Intelligent management of hot stamping production: dynamic job configuration based on rule engine

Min Li[1,†], Hua Rao[1], Weiyong Li[1], Jiarui Wang[1], Zhiming Deng[1], Danfan Zeng[1] and Yisheng Zhang[2]

[1]*School Of Information Engineering, Jiangxi University of Technology, Nanchang 330029, China*
[2]*State Key Laboratory of Materials Processing and Die & Mould Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[†]*E-mail: limin85@jxut.edu.cn*
www. jxut.edu.cn

The swift evolution of automobile products results in frequent changes to the business rules governing the hot forming production line. The strong connection between the original IT system's business rules and application code leads to limited flexibility and dynamism within the core module of the job-shop scheduling system. To address this, a development method based on a rule engine is proposed during the transformation of the original system process rule management. This approach enhances the application of the Rete algorithm, enabling dynamic configuration of business rules, fostering system universality, and offering new insights for project development feasibility.

Keywords: Hot stamping; Job scheduling; Rule Engine; Process management; Industry 4.0.

1. Introduction

Body lightweight is one of the effective approaches to achieving energy efficiency and emission reduction in the automotive industry [1]. Hot stamping technology, as a highly promising technique for automotive body lightweight, has witnessed an increasing frequency of changes in its production process due to market-driven demands such as product diversification and replacement [2,3]. With the advent of Industry 4.0, intelligent manufacturing has become the core of the new generation industrial revolution, where intelligent business process management plays an indispensable role in realizing this vision. In the IT systems of traditional hot stamping production enterprises, business rules are often implemented through program code, which is embedded within other business code. This can result in high costs for system upgrades and maintenance when frequent manufacturing requirements and business rule changes occur. In some cases, it may even require a complete system redesign or restructuring. The Rule Engine, derived from AI (Artificial Intelligence) systems, provides functionality for defining, registering, and managing business rules. It ensures the consistency of business rules and determines their relationships. Typically, the rule engine serves as a crucial component of the business rule management system and can enforce one or more business rules by adjusting their priority order.

This paper focuses on the transformation of the core business process management system in a hot stamping production enterprise. It discusses the application of a

Y. Zhang and M. Ma (eds.), *Proceedings of the 7th International Conference on Advanced High Strength Steel and Press Hardening (ICHSU 2024)*, Atlantis Highlights in Materials Science and Technology 3,
https://doi.org/10.2991/978-94-6463-581-2_66

development method based on a rule engine to achieve the separation of business logic and application code. The study explores the composition, architecture design, and key technologies involved in implementing this new development method. The objective is to identify practical solutions to the common challenge of separating business rules from application code in IT systems.

2. Inference steps of the rule engine

A rule engine is a system that applies logical judgments to data and then carries out predefined actions on the data or the system itself [4]. Its primary goal is to detach business logic from application logic, making business logic more autonomous. By using a rule engine, business rules can be separated from program code, allowing business logic rules independence from system program code. This makes the maintenance and management of business rules more efficient [5]. In this new system architecture, business rules can be created, modified, and removed separately by business management personnel without requiring the entire application system to be recompiled and deployed. This greatly enhances the application system's responsiveness and convenience when it comes to rule changes.

In logic, two commonly used algorithms for logical reasoning are induction and deduction. Induction involves deriving the whole from parts and moving from specific cases to general cases, starting from basic facts, relying on experience and empirical evidence, and drawing general conclusions. By contrast, deduction is based on general logical hypotheses. It involves constantly seeking facts that meet the hypotheses to draw specific conclusions [6].

When business requirements are continually changing, and reasoning involves numerous instances, multiple facts are required to meet business rules concurrently, leading to conflicting conclusions. This situation calls for the introduction of a rule engine. Most existing rule engine products use the deduction method. The Rete algorithm, the most widely used efficient deduction algorithm in rule engines, compiles the rule package into a Rete grid. It then matches the facts one by one from the root node to the leaf node layer by layer, recording the matching process in each node, significantly reducing calculation and improving matching speed [7].

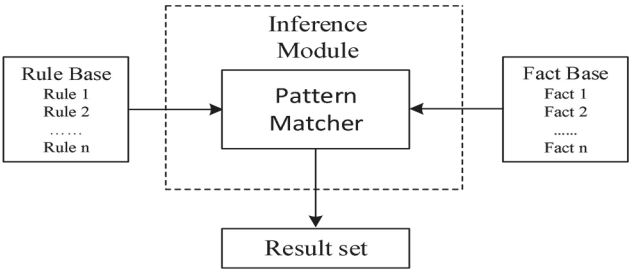

Fig. 1.Structure of the Rule Engine Inference Module

The schematic diagram of the structure of the common rule engine inference module is shown in Fig.1.

The inference module first reads data from the fact base while simultaneously retrieving rules and related data from the rule base. It then performs matching calculations between the rules and facts through the pattern matcher. After successful matching, the result is output to the result set. Finally, any conflicts in the result set are resolved, and the ultimate result is obtained.

3. Dynamic job scheduling system based on rule engine

A typical hot stamping production enterprise encompasses warehousing, uncoiling and blanking processes, a hot stamping production line, a laser production line, a shot blasting production line, as well as parts testing and packaging workshops. The main workflows of a typical hot stamping company is illustrated in Fig. 2.

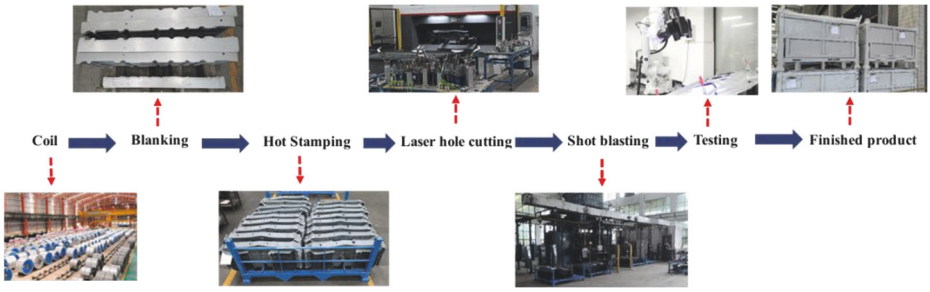

Fig. 2.The main workflows of a typical hot stamping company

The goal of intelligent management in hot stamping production is to maximize the time and space benefits of materials while ensuring the smooth production of the enterprise, and to rationally connect various process links to achieve the best economic benefits.

The basic rules are as follows: 1) Arrange molds with the same task on the same production line as much as possible to reduce the production change time.2) Maximize the utilization rate of each production line. 3) Strictly follow the production plan priority. 4)Strive to eliminate waiting time between processes.

However, as the business needs of the production workshop continue to change, the types and priorities of operations will also change, and the original IT system cannot respond to this dynamic adjustment in a timely manner.

Commonly employed algorithms for resolving workshop job scheduling problems include the simulated annealing algorithm, genetic algorithm, and others. The simulated annealing algorithm is an adaptable and optimized random search algorithm, but its drawbacks include the inability to utilize expert knowledge and slow solving speed. Genetic algorithms have widespread application in workshop job scheduling problems, but they are prone to premature convergence issues, leading to local optimization solutions, and also possess limitations in coding adaptability. Additionally, production job scheduling

problems have numerous influencing factors and unique challenges, demanding advanced optimization algorithms [8].

Given on-site research findings and enhancements to traditional algorithms, a dynamic job scheduling model based on the rule engine for real-time transaction processing (as depicted in Fig.3) has been designed. In this new model, instead of listing all job plans and then performing job scheduling combinations as with traditional shop floor job scheduling solutions, each job currently in progress or pending is entered into the rule engine for real-time optimization and solution. This rule engine is based on a series of business rules solved by the rule inference device, and the solution results are placed in the rule execution queue. Finally, these results are executed in sequence by the execution engine, with feedback provided to the job queue.

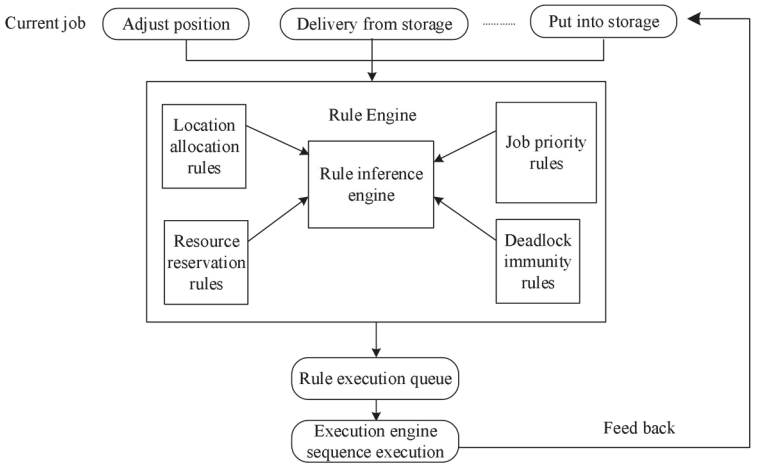

Fig. 3.Dynamic job scheduling model based on rule engine

4. Key Technologies

The Rete algorithm is currently the most efficient forward chain inference algorithm [9]. Its core idea involves compiling the rule package into a Rete grid, matching the facts in the working memory layer by layer from the root node to the leaf node, and recording the matching process in each node of the tree to significantly reduce computational complexity and improve matching speed [10].

The Rete algorithm constructs a knowledge base into a rule network, which includes Root nodes, α Nodes, β Nodes, and Result nodes. The process of the rule network matching for nodes in the original Rete algorithm is depicted in Fig.4. The Root node classifies facts and guides them into the corresponding α Nodes. α Nodes filter out some facts through fixed pattern matching, enhancing the matching efficiency of rule networks. β Nodes are primarily used to match the relationship between facts and store intermediate fact results, constituting an essential part of the inference process. The fact set follows the root node of the rule network, matches nodes α and β until all facts match the Result node, resolving conflicts and obtaining the final result.

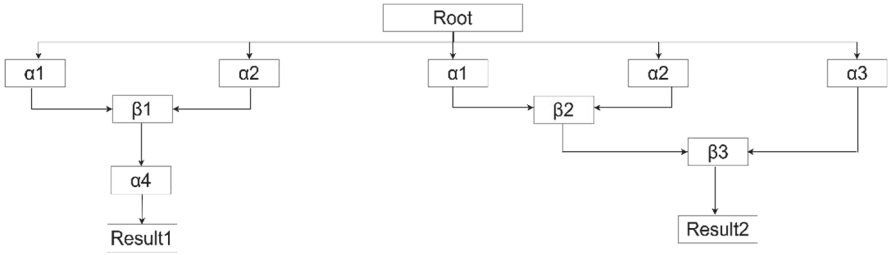

Fig. 4. Rule network matching process of the original Rete algorithm

The improved Rete algorithm primarily eliminates the redundancy of network nodes. During the construction of the Rete network, all nodes are scanned and checked, ensuring that nodes with the same rules are not duplicated. This approach saves storage space and accelerates execution efficiency. The rule network matching process of the improved Rete algorithm is illustrated in Fig.5.

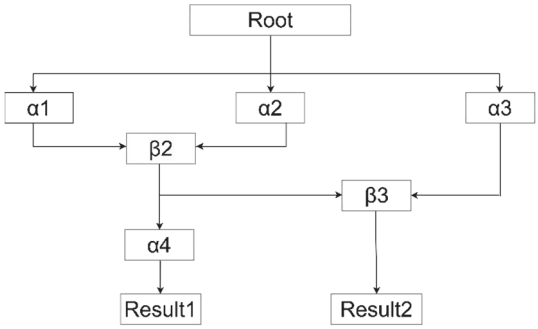

Fig.5.Rule network matching process of the original Rete algorithm

The rule engine first compiles the execution rules. It utilizes the improved Rete algorithm to reuse all α nodes after establishing the root node, then filters out part of the content through the matching rules to obtain the β node, saving the intermediate matching results. After that, it performs rule matching filtering and finally arrives at the Result node to eliminate all conflicts, achieving the final execution result through the execution engine.

5. Experimental Design

The function module for searching steel coil storage locations for overhead crane operations in hot stamping production enterprise is tested using both the rule-based engine development mode and the traditional mode. The primary testing content is focused on resolving the speed problem.

The solution model is segmented into five sets of data for testing and the average matching time is compared. The number of matching locations gradually increases from 1 to 10,000. Based on the functional code implemented by the rule engine development method, when the number of matching repositories increases from 1 to 10,000, the time for rule inference calculation gradually increases. For instance, when 1,000 repositories are involved, it takes only 0.08 seconds for the rule engine to provide the correct job scheduling

arrangement, and when 10,000 repositories are participating in the matching calculation, it only takes 0.78 seconds. The detailed test data is shown in Table 1.

Table 1 Time required for rule-based development methods

Number of storage places	Group 1	Group 2	Group 3	Group 4	Group 5	Average matching time (ms)
1	10.7	10.6	10.7	10.8	10.7	10.7
100	23	14	16	18	20	18.2
1000	78	65	92	80	80	79
10000	791	778	767	788	788	782.4

Table 2 depicts the time required for calculating the number of library locations based on the function code implemented by traditional development methods.

Table 2 Time required for traditional development methods

Number of storage places	Group 1	Group 2	Group 3	Group 4	Group 5	Average matching time (ms)
1	21	22	28	28	25	24.8
100	58	57	56	64	55	58
1000	291	327	312	335	324	317.8
10000	1290	1301	1352	1265	1223	1286.2

Through the above analysis of the results of the equal matching calculation test of the systems developed in the two development modes, the results are shown in Fig.6.

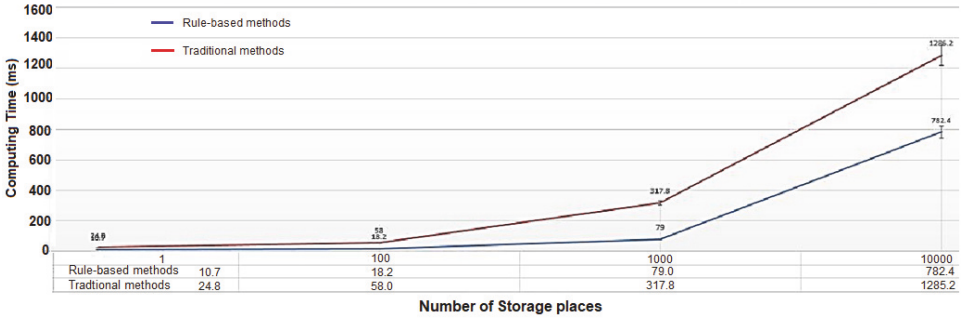

Fig. 6. Comparison of time consumption of equivalency calculation between two development modes

Through the comparative analysis of the results of the equivalent matching calculation test from the two development methods, it is observed that the development mode based on the rule engine is notably shorter than that based on the traditional development mode, resulting in significantly improved efficiency. Additionally, the development mode based on the rule engine presents two distinct advantages:

(1) Flexibility improvement: Business rule changes are efficiently implemented by modifying only the rule configuration file instead of the business program code, allowing timely implementation of the changes without requiring system re-deployment and recompilation.

(2) Centralized rule management: All business rules are centrally housed in the rule base for management, offering a user-friendly interface that facilitates non-technical personnel to directly modify the business rules.

6. Conclusion

This paper introduces a new development mode based on the rule engine, enhancing system flexibility and versatility by reducing the coupling between business rules and code. By dissecting the core business process of hot stamping production enterprise, a dynamic operation scheduling model based on the rule engine is established. Business rules are implemented in the rule engine and put into practice at the production site. The effective improvement of the new development mode is demonstrated through production practice and performance tests, providing viable reference and ideas for project development.

Acknowledgements

This research work is supported by Key Research Project of Science and Technology of Jiangxi Provincial Department of Education, No. GJJ2202603; Management Science Foundation of Jiangxi Province, No. 20232BAA10042.

References

1. H. Karbasian, A.E. Tekkaya. A review on hot stamping, journal of Materials Processing Technology, 210 (2010)2103-2118
2. Zhang Yisheng, Wang Yilin, Zhu Bin, et al. Research progress in hot stamping forming technology based on multi-component integration, *Journal of Plastic Engineering* **30**,1(2023).
3. Y. S. Zhang, Z. J. Wang and L. Wang, Progress in hot stamping process and equipment for high strength steel sheet, *Journal of Plasticity Engineering* **25**, 11 (2018).
4. Ren Bin, Wang Zilin. Flexible Optimized Scheduling of Sheet Metal Production Lines Based on Drools Rule Engine, *High Technology Letters* **32**, 57(2022).
5. Wu Danfeng, Zeng Guangping, Yan Jingying. Research on Rete Algorithm Supporting Evolutionary Rule Engine, *Journal of Computer Application and Research* **30**, 1747(2013).
6. Huang Liefu. Design of a ticket after-sales system based on a rule engine,*Modern Information Technology* **4**, 8(2020).
7. Du Xingyi, Hu Jing, Jiang Wei, Song Tiecheng. Design of IoT data query method based on Drools rule engine, *Measurement and Control Technology* **41**, 71(2022).
8. Fu Xuhui, Kang Ling. Research on the Premature Problems of Genetic Algorithms, *Journal of Huazhong University of Science and Technology (Natural Science Edition)* **7**, 53(2003).
9. You Junxin, Rao Ruonan, Zhan Xiaofeng. Rule-based Web Framework, *Computer Application and Software* **2**, 4(2007).
10. Wang Xiaoguang, Yang Dan. Research on the application of rule engines in distributed environments, *Journal of Computer Application and Research* **26**, 1825(2009).