Experience with Industrial adoption of Business Process Models for User Acceptance Testing

Deepali Kholkar, Ajay Deshpande, Aditya Shetye, Pooja Yelure, Harshit Tiwari
Tata Consultancy Services,
India
(deepali.kholkar, ajay.deshpande, aditya.shetye, pooja.yelure, harshit.tiwari)@tcs.com

Abstract. Model based testing or the generation of tests from machine readable models has been widely deployed in industry for testing embedded systems and devices. Attempts are being made to extend its use to business systems. However, in spite of its potential for process improvement, its large-scale adoption for testing business systems is not yet seen, mainly due to little data being available on such use. This paper presents the findings from industrial deployment of a business process model based testing approach for User Acceptance Testing of large banking and insurance systems. The approach met with easier acceptance from the user community due to use of business process models and has proved to scale to very large models. It resulted in an overall productivity benefit of 20-30% in test design and planning, in addition to digitization of domain and process knowledge and has been successfully adopted organization-wide. Benefits as well as issues faced in large-scale adoption are discussed along with solutions found and open problems.

1 Introduction

Model based testing (MBT) has been extensively researched over the last few decades [2][4][6][7][10-12]. Its industrial adoption is common for safety-critical systems, however, only recently it is being applied for testing of business applications [4]. Large-scale adoption in this area is not seen, an important reason being lack of available data on such deployments. Other factors are few off-the-shelf implementations of MBT, difficulty of switching to the modeling paradigm, cost and complexity of creating models, large size of generated test suites and scalability [7].

Traditionally MBT methods have used state machines or formal specifications as input [5][6], both of which are hard notations for non-technical teams that test business applications. It is only recently, with the advent of business process models (BPM) that are easily understood by business and test teams that this community could be considered a target user set for MBT. BPM has grown in popularity to become the de-facto method for documenting business processes. Standards such as OMG's Business Process Modeling Notation (BPMN) have brought uniformity in modeling formats across the available BPM tools [3]. Scalability becomes an issue with increase in size of state models [6].

Most available MBT tools focus on automated generation of tests without test data generation [3][4]. Conformiq generates test data as well [5]. Validation of model content is not addressed in available tools. We developed a toolset for BPM based MBT in-house since at the time there was no available work using BPM for test generation and also to apply constraint solving techniques for process model validation and test data generation.

This paper reports on the experience of applying our MBT approach [1] in an organization that conducts User Acceptance Testing (UAT) for major financial corporations. A number of pilot case studies were conducted on large real-world financial applications. The paper documents the findings from this experience including the preparatory work necessary to apply the approach on a large scale across the organization, results from the pilot case studies, problems encountered and open issues. Results indicate an overall productivity gain of 20-30% in test planning and more accurate estimation of testing effort due to digitization of process and domain knowledge. Performance of our method on other measures, viz. cost and complexity, scalability and test suite size is also discussed.

2 The UAT context

User Acceptance Testing of business applications involves validation for important/ critical user stories. The UAT process in industry is predominantly manual with consequently low productivity and has heavy domain knowledge dependency.

The organizational unit that applied our approach is a 1200+ member team responsible for the UAT for major multinational banks and insurance corporations, on financial applications in areas of Corporate Banking, Consumer Banking, Capital Markets and Insurance. UAT is done on each software release before it is rolled out into production.

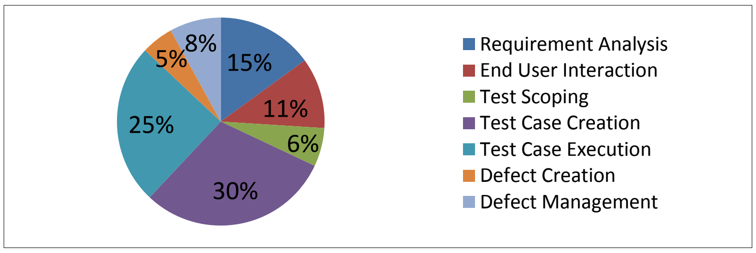

Fig.1. Effort distribution in UAT

Critical activities performed for each release and their average effort distribution in the manual UAT process are depicted in **Fig.1**. This data was obtained from effort logs maintained by the team as part of their operational process. As seen from the figure, most effort is spent in Requirement Analysis, Test Case Creation and Execution. The first two of these constitute the Test Planning phase which also includes Test Scoping.

Table 1. Problems faced by UAT team

1. Productivity	• Improve productivity in Requirement Analysis, Test Creation and Execution, Test Maintenance
2. Quality	• Dependence on Test lead for coverage of test scope. Errors in scoping lead to rework and potential post production defects • Lack of standardization in test processes and formats • Unavailability of end-to-end process knowledge since teams work in silos. Lack of awareness of impact of change on upstream and downstream systems results in post-production defects
3. Planning	• Inaccuracy of testing estimates leading to schedule slippage • Need for accurate change impact estimation

The unit was faced with multiple challenges such as showing year-on-year productivity, safeguarding domain knowledge of the team when faced with attrition and accurate test cycle estimation to avoid production release delays. Specific problems to be tackled in order to address these were identified by the UAT team as in **Table 1**.

Although taken from the UAT context, the activity break-up as well as challenges are true of the Testing activity in general and form a set of goals and metrics for measuring effectiveness of an MBT approach.

3 The MBT Approach

Our MBT approach [1] and toolset for it named Assurance Workbench (AWB) is depicted in **Fig.2**. The approach comprises capturing a model of the application and automated test generation from the model in two stages – Scenario and Test case generation and is described here using one of the pilot case studies viz. Insurance New Business as a running example.

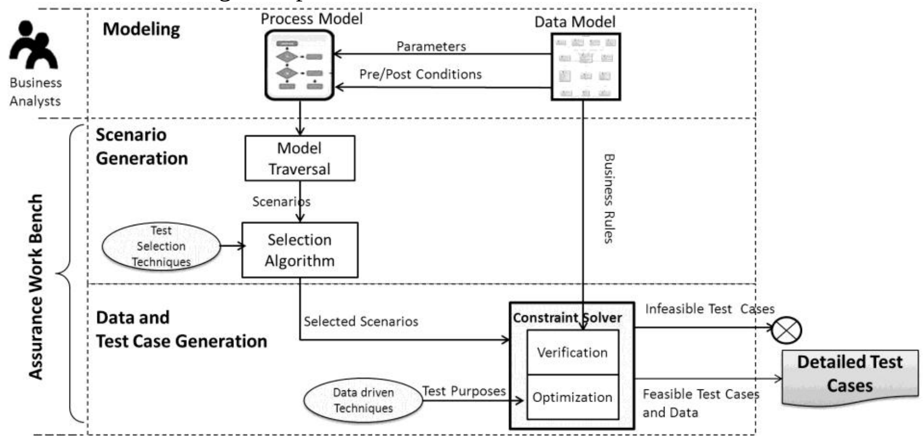

Fig.2. Our MBT approach

3.1 Modeling

Business Analysts capture the important workflows of the system in the form of business process maps. **Fig. 3** shows the top level New Business process flow drawn in BPMN, which depicts issuance of a new Insurance policy for a customer, made up of Activities (lowest level atomic tasks) such as *Select Product*, Sub-Processes (complex tasks) such as *Generate Illustration* and Gateways (decision or branch points). Each sub-process is detailed in a separate process diagram. The New Business process hierarchy consisted of this main process and 12 sub-processes, containing a total of 12 gateways, 28 branches and 75 activities.

The domain data model is captured as a Unified Modeling Language (UML) class diagram, shown on the right in **Fig. 3**. Business Rules are specified as Object Constraint Language (OCL) constraints on the data. The New Business pilot data model had 6 classes, 20 attributes and 4 business rules. A sample Business Rule is *SexTypeSelectionRule* which states that when StateCode is NY, Sex can be specified as Unisex only, else as Male/ Female. The OCL expression for this rule is
((*self.StateCode* = Insurance::Enum_State_Code::NY) and *self.Sex* = Insurance::EnumSex::Unisex)) or not(*self.StateCode* = Insurance::Enum_State_Code::NY)

where *self* denotes the class *InsuredDetails* to which this rule is attached, of which *Sex* and *StateCode* are attributes.

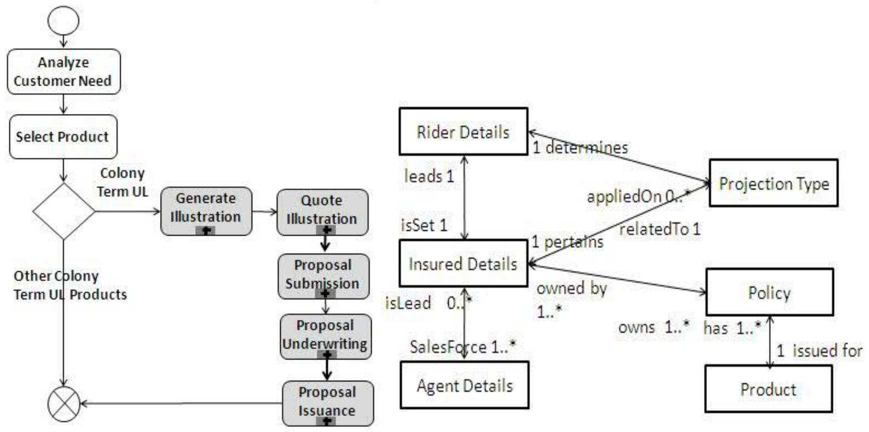

Fig. 3. Process and Data model: Insurance New Business

Input and output parameters of process steps form the link between the process and data models as shown in **Fig.2**. E.g. an instance *p* of class *Product* is attached as input parameter to the task *Select Product* of **Fig. 3**. Processes can be annotated with rules in the form of pre/ post conditions for tasks e.g. *Colony Term UL* in **Fig. 3**. These are expressed as OCL constraints on parameters of the process. The OCL expression for pre-condition *ColonyTermUL* is
p.ColonyTermULType = Insurance::Enum_ColonyTermULType:: ColonyTermUL
where *p:Product* is a parameter.

3.2 Scenario Generation

The process model graph is traversed using standard Breadth-first traversal to generate an exhaustive set of paths. Each generated end-to-end path consisting of sequence of branches, conditions and tasks forms a business/ test **scenario.** Branches are sufficient to uniquely identify a scenario. **Table 2** shows the business scenario matrix (BSM) for the New Business process generated by our tool, in which each row is a scenario defined by the branches in the column values. Each column denotes a gateway. e.g. Row #1 is the scenario defined by branches SelectProduct=Colony Term UL, CustomOption=Y, Projection= LGS and so on. Only 10 of the 12 gateway columns are shown here due to space constraint.

The set of scenarios generated is usually very large and can be reduced using the following Test Selection strategies

- **Minimal selection:** Selects a minimal test set from the exhaustive set, covering every element of the process graph at least once, i.e. gives standard node-edge coverage. Useful to get the most efficient test-suite covering all system functionality. The minimal test set for New Business contained 7 scenarios out of 290, shown in the BSM in **Table 2**.
- **All Combinations/ Pairwise selection:** Selects the minimal test set covering all combinations/ pairwise [9] combinations of branches from selected gateways. Used

when maximum coverage of combinations needs to be ensured for critical branches of the workflow.

- **Selective generation:** Selects a reduced test set containing paths passing through selected Gateways/ Activities/ Pre/ Post Conditions only. Used when test cases for only specific parts of the system are required. Useful for getting the impacted test cases by specifying elements of the model which have changed.

Table 2. Scenario Matrix for New Business

#	**Select Product**	**Cus-tom Op-tion**	**Projec-tion**	**Face Amt**	**Catch up Pro-vision**	**Catc hup Ti-er1**	**Pro-jec-tion Op-tion**	**Rid-er**	**Waiver of speci-fied premi-um**	**Form Generation**
1	CT- UL	Y	LGS	N	Y	N	Y	Y	Y	Welcome Letter
2	CT- UL	N	-	-	-	-	-	N	-	PA Disclosure
3	CT- UL	Y	LGS	Y	Y	N	N	Y	Y	Policy summary
4	CT- UL	Y	LGS	Y	Y	Y	N	Y	N	Demo of CPB
5	CT- UL	Y	LGS	Y	N	-	N	N	-	-
6	CT- UL	Y	LGNS	-	-	-	-	-	-	-
7	CT- OUL	-	-	-	-	-	-	-	-	-

3.3 Test Case Generation

For each **scenario** selected, pre- and post-condition and business rule constraints are automatically translated by the toolset to a specification for the model checker Symbolic Analysis Laboratory (SAL) [8]. *Sal-atg* is used to solve these constraints to find a solution state that represents appropriate data values for testing the scenario. Conflicting constraints result in no solution and constitute infeasible paths that are filtered out as depicted in **Fig.2**.

A **test case** is defined as the **scenario** plus **data.** Multiple test cases can be obtained for a scenario using any of several data-driven strategies, viz.

- **Boundary Value Analysis (BVA)**: Generates 5 values per integer attribute, on and either side of each bound of value range.
- **All Value Coverage:** Covers all valid values of an attribute.
- **Positive/ Negative tests:** User-defined combinations of data values

Each data condition is translated to a test purpose for SAL, as shown in **Fig.2**. SAL tries to satisfy multiple test purposes in each solution, generating an optimized solution set. A second level of infeasible test filtering takes place here since conflicting combinations of test purposes, workflow paths and business rules are eliminated by the solver. E.g. when attribute StateCode (values: CA, NY, VT, NJ, MT) and Sex (values: Female, Male, Unisex) selected for All Value coverage, test purposes created are

TestPurpose01= (StateCode = CA), TestPuropose02 = (StateCode = NJ)…
TestPurpose11= (Sex= Female), TestPurpose12 = (Sex= Male)...

Exhaustive set would be 5*3 = 15 combinations. SAL combines multiple independent test purposes and generates an optimized set of 6 solutions that conform to the business rules, in this case *SexTypeSelectionRule* explained above.

Automated validation for conflicts and generation of test data sets in accordance with rules enforces semantic correctness.

A sample generated test case from the New Business test suite is shown in **Table 3** and is defined by the test (pre-) conditions, sequence of SubProcesses and Tasks and their parameters with generated data values.

Table 3. Test case sample from New Business

Test Case #	**Test Condition**	**Sub Process**	**Task**	**Parameter field**	**Value**
1	Colony Term UL		Customer Need Analysis		
			Select Product		
		Insured Details	Select Insured Details Tab	InsuredAge	27
				Branch	16
				State_Code	NY
				Sex	Unisex
	…………..				
	Rider = Y	Rider Details	Select Policy Rider Tab	PolicyOwnersAge	64
				Child Rider Units	0
				Waive_premium	Y
				Face Amount	50,000

4 Application of MBT

The UAT organization selected key processes from various application areas for carrying out the pilot case studies. A total of 20 pilots were carried out, of which results from 9 are presented here. The processes chosen for the case studies were all large and complex to check scalability for real-life models. Three preparatory stages were needed before actual application of MBT could commence

1. Modifications to existing UAT process to accommodate BPM
2. Arriving at a strategy for modeling and
3. Training the workforce on modeling.

This was the additional investment for effecting the process change, which the organization decided to make, expecting the knowledge captured in BPM would help manage the problem of knowledge loss due to attrition and facilitate reuse.

4.1 Changes to the UAT process

The manual UAT process relies completely on the domain knowledge of Business Analysts and Test Leads for test design. Business Analysts create Business Requirement and Functional Requirement Documents (BRD and FRD) for each feature. These are used by Test Leads as the basis for deciding testing scope and designing tests. Execution of Test cases is manual.

The new BPM based process required analysts to create process maps and specify business rules instead of BRD/ FRD. Impacted test cases, UAT scenarios and test cases were to be generated automatically from the model. **Fig. 4** depicts the existing and changed UAT process.

4.2 Adoption of BPM

A modeling scheme was required in order to ensure uniformity across models created by different users, so that results across teams could be compared.

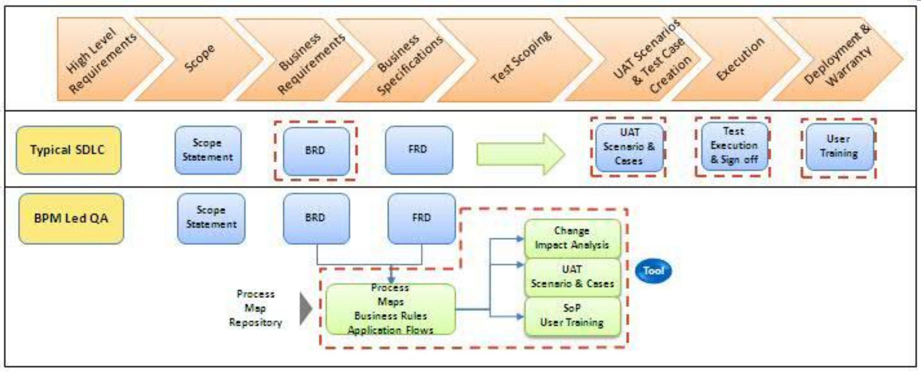

Fig. 4. Comparison of existing and MBT approaches for UAT

The UAT organization devised an architecture clearly defining the levels of abstraction at which the process model would be captured. Levels 1-4 in the process hierarchy capture generic information pertaining to the domain while Levels 5 and 6 contained system specific details. These are logical levels that may map to several physical levels.

- Level 1 gives the **product** level information
- Level 2 maps contain core **processes** that fall under that product
- Level 3 maps list down the **sub-processes** for the corresponding processes
- Level 4 maps have **activities** performed for a sub-process.
- Level 5 maps have the specific **tasks** that a user will perform to execute an activity. Data parameters and business rules are tagged here
- Level 6 maps have the **key strokes** that a user performs to execute the tasks.

This standardization enables automated processing of the information during test generation. Level 5 of the model is used to output the detailed test case sheet in **Table 3**. Level 6 is used to generate test cases at user interface action level, i.e. test scripts that can directly be used by testers for manual execution.

Separation of domain and system-specific aspects into separate layers was extremely interesting. The former could now be modeled by domain experts who did not need to know operational details of systems. Secondly, domain-specific or application-specific test cases could be generated by traversing down to the appropriate level in the model.

Training given to users included this architecture, the modeling notation, data modeling and specification of business rules through a simplified user interface. To ensure consistency of modeling, a maker-checker process is followed where a checker reviews content, levels, notation and standards for each process created by a maker.

4.3 Pilot Case Studies

The nine pilot case studies discussed in this paper along with the problem areas/ key features they represent are listed below.

- **Credit Cards**: Inaccurate testing estimates, non-standard test case formats and dynamic test cycles.
- **Payments process 1**: Immediate need from the end-user organization to show productivity benefits. Important module impacting 9 upstream and downstream sub-systems
- **Securities Trading**: Improper test planning, post production defects, high attrition problem. Complex functionality with lot of variations.
- **New Business**: Wide range of reusable functionalities.
- **Trade Processing**: Most mature process with good information of requirements, Planning, execution and Operations. Critical module impacting 45 upstream and downstream sub-systems
- **Wealth Management (WM), Payments processes 2 and 3, Check Processing**: Very large and complex processes with loops and multiple entry points. WM impacted 30 upstream and downstream sub-systems.

Table 4. gives the size of each process in terms of the number of sub-processes, gateways, gateway branches and leaf-level activities. The number of branches impact number of paths, adding complexity to scenario generation. Number of pre-conditions, classes and attributes signify complexity for the constraint solver. Number of Activities adds complexity to test execution. *NA* indicates data modeling was not done for the last four; being very large models, it was decided to focus only on their scenario generation, described in the next section.

Table 4. Complexity Of Case Study Models

Model	Sub Pro-cesses	Gate way s	Branch es	Pre-Condition constraints	Activities/ Test Steps	No of classes	No of attrib trib-utes
Credit Cards	5	4	14	26	31	3	6
Payments process 1	15	7	53	55	73	5	39
Securities Trade	67	17	100	56	147	8	71
New Business	12	12	28	26	75	6	20
Trade Processing	33	14	55	50	22	9	49
Wealth Management	103	142	401	622	573	NA	NA
Payments process 2	52	36	193	269	322	NA	NA
Check Processing	53	26	99	83	225	NA	NA
Payments process 3	112	71	637	775	849	NA	NA

For each pilot case study, models were captured, test generation and impact analysis done for a set of identified changes to the initial model. Generated tests were reviewed by the UAT team. Results are discussed in the section below.

5 Results

Results of test generation from the pilot case studies are listed in **Table 5** in chronological order. Column 3 shows the size of the generated test set, being quite large, test selection was employed to get a reduced test set. Column 4 shows the selection strategies chosen in each case by the UAT team. Column 5 gives the size of the reduced test set and Column 6 the percentage of tests selected in it.

5.1 Coverage

Exhaustive coverage of the specification was achieved using automated generation, from which desired coverage could be achieved using selection. Minimal selection gave a very efficient test set covering all tasks and branch conditions in the process at least once. Payments and SFS needed combination coverage of critical branch conditions, which was easily possible using All combinations selection. Column 6 shows that in every case, a very small percentage of tests was sufficient to satisfy the selection criteria. Any other paths in the workflow that are not part of this selection but are of interest could be included by additionally using Selective generation described in Section 3.2. In cases where particular data values needed to be tested for, tests were obtained using Positive/ Negative selection.

Table 5. Test case generation results

Application process	#of Sub-Processes	Exhaustive Test Cases (paths)	Selection Type	Reduced Test Cases Selection	% Test cases selected	Manual Test Planning Effort (ph)	Automated Test Planning Effort (ph)	Effort Reduction %
Credit Cards	5	150	Minimal	5	3.33%	108	84	22
Payments process	15	8076	AllCombinations	12	0.15%	672	491	27
Securities Trade	67	17 million	AllCombinations	33	0.01%	1632	1224	25
New Business	12	290	Minimal	7	2.42%	163	120	24
Trade Processing	33	1741	Minimal	12	0.69%	192	146	24
Wealth	103	> 4 million	Minimal	47	0.01%	1200	900	25
Payments process 2*	52	> 300 million	Minimal	128	NA	NA	NA	NA
Check Processing*	53	> 30 million	Minimal	226	NA	NA	NA	NA
Payments process 3*	112	> 1 billion	Minimal	190	NA	NA	NA	NA

***Total count of test cases not available since being very large model, processed in parts as described in section on Scalability below. NA: Effort data not available yet; data collection still underway*

5.2 Productivity

The last three columns in **Table 5** show the efforts for manual test creation, automated generation using MBT and the percentage reduction or overall productivity gain achieved, which ranges from 22-27%. The effort figures have been taken from a tracker in which the time spent on each activity by each person per release is manually captured.

The MBT effort includes time taken for initial creation and validation of the model, auto-generation and validation of tests. Average time for complete scenario generation is under 4 minutes. Average time taken by the constraint solver for data generation per test case (single data set) is between 1-2 minutes.

Change Management. Effort comparison between change impact analysis in the manual process and MBT approach for one of the projects is shown in **Table 6**.

Effort for understanding the change is the same in both cases. Impact computation in MBT is done using Selective generation (3.2) for the changed items in the model and takes 40% less time than in the manual case, as seen in **Table 6**. The overall productivity gain in total effort is 30%, as seen from the table.

Table 6. Effort comparison for Change Impact Analysis

Manual process (Effort in hrs)		**MBT (Effort in hrs)**	
Understand CR	16	Understand CR	16
Gap/ Impact analysis	40	Process flow creation/ modification in model	24
Discussion with stakeholders / SMEs	18	Discussion with stake holders/ SMEs	18
Update BRD	8	BRD/ FRD generation	Negligible
Total effort	82	**Total effort**	58

Effort needed to make modifications in the test suite in manual and MBT approaches is shown in **Table 7**. Effort for scenario and test case modification is shown separately. Whether the change is an addition or update of functionality, the effort to modify/ re-write tests impacted by changes is very high in the manual case. In the MBT approach, change is made centrally in the model and automated re-generation of scenarios and tests resulted in 20-30% savings.

Table 7. Effort comparison for Test suite maintenance for a Change Request

Type of Change	**Team Size**	**Scenarios**			**Test Cases**		
		# of Scenarios to create/ update	**Manual effort (Person hrs)**	**MBT Effort (Person hrs)**	**# of Test Cases to create/ update**	**Manual effort (Person hrs)**	**MBT effort (Person hrs)**
Addition of New Functionality	30	10	40	32	45	200	156
Update Existing Functionality		6	30	23	120	360	260

It is interesting to note that for scenarios, effort needed was more for update of existing functionality than for addition, in both manual as well as MBT, since one needs to understand existing functionality before making the change. Conversely for test cases, effort was more for addition of new functionality in both approaches, since a lot of detail has to be added, while update takes less time due to availability of detail.

Activity-wise Productivity. The graph in **Fig. 5** gives an activity wise comparison of the approaches.

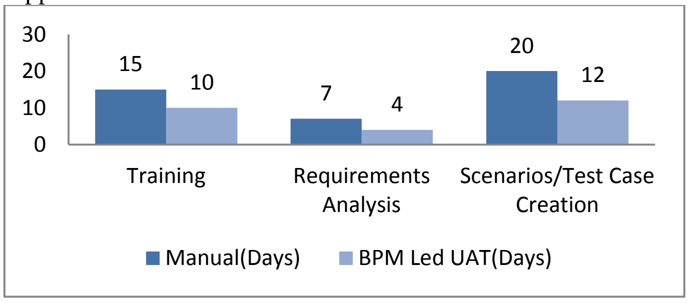

Fig. 5. Activity wise productivity comparison of Manual vs. MBT approach

It shows that MBT gives a productivity advantage of about 30-40% over the manual approach for each activity. Having knowledge captured in BPM brings down the time taken to train new members of the team. The time taken for requirement analysis is reduced due to structured models and automated impact computation. Productivity in scenario/ test case creation has been discussed above, where the overall productivity was found to be 20-30% as opposed to activity-wise 30-40% shown here, as a result of factoring in modeling effort.

The resultant test cases being detailed to keystroke level enables quick execution. Creation and maintenance of such detailed tests manually would be extremely inefficient. Breaking down maps into multiple levels has helped in gaining detailed tests and better test coverage.

5.3 Scalability

Securities Trading was a large and complex process with 67 sub-processes. The total number of possible paths was 17 million as seen in the table. The toolset encountered a scalability issue and ran out of memory in trying to apply selection on the entire path set.

To address this problem, the graph was divided into sub-graphs and selection applied on sub-graphs. Selected paths from the first sub-graph were connected to paths from the next sub-graph to form a bigger sub-graph on which selection was applied. This was iteratively done until all sub-graphs covered, yielding the minimal set of 33 paths.

5.4 Quality

The quality problems stated in Section 2 are addressed by MBT as discussed here.

Lack of optimized test coverage. In the existing UAT process, Test scope had to be determined by the Test Lead who had to ensure coverage as well as efficiency. Achieving this balance and designing tests keeping business rules in mind was an effort-intensive process. This effort had to be put in for every change to the test suite as well. Automated test selection allowed the Test lead to focus on just specifying test scope, while the actual test creation and optimization is automated, giving productivity and accuracy at no extra cost once models were in place. Simple Minimal selection provided a basic test set with guarantee of element coverage, which could be built upon.

Standardization in test design. The approach defines a process for test design that is systematic and repeatable since it uses machine-manipulable models. Validation against end-user requirements is the goal of testing, which is achieved through generation from end-user models. All teams can now follow the same process and generated test formats as opposed to differing processes and formats earlier.

Unavailability of end-to-end process knowledge. Standardization in the modeling approach has enabled digitization of domain as well as process knowledge and made it available to test teams. Impact on upstream and downstream processes can be ascertained by querying the model.

5.5 Planning

Accuracy in test planning and estimation. Test generation from the requirement model for the specified scope gives the exact number of tests to be run, enabling accurate planning as opposed to the Test lead having to estimate manually using his knowledge and experience. Automated assessment of change impact helps estimate the testing effort for changes and plan more accurately.

6 Discussion and Conclusion

Here we discuss our experience in the context of the problems cited in Section 1 that make adoption of MBT difficult. [2] gives a number of evaluation criteria for MBT. We have covered a majority of these in our evaluation.

6.1 Qualitative findings, Lessons learnt and open issues

Changing to the modeling paradigm. Modeling needed test teams to begin visualizing requirements in the form of process maps. Business Analysts were doing this earlier, now entire test teams had to be trained on modeling.

Modeling guidelines had to be formulated to avoid modeling errors that lead to problems in test generation. Model validations were programmed into the toolset to automate them to the extent possible. Training test teams to write business rules in OCL would not have been viable. A user interface had to be developed to simplify specification, however, specifying complex rules is still hard.

Cost of deployment. Adoption of MBT needs significant lead time before the new testing process can be rolled out. As discussed in Section 4, transition to the new process and a modeling strategy need to be worked out. Procuring a BPM tool, training personnel, creation of the as-is-models, evaluation against existing tests all contribute to initial cost of deployment although the investment is later expected to pay back through productivity gain and easier maintenance.

The UAT organization spent 2 elapsed months to come up with model architecture. For the initial roll-out, 300 people were trained on modeling and spent 6 months creating as-is process maps and generating tests, covering 60% of the total processes in various core areas. They later trained the remaining team. Area-wise summary of MBT deployment is detailed in table below.

Table 8. Area wise MBT pilot statistics

Business Area	Process maps	Scenarios (Post selection)
Corporate Banking	212	1650
Consumer Banking	140	2800
Capital Markets	260	1800
Insurance	180	900

Limitations of existing tools and technologies. There is a need for standardization of model interchange formats across BPM tools. Currently every modeler exports in different format and does not export all of the information in the model, making it difficult for generative tools like ours to support models from multiple tools.

Limitations of our approach. Our data generation approach has limitations - it generates data from the valid values provided to it. These need to be populated with real data from project databases for effective testing. It is a hard problem since data integrity between elements will need to be preserved. In the current approach, users had to manually replace some data in the tool generated output before executing the tests.

Scalability of constraint solving. Scalability problems in scenario generation were resolved. The extremely large models like Wealth Management, Payments process 3 etc. confirmed the approach scales for very large process models. However, scalability issues crop up in constraint solving when the size of the data model, particularly number of attributes and their value ranges increases substantially. Currently upto 100 attributes and integer ranges of upto 1000 values are handled by our toolset. Number of entities, associations or constraints do not pose a problem. Easier scripting interface, capability to handle complex data types and higher value ranges are needed in model checkers so that they can be more widely used with business applications.

6.2 Conclusion

Although there is an associated cost of deploying MBT, the modeling formalism enables automation of the test design and change impact assessment, bringing accuracy and efficiency to the process. This has the potential to pay back in the long term not only through productivity gain but through improved quality of testing due to a qualitative improvement in the process and a systematic, repeatable process. The user organization in our case, apart from using models to obtain productivity in Test planning, leveraged their investment by using the models for analysis, training, process optimization and transition.

Extending test generation to include creation of scripts for automated execution would give much greater productivity benefit since as discussed in Section 2, test execution is time-consuming. We have implemented this in our approach but its application poses problems due to large variation in user interface controls, platforms and coding styles.

The main difficulty in proliferation of MBT is high learning curve, cost of deployment, lack of data and process knowledge regarding its use e.g. modeling and optimization strategies. Reuse of models would lower the cost of MBT deployment. Emergence of supporting tools and methods and interoperability between tools as described above would make MBT adoption easier. The advantages of MBT make it worthwhile to work on reducing costs through improved methods and putting models to better use so as to make its application in industry viable.

7 References

1. Kholkar, D., Goenka, N., Gupta P.: Automating Functional Testing using Business Process Flows. In: 2ndWorkshop on Advances in Model-Based Software Engineering, 4th ISEC (2011)
2. Monalisa Sarma, P. V. R. Murthy, Sylvia Jell, Andreas Ulrich, Model-Based Testing in Industry: A Case Study with Two MBT Tools, in ACM Proceedings of 28th International Conference on Software Engineering (ICSE), May 7-9, 2011, Cape Town, South Africa.
3. ARIS: http://www.ariscommunity.com/test-designer
4. Smartesting http://www.smartesting.com/index.php/cms/en/solution/why-smartesting
5. Conformiq http://www.conformiq.com/cases/
6. I. Craggs, M. Sardis, T. Heuillard, AGEDIS Case Studies: Model-Based Testing in Industry, in: Proc. 1st Eur. Conf. on Model Driven Software Engineering, 2003, pp. 129–132.
7. A. Hartman, Adaptation of Model Based Testing to Industry, Agile and Automated Testing Seminar, Tampere University of Technology, 2006
8. Symbolic Analysis Laboratory (SAL) model checker http://sal.csl.sri.com
9. Combinatorial and Pairwise Testing http://csrc.nist.gov/groups/SNS/acts/index.html
10. Hartmann J., Vieira M., Foster H. and Ruder A.. 2005. A UML Based Approach to System Testing. Innovations in Systems and Software Engineering, vol. 1 (1), 2005, 12-24.
11. Briand L.C., Labiche,Y. 2002. A UML-based approach to system testing. Software Syst Model 1(1), 10–42
12. Tibor Bakota, Árpád Beszédes, Tamás Gergely, Milán Imre Gyalai, Tibor Gyimóthy, and Dániel Füleki. Semi-Automatic Test Case Generation from Business Process Models SPLST ’09 and NW-MODE ’09 (2009)