Comparative Analysis of Model Integration Paradigms in Model-Driven Software Development

Aisha Fatima, Hamza Ali
National University of Computer and Emerging Sciences, Lahore, Pakistan

Abstract—This survey paper shows the recent state of model comparison as it's applies to Model Driven engineering. In Model Driven Engineering to calculate the difference between the models is a very important and challenging task. There are number of tasks involved in model differencing that firstly starts with identifying and matching the elements of the model. In this paper, we discuss how model matching is accomplished, the strategies, techniques and the types of the model. We also discuss the future direction. We found out that many of the latest model comparison strategies are geared near enabling Meta model and similarity based matching. Therefore model versioning is the most dominant application of the model comparison. Recently to work on comparison for versioning has begun to deteriorate, giving way to different applications. Ultimately there is wide change among the tools in the measure of client exertion needed to perform model comparisons, as some require more push to encourage more sweeping statement and expressive force.

Keywords—Model comparison, model clone detection, model versioning, EMF Model, model diff.

I. Introduction

In software development a very famous technique used is Model Driven Engineering (MDE). Model Driven Engineering basically put emphasis on developing and exploiting domain models as compared to computing concepts. The basic purpose of MDE approach is to increase the productivity between the systems.

MDE consists of using high level software devices. MDE becomes more prevalent in the software engineering. The need for effective approaches for finding the similarities and differences among high-level software models becomes imperative.

Since MDE involves being consumed initial class artifacts for developers. It not single merit being a stand-alone task but helps engineers inside additional MDE tasks such as model composition, inferring and testing of model transformations [1].

The model comparison is important in MDE. There are generally not any definitive surveys towards the model comparison research. There are few papers that touch on the top and they examine only very few techniques and some specific models.

In this paper, we discuss the current state involving model comparison research along with discuss the area's in future directions.

The purpose to describe the approaches to accomplish model comparison the numerous techniques are taken and their models are categorized. This survey works extremely well reference guide for developers organized through the types of models being compared. If they must work with the specific model type they can use this survey to recognize the approach is usually right regarding them or not. The paper background all about models which categorizes and describes the existing model comparison approaches by the type and subtype that they compare summary and future directions regarding the model comparison are discussed in this paper.

Section II discusses the different Model step by step. In Section III we give summary and future direction and in the last section, conclusion of this paper has been given.

II. Different Model

The model comparison in model driven engineering also refers straight on the act of involving, identifying similarities and differences between model elements. The versioning, model clone detection, model comparison is the additional areas of model driven engineering.

A. Model Comparison

The technique in [1] describes that model comparison is an operation. It classifies the elements into four categories:

1. Elements match and confirm.
2. Elements item match and do not confirm.
3. Elements that do not match in addition to within the domain connected with comparison**.**
4. Elements items do not match along with usually are not on the domain connected with comparison.

Matching refers for the elements. It represents the artifact even though conformance can be additional matching criteria. The example involving non-conformance is UML class diagram.

In the context of model versioning, model comparison has been decomposed directly into three phases' calculation, representation, and visualization [2].

B. Model Versioning

The need regarding collaboration among teams throughout MDE projects will be critical. Traditional software projects achieve the Version Control Systems (VCS) in the same way CVS along with Subversion. Similarly intended for MDE will be imperative. The model may work independently but later on always be able to reintegrate updated versions into the main

project repository. Traditional VCS methods do not run nicely in products as they are unable to handle model specific elements much like the “dangling reference” problem and others [3].

Model versioning will be broken in to various other phases by various other people [4]. Generally, it is usually seen regarding model comparison or matching the model elements correspond to help in detection of differences and conflicts.

C. Model Clone Detection

The example of model comparison being consumed in a crafted context is usually model clone detection. In traditional software projects a good code clone refers for the collections involving rule that happen to be such as single in a number of measure associated with similarity [5]. One common reason that code clones arise inside these projects may be the implementation of a similar concept throughout ones system. The problem in code clones is a great change in the actual sole identify that how the system must be updated in multiple places. The research in code clones is very mature. There are numerous techniques and tools for the exchange in them [6].

The analogous problem involving model clones refers to groups associated with model elements which are exhibited to be able to become similar in several defined fashion [7]. The comparison with clone detection, research in model clone can be quite limited [8].

D. Model Comparison Strategies and Techniques

In this section we categorize existing model comparison methods and discuss the strategies, techniques of any model comparisons.

The technique compares UML devices and uses their UUIDs are actually proposed [9]. The method transforms UML machines to graphs next traverses each tree level with the purpose regarding searching for identical UUIDs. The current process takes straight into differences among the matched model elements like features and relationships.

The technique in [10] for model matching derives the signature-match rules based towards abstract syntax of a Meta model describing the modeling languages. Specifically, they say that three equipment matches if they belong towards same Meta class have the same title and also the same primary context, such as the current surrounding structure of the model comprised of neighbors along with descendants. They state how the method is actually extended to help work with any MOF based modeling languages. The additional rules are actually further through extending the model with appropriate stereotypes which the method can interpret.

A model versioning tool designed to work with many kinds of UML products in [11] possibly help environments. It does not perform model matching just like almost all elements are usually linked to the previous version, starting with baseline version. Differences and conflicts are generally detected from processing XML Metadata Interchange (XMI) files in addition to using UML-specific knowledge in order to calculate which elements have been added, modified or maybe deleted.

In [12] produced the current Mqlone tool to experiment with the idea associated with detecting UML model clones. They convert XMI files via UML case models along with turn them in to Prologue.

EMF Models

Eclipse Modeling Framework (EMF) devices are MOF Meta devices. The idea will certainly define Meta models such as UML. They use the Eclipse development environment. EMF Compare [13] is actually an Eclipse project rather compared to relying on EMF models UUIDs. They used similarity based-matching to allow the tool to always be added generic and helpful within a good number of situations. The matching calculation will be according to various statistics and also metrics that happen to be combined to help generate a match score. This includes analyzing ones name, content, type, along with relations of an elements.

The Top Cased technique in [14] is a project providing in MDE environment that benefits EMF equipment and created straight with regard to the measures critical applications and systems. They perform the matching and differencing using static identity-based matching. Another model versioning tool [15] item will probably make use of any EMF-based model. This approach they do both version-specific comparisons like Odyssey VCS, termed syntactical and semantic comparisons. Semantic comparisons are happened to be completed from semantic views. Semantic views throughout the actual context are usually the resulting models that come by the user-defined model transformation. They execute towards the original equipment being compared to provide device meaning from a personalized view regarding interest these transformations are specified in the Atlas Transformation Language.

E. Model Matching Approaches

The different approaches for the Model matching are discussed. Fig. 1 shows the overall scenario of these all approaches.

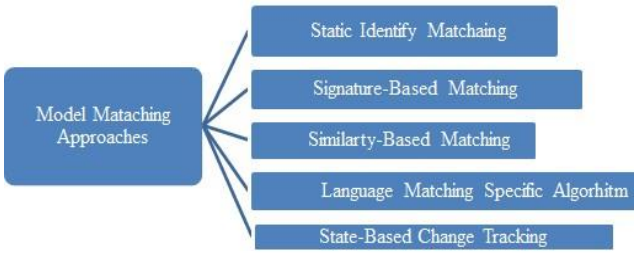

Fig. 1 Model Matching approaches

1. Static Identify Matching

In this approach each model can be constant and nonvolatile identifier at this creation. Therefore, the basic model can be based on the corresponding identifiers [16] as was discussed in [17]. The advantage of this approach that approach is that it not takes any perception forms the user and approach also fast. The other point this approach cannot be used for the Model that it can be independently constructed from the each other and the model representation technologies not support the maintenance of the identifiers that can be unique. The set notation loosely [18] the symmetric delta may be written as:

$$(v1, v2) = (v1 \text{ n } v2) [(v2 \text{ n } v1)]$$

Signature-Based Matching

In [19] the authors can be proposed and discuss the all limitation of the static based matching and gives a new method of signature based matching. In this method to identify the model is not the static but the signature can be calculated from the user defined function as the language of model [20]. The method of model that can be independently made, also be comparing from with each other's and this approach not only dependent on the identities. In this method the static-identify approach no effort can be required. Rather the developer can make different function that can used to calculate the different model identities.

Similarity-Based Matching

In Static Identity based matching and in signature based. matching the elements of the matching models were matched on the basis of true/false identities but static or dynamic matching is applied due to which these approaches represent the models in the form of typed attribute graphs and on the base of the similarities between the properties of the elements of matching models they identify the similarities between the elements of the matching models. However, all the properties of model elements are not of equal importance for matching model. It is more likely to match the classes with matching names than that of matching classes with matching values in the abstract feature. Therefore, such algorithms are needed to be provided which are related to similarity based algorithms along with configuration which specifies relative weight of its each feature. The syntactic information of the element is also called its signature, hence called signature-based matching [21].

Language Specific Matching Algorithm

This portion contains matching of the algorithm designed for specific modeling language to discusse the .UML Diff in [22] and the work in [23] where state charts of the UML models are targeted, respectively. In order to provide accurate results this technique join the semantics of targeted languages, to provide accurate outcomes, this is the main advantage of this matching technique and it also reduces the search space. When comparing UML models, when we are matching the two classes with the same name whatever their package structure is the UML specific types of elements. Moreover, it can integrate the knowledge that in order to reduce the number of comparison to increase performance to match only those operations whose classes are known to match, having same parameters and properties. When modeling a system using UML that is to be applied in a single inheritance language like Java, then simplifications can be associated by the algorithm based on particular feature's value, while the value of their general properties can be ignored. Although, all these benefits needs much work, as in static identity based matching approach no user effort is needed while in signature based matching approach the only right of signature generator, while in language specific matching method to specify complete matching algorithm which needs a lot of effort. To make the custom match algorithm development, they use methods like EMF compare which provides a set-up which is able to computerize the unimportant parts of the contrast procedure, and allow the developers to focus on the method for comparison.

State-Based Tracking Changes

State based tracking can be used for the comparing two models for example the version and its successor after the change occurred. This activity has two steps one is the matching and ether is the comparing. In the comparing phase, the each node in the one phase and the other node is found on other phase. The matching can be used for the matching the similarity of the node. If the model uniquely identifies then it is O (1) on the other hand the O (n2) can be used for the n nodes [24], [25].

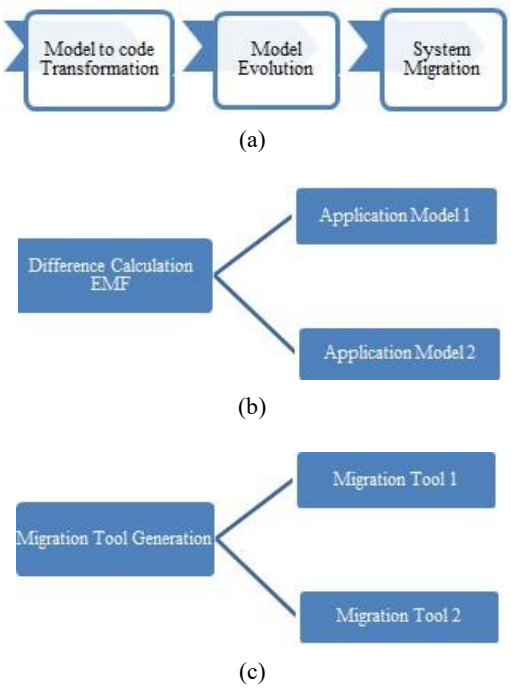

Fig. 2 (a) Model transforms (b) Diff calculation (c) Migration tool gen

F. Evolution of Data Intensive Web Application by Model Driven Techniques

Model driven engineering will for the development of the web application the approach can tell about the migration and data intensive approach for the web application. Model differencing techniques are detecting the differences [26] that will be the migration facility. The migration facility can be detecting the modification during the entire model's lifecycle and also the aspects that are not we derive from the source models, the approach can be confirmed on the web content and the WebML. The general migration approach is shown in Fig. 2.

G. Model-Driven Development of Web Information Systems

In this approach model driven approach for the web application is called the MODEWIS (Model Driven development of web applications).In this MDE approach OMG's Model driven Architecture principles are made between the three levels of abstraction [27]: The Computational Independent Model(CIM) The Platform Independent Model (PIM) level The Platform Specific Model (PSM) level. Further they can differentiate two levels in the Platform Specific Model level:

 Abstract-Platform Specific Model (APSM) Specific-Platform Specific Model (SPSM).

This will be shown that the MODEWIS is the evolutionary for the model driven process.

The APSM common web characteristics but in the SPSM provide the real implementation of the platforms.

H. Model Matching Approaches Table

Table I shows the model matching approaches.

TABLE I
MODEL MATCHING APPROACHES

Approaches	Static Identity-based	Customization Support	Similarity-based
Alenen and Porres[28]	UML Specific	-	-
DSMDiff[29]	-	Meta Model Independent	-
EMF Compare[30]	-	Meta Model Independent	Custom matching algorithm
SI Diff[31]	-	Meta Model Independent	Weight configuration
TOPCASED[32]	Meta Model Independent		-
UML Diff[33]	-	UML Specific	-
ECL[34]		Meta Model - Independent	DSL for Specifying custom

I. Met model-Agnostic Approaches

The comparison procedures probably confirm the arbitrary Meta model assuming the idea to catered properties.

Examples involving Met model-independent techniques this show similarity-based matching methods include the current Epsilon Comparison Language [35] along with the Domain Specific and Model Difference DSMDiff [36]. DSMDiff is usually the extension connected with perform carried out in UML model comparison techniques. DSMDiff functionalities both similarity and also signature based matching. The similarity based matching focuses towards similarity regarding edges among additional model nodes. DSMDiff evaluates differences between matched elements in addition to consider them routed deltas. While DSMDiff was formulated making use of DSMLs specified for the Generic Modeling Environment (GME), one's strategy will be for longer times make use of any kind of DSML creation tool. DSMDiff propose allowing user interaction that enables one to select the mappings (matches) via listing connected with applicable candidates.

Epsilon Comparison Language (ECL) are developed following DSMDiff and also SiDiff and attempts in order to address ones fact that it is predecessors do not give intended for model in order to configure language specific particulars. The idea can help in matching model elements via different meta models[37]. That is accomplished with an imperative high-level manner. ECL makes it possible for modelers to help specify model comparison rule-based algorithms to identify matched elements inside additional models.

A plugin with regard to meta-case applications are developed [38].This is effective model version comparison regarding devices defined from a meta-Case tool. Meta-Case tools run similarly to help its case counter parts except these are generally not constrained through a good particular schema and Meta model. The particular plugin matches all the elements through the unique identifiers and calculates the differences just like sent deltas [39] describe a graph-based VCS. This really is quite similar with item operates in Meta case products and matches them applying baselines and unique identifiers. Differences are calculated as routed deltas with in respect to previously versions.

In [40] a version control system developed and the idea will detect both structural and textual differences between versions of a wide array regarding software artifacts. The actual approach utilizes similarity-based matching via assigning just about all artifacts an identifier. It encapsulates the current element and also representing them in the same way nodes within a sent attributed graph, just like model clone approaches.

In [41] discussed the prerequisites pertaining to difference representations. The current Meta modeling techniques, like MOF, not satisfy them. They provide their particular metamodeling program to define differences to be able to it. They give the model comparison approach in addition to prototype that allows end user configurations associated with what combination of a four model-matching techniques are used. They provide the examples where they extend to perform accomplished previously regarding SiDiff, combining with some other matching techniques, similar to using a UUID. The generality comes in a cost of any large range connected with configuration, work, and also user interaction.

There are measures that translate products in another language or even notation. The item maintains semantics of the machines to facilitate model comparison. Individual example would be the run done [42] of which they propose an abstract equivalence notion regarding object models, within some other words, the means of representing objects that enables them for always be compared. They use an alphabet, which is the set regarding relevant elements that is to be compared, and views, that are mappings. The item express the different ways that individual element with single model can be interpreted by elements of a different model. Similarly, in [43] the semantic diff operators are discussed, that will represent the relevant semantics associated with each models. Semantics tend to be represented with the utilize of mathematical formalisms supply the tools cddiff and also addiff with regard to class diagram differencing in addition to activity class diagram differencing, respectively. Some other examples regarding translating models in another language include UML models being translated straight into Promela equipment [44]-[46] although the actual operate do not intend to perform model comparison nor differencing explicitly.

QVT-Relations (QVT-R) permits with regard to a great declarative specification connected with two-way (bidirectional) transformations additional expressive than the current additional QVT languages. The particular expressiveness makes it possible for pertaining to the form involving model comparison through

its check only mode, which is to be the mode by which products are generally checked for consistency rather as compared to generating changes [47] in brief; game theory is applied to be able to QVT-R by having a verifier and refuter. The verifier confirms the settlement will certainly succeed and the refuter's objective is actually in order to disprove it.

J. Methods for Behaviour/Data-Flow Models

1. Simulink and MATLAB Models

In [48] will be an approach that uses ideas coming from graph theory in addition to can be applicable in order to any model that is represented being a data-flow graph. Machines are usually first flattened in addition to unconnected lines usually are removed. Subsequently, these are generally normalized from shipping each of the blocks in addition to lines found inside the models. Similarly, eScan along with aScan algorithms attempt to be able to detect exact matched and also approximate clones, respectively [49]. Exact-matched clones usually are groups associated with model elements having the same size and aggregated labels, in which contain topology information along with edge and node brand information. Approximate clones tend to be the individual that happen to be not exactly matching but fit a number of similarity criteria. AScan uses vector-based representations regarding graphs. The idea monitor a good sub-set of structural has about the graph. This will be later refuted, however, involving Clone Detective [50]. AScan will be capable to detect approximate clones although Clone Detective will be not. Much like Clone Detective, most of these algorithms utilize similarity-based matching.

In [51], method deal with syntactic clones with structural similar copies detected. Applying normalization strategies its graph use transformations, they extend these types of approaches to repay semantic clones this will probably have similar behaviour but other structure.

Most recently in [52], Simulink are explaining that detects the miss clones in Simulink models. This can be performed coming from modifying existing code clone procedures to use the current textual representations of a Simulink models. In comparison to Clone Detective, they detect the same exact clones along with several extra near-miss ones.

2. Sequence Diagram

In [53] discover duplication with series diagrams. They convert sequence diagrams in the array in addition to represent this array as being a suffix tree. The tree will be traversed along with duplicates are generally extracted from looking for its longest common prefix, or elements. This lead for the leaf node, associated with two suffixes. Duplicates usually are defined being a set connected with sequence-diagram fragments that contain the same elements and find one same sequence-diagram was made relationships. Such as the model clone approaches discussed, the method utilizes a variation involving similarity-based matching. Equally this comparing the graph representation of the fragment‘s elements.

3. State Chart Diagrams

In [54], match state chart diagrams for the model merging. They carry out via heuristics that include looking on terminological, structural, along with semantic similarities between models. The heuristics are generally split directly into two categories. The static heuristics work with attributes without having semantics, much like the names, features of elements, and behavioural heuristics, which obtain pairs the item, have similar dynamic behaviour. For the employ associated with heuristics, the approach requires a domain expert go shopping through the relations in addition to complete or maybe remove relations, accordingly, to get ones required matching relation. This approach utilizes both similarity-based matching for the static heuristics in addition to custom language were made matching through dynamic heuristics.

K. Methods for Structural Models

This division discusses methods for the structural models. The comparing and also differencing software structural diagrams performed through [55].

1. UML Structural Models

In [56], custom language specific matching name similarity and UML structure-similarity to name matching elements. These kinds of metrics are usually combined and also compared against a great user-defined threshold. It is intended to be a model versioning reasoned. The item discovers changes designed through solitary version of any model for one to another.

Reference [57] focuses with UML class diagram differencing. It uses the combination involving static identitybased and also similarity-based matching in the evaluation function, in which the current quality of a match. Similarly, [58] can be a plugin produced for its Fujaba (From Uml in order to Java and Back Again) tool suite that enables for end user sent matching regarding elements. Specifically, users will probably Click match candidates that happen to be ranked according to be able to a great similarity measure it is a combination regarding static identity-based in addition to similarity based matching, including UMLDiff..

Reference [59] utilizes signature-based matching to be able to compare in addition to compose UML class equipment to assist within Aspect-oriented modelling [60]. The matched based on it is signatures, or even property values associated from the class. Each signature features a signature type, that is to be the set associated with properties. Using the KerMeta8, an model querying language, the current signatures consumed pertaining to comparison are usually derived from the tool based towards the possesses it's about the meta model.

In [61] translate UML class diagrams in to ALCQI, a “simple” description logic representation. They show that this is possible to reason information on UML class diagrams. In the same way ALCQI description logic representations and gives an encoding coming from UML class diagrams in order to ALCQI. Although the translation does not maintain the entire semantics of an UML classes, it preserves enough of the idea to confirm intended for class equivalence. They use UML-specific semantics, it argue that there is usually an application form regarding language-specific matching.

In the [62] extend perform on semantic differencing and provide a translation prototype, called CD2Alloy. The idea converts UML classes in Alloy. The Alloy signal consists of constructs. It recognize the corresponding elements regarding to UML class diagrams and will allow intended for semantic comparisons, similar to determining if solitary model can be a refinement regarding another. It considered being a customlanguage catered comparison due to the Utilize regarding UML semantics.

2. Met model-Agnostic Approaches

Preliminary operate on model comparison was carried out through [63] in which they devised a good comparison approach with regard to any kind of structured document. They convert the information representing the current statement structure in a graph consisting regarding nodes the item have identifiers derived because of the corresponding elements they represent. The approach, that is to be analogous towards the model clone identification techniques, benefits similarity-based matching in addition to describe differences in relation to delivered deltas.

In [64] very including UMLDiff except SiDiff uses the simplified underlying comparison model throughout order to help handle any kind of equipment held in XMI format. Similarly in order to UMLDiff, the idea functionalities similarity based metrics. That is performed throughout respect for the elements' similarity metrics. An example of a weighted similarity is using a class element get ones similarity involving its class name weighted current highest. No matter whether a good uniquely identifying element is usually matched, these types of being a class name, these are straight identified like a match. That is followed via top-down propagation involving the matching pair. The particular top-down approach will allow for its algorithm to reduce differences through evaluating a good correspondence table that is the output of a matching phase. Similarly towards the translation associated with UML class diagrams in ALCQI [65], propose a great comparison measure with regard to description logics, including anybody taken in the Semantic internet. This is completed through existing ontology semantics. They describe a good semantic similarity measure. It is able operate the semantics of your ontology that the concepts refer to.

3. Methods for Product Line Architecture

In [66], the comparison involving the products line machines. The assumption inside this function will be how the comparison can be being carried out between only two types of the same artifact. Comparison will be carried out recursively along with the increasingly fine grained equally ones algorithm delves deeper in to the current product-line hierarchy. The particular approach engages similarity-based matching: along elements with the hierarchy compare interfaces, optionality, along with type; along with higher level elements compare ones elements contained throughout them. Differences are usually represented just as dispatched deltas.

In [67] discussed the good framework for comparing individual products. They utilize similarity-based matching, The idea is, items are generally viewed as model elements along with a great match is actually defined Just as your own case where two model elements have features which can be similar enough to always be above the defined weighted threshold. The authors note the idea "(their) refactoring framework will be applicable to help a variety of model types, such as UML, EMF or Mat lab/Simulink, and in order to different compare, match in addition to merge operators".

4. Methods for Process Models

In [68] discussion the need for ascertaining differences among software development process models and also outline a difference system would require. They devise Delta-P [69], in which may use numerous UML technique models. Delta-P converts system equipment into Resource Description Framework (RDF). The next performs a great identity-based compare along with calculates differences. They Utilize staticidentity based matching as unique identifiers. Differences are represented like delivered deltas, which might be grouped together to be able to application form higher level deltas. Similarly, [70] discuss three similarity metrics this help compare maintained process models: node matching similarity, in which compares ones labels and attributes attached in order to program model elements; structural similarity, that evaluates labels along with topology; and behavioural similarity, that looks on labels together with causal relations through the technique models.

III. Summary and Future Direction

Table II summarizes the techniques discussed in the paper organized through the type along with sub type involving model to compare.

TABLE II
DESCRIPTION OF MODELS

Types Of Model	Sub type of model	Specific Approach Tool	Matching Strategy+	Primary Use#
Multiple type of Models	*Uml Models*	Alnan	Static Identity based	Model Versioning
		RSA	Static Identity based	Model Versioning and Model Merge
		Obsi	Static identity	Model Versioning
		Odyssey vcs	-	Model Versioning
	EMF Models	Selonen	Signature Based	Model Merge
		EMF compare	Similarity based	Model Versioning
		Top cased	Static identity	Model Versioning
		Smover	Static identity	Model Versioning
		Modeling	Static identity and Similarity based	Model Versioning
		ECL	Similarity based	Model Merge and Model Transformation Testing
		DSM diff	Similarity based	Model Transformation Testing and Model Versioning
	Meta Models Agnostics Independent	Mehra-meta	Static identity	Model Versioning
		Van den brain	Static identity, Similarity based, Signature Based, Custom Language Specific	Model Versioning and Model Merge
		Nguyen	Similarity based	Model Versioning
		QVT-R	Static identity	Model Transformation Specification
Data Flow Model	*Simulink*	Clone Detection	Similarity based	Verification
		Ascan/Nscan	Similarity based	Verification
		Simone	Similarity based	Verification
	UML sequence Diagram state charts	Liu	Similarity based	Verification
		Nejati	Similarity based and Custom Language Specific	Model Merge
Structural Models	*UML Models*	UML Diff	Custom Language Specific	Model Versioning
		UML Diff	Static identity and Similarity based	Model Versioning
		Reddy	Signature Based	Aspect oriented Model
		Mirador	Similarity based	Model Versioning and Model Merge
		CD2 Alloy	Custom Language Specific	General Comparison
		ALCQI	Custom Language Specific	General Comparison
	MetaModel-Agnostic	Chaw the	Similarity based	Model Versioning
		Serif	Similarity based	Model Versioning
Product line Architecture	*Any PLA*	Chen	Similarity based	Model Versioning
		Rubin	Similarity based	Model Merge
Process Models	*Software Process Model*	Delta-P	Static identity	Model Versioning

Just as seen in the table, similarity-based matching will be the all commonly employed strategy. This is clear the item solitary future direction connected with function within the particular area will be the focus with tools the idea might employ equipment. It confirms to help the arbitrary Meta model. The result is consistent with the recent trend within domain-specific modeling.

The majority associated with work with model comparison appears in order to the model versioning. Much of a recent operate is focusing at model transformation testing along with model clone detection. The new extensions of existing model comparison methods are being attempted just like the extension regarding model clone detection to be able to detect common sub-structures and patterns inside machines [71]. These kinds of patterns will probably ideally supply from project engineers for to facilitate analysis along with assistance on the development connected with future MDE projects. Many strategies require not any user interaction just like they function under specific conditions or maybe usually dynamic enough for to realize the current context or maybe Meta equipment they are signing with.

IV. CONCLUSIONS

Model comparison is really a relatively a broad research place this can be very ticks to help MDE. It's been implemented in various forms in addition to regarding numerous purposes, predominantly in model versioning, merging and clone detection.

We have given a good overview of the area, and have observed the majority associated with recent strategies pertaining to equipment belonging to be able to arbitrary metamodels. Similarity-based matching could be the approach recognized by almost all methods. Model versioning appears to be the current just about all common goals for model comparison up to help the actual point, but it is starting to shift. Lastly, several approaches demand additional end user effort to be able to the function model comparison; however it is to help facilitate flexibility and also

strength. Numerous of a techniques demand no individual interaction because they are intentionally constrained as well as are generally made to financial transaction throughout multiple situations.

REFERENCES

[1] D., Paige, Kolovos, R., and Polack, F. (2006). Model comparison: a foundation for model composition and model transformation testing. In IWGIMM, pages 13–20

[2] Pierantonio and Brun, A. (2008). Model differences in the Eclipse modelling framework. The European Journal for the Informatics Professional, pages 29–34.

[3] K., Seidl, Altmanninge, and Wimmer, M. (2009). A survey on model versioning approaches. International Journal of Web Information Systems, 5(3):pages271–304.

[4] Alanen, M. and Porres, I. (2003). Difference and union of models.Springer-Verlag Berlin Heidelberg 2003: pages 2–17.

[5] Koschke, R. (2006). Survey of research on software clones. Information and Software Technology 55 (2013),pages:1165–1199

[6] Cordy, Roy, C., J., and Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Science of Computer Program-Ming, 74(7):pages:470–495

[7] Deissenboeck, F., Hummel, B., Jürgen's, E., Schatz, B., Wagner, S., Girard, J., and Teuchert, S. (2009). Clone detection in automotive model-based development. In ICSE: pages 603–612.

[8] Deissenboeck, F., Hummel, B., Juergens, E., Pfaehler, M., and Schaetz, B. (2010). Model clone detection in practice. In IWSC, pages 57–64.

[9] Ohst, D., Welle, M., and Kelter, U. (2003). Differences be-tween versions of UML diagrams. ACM SIGSOFT Soft-ware Engineering Notes, 28(5), pages: 227–236.

[10] Dirk Ohst, Michael Welle and UdoKelter,September 1–5, 2003,Differences between Versions of UML Diagrams, *IN ESEC/FSE'03*:pages:1-10

[11] Selonen, P. and Kettunen, M. (2007).Met model-based inference of inter-model correspondence. In ECSMR: Pages 71–80.

[12] Oliveira, H., Murta, L., and Werner, C. (2005). Odyssey-vcs: a flexible version control system for UML model elements. In SCM:pages 1–16.

[13] Brun, C. and Pierantonio, A. (2008). Model differences in the Eclipse modeling framework. The European Journal for the Informatics Professional: pages 29–34.

[14] Farail, P., Gaufillet, P., Canals, A., Le Camus, C., Sciamma, D., Michel, P., Cregut, X., and Pantel, M. (2006). The top cased project: a toolkit in open source for critical aeronautic systems design. ERTS: pages 1–8, electronic.

[15] Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., and Quesada, J. (2002). Maude: specification and programming in rewriting logic. Theoretical Computer Science, 285(2): pages: 187–243.

[16] M. Alanen and I. Porres. Difference and Union of Models. In UML 2003 - The Unified Modeling Language, Springer-Verlag, 2003, volume 2863 of LNCS, pages 2–17.

[17] P. Farail, P. Gaufillet, A. Canals, C. L. Camus, D. Sciamma, P. Michel, X. Crgut, and M. Pantel. AFIS 2006 Conference, The TOPCASED project: a Toolkit in Open source for Critical Aeronautic Systems Designpages:1-6

[18] In ERTS06, 2006.Sabrina Fortsch and Bernhard Westfechtel. Differencing and merging of software diagrams - state of the art and challenges. In ICSOFT (SE), pages 90-99.

[19] B. Collins-Sussman, B. Fitzpatrick, and C. Pilato. Inc., 2004, Version Control with Subversion. For Subversion 1.1. O'Reilly & Associates, pages:1-299

[20] R. Reddy, R. France, S. Ghosh, F. Fleurey, and B. Baudry. 2005, Model composition a signature-based approach. In AOM Workshop:pages:1-7

[21] Raghu Reddy, Robert France, Sudipto Ghosh, Franck Fleurey, and Benoit Baudry. October 2005, Model composition - a signature-based approach. In Aspect Oriented Modeling,pages:1-7

[22] Z. Xing and E. Stroulia. 2005, UMLDiff: an algorithm for objectoriented design differencing. In ASE'05. ACM: pages 54–65

[23] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave. Matching and merging, AT&T Laboratories–Research: pages:1-10

[24] Lindholm, T., Kangasharju, J., Tarkoma, (2006), Fast and simple xml tree differencing by sequence alignment. In: DocEng '06, ACM pages:75-84

[25] Treude, C., Berlik, S., Wenzel, S., Kelter, (2007), Difference computation of large models. In: ESEC-FSE '07, ACM pages:295-304

[26] Antonio, Cicchetti, Davide, Di Ruscio, Ludovico Iovino, Alfonso Pieantonio (2011). Managing the evolution of data intensive web application by model driven teachniques,Springer-Verlag,pages:1-31

[27] Ali Fatolahi, Stéphane S. Some. (2014), Assessing a Model-Driven Web-Application Engineering Approach. Journal of Software Engineering and Applications, pages:360-370

[28] M. Alanen and I. Porres. Difference and Union of Models. In UML 2003 - The Unified Modeling Language, volume 2863 of LNCS, SpringerVerlag: pages 2–17.

[29] Yuehua Lin, Jeff Gray & Frédéric Jouault DSMDiff: a differentiation tool for domain-specific models,pages:1-30

[30] Cédric Brun, Obeo France.(2008) Comparing and Merging Models with Eclipse An update on EMF Compare.pages:1-36

[31] Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave (2007), Matching and merging of state charts specifications. IEEE Computer Society: pages 54–64

[32] Pontisso, N.; French Space Agency, Toulouse; Chemouil, D. TOPCASED Combining Formal Methods with Model-Driven Engineering: pages1-12.

[33] Xing and Eleni Stroulia (2005), UMLDiff: An Algorithm for Object-Oriented Design Differencing Zhenchang.ACM, pages:54-65.

[34] R. Reddy, R. France, S. Ghosh, F. Fleurey, and B. Baudry. (2005) Model composition - a signature-based approach.AOM,pages:1-7.

[35] Kolovos, (2009). Establishing correspondences between models with the epsilon comparison language. In Model Driven ArchitectureFoundations and Applications, pages 146–157.

[36] Lin, Y., Gray, J., and Jouault, F. (2007). DSMDiff: a differentiation tool for domain-specific models. European Journal of Information Systems, pages: 349–361.

[37] Kolovos, D., Di Ruscio, D., Pierantonio, A., and Paige, R. (2009). Different models for model matching: An analysis of approaches to support model differencing. In CVSM: pages 1–6.

[38] Mehra, A., Grundy, J., and Hosking, J. (2005). A generic approach to supporting diagram differencing and merging for collaborative design. In ASE, pages: 204–213.

[39] Nguyen, T. (2006). A novel structure-oriented difference approach for software artifacts. In CSAC, volume 1, pages: 197–204.

[40] Van den Brand, M., Protic, Z., and Verhoeff, T. (2010). Generic tool for visualization of model differences. In IWMCP, pages: 66–75.

[41] Gheyi, R., Massoni, T., and Borba, P. (2005). An abstract equivalence notion for object models. Electronic Notes in Theoretical Computer Science, pages:3–21.

[42] Maoz, S., Ringert, J., and Rumpe, B. (2011). A manifesto for semantic model differencing. In ICMSE, MOD-ELS'10, pages: 194–203.

[43] Henry Chesbroug, JIM Spohrer A Research manifesto for Services sciences. In Communication of the ACM, July 2006, vol 49, pages: 3540.

[44] Chen, J. and Cui, H. (2004). Translation from adapted UML to promela for corba-based applications. Model Checking Software, pages: 234–251.

[45] Latella, D., Majzik, I., and Massink, M. (1999). Automatic verification of a behavioural subset of UML state chart diagrams using the SPIN model-checker. Formal Aspects of Computing, pages: 637–664.

[46] Lilius, J. and Paltor, I. (1999). Formalising UML state ma-chines for model checking. UML, pages: 430–444.

[47] Stevens, P. (2009). A simple game-theoretic approach to checkonlyqvt relations. Theory and Practice of Model Transformations, pages:165–180.

[48] Deissenboeck, F., Hummel, B., Jurgens, E., Schatz, B., Wagner, S., Girard, J., and Teuchert, S. (2009). Clone detection in automotive model-based development. In ICSE, pages:603–612.

[49] Pham, N., Nguyen, H., Nguyen, T., Al-Kofahi, J., and Nguyen, T. (2009). Complete and accurate clone detection in graph-based models. In ICSE, pages: 276–286.
[50] Deissenboeck, F., Hummel, B., Juergens, E., Pfaehler, M., and Schaetz, B. (2010). Model clone detection in park-tice. In IWSC, pages:57–64.
[51] Al-Batran, B., Schatz, B., and Hummel, B. (2011). Seman-tic clone detection for model-based development of embedded systems. Model Driven Engineering Languages and Systems, pages: 258–272.
[52] Alalfi, M. H., Cordy, J. R., Dean, T. R., Stephan, M., and Stevenson, A. (2012). Models are code too: Near-miss clone detection for Simulink models. In ICSM, volume 12.pages:1-10
[53] Liu, H., Ma, Z., Zhang, L., and Shao, W. (2007). Detecting duplications in sequence diagrams based on suffix trees. In APSEC, pages:269–276.
[54] Nejati, S., Sabetzadeh, M., Chechik, M., Easterbrook, S., and Zave, P. (2007). Matching and merging of state charts specifications. In ICSE, pages:54–64.
[55] Rho, J. and Wu, C. (1998). An efficient version model of software diagrams. In APSEC, pages:236–243.
[56] Xing, Z. and Stroulia, E. (2005). UMLDiff: an algorithm for objectoriented design differencing. In ASE, pages:54–65.
[57] Girschick, M. (2006). Difference detection and visualization in UML class diagrams. Technical University of Darmstadt Technical Report TUD-CS-2006-5, pages: 1– 15.
[58] Barrett, S., Butler, G., and Chalin, P. (2010). Mirador: a synthesis of model matching strategies. In IWMCP, pages:2–10.
[59] Reddy, R., France, R., Ghosh, S., Fleurey, F., and Baudry, B. (2005). Model Composition - A Signature-Based Approach.pages:1-7
[60] Elrad, T., Aldawud, O., and Bader, A. (2002). Aspect-oriented modeling: Bridging the gap between implementation and design. In Generative Programming and Component Engineering, pages: 189–201.
[61] Berardi, D., Calvanese, D., and De Giacomo, G. (2005). Reasoning on UML class diagrams. Artificial Intelli-gence, pages:70–118.
[62] Maoz, S., Ringert, J., and Rumpe, B. (2011). Cd2alloy: Class diagrams analysis using alloy revisited. Model Driven Engineering Languages and Systems, pages: 592– 607.
[63] Chawathe, S., Rajaraman, A., Garcia-Molina, H., and Widom, J. (1996). Change detection in hierarchically structured information. In ICMD, pages: 493–504.
[64] Kelter, U., Wehren, J., and Niere, J. (2005). A generic difference algorithm for UML models. Software Engineering, pages:105–116.
[65] Altmanninger, K., Kappel, G., Kusel, A., Retschitzegger, W., Seidl, M., Schwinger, W., and Wimmer, M. (2008). AMOR-towards adaptable model versioning. In MCCM, volume 8, pages: 4–50.
[66] Chen, P., Critchlow, M., Garg, A., Van der Westhuizen, C., and van der Hoek, A. (2004). Differencing and merging within an evolving product line architecture. PFE, pages: 269–281.
[67] Rubin, J. and Chechik, M. (2012). Combining related products into product lines. In 15th International Conference on Fundamental Approaches to Software Engineering,pages:1-15
[68] Soto, M. and Munch, J. (2006). Process model difference analysis for supporting process evolution. Software Process Improvement, pages: 123–134.
[69] Soto, M. (2007). Delta-p: Model comparison using seman-tic web standards. Softwaretechnik-Trends, pages:27–31.
[70] Dijkman, R., Dumas, M., Van Dongen, B., Karik, R., and Mendling, J. (2011). Similarity of business process models: Metrics and evaluation. Information Systems, pages:498–516.
[71] Stephan, M., Alafi, M., Stevenson, A., and Cordy, J. (2012).Towards qualitative comparison of simulink model clone detection approaches. In IWSC, pages:84–85.