Design and Implementation of Automatic Code Generation Method Based on Model Driven

Hailang Chen
School of EIE
Heyuan Polytechnic
Heyuan, Guangdong 517000, China
Email: chenhailang522@foxmail.com

Abstract—With the continuous development of software development technology, people have summarized more and more repetitive work in the process of software development. These repetitive tasks increase the developer's workload of repetitive coding, extend the market demand response time, and then increase development costs. At the same time, in the team development process, due to the uneven code writing ability among the team members, the coding style is not uniform, and the quality and maintainability of the code cannot be guaranteed. The article builds a software development model from the perspective of public functions such as software addition, deletion, modification, and query. On this basis, a code automatic generation engine based on Rete algorithm is designed to realize rapid software code generation. Application practice shows that this method can quickly generate codes with uniform coding style, and improve software development efficiency and code quality.

1. Introduction
With the continuous deepening of the information age, there are more and more types of software and strong demands. Therefore, whether the software development efficiency can be improved on the premise of ensuring the quality of the software, in order to quickly respond to market demands is the key concern of the software industry. Although most of the existing software development has a modeling process, such as waterfall model, fountain model, spiral model and incremental model, system modeling is mainly aimed at the analysis and design of the system in the early stage of development, and the later implementation still needs developers to manually write code, which does not really increase the efficiency of software development. In addition, many developers will use UML or other modeling languages to describe the process of software development projects in the early stages of software development. A large number of documents and design models will be generated in the early stage of development, and these documents and models only exist in the design stage. When entering the implementation stage, the value of existing documents and design models cannot be reflected as they should.

Aiming at the problems of long software development cycle, disconnection of software design and implementation, and poor software code reuse, we have found through software functional analysis that although there are many types of softwares at present, their main functions such as implementation are still based on data addition, modification, deletion and search. and different software is mainly distinguished by different business processes and interface styles. Therefore, through the study of automatic code generation technology, continuously improving the level of
software abstraction and liberating software development from increasingly complex technologies and endless frameworks will exert important meanings in accelerating the rapid prototyping of software, solving code reuse issues, and improving development efficiency.

2. Code automatic design

2.1. Basic principles of automatic code generation

The operation of the software is the operation of the data, and the operated data is mainly database tables or entity objects [1]. The development process is the process of creating and superimposing program pages realized by operation, and the number of program pages is a limited set. According to the finite state automaton theory, the management information system code generation process can be represented by a $M=(P, D, O, \delta, p0)$ quintuple, of which:

- $P$ represents a limited set of pages,
- $D$ is a limited set of data,
- $O$ is a limited set of operations,
- $\delta: (P \times D \times O) \rightarrow P$, called the transition function
- $p0 \in P$ is the initial state, that is, the state where the target source code has not been generated.

According to the above abstract definition of the automatic generation process of management information system code, a code automatic generation tool can be designed in the following way: assuming that it is in the initial state of raw code, when the code engine accepts operation input instructions, the code engine combines the page generation rules (That is, the code template) After the transformation of the movement function is completed, the target source code can be generated, and the next state is entered, that is, $\delta(pn-1,d,o)=pn$ ($n \in N^*$).

2.2. Model design for automatic code generation

According to the basic principles of code generation, this paper builds a data model as an input to the code generation process to provide the necessary parameter information for automatic code generation. At the same time, in order to better realize software reuse, the software code structure is similar or the public part is analyzed and studied to complete the customization of the code generation model. After the template engine receives the model data input, it processes the data through parsing, matching, and encoding to achieve the target source code output. The model of code generation is shown in Figure 1.

![Figure 1 Design of code automatic generation model](image-url)

3. Code automatic generation

3.1. Code template customization

The template is the final prototype of the code [2]. According to the composition of the code, it can be divided into static objects and dynamic objects. Static objects refer to the part of the code that is output directly without changing. This object is usually the target source code that has undergone rigorous
testing. Dynamic objects need to introduce special dynamic labels and be customized through variables or functions. When template analysis is driven by the template engine, the object needs to use the data model to complete the dynamic conversion of the dynamic object to the target source code.

Code template customization usually requires specific customization in combination with the development architecture or language specifications adopted by the information management system to ensure that the final generated target code has high availability and maintainability. This article selects the development specification of MVC management information system based on J2EE as an example, and introduces the process of customizing the code template of the software. The MVC framework can be divided into presentation layer, business logic layer, data persistence layer and domain module layer. It has the characteristics of good reusability, easy maintenance and clear construction structure, which is conducive to separate management of data and templates. According to the various layering characteristics of the MVC framework, this study needs to design four types of templates for generating target source code for each layer of J2EE. The template definition automatically generated by the code is shown in Table 1.

Table 1 Defined code templates automatically generated

| Template layer    | Template name | Template description                                      |
|-------------------|---------------|-----------------------------------------------------------|
| Presentation layer| View          | Provide user access page templates                        |
| Control layer     | Action        | Provide user request control template                      |
| Business logic layer| Service      | Provide business logic processing templates when calling related models |
| Data access       | Dao           | Data persistence template                                  |

3.2. XML-based data model customization

The data model is mainly a record file that defines the required entity object data and function operation information of the software. This file needs to be used to replace the dynamic part of the template file to generate the target source code. This article uses the XML file format to customize the data model. XML is a simple, extensible, platform-independent, and widely used markup language, which is convenient for recording and maintaining relational entity object data and function operation information, and is also helpful for the template engine to parse this type of file. The XML-based data model not only provides complete and necessary information for generating target source code, but also a complete maintenance document. When user needs change, only need to make corresponding changes or adjustments to the data model file, and then regenerate the target source code, without modifying the target source code. The defined format is shown below.

Among them:
①key, property and method can be multiple or one
②The type of method is a limited set of options for operations, which mainly include the three major types of operations on relational data, verification data, and irrelevant data. Relational data operations are insert, update, delete, and check on database data. The operation of verifying data is to check the specified data. Irrelevant data operations are used to generate virtual access paths to hide the actual path and jump to the page.

3.3. Template engine based on Rete algorithm

The Rete algorithm is a solution proposed by Forgy for the inefficiency of the production inference engine \[3\]. At present, most rule inference engines use the Rete algorithm as the core inference algorithm, such as POS, CLIPS, etc. are all based on the Rete algorithm \[4\]. Its basic idea is to dynamically build a Rete matching network according to the content, make full use of the two characteristics of temporal redundancy and structural similarity in the rule system, and save the
information left by all past matching processes to sacrifice the space in exchange for the matching efficiency of the system mode[5]. There are four main types of nodes in the Rete rule network: root node (root), type node, alpha node, and beta node. The root node is the entrance to the Rete network and allows all the facts to pass. Type nodes are used to store various types of facts. The alpha and beta nodes are respectively used for attribute testing of a single event and conditional binding testing between multiple events.

This article uses the example of the method target source code generation of the data model (x1, x2, x3, x4, x5, x6 corresponding to insert, update, delete, search, validate, and virtual, respectively), and establishes a partial template engine inference network based on the Rete algorithm (As shown in Figure 2), where:

X1: generating a target source data operation type is "increase", the corresponding type of node U;
X2: generating a type of "modified" in the source data operation a target, corresponding to the node type is U;
X3: generating a type of "delete" operation of a target data source, corresponding to the node type is U;
X4: generating a type of "query" method of operation target data source, corresponding to the node type is U;
X5: generating a type of "check" operation of a target data source, corresponding to the node type are C;
X6: generating a type of "virtual path Access" method of operation target data source, corresponding to the node type V.

As shown in Figure 2, when performing rule inference on target source code generation, fact data (Facts) is the template file data to be inferred. It is described in terms of triples (objects, attributes, values). For example, the data modification method for generating user entity objects can be described as (Users entity object, Method, update). The pseudo-code implementation of the template engine based on the Rete algorithm is described as follows:

Algorithm input: list of triples of entity data Facts = {Fa1, Fa2, ..., Fan};
Algorithm output: rules Ru matched by the code generation model , R={∅, Ru1, Ru2, ..., Run};
Rete Match (Facts)
1. Result to ∅ //Initial match result
2. Get Next Rule (Nets) to rule //Remove rule patterns one by one from Nets
3. WHILE rule IS NOT NULL //Determine whether the current detection mode exists in the corresponding AlphaNode
   4. rule is TRUE //The current detection mode has a identification
   5. FOR pattern in rule.patterns //Build JoinNode in Beta network
   6. Pattern is FALSE //Beta network node processing status indicator
   7. FOR fact in rule.Facts //Process entity data to be inferred
   8. match (fact, pattern) to Pattern
   9. IF Pattern is TRUE BREAK; // Jump out of the current FOR loop and repeat step 5
   10. END FOR
   11. IF Pattern is FALSE
   12. Rule is FALSE //The current detection mode is processed
   13. BREAK;
   14. END IF
   15. END FOR
   16. Get Next Rule (Nets) to rule
   17. RETURN rule; //Return the matched rule

4. Application and effect analysis of automatic code generation
This article takes the "Heyuan City Letters and Calls Information Management System" as an example to complete the production and deployment, and conducts statistical analysis on the code generation and final release code using this code automatic generation tool to complete the description of the role and effect of the tool. The code generated by the code generation tool refers to the original generated code that has been modified, and the final release code refers to all the code when the system is officially put into use. In order to obtain code-related statistics, this article uses the “Source Counter” source code statistics tool to perform statistics on the code generated by this code automatic generation tool and the final released code. Before using “Source Counter” statistics, set the coding productivity to 4000 lines per person per month, the number of working days in a month is 20 days per person, and the monthly cost is 10,000 yuan per person. According to the results obtained by the statistical tool and its corresponding analysis, the results are shown in Table 2:

|                | Code generation tools | Code released |
|----------------|-----------------------|---------------|
| Statistics     | Rows                  | Ratio         | Rows                  | Ratio         |
| Total rows     | 143768                | 100%          | 169834                | 100%          |
| Code lines     | 104761                | 74.63%        | 127367                | 75.00%        |
| Annotation lines| 26142                 | 17%           | 28873                 | 17%           |
| Blank lines    | 12865                 | 8.46%         | 13594                 | 8%            |
| Total cost     | 286902.5              |               | 318417.5              |               |

It can be known from Table 2 that the code generation ratio developed by this information system can reach 84.65%. Assuming that a team of 5 people coded the project, it would take about 6 months to complete without using the automatic code generation tool. After using the automatic code generation tool, it only took one and a half months to complete the coding work.

5. Conclusion
This article uses finite state automaton theory to discuss the basic principles of code generation
technology, and on this basis, the code automatic generation model is designed from the perspective of software functionality. The model mainly includes template files and code generation engines. An XML-based model customization method is proposed and implemented, and an automatic code generation engine based on Rete algorithm is constructed. Application examples show that this method of automatic code generation can complete most of the code writing work, thereby shortening the development cycle of the system and reducing the difficulty of development. At the same time, limited to the model performance capability of the code generation technology design, for complex and difficult development needs, the automatic code generation ratio can not reach 100%. In the future, we can continue to improve the ability to automatically generate code by performing higher-level abstractions on public modules and extending code customization.

Acknowledgments
This work was supported in part by the Fund for “Promoting the reform of higher education by using big data technology, Energizing teachers and students to explore the future” (FPE) and Natural Science Foundation of Guangdong Education Department. We would like to thank all sponsors and reviewers for their detailed comments on our paper.

References
[1] J ANDRADE, J ARES, An architectural model for software testing lesson learned systems, Information and Software Technology, vol.55, no.3, 2012, pp.18-34
[2] Wu Budan, Automatic code generation of workflow application system based on template, Journal of Huazhong University of Science and Technology, vol.12, no.2, 2013, pp.18-21.
[3] Forgy C L, Rete: a fast algorithm for the many pattern/mane object pattern match problem. Artificial Intelligence, vol.19, no.1, 1982, pp.17-37
[4] Sun Xindeng, An improved Rete algorithm based on sharing model. Journal of Automation, vol.43, no.9, 2017, pp.1571-1579.
[5] Tong Fei Zhou, Yu Qing Pan, Li Rong Huang, (2017) Research on Personalized E-Learning Based on Decision Tree and RETE Algorithm. In: International Conference on Computer Systems, Electronics and Control (ICCSEC). Dalian, China. pp.1392-1396.
[6] Chenghua Tang, Yi Xie, “An Improved Object-Oriented Rete Algorithm and Network Structure Mode,” In: 2nd International Symposium on Information Engineering and Electronic Commerce. IEEE Symposium on. IEEE, 2010, pp. 1-4.