The Development of a Financial Product Factory System

Seong-cheol Choi[a] and Sang-hoe Koo[b]

[a]*Dept. of Digital Management, Graduate School, Korea University*
1, 5-ka, Anam-dong, Seongbuk-ku, Seoul 136-701, South Korea
Tel: +82-2-3290-3999, Fax: +82-2-3290-3999, E-mail:laugh2r@korea.ac.kr

[b]*Dept. of Digital Management, Graduate School, Korea University*
1, 5-ka, Anam-dong, Seongbuk-ku, Seoul 136-701, South Korea
Tel: +82-2-3290-3999, Fax: +82-2-3290-3999, E-mail:skoo@korea.ac.kr

Abstract

Product factory is a real-time financial product design system for the Internet customers. Recently, as the number of the Internet customers increases, the importance of the product factory becomes more emphasized. However, there is not much research performed regarding its definition, properties, requirements, nor implementation.

In this research, we make a clear definition of product factory, and analyze the requirements of the system from the perspectives of functions and services, and we propose an architecture that reflects the analyzed requirements. In additions, we implemented a prototypical system based on the proposed architecture to prove the usefulness of this research.

Keywords:

Product Factory; Artificial Intelligent in Design; Rule-based System; Case-based Reasoning; Constraint Satisfaction; Truth Maintenance System

Introduction

Recent advances in the Internet and the telecommunication technologies have significant impacts on various industries. Banking industry is not an exception. Many novel services are introduced (electronic cash, smart cards, etc.) and many existing services are automated (account opening, loan applications, etc.), which, in turn, leads to the increase of online bank customers. As the number of online customers increases, customers came to have more information regarding bank firms as well as their products or services. This results in the increased power of customers. The more power customers have, the more demanding customers become. If the bank does not satisfy the demands of customers, they may leave to other banks any time.

According to the recent research [Jeong, 2001], the number of bank products of Korea is almost similar to that of advanced countries, but the diversity in bank products lacks. This lack of diversity is the main reason for not satisfying customers' various demands. The causes for this lack are as follows: new products are still manually designed, and the design processes are not flexible enough to design novel products with ease. A product factory is a real-time financial product design system for the Internet customers. Using this, products that satisfy individual customers' demands are automatically designed. However, unfortunately, there is not much research performed regarding its definition, properties, requirements, nor implementation.

In this research, we make a clear definition of product factory, and analyze the requirements of the system from the perspectives of functions and services, and we propose an architecture that reflects the analyzed requirements. In additions, we implemented a prototypical system based on the proposed architecture to prove the usefulness of this research.

A product factory is an automated financial product design system. Design system involves very complicated cognitive processes. AID (AI in Design) is a field of AI (Artificial Intelligence) research that focuses on design automation using technologies developed in AI. This research is in dept to the research performed in AID [Vancza, 1999] [Papavasileiou et al., 2002] [Mahler et al., 1997] [Chandrasekaran, 1990].

A Financial Product Factory

Definition of Financial Product Factory Systems

To develop a product design system, we need a formal representation of products that the design system can work with. In this research, we represent a product as a set of attributes. And the attributes may take a symbolic value, a set of symbolic values, a numeric value or an interval of numeric values. Using the representation, we can formalize the product design problem as a constraint satisfaction problem as follows:

***Find Product* = $\{a_1, a_2, \ldots, a_n\}$, *given* $\{c_1, c_2, \ldots, c_n\}$**

Where $a_1, a_2, \ldots, a_n$ are the attributes required to define the product, and $c_1, c_2, \ldots, c_n$ are the constraints that the to-be-designed product must satisfy.

Requirements of Financial Product Factory Systems

Though lots of discussions are made regarding product factory systems [Syscorp], financial product factory systems have not been developed so far. To develop a novel system, we first need to identify the requirements of the system. In this research, we identify the requirements of financial product factory systems from the perspectives of functions and services as follows:

1) *Online real-time design system*: The system must be able to design financial product real-time for the online customers by analyzing and satisfying the customers' needs.
2) *Quick response time*: The system must be able to design products quickly enough for online users. To do this, the knowledge required for design must be systemically managed and the design process must be well engineered.
3) *Decision support*: Service must be decision supportive. That is, customers must be able to alter the various options to compare alternative product designs to select the most appropriate one.
4) *Product consistency*: Design service must provide consistent product designs. It is important for the product to satisfy individual customer's needs. However, it is also desirable to provide product designs that do not display much difference from the existing ones.

System Architecture

Considering the requirements discussed in the previous section, we propose a system architecture as shown in <Figure 1>.

The proposed architecture is composed of four modules (shown as rectangles in the figure) including Rule-based typing module, CBR(Case Based Reasoning)-based recommendation module, CS(Constraint Satisfaction)-based design module, and TMS(Truth Maintenance System)-based decision support module. The following statements describe how customer interacts with these modules when he wants to purchase a financial product.

1) Customer enters his financial information such as name, loan purpose, loan amount, loan term, mortgage, etc.
2) According to the information entered, Rule-based typing module selects a proper product type, for examples, mortgage loan or credit loan.

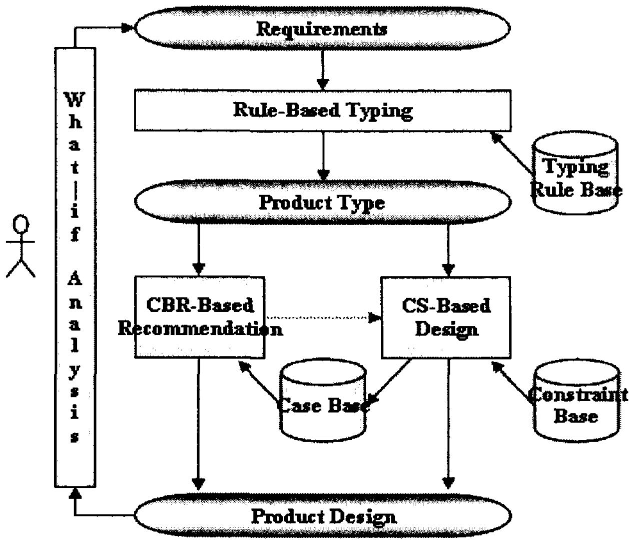


Figure 1 - Architecture of Financial Product Factory

3) Among the past case products in the case base of the selected product type, CBR-based recommendation module selects the most appropriate product according to the information entered.
4) If CBR-based recommendation module fails to select a recommendation, CS-based design module designs and provides a new product, based on various constraints such as customer information or bank policies.
5) TMS-based decision support module supports what-if analysis. Customers may alter various options to view and compare alternative product designs to select the most appropriate one.

Rule-based Typing Module

This module contains various rules useful for selecting product types. In financial firms, product types are usually classified hierarchically. In this research, these rules are managed according to this hierarchy. The rules take "IF-Then" form. The following is an example rule used to select a housing loan.

IF loan-purpose = housing AND
income-level >= average AND
mortgage = yes
THEN loan-type = housing-loan

Case-based Recommendation Module

The case base contains past representative cases each of which is composed of a pair of product description and customer information. The customer information contains not personal information but financial information such as loan purpose, loan amount, loan term, mortgage, etc. Current customer's request is compared against this financial information, and the most similar case is selected, if exists, and selected product is recommended to the customer for purchase.

In this research, the similarity between cases (or products) is a weighted sum of similarities between attributes that comprise the cases. Attribute similarities are measured in various ways according to their types as follows.

1) Similarity between attributes with one symbolic value:

   $sim(a, b) = 1 \text{ if } a = b, 0 \text{ if } a \neq b$

2) Similarity between attributes with more than one symbolic values:

   $sim(a, b) = size(a \cap b) / size(a \cup b)$

   * *size(a)* is the number of attribute values that the given attribute (that is, *a*) may take.

3) Similarity between attributes with numeric values:

   $sim(a, b) = 1 - (|a - b| / max\{difference\})$

   * *max{difference}* means the maximum difference that can be found for the given attribute.

4) Similarity between attributes with interval values:

   $sim(a, b) = length(a \cap b) / length(a \cup b)$

   * *length(a)* is the length of the numeric interval that the given attribute may take as value.

The case similarity is the weighted sum of individual attribute similarities. These weights are assumed to be provided by banking experts.

$$SIM(A, B) = \sum w_i * sim(a_i, b_i), \quad where \sum w_i = 1$$

The case is selected for recommendation when the SIM(A, B) is greater than a given threshold value. The threshold value is also provided by banking experts.

CS-based Design Module

This module designs a product satisfying various constraints. The constraints are posed either upon individual attributes (intra-attribute constraints) or upon relationships between attributes (inter-attribute constraints). These constraints are provides as IF-THEN rules and design process is performed by forward chaining algorithms.

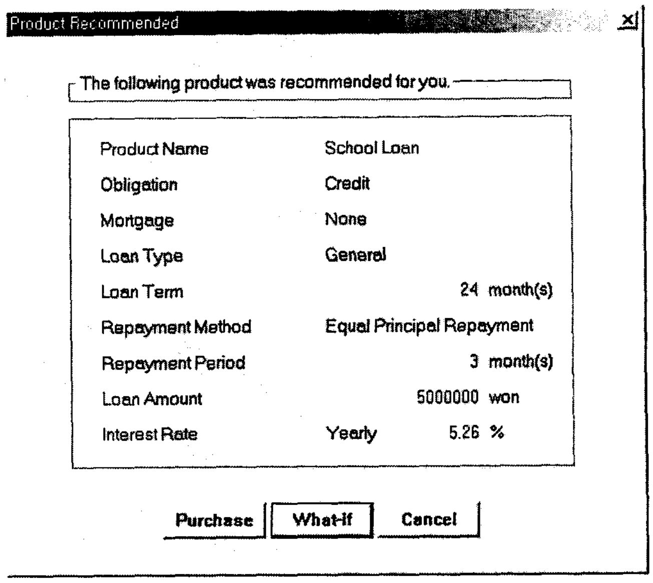
TMS-based Decision Support Module

TMS (Truth Maintenance Systems) provides efficient ways to retract rule firings in production systems [Hindi and Lings, 1994]. To perform what-if analysis, we need to assert and retract facts and rule firings repeatedly. Utilizing TMS in our system, we could successfully provide what-if analysis facility.

Implementation and Test Runs

In this research, we implemented a prototypical system of the proposed architecture, and successfully run representative test cases. For implementation, we use MS Windows 2000 as platform, MS Access as DBMS, and CA's Aion 9.1 as an IDE [CA].

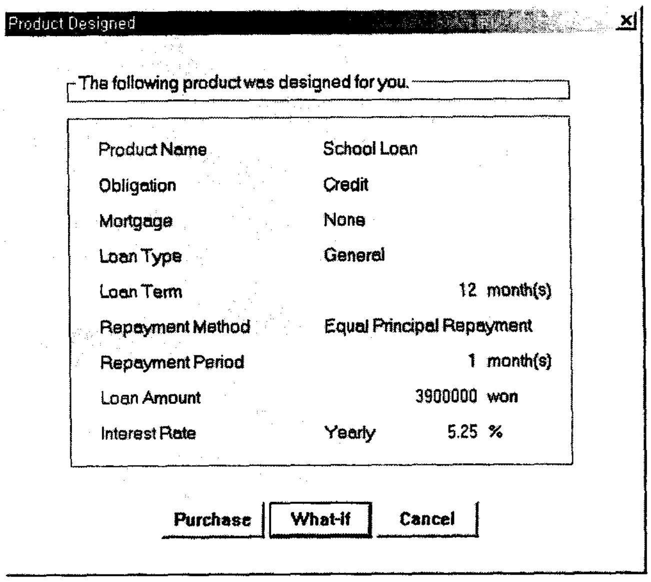
Assuming that the user enters personal information, past transaction history, loan purpose, loan amount, mortgage, etc., <Figure 2> displays the recommended product retrieved from the case base. <Figure 3> displays a product that is designed when the recommendation module fails to find a similar one. <Figure 4> displays a GUI that user may alter input parameters to perform what-if analysis.



The following product was recommended for you.

Product Name	School Loan	
Obligation	Credit	
Mortgage	None	
Loan Type	General	
Loan Term		24 month(s)
Repayment Method	Equal Principal Repayment	
Repayment Period		3 month(s)
Loan Amount		5000000 won
Interest Rate	Yearly	5.26 %

Figure 2 - Recommended Product



The following product was designed for you.

Product Name	School Loan	
Obligation	Credit	
Mortgage	None	
Loan Type	General	
Loan Term		12 month(s)
Repayment Method	Equal Principal Repayment	
Repayment Period		1 month(s)
Loan Amount		3900000 won
Interest Rate	Yearly	5.25 %

Figure 3 - Newly Designed Product

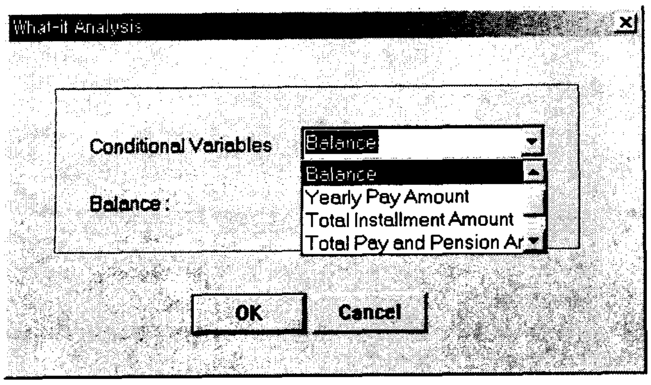


Figure 4 - What-if Analysis

Conclusion

In this research, we made a clear definition of product factory, and analyze the requirements of the system from the perspectives of functions and services, and we propose an architecture that reflects the analyzed requirements. In additions, we implemented a prototypical system based on the proposed architecture to prove the usefulness of this research.

The proposed system has a limitation in the sense that the design process only deals with the design of values of product attributes (value-level design). The system could not design new attributes (attribute level design), and not design new product structures (structure level design). Our future research efforts will be made toward these types of design problems.

Acknowledgments

This research has been performed from the support of Korea University special research fund (#K0202900).

References

[01] Brown, D. C. (1998). "Intelligent Computer-Aided Design," Revision of 1993 Article, Computer Science Department, Worcester Polytechnic Institute.

[02] Brown, D. C., and Grecu, D. L. "An 'AI in Design' View of Design," in Preparation for Journal Submission.

[03] Chandrasekaran, B. (1990). "Design Problem Solving: A Task Analysis," *AI Magazine*, Vol. 11, No. 4, Winter, pp. 59-71.

[04] Hindi, K., and Lings, B. (1994). "Using Truth Maintenance Systems to Solve the Data Consistency Problem," Computer Science Department, University of Exeter.

[05] Jeong, Y. (2001). "Impacts of the Introduction of New Financial Products after the Foreign Exchange Crisis," Samsung Economic Research Institute.

[06] Maher, M. L., and Garza, A. (1997). "Case-Based Reasoning in Design," *IEEE Expert*, Vol. 12, No. 2, pp. 34-41.

[07] Papavasileiou, A. et al. (2002). "Application of the Artificial Intelligence Methods in CAD/CAM/CIM Systems," *Mechanization*, Sofia, Vol. 44-45, pp. 75-79.

[08] Totton, K. A. E., and Flavin, P. G. (2001). "An Overview of Computer Aided Decision Support," Prepared for Computer Aided Decision Support in Telecommunications.

[09] Vancza, J. (1999). "Artificial intelligence support in design: A survey," Computer and Automation Research Institute, Hungarian Academy of Sciences.

[10] CA, http://www3.ca.com/Solutions/Product.asp?ID=250

[11] Syscorp, http://www.syscorp.com/Product_Factory.Htm