Model Transformations Carried by the Traceability Framework for Enterprises in Software Industry

Gullelala Jadoon[1], Muhammad Shafi[2], and Sadaqat Jan[3]
[1]Department of Information Technology, University of Haripur, Pakistan
[2]Faculty of Computing and Information Technology, Sohar University, Oman
[3]University of Engineering and Technology, Mardan, Pakistan

Abstract: *The developmental paradigm in the software engineering industry has transformed from a programming-oriented approach to model-oriented development. At present, model-based development is becoming an emerging method for enterprises for constructing software systems and services most proficiently. In Capability Maturity Model Integration (CMMI) Level 2, i.e., Managed, we need to sustain the bi-directional trace of the transformed models for the administration of user requirements and demands. This goal is achieved by the organization after applying the particular practices suggested by CMMI level 2 process area of Requirements Management (RM). It is very challenging for software developers and testers to maintain trace, particularly during the evaluation and upgrading phases of development. In our previous research work, we proposed a traceability framework for model-based development of applications for software enterprises. This work is the extension of our previously presented research work in which we have anticipated the meta-model transformations according to the Software Development Life Cycle (SDLC). These meta-models are capable of maintaining the trace information through relations. The proposed technique is also verified using a generalized illustration of an application. This transformation practice will give a foundation to software designers to maintain traceability links in model-driven development.*

Keywords: *Requirements Management, traceability, Model-driven, SDLC, CMMI.*

Received February 23, 2020; accepted June 9, 2020
https://doi.org/10.34028/iajit/17/4A/1

1. Introduction

Requirements Management (RM) is the foundation of software development, as well as the progressive stages where there is a conflict between clashing requirements. That is the reason RM practices and procedures assume a first job in resolving these disputations. RM is generally an isolated activity and should be transformed and unified at the later development steps. As specified by the software experts, presently, there is no authentic answer for the joined administration of shared artefacts in RM.

Accordingly, we have to recommend a methodology to apply reasonable RM practices to catch the particular professional needs of such business systems. Any Requirement Management practice must be realized as acceptable if it reserves the time and cost; a client is devoting. Farther, method assessment can help organizations satisfactorily distinguish practices to distinguish what is the foremost demand of customers and concluding innovative practises for their requests. The consolidation of the developing constituents into the Quality Function Deployment (QFD) exploits these specifications more visibly and understandably to pick up the primary thought to progress development processes. Earlier work from this research area demonstrates that such models can be viably merged to distinguish client requirements and enhance system development. The Product-Service System (PSS) has revolved into the most extraordinary practice in developing smart data systems and services, which include extensive requirements techniques. This indication comprises the administration of different measurements, for example, platform requirements, performance, smartness, scalability, customization, and client agreement. A substantial trial is to deal with these properties at the primary step of requirements designing and to determine the insight of PSS related clashes on requirements [28, 38]. Figure 1 shows different practices involved at level 2 of Capability Maturity Model Integration (CMMI) and their names.

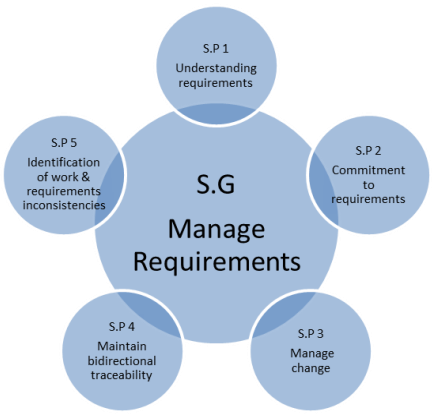

Figure 1. Requirements management.

The activities of RM are executed at the 'Managed' level of CMMI, i.e., level 2. Numerous software companies at this step try to follow these practices; yet, they don't fast-track to process innovation. Such firms need evolvement in improving their schemes and applying inventive techniques in their procedures to make them beneficial [30]. Attaining stage 2 of CMMI implicates RM by employing automated and manual ways of management. Such practises containing traceability enhancement, updation activities, and software quality measurement. A wide-ranging scope of such practices is under preparation for this purpose [8]. The drive of this research is to analyze the previous model-oriented traceability framework and propose techniques to transform different models while keeping the trace links of each meta-model. It also focuses on discussing the model, identifying outcomes, and suggesting the appropriate method for transformation. Our research extends our previously presented work based on the traceability framework designed for model-oriented software development [14]. In the current work, the following research statements are addressed:

1. We are providing an in-depth analysis of previously proposed requirements traceability models in the literature and their shortcomings.
2. We are analyzing the effectiveness of the proposed MDD framework to conclude the traceability of requirements for the software industry.
3. The performance assessment of the proposed framework using a real-time application.

The rest of the paper is structured as follows: section 2 comprises the literature review, section 3 is based on the proposed framework, section 4 covers experiments and results while section 5 concludes the paper with future directions of the research.

2. Literature Review

Buglione *et al*. [5] concluded that software companies are constantly working on their process maturity by following different system management models. These models can illustrate the process maturity at all phases of system development. Using these maturity models, all the Software Development Life Cycle (SDLC) activities are refined as CMMI describes specific goals and practices to be followed at each developmental stage. Unal *et al*. [41] concentrated their research study on ISO and CMMI process models to be adopted for standard software development. Gonçalves *et al*. [11] also worked on CMMI standard and concluded that it is the commonly used evaluation means in software production. CMMI DEV provides best practices for developers to advance and innovate their processes according to the will of customers. Keshta *et al*. [18] and Souali *et al*. [39] proposed different frameworks for the implementation of RM practices, especially the trace related features. These trace models have been assessed using real application data of software houses. Automated tools such as StarUML, DOORS, Requirements Requisite Pro, RUT, Capra, and IBM Rational Rose are used for RM activities, including traceability maintenance [17, 27]. These tools provide requirements traceability coverage for the source code based development and support code alteration, its performance precision, and reliability with the user-defined requirements [29, 36]. As far as the overhead of the application's multi-platform is addressed, Dubois *et al*. [7] proposed DARWIN4REQ. It is a requirements traceability tool capable of preserving the trace links of heterogeneous meta-models. In order to target a specific functionality of the application for maintenance or evaluation, we need to localize that component. It is possible only if we have the complete links to that component in forwarding and backward directions. According to a research study conducted by Li *et al*. [22] there should be a defined trace criterion on which the whole system is evaluated. This criterion is essential for trace update and retrieval. Among the trace methods, the graph-based methodology proposed by Schwarz *et al*. [36] has better performance in the case of model-oriented systems but with some limitations. For better organization of the code, refactoring and semantic coherence are highly recommended by the software experts as they yield a sound quality system.

Mahmoud *et al*. [25] and Guo *et al*. [12] used refactoring and semantics analysis on the source code to make it consistent to incorporate better traceability. Traceability is viewed as a significant attribute to enhance the developmental and maintenance related activities both in formal and informal modelling [19, 34]. Presently lean and agile development methodologies known as incremental strategies are widely used in software industries because of their certain advantages. Paige *et al*. [31] recommended that due to the continuous change in code; in lean development, the trace links denigrate. International Standard Organization recommends 9001 and 90003 quality standards to be applied to preserve the refactoring and consistency of the source code development. Hegedus *et al*. [13] suggested a query-based traceability model for model-based development. In this method, agile development queries are employed to link the meta-models to localize a module. ICONIX paradigm proposed by Malinao *et al*. [26] uses Unified Models for the trace of the meta-models like use cases using robustness diagram. This paradigm uses depth-first search algorithm for design-driven testing in use cases and class diagrams. Other requirements quality matrices such as completeness, consistency, clarity, and correctness are also used to evaluate software performance using traceability links [23, 26].

Different requirements traceability techniques have been proposed by scientists to improve validation and maintenance of software. Potdar and Routroy [32]

employed a dynamic grouping of the code units to maintain change traces in agile modelling due to rapidly changing requirements. He suggested agile manufacturing enablers, joint with a non-textual traceability practise, typically used in an agile environment. Tariq *et al*. [40] proposed trace linkage techniques where activities are traced through mapping. Jyoti and Chhabra [15] proposed a non-textual trace technique which is uni directional. Though, it offers trace in both the directions, the links denigrates while mapping code to the outcomes in case of large number of modules. Samalikova *et al*. [34] emphasized the oral assessment activities and performed a survey to represent trace throughout the appraisal process. Still, the results display that it is a uni-directional methodology for trace. Gonçalves *et al*. [11] concerted on following HCI codes in handling requirements, but it is confined to the user interface of application. Keshta *et al*. [18] has analyzed the performance of systems after several alterations are made and proposed a traceability framework for processing change in requirements. A. Zanatta and Vilain [43] suggested agile scrum methods for incremental software development as they are lean and flexible, which guarantees links support between code and other meta-models. Garzas and Paulk *et al*. [9] suggested scrum mapping methodology to provide bi-directional trace following CMMI dev.

Filion *et al*. [8] designed Atlassian JIRA an automated trace tool that provides a discrete bi-directional trace for bug tracking and requirements variation management. A practice used by visual metaphors has been suggested by L. Buglione *et al*. [5] which works by choosing specific development outcomes to combine. Violante *et al*. [42] suggested QFD methodology to achieve CMMI RM goals using advanced tool customization. Qasaimeh and Abran [33] has recommended an evaluation method for code trace management. Beecham *et al*. [3] proposed a RM model yet; the study scope is limited to source code. Kahkonen and Abrahamsson [16] has suggested another traceability methodology for incremental eXtreme Programming (XP) development, limited to code artefact. Patrick Mader has joined SQL with VTML by means of UML models, especially UML class models and use cases, to generate trace requests [25]. Schwarz *et al*. [36] recommended another trace methodology called Graph-based traceability, but its preservation is complex in case of an additional line of code. Laghouaouta *et al*. [21] used soft goal arrangements to supplement models to the up-to-date requirements and store a trace hierarchy after thorough regression testing. Still, the proficiency got poorer because of an increase in response time. Aizenbud-Reshef *et al*. [1] suggested that the intricacy of MDP lies in the preservation of traceability relations while combining different transformed models. Sango [35] carried a research study for component-based traceability framework that utilized the UPPAL automated tool for validation that provides trace at the component level only. Seibel *et al*. [37] presented a trace model, which attained traceability both manually and in an automated way, but it is not a thorough mechanism. Chanda *et al*. [6] recommended a trace practise that maps system services to link them to UML classes in Service-Oriented Architecture (SOA). Gayer *et al*. [10] proposed a lightweight trace mechanism for incremental agile-based development suitable for small applications. Kahkonen and Abrahamsson [16] followed CMMI level 2 practices for agile XP approach for providing single- directional trace.

Throughout this literature review, we came across a number of trace techniques that are accompanied by some limitations including its incomplete coverage, text-based methodologies, source code oriented trace, time overhead, weak trace links, lack of meta-model trace and trace heterogeneity.

The uni-directional and bi-directional trace performance evaluation conceded after a systematic literature review is gathered in the form of the following assessment tables.

Table 1. Bi-directional traceability practices.

Paper	Technique used	Bi-directional Trace
Remapping Of CMMI Level 2 KPA's For Development Process Improvement. [25]	Mapping	Bidirectional traceability
Identifying High Perceived Value Practices Of CMMI 2 Empirical Study. [30]	Face-to-face questionnaire-based survey.	Bidirectional traceability
Toward Implementation Of Requirement Management Specific Practices. [18]	Workflow model for Requirement change management.	Bidirectional traceability.
A Case Study Of Process Improvement With CMMI Dev. And Scrum In Spanish Companies. [9]	Scrum methodology	Bidirectional traceability
Extending An Agile Method To Support RM And Development In Conformance To CMMI.[43]	Agile trace Development	Bidirectional traceability
Using The Expert Panel To Validate A Requirement Process Improvement Model. [3]	RE model	Bidirectional traceability
Using Atlassian Tools For Efficient Requirement Management. [5]	Atlassian JIRA, confluence Tool	Bidirectional traceability
The Lego Maturity And Capability Model Approach. [20]	A Visual component-based approach.	Bidirectional traceability
An Integrated Approach To Support The Requirement Management (RM) Tool Customization For A Collaborative Scenario. [42]	QFD, Kano's model and Tontini's method.	Bidirectional traceability
A Visual Language For Modelling And Executing Traceability Queries. [23]	Visual Language for Modeling bidirectional trace. (VTML)	Bidirectional traceability
A Dedicated Approach For Model Composition Traceability. [21]	Soft goal Trees.	Bidirectional traceability
Model Traceability. [1]	Model-Driven Traceability	Bidirectional traceability.

Also, the findings of the uni-directional trace are listed in Table 2.

Table 2. Uni-directional traceability practices.

Paper	Technique used	Uni-directional Trace
Requirement Traceability Through Retrieval Using Dynamic Integration. [15]	Automatic technique merged with non-textual based techniques.	Text-based uni-directional traceability.
Process Mining Support For Capability Maturity Model Integration Based Software Process Assessment. [34]	Interviews, oral audit sessions, process reviews, quality manuals.	Uni-directional traceability.
An Audit Model For ISO 9001 Traceability Requirements in Agile XP Environment. [33]	Code Evaluation.	Uni-directional code traceability.
Achieving CMMI Level 2 With Enhanced Extreme Programming Approach. [16]	An agile method, extreme programming.	Uni-directional traceability.
A Component-Based Model-Driven Approach With Traceability Of Concerns. [35]	UPPAL tool.	Uni-directional component traceability.
Dynamic Hierarchical Mega Models: Comprehensive Traceability And Its Efficient Maintenance. [4]	Model-Driven Automated and manual Traceability.	Comprehensive traceability is not covered.
Traceability Between Service Component And Class: A Model Based Approach. [6]	A model-oriented trace mechanism for Services.	Uni-directional traceability for service-based systems only.

3. Proposed Framework

In our previous work, we proposed a model-based traceability framework for MDD that is proficient of identifying and locating each requirement from Software Requirement Specification (SRS) to respective meta-model of the application because of the limitations in previous research studies [14]. The corresponding trace tables reserve the complete trace information. The framework represents different meta-models of Unified Modelling Language (UML) such as use case, class models, and ERD, etc., that are transformed to form a complete system, as shown in Figure 2.

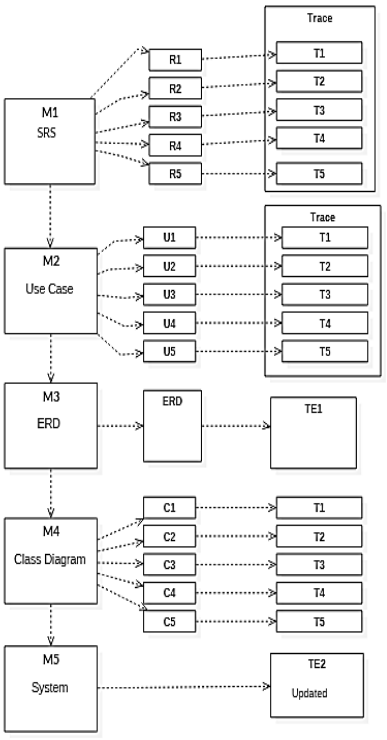

Figure 2. Proposed traceability framework.

In Figure 2, M1 is representing the first UML Model; M2 is identifying the second UML Model, while M3 is representing the third UML Model, and so on. M1 is the SRS document which is representing five requirements R1, R2, R3, R4, and R5 represented by trace T1, T2, T3, T4, and T5. M1 model is transformed into M2, which is the UML use-case model. Here we have five use case models U1, U2, U3, U4, and U5 representing the particular requirements and trace numbers. Similarly, the M2 model is transformed into M3, which is the entity-relationship diagram. It results in the comprehensive traceability of the ERD. M3 model is transformed into M4, which represents the class diagrams of the system C1, C2, C3, C4, and C5. The traces of these diagrams are updated in the trace table as T1, T2, T3, T4, and T5, respectively. M4 model is transformed into M5, which is the system generated from the UML models. Its generalized trace is represented by TE2, as shown in the framework above. We can further extend this framework up to several transformed models where the respective trace information is recorded in the corresponding trace table.

4. Model Transformations

As we know that in model-driven development, the initial model is transformed into a number of meta-models, as shown in Figure 3, where each derived model represents specific developmental information. The transformations are represented by T1, T2, T3, and T4.

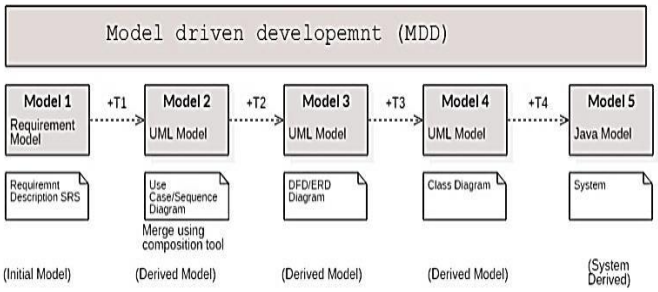

Figure 3. Meta-Model transformations.

In Figure 2, Model 1 represents the Software Requirements specification, Model 2 represents the Usecase/Sequence models, Model 3 represents Entity Relationship Diagram, Model 4 represents Class Diagrams, and Model 5 represents the required system model which is a platform-specific model of the desired application.

5. Experiments and Results

To validate the performance of our proposed trace model, we have selected an android application Mathematical Logic developed by Kuczynski [20], a professor at the University of California. The primary user interface of the application is depicted in Figure 4.

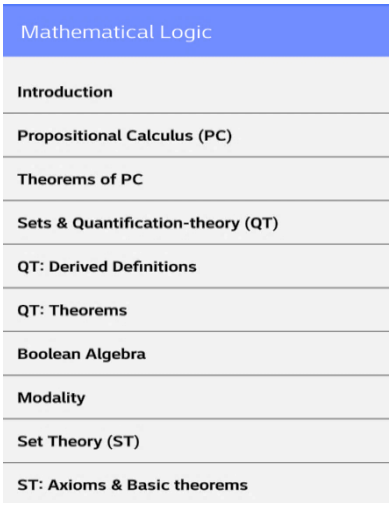

Figure 4. Mathematical logic application.

The application is developed to aid the students in understanding basic mathematical concepts like Calculus, Boolean algebra, and important theorems. The suggested traceability framework is applied to the application mentioned above and generated the following meta-model transformations shown in Figures 5, 6, 7, 8, and 9.

In Figure 5, M1 is presented that depicts the core requirements of the application. Each requirement is mapped to its related trace. M1 is Platform independent model showing the SRS.

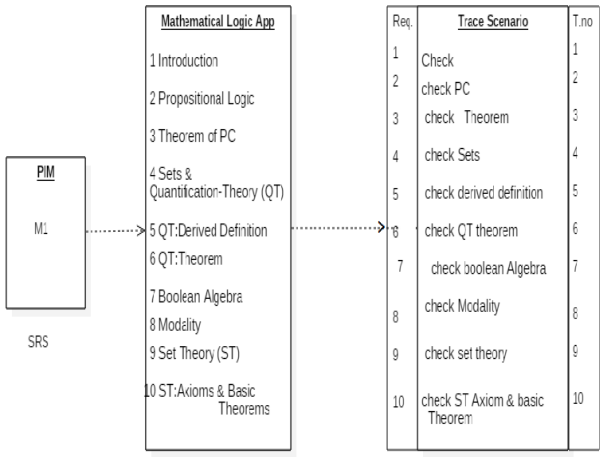

Figure 5. Model 1.

In Figure 6, M2 is presented that is the use case model of the application. Each requirement is represented by a use case which is further mapped to the trace table.

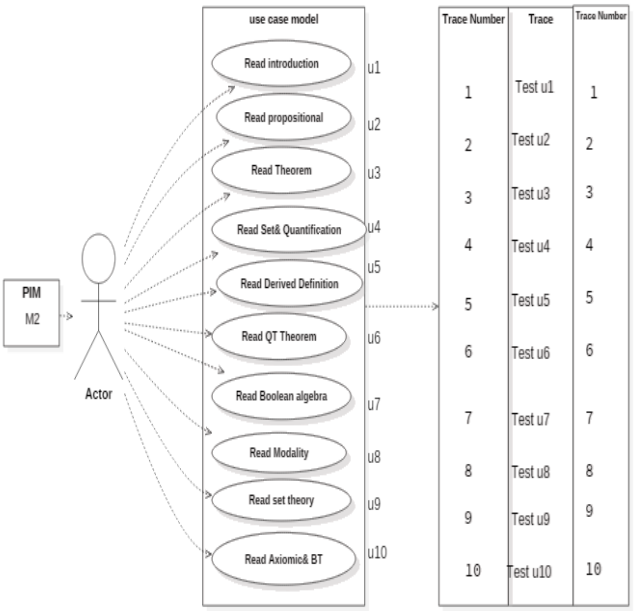

Figure 6. Model 2.

In Figure 7, M3 is presented, which is the entity-relationship diagram of the application. The basic entities are represented as E1, E2, and E3. In the trace table, each requirement is mapped to its previous trace number, followed by the entity information in the third column of the table. It is clear that the previous trace information is preserved in the upcoming tables, and the trace links are maintained, which helps during software testing. In the ER Diagram, we can put all the necessary details used to understand that entity and can also show the relationships among all the entities. This will help to understand the complete functionality of the application.

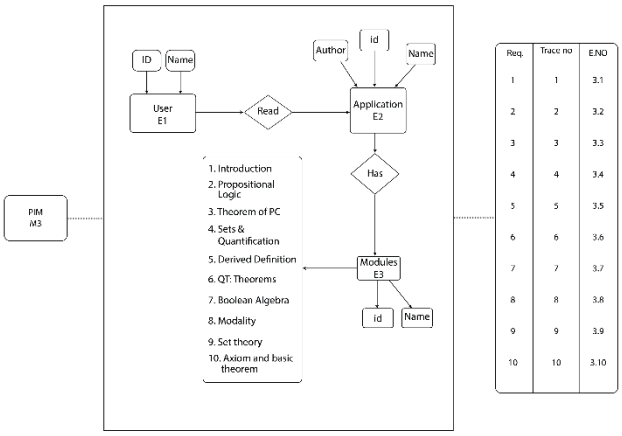

Figure 7. Model 3.

In Figure 8 below, M4 is represented, which shows the class diagrams of the application. The trace table has also been updated to map the respective classes.

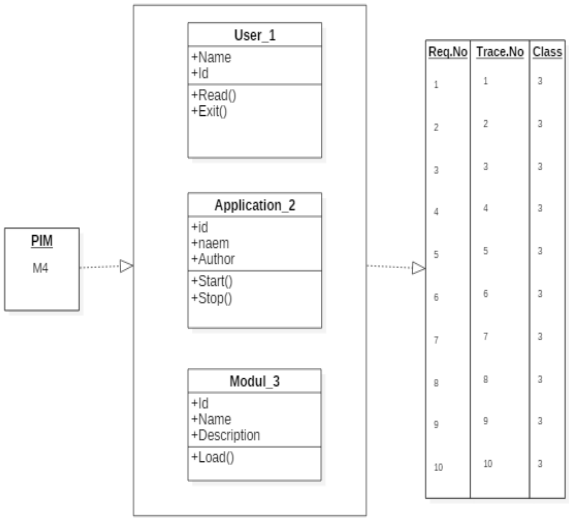

Figure 8. Model 4.

In Figure 9, M5 is shown, which is the required android application platform-specific model. The trace table shows the requirement, followed by the trace number, corresponding class, and entity. In this way, all the link information is maintained.

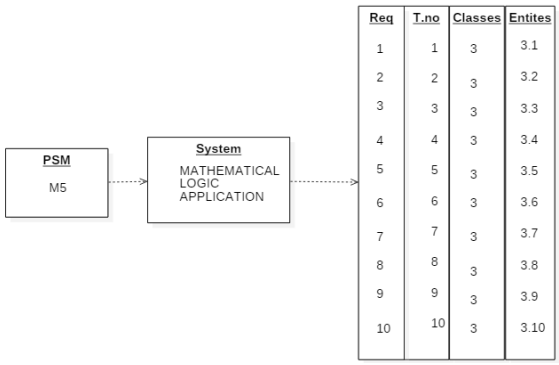

Figure 9. Model 5.

With the help of this traceability framework, we can sustain the links of previous meta-models. This traceability of previous and fore coming models is very useful for verification purposes and also during the software maintenance phase. The trace tables provided are used to record the exact links of all the modules throughout the application. This traceability is traversed in both the forward and backward directions. From the above experiment, it is clear that the proposed framework is most suitable for model-oriented applications as compared to the ordinary agile applications. In Figure 10, we have shown the traceability coverage in agile and model-driven projects based on the previous study.

In our earlier research, we applied the proposed framework on five agile-based projects. We measured the percentage of traceability at the phases of requirements engineering, architecture, and design, detailed design, modelling, development, and coding and maintenance. It shows 90% trace coverage at the stages of development and coding.

Similarly, the framework is applied to the remaining model-based projects, and the trace coverage percentage on the various developmental phases is analyzed. The trace results yield better performance in the case of model-based applications, i.e., up to 95%. The only step that showed less performance is the development and coding phase. This is because the traceability framework is a UML based methodology that does not cover the source code based trace.

In Figure 10, the trace performance comparison of both the agile development and model-based development are combined. This graph shows that the trace convergence is least in agile development as compared to the other one. The traditional development shows better results in source code development while in model-based development; it is less at this phase. This is because there is less emphasis on source code in model-based development, and the system is generated through model transformations.

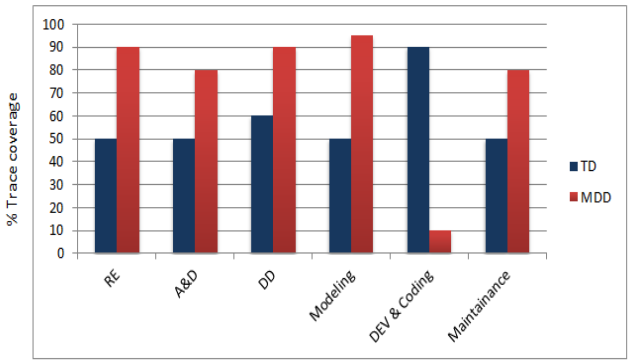

Figure 10. MDD and traditional development.

6. Conclusions

From this research study, we can conclude that the performance of our suggested framework has shown improved results for small and medium model-driven software systems. In the case of traditional development, the trace links are hard to update by the

developers whenever any modifications are made to the system. This results in a weak quality system that doesn't provide maintenance support and affects the reputation of the enterprise.

On the other hand, the experiments proved that model-based systems are easy to maintain and traverse because there is minimal code development, and all the development artefacts are models. Our proposed framework can be applied to any model-based application designed by the organization to generate and maintain the trace information very efficiently and effectively. The future work comprises refining this model-based framework to make it more applicable and efficient to larger software systems. Moreover, we can further generate the trace results of different applications in different small and medium enterprises and compare trace efficiency.

References

[1] Aizenbud-Reshef N., Nolan B., Rubin J., and Shaham-Gafni Y., "Model Traceability," *IBM Systems Journal*, vol. 45, no. 3, pp. 515-526, 2006.

[2] Babar M., Khattak A., Arif F., and Tariq M., "An Improved Framework for Modelling Data Warehouse Systems Using UML Profile," *The International Arab Journal of Information Technology*, vol. 17, no. 4, pp. 562-571, 2020.

[3] Beecham S., Hall T., Britton C., Cottee M., and Rainer A., "Using an Expert Panel to Validate a Requirements Process Improvement Model," *Journal of Systems and Software*, vol. 76, no. 3, pp. 251-275, 2005.

[4] Bokhari M. and Siddiqui S., "Metrics for Requirements Engineering and Automated Requirements Tools," *in Proceedings of 5th National Conference Computing For Nation Development*, New Delhi, 2011.

[5] Buglione L., Wangenheim C., Hauck J., and McCaffery F., "The LEGO Maturity and Capability Model Approach," *in Proceedings of 5th World Congress for Software Quality*, Shanghai, 2011.

[6] Chanda J., Sengupta S., Kanjilal A., and Bhattacharya S., "Traceability Between Service Component and Class: a Model Based Approach," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 6, pp. 1-5, 2012.

[7] Dubois H., Peraldi-Frati M., and Lakhal F., "A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems," *in Proceedings of 15th IEEE International Conference on Engineering of Complex Computer Systems*, Oxford, pp. 233-242, 2010.

[8] Filion L., Daviot N., Bel J., and Gagnon M., "Using Atlassian Tools for Efficient Requirements Management: an Industrial Case Study," *in Proceedings of Annual IEEE International Systems Conference*, Montreal, pp. 1-6, 2017.

[9] Garzas J. and Paulk M., "A Case Study of Software Process Improvement with CMMI-DEV and Scrum in Spanish Companies," *Journal of Software: Evolution and Process*, vol. 25, no. 12, pp. 1325-1333, 2013.

[10] Gayer S., Herrmann A., Keuler T., Riebisch M., and Antonino P., "Lightweight Traceability for the Agile Architect," *Computer*, vol. 49, no. 5, pp. 64-71, 2016.

[11] Gonçalves T., Oliveira K., and Kolski C., "A Study About HCI in Practice of Interactive System Development Using CMMI-DEV," *in Proceedings of 29th Conference on l'Interaction Homme-Machine*, Poitiers, pp. 169-177, 2017.

[12] Guo J., Cheng J., and Cleland-Huang J., "Semantically Enhanced Software Traceability Using Deep Learning Techniques," *in Proceedings of IEEE/ACM 39th International Conference on Software Engineering*, Buenos Aires, pp. 3-14, 2017.

[13] Hegedus A., Horvath A., Rath I., Starr R., and Varro D., "Query-Driven Soft Traceability Links For Models," *Software and Systems Modeling*, vol. 15, no. 3, pp. 733-756, 2016.

[14] Jadoon G., Shafi M., and Jan S., "A Model-Oriented Requirements Traceability Framework for Small and Medium Software Industries," *in Proceedings of International Arab Conference on Information Technology*, Al Ain, pp. 91-96, 2019.

[15] Jyoti and Chhabra J., "Requirements Traceability Through Information Retrieval Using Dynamic Integration of Structural and Co-change Coupling," *in Proceedings of 1st International Conference on Advanced Informatics for Computing Research*, Jalandhar, pp. 107-118, 2017.

[16] Kahkonen T. and Abrahamsson P., "Achieving CMMI Level 2 with Enhanced Extreme Programming Approach," *in Proceedings of International Conference on Product Focused Software Process Improvement*, Kansai Science City, pp. 378-392, 2004.

[17] Kamalabalan K., Uruththirakodeeswaran T., Thiyagalingam G., Wijesinghe D., Perera I., Meedeniya D., and Balasubramaniam D., "Tool Support for Traceability of Software Artefacts," *in Proceedings of Moratuwa Engineering Research Conference*, Moratuwa, pp. 318-323, 2015.

[18] Keshta I., Niazi M., and Alshayeb M., "Towards Implementation of Requirements Management Specific Practices (SP1.3 and SP1.4) for Saudi Arabian Small and Medium Sized Software Development Organizations," *IEEE Access*, vol. 5, pp. 24162-24183, 2017.

[19] Kleffmann M., Rohl S., Gruhn V., and Book M., "Establishing and Navigating Trace Links between Elements of Informal Diagram Sketches," *in Proceedings of IEEE/ACM 8th International Symposium on Software and Systems Traceability*, Florence, pp. 1-7, 2015.

[20] Kuczynski J., "Mathematical Logic," Google Playstore, Available at: https://play.google.com/store/apps/detail?id=com.shinwari.mathematicallogic, Last Visited, 2019.

[21] Laghouaouta Y., Anwar A., Nassar M., and Coulette B., "A Dedicated Approach for Model Composition Traceability," *Information and Software Technology*, vol. 91, pp. 142-159, 2017.

[22] Li Z., Chen M., Huang L., Ng V., and Geng R., "Tracing Requirements in Software Design," *in Proceedings of International Conference on Software and System Process*, Paris, pp. 25-29, 2017.

[23] Mader P. and Cleland-Huang J., "A Visual Language for Modeling and Executing Traceability Queries," *Software and Systems Modeling*, vol. 12, no. 3, pp. 537-553, 2013.

[24] Mader P. and Egyed A., "Do Developers Benefit From Requirements Traceability When Evolving And Maintaining A Software System?," *Empirical Software Engineering*, vol. 20, no. 2, pp. 413-441, 2015.

[25] Mahmoud A. and Niu N., "Supporting Requirements to Code Traceability Through Refactoring," *Requirements Engineering*, vol. 19, no. 3, pp. 309-329, 2014.

[26] Malinao J., Tiu K., Lozano L., Pascua S., Chua R., Magboo M., and Caro J., "A Metric for User Requirements Traceability in Sequence, Class Diagrams, and Lines-Of-Code via Robustness Diagrams," *Theory and Practice of Computation*, Springer, 2013.

[27] Maro S. and Steghofer J., "Capra: A Configurable and Extendable Traceability Management Tool," *in Proceedings of IEEE 24th International Requirements Engineering Conference*, Beijing, pp. 407-408, 2016.

[28] Mavin A., Wilkinson P., Teufl S., Femmer H., Eckhardt J., and Mund J., "Does Goal-Oriented Requirements Engineering Achieve Its Goal?," *in Proceedings of 25th IEEE International Requirements Engineering Conference*, Lisbon, pp. 174-183, 2017.

[29] Nassar B. and Scandariato R., "Traceability Metrics as Early Predictors of Software Defects," *in Proceedings of IEEE International Conference on Software Architecture*, Gothenburg, pp. 235-238, 2017.

[30] Niazi M. and Babar M., "Identifying High Perceived Value Practices of CMMI Level 2: An Empirical Study," *Information and Software Technology*, vol. 51, no. 8, pp. 1231-1243, 2009.

[31] Paige R., Matragkas N., and Rose L., "Evolving models in Model-Driven Engineering: State-of-the-art and Future Challenges," *Journal of Systems and Software*, vol. 111, pp. 272-280, 2016.

[32] Potdar P. and Routroy S., "Analysis of Agile Manufacturing Enablers: A Case Study," *Materials Today: Proceedings*, vol. 5, no. 2, pp. 4008-4015, 2018.

[33] Qasaimeh M. and Abran A., "An Audit Model for ISO 9001 Traceability Requirements in Agile-XP Environments," *Journal of Software*, vol. 8, no. 7, pp. 1556-1567, 2013.

[34] Samalikova J., Kusters R., Trienekens J., and Weijters A., "Process Mining Support for Capability Maturity Model Integration-Based Software Process Assessment, in Principle and in Practice," *Journal of Software: Evolution and Process*, vol. 26, no. 7, pp. 714-728, 2014.

[35] Sango M., "A Component-Based Model-Driven Approach with Traceability of Concerns: Railway RBC Handover Case Study," *in Proceedings of Young Reseachers Seminar*, Rome, 2015.

[36] Schwarz H., Ebert J., and Winter A., "Graph-Based Traceability: a Comprehensive Approach," *Software and Systems Modeling*, vol. 9, no. 4, pp. 473-492, 2010.

[37] Seibel A., Neumann S., and Giese H., "Dynamic Hierarchical Mega Models: Comprehensive Traceability and its Efficient Maintenance," *Software and Systems Modeling*, vol. 9, no. 4, pp. 493-528, 2010.

[38] Song W., "Requirement Management for Product-Service Systems: Status Review and Future Trends," *Computers in Industry*, vol. 85, pp. 11-22, 2017.

[39] Souali K., Rahmaoui O., and Ouzzif M., "An Overview of Traceability: Definitions and Techniques," *in Proceedings of 4th IEEE International Colloquium on Information Science and Technology*, Tangier, pp. 789-793, 2014.

[40] Tariq A., Khan S., and Iftikhar S., "Remapping of CMMI level-2 KPA's for Development Process Improvement of Software-As-A-Service (Saas) Cloud Environment," *in Proceedings of International Conference on Open Source Systems and Technologies*, Lahore, pp. 43-51, 2014.

[41] Unal A., Karaomer R.B., and Kaynak O., "Analysis of the Practices for the CMMI-SVC in an ISO/IEC 20000-1 Certified Organization," *in Proceedings of European Conference on Software Process Improvement*, vol. 567-577, 2017.

[42] Violante M., Vezzetti E., and Alemanni M., "An Integrated Approach to Support The Requirement Management (RM) Tool Customization for a Collaborative Scenario," *International Journal*

on Interactive Design and Manufacturing, vol. 11, no. 2, pp. 191-204, 2017.

[43] Zanatta A. and Vilain P., "Extending an Agile Method to Support Requirements Management and Development in Conformance to CMMI," *HIFEN*, vol. 30, no. 58, 2006.

Gullelala Jadoon received her M.S. and Bachelors' degree in Software Engineering from University of Engineering and Technology, Peshawar, Pakistan. She received University Gold Medal in Bachelors' in Software Engineering. She also received distinction in MS thesis titled, "Traceability Model for Requirements Management: Implementation of CMMI practices". She is currently working as Lecturer with the Department of Information Technology, University of Haripur, Pakistan. She has supervised a number of students' research projects in the field of computer science and software engineering.

Muhammad Shafi did his bachelor from Ghulam Ishaq khan Institute of Engineering Sciences and Technology and PhD from Loughborough university UK in 2005 and 2010 respectively. He has served at various universities including University of Engineering and Technology Peshawar, University of Science and Technology Bannu, Islamic University in Medina Saudi Arabia and Air University Islamabad. Currently, he is serving as associate professor at Sohar University in Oman. Computer vision, machine learning, human computer interaction, mobile computing, and software engineering are his areas of research. He has published more than 50 papers in various reputed journals and conferences. He has also worked in software development projects for various multinational companies.

Sadaqat Jan received his Ph.D. degree from Brunel University, London, UK. He is working as a professor in the Department of Computer Software Engineering, University of Engineering and Technology, Mardan. His research interests include semantic web, data mining, HCI, requirement engineering, and knowledge engineering.