Model based testing of Datawarehouse

Kuldeep Deshpande

Capgemini
Pune, Maharashtra, 411038, India

Abstract

Testing forms a major part of development lifecycle of a software system. Testing is responsible for ensuring software product meets intended quality of software systems. Large percentage of failure of Datawarehouse projects necessitates an investigation into alternative techniques for all aspects of datawarehouse development lifecycle including testing.

Model based testing is a technique applied for many software products and has proved to help increase the quality while reducing overall cost of the software systems. In this paper, we propose application of model based testing to datawarehouse systems and discuss how it can help solve some of the challenges in datawarehouse testing. This paper also proposes implementation details and future roadmap for model based datawarehouse testing.

Keywords: *Datawarehouse, Testing, Model based Testing, Model driven architecture.*

1. Introduction: overview of datawarehouse testing

Development of datawarehouse (DW) systems is one of the most risky IT projects. Failure of datawarehouse projects has been reported in literature. Failure rate in datawarehouse is as high as 50% [1].

Such a high rate of failure necessitates need for an investigation in all aspects of development life cycle of datawarehouse. Requirements management of datawarehouse has been thoroughly studied in literature [13]. Similarly, various design aspects such as data mart design, datawarehouse evolution, ETL design and modeling have been reported in literature [23], [19].

Ralph Kimball has stressed on need for a controlled testing by datawarehouse end users in a user acceptance testing phase [14]. According to him datawarehouse testing has to encompass 3 phases i.e. unit testing, QA testing which is independent testing by a team of testers and UAT by end users. He has also proposed a template for ETL testing test cases. In [3] authors have proposed a framework for datawarehouse testing. This work lists artifacts to be tested in datawarehouse project: conceptual schema, logical schema, ETL procedures, database and front end. These artifacts have been proposed to be tested using various types of test such as performance test, functional test, stress test, recovery test, security test, usability test and regression test. The authors then describe which artifacts should undergo which kind of test and at what time in lifecycle of the project. This study is an important DW testing literature that considers how, what and when of DW testing. However there have been very few similar studies proposing end to end DW testing methodology. In [4] a set of best practices for DW testing has been proposed. This paper has stressed on importance of testing various business rules.

1.1 Challenges in datawarehouse testing

In [15] a systematic literature survey of datawarehouse testing literature is done followed by a set of interviews of practitioners. This study lists challenges for datawarehouse testing as follows:

- Product related challenges: This category includes all challenges related to inputs for testing OR artifacts produced during testing. This category of testing challenges includes lack of clarity on requirements, unavailable source data for testing, insufficient testing data and lack of richness in test data, lack of time for designing and planning large number of test cases and time required for executing long running programs during testing.
- Process related challenges: This category includes challenges related to activities performed during testing like lack of clarity on ownership of various testing activities, limited time for testing, bad source data quality and lack of traceability of datawarehouse requirements.
- Resource related challenges: This category includes challenges related to human resources and tools like lack of business and business knowledge and testing skills and lack of formal testing tools for DW.

Rest of this paper is organized as follows: In section 2, we discuss model based development approach for datawarehouse, in section 3 we give an overview of model based testing in general for software products. In section 4, we introduce the concept of model based testing for datawarehouse and discuss how it can help overcome some of the challenges discussed in this paper. Proposed

implementation methodology for model based testing of datawarehouse is also described in this section. Finally, in section 5, we discuss future roadmap for model based testing for datawarehouse.

2. Emergence of Model driven approach for DW

As per CWM, model driven approach is defined as a "standard framework for software development that addresses the complete life cycle of designing, deploying, integrating, and managing applications by using models in software development. [12]"

Model Driven Approach is based on 3 main types of models: PIM, PSM & CIM

A Computation Independent Model (CIM) is also often referred to as a business or domain model because it uses a vocabulary that is familiar to the subject matter experts. It presents exactly what the system is expected to do, but hides all information technology related specifications to remain independent of how that system will be (or currently is) implemented [12].

Platform Independent models (PIM) are created in such a way that these can be used for building systems on any platform. Technical details of system are abstracted in the PIM.

A Platform Specific Model (PSM) is a platform dependent specification of the system. It uses system specifications in the PIM and generates details of how these specifications should be implemented for the platform.

In recent years, there has been extensive research in the area of Model Driven Architecture for Datawarehouse.

In [12], authors have described how MDA and CWM (Common Warehouse Metamodel) can be used for requirements gathering and design of Datawarehouse.

Similarly in [2], a datawarehouse framework (DWF) and Unified process (2TUP) has been proposed for development of datawarehouse using model driven architecture.

Currently a number of tools exist in the market for exchange of metadata across various phases of DW lifecycle. For example

- Kalido (www.kalido.com) is agile information management software. Kalido Information engine lets a business user define business information model which is used to drive many of the activities in development of the datawarehouse. ETL developers map the source data to business information model. Kalido tool creates physical tables to load the source data and the transformations to load this data [28].
- Meta Integration (MI) Model Bridge (http://www.metaintegration.net): This is a tool that provides integration of metadata across various data modeling, ETL and reporting tools as well as metadata repositories [27].
- Metadata tools such as Adaptive (www.adaptive.com) can be used to build a business glossary / taxonomy. Business rules / transformations can be specified in the metadata tool. The metadata tool creates metadata in CWM format which can be used for creating ETL code and data model [26].

3. Model based testing

Benefits of model driven approach go beyond just development of software using a model of the software system. Model based testing is a testing technique for automated generation and execution of test cases based on formal models of system under test (SUT). Model based testing completely automates testing process [7]. Mode based testing can be applied at various levels of testing i.e. unit testing, integration testing and system testing [16].

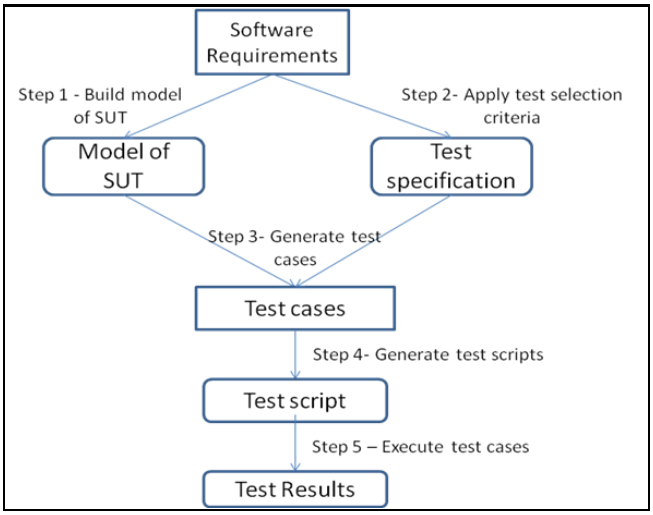

Fig. 1 Model based testing steps

3.1 Steps in Model based testing

Steps in Model of MBT can be described as follows:

- Step 1 - A model of SUT is built based on requirements or specification documents. This model is intended to be used for generating test cases. Hence required level of details for generation of test cases is added and details that are not required for test case generation are abstracted.
- Step 2 – A criteria is defined based on the requirements to select the test cases. Objective of the test selection criteria is to make sure that defined set of test cases detect maximum number of sever defects in software at acceptable cost [10].
- Step 3 – After deciding the criteria to generate test cases and getting requirements for the SUT, a set of test specifications is generated. A test specification is an

abstract description of test cases that can be translated into a test script.

- Step 4 – By using test specification as an input a set of executable test scripts is generated. This test script can be directly executed on SUT.
- Step 5 – This is the final step in MBT that includes executing the test case and recording the output.

3.2 Important features of Model based testing

- Online – offline: In online testing, MBT tool creates as well as executes the test cases. In offline MBT, test cases are created by the MBT tool, but not executed at the same time.
- Test coverage / test selection criteria: Software test coverage criteria allow the identification of the percentage of the software that has been evaluated by a set of test cases. Data coverage criteria help to identify how to choose few data values from a large set for testing purpose. Boundary analysis and domain analysis have been recommended in literature as possible data coverage criteria. Structural model coverage criteria use knowledge of structure of the model to choose few test cases to be executed. Few other test selection criteria such as stochastic, random and requirement based selection criteria have been recommended in literature [10].
- Test generation algorithm: Real advantage of MBT lies in usage of algorithms to automatically create test cases. Models generated from Finite state machines, theorem proving, model checking and symbolic execution have been proposed for generation of test cases.

3.3 What models can be used as source for model based testing

Generally a behavioral model of the SUT derived from model describing the software is used as input for model based testing. Design or architectural models do not contain sufficient details such as control flow OR data flow to enable generation of test cases [16]. This necessitates creation of a separate model for generating test cases. In case of model driven development, model used for code generation and model used for test case generation needs to be different. Various models have been recommended to be used as testing models for generation of test cases. For example, UML2.0 testing profile has been recommended as source for MBT where UML is used as test specification. As per survey in [16], majority of the MBT approaches use State chart diagram, UML, class diagram, sequence diagram and finite state machine for generation of test cases.

3.4 Tool support for model based testing

One of the most important aspects of MBT is level of automation. Automation means less cost, time and effort to generate test cases [16]. In [18] a systematic review of tool support for automation has been done. This study has been conducted on 27 MBT tools to find out what are tools available in the market, level of automation and test coverage criteria used in each tool. GOTCHA-TCBeans, MBT, TestOptimal, AGEDIS, ParTeg, Qtronic, Test Designer, Spec Explorer, NModel are some of the tools studied in literature.

3.5 Advantages and limitations of model based testing

Once the system model is created, model based testing can be executed with lesser effort. This ensures early integration of testing in development process resulting in increased quality.
Automatic test case generation is time and cost effective. After creating test model, test generation can be automated.
In case of changes in the system design, effort for regression testing is very less if model based testing is used. Only work involved in model based testing is to change the system model and rest of the testing process can be automated [16].
Test goals can be defined in the test model. By doing this, a certain set of common test cases are already pre-created in the model. This ensures that common test cases are not missed out.
Manual creation of test cases and test scripts has limitations of cost and time to create the test cases. Because of automation of test case creation, model based testing provides better test coverage.
MBT makes software engineers able to accomplish testing activities in parallel with the development of software [16]

3.6 Model based testing in agile development

Agile development is a software development methodology focusing on shorter development iterations aimed at delivering working software to users. In [21] author has argued that Agile development and Model Based Testing complement each other. Using MBT in agile development helps in better test coverage, improves flexibility and helps in automated Test driven development.

4. Model based testing for Datawarehouse

In this section we discuss a model based testing approach for datawarehouse testing.
Majority of model based testing approaches have been applied for UML based object oriented modeling software. In [16] a survey of model based testing has been done and the survey concludes that model based testing is most widely applied to

object oriented software followed by that for reactive systems and safety critical systems. The survey reports application of MBT for embedded systems and web applications, but application of MBT to database systems or datawarehouse applications in particular are not reported.

4.1 Motivating example:

We discuss proposed approach for model based testing of datawarehouse with the help of an example. Consider a data source including 2 tables: Retail_sales_store and Retail_sales_online that includes transactions for sales done in the store and sales done through online portal respectively. Details of items sold are stored in Item table. We would like to populate this transaction level data in a fact table called as Fact_retail_transaction. Structures of these tables is shown below

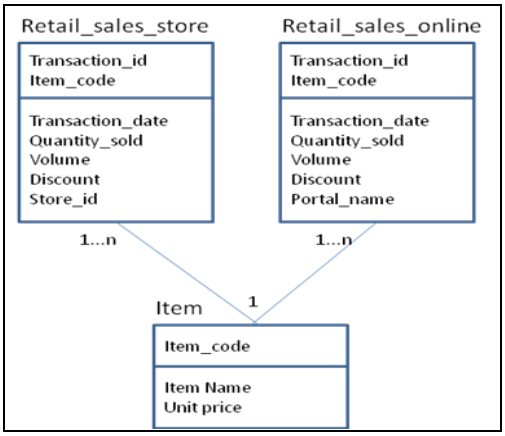

Fig. 2 Motivating example – source tables

4.2 Model of SUT for testing

One of the most important activities in MBT is building model of SUT for the purpose of testing. We propose to use a Data Mapping in UML using attributes as first class citizen of the model. In [22] a UML based approach for modeling data flow has been proposed. In this paper 4 levels of data flow have been proposed: database level mapping, dataflow level mapping, table level mapping and attribute level mapping. We propose to make use of attribute level and table level mappings to model a data flow to construct model of SUT. Table level mapping model will capture relationship between source and datawarehouse tables. Attribute level mapping model will capture relation between source and target attributes and transformations involved. We propose to modify the approach proposed in [22] to include <<Transformation>> class to model relation between 2 attributes. Methods in <<Transformation>> class will capture details of transformation between source and target elements. Model of SUT for the motivating example is shown in figure 3 and 4.

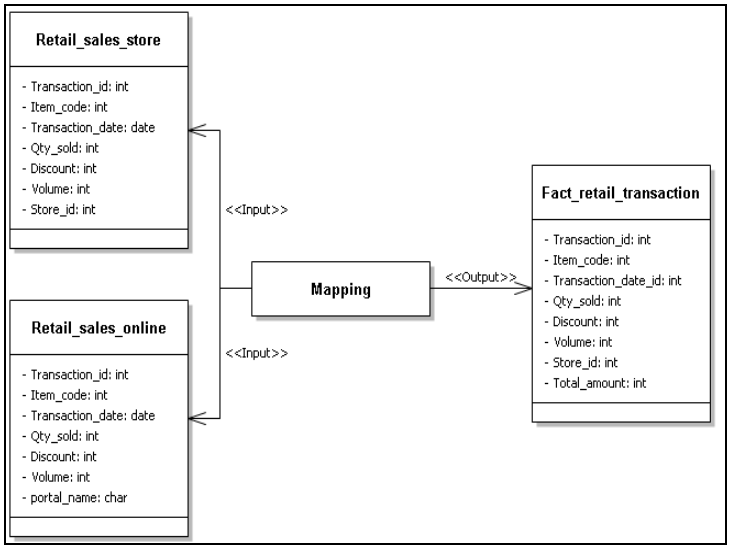

Fig. 3 Model of SUT – table level mapping

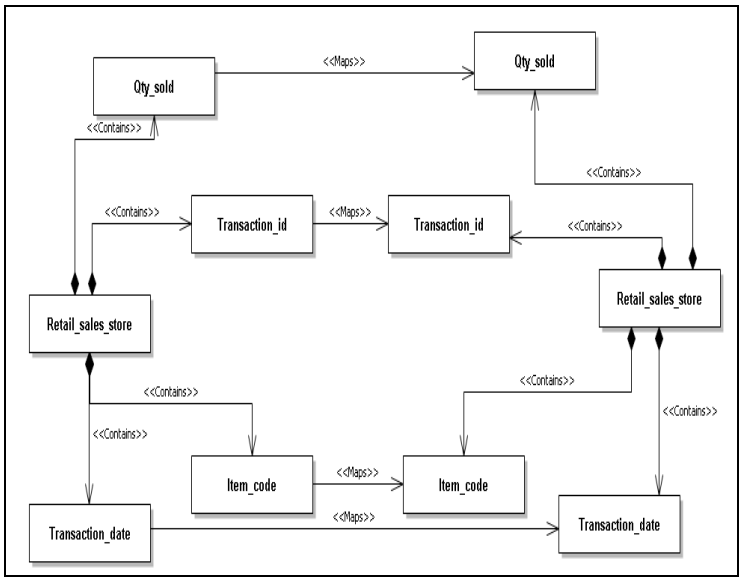

Fig. 4 Model of SUT – Attribute level mapping

4.3 Test data generation

Lack of sufficient data is a major challenge in Datawarehouse testing. We propose to make use of equivalence partitioning techniques for generation of data. This is a technique in which input domain of values is divided in categories and test cases are designed in such a way that each category of data is covered at least once. During creation of SUT model, information regarding categories of data will be captured from the business user and random data generation techniques will

be used for generation of test data in each class. Categories of data for testing should include valid as well as invalid data. This generated test data will be used in case source data is not available. In the motivating example, following is an example of capturing data categories.

Column	Min. Value	Max. Value	Valid / Invalid	Data to be generated
Discount	0	100	Valid	90%
Discount	-10	-1	Invalid	5%
Discount	101	150	Invalid	5%

Fig. 5 Data categories for test data generation

In the above example, during equivalence partition based data generation, 90% of the generated data will be in the range 0 to 100 and 10% of the values generated for testing will be invalid values (less than 0 and greater than 100)

4.4 Test case generation

We propose to generate test cases based on UML models described in section 4.2. Our approach is based on approach described in [23]. Different UML diagramming tools such as MagicDraw, Rational Rose and NClass allow UML model to be exported to XML file. The exported file contains XML tags describing UML diagram. In our proposed approach, each mapping between source and target attribute and tables is modeled as a class. Each such class will contain transformations (e.g. conversion from Euro to dollars, mathematical operation between 2 attributes etc) as methods in the class. Based on the XML tags which mainly describe type of transformation between source and target tables and involved attributes, a standard set of test cases is proposed to be generated using a parser. Standard type of testing that can be applied in datawarehouse testing can be categorized into following types [25]:

- Constraint testing – objective is to validate uniqueness, relationships etc.
- Source to target counts – objective is to validate of record count in source matches record count in target and that ETL process does not drop any records.
- Source to target data validations – objective is to ensure that data transformations are correctly applied on the source attributes before loading into target.
- Error processing – objective is to ensure that invalid records in the source table are correctly moved to appropriate error table.

Our proposed parser for test case generation will read each XML tag, interpret the type of transformation and attributes involved and apply testing validations described above to the involved attributes. In the motivating example, table level mapping generates a class "Public class mapping" indicating that there exists a mapping between retail_sales_store, retail-sales_online and fact_retail_transaction tables. The XML parser will generate a test case to validate that number of records in source and target tables is same.
Datawarehouse test cases may take long time to execute [15] considering high data volumes. Hence we propose that model based testing for DW should be an offline testing in which user can control execution of test cases.

Process flow of proposed model based testing based on methodology discussed in this paper is shown in figure 6

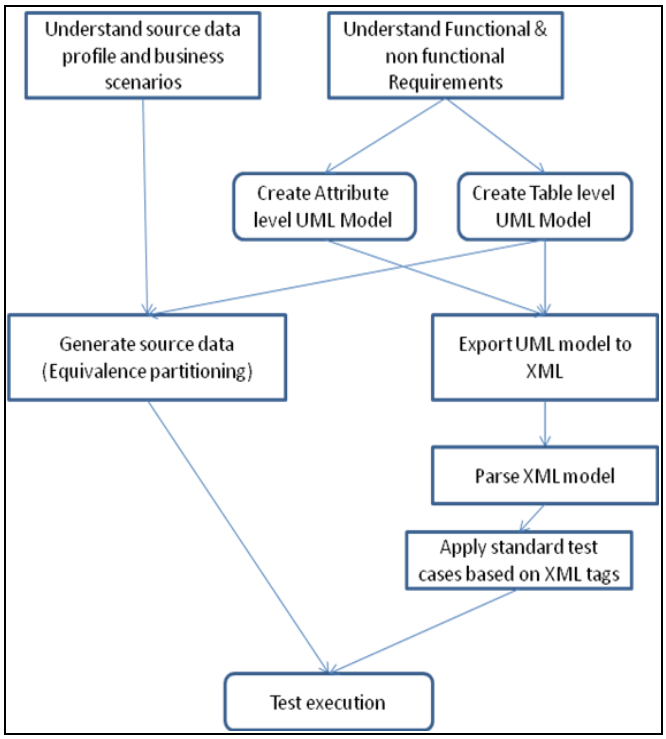

Fig. 6 Process flow of model based testing of datawarehouse

4.5 Advantages of proposed approach

- Lack of source data for testing, lack of richness in test data has been reported as a major challenge in DW testing. As seen in above architecture, our proposed MBT approach includes a test data generator. The test data generated for testing is proposed to include all possible valid and invalid data categories. This can help in generating required quantity of test data with possible data value

combinations. Also by using business user feedback for creating data partitions, this approach generates test data that mimics real life scenarios.

- Limited time for testing is a major issue in DW testing due to large number of test case combinations, need to understand business domain to create test cases and lack of good quality source data for testing [15]. With Model based testing, task of test case generation is automated and large number of combinations of test cases can be generated in a limited period.
- Changing nature of requirements is a feature of Datawarehouse development. This is because business cannot finalize data reporting requirements unless they see the data which is possible only after datawarehouse development. This uncertain nature of Datawarehouse requirements leads to frequent changes in specifications and subsequent need for changes in test cases. With model based testing, a change in test model can result in automated change in test cases and generation and execution of test cases.

5. Future roadmap for model based testing for datawarehouse

In this paper we have provided, to the best of our knowledge, first approach based on model based testing of datawarehouse. Model based testing approach described in this paper needs to be experimentally established. For this, it is important that an extension of UML needs to be created for representing various transformations used in Datawarehousing process. Test data generation is an important aspect of our proposed approach. Creation of rich set of test data avoiding combinational explosion of data needs to be done.

Proposed approach needs to be empirically validated and it needs to be verified whether model based testing results in reduced effort for DW testing and also results in better test coverage.

6. Conclusions

In this paper we reviewed state of the art of datawarehouse testing and challenges faced in it. With increasing business expectations regarding content and accuracy of datawarehouse data and tighter budgets, newer techniques for testing need to be developed. In this paper we have proposed application of model based testing for testing of datawarehouse. We believe that usage of models of software under test for datawarehouse for testing purpose will reduce testing effort and cost and increase code coverage and hence quality of datawarehouse.

References

[1] A 50% Datawarehouse failure rate is nothing new available at http://it.toolbox.com/blogs/bounded-rationality/a-50-data-warehouse-failure-rate-is-nothingnew-4669

[2] Moez Essaidi and Aomar Osmani, A Unified Method for Developing Data Warehouses, Journal of Computational Methods in Sciences and Engineering, Volume 10 Issue 1-2S1, 2010,Pages 119-134

[3] Matteo Golfarelli and Stefano Rizzi, A Comprehensive Approach to Data Warehouse Testing, in DOLAP '09 Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP,2009, Pages 17-24

[4] A. Mookerjea and P. Malisetty. Best practices in datawarehouse testing. In Proc. Test, New Delhi, 2008.

[5] Jaiteg Singh and Kawaljeet Singh,Statistically Analyzing the Impact of Automated ETL Testing on the Data Quality of a Data Warehouse, International Journal of Computer and Electrical Engineering, Vol. 1, No. 4, 2009, Pages 488-495

[6] Mohd. Ehmer Khan,Different Forms of Software Testing Techniques for Finding Errors, International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, 2010, Pages 11-16

[7] Mark Utting,Position Paper: Model-Based Testing, in Verified Software: Theories, Tools, Experiments (VSTTE) conference on the "Program Verifier" grand challenge, 2005

[8] Mitu Dhull and Archana Sharma,Testing the relational Database,International Journal of Computer Science Issues, Vol. 7, Issue 3, No 5, 2010, Pages 47-52

[9] Pavol Tanuska, Pavel Vazan and Peter Schreiber,The Partial Proposal of Data Warehouse Testing Task, in 2009 International Symposium on Computing, Communication, and Control, 243-247

[10] Utting, M., Pretschner and A. and Legeard, B., A taxonomy of model-based testing approaches, Softw. Test. Verif. Reliab., Volume 22, Issue 5, pages 297–312, August 2012

[11] Raj Kamal and Nakul , Adventures with Testing BI/DW Application:On a crusade to find the Holy Grail, Available from <http://msdn.microsoft.com/en-us/library/gg248101.aspx>, [31st Jan 2012].

[12] Prof. Dr. Marc Scholl and Prof. Dr. Harald Reiterer,Model-Driven Architecture (MDA) and Data Warehouse Design, in seminar Business Intelligence Model-Driven Architecture (MDA) and Data Warehouse Design, 2006

[13] Golfarelli and Matteo.,From User Requirements to Conceptual Design in Warehouse Design: A Survey., In Data Warehousing Design and Advanced
Engineering Applications: Methods for Complex Construction, ed. Ladjel Bellatreche, 2010, Pages 1-16

[14] R. Kimball and J. Caserta, The Data Warehouse ETL Toolkit. John Wiley & Sons, Indianapolis, 2004.

[15] Muhammad Shahan Ali Khan and Ahmad ElMadi, Data Warehouse Testing an Exploratory Study, MS Thesis, School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden, 2011

[16] A.C. Dias-Neto, R. Subramanyan, M. Vieira and G.H. Travassos, Characterization of Model-based Software Testing Approaches, TRES-713/07, PESC-COPPE/UFRJ, 2007.
[17] Pavol Tanuška, Oliver Moravčík, Pavel Važan, Fratišek Miksa, The Proposal of Data Warehouse Testing Activities, in Proceedings of the 20th Central European Conference on Information and Intelligent Systems, 2009, Pages 7-11
[18] Muhammad Shafique and Yvan Labiche, A Systematic Review of Model Based Testing Tool Support, Carleton University, Technical Report, 2010
[19] Panos Vassiliadis, Alkis Simitsis and Spiros Skiadopoulos, Conceptual modeling for ETL processes, in DOLAP'02, 2002, Pages 14-21
[20] Object Management Group 2003, CWM 1.1 Available from: <http://www.omg.org/spec/CWM/1.1/>, [31st Jan 2012].
[21] Faragó, David. "Model-based Testing in Agile Software Development.", in 30. Treffen der GI-Fachgruppe Test, Analyse & Verifikation von Software (TAV), Testing meets Agility, 2010, Pages 1-4
[22] Sergio Luján-Mora, Panos Vassiliadis and Juan Trujillo, Data Mapping Diagrams for Data Warehouse Design with UML, in In Proc. 23rd International Conference on Conceptual Modeling, 2004, Pages 191-204
[23] Vinaya Sawant and Ketan Shah, Automatic Generation of Test Cases from UML Models, in International Conference on Technology Systems and Management (ICTSM) 2011 Proceedings published by International Journal of Computer Applications® (IJCA), 2011, Pages 7-10
[24] Matteo Golfarelli, From User Requirements to Conceptual Design in Data Warehouse Design – a Survey, In Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction. L. Bellatreche (Ed.), IGI Global, 2009, Pages 1-14
[25] R. Cooper and S. Arbuckle. How to thoroughly test a data warehouse, in Proc. STAREAST, Orlando, 2002.
[26] Adaptive, Available from: <http://www.adaptive.com/>, [31st Jan 2012].
[27] Meta Integration Technology Inc, Available from: <http://www.metaintegration.net/>, [31st Jan 2012].
[28] Kalido Inc, Available from: <www.kalido.com/>, [31st Jan 2012].

Kuldeep Deshpande completed BE (Mechanical Engineering) from Pune university in 1997 and MS (Software Systems) from BITS Pilani. He is pursuing Ph.D in Information systems management from ITM University. Since 2000, he has worked in Capgemini consulting in the Datawarehousing and Business Intelligence practice.