맞춤형 금융상품 설계시스템의 개발

최성철*, 이성하**, 주정은***, 구상회****

Development of a Financial Product Factory System

Seong-cheol Choi, Seong-ha Lee, Jung-eun Ju, Sang-hoe Koo

Abstract

맞춤형 금융상품 설계시스템(Financial Product Factory System)이란 온라인으로 접근하는 고객의 요구사항을 고려하여 고객에게 가장 적합한 금융상품을 실시간으로 설계하여 제공하는 시스템이다. 최근 들어 인터넷 뱅킹 고객의 수가 급증함에 따라 맞춤형 금융상품 설계시스템의 필요성이 대두되고 있으나, 이러한 시스템의 정의나 성격, 필요 기능, 구축 방안에 대한 연구가 되어 있지 않은 실정이다. 본 연구에서는 맞춤형 금융상품 설계시스템의 정의를 내리고, 이 시스템이 갖추어야 할 요구사항을 시스템과 서비스 측면에서 분석한 후, 이 요구사항을 반영하는 시스템의 아키텍처를 제안·구현한다.

키워드 : Product Factory, Artificial Intelligence in Design,
Rule-based Systems, Case-based Reasoning,
Constraint Satisfaction, Truth Maintenance Systems

* 고려대학교 일반대학원 디지털경영학과 석사과정, laugh2r@korea.ac.kr
** 고려대학교 일반대학원 디지털경영학과 석사과정, lilyan@korea.ac.kr
*** 고려대학교 일반대학원 디지털경영학과 박사과정, kquilt@korea.ac.kr
**** 고려대학교 일반대학원 디지털경영학과 교수, skoo@korea.ac.kr
※본 연구는 고려대학교 특별연구비(#K0202900)의 지원으로 이루어졌음.

1. 서론

인터넷과 정보통신기술의 발달은 전 산업 분야의 경제활동 방식에 지대한 영향을 미치고 있다. 금융업 역시 이러한 영향으로부터 자유로울 수 없다. 전자화폐, 스마트카드와 같은 새로운 서비스가 등장하고 많은 금융업무가 전자화·자동화되어 가고 있다. 이에 따라 창구에서 고객을 상대로 직접 업무를 처리하는 직원의 숫자는 점점 줄어들고, 대신 정보기술을 다루는 직원의 숫자는 점차 늘어나고 있다. 고객과의 접점이 창구에서 인터넷으로 변화함에 따라 은행은 하루 24시간 장소에 구애받지 않고 지속적인 서비스를 제공할 수 있게 되었으며, 대부분의 금융업무를 전자화하여 수행할 수 있게 되었다. 또한 오프라인에서는 상당한 비용이 수반되었던 일대일 마케팅(One-to-One Marketing)이 인터넷 뱅킹을 통하여 보편화되고 있다[Booz, Allen & Hamilton].

인터넷 기술의 향상과 인터넷 사용자의 증가는 비금융기업의 은행업 진출을 가속화하며, 온·오프라인을 막론하고 계속해서 크고 작은 금융기관을 모방하는 인터넷 업체를 출현시켰다. 이러한 변화는 자연스럽게 금융산업의 경쟁심화로 이어지고 있다. 한편 인터넷의 보급은 금융서비스의 대상인 고객에게도 매우 큰 변화를 주고 있다. 그 중 하나가 고객 니즈의 다양화이다[최행진, 2001]. 즉 연령, 직업, 사회계층, 생활양식에 따라 금융상품에 대한 니즈가 매우 다양하게 나타난다. 이렇게 심화된 경쟁구조 하에서 금융기관은 고객의 다양한 요구에 능동적으로 재빠르게 대처하여 고객의 요구를 만족시켜야만 생존하고 경쟁력을 유지할 수 있게 되었다.

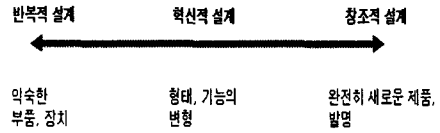
최근 조사에 따르면[정영식, 2001], 우리나라의 금융상품 종류는 선진국 수준인데 반해 상품의 완결성은 그에 훨씬 미치지 못한다. 외환위기 이후 우리나라는 금융 개방화와 제도 선진화에 따라 다양한 금융상품을 새로이 개발하였지만, 이를 제대로 운용할 수 있는 제도나 시스템은 갖추지 못하고 있다. 실제로 금융기관에서는 아직까지 금융상품을 추천하거나 설계하는 과정의 대부분을 사람이 직접 하고 있으며, 그나마 설계 과정이 고정적이고 유연성이 결여되어 있어 개별 고객을 위한 맞춤형 상품을 설계하고 제공하기가 매우 힘든 실정이다. 이러한 문제점을 해결하기 위한 방안으로 맞춤형 상품 설계시스템(Product Factory)이 선진 금융기관에서 논의되고 있다[Syscorp]. 맞춤형 상품 설계시스템이란 인터넷을 통하여 접근하는 고객의 요구사항을 온라인으로 분석하여 고객에게 가장 적합한 금융상품을 실시간으로 설계·제공할 수 있는 자동화된 금융상품 설계시스템이다. 현재 이러한 시스템의 필요성은 인식하고 있지만, 실제로 맞춤형 금융상품 설계시스템의 정의, 시스템의 요구사항, 구현 방안 등에 대한 구체적인 연구는 거의 이루어지지 않은 실정이다.

본 연구에서는 맞춤형 상품 설계시스템의 개념을 정의하고, 이 시스템이 갖추어야할 요구사항을 서비스와 시스템 측면에서 분석하며, 이들 요구사항을 만족시키는 아키텍처를 제안한다. 또한 제안된 아키텍처의 유효성을 입증하기 위해 간단한 프로토타입 시스템을 구현하여 이 시스템을 금융상품 중 대출상품에 적용한 사례를 보인다.

2. 지능형 설계 (AID: AI in Design)

설계(design)란 주어진 목적을 구현하기 위한 조형물(artifacts)의 구성요소를 합리적으로 선택하고 선택된 요소를 올바르게 조합하는 방법을 계획하는 과정이다[Caplan, 1984]. 설계 문제(design problem)는 만들어질 조형물이 갖추어야 할 일련의 기능들과 그 기능을 수행하기 위하여 만족되어야 할 제약조건(constraints)으로 명세되어진다[Brown & Grecu]. 그리고 설계안(design solution)은 설계 문제를 해결하는 솔루션으로서 목표 조형물을 나타내는 모든 구성요소들과 요소들 간의 관계(structural relationships)에 대한 명세로 구성된다. 설계 과정(design process)은 설계 문제의 명세로부터 설계안을 찾는 과정으로서, 수없이 많은 복잡한 문제들을 수반하며 이들 문제의 해결 방법을 찾아야 하는 복잡한 과정이다. 이를 위해서 설계 과정은 새로운 지식을 필요로 하거나 창의적인 시도를 수반하기도 한다. 이처럼 설계 과정은 복잡한 사고를 필요로 하는 비구조적인 행위(ill-structured activity)이기 때문에[Rodgers, 1998], 인공지능(AI: Artificial Intelligence) 분야에서 설계 문제를 해결하기 위한 많은 연구가 진행되고 있다[Vancza, 1999]. 지능형 설계(AID: AI in Design)란 인공지능 기법을 적용하여 특정 영역에서 발생되는 설계 문제를 해결하기 위한 기술로서[Brown & Grecu], 설계안을 찾기 위해 탐색이나 제약만족기법 등이 활용되고 있으며[Vancza, 1999], 설계 과정의 학습을 위해 기계학습 기술이 활용되며[Papavasileiou et al., 2002], 설계 협업을 위해 다중에이전트 시스템[Vancza, 1999]이나 사례기반추론[Maher & Garza, 1997] 등이 활용되고 있다.

설계 과정은 과정의 복잡도에 따라 반복적 설계(Routine Design), 혁신적 설계(Innovative Design), 창조적 설계(Creative Design)의 세 가지 유형으로 나눌 수 있다((그림 1))[Brown, 1998]. 반복적 설계는 지식이라든가 문제해결 전략을 미리 알고 있는 일상적이며 반복적으로 수행되는 설계를 말하고, 혁신적 설계는 지식만 어느 정도 알고 있으며 설계 과정에 대한 전략적 지식은 가지고 있지 않은 상태에서 수행하는 설계를 말한다. 창조적 설계는 지식과 전략에 대한 정보가 전혀 없는 상황에서 행해지는 설계이다. 현재까지의 지능형 설계에 관한 연구를 살펴보면 주로 반복적 설계에 집중되어 있으며[Rodgers, 1998], 혁신적 설계에 관한 연구도 부분적으로 수행되고 있다[Chandrasekaran, 1990].



(그림 3) 설계 유형
[Rodgers, 1998]

본 연구의 목적은 맞춤형 금융상품 설계시스템의 개발에 있다. 맞춤형 상품 설계란 온라인 실시간으로 고객의 니즈를 분석하여 고객의 니즈에 적절한 상품을 제공하는 것이다. 이때 고객의 니즈에 적합한 상품이 이미 존재하는 경우에는 이 상품을 고객에게 제시하지만, 만약 존재하지 않으면 새로운 상품을 설계하여 고객에게 제시함으로써 고객이

이를 선택할 수 있도록 한다. 금융상품은 대출 종류, 대출금, 이자 등과 같은 기호적(symbolic) 속성과 수치적(numeric) 속성에 의하여 정의되며, 이들 속성 간의 관계성(relationships)은 존재하지만(예를 들면, 대출 종류에 따른 대출금의 한도), 물리적 조형물에서 볼 수 있는 구조적인 제약조건(structural constraints)은 존재하지 않는다. 따라서 금융상품의 설계 문제와 설계안 모두 이들 속성과 속성 간의 관계로 기술되며, 설계 과정은 이들 관계성을 만족시키는 속성의 조합을 찾는 문제로 단순화된다. 금융상품 설계 문제는 여러 가지 AID 기술 중에서 사례기반추론이나 제약만족기법이 적절히 사용될 수 있다. 또한 금융상품을 설계하는 과정은 금융상품을 정의하는데 필요한 다양한 속성값을 채우는 과정으로, 고객의 요청이 있을 때마다 반복적으로 발생한다. 그러므로 본 설계 문제는 혁신적이거나 창조적인 설계 문제보다는 반복적 설계 문제에 가깝다.

3.맞춤형 금융상품 설계시스템 (Product Factory)

3.1.맞춤형 금융상품 설계시스템의 정의

본 연구의 목적은 앞서 말한 바와 같이 맞춤형 금융상품 설계시스템의 개발에 있다. 금융상품 설계시스템이 개발되기 위해서는 먼저 구조화된 금융상품의 표현 방법이 개발되어야 한다. 본 연구에서는 금융상품(financial product)을 속성의 집합으로 표현한다.

$$Product = \{a_1, a_2, a_3, \cdots, a_n\}$$

예를 들어, 금융기관에서 취급하는 상품 중 '종합통장'은 예금평잔, 채권보전, 대출한도, 대출기간, 대출방법, 상환방법, 대출금리 등의 속성을 갖는다. 이들 속성이 갖는 값은 기호와 수치로 표현될 수 있다. 또한 이들 속성에는 여러 가지 제약조건이 수반될 수 있다.

종합통장 = { 예금평잔(B), 채권보전(S), 대출한도(L), 대출기간(T), 대출방법(M), 상환방법(R), 대출금리(I) }

$B \geq 0$, Unit: Won
S = {"담보","연대보증","신용"}
$L \leq 10{,}000{,}000$, Unit: Won
$1 \leq T \leq 12$, Unit: Month
M = "at any time"
R = "at any time"
$10.0 \leq I \leq 14.0$, Unit: %

맞춤형 금융상품 설계 과정(design process)은 고객의 조건과 니즈를 제약조건으로 전제하고 이를 만족하는 상품을 설계하여 제시하는 과정으로 볼 수 있다. 위 상품을 표현하는 방법을 이용하여 맞춤형 금융상품 설계 과정을 정의하면 다음과 같다. $C_1,\cdots,C_n$은 고객의 조건이나 니즈 등을 나타낸다.

$$Find\ p = \{a_1, a_2, a_3, \cdots, a_n\},$$
$$Given\ \{C_1, \cdots, C_n\}$$

본 연구에서 개발하고자 하는 맞춤형 금융상품 설계시스템(product factory)은 위 설계

과정을 자동화하여 인터넷 뱅킹 고객에게 그의 니즈에 부합되는 상품을 온라인 실시간으로 설계하여 주는 시스템이다.

3.2. 맞춤형 금융상품 설계시스템의 요구사항

본 연구에서는 앞 절에서 정의한 맞춤형 금융상품 설계시스템의 요구사항을 대 고객 서비스와 시스템의 기능이라는 측면에서 고려하여 다음과 같이 제시한다.

□ 실시간 설계(real-time design): 본 시스템을 이용하기 위하여 고객은 시스템에 접속하여 갖가지 자신의 금융정보와 니즈를 입력한다. 그러면 시스템은 실시간으로 고객의 니즈를 분석하여 적절한 상품 설계를 수행하여야 한다.

□ 의사결정 지원(decision support): 또한 고객이 여러 가지 조건에 변화를 주면서 제시되는 다양한 상품을 상호비교 분석하는 과정에서, 시스템은 고객이 가장 적합한 상품을 선택할 수 있도록 지원해야 한다.

□ 빠른 반응시간(quick response): 고객에게 실시간으로 상품을 설계하여 제시하기 위해서는 설계 프로세스와 설계에 필요한 지식을 체계적으로 관리하고, 설계 과정을 공학적으로 설계하여 시스템의 반응 시간을 최소화할 필요가 있다.

□ 상품일관성(product consistency): 설계될 상품이 개별 고객의 니즈를 만족시키면서 동시에 기존 상품의 설계와 크게 다르지 않도록 상품 간의 일관성을 유지하여야 한다.

3.3 맞춤형 금융상품 설계시스템의 아키텍처

본 연구에서는 앞 절에서 제시한 맞춤형 금융상품 설계시스템의 요구사항을 고려하여 (그림 2)와 같은 아키텍처를 제안한다. 제시된 아키텍처는 상품의 유형 분석, 최적상품의 추천, 최적상품에 대한 설계, 고객의 의사결정 지원의 네 가지 모듈로 구성된다(그림에서 사각형으로 도식화된 부분). 본 시스템의 서비스 시나리오를 이 아키텍처와 관련하여 설명하면 다음과 같다.

□ 먼저 고객은 본 시스템의 사용자 인터페이스를 통하여 자신의 거래실적 및 거래기간, 신용도, 대출 목적, 필요 금액 등 상품 설계에 필요한 데이터를 입력한다.

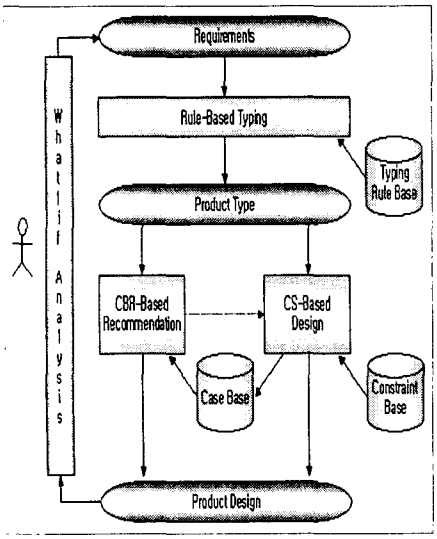
□ 최적상품 유형분석 모듈(Rule- based Typing Module)은 고객이 입력한 내용에 따라 '학자금 대출', '주택마련 대출', '생활자금 대출' 등 적절한 상품 유형을 결정한다.

□ 사례기반 추천 모듈(CBR-based Recommendation Module)은 앞 모듈에서 결정된 상품유형의 사례를 모아 놓은 사례베이스를 검색하여 고객에게 가장 적합한 상품을 검색하여 제시한다.

□ 만약 앞 단계에서 추천할 사례를 검색하는 데 실패할 경우, 제약만족기반설계 모듈(CS-based Design Module)이 고객이 입력

한 사항, ②단계에서 결정된 상품유형, 그리고 금융상품에 관한 일반적인 제약조건을 동시에 고려하여 고객에게 적합한 새로운 상품을 설계하여 제시한다.

⑤ 고객은 ③ 또는 ④단계에서 제시된 상품을 살펴보고 필요시 의사결정 지원 모듈(What-if Analysis Module)을 이용하여 여러 가지 조건에 변화를 주면서 제시되는 다양한 상품을 상호비교 분석함으로써 자신에게 가장 적합한 상품을 선택한다.

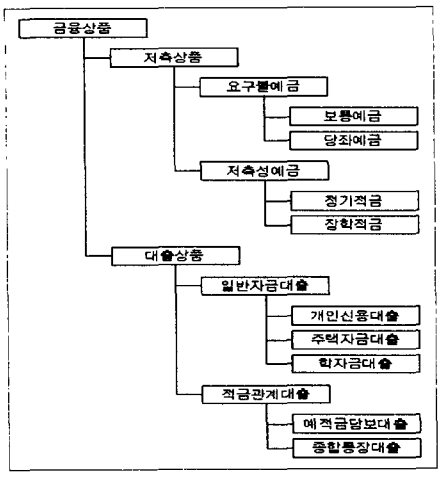


(그림 2) 금융상품
추천/설계시스템 아키텍처

위에서 설명된 아키텍처는 앞 절에서 설명한 요구사항을 만족시키도록 구성되었다. 아키텍처 상에서 사례베이스(case base)와 제약베이스(constraint base) 모두 다양한 금융상품에 대해 종류별로 세분화되어 있기 때문에, 먼저 유형분석 모듈에 의하여 유형이 결정되면 해당 유형과 관련된 사례와 제약조건만을 선택하여 실행함으로써 실시간 설계(real-time design)와 빠른 반응시간(quick response)에 도움이 되도록 하였다. 또한 사례기반추천 및 제약만족기반설계 과정이 실시간 설계와 빠른 반응시간을 보장할 수 있도록 구조화되어 개발되었다. 그리고 What-if 분석 모듈을 제공하여 고객이 올바른 의사결정을 할 수 있도록 설계하였다(decision support). 또한 상품의 설계시 사례베이스를 먼저 검색하게 함으로써, 새로운 상품의 설계를 가능한 한 피할 수 있도록 하였다(product consistency).

3.3.1 최적상품 유형분석 모듈(Rule-based Typing)

본 모듈은 룰베이스시스템(Rule based systems)으로서, 고객의 거래실적과 거래기간 등의 데이터를 분석하여 고객에게 해당되는 금융상품의 유형과 정보를 결정한다. 본 모듈에 사용되는 정보는 양적인 면에서 매우 많다. 따라서 이들 룰을 효율적으로 저장하고 색인할 수 있는 방법이 필요하다. (그림 3)은 금융기관에서 사용되는 내부지침을 근거로 분류해 놓은 금융상품을 트리로 나타낸 것이다. 금융기관 내부적으로는 상품을 4단계 수준의 트리로 구성하지만, 실제로 고객에게 금융상품을 전달할 때에는 2단계에서 4단계로 바로 보여주는 것이 일반적이다. 본 연구에서는 이 트리를 기준으로 룰을 계층적으로 구조화하여 저장 관리한다.

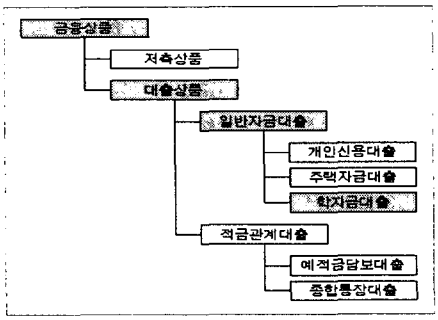


(그림 3) 금융상품 유형 분석 트리

본 모듈의 상품유형 결정 과정을 사례를 통하여 살펴본다. A는 27세의 대학원생으로 등록금 마련을 목적으로 대출을 신청한다고 가정하자. A는 학생 신분이기 때문에 담보를 설정하지 않고, 자신의 신용만으로 대출상품을 이용할 수 있다. A의 금융정보를 요약하면 다음과 같다.

나이 : 27세
직업 : 대학원생
거래기간 : 5년(60개월)
예금평잔 : 100,000원
신용도 : 보통
채권보전 : 신용
담보물 : 없음

A의 정보를 시스템에 입력하면, 본 모듈은 (그림 4)에서 보는 바와 같이 [금융상품]-[대출상품]-[일반자금대출]의 단계를 거쳐 최종적으로 [학자금대출]이라는 항목으로 상품의 유형을 결정한다.



(그림 4) 금융상품 유형 분석 결과

3.3.2 최적상품 추천 모듈(Case- based Recommendation)

본 모듈은 사례베이스로부터 고객의 조건과 요구에 가장 유사한 상품 사례를 찾아 이를 추천한다. 가장 유사한 상품 사례를 검색하기 위해서는 유사도를 측정하기 위한 방안이 개발되어야 한다. 본 연구에서는 유사도를 상품의 속성값에 기초하여 산출한다. 상품이 갖는 속성은 몇 가지 측면에서 분류할 수 있다. 먼저 속성은 조건적 속성과 결과적 속성으로 나눌 수 있는데, 직업, 담보, 신용도 등은 조건적 속성에 속하며, 이들 속성 값에 따라 결정되는 이자율, 대출한도 등은 결과적 속성에 속한다. 본 연구에서는 조건적 속성을 근거로 유사도를 측정한다. 또한 속성은 속성이 취하는 값에 따라 수치 속성(numeric valued attribute)과 기호 속성(symbolic valued attribute)으로 나눌 수 있다. 직업이나 담보종류 등은 기호 속성에 속하며 이자율, 대출금액 등은 수치 속성에 속한다. 속성이 수치값을 갖는냐 기호값을 갖느냐에 따라 유사도를 측정하는 방식이 달라져야 한다. 본 연구에서는 다음과 같은 방식으로 개별 속성 사이의 유사도를 측정한다.

① 수치속성(numeric valued attribute)

수치 속성의 유사도는 아래 식에 의하여 계산한다. 아래 식에서 *a*와 *b*는 두 상품의 동일 속성(예를 들면, 대출금)에 대한 값이며, *sim(a, b)*는 두 속성 값 사이의 유사도이다. max *difference*는 해당 속성값이 가질 수 있는 가장 큰 차이를 의미하며, |*a*-*b*|는 *a*와 *b* 차이의 절대값이다. *sim(a, b)*는 0과 1사이의 실수값을 가진다. 예를 들어 보자. 생활자금 대출의 경우 대출금액이 100만원부터 1000만원 사이이고, 두 상품의 대출 자금이 각각 500만원과 700만원이라 가정하자. 그러면 *a* = 5,000,000, *b* = 7,000,0000, max *difference* = 10,000,000 - 1,000,000 = 9,000,000, |*a*-*b*| = |5,000,000 - 7,000,000| = 2,000,000이다. 따라서 *sim(a, b)* = 1 - (2,000,000 / 9,000,000) = 0.78이다.

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

② 기호속성(symbolic valued attribute)

본 연구에서는 기호 속성을 '예/아니오'나 '있다/없다' 등과 같은 바이너리(Binary) 타입과 세 가지 이상의 값을 가지는 노미널(Nominal) 타입으로 나눈다. 바이너리 타입의 경우에는 두 값이 같으면 *sim(a, b)*는 1이, 그렇지 않으면 *sim(a, b)*는 0이 된다. 노미널 타입은 두 속성의 교집합의 개수를 합집합의 개수로 나누어 유사도를 계산한다. 이 경우에 *sim(a, b)*는 0과 1사이의 실수 값을 갖는다.

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

$$sim(a, b) = \frac{size(a \cap b)}{size(a \cup b)}$$

위에서 설명한 유사도는 개별 속성에 대한 유사도이다. 사례의 유사도를 측정할 때 사용되는 것은 이러한 개별 속성의 유사도가 아니라 상품의 유사도이다. 상품이란 개별 속성의 집합으로 정의된다. 따라서 이들 개별 유사도를 종합할 수 있는 방식이 필요하다. 본 연구에서는 개별 속성의 유사도를 지역유사도(local similarity) 그리고 상품의 유사도를 전역유사도(global similarity)라 정의하고, 전역유사도는 지역유사도의 가중평균(weighted sum)으로 계산한다. 지역유사도의 가중치는 각 속성의 유사도 측정에 관한 중요도를 의미한다. 아래 식은 전역유사도를 측정하는 식이다. 사례를 추론하는 과정에서 특정한 속성에 [0, 1] 사이의 실수로 된 가중치를 설정한다.

$$SIM(A, B) = \frac{\sum_{i=1}^{p} w_i sim_i(a_i, b_i)}{\sum_{i=1}^{p} w_{i=1}}$$

사례 검색 과정에서 고객이 입력한 데이터와 정확하게 일치하는 사례가 존재할 경우에는 해당 사례를 적용하고, 그렇지 않을 경우에는 유사도가 가장 높은 사례를 찾아 그 사례가 임계치(threshold)보다 높을 경우에 추천 사례로 적용한다. 본 연구에서는 유사도가 동일한 상품이 검색되는 경우에 과거 추천 횟수가 가장 많은 사례를 선택하여 추천한다.

A의 정보를 분석한 결과, A는 '학자금대출'

이라는 상품 유형에 속한다. 시스템은 '학자금대출' 상품과 관련된 상품 사례들과 A의 사례와의 속성(조건적 속성)들 간의 유사도(지역유사도)를 계산한 다음, 각 속성의 중요도에 따라 가중치를 설정하고 이를 합산한 후 사례 간 유사도(전역유사도)를 측정하여 기준 임계치를 넘어서는 사례들 중 유사도가 가장 높은 사례를 A에게 추천한다.

3.3.3 최적상품에 대한 설계 모듈

본 모듈은 사례에 기반한 최적상품의 추천이 실패할 경우, 제약만족(Constraint Satisfaction) 기법으로 고객의 제약조건을 고려하여 개별 고객을 위한 새로운 금융상품을 설계·제공한다. 본 연구에서는 금융상품을 설계하는 과정에서 발생하는 제약조건을 속성 자체의 제약조건(intra-attribute constraints)과 속성 간의 제약조건(inter-attribute constraints)으로 나누어 이들을 룰 형태로 표현한다. 또한 제약조건을 만족하는 설계안을 찾는 과정은 후방향추론(backward chaining) 방식의 룰베이스시스템을 이용하여 구현한다. 아래는 이들 조건의 표현 방법과 설계 과정을 구현한 룰을 나타낸다.

① 속성 자체의 제약조건(intra- attribute constraints)

아래는 속성 자체의 제약조건을 예시한 것이다. 아래 예는 bankingTerm(거래기간) 속성의 제약조건이 만족되는지를 판단하여 만족되면, bankingTermGrade (거래기간등급) 속성의 값을 설정하도록 되어 있다. 본 연구에서 상품의 설계안은 속성값의 집합으로 정의되기 때문에, 속성값을 설정하는 과정을 곧 설계 과정으로 볼 수 있다.

```
Rule bankingTerm
IfRule bankingTerm >= 3 And
  bankingTerm < 12
Then
  bankingTermGrade = "E"
End
```

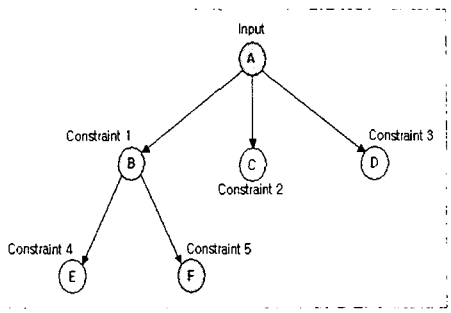
② 속성 간의 제약조건(inter-attribute constraiints)

아래 예는 속성 간의 제약조건이 만족되면, 특정 속성의 값을 설정하고 아울러 상품을 추천하는 코드를 구현한 부분이다. 이 룰은 앞서 설명한 룰과 마찬가지로 속성값들이 설정되는 상품 설계 과정의 일부를 구현한 것이다. 아래 예는 전체적으로 금융상품을 추천할지 결정하는 룰이다. loanAmount(대출금액)의 값이 loanLimit(대출한도)의 값을 초과하는지 판단하여 만족하는 경우에는 approval Status(승인상태)의 플래그(flag)를 "is_approved(승인)"로 할당한다. approval Status의 값에 대한 후방향추론을 수행하여 값이 제대로 들어있는지 확인한다. approvalStatus의 값이 "is_approved"이면 대출상품을 추천한다.

```
Rule loanAmount
IfRule loanAmount <= loanLimit
Then
  approvalStatus = "is_approved"
End
Infer
  BackwardChain (->approvalStatus)
End
```

```
If approvalStatus = "is_approved"
Then
  recommendLoanProduct
End
```

제약조건의 룰을 포스트(post)할 때에는 모든 룰에 대하여 추론하지 않고 (그림 5)에서 나타난 바와 같이 트리의 노드(A, B, F)에 포함된 제약조건만 가지고 추론을 실행한다. (그림 5)에서 각 노드는 제약조건을 의미한다. 노드 A에서 입력한 값을 기준으로 제약조건을 만족하면 노드 C와 D는 거치지 않고 바로 노드 B로 이동하며, 노드 B에서의 제약조건이 만족하면 노드 F로 이동한다. 본 연구에서는 이렇게 제약조건을 구조적으로 관리함으로써 추론시간과 상품 설계시간을 줄일 수가 있었다.



(그림 5) 제약조건 적용 과정

3.3.4 고객의 의사결정 지원 모듈

What-if 분석은 모델에 포함된 요소들의 변화에 대응되는 결과를 살펴봄으로써 '만약 ~라면'이라는 질문에 대답하기 위하여 사용한다[Totton & Flavin, 2001]. 본 연구에서는 What-if 분석 기능을 제공하여 고객이 여러 조건에 변화를 주면서 추천되는 상품을 살펴봄으로써, 자신에게 가장 적절한 금융상품을 선택할 수 있도록 하였다. 고객은 조정이 가능한 속성인 예금평잔, 신용도 등의 거래정보를 변경시키면서 추천되는 상품의 설계안을 비교 분석할 수 있다. 기존 룰베이스시스템을 이용한다면 What-if 분석을 수행하기 위해서 수없이 많은 clause를 assert하고 retract해야 하며, 이에 따라 추론에 소요되는 자원의 양이 기하급수적으로 커질 수 있다. 본 연구에서는 TMS (Truth Maintenance Systems)기술을 활용함으로써, clause의 assert와 retract 대신 truth값을 조정 관리함으로써 추론의 속도를 빠르게 유지하였다[Hindi & Lings, 1994]. 아래 룰은 본 연구에서 사용된 TMS의 표현방식이다. 일반적인 프로그램에서는 변수에 값을 할당할 때 'bankingTerm = tempInteger'와 같이 표현하지만, (그림 9)에서 보는 바와 같이 Aion에서는 'bankingTerm ?= tempInteger'로 표현함으로써 향후에 'bankingTerm' 변수의 값이 일시적으로 바뀌더라도 최초에 할당된 값이 유지된다.

```
If Decode(tempInteger,inputValue)
Then
  bankingTerm ?= tempInteger
End
```

4. 시스템 구현 및 실행 사례

본 연구에서는 앞 장에서 설계한 내용에 따라 맞춤형 금융상품 설계시스템의 프로토타입을 구현하였다. 시스템은 클라이언트/서버 환경으로 구축되었으며, 운영체제와 데이터

베이스는 MS (Microsoft)사의 Windows 2000 Server와 Access 2000을 이용하였고, 개발도구로서 CA(Computer Associates)사의 Aion(CleverPath Aion Business Rules Expert 9.1)을 이용하였다. 본 장에서는 개발된 프로토타입과 함께 대출상품의 설계 과정을 사례로서 제시한다.

(그림 6)은 금융상품의 유형을 분석하는 룰이다. 유형분석 룰은 Aion의 유틸리티 중 하나인 의사결정 테이블(decision table)을 이용함으로써 모든 룰을 일일이 만들지 않고서도 쉽게 룰의 일관성, 완전성 등을 유지하면서 생성 관리할 수 있도록 하였다. 상품의 유형 결정 룰은 대출 목적과 채권보전, 담보물, 직업 등의 속성이 룰의 조건이 되며, 상품의 유형, 기준 대출한도, 대출금리 등이 결과를 구성한다.

	"담보"	"부동산"	"회사원"	
			"공무원"	
			"전문직"	
			"대학(원)생"	
			"기타"	
"학자금 마련"	"신용"	"없음"	"회사원"	X
			"공무원"	X
			"전문직"	X
			"대학(원)생"	
			"기타"	X
	"담보"	"부동산"	"회사원"	
			"공무원"	
			"전문직"	
			"대학(원)생"	
			"기타"	

(그림 6) 금융상품 유형 분석 룰

(그림 7)은 고객이 입력한 속성과 사례베이스에 저장되어 있는 상품의 속성 간의 유사도를 계산하여 계산된 유사도에 따라 상품을 추천하는 기능을 구현한 것이다. 본 연구에서는 이 기능을 Aion의 룰을 이용하여 구현하였으며, 이 룰 안에서 함수를 호출하여 전역유사도를 계산하도록 되어 있다. 그림에 나타난 바와 같이 상품의 전역유사도는 나이, 연소득 금액, 근무연수, 예금평잔, 정기예적금 금액, 급여(연금) 이체액, 신용카드 이용액, 신용도 등의 속성을 기초로 하여 계산되며, 전역유사도가 기준 임계치인 0.8을 넘을 경우에 해당 상품을 추천하도록 되어 있다. 여기서 기준 임계치와 전역유사도 측정을 위한 각 속성의 가중치는 금융전문가와 상의하여 결정하였다.

```
Bind pProductFactory To ProductFactory
Bind pLoanProduct To LoanProduct

Rule searchLoanProduct
Ifmatch pProductFactory,pLoanProduct
Where
    pProductFactory.getProductType    = pLoanProduct.prod
    pProductFactory.getSecurityType   = pLoanProduct.secu
    pProductFactory.getSecurity       = pLoanProduct.secu
Then
    pLoanProduct.Chosen = True
    pProductFactory.loan = pLoanProduct

    ageMatchRate                  = numericMatch(pProduc
    annualIncomeMatchRate         = numericMatch(pProduc
    yearsInBusinessMatchRate      = numericMatch(pProduc
    balanceMatchRate              = numericMatch(pProduc
    totalSavingsDepositsMatchRate = numericMatch(pProduc
    totalPayNPensionMatchRate     = numericMatch(pProduc
    totalCreditcardMatchRate      = numericMatch(pProduc
    bankingTermMatchRate          = numericMatch(pProduc
    creditRatingMatchRate         = symbolicMatch(pProdu

    sim = (ageMatchRate + annualIncomeMatchRate + yearsI

    If sim >= pProductRecommendation.threshold And sim >
        pProductRecommendation.recommendStatus   = True
        pProductRecommendation.maxSim            = sim
```

(그림 7) 금융상품 추천 룰

(그림 8)은 최적상품을 설계하는 과정에서 속성 간 제약조건을 적용하여 대출금리를 계산하는 룰이다. (그림 8)의 룰은 'loanAmount <= loanLimit'이 참일 경우 실행되며, 속성마다 가지고 있는 제약조건에 따라서 대출금리의 계산 방식이 달라지는 것을 알 수 있다.

```
Rule loanAmount
IfRule loanAmount <= 5000000
Then
    InterestRate = pProductFactory.getInterestRateCriteria + (pProductI
End

Rule loanMethod
IfRule loanMethod = "마이너스대출"
Then
    InterestRate = pProductFactory.getInterestRateCriteria + (pProductI
End

Rule creditRating
IfRule creditRating = "좋은 편임"
Then
    InterestRate = pProductFactory.getInterestRateCriteria - (pProductI
End

Rule creditRating
IfRule creditRating = "보통"
Then
    InterestRate = pProductFactory.getInterestRateCriteria - (pProductI
End

Rule securityType
IfRule securityType = "담보"
Then
    If security = "예적금" Then
```

(그림 8) 금융상품 설계 룰

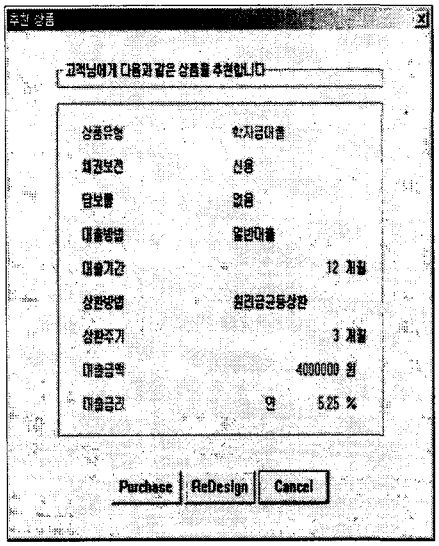
(그림 9)는 고객의 데이터 중 What-if 분석에 사용될 속성에 대하여 Aion의 TMS(Truth Maintenance System) 기능을 적용한 것이다. (그림 9)에서 상위 그림은 What-if 분석을 위한 변수를 정의하는 부분이고, 하위그림은 정의된 변수를 이용하여 TMS 기능을 적용한 것이다.

```
If IsKnown(->name) Then
    txtName.setText(name)
End

If IsKnown(->age) Then
    txtAge.setText(Format(age))
End

If IsKnown(->yearsInBusiness) Then
    txtYearsInBusiness.setText(Format(yearsInBusiness))
End

If IsKnown(->job) Then
    cmbJob.setText(job)
End

If IsKnown(->loanPurpose) Then
    cmbLoanPurpose.setText(loanPurpose)
End

If IsKnown(->securityType) Then
    cmbSecurityType.setText(securityType)
End

If IsKnown(->security) Then
    cmbSecurity.setText(security)
End
```

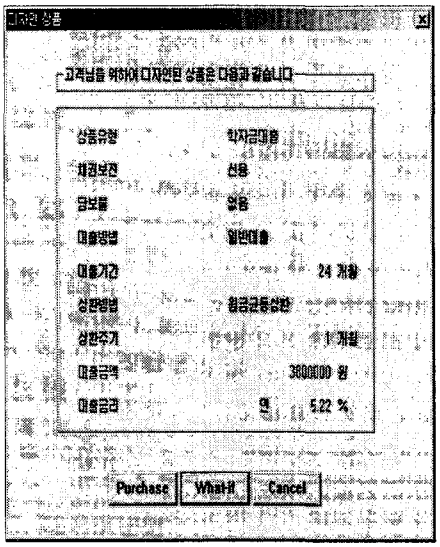
```
If Decode(tempString,txtName.Text) Then
    name ?= tempString
End

If Decode(tempInteger,txtAge.Text) Then
    age ?= tempInteger
End

If Decode(tempInteger,txtYearsInBusiness.Text) Then
    yearsInBusiness ?= tempInteger
End

If Decode(tempString,cmbJob.Selection) Then
    job ?= tempString
End

If Decode(tempString,cmbLoanPurpose.Selection) Then
    loanPurpose ?= tempString
End

If Decode(tempString,cmbSecurityType.Selection) Then
    securityType ?= tempString
End

If Decode(tempString,cmbSecurity.Selection) Then
    security ?= tempString
End
```

(그림 9) What-if 분석을 위한 룰

(그림 10)은 고객이 입력한 고객정보, 거래정보, 대출정보를 기반으로 사례베이스에 저장된 금융상품 사례 중 가장 유사한 것을 검색하여 '학자금대출'이라는 상품이 추천된 화면이다.

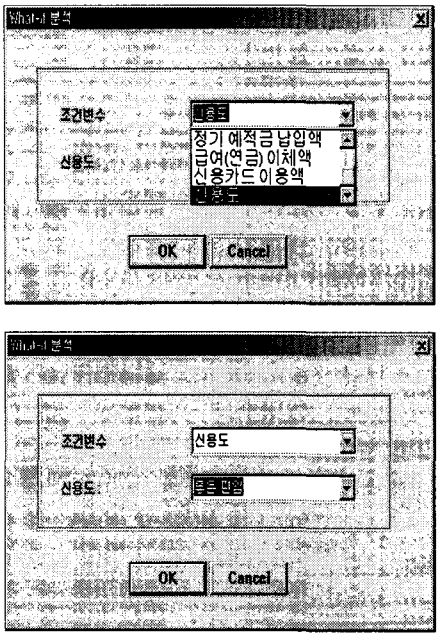


(그림 10) 추천된 금융상품 화면



(그림 11) 설계된 금융상품 화면

(그림 11)은 유사 사례를 찾지 못한 경우, 고객의 데이터를 분석하여 금융상품을 새롭게 설계 한 예이다. 이 단계에서 상품이 마음에 들면 'Purchase' 버튼을 눌러 상품을 바로 구매할 수 있도록 하였다. 'What-if' 버튼을 누르면 (그림 12)와 같은 분석 화면이 나타나, 고객이 임의로 속성값을 수정하면 대체상품을 확인할 수 있다. 이 그림에서 수정 가능한 속성은 예금평잔, 연소득 금액, 정기 예적금 납입액, 급여(연금) 이체액, 신용카드 이용액, 신용도와 같은 거래정보이다. 고객은 이 속성을 수정해 가면서 최초에 설계된 상품과 변경된 설계 상품을 비교할 수 있다.



(그림 12) What-if 분석 화면

5. 결론

본 연구에서는 고객의 유형에 따라 금융상품을 추천하거나 고객의 제약조건을 분석하여 가장 적합한 상품을 설계할 수 있는 금융상품 설계시스템을 제안하였다. 이 과정에서 맞춤형 금융상품 설계시스템의 개념에 대하여 정리하고 금융상품 설계시스템이 갖추어야 할 요구사항들을 도출하였으며, 이러한 요구사항을 반영한 시스템의 아키텍처를 제시하였다. 금융상품 설계시스템은 최적상품 유형분석, 최적상품 추천, 최적상품에 대한 설계, 고객의 의사결정 지원의 네 가지 모듈로 구성하였다. 최적상품 유형분석 모듈은 금융상품의 유형을 분석하기 위한 모듈로 RBS를 통해 고객의 거래실적과 거래기간 등의 데이터를 분석하여 고객에게 해당되는 금융상품의 유형과 정보를 제시한다. 최적상품 추천 모듈은 고객의 데이터와 금융상품의 유형을 종합적으로 분석하여 CBR을 기반으로 사례베이스로부터 고객의 조건과 요구에 가장 유사한 금융상품을 검색하여 추천한다. 최적상품에 대한 설계 모듈은 CS 기법으로 고객의 제약조건을 분석하여 개별 고객을 위한 새로운 금융상품을 설계·제공한다. 고객의 의사결정 지원 모듈은 TMS를 기반으로 고객의 올바른 상품구매 의사결정을 지원하여 고객이 여러 조건상의 변화를 주면서 What-if 분석을 통해 최적의 금융상품을 선택할 수 있도록 하였다.

본 연구에서의 금융상품 설계는 하나의 상품을 구성하고 있는 속성들의 값을 일정한 기준에 따라 결정하는 수준(value level design)에 머물고 있다. 이는 모든 상품이 같은 속성을 가지고 있고, 설계 과정에서 속성 값을 정하는 기준은 변하지 않는다는 전제에서 이루어졌다. 예를 들어, 이자율의 경우 동일한 상품이라고 하더라도 해마다 달라지는데, 본 연구에서는 이러한 상황을 고려하지 않았다. 따라서 상품을 추천하는 과정에서 과거의 사례에 포함되어 있는 속성들 중 이자율과 같이 과거의 기준과 현재의 기준이 바뀌는 속성이 발생할 경우에는 사실상 사례로서 추천하기가 어렵다. 또한 설계 과정에서 상품마다 갖는 속성이 달라질 경우에는 상품을 새로이 설계하는 것 자체가 불가능해진다. 향후에는 이러한 문제점들을 보완하여 계속적으로 변하는 기준에 대한 관리 방안과 함께 상품마다 각기 다른 속성을 구성할 수 있도록 속성 간의 연결 관계(attribute level design)에 대한 연구를 수행할 것이다.

참고문헌

[1] 정영식, 외환위기 이후 금융신상품의 도입과 영향, 삼성경제연구소, 2001.

[2] 최행진, 금융환경 변화에 따른 은행의 소매금융시장 세분화 전략에 관한 연구, 중앙대학교 석사학위논문, 2001.

[3] Brown, D. C., Intelligent Computer-Aided Design, WPI, 1998.

[4] Brown, D. C. & Grecu, D. L., An 'AI in Design' View of Design, WPI, in preparation for journal submission.

[5] Chandrasekaran, B., Design Problem Solving: A Task Analysis, AI Magazine, Winter, 1990, pp. 59-71.

[6] Hindi, K. & Lings, B., Using Truth Maintenance Systems to Solve the Data Consistency Problem, University

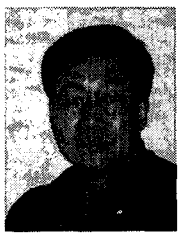
of Exeter, 1994.
[7] Maher, M. L. & Garza, A., "Case-Based Reasoning in Design", IEEE Expert, Vol. 12, No. 2, 1997, pp. 34-41.
[8] Papavasileiou, A. et al., "Application of the Artificial Intelligence Methods in CAD/CAM/CIM Systems", Mechanization, Sofia, Vol. 44-45, 2002, pp. 75-79.
[9] Rodgers, P. A., "The Role of Artificial Intelligence as 'Text' within Design", Design Studies, Vol. 19, No. 2, 1998, pp. 143-160.
[10] Totton, K. A. E. & Flavin, P. G., An Overview of Computer Aided Decision Support, 2001.
[11] Vancza, J., Artificial Intelligence Support in Design: A Survey, Hungarian Academy of Sciences, 1999.
[12] Caplan, R., *By Design*, McGraw-Hill Paperbacks, 1984.
[13] Booz, Allen & Hamilton, http://www.boozallen.com.
[14] Syscorp, http://www.syscorp.com/Product_Factory.Htm.

□ 저자소개

** 최성철
2003년 고려대학교 대학원 디지털경영학과졸업(석사)
2003년~현재: 한국컴퓨터 어소시에이트 연구원
Email:laugh2r@korea.ac.kr

*** 이성하
2003년 고려대학교 대학원 디지털경영학과졸업(석사)
연구분야: 정보검색, AI in Design
Email: lilyan@korea.ac.kr

**** 주정은
2002년 고려대학교 대학원 디지털경영학과졸업(석사)
2003년~현재: 고려대학교 대학원디지털경영학과 박사과정
연구분야: 정보검색, 지능형 에이전트, AI in Design, Business Rule Engines
Email: kquilt@korea.ac.kr



***** 구상회
1994 미국남가주대학 전산학 박사
1995~현재: 고려대학교 경영정보학과 교수
연구분야: 인공지능, 지능형 에이전트, Business Rule Engines
Email: skoo@korea.ac.kr

◈ 이 논문은 2003년 9월 30일 접수하여 2차 수정을 거쳐 2003년 12월 15일 게재확정되었습니다.