An Evaluation of Multi-Level Modeling Frameworks for Extensible Graphical Editing Tools

Kosaku Kimura[1] and Kazunori Sakamoto[2]

[1] Fujitsu Laboratories, Japan
[2] National Institute of Informatics, Japan
kimura.kosaku@jp.fujitsu.com, exkazuu@nii.ac.jp

Abstract. We need to have comprehensive knowledge about when, where and how we should use multi-level modeling methodologies and frameworks. There are previous work that introduce patterns and confirm the applicability of the methodologies. However, it is still difficult to select frameworks for various kinds of applications. In this paper, we focus on graphical editing tools as an application of multi-level modeling frameworks, and evaluates the capability for modeling and extensibility of multi-level modeling frameworks. We introduce a dataflow model as an example model of graphical editing tools. The evaluation result shows that Melanee can describe the dataflow model more accurately and can extend metamodels and metametamodels with less changes to existing elements.

1 Introduction

Model-driven engineering (MDE) is for facilitating automation and abstraction of software developments. Providing graphical editing tools based on MDE technologies is a typical approach for developing complex software with less effort. Diagrams edited on graphical editing tools encapsulate complex but unimportant information in order to help software developers to concentrate what really matters for their work.

Several standardized specifications and frameworks contribute the dissemination of MDE. Meta Object Facility (MOF)[3] provided by Object Management Group (OMG) is one of the standardized metamodeling specifications. Eclipse Modeling Framework (EMF)[4] is one of the popular MOF-compliant frameworks and has many mature sub-projects implementing OMG standards. There are several EMF-based frameworks for developing graphical editing tools that are extensible by plugins.

Extensibility is one of the most important feature for graphical editing tools. Graphical editing tools should be easily extensible for third-party developers in order to create applications for emerging domains. EMF provides the method

[3] http://www.omg.org/mof/
[4] https://eclipse.org/modeling/emf/

for extending models. However, in EMF, we can access only two levels, model and its metamodel, and it is difficult to define our own metametamodel. This problem makes limitations for extending graphical editing tools. For example, adding a new concept of diagrams by third-party developers is difficult due to metametamodel.

Multi-level modeling is an approach that can define and manipulate the arbitrary number of levels of metamodels. There are several methodologies for multi-level modeling: orthogonal classification architecture (OCA)[2, 3, 5], potency-based multi-level modeling[6, 15], powertype-based metmodeling[8, 9], dual deep instantiation (DDI)[14], etc. There are also several frameworks for multi-level modeling: Melanee [4], MetaDepth[7], Diagram Predicate Framework (DPF)[12], etc.

We need to have comprehensive knowledge about when, where and how we should use those multi-level modeling methodologies and frameworks. Lara et al. [13] elicited several patterns of multi-level models from metamodels in various domains and described how we apply methodologies to the patterns. Their work contributes to select methodologies for applying to our metamodels. However, it is still difficult to choose frameworks for various kinds of applications.

This paper focuses on graphical editing tool as an application of multi-level modeling frameworks, and evaluates the capability for modeling and extensibility of multi-level modeling frameworks. Our evaluation contributes to select the framework that is the most appropriate for graphical editing tools.

The remainder of this paper is organized as follows. Section 2 describes a dataflow model as an example of graphical editing tools. Section 3 evaluate frameworks by defining and extending the dataflow model on them. In Sect. 4 we discuss the result, and our conclusions are presented in Sect. 5.

2 Models for Graphical Editing Tools

In this section, we describe a model of typical graphical editing tools and introduce a dataflow model for using in the evaluation of the paper.

A typical graphical editing tool consists of a palette and a canvas. The palette displays icons of building blocks for composing a diagram. On the canvas, we put icons from the palette and create connections between icons for defining a diagram.

We consider the typical graphical editing tool has a five-level model. Table 1 shows the overview of the five-level model of the graphical editing tool. Models

Table 1. Five-level model of graphical editing tools.

Level	Overview
M4	MOF
M3	Metametamodels defining conceptual types of elements
M2	Metamodels defining types of elements displayed in palette
M1	Models of the diagram edited on canvas
M0	Real-world software

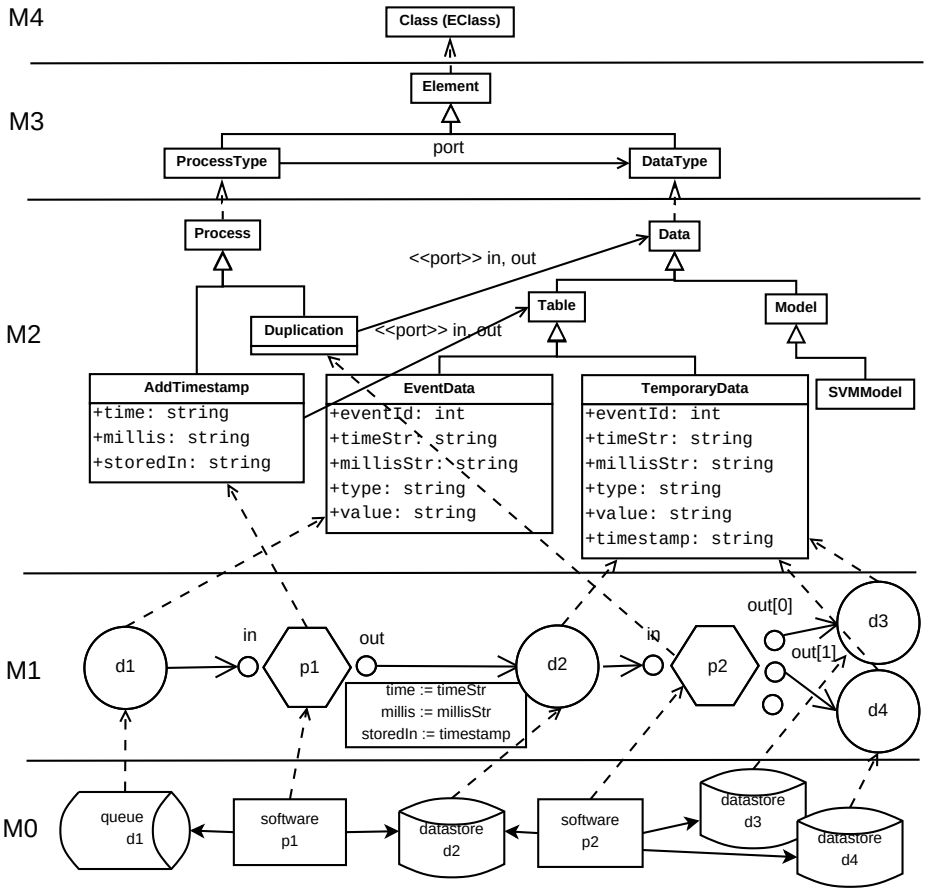

Fig. 1. Dataflow model.

residing in level **M4** is the metametamodel of the metamodeling architecture such as MOF. Beware that definitions of each level of the five-level model differs from the original MOF definitions such that the MOF metametamodel resides in level **M3**. The model only allows instantiation relations between elements of different levels in the model. Metametamodels of the diagram edited on the tool reside in level **M3**. Metamodels that define types of elements displayed in the palette reside in level **M2**. Models residing in level **M1** are diagram instances edited on the canvas, and level **M0** is the real-world software defined by diagram instances in level **M1**.

Now, we introduce a dataflow model as an example of models of graphical editing tools. Figure 1 shows the dataflow model. The dataflow model is used to define data processing applications[11, 16]. The dataflow model consists of process nodes and data nodes. There are two conceptional types in level **M3**: `ProcessType` and `DataType`. `ProcessType` has association `port` to `DataType`, and `DataType` does not have any association to `ProcessType`, which means only process nodes have ports for connecting to data nodes. There are

two classifications of elements from two super types: `Process` and `Data`. In Fig. 1, `AddTimestamp` and `Duplication` are subclasses of `Process`. `EventData` and `TemporaryData` are subclasses of `Table`. `Table` is a data type such that consists of tuples having several fields . Subclasses of `Table` (e.g., `EventData` and `TemporaryData`) are data types such that are stored in a relational database or a message queue. `AddTimestamp` and `Duplication` have association `in` and `out` to `Table` and `Data`, respectively. `AddTimestamp` produces a `Table`-type data by adding a field for calculated timestamp value to another `Table`-type data, and `Duplication` producing multiple data by duplicating the input data. Their associations are instances of association `port`, which means process nodes that are instances of `AddTimestamp` and `Duplication` have input and output port in level **M1**. The input and output ports in level **M1** are instances of association `in` and `out`.

3 Evaluation

In order to help selecting the framework that is most appropriate for models of graphical editing tools, we apply multi-level modeling frameworks to define the dataflow model[5] and evaluate the frameworks from the following viewpoints.

Capability for modeling: how well we can represent the dataflow model as it is,

Extensibility: how easy we can extend metamodels and metametamodels of the dataflow model.

We select EMF and two multi-level modeling frameworks, Melanee and MetaDepth[6], which are still actively maintained and support MDE tools such as model transformation from the list of tools in the multi-level modeling wiki [1]. Table 2 shows the three frameworks and their characteristics.

EMF is a two-level modeling framework based on MOF. Regarding Fig. 1, if we use EMF, we have to define level **M3** and **M2** in the same level. There are several methods for defining two levels in one level[13, 10]. In this paper, we

Table 2. Selected frameworks

Framework	Methodology	Top-level metamodel elements (excerpted)
EMF	Two-level	`EClass, EReference, EAttribute`
Melanee	OCA	`Entity, Attribute, Connection, Inheritance`
MetaDepth	Potency-based	`Node, Edge, Field`

[5] The files can be downloaded from `https://github.com/dataflow-mlm/dataflow-mlm`

[6] We have tried DPF, which is also actively maintained and supports several EMF subprojects. However, we found DPF does not provide how to define attributes, and thus it is difficult to define diagrams whose nodes have attributes like the dataflow model.

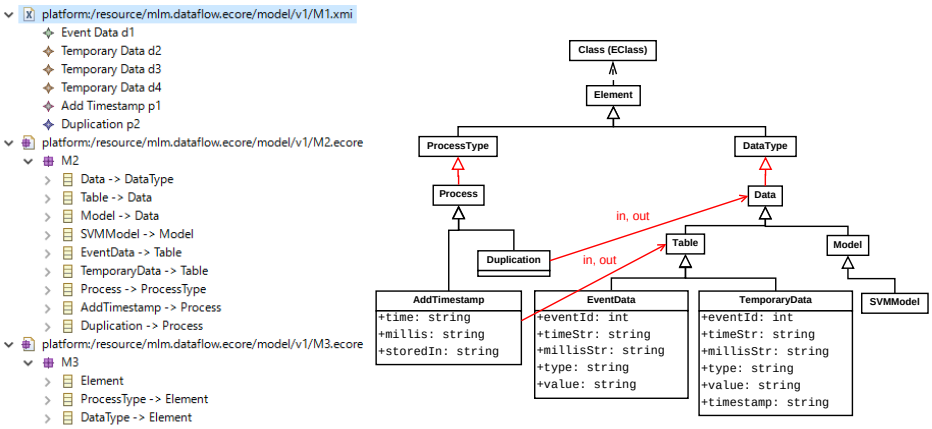

Fig. 2. Dataflow model by EMF.

Fig. 3. Class diagram of Fig. 2.

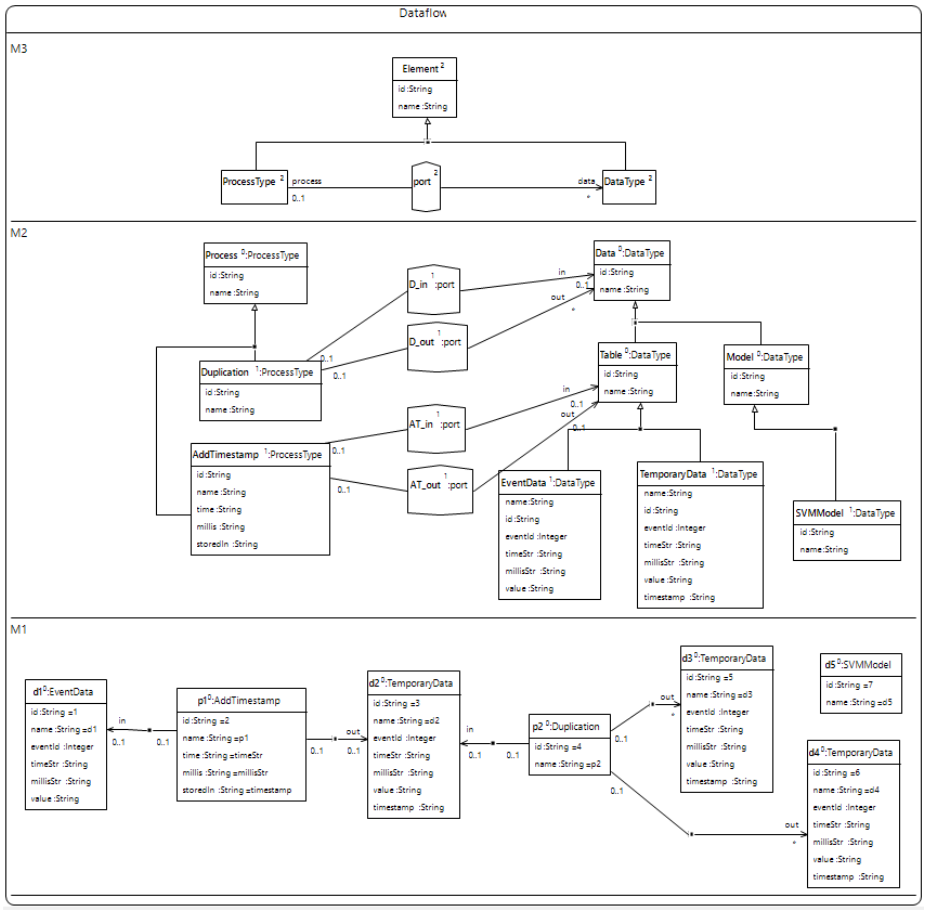

Fig. 4. Dataflow model by Melanee.

```
Model M3@2 {
  Node Element { _id: String; name: String; }
  Node DataType : Element    { process: ProcessType[*]; }
  Node ProcessType : Element { data: DataType[*]; }
  Edge port (DataType.process, ProcessType.data) {}
}
M3 M2 {
  abstract DataType Data
    { _in: Process[0..1] {process}; _out: Process[0..1] {process}; }
  abstract DataType Table : Data {}
  DataType EventData : Table
    { eventId: int; timeStr: String; millisStr: String; value: String; }
  DataType TemporaryData : Table
    { eventId: int; timeStr: String; millisStr: String; value: String;
      timestamp: String; }
  abstract DataType _Model : Data {}
  DataType SVMModel : _Model {}
  abstract ProcessType Process {}
  ProcessType AddTimestamp : Process
    { _in: Table[0..1] {data};
      _out: Table[0..1] {data};
      time: String;
      millis: String;
      storedIn: String; }
  ProcessType Duplication : Process
    { _in: Data[0..1] {data};
      _out: Data[*] {data}; }
  port AT_in
    (Data._out, AddTimestamp._in) {}
  port AT_out
    (Data._in, AddTimestamp._out) {}
  port D_in
    (Data._out, Duplication._in) {}
  port D_out
    (Data._in, Duplication._out) {}
}
M2 M1{
  EventData d1
    { _id = 1; name = d1; }
  AddTimestamp  p1
    { _id = 2; name = p1;
      time = timeStr; millis = millisStr;
      storedIn = timestamp; }
  TemporaryData d2 { _id = 3; name = d2; }
  Duplication   p2 { _id = 4; name = p2; }
  TemporaryData d3 { _id = 5; name = d3; }
  TemporaryData d4 { _id = 6; name = d4; }
  AT_in (d1, p1);
  AT_out(d2, p1);
  D_in  (d2, p2);
  D_out (d3, p2);
  D_out (d4, p2);
}
```

Fig. 5. Dataflow model by MetaDepth.

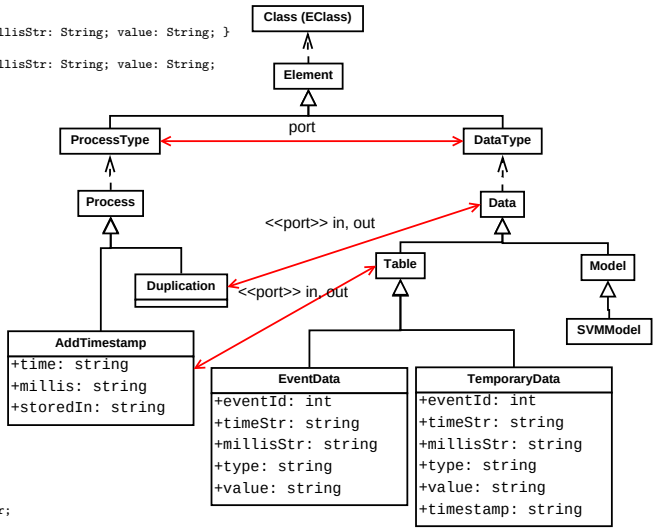

Fig. 6. Class diagram of Fig. 5.

used the simple static methods for achieving extensibility; instantiation between level **M3** and **M2** are defined as inheritance relations, and instantiations of association `port` are omitted.

Melanee[4] is a multi-level modeling framework implementing OCA based on EMF, and has plugins for supporting Object Constraint Language (OCL) [7] and ATL Transformation Language (ATL)[8]. The top-level metamodel of Melanee has elements to distinguish the types of associations such as `Connection` and `Inheritance`. Melanee has a graphical modeling editor, and we can edit multiple levels on the canvas of the editor at the same time.

MetaDepth[7] is a potency-based multi-level modeling framework. The top-level metamodel of MetaDepth mainly consists of `Node` and `Edge`. The instances of `Node` and `Edge` have `Field`s for defining their attributes. MetaDepth supports Epsilon languages[9] for model transformation, model validation and so on.

[7] `http://projects.eclipse.org/projects/modeling.mdt.ocl`
[8] `https://eclipse.org/atl/`
[9] `http://www.eclipse.org/epsilon/`

Table 3. # of limitations for defining the original model.

Framework	# of limitations
EMF	3
Melanee	0
MetaDepth	1

3.1 Capability for Modeling

We defined the dataflow model shown in Fig. 1 on each framework, and confirmed existing limitations for defining the original model by each framework. Table 3 shows the number of limitations for defining the original model by each framework.

Figures 2 and 3 show the dataflow model defined by EMF. We found there are three limitations; 1) inheritance relations from `ProcessType` and `DataType` should be used instead of instantiation relations, 2) association between `ProcessType` and `DataType` cannot be created, and 3) the associations from `Duplication` and `AddTimestamp` cannot be defined as the instances of the association of `ProcessType`.

Figure 4 shows the dataflow model defined by Melanee. Melanee can completely define the original dataflow model as it is, because it can represent instantiation relations between levels and instantiate references of level **M3** at level **M2**.

Figure 5 and 6 show the dataflow model defined by MetaDepth. We found there is a limitation; Although MetaDepth can instantiate references of level **M3** at level **M2**, the references cannot be defined as unidirectional relations. This means `Data` and `Table` also have references to `Duplication` and `AddTimestamp`, respectively, and it causes violations of the original connective rules between types of `Process` and `Data`.

3.2 Extensibility

We consider that there are two kinds of extentions of graphical editing tools: adding a new building block to palette and introducing a new notation in the diagrams. In accordance with the two kinds of extentions, we used the following two scenarios.

A) Adding a new type in level M2: as shown in Fig. 7, we added process type `Filter` in level **M2**. `Filter` is the process type that picks up data records satisfying the conditions given by attributes of the process.

B) Adding a new conceptual type in level M3: as shown in Fig. 8, we added `ControlType` in level **M3**. `ControlType` is a conceptual type for configuring execution of real-world software and has references to `ProcessType` and `DataType`. We also added `Control` and `Schedule` in level **M2**. They are instances of `ControlType`, and `Schedule` is the type of elements that schedule the execution of the referenced process.

Table 4 shows the number of modifications of existing parts as a result of extending the model for each framework. We found we do not have to modify anything in both scenarios using EMF. Regarding Melanee, we just had to

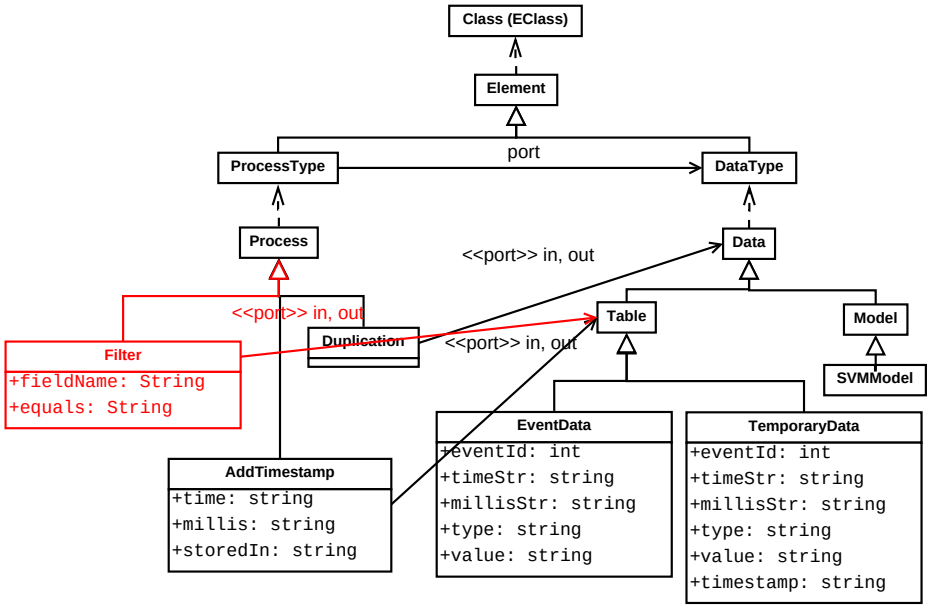

Fig. 7. Adding a new type, `Filter`, in level **M2**.

Table 4. # of existing parts we have to modify in two scenarios.

Framework	Scenario A	Scenario B
EMF	0	0
Melanee	1	1
MetaDepth	0	4

modify the element representing the inheritance relation among `Process` and its subclasses in scenario A. There is only one modification of the inheritance relation among `Element` and its subclasses in Scenario B as well. Regarding MetaDepth, scenario B requires four modifications of bidirectional references between process types and data types.

4 Discussion

We could say that Melanee is more suitable than the other two frameworks to define models of graphical editing tools. Regarding the capability of modeling, Melanee can define the original dataflow model most precisely. Regarding the extensibility, EMF has the least modifications in both scenarios. Melanee also has the least modifications next to EMF.

The dataflow model having two conceptual types and a unidirectional relation between the types in its metametamodel has the generality as the models of graphical editing tools, because there are various kinds of graphical editing tools for defining procedures such as dataflow, workflow and process flow.

We consider that the results of conducting two scenarios for extending models also have the generalities because of the following facts. Regarding the scenario

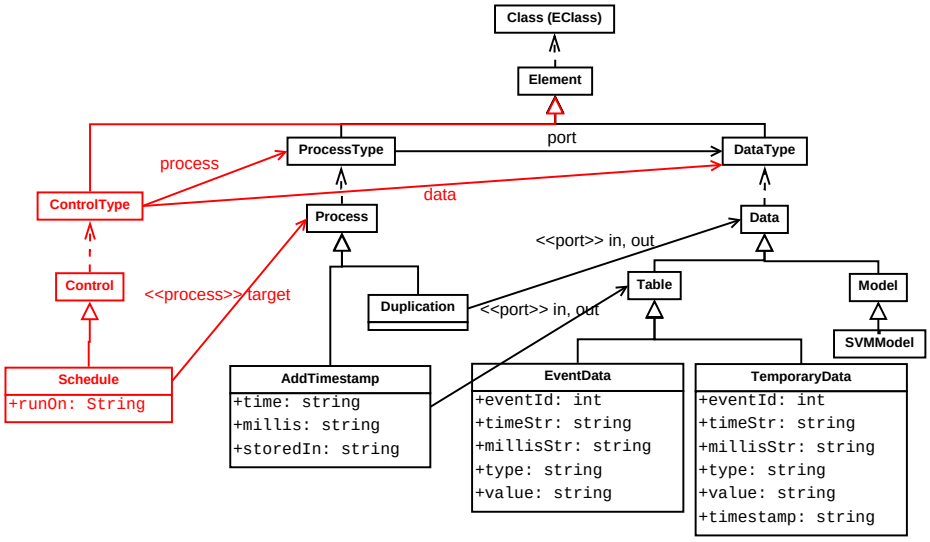

Fig. 8. Adding a new conceptual type, `ControlType`, in level **M3**.

A, the evaluation result basically depends on the number of additional types in level **M2**. Adding process types or data types just causes the changes of inheritance relations. Regarding the scenario B, the evaluation result depends on the number of additional relations between conceptual types in level **M3**. If Melanee needs modifications of relations, MetaDepth generally needs much more modifications than Melanee.

We consider that there is a limitation in our conclusion. The dataflow model contains only two of the five patterns shown in [13]: the *relation configurator* pattern for defining relations in metamodels by instantiating relations in metametamodels, and the *element classification* pattern for representing classfications of types in metamodels. We have to conduct evaluation by applying to other models in order to confirm whether or not our conclusion is valid for other three patterns.

5 Conclusions

In this paper, we have evaluated the following frameworks regarding the capability for modeling and their extensibility: EMF, Melanee and MetaDepth. We have used the dataflow model as an example of graphical editing tools, which is five-level metamodels including MOF at the top level.

The evaluation results show that Melanee is more suitable than the other three frameworks for graphical editing tools. Melanee can represent the dataflow model most precisely and has less modifications for extending metamodels and metametamodels. However, we consider that the multi-level modeling frameworks including Melanee still have difficulities regarding developing graphical editing tools and need more compatibilities with MDE tools to facilitate the tool developments.

References

1. Multi-level modeling wiki, http://homepages.ecs.vuw.ac.nz/Groups/MultiLevelModeling/WebHome
2. Atkinson, C., Gutheil, M., Kennel, B.: A flexible infrastructure for multilevel language engineering. Software Engineering, IEEE Transactions on 35(6), 742–755 (Nov 2009)
3. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. Software, IEEE 20(5), 36–41 (Sept 2003)
4. Atkinson, C., Gerbig, R.: Flexible Deep Modeling with Melanee. In: Reimer, S.B.U. (ed.) Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband. Modellierung 2016, vol. 255, pp. 117–122. Gesellschaft für Informatik, Bonn (2016), http://subs.emis.de/LNI/Proceedings/Proceedings255/117.pdf
5. Atkinson, C., Kennel, B., Goß, B.: The level-agnostic modeling language. In: Software Language Engineering, pp. 266–275. Springer (2011)
6. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, pp. 19–33. Springer (2001)
7. De Lara, J., Guerra, E.: Deep meta-modelling with metadepth. In: Objects, Models, Components, Patterns, pp. 1–20. Springer (2010)
8. Henderson-Sellers, B., Gonzalez-Perez, C.: The rationale of powertype-based metamodelling to underpin software development methodologies. In: Proceedings of the 2nd Asia-Pacific Conference on Conceptual Modelling - Volume 43. pp. 7–16. APCCM '05, Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2005)
9. Henderson-Sellers, B., Gonzalez-Perez, C.: On the ease of extending a powertype-based methodology metamodel. Meta-Modelling and . WoMM 2006 pp. 11–25 (2006)
10. Kimura, K., Nomura, Y., Tanaka, Y., Kurihara, H., Yamamoto, R.: Practical multilevel modeling on mof-compliant modeling frameworks. In: MULTI 2015–Multi-Level Modelling Workshop Proceedings. p. 43 (2015)
11. Kimura, K., Nomura, Y., Tanaka, Y., Kurihara, H., Yamamoto, R.: Runtime Composition for Extensible Big Data Processing Platforms. In: 2015 IEEE 8th International Conference on Cloud Computing. pp. 1053–1057 (2015)
12. Lamo, Y., Wang, X., Mantz, F., MacCaull, W., Rutle, A.: DPF Workbench: A Diagrammatic Multi-Layer Domain Specific (Meta-) Modelling Environment. In: Computer and Information Science 2012, pp. 37–52. Springer (2012)
13. Lara, J.D., Guerra, E., Cuadrado, J.S.: When and how to use multilevel modelling. ACM Trans. Softw. Eng. Methodol. 24(2), 12:1–12:46 (Dec 2014), http://doi.acm.org/10.1145/2685615
14. Neumayr, B., Jeusfeld, M.A., Schrefl, M., Schütz, C.: Dual Deep Instantiation and Its ConceptBase Implementation. In: CAiSE. pp. 503–517. Springer (2014)
15. Neumayr, B., Schrefl, M.: Abstract vs concrete clabjects in dual deep instantiation. In: MULTI 2014–Multi-Level Modelling Workshop Proceedings. pp. 3–12 (2014)
16. Nomura, Y., Kimura, K., Kurihara, H., Yamamoto, R., Yamamoto, K., Tokumoto, S.: Massive event data analysis and processing service development environment using dfd. In: Services (SERVICES), 2012 IEEE Eighth World Congress on. pp. 80–87 (2012)