# Research on Historical Concept Modeling Language and Modeling Tool based on Model Driven Architecture

Mingzhu Ma,Hongxing Liu
College of Computer and Artificial Intelligence
Wuhan University of Technology
Wuhan, China
1464419897@qq.com

Zhenqi Li
Server R&D
Wuhan Baijia Cloud Technology Co., Ltd.
Wuhan, China
782371833@qq.com

***Abstract*—This paper proposes a framework for a model-driven historical compilation method. The basic idea is to distinguish historical conceptual models and historical works models, so as to support large and medium-scale historical compilations of "top-down, gradual refinement". It focuses on defining the historical concept modeling language, and designing its abstract grammar, concrete grammar and semantics. Based on MetaEdit+, a preliminary graphical historical concept modeling tool is designed and implemented.**

***Keywords-component; model-driven; historical concept modeling language; historical concept modeling tool***

## I. INTRODUCTION

The compilation of historical works is referred to as historical compilation. In the traditional compilation practice, after collecting certain historical materials and documents, historians integrate relevant historical materials and documents, and finally describe history through words. The problems brought about by this compilation process are the lack of a unified and recognized compilation process and method, the lack of detailed definition and division of the compilation process, and the lack of high-level professional auxiliary tools. This article considers the application of modeling ideas in the computer field to the field of historical compilation to assist historians in the compilation of history. In this process, through the historical modeling, the work results of each stage of historical compilation are formally expressed.

This paper attempts to explore the method of computer-assisted compilation of historical works from a higher level. It believes that history is essentially a number of historical models used to describe related events, people, environments, objects and connections in a specific world. Historical model is a representation of history, historical compilation is historical modeling, and the language used in historical modeling is called historical modeling language. Drawing on the successful experience in the field of software and databases, we believe that the compilation of historical works can distinguish conceptual models, logical models, and physical models from abstract to concrete. The historical conceptual model is a basic platform-independent model that can be converted to a variety of platform-related models. In this paper, we focus on the conceptual model of history.

In the second II, we propose a model-driven history compilation method. Section III define the history concept modeling language. In the section IV, we implement a graphical history modeling tool. Section V summarizes the research content of this paper.

## II. MODEL-DRIVEN HISTORICAL COMPILATION METHOD

### A. The basic idea of model-driven architecture

Model-driven architecture (MDA) is a new software development method proposed by the Object Management Organization[1][2]. The basic idea of MDA is to separate the model from the implementation, that is, to separate the expression of the model from the implementation of the model by taking the model as the center. MDA proposed the concepts of Platform Independent Model (PIM) and Platform Specific Model (PSM). Platform refers to a series of resources needed to realize the system. For example, in the field of historical compilation, the technology and carrier of historical compilation can be regarded as platforms. PIM has nothing to do with specific platforms and is a highly abstract model. The PSM is related to the specific platform and is a less abstract model. Under the framework of MDA, the process of software development is that domain experts model the business logic of the system to form PIM, then automatically or semi-automatically convert the PIM to the PSM under each platform, and finally, the source code is generated by the PSM[3][4].

### B. Model-driven historical compilation method

MDA software design idea is introduced into the field of historical compilation to explore the model-driven historical compilation method. The main content of the model-driven historical compilation method is to design and establish historical models, that is, historical modeling. Complete historical modeling includes historical conceptual modeling, historical logic modeling, and historical physical modeling.

The architecture of the model-driven historical compilation method can be represented in Figure 1.

From the perspective of MDA modeling, historical works is essentially a model, which we call historical models. Historical works belong to narrative works. Using narratology theory[5] to analyze historical works, historical works can be divided into "story layer" and "discourse layer". According to the abstract level of modeling, MDA distinguishes platform independent model (PIM) and platform specific model (PSM). Regarding the target

2473-3636/21/$31.00 ©2021 IEEE
DOI 10.1109/DCABES52998.2021.00054

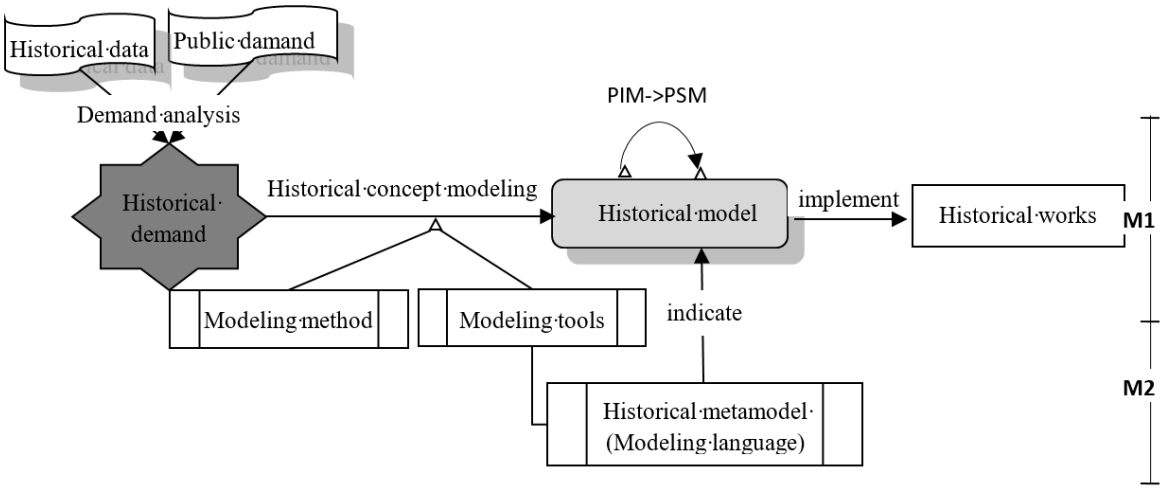



Figure 1. Framework for a model-driven historiography approach

media or carriers of story works (such as books, movies, animations, etc.) as a platform for carrying stories, then the "story layer" of historical works can be regarded as PIM, "discourse layer" " can be regarded as PSM. MDA divides the model into 4 layers. We focus on the model layer (M1) and the meta model layer (M2).

The historical conceptual model is a PIM. When modeling historical concepts, according to the "God's perspective", arrange events based on the actual time when the event occurred, and describe the connection between the events and the connection between the events and the main characters and places. At this time, the rhetoric is not considered. On the basis of historical conceptual models, consider the use of various "discourse" methods, focusing on narrative perspective (such as the third person), narrative time (such as flashback), narrative space and other methods, and finally form historical works (PSM). The model-driven history compilation method fully draws on the experience of database design and other fields, considers the separation of concerns, and the use of auxiliary tools, which helps historians to compile large and medium-sized historical works.

## III. HISTORICAL CONCEPT MODELING LANGUAGE

The historical concept modeling language is a modeling language used to express historical concept models. This section describes the historical concept modeling language from three aspects: abstract syntax, concrete syntax and semantics.

### *A. Abstract syntax*

Abstract syntax defines what types of components exist in the modeling language, and how these components are combined by other components [6]. In this paper, abstract syntax is composed of building blocks and general rules.

This paper uses MetaEdit+[7] meta-model GOPPRR to design building blocks. The building blocks in the historical concept modeling language are designed as follows:

*1)* Design of building block Graph: Create a Graph from Graph Tool and name it "History Modeling Tool". "History Modeling Tool" contains two attributes: Model Name and Model Description, which are used to record the model name and the description of the model.

*2)* The design of the building block Object: The "event" and other elements in the meta-model are designed through Object. There are seven types of Object type elements, as shown in Table I.

TABLE I. ELEMENT OF TYPE OBJECT

| ***Object*** | ***Attributes*** | ***instructions*** |
|---|---|---|
| Start | no | The beginning of the model |
| End | yes | End of model, including model end and process end |
| Event | yes | Major elements in the historical model |
| Character | yes | The person concerned with the event |
| Context | yes | Time, place, etc |
| Article | yes | An important tool, device, invention, etc |
| Timeline | yes | The date of the incident |

*3)* The design of the building block Property: each property contains the property name and property type.The representation of the attribute is as show in formula(1).

$$NP.Property::Type \quad (1)$$

NP represents the non-attribute building block in the meta-model, Property represents the attribute that NP has, and Type represents the type of the Property.

### *B. Concrete syntax*

In this paper, the concrete syntax is the graphical representation of the modeling element. There is a close connection between the graphical representation of the modeling element and the abstract syntax, which is usually defined by mapping the abstract syntax to the notation[8].Figure 2 shows the mapping between abstract syntax and concrete syntax.

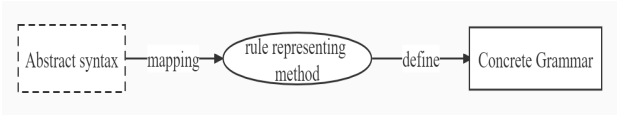



Figure 2. From abstract syntax to concrete syntax

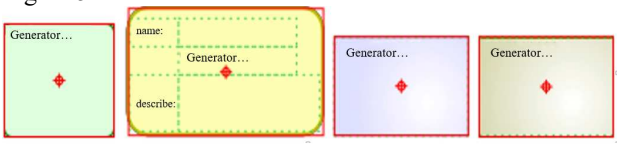
With the help of graphical editing tool "Symbol Editor", specific grammar can be easily realized. The modeling elements of the graphical representation designed in this paper include three types:

*1)* Object elements, including Event, Character, Context and Article.

*2)* Contact elements, including Flow, External Flow, Link, and Projection.

*3)* Role elements including From, To, External Flow From, External Flow To, Link From, Link To, Projection From, and Projection To.

The concrete syntax is shown in Figure 3, Figure 4 and Figure 5.




(a)event (b)Character (c)Context (d)Article

Figure 3. A concrete representation of object

(a) External Flow From (b) Flow From (c) Link From

d) External Flow To (e) Flow To (f) Link To

Figure 4. A concrete representation of a role

(a) External Flow (b) Flow (c) Link

Figure 5. A concrete representation of the relationship icon

### C. *The semantic*

Semantics is the interpretation and logical representation of the meta-model[9][10]. In the paper, the semantics is designed for the field of historical modeling, it is the explanation and description of the historical meta-model. If it is separated from the field of historical conceptual modeling, the semantics is meaningless. In the design of this article, the semantics are mainly restricted by the model interpreter

The model parser is mainly the editor generator definition language MERL. Through the editing of the MERL code, the model can be detected in real time, errors in the model can be found in time, and corresponding prompt information can be given according to actual needs.

As shown in Figure 6, in the model parser, two parts are defined. In the upper rectangular box are the corresponding inspection rules designed. Click each line to display the corresponding MERL code in the lower rectangular box.

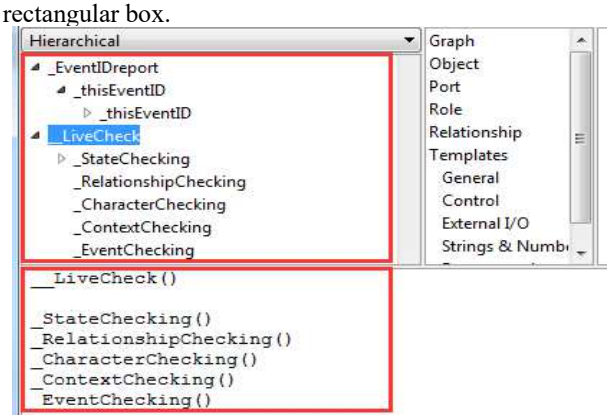



Figure 6. Model parser interface

## IV. DEVELOPMENT OF HISTORICAL CONCEPT MODELING TOOLS

The tool interface includes:

- The layout of object elements and associated elements in the toolbar.
- Drawing area of model graph.
- location of information indicating violation of integrity constraints.
- Display position of elements in the model diagram.

The modeling tool should be able to visually display the connections between events, characters, environments and objects. Historians can create and delete elements of events, characters, environments and objects through historical modeling tools, and be able to edit the attributes of these elements. Two elements can be connected through some kind of connection. When performing these operations, the historical modeling tool should be able to check the integrity constraints in real time. If the integrity constraints are violated, information prompts will be given to facilitate the modification of the model. The interface of the historical modeling tool is shown in Figure 7.

## V. CONCLUSION

This article explores the model-driven historical compilation method, focusing on the historical concept modeling language. It defines the historical concept modeling language from three methods: abstract syntax, concrete syntax and semantics, and represents historical concept models from a higher level of abstraction. Finally, a preliminary historical modeling tool is implemented to support historical concept modeling. The research in this article is only a preliminary exploration of the model-driven history compilation method. Therefore, there are still many deficiencies in the design and implementation of the research and modeling tools. The basic syntax and integrity constraints of the historical modeling language need to be further improved. It can adapt to a more general history.

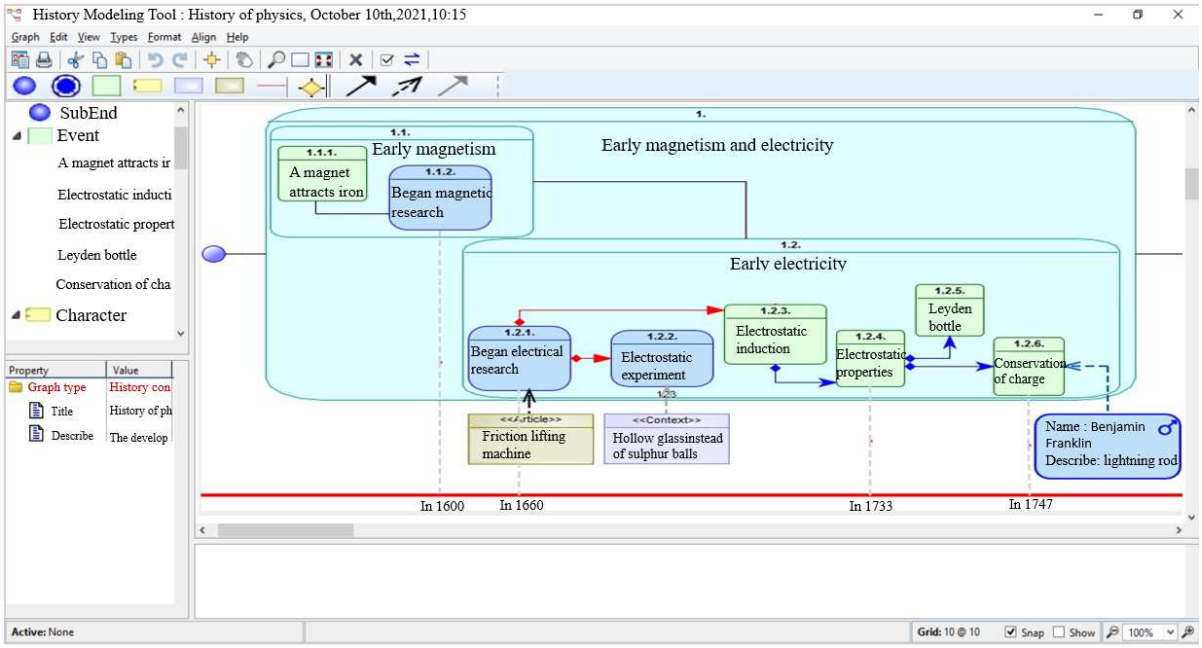



Figure 7. History modeling tool interface

## REFERENCES


[1] OMG. Model Driven Architecture(MDA) Guide rev.2.0[EB/OL]. [2018-01]. http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.

[2] OMG, THE OMG SPECIFICATIONS CATALOG [EB/OL]. [2018-01], https://www.omg.org/spec/.

[3] Hongxu Sun. Research and Realization of MDA Model Conversion Method[D]. Harbin :arbin Engineering University,2012.

[4] Yuanqing Zhang. Design and Implementation of Model Conversion Tool Based on MDA[D]. Harbin :arbin Engineering University,2011.

[5] Yinde Zhang, Narrative Research[C]. China Social Sciences Press,1986.

[6] Wile D S, “Abstract syntax from concrete syntax”, Proceedings of the 19th international conference on Software engineering. ACM, 1997,pp. 472–480.

[7] J.-P. Tolvanen, "MetaEdit+ for collaborative language engineering and language use (tool demo)",Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering, 2016, pp. 41-45.

[8] D. Bork, D. Karagiannis and B. Pittl, "Systematic analysis and evaluation of visual conceptual modeling language notations", 2018 12th International Conference on Research Challenges in Information Science (RCIS),2018,pp.1-11.

[9] D. Durisic, C. Motta, M. Staron and M. Tichy, "Co-Evolution of Meta-Modeling Syntax and Informal Semantics in Domain-Specific Modeling Environments — A Case Study of AUTOSAR", 2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2017, pp. 189-198.

[10] Dániel Urbán, Zoltán Theisz and Gergely Mezei,” Self-describing operations for multi-level meta-modeling”, 6th Internationalh hokgk f Conference on Model-Driven Engineering and Software Development, 2018, pp. 519–527.