A Survey on Model-based Testing Approaches: A Systematic Review

Arilo C. Dias Neto[1] Rajesh Subramanyan[2] Marlon Vieira[2] Guilherme H. Travassos[1]

[1] Federal University of Rio de Janeiro - COPPE
P.O. Box 68.511, Zip Code: 21.941-972
Rio de Janeiro, RJ, Brazil
{acdn,ght)@cos.ufrj.br

[2] Siemens Corporate Research – SCR
755 College Road East, 08540
Princeton, NJ, USA
{rajesh.subramanyan,marlon.vieira}@siemens.com

ABSTRACT
This paper describes a systematic review performed on model-based testing (MBT) approaches. A selection criterion was used to narrow the initially identified four hundred and six papers to focus on seventy-eight papers. Detailed analysis of these papers shows where MBT approaches have been applied, the characteristics, and the limitations. The comparison criteria includes representation models, support tools, test coverage criteria, the level of automation, intermediate models, and the complexity. This paper defines and explains the review methodology and presents some results.

Categories and Subject Descriptors
D.2.5 [**Testing and Debugging**]: Testing Techniques – survey on Model-based Testing approaches.

General Terms
Measurement, Performance, Reliability, Experimentation, Security, Verification.

Keywords
Testing Approaches, Model-Based Testing, Test Case Generation, Survey, Systematic Review

1. INTRODUCTION
Model-based Testing (MBT) provides a technique for automatic generation of test cases using models extracted from software artifacts [1]. Software quality is controlled with reduced costs. A formal model, e.g. finite state machines, UML diagrams, describing the software or system behavior is needed.

According to Dalal [1], the automation of a MBT approach depends on three key elements: (i) the ~~*model*~~ used for the software behavior description, (ii) the ~~*test generation algorithm*~~ (criteria), and (iii) ~~*tools*~~ that generate supporting infrastructure for the tests. Other important characteristics are testing levels of MBT, automation levels, and complexity of non-automated steps.

This paper provides a range of options towards the selection of an MBT approach to a test practitioner for a given project. This work was performed by COPPE/UFRJ and SCR who have worked on testing ([3][4]) including MBT, and are investigating new MBT and automation solutions.

[5],[6], and [7] describe MBT related surveys on test data generation techniques, supporting tools, and test case generation approaches respectively. However, no formal survey on the analysis of MBT approaches have been found. To our knowledge, this is the first scientific survey paper on MBT approaches using a formal methodology – Systematic Review [8].

WEASELTech'07, November 5, 2007, Atlanta Georgia, USA

We identify and characterize the available MBT approaches through qualitative and quantitative analysis. This paper does not attempt to compare the merits of different approaches; it rather extracts absolute information as stated by the authors of that approach. Existing approaches were classified under UML (a widely used standard) and non-UML categories for ease of comparison. Different testing levels require different test case generation strategies. Approaches for functional testing are analyzed separately. MBT characteristics such as testing level, behavioral model, intermediate models, the level of automation, automated support, and test generation criteria are analyzed.

The paper is organized as follows. Section 2 describes the planning, execution, and a result summary of the literature survey. Section 3 presents the quantitative and qualitative results from the analysis of 78 papers. Section 4 discusses important issues and explores future trends on automation of MBT approaches. Finally, Section 5 presents the conclusions and future work.

2. SYSTEMATIC REVIEW: SURVEY PLANNING AND EXECUTION
A systematic review is a method to identify, evaluate and interpret the pertinent research about a particular research question [8]. Planned steps in this literature include: goal definition, research questions, source selection, search strings, paper inclusion criteria, classification, and information to be extracted from each MBT approach. Using the approach described in [8] and [9], we summarize the planned steps below:

- **Goal**: to characterize MBT approaches from publicly available technical literature based on what is relevant for MBT. The details are discussed in later sections.
- **Research question**: What are the published MBT approaches and what are their main characteristics?
- **Sources**: five digital libraries (IEEEXplorer, ACM Portal, INSPEC, Compendex IE, Web of Science), as well as websites and conference proceedings.
- **Search string**: (approach or method or methodology or technique) and (("model based test") or ("model based testing") or ("model driven test") or ("model driven testing") or ("specification based test") or ("specification based testing") or ("specification driven test") or ("specification driven testing") or ("use case based test") or ("use case based testing") or ("use case driven test") or ("use case driven testing") or ("uml based test") or ("uml based testing") or ("uml driven test") or ("uml driven testing") or ("requirement based test") or ("requirement based testing") or ("requirement driven test") or ("requirement driven testing") or ("finite state machine based test") or ("finite state machine based testing") or ("finite state machine driven test") or ("finite state machine driven testing")) and (software).

Each MBT approach is classified in one of the five categories:

[A] Model representing information from software requirements (functional testing) and is described using UML diagrams.
[B] Model representing information from software requirements and is described using any non-UML notation.
[C] Model representing information from software internal structure (architecture, components, interfaces, units;

structural testing) and is described using UML diagrams.

[D] Model representing information from software internal structure and is described using any non-UML notation.

[E] Papers collected during the search, but unrelated to MBT and therefore excluded.

406 papers were collected (Table 1) for classification from which 204 papers were excluded as their scope was either unrelated to this work, or were repeated, or were not available electronically. From the remaining 202 papers, currently 78 papers have been analyzed. Of these, 47 papers are UML based. Papers with high number of citations were selected for first pass as these papers/approaches have higher visibility in the community. The complete list of the 406 identified papers with classification is available in [10].

Table 1. Classification of Selected Papers

Papers Categories	Amount of Papers	Percent of Papers
Category "A"	26	6,40%
Category "B"	93	22,91%
Category "C"	21	5,17%
Category "D"	62	15,27%
Category "E"	147	36,21%
Not Classified	57	14,04%
Total of Papers	**406**	**100%**

From a review of the main characteristics of MBT approaches, [1], [2], etc. and academic and industry experience reports, fourteen fields (Table 2) were identified for extracting information from the selected papers. These fields were defined after reviewing the main characteristics of MBT approaches e.g. [1], [2], and academic and industry experience reports.

Table 2. Fields to characterize a MBT approach

Field	Importance for MBT
Testing Level	Define which level of abstraction is used by the MBT approach to check the software behavior. The levels used are: system, integration, unit and regression testing.
Supporting Tools	Defines the existence or not of supporting tools. A MBT approach may automate its steps by algorithms or guidelines, but it is required a tool to make feasible the execution of these steps. It is important to observe which steps the tools can support, and what are their limitations
Software Domain	Define the scope that a MBT approach can be applied, such as Execution Platform or Development Paradigm – Embedded System, Client-server System, OO System, Real-time System, Web Application, etc.
Behavior Model	Represents the software characteristics that may be tested by a MBT approach. It could represent the main limitation of an approach from what information it can represent or not. Sometimes the model is used for a specific application domain and cannot be used in another domain. Three aspects are important regarding the model: (1) how easy and automated is its development, (2) if it has all information necessary for test generation, and (3) how easy is to extract test cases from it.
Tests Coverage Criteria	Define the rules used to generate test cases from the software model. There are two types of criteria: data-flow and control-flow. They define the effort and quality of the results generated automatically by a MBT approach.
Test Cases Generation Criteria	Describe the steps to be followed for a MBT approach from the behavior model for test case generation or execution. They define what can be automated and what cannot, and how to integrate these steps.
Automation Level	Defines the proportion of the number of automated steps that compose a MBT approach per the number of steps. The main characteristic of MBT is the possibility of automated generation (and execution) of tests. Automation may mean less cost, time, and effort to generate test cases. Usually, MBT approaches are composed of automated and non-automated steps. We have used this metric to indicate quantitatively how automated is a MBT approaches.
Complexity Level of the Steps	Described indication regarding the effort, cost and time to perform the steps in a MBT approach, the complexity of automated steps are usually low. However, non-automated steps may require different effort, skill, or cost to be performed. Thus, it is necessary to analyze their complexity level indicating more precisely the automation level of an approach.

After identification, classification and information extraction of individual MBT approaches, the next step is to analyze them together. This is discussed in the next section.

3. ANALYSIS OF MBT APPROACHES

The following section presents the results of quantitative and qualitative analysis.

3.1 Quantitative Analysis

Statistical results on testing level that the approaches are applied to, level of automation for each approach, models used to describe the software behavior, tools and intermediate model are presented in this section. This information may present scenarios on Software Engineering not covered by MBT approaches.

3.1.1 Testing Level

This shows the application scope of MBT approaches (Table 3).

Table 3. Testing Level Analysis

Testing Level	# approaches	Supporting tool	
		Use	Not cited
System	48	32	12
Integration	17	7	10
Unit/Component	8	5	3
Regression	4	2	2

MBT have been mostly applied for System Testing (66% of all approaches) as MBT was originally intended for System Testing. Subsequently, MBT approaches were applied to the other levels - Integration Testing (22%), Regression Testing (5%), and Unit Testing (10%). Unit testing is not a usual abstraction level for MBT. The purpose of Unit Testing is to test small modules, functions, or classes after the implementation. Other approaches to support structural testing from the source code are better.

3.1.2 Automation Level

The proportion of automated steps that compose an approach and the complexity level of its non-automated steps are relevant to automation level analysis. A single non-automated step in a significantly automated approach can be harder than several manual ones in another MBT approach.

The non-automated has different complexity levels (Table 5) depending upon modeling of software behavior (every approach need to execute this task), choice of testing criteria, modeling of intermediate model, or translation from one model into another.

- **Low** ✔ : steps may be automated or involve a simple task;
- **Medium** ⚠ : e.g. intermediate model modeling or test data definition;
- **High** ✘ : e.g. manual translation between models.

MBT approaches typically have most steps automated except the first step - modeling of software behavior. Only 9 approaches have a non-automated step with high complexity;

3.1.3 Supporting Tools

MBT approaches have tools to support executing their steps (64%) and are primarily applied for System Testing approaches (Table 3). Estimating accurately the number of tools supporting test case generation process is hard due to the large number of tools under deployment, and proprietary and freeware tools that have been developed. Counting all tools or analyzing all the feature is not the intent and beyond the scope of the paper.

3.1.4 Model used for test case generation

We differentiate between the MBT approaches using UML and

non-UML models. UML Statechart (27 approaches), Class (19) and Sequence (19) Diagrams are models most often used. Approaches not using UML include Finite State Machine (7 approaches) and Z Specification (4).

Table 4. Number of approaches for Model

Behavior Model	# MBT approaches
Statechart Diagram	27
Class Diagram	19
Sequence Diagram	19
Use Case Diagram	11
OCL	11
(Extended) Finite State Machine	10
Activity Diagram	9
Collaboration Diagram	8
Object Diagram	7
Graph	7
Z Specification	4
…	…

3.2 Qualitative Analysis

Limitations, cost, effort, complexity, input, pre-requirements or skill required to use MBT approaches, and the quality of the outputs generated for them are discussed below.

3.2.1 Testing Coverage Criteria

Testing coverage may use a control-flow or a data-flow strategy. MBT approaches until 1999 predominantly used data-flow ([1][11][12]) based test coverage. After 1999, control-flow criteria has been more common ([13][14][15]). With increasing size and complexity of the system with each year, it's necessary to evaluate not just small modules, but full systems, making control-flow a better suited approach.

Languages/methods used for software modeling were previously limited and focused on black-box technique, and the systems were considered simple functions or a small module. However, the languages/methods developed recently are able to describe software structural information, making easier the analysis of software control-flow.

3.2.2 Behavior Model Limitations

The **model** must be correct in order to generate test cases accurately. To ensure the accuracy of the model, model checking strategies have been developed integrated with MBT approaches ([13][16][17][18][19]).

The expressive power of the notation or language used to describe the model is important and should be known prior to applying the MBT approach. Notations that do not permit certain descriptions decrease the testing quality. UML, FSM, B Language, and Z notation have been used and have presented interesting results.

3.2.3 Cost and complexity of applying MBT approach

The cost and effort complexity of applying a MBT approach has been analyzed. The levels of complexity were defined based on the following characteristics:

- **High:** Manual steps, no support tool, translation between models needed, code-instrumentation, and incomplete output.
- **Intermediate:** complex model notation, additional effort for modeling (intermediate model), and using of a proprietary tool.
- **Low:** Use of known Models (UML, FSM, B), fully automation, tools available, outputs generated in known format, and empirical evaluation.

Detailed analysis, charts, and spreadsheets of all surveyed MBT approaches are available in [10].

4. IMPORTANT ISSUES REGARDING MBT APPROACHES

Issues that could represent future perspectives in this field are:

4.1 Behavior Model Building

This is key in defining the scope of MBT approach application. Each model is limited on what it can or cannot describe. Models must be used according to the characteristics of the software artifacts domain or platform.

Integrating the MBT approach to the software development process is important. As observed in Table 5, some approaches need models designed exclusively for it. These approaches have limited usage due to learning overheads. Building different models for design and testing is an additional and unnecessary effort in the software project. A test model must describe specific characteristics relevant for the design or coding activities. Re-using or extracting a test model from the software behavior model improves the test team productivity and software quality, since it may support tracking changes in the software specification, and simplifies testing regression activities (e.g.: [4], [13], and [16]). [20] describes a test specific model. However, a change in software specification requires an additional effort to analyze these changes for regression testing.

4.2 Supporting Tool Integration

Tools must support non-automated (usually modeling) and automated (usually generation) steps. Lack of such tools makes practical application difficult. Manually applying an algorithm to generate test cases is not trivial. Manual integration between tools results in high costs. The testing activities should be integrated with the software development process (Section 4.1).

The importance of the integrating automated and manual steps can be observed here: In approach A, the test case generation is automated using a tool, however the input model is a text or XML file developed manually without a support tool. The effort of this modeling task can limit the usage of this MBT approach.

The main questions are: How to automate the integration of supporting tools for MBT into the software development process? How to integrate design and testing models?

Tools allowing the integration between models generated from the software development and testing models must be used. Common models reduce the time and effort to perform this task. xMapper [21], DIXSE [22], and DOME [23] integrate tools. Proprietary tools supporting MBT or tools integration may be feature-rich, but licensing fees limit its use in some organization.

4.3 Increasing Automation levels, reducing requirements to use MBT approaches

Automation level determines the practical feasibility of using an MBT approach. Complex non-automated steps makes an approach unfeasible. Complexity levels depend on modeling language, algorithms, and choices necessary to perform the step.

The requirements for using a MBT approach must be obtainable with low effort, time, and skill to perform the MBT steps. Reducing requirements and increasing automation in future by:

- Use simple and known language or notation, like UML, for software modeling;
- Use testing tools integrated with the sw development process;
- Automate the maximum of possible steps in a MBT approach.

4.4 Limitations of MBT solutions

The application domain, input formats, generated output, and modeling language limits the usage areas of a MBT approach. Before applying an MBT approach to a software project, its requirements, i.e., what information needs to be used, and has this been supplied by the software project has to be ascertained.

Other limitations include software quality characteristics that an approach is able to evaluate or the skill level required to use it.

4.4.1 Software Quality Characteristics

An application requires testing of both functional and quality characteristics or non functional requirements (NFR) such as access control, performance, security, usability and reliability.

Table 5. MBT approach characterization

Author/Title	Cat	Testing Level	Software Domain	Level of Automation	Behavior Models	Tools to support	Complexity of Steps	Testing of NFR
Abdurazik and Offutt, 2000	C	System Testing	OO	2/3	Collaboration Diagram	ND	✓	No
Ali *et al*., 2005	C	Integration Testing	OO	3/4	State Diagram, Collaboration Diagram, and SCOTEM	Yes	✓	No
Ammann and Offutt, 1994	B	System Testing	Not defined	1/5	Z Specification	ND	✓	No
Andrews *et al*., 2005	B	System Testing	Web Application	3/8	Finite State Machine	Yes	✗	No
Barbey *et al*., 1996	B	Unit Testing	OO	3/4	CO-OPN/2	Yes	✓	No
Basanieri and Bertolino,2000	C	Integration Testing	OO	3/7	Use Case, Sequence, and Class Diagrams	ND	✗	No
Bellettini *et al*., 2005	C	System Testing	Web Application	4/6	Extended Class, Statechart Diagrams	Yes	✓	No
Bernard *et al*., 2006	A	System Testing	Not defined	4/5	Class, Object, Statechart Diagrams, OCL	Yes	✓	No
Bertolino *et al*. ,2003	C	Integration Testing	Component-based Software	2/3	Use Case, Sequence, Class Diagrams	Yes	✓	No
Bertolino *et al*., 2005	C	Integration Testing	OO	2/4	Sequence, State Diagrams, Reasonably Complete Model	Yes	✗	No
Beyer *et al*., 2003	C	Integration Testing	Not defined	2/3	Annotated Sequence Diagram and MCUM	Yes	✓	No
Botaschanjan *et al*., 2004	A	System Testing	OO	0/3	Object, Sequence Diagrams and OCL	ND	⚠	No
du Bousquet *et al*., 1999	D	System Testing	Synchronous Reactive Software	3/5	Environment description	Yes	✓	Security
Briand and Labiche, 2002	A	System Testing	OO	8/9	Use Case + Activity Diagram, Sequence, Collaboration, Class Diagrams, Data Dictionary (OCL)	Yes	✓	No
Briand *et al*., 2002	A	Regression Testing	OO	2/3	Class, Use case and Sequence Diagrams	Yes	✓	No
Briand *et al*., 2002	C	Integration Testing	OO	2/2	Class Diagram and Graph	ND	✓	No
Briand *et al*., 2004	C	System Testing	OO	3/4	Statechart + OCL, Class Diagram, Transition Test Sequence	Yes	✓	No
Briand *et al*., 2004	C	System Testing	OO	3/4	Statechart + OCL, event/action flow graph	ND	✗	No
Briand *et al*., 2006	C	Unit Testing	COTS	3/7	Class, Sequence Diagrams + OCL, CSPE constraints, graph	Yes	⚠	No
Carpenter, 1999	A	System Testing	Safety-critical System	2/5	Use case, Sequence Diagrams	ND	✗	No
Cavarra *et al*., 2003	A	System Testing	OO	2/3	Class Diagram, State Diagram, Object Diagrams, Intermediate Format (ASM)	Yes	✓	No
Chang and Richardson, 1999	D	Unit Testing	Not defined	3/5	ADL Specification and Test Data Description (TDD)	Yes	⚠	No
Chen *et al*., 2002	A	Regression Testing	OO	2/3	Activity Diagram	ND	✓	No
Chen *et al*., 2005	D	Integration Testing	Not defined	3/4	Condition Data Flow Diagram - CDFD (SOFL)	ND	⚠	No
Chevalley and Fosse, 2001	A	System Testing	Critical Software	3/5	State Diagram	Yes	✓	No
Crichton *et al*., 2001	C	System Testing	OO	2/3	System Model (Class, Object and State Diagram) and Test Directives (Object and State Diagrams), Intermediate Format	ND	✓	No
Dalal *et al*., 1999	D	Unit Testing	Not defined	2/3	Requirements described in AETGSpec	Yes	✓	No
Deng *et al*., 2004	A	System and Regression Testing	OO	0/4	Use Case, Class, Statechart, Sequence, Collaboration, Activity Diagrams	Yes	⚠	No
Friedman *et al*., 2002	B	System Testing	Concurrent Software	4/6	Finite State Machine	Yes	✓	No
Gargantini and Heitmeyer,1999	B	System Testing	Not defined	1/3	SIS Specification	Yes	✓	No
Garousi *et al*., 2006	C	System Testing (Stress Testing)	Distributed System	2/4	Class, Sequence, Context, Network Deployment, and Modified Interaction Overview Diagrams, Control flow model, network interconnectivity tree, network traffic usage patterns, and inter-SD constraints	ND	⚠	Efficiency and Reliability
Gnesi *et al*., 2004	C	System Testing	OO	2/3	Statechart Diagram and IOLTS	ND	✓	No
Gross *et al*., 2005	C	Integration Testing	OO	2/3	U2TP and TTCN-3	ND	✓	No
Hartman and Nagin, 2004	A	System Testing	Distributed System	3/6	Combination of class state and object diagrams, State Diagram, and XML file	Yes	✓	No
Hartmann *et al*., 2000	C	Integration and Unit Testing	OO	3/4	Statechart Diagram and Graph	Yes	✓	No
Hong *et al*., 2000	B	System Testing	Not defined	5/6	Statechart, Extended Finite State Machine	Yes	✓	No

Kansomkeat and Rivepiboon, 2003	A	System Testing	OO	2/3	Statechart Diagram, Testing Flow Graph	ND	✓	No
Kim *et al.*, 1999	C	Unit Testing	OO	3/4	State Diagram, EFSM	ND	✓	No
Legeard *et al.*, 2004	D	System Testing	Critical Software	4/6	Formal Specification (B or Z)	Yes	✓	No
Linzhang *et al.*, 2004	A	System Testing	OO	2/3	Activity Diagram	Yes	✓	No
Liuying and Zhichang, 1999	A	System Testing	OO	3/7	State Diagram, FSM	Yes	✓	No
Lucio *et al.*, 2005	A	System Testing	OO	3/5	Class Diagram (Fondue), UML collaboration diagrams, UML state diagrams, OCL operations.	ND	⚠	No
Lund and Stølen, 2006	A	System Testing	Not defined	ND	Sequence Diagram	ND	ND	No
Mandrioli *et al.*, 1995	B	System Testing	Real-time and Critical System	3/6	Software Specification in TRIO	Yes	✓	Efficiency
Meyer and Sandfoss, 1998	A	System Testing (GUI)	Not defined	2/5	Use Case + Operational Profile - Test Engineering Model	Yes	✗	No
Mingsong *et al.*, 2006	A	System Testing	Java Programs	1/4	Activity Diagram	Yes	⚠	No
Murthy *et al.*, 2006	A	System Testing	Not defined	3/3	Extended Statechart Diagram, EXTENDED CTGM	Yes	✓	No
Nebut and Fleurey, 2006	A	System Testing	OO Embedded Software	3/5	Use Case Diagram, Sequence Diagram, Simulation Model	Yes	⚠	No
Offutt and Liu, 1999	B	System and Unit Testing	Critical Software	2/4	DNF, CDFD (SOFL), S-Module (SOFL), I-Module (SOFL)	ND	✓	No
Offutt and Abdurazik, 1999	A	System Testing	OO	3/4	Statechart Diagram	Yes	✓	Efficiency
Offutt *et al.*, 2003	A	System Testing	OO	6/8	Statechart Diagram, Specification Graph	Yes	✗	No
Olimpiew and Gomaa, 2005	A	System Testing	Software Product Line	ND	Feature model, use case model, static [class] and dynamic [statechart and object interaction] models, component-based software architecture models	ND	ND	No
Paradkar, 2004	B	System Testing	Reactive System	2/3	Operational Model (SALT), Extended Finite State Machine	Yes	✓	No
Parissis and Ouabdesselam, 1996	B	System Testing	Reactive Software	4/5	Environment Specification and the safety properties	Yes	✓	Security
Pretschner *et al.*, 2001	B	System Testing	Reactive and Embedded System	4/5	System Structure Diagrams, State Transition Diagram, Message Sequence Charts, Data Type Diagram	Yes	✓	Efficiency
Richardson *et al.*, 1992	D	System Testing	Reactive systems	2/4	In the case study, RTIL and Z Specification	ND	⚠	Efficiency and Security
Richardson and Wolf, 1996	D	Integration Testing	Not defined	2/4	CHAM	ND	✓	No
Riebisch *et al.*, 2002	A	System Testing	OO	5/6	Use Case, Statechart Diagrams, Graph and Usage Model	Yes	✓	No
Rumpe, 2003	A	System Testing	OO	0/3	Object Diagram, Sequence Diagram, OCL	ND	✗	Efficiency
Satpathy *et al.*, 2005	B	System Testing	Not defined	7/8	State Coverage Graph (B)	Yes	✓	No
Scheetz *et al.*, 1999	C	Integration Testing	OO	2/4	Class Diagram and State Diagram + OCL	Yes	⚠	No
Sinha and Pardkar, 2006	D	Integration Testing	Web service	2/3	WSDL-S, EFSM	ND	✓	No
Sokenou, 2006	C	Integration and Unit Testing	OO	3/4	Sequence Diagram, Protocol Statechart, OCL	ND	✓	No
Stobie, 2005	B	System Testing	Any Category	3/4	Finite State Machine and Abstract State Machine Language	Yes	✓	No
Stocks and Carrington, 1996	D	Unit Testing	Not defined	ND	Test Template, describing valid inputs and outputs	Yes	ND	No
Tahat *et al.*, 2001	B	System and Regression Testing	Distributed and Embedded System	3/5	Requirements expressed in Textual and SDL formats, EFSM	ND	✓	No
Tan *et al.*, 2004	B	System Testing	Not defined	3/4	Software Specification in LTL	Yes	✓	Security and Efficiency
Traore, 2003	A	System Testing	OO	4/6	Statechart, Class and Sequence Diagrams	Yes	⚠	No
Vieira *et al.*, 2006	A	System Testing (GUI)	Not defined	2/4	Use Case + Activity Diagram and Class Diagram	Yes	⚠	No
Wu *et al.*, 2003	C	Integration Testing	Component-based Software	ND	UML Collaboration/Sequence or Statechart Diagram	ND	✓	No
Xu and Xu, 2006	D	Integration Testing	Aspect-oriented programs	4/5	State Model (AOP)	ND	✓	No
Zhen *et al.*, 2004	C	Integration Testing	OO	0/4	Interaction, Activity and State Diagrams	ND	✗	Efficiency

(*) The full list with all approaches and their references are available at http://www.cos.ufrj.br/uploadfiles/1188491168.pdf.

Table 5 shows that only 10 approaches test specific type of NFR.

- Users don't follow the same pattern of behavior when executing an application. The inconsistent behavior is difficult to model to be used for test case generation.
- It is not clear how to model quality characteristics not addressed by MBT approaches such as network flow, software usability, security requirements?

4.4.2 Skill level required to use a MBT approach

Knowledge on software modeling notations, test criteria, test metrics, or languages to generate test scripts is tester skill requirements. Different approaches require different skills and these are needed ahead of usage. The main issue is how to minimize the human skill required and simultaneously maximize the automation level?

4.5 Experimental Evaluation of MBT solutions

There is a lack of empirical results on MBT approaches. In [24], the authors analyze the knowledge maturity levels of software testing approaches from different empirical studies between 1979 and 2004. The results reveal a technological gap, and automated testing techniques are rarely transferred to the industry.

Empirical studies should observe the following characteristics:

- Risks and impact of transferring an approach from academic to industrial environment;
- Efficiency of its results, analyzing its coverage;
- How easy is to apply it in real software projects;
- Effort, time and cost to use it;
- Which contexts it can be used, and what are its limitations.

In the 78 analyzed papers, only few papers describe approaches in an industrial environment. Usually, they have been developed for specific context and do not get introduced to the industry.

Data in this survey and elsewhere about the contexts where MBT approaches are infrequently applied or the effort and cost to use them give direction for new research topics.

The systematic literature review analyzed and presented quantitative and qualitative information on MBT approaches. A partial set of results could be presented in this paper, with the focus on automation. Due to space restrictions, detailed individual analysis could not be presented here. The complete results are available in [10], where the individual analysis of each issue presented in this section is described for all surveyed MBT approaches.

5. CONCLUSIONS AND FUTURE WORK

The paper is a survey characterizing and analyzing MBT approaches from 78 technical papers. Important issues on automation of MBT approaches that are indicators of future trends are described. Current deficiencies in MBT are:

- MBT approaches are usually not integrated with the software development process. The models used for tests generation are not integrated with artifacts from the software development process, or were defined exclusively by a MBT approach. Integration tools must be used to support this problem;
- The MBT approach usually cannot represent and test NFR's, such as user behavior, software usability, security and reliability. Solutions for these issues must be provided;
- Requirements to use a MBT approach include knowledge about the modeling language, testing coverage criteria, generated output format, supporting tools make the usage difficult/unfeasible; these need to be minimized;
- Most MBT approaches are not evaluated empirically and/or not transferred to the industrial environment. Empirical knowledge provides precise information about costs, effort, quality and context to apply a MBT approach.

Expanding on the qualitative analysis and providing recommendations to practitioners is proposed for the future.

ACKNOWLEDGMENTS

Dias Neto would like to thanks FAPEAM for the financial supporting. Prof. Travassos is a CNPq researcher. This work has been developed in context of eSEE and SCR grant.

REFERENCES

[1] S. Dalal et al. (1999), "Model-based testing in practice", In: ICSE'99, May, pp. 285--294.

[2] A. Pretschner (2005), "Model-based testing", In:ICSE, p.722-723.

[3] G.M. Lima, G.H. Travassos (2005), "A Strategy for Object-Oriented Software Integration Testing". In: LATW'2005.

[4] M. Vieira, J. Leduc, B. Hasling, R. Subramanyan and J. Kazmeier (2006), "Automation of GUI testing using a model-driven approach", In: AST'06, ACM Press.

[5] J. Edvardsson (1999), "A survey on automatic test data generation". In: 2nd ECSEL, pages 21--28. October.

[6] A. Hartman (2002), "Model Based Test Generation Survey", Technical Report, available on 11/2006 at http://www.agedis.de/downloads.shtml.

[7] M. Prasanna et al. (2005), "Survey on Automatic Test Case Generation", Academic Open Internet Journal, available at http://www.acadjournal.com/2005/v15/part6/p4/.

[8] B. Kitchenham (2004), "Procedures for Performing Systematic Review", Joint Technical Report Software Engineering Group, Department of Computer Science Keele University, UK, and Empirical Software Engineering, National ICT Australia Ltd.

[9] J. Biolchini, P.G. Mian, A.C. Natali, G.H. Travassos (2005), "Systematic Review in Software Engineering: Relevance and Utility", Technical Report ES-679/05, PESC-COPPE/UFRJ. Available at http://www.cos.ufrj.br.

[10] A.C. Dias Neto, G.H. Travassos, R. Subramanyan, M. Vieira (2007), "Characterization of Model-based Software Testing Approaches", Technical Report ES-713/07, PESC-COPPE/UFRJ. Available at http://www.cos.ufrj.br/uploadfiles/1188491168.pdf.

[11] J. Chang, D.J. Richardson (1999), "Structural specification-based testing: automated support and experimental evaluation", In: 7th ESEC, ACM Press, pp. 285-302.

[12] I. Parissis, F. Ouabdesselam (1996), "Specification-based testing of synchronous software", In: Symposium on the Foundations of Software Engineering (21), pp.127 - 134.

[13] L.C. Briand, Y. Labiche (2002), "A UML-Based Approach to System Testing", Technical report, Carleton University.

[14] C. Mingsong, Q. Xiaokang, L. Xuandong (2006), "Automatic test case generation for UML activity diagrams", In: AST '06, ACM Press.

[15] K. Stobie (2005), "Model Based Testing in Practice at Microsoft", Electronic Notes in Theoretical Computer Science 111(SPEC ISS), 5-12.

[16] A. Bertolino, E. Marchetti, A. Polini.(2003), "Integration of 'components' to test software components", Electronic Notes in Theoretical Computer Science, 82(6), pp.49-59.

[17] A. Gargantini, C. Heitmeyer (1999), "Using model checking to generate tests from requirements specifications", In: ESEC/FSE-7, ACM Press.

[18] M. Satpathy, M. Leuschel, M Butler (2005), "ProTest: An Automatic Test Environment for B Specifications", In: Electronic Notes in Theoretical Computer Science, 113 - 136.

[19] L. Tan, O. Sokolsky, I. Lee (2004), "Specification-based testing with linear temporal logic", In: IRI'2004, 493-498.

[20] M. Utting, A. Pretschner, B. Legeard, B., "A taxonomy of model-based testing", Technical report 04/2006, Department of Computer Science, University of Waikato, April, 2006.

[21] R. Spinola, M. Kalinowski, G. Travassos (2004), "A Mechanism for CASE Tools Integration", Brazilian Symposium on Software Engineering (SBES), Brasília (in Portuguese).

[22] P. Rodriguez-Gianolli,, J. Mylopoulos (2001), "A Semantic Approach to XML-based Data Integration". In: 20th ICCM, Japan.

[23] Z. Cui, M.D.J. Cox, D.M. Jones (2001), "An Environment for Managing Enterprise Domain Ontologies". M. Rossi and K. Siau (eds.) Information Modelling in the New Millennium, Idea Group Publishing, London.

[24] N. Juristo, A.M. Moreno, S. Vegas (2004) "Reviewing 25 years of testing technique experiments". Empirical Software Engineering: An International Journal, 9, March.