. These weights are continuously updated during the productive usage of the system and represent the frequency by which this particular path has been treated. The basic idea is that process transitions which are relatively often used imply a higher risk in case of erroneous processes or system inconsistency for the daily business [AJ04].

### 3.4.1 Risk Ranking Methods

Valuation of risk always depends on various factors. Therefore we introduce a palette of methods for risk ranking and analyze implied biases. Let  $T$  be the set of test sets  $s_1, \dots, s_m$  for each process flow. Furthermore let  $R_{\hat{s}}$  be the set of all risk values for process transitions contained in test set  $\hat{s}$  with  $R_{\hat{s}} = \{r_1, \dots, r_n\}$ .

- *Mean-based Risk Ranking* generates a ranking for test set  $\hat{s}$  based on the mean risk value of all process transitions contained in the process flow tested by  $\hat{s}$ .

$$MR = \frac{1}{n_{\hat{s}}} \sum_{i=1}^{n_{\hat{s}}} r_{\hat{s}i}$$

As process flows that include many transitions yield a higher risk, it is reasonable to take the path length in account. Normalization leads to

$$MRR(\hat{s}) = \frac{\sum_{i=1}^{n_{\hat{s}}} r_{\hat{s}i}}{\sum_{s=1}^m \sum_{j=1}^{n_s} r_{sj}} \in [0,1]$$

The mean-based risk rank method tends to ignore a small number of transitions that are very critical compared to the mean (outliers) which reduces the sensibility for rare highly critical process transitions.

- *Variance-based Risk Ranking* focuses more on critical outliers than on the mean risk value of a process flow. Meaning process flows containing transitions with high risk are ranked very high even though they might contain a large set of transitions yielding a low risk.

$$VR = \frac{1}{n_{\hat{s}}} \sum_{i=1}^{n_{\hat{s}}} (r_{\hat{s}i} - \bar{r}_{\hat{s}})^2$$

Normalization leads to

$$VRR(\hat{s}) = \frac{\sum_{i=1}^{n_{\hat{s}}} (r_{\hat{s}i} - \bar{r}_{\hat{s}})^2}{\sum_{s=1}^m \sum_{j=1}^{n_s} (r_{sj} - \bar{r}_s)^2} \in [0,1]$$

- The *Hybrid Risk Ranking* combines the mean-based and the variance-based risk rank method. The proportion of each component is determined by weights  $w_{MRR}$  and  $w_{VRR}$  with  $w_{MRR} + w_{VRR} = 1$ .

$$HRR(\hat{s}, w_{MRR}, w_{VRR}) = w_{MRR} MRR(\hat{s}) + w_{VRR} VRR(\hat{s})$$

### 3.4.2 Example and Discussion

For illustration and evaluation purpose Figure 1 depicts three process flows *A*, *B* and *C* as well as frequencies respectively risk values for each transition.

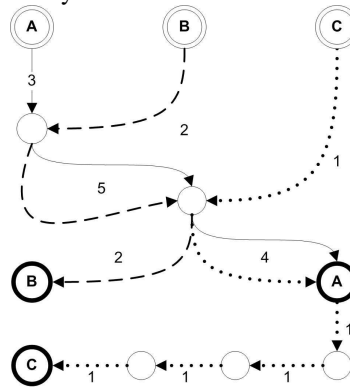


Figure 1 Process Flow Scenario

In order to rank the test sets for each process flow proposed risk ranking methods have been applied to the scenario (see Table 1).

Test Set <i>s</i>	Risk Rank Methods		
	$MRR(s)$	$VRR(s)$	$HRR(s, 0.9, 0.1)$
<i>A</i>	0.40 <sub>(1)</sub>	0.11 <sub>(3)</sub>	0.37 <sub>(1)</sub>
<i>B</i>	0.30 <sub>(2)</sub>	0.49 <sub>(1)</sub>	0.32 <sub>(2)</sub>
<i>C</i>	0.30 <sub>(2)</sub>	0.41 <sub>(2)</sub>	0.31 <sub>(3)</sub>

Table 1 Evaluation of Risk Rank Methods

Table 1 shows that applying the mean-based risk rank to the test sets assigns the highest risk rank to process flow *A* as the average frequency of its transitions is relatively high. On the other hand, applying the variance-based risk rank yields a different outcome. Process flow *B* gets the highest rank because its mean frequency is relatively low but *B* also contains the most critical transition with a high frequency.

Combining both methods by applying the hybrid risk rank with a stronger weighting on the mean-based component leads to *A* as the most critical process flow. On average process flow *A* contains high frequent transitions and it also contains the most critical one which implies that the corresponding test set has to be tested first.

### 3.4.3 Risk Value Adaption Methods

To incorporate knowledge about critical process transitions into the system it is reasonable that risk values are assigned by the software vendor based on best-practice information in the first place. Although in the productive stage of the system it is essential for efficiently generating risk ranks to continuously update risk values weighting process transitions. Every time a process transition is activated the corresponding risk value has to be adapted using an adequate algorithm. This procedure facilitates a learning progress which keeps the system up-to-date and provides the basis for just-in-time risk-based testing. Similar to the risk rank methods, it depends on the testing focus which risk value adaption method should be implemented. Therefore we introduce a set of methods for risk value adaption and analyze induced biases.

- *Linear-recursive Risk Value Adaption* increases risk values at constant proportions. This implies that every activated transition regardless of its former risk valuation is increased equally.

$$r_i(t + 1) = r_i(t) + c$$

In a long-run this leads to a risk underestimation of transitions involving newly added processes because well-established process transitions have gained a relatively high risk value over a long period of time. Therefore they receive a much higher rank compared to new and maybe more critical transitions.

- *Logistic-recursive Risk Value Adaption* is based on the set of logistic or sigmoid functions describing an exponential growth with a certain degree of saturation.

$$\frac{\Delta r_i}{r_i} = \frac{r_i(t + 1) - r_i(t)}{r_i(t)} = \left(1 - \frac{r_i(t)}{L}\right)$$

Solving for  $r_i(t + 1)$  results in the recursive form

$$r_i(t + 1) = r_i(t) + \left(1 - \frac{r_i(t)}{L}\right)$$

The characteristic of saturation assures that risk values of new process transitions are increasing relatively fast which compensates the main disadvantage of the linear-recursive risk value adaption. Even after a long period of risk value updates newly treated process transitions can gain an equally high risk rank compared to often adapted process transitions.

### 3.5 Summary and Big Picture

This section summarizes our approach for process-oriented test automation of highly configurable business solutions and illustrates the main steps. In design-time all reasonable process flows have to be formally described in a Process Flow Matrix. Additionally a repository of test modules has to be developed. Each test module tests one process step and its capabilities are determined by well-defined interfaces.

At run-time graph search algorithms extract all reasonable paths (process flows which have to be tested) in the Process Flow Matrix in order to generate a repository of test sets. These test sets are filtered based on the user configuration. Every test set containing a process step which is deselected by the user is being removed from the repository. The result is a repository of relevant test sets which are ordered with regard to their risk rank. The risk rank is computed by one of the proposed metrics (MRR, VRR, HRR) depending on the requirements of the application scenario. Finally the repository of ordered test sets functions as a blueprint for the composition of the test modules. This step results in a set of ordered tests which are ready for execution.

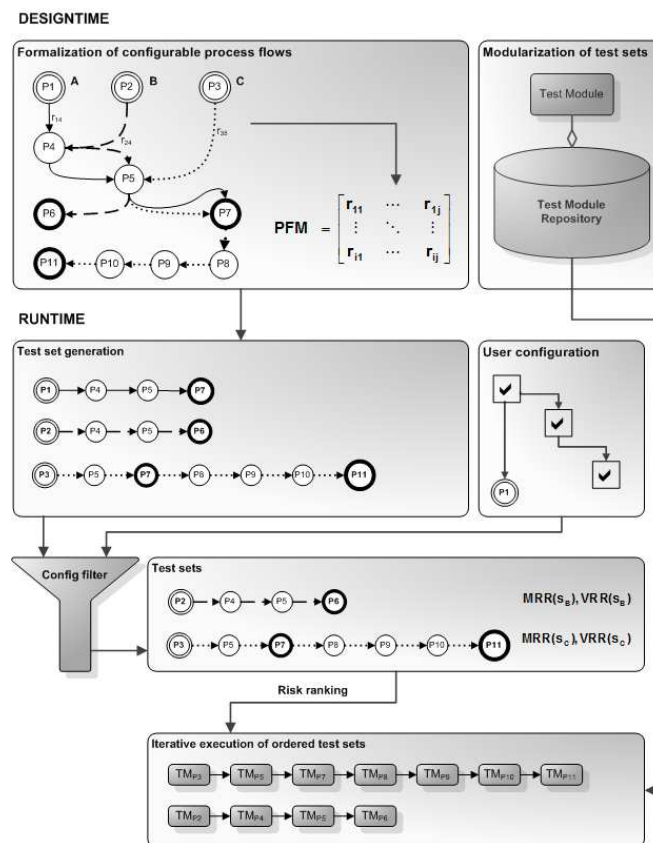


Figure 2 Approach for Process-oriented Test Automation

## 4 Conclusion

In this paper we provided an approach for process-oriented test automation of configurable business solutions. This approach includes concepts for modularization of test sets, formal description of process flows as well as automated test set generation and risk-based ranking.

According to this approach a test automation framework has been developed for a hosted business solution of a major German software company. The framework has been evaluated in cooperation with a series of beta customers and is going to be released as a part of the final product. Future work will focus on further evaluating and improving the process-oriented test automation framework on the basis of real-life feedback.

## References

- [AJ04] Ames, A. K.; Jie, H.: Critical Paths for GUI Regression Testing, Univ. of California, Santa Cruz, 2004.
- [CTC05] Chen, W. K.; Tsai, T. H.; Chao, H. H.: *Integration of Specification-Based and CR-Based Approaches for GUI Testing*, IEEE Computer Society Washington, DC, USA, 2005, pp. 967-972.
- [Du03] Dustin, E.: *Effective Software Testing: 50 Specific Ways to Improve Your Testing*, Addison-Wesley Professional, 2003.
- [DRP 99]Dustin, E.; Rashka, J.; Paul, J.: *Automated Software Testing: Introduction, Management and Performance*, Addison-Wesley Professional, 1999.
- [GF99] Graham, D.; M. Fewster, M.: *Software Test Automation*, 1999.
- [Gr03] Gregory. T.: *The Economic Impacts of Inadequate Infrastructure for Software Testing*, Collingdale, PA: DIANE Publishing Co, 2003.
- [HS05] Huhns, M. N.; Singh, M. P.: *Service-oriented Computing: Key Concepts and Principles*, 2005, pp. 75-81.
- [IA86] Imai, H.; Asano, T.: *Efficient Algorithms for Geometric Graph Search Problems*, SIAM, 1986, pp. 478.
- [Jo02] C. Jorgensen, *Software Testing: A Craftman's Approach*, CRC Press, 2002.
- [MP01] Memon, A. M.; Pollack, M. E.: *Coverage Criteria for GUI Testing*, 8th European Soft. Engineering Conference (ESEC) & 9th ACM SIGSOFT Intern. Symp. on the Foundations of Soft. Eng. (FSE-9), 2001, pp. 48-109.
- [MPS01] Memon, A. M.; Pollack, M. E.; Soffa, M. L.: *Hierarchical GUI Test Case Generation Using Automated Planning*, 2001, pp. 144-155.
- [My04] Myers, G. J.: *The Art of Software Testing*, John Wiley and Sons, 2004.
- [WM06] Woods, D.; Mattern, T.: *Enterprise SOA: Designing IT for Business Innovation*, O'Reilly, 2006.
- [ZH03] Zhou, R.; Hansen, E.: *Sparse-memory Graph Search*, 2003, pp. 1259–1266.