# CDSS를 위한 ArdenML XSLT 개발에 관한 연구

-VB.NET을 중심으로-

연세대학교 보건대학원

보건정보관리학과

김 강 수

# CDSS를 위한 ArdenML XSLT 개발에 관한 연구

-VB.NET을 중심으로-

지도 채 영 문 교수

이 논문을 보건학석사 학위논문으로 제출함

2007년 6월 일

연세대학교 보건대학원

보건정보관리학과

김 강 수

김강수의 보건학석사 학위논문을 인준함.

심사위원 _____ 인

심사위원 _____ 인

심사위원 _____ 인

연세대학교 보건대학원

2007년 6월 일

# 감사의 글

어느덧 시간이 흘러 벌써 2년 반이 지났습니다. 그리고 이렇게 감사의 글을 쓰고 있는 지금 지나간 시간들이 마치 영화를 보듯 생생히 기억납니다. 아름다운 영화를 만들어 주신 하나님께 감사드립니다.

바쁘신 일정 가운데에서도 부족한 저에게 열심을 아끼지 않으셨던 채영문교수님께 먼저 감사의 말씀을 드립니다. 또한 기술적인 부분을 지적하시며 방향을 제시해 주신 김석일교수님, 마지막까지 논문의 문장 하나하나까지 지적해 주시며 세심한 배려를 해주신 이병화교수님 감사드립니다.

늦은 시작에 용기를 북돋아 주시며 흔쾌히 공부를 승낙해 주신 조현정 회장님, 물심양면으로 지원해주신 전진옥사장님, 업무적인 부족함을 뒤에서 도와주신 조경호이사님 감사드립니다.

이제는 동기가 아니라 사회의 친구가 되어버린 이창희, 고종현, 윤수진, 이순근, 백혜성 선생님, 그리고 진달래조교, 선후배 선생님들, 특히 많은 조언을 해주신 이여진선생님 진심으로 감사드립니다.

효도하지 못했던 미안한 마음을 언제나 사랑으로 감싸주시며 걱정해주신 어머님, 아버님, 고생한다고 언제나 뒤에서 기도해주시는 큰형님,큰형수님, 애틋한 사랑으로 늘 챙겨주시는 작은형님, 작은형수님 감사드립니다.

멀리 떨어져 있어서 늘 찾아뵙지 못하는 미안한 마음을 격려로써 감싸주신 장인어른, 장모님, 처제, 처남 가족들 감사드립니다.

학업중에 사랑으로 다가와 내 인생의 커다란 힘이 되어주고, 든든한 후원자를 자처해준 사랑하는 아내 나영이, 그리고 뱃속에서 발로차며 아빠를 응원해준 사랑하는 내 아들에게 감사드립니다. 모두들 행복하세요!

2007년 6월

김 강 수 올림

# 목 차

# 표 목 차

# 그 림 목 차

# 국문요약

이 연구는 임상적 의사결정지원시스템(CDSS, Clinical Decision Support System)구현을 위한 ArdenML(Arden Markup Language)의 XSLT(eXtensible Style Language Transformation)를 개발하기 위해 시도하였다. Arden Syntax는 지식의 공유와 재사용의 장점때문에 HL7 표준 언어로서 채택되었지만 Curly Brace 문제와 컴파일러를 별도로 개발해야 하는 문제를 지니고 있다. Arden Syntax 컴파일러 개발에는 많은 노력이 필요할 뿐아니라 조직마다 자체 시스템에 맞게 임상규칙(MLM, Medical Logic Module)이 재 작성되어야 하고, 이때 마다 해석상의 오류를 발생시킬 수 있다.

Arden Syntax는 Category와 Slot, ArdenML은 Tag, Visual Basic은 Comment, Function, Events, Class, Procedure의 구조적 차이가 있었다. 데이터타입은 대부분의 형식이 동일하나 Time 및 Duration 같은 경우는 Visual Basic의 Date형식과 비슷했다. 연산자는 일반적인 컴퓨터 언어에서 지원하는 논리, 산술, 대입, 비트, 비교연산자를 지원하면서 의료정보의 특성상 자주 사용되고 많이 쓰이는 것들을 미리 정하여 하나의 연산자로 자동수행 될 수 있게 다양한 연산자가 정의되어 있었다. 제어문 및 지시문은 일반적인 컴퓨터언어에서 지원하지 않는 형태의 문장으로 특수한 목적을 가지고 정의되어 있는 지시문이 대부분이였다. 이러한 분석자료를 이용하여 XSLT를 개발하였다. 개발된 XSLT를 통하여 ArdenML를 Visual Basic으로 변환하였고, 기존시스템과 연계가능하도록 Interface 모듈을 개발하였다.

따라서 이 연구에서는 ArdenML을 분석하고 범용 컴퓨터언어(Visual

Basic, C#, Java 등)로 변환이 가능한 XSLT와 인터페이스를 개발하였으며 고혈압 임상규칙을 검증하였다. 대부분의 CDSS가 독립적으로 실행되어 중복입력을 통해서만 진단이 가능하지만, 이 시스템은 기존의 시스템과 단순히 변환 및 컴파일만을 통해 연계 가능함으로써 향후 규칙기반시스템(RBMS, Rule Base Management System)으로의 발전이 가능하다.

# I. 서론

## 1. 연구의 배경 및 필요성

최근 EMR이 성숙단계로 들어서면서 정보기술을 이용하여 의료서비스 질을 향상시키기 위한 CDSS의 구현이 주요 관심사가 되고있다. CDSS가 중요한 만큼 ASTM(1992년), HL7(1998년)에서 Arden Syntax를 의료분야에서 지식표현의 정식 표준언어로 채택하여 실용화하는 연구가 지속되고 있다.

Arden Syntax는 주어진 상황에서 임상적인 의사결정을 어떻게 할 지 표현해 주는 MLM을 작성하는데 사용하는 일종의 컴퓨터 언어이다. CDSS에 사용되는 언어로 GLIF, GELLO, SAGE, ASBRU 등이 더 있지만 Arden Syntex가 임상규칙의 형태면에서도 가장 실행 하기 가까운 형태를 갖고 있다고 알려져 있으며 GLIF(Guideline Interchange Format)에서도 최종로직은 Arden Syntax를 변형하여 사용하고 있다(Peleg, 2001).

즉, 임상현장에서 검증된 지식은 Arden Syntax로 표현되고 이를 이용한 MLM으로 작성되고 임상규칙을 작성하여 널리 배포하여 서로의 임상경험과 지식을 공유, 재사용, 진료시점에 현장에서 적용할 수 있는 CDSS을 만들게 된다(Jadhav, 2003).

외국에서 임상규칙을 전산화하여 환자진료시점에 직접 활용 하고자 하는 노력이 많이 이루어지고 있지만, 각 의료기관에서 여러 종류의 전자의무기록(EMR, Electronic Medical Record)을 사용함에 따라 임상규칙모델을 만드는 언어와 지식베이스가 달라지고, 설치하고자 하는 환경의 자료구조에 따라 실행엔진도 다르다는 점이 문제점으로 대두되고

있다(Karadimas, 2002). 전산화된 임상규칙은 그 안의 CDSS가 병원의 정보 시스템에 있는 자료를 이용할 수 있어야 정확한 결과를 제시해 줄 수 있기 때문이다.

Arden Syntax는 현재 버전 2.6까지 발전되었다. 그러나 현재의 수준은 임상규칙을 기술하기 위한 일정규격을 규정하고 있으나 컴파일러가 없어서 작성된 임상규칙에 대한 검증이 힘들고 기존의 시스템과 연계하여 실행가능 한 형태로 활용될 수 없다(Karadimas, 2002; Hripcsak, 1991).

컴파일러를 활용하는 기존의 방식은 Arden Syntax를 실행하기 위해 Java(Karadimas, 2002), C++(Kuhn, 1993; Gao, 1993), C++인터프티터(Hripcsak, 1991)등 각각의 시스템에 맞는 언어로 재개발 하여야 하는 문제를 만들었다.

따라서 다양한 시스템환경에 범용적으로 적용 가능한 컴파일러를 개발하기 위해서는 Arden Syntax의 XML(eXtensible Markup Language) 표현방식을 사용해야 한다. Arden Syntax로 구성된 구문을 XML Schema로 표현하고 시스템과 연계하여 실행가능한 형태로 활용될 수 있다(Sailors 2001).

ArdenML은 이러한 문제점들을 해결하기 위해 연구되고 있는 Arden Syntax의 XML버전으로 임상규칙의 작성과 시스템언어로의 변환이 용이한 방법이다. ArdenML은 XML의 Schema구조를 가지고 있어서 작성상의 오류나 문제를 즉시 발견하고 수정할 수 있으며 XSLT를 사용하여 작성된 MLM을 사용자의 변경없이 자동으로 컴퓨터언어로의 변환이 가능하다.

Arden Syntax는 HL7 표준언어로 개발되었지만 Curly Brace문제와 이종시스템간의 컴파일러의 다양성 문제로 인해 최근까지 사용이 제한되어 왔다.

서로 다른 시스템의 구조 때문에 동일한 MLM에 대해 시스템 특성에

따라 각기 다른 컴파일러가 개발되면 이들의 결과는 임상판단 과정상에 해석상의 오류가 발생하여 본래 목적인 지식의 공유와 재사용의 효과는 반감된다. 따라서 Arden Syntax는 컴퓨터나 의료시스템에서 전문가의 지식을 표현하고 공유할 수 있는 도구지만 EMR환경이 제공하는 지식베이스나 기술의 다양성 때문에 효과를 발휘하고 있지 못한 실정이다.

이 시스템언어와 구조의 다양성으로 인한 문제를 극복하기 위해 공통적으로 적용되는 ArdenML의 적용과 범용 프로그램으로 전환하기 위한 XSLT의 개발이 필요하다. XSLT로 변환된 임상규칙은 EMR, OCS, PACS와 연동되는 프로그램으로 HIS(Hospital Information System)에 통합됨으로써 전문가들의 지식표현과 의사결정 정확도와 접근성을 높일 수 있다.

따라서 이 연구의 궁극적인 목적은 ArdenML 2.6을 기반으로 이를 활용하는 XSLT를 개발하여 검증함으로써 CDSS실용화의 전환점을 제공하는 것이다.

## 2. 연구목적

이 연구는 ArdenML 2.6에 포함된 임상규칙을 분석하여 범용 컴파일러로 XSLT를 개발함으로써 CDSS의 편리성과 접근성을 높이고자 시도한다.

구체적인 목적은 다음과 같다.

첫째, 임상규칙을 표현하는 ArdenML 구문을 분석하고 Visual Basic 언어와 비교한다.

둘째, ArdenML을 변환하는 XSLT를 개발한다.

셋째, EMR시스템과 연동할 수 있는 Interface를 개발하여 고혈압 판단규칙으로 XSLT의 기능을 평가한다.

# II. 이론적 배경

## 1. Arden Syntax

Arden Syntax는 의학 알고리즘을 MLM이라는 지식모듈로 표현하는 표준화되고 정형화된 컴퓨터 절차 언어이다.

이는 임상에서 발생하는 다양한 지식을 쉽게 표현하고 작성된 지식의 공유를 통해 중복해서 작성되는 노력을 피할수 있고, 수정된 내용의 빠른 보급과 작성자에게 더 정확하고 논리적인 작성을 유도할 수 있다(Ohno-Machado, 1998).

Arden Syntax는 American Society for Testing and Materials (ASTM)에서 E-1460으로 이미 표준으로 채택했으며, 부위원회인 E31.15 Health Knowledge Representation에서 1992년에 Arden Syntax Version 1.0을 채택했다(Jenders, 2002).

1998년, 이 표준은 HL7으로 이관되었고 HL7의 Arden Syntax 기술위원회와 임상의사결정지원위원회에 의해 표준이 유지 및 관리되고 있다. Arden Syntax Version 2.0은 1999년 9월에 HL7과 ANSI에 의해 정식 표준으로 채택, 발표되었다. 가장 최근 버전은 2.6이고 XML 기반의 버전이 현재 개발 진행 중에 있다.

### 가. Arden Syntax MLM의 구조

Arden Syntax를 위한 MLM은 하나의 의학적 판단을 할 수 있는 충분한 알고리즘을 담고있는 단위다. 이 MLM 은 <그림1>과 같은 구조의

ASCII형태로 구성된다.

```
maintenance:
slotname: slot-body;;
slotname: slot-body;;
...
library:
slotname: slot-body;;
...
knowledge:
slotname: slot-body;;
...
end:
```

**그림 1. MLM구조**

MLM은 Categories와 Slots으로 구성되며, 이 Slot은 Slot-Body의 형태에 따라서 Textual-List, Textual Slot, Coded Slot, Structured Slots으로 구분할 수 있다.

Categories는 Maintenance, Library, Knowledge가 있고, 각각의 Categories에는 다양한 Slots으로 구성된다.

그 구성을 보면 다음 <표1>과 같다.

표 1. MLM Catogories And Slots의 구성과 설명

| 이름 | Category/Slot | 설명 |
|---|---|---|
| **Maintenance Category** | **Category** | **MLM관리를 위한 항목 모음** |
| Title | Slot | 제목 |
| Mlmname | Slot | MLM Name |
| Arden Syntax version | Slot | Arden Syntax의 적용된 Version |
| Version | Slot | MLM의 Version. History관리 목적 |
| Institution | Slot | MLM을 작성한 기관 |
| Author | Slot | 작성자 |
| Specialist | Slot | 작성자 및 검토자 |
| Date | Slot | 작성일자 |
| Validation | Slot | 검증 여부(testing, validating…) |
| **Library Category** | **Category** | **MLM에 관련 된 Purpose, Explanation** |
| Purpose | Slot | 목적 |
| Explanation | Slot | 설명 |
| Keywords | Slot | Search를 위한 Keywords |
| Citations | Slot | (Optional) 평론 |
| Links | Slot | (Optional) 연결정보 |
| **Knowledge Category** | **Category** | **MLM의 실제 임상진료지침 내용** |
| Type | Slot | "Data_Driven" |
| Data * | Slot | DataBase에서 필요한 자료 Read |
| Priority | Slot | Default 50, 1~99의 높은 우선순위 |
| Evoke* | Slot | MLM이 실행되어야 하는 시점 정의 |
| Logic * | Slot | Logical algorithm |
| Action * | Slot | Conclude가 True이면 실행 |
| Urgency | Slot | Action의 중요한 정도 표시(1~99) |

## 나. Arden Syntax의 문법

Arden Syntax는 기본적으로 진료 환경에서 진료를 보는 의사나 간호사들이 임상의 경험을 로직적으로 표현하기 위한 컴퓨터 언어로 개발되었다. 때문에 근본적으로 전산학을 공부한 컴퓨터 엔지니어 관점의 어려운 구조나 문법으로 구성되어 있지 않고, 쉬운 영어의 표현방식으로 임상규칙을 표현할 수 있도록 쉬운 문법체계로 개발되었다.

Arden Syntax의 문법구조를 자세히 알기 위해서는 Arden Syntax 의 BNF(Backus-Naur form)를 참고하면 된다(www.hl7.org). 그 중 주요한 내용을 정리하면 다음과 같다.

1) 데이터타입(Data Type)

Arden Syntax는 다양한 임상데이터를 처리할 수 있도록 여러가지 형태의 Data Type을 가지고 있다. 기본적인 형태는 Null, Boolean, Number, String, List등이 있고, 특이한 형태는 Time, Duration, Term Query Result, Object 등이 있다.

2) 연산자(Operators)

Operator는 단항연산자, 이항연산자, 삼항연산자 등이 있고, 논리연산자는 대부분의 컴퓨터언어에서 표현하는식과 동일하다. 이중 특이한 연산자로는 Is 연산자나 Occur/Occurred 연산자 등이 있다.

3) 제어문(Control)

제어문에는 If-Then-Else문이 있고, 반복문은 While-Loop와

For-Loop 등이 있다. 각각의 문법은 다음과 같다.

- If-Then

```
IF <expr1> THEN
<block1>
ELSEIF <expr2> THEN
<block3>
ELSEIF <exprN> THEN
<blockN>
ELSE
<blockE>
ENDIF;
```

- While-Do

```
WHILE <expr> DO
<block>
ENDDO;
```

- For-Loop문

```
FOR <identifier> in <expr> DO
<block>
ENDDO
```

4) 기타

그 외에 Arden Syntax만이 가진 특징적인 문법은 Action Slot과 Evoke Slot에 있다. Action Slot은 Logic Slot의 Conclude가 True로 결정 되었을때, 실행되는 Slot이며 원칙적으로 진료공급자에게 메시지를 보내거나 결과값을 전달한다. 잘 작성된 Action Slot은 기본적으로 Return문이나 Write문으로 구성되면 된다.

Evoke Slot은 언제, 어떤 시점에 해당 MLM이 실행되거나 점검할 지에 관한 부분이 정의되어 있다.

## 다. Arden Syntax 의 예제

지금까지 정의된 MLM의 구조와 Arden Syntax의 문법에 맞는 가장 단순한 Arden Syntax 예제를 보면 다음과 같다.

```
                                MLM
logic:   /* calculate fractional excretion of sodium */
         let fractional_na be 100 * (urine_na / urine_creat)/
            (serum_na / serum_creat) ;
         /* if the frational Na is invalid (e.g., if the */
         /* urine or serum sample is QNS) then stop here */
         if fractional_na is null then
                  conclude false ;
         endif ;
         /* check whether the fractional Na is low */
         let low_fractional_na be  fractional_na < 1.0 ;
         /* send the message */
         conclude true ;
         ;;
end:
```

**그림 2. Arden Syntax 예제**

이 예제는 환자의 Urine(소변)내 Na(나트륨)값을 Urine내 Creatine으로 나눈값과 Serum(혈액)내 Na값을 Serum내 Creatine으로 나눈값의 비율을 구해서 해당 비율이 1.0을 기준으로 낮은수치는 낮은수치에 관한 경고를 표시하고, 높은 수치는 높은 수치에 관한 경고를 표시한 MLM이다.

## 2. ArdenML

Arden Syntax는 그 개발 취지에서 나타나듯이 개인과 개인, 정보시스템과 정보시스템, 의료기관과 의료기관간의 전산화된 지식의 공유가 목적이다. 따라서 그 형태 또한 ASCII파일 형태를 취하고 있으며, 작성하는 문법 또한 임상에서 직접 작성 할 수 있는 쉬운 영어 표현방식 과 유사하다(Jenders, 2001). 하지만 이런한 표현 방식은 MLM을 작성할 수 있는 전문 Editor나 편집기 또는 Builder가 개발되면서 그 작성적인 측면에서는 많이 개선 되었으나 여전히 문제를 가지고 있다.

그 문제는 작성된 Arden Syntax를 실행 가능한 형태로 만들기 위해서는 Arden Syntax를 위한 컴파일러가 필요한데 이를 개발하는 것은 상당히 어려운 문제이다. 물론 많은 기관이나 벤더에서 Arden syntax를 위한 컴파일러를 C++, Mumps, Java 등의 형태로 개발 보급하고 있으나 이는 각각의 의료시스템에 맞는 또다른 개발이 되어야 하는 문제가 다시 발생한다.

이러한 문제를 해결할 수 있는 방법이 XML이다. 현재 Arden Syntax Version 2.1이후부터 XML형태의 Schema가 개발되어 이미 제공되고 있으며 많은 다른 연구에서도 Schema의 변경을 포함하여 다양한 연구를 하고 있다. 이중 ArdenML로 개발된 XML Version을 살펴본다.

표 2. Slot Description of Arden Version 2.1 and ArdenML

| Category/Slot | Slot Body Types in Version 2.1 | Schema | | XSL template type |
|---|---|---|---|---|
| | | Type | Use | |
| **Maintenance Category** | | | | |
| Title | textual | String | required | Push |
| Mlmname | coded | String | required | Push |
| Arden Syntax Ver | coded | Simple type based on string with enumeration value | required | Push |
| Version | textual | String | required | Push |
| Institution | textual | String | required | Push |
| Author | textual list | Complex type | required | Push |
| Specialist | textual | Complex type | required | Push |
| Date | coded | Date | required | Push |
| Validation | coded | Simple type based on string with enumeration value | required | Push |
| **Library Category** | | | | |
| Purpose | textual | String | required | Push |
| Explanation | textual | String | required | Push |
| Keywords | textual list | Complex type | required | Push |
| Citations | structured / textual | Complex type | optional | Push |
| Links | structured / textual | Complex type | optional | Push |
| **Knowledge Category** | | | | |
| Type | coded | String | required | Push |
| Data * | structured | Complex type | required | Pull |
| Priority | coded | String | optional | Push |
| Evoke* | structured | Complex type | required | Pull |
| Logic * | structured | Complex type | required | Pull |
| Action * | structured | Complex type | required | Pull |
| Urgency | coded | String | optional | Push |

자료: Jenders, RA.The Arden Syntax for Medical Logic Systems [Web Page].

MLM은 3개의 Category와 여러 그룹의 Slot들로 구성 된다. 그중 Slot은 <표2>에서와 같이 Slot-Body Type이 있는데 이는 각각의 Slot별로 다르게 정의되어 있다.

기본 Slot-boby에는 3가지 종류가 있는데, Textual Slot은 자유롭게 작성하는 형태의 Slot이고, Coded Slot은 숫자나 날짜 또는 미리 정해진 예약어 등이 사용되는 Slot이다. 마지막으로 Structured Slot은 문장구조를 같는 형태의 문법을 사용하는 Slot이다.

다른 연구에서는 Coded와 Textual Slot Type을 적용한 Schema 연구가 있었다(Jadhav & Sailors, 2003).

ArdenML의 기본 Schema구조는 다음과 같다.

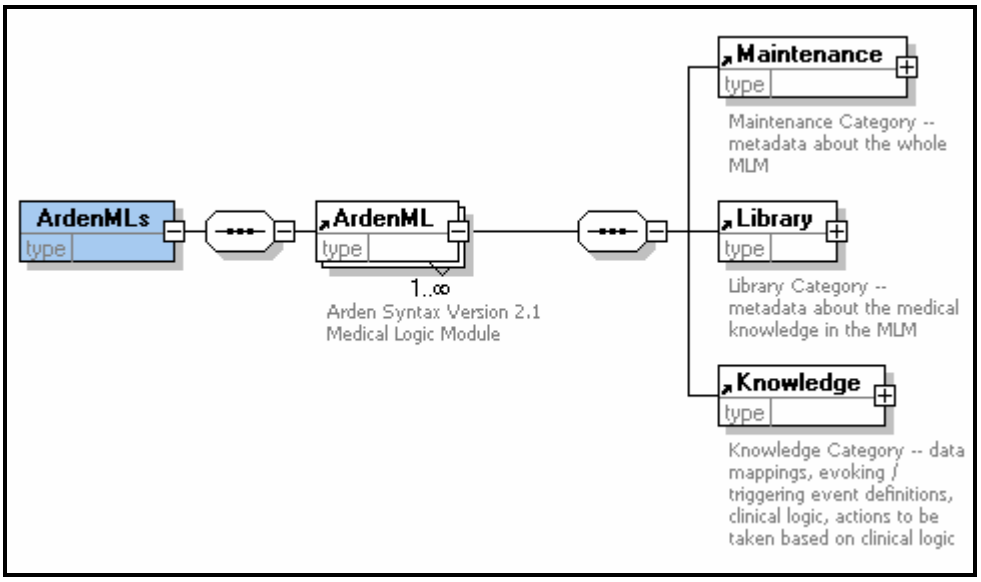


**그림 3 . Top-Level schema of ArdenML**

자료: Jenders, RA.The Arden Syntax for Medical Logic Systems [Web Page].

최상위 Schema에는 시작Tag가 <ArdenMLs>로 시작되고 이 시작 Tag안는 무수히 많은 <ArdenML>이 존재할 수 있다. 각각의 ArdenML에는 MLM의 문법구조인 3가지 Categories가 있다.

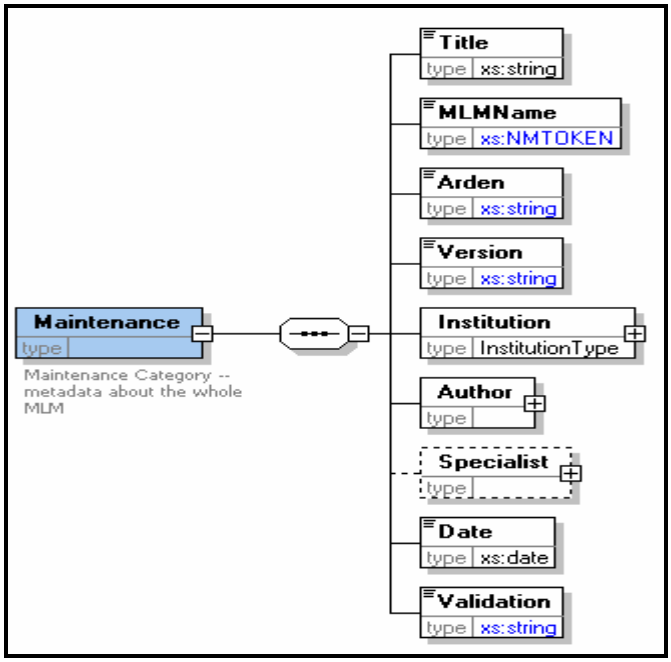


**그림 4. Schema of maintenance category**

자료:Jenders, RA.The Arden Syntax for Medical Logic Systems[Web Page].

첫번째 Category는 Maintenance로 MLM의 구조와 동일하다.

Maintenance Category는 순서대로 Title, MLMName, Arden, Version, Institution, Author, Specialist, Date, Validation으로 구성되어 있으며, XML Schema 문법상 그 순서는 바뀔 수 없다.

위의 표시방법은 XML개발도구에서 표현하는 방식으로 Box는 Tag를 나타내며 Box에는 Tag이름과 Data Type이 표시된다. Box의 우측 중간에 +표시는 Child Tag가 더 있다는 표시이고 마우스로 클릭하면 더 자세한 내용을 볼 수 있다.

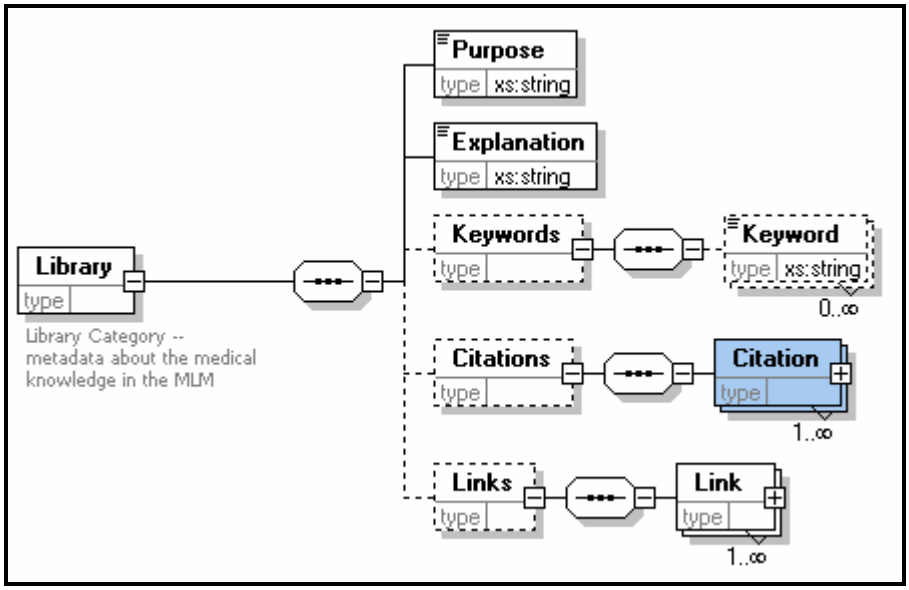


그림 5. Schema of library category

자료: Jenders, RA.The Arden Syntax for Medical Logic Systems [Web Page].

두번째 Category는 Library로 MLM의 구조와 동일하다. Library Category 는 순서대로 Purpose, Explanation, Keywords, Citations, Links로 구성되어 있으며, XML문법상 그 순서는 변경될 수 없다. 표시 문법중 0..∞ 는 반복해서 여러개 발생할 수 있는 표시이다.

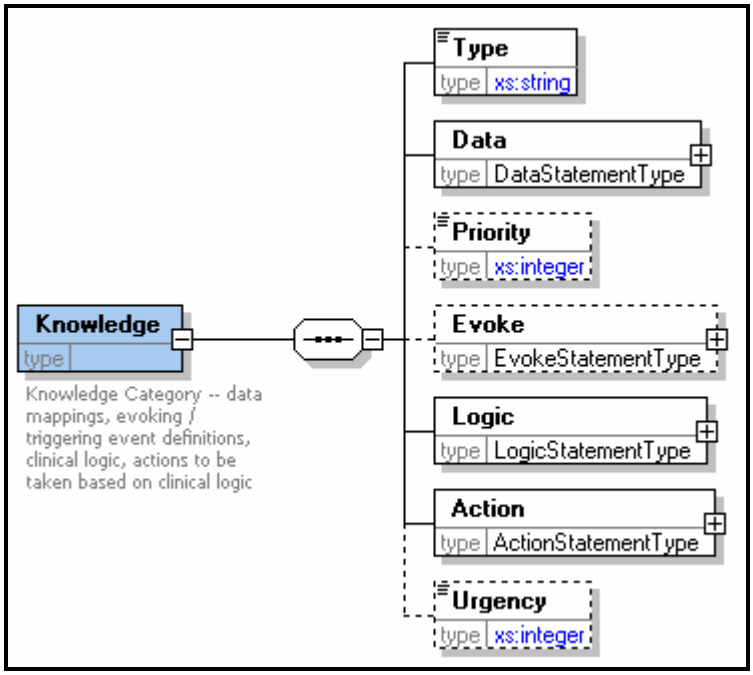


그림 6. Schema of knowledge category

자료: Jenders, RA.The Arden Syntax for Medical Logic Systems [Web Page].

마지막 Category는 Knowledge이다. 이 역시 MLM의 구조와 동일하고, 그 구성은 Type, Data, Priority, Evoke, Logic, Action, Urgency로 구성된다.

## 3. XSLT

이 문법은 XML문서를 다른 XML문서로 변환하는 XML Stylesheet Language이다.

XML이 인터넷상의 데이터 전송을 위해 설계되었다면, XSL은 이러한 XML문서를 처리하기 위해 설계되었다. XSL은 XML의 장점인 플랫폼 독립적이고, 언어 독립적이라는 점을 그대로 계승한 XML Application이라고 생각하면 된다. 이러한 XSL로 생성한 문서는 XSL 스타일시트라고 불리며, XML Schema 그리고 XML문서와 함께 데이터로써의 가치를 지니게 된다.

XSL은 크게 2가지의 파트로 구성된다. 하나는 XML문서의 변환(Transformation)을 위한 언어, 다른 하나는 포맷팅 구문(Formatting Semantics)을 기술하기 위한 어휘(Vocabulary)로 이루어져있다.

변환을 위한 언어인 XSLT(XSL Transformations)는 XPath(XML Path Language)와 함께 사용하여 XSL의 기본 골격을 구성한다. 이러한 기본 골격위에 포맷팅을 위한 어휘를 사용하여 XML문서를 표현하기도 하지만, 다른 형태의 문서를 생성하기도 한다.

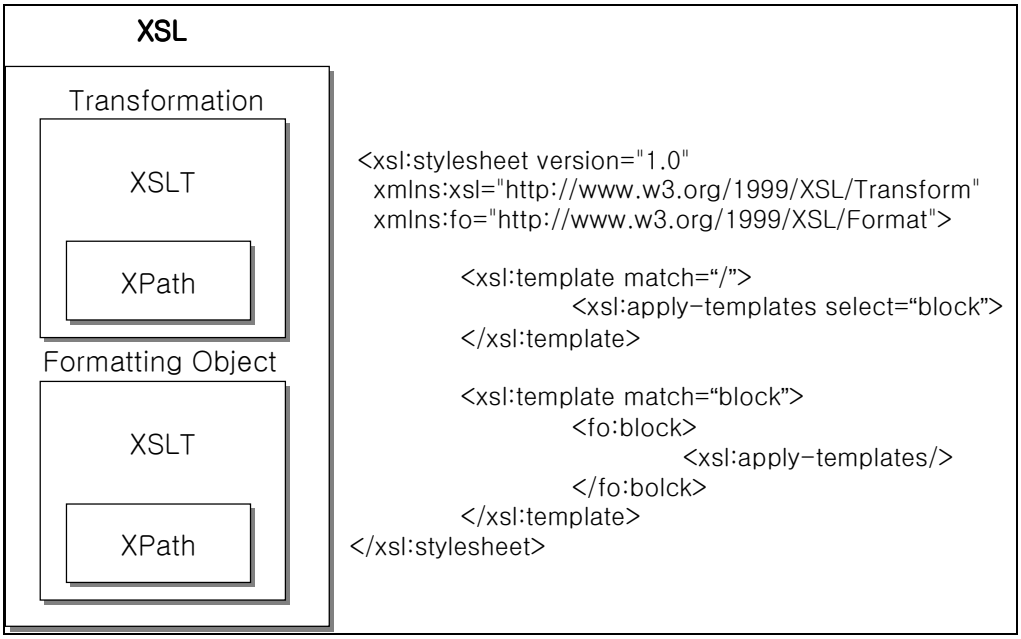


**그림 7. XSL 구성**

<그림7>에서 보이는 XSL 스타일 시트는 하나의 잘구성된(Well-Formed) XML문서다. 최상위 엘리먼트(Element)인 xsl:sytlesheet와 xsl:template, xsl: apply-templates, fo:block은 모두 시작 태그(Start-Tag)와 끝 태크(End-Tag) 혹은 빈 태그(Empty Tag)로 구성되고, XML 원칙을 준수 하고 있다.

XSL은 다양한 구성요소들을 포함하기 위해 네임스페이스(Name-spaces)를 사용하여 각 구성요소의 엘리먼트들을 구분 및 처리 하도록 규정하고 있다. 네임스페이스에 사용되는 URI(Uniform Resource Identifier)는 스펙에 기술되어 있는 것을 사용한다.

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

xsl:stylesheet는 XSLT의 최상위 엘리먼트로 XSL 스타일 시트에서 유일 하도록 규정하며, 엘리먼트에 해당하는 속성을 기술하여 스타일 문서의 추가 정보를 나타내도록 한다.

xsl:을 접두사로 갖는 엘리먼트 이름은 "http://www.w3.org/1999-/XSL/Tra-nsform"의 엘리먼트로서 처리하라는 의미를 갖는다. fo:를 접두사로 갖는 엘리 먼트이름은 http://www.w3.org/1999/XSL/Format 의 엘리먼트로서 처리하라는 의미를 갖는다.

여기서 xsl:은 XSL의 기본골격 및 최상위 엘리먼트에 사용되며, fo:는 포맷팅 을 위한 엘리먼트에 사용된다.

XSL의 기본은 룰(Rule)들로 구성된다. 즉, 룰기반 처리가 이루어 진다는 뜻 으로 룰은 XSLT의 엘리먼트중 template을 사용하며, 템플리트룰(Tempalte Rule) 혹은 생성규칙(Construction Rule)으로 불린다. 템플리트룰은 xsl:template 엘리먼트에 match 속성의 값인 패턴(Pattern)이라고 불리는 표현에 따라 처리된다. 이러한 표현은 XPath를 사용하며, XPath는 입력트리(Source Tree)인 XML문서에 접근하기 위한 언어다. 템플리트룰 내부에서 사용되는 xsl:apply-templates는 템플리트 룰들을 연결하여 하나의 결과트리(Result Tree)를 생성하 는 역할을 한다.

```
<xsl:template match="/">
 <xsl:apply-templates select="block"/>
</xsl:template>
```

XPath는 XSLT 1.0과 함께 1999년 11월 16일 현재 W3C(World Wide Web Consortium)에 XPath 1.0 으로 권고(Recommendation) 되어있다. XPath는 XSLT의 표현 언어(Expression language)로써 XSLT의 속성에서 사용된다.

XPath는 XML 문서의 특정 엘리먼트(속성,PI ..) 위치를 어떻게 기술하는지 등의 기본적인것을 제공하며, 간결하게 표현 할 수 있다. 또한 XPath는 XSLT와 XPointer에서 기반으로 사용되도록 설계되어 있다. 때문에 XPath는 문자열 기반의(String-based) 문법을 바탕으로 속성이나 URIs에서 표현 가능하다. 그러나, XSLT와 XPointer에서 사용되는 XPath가 완전히 동일하지 않기때문에 핵심함수(Core Function)를 두어 기능적으로 같도록 한다.

## 4. RBMS

규칙기반관리시스템(RBMS, Rule Base Management Sysmte)은 업무규칙, 절차, 내용, 담당자 지식 및 경험 등 다양한 비즈니스 로직을 일정한 규칙(Rule)로 표현하고, 규칙엔진으로 프로세스를 자동 제어하는 형태의 개발 방법론이다. 이전에는 규칙을 바꾸기 위해 시스템의 내부 소스코드를 수정해야 하는데 비해 RBMS로 구현된 시스템에서는 업무절차나 제어로직이 들어 있는 규칙만 수정하면 되기 때문에 기존 방식보다 간편하고 유연하다. 따라서 임상규칙을 관리하고 운용하기 위한 개발자, 사용자, 관리자 등 3자 통합관리환경을 갖춘 시스템을 의미하는 것으로 병원내부에 존재하 는 임상규칙, 절차, 노하우 등의 로직을 업무프로그램(Application)으로 부터 분리하여 Rule Engine을 관리함으로써 개발 생산성을 극대화하고 유지보수를 용이하게 하는 규칙관리시스템을 뜻한다.

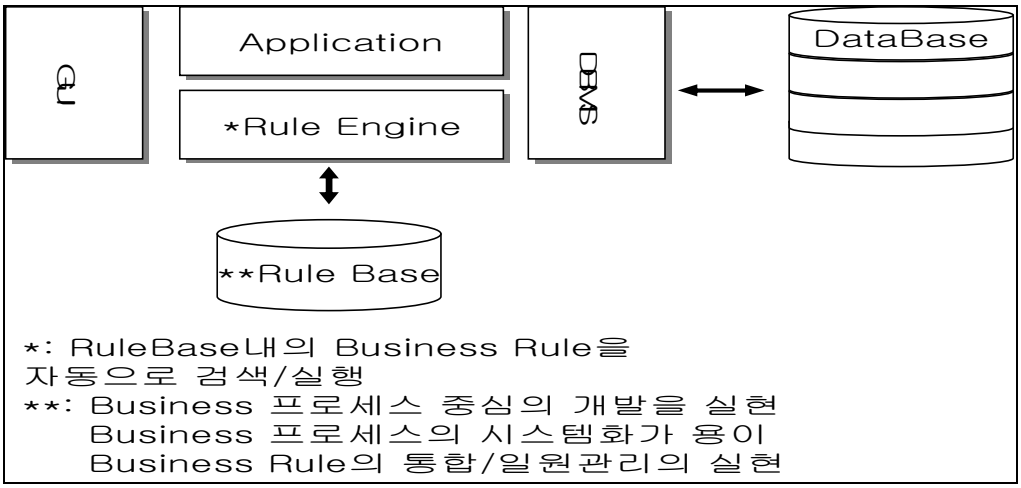


**그림 8, Rule Base Management System**

자료: (www.koreaexpert.com/technology/technology.html)

RBMS 아키텍처가 등장한 가장 큰 이유는 IT환경의 효율성과 관리환경을 극대화하기 위해서이다. 1970년대 이전의 시스템은 하나의 Application으로 구성되어 있었으며, 데이터관리 측면의 효율성이 부각되고, 데이터관리의 중요성이 인식되면서, DBMS가 등장하였고, 1980년대의 시스템은 Application에서 GUI를 분리하여 사용자화면을 구성하게 되었으며, 이로써 3-Tier아키텍처가 완성되었습니다. 하지만, 가장 중요하면서도 여전히 해결을 할수 없었던 부분이 바로 Business로직 관리였으며, 1990년대에서야 이를 해결하기 위한 방안인 RBMS아키텍처가 등장하게 되었다.

# III. 연구방법

## 1. 연구대상 및 범위

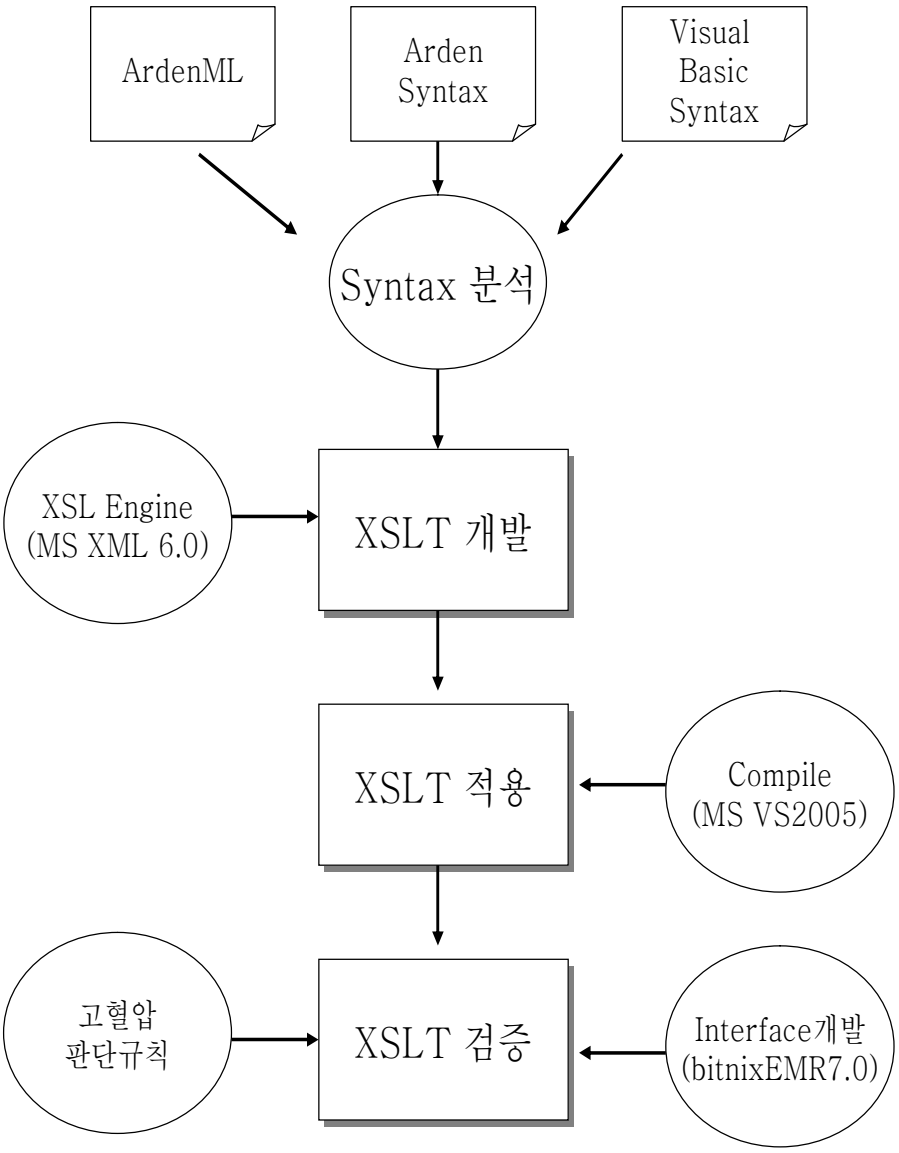


**그림 9. 연구의 틀**

연구대상은 ArdenML에 포함된 임상규칙, XSLT, 컴파일러, 사용자 프로그램이다. 연구범위는 ArdenML을 분석하고 XSLT 개발과 검증하는 부분을 포함한다. 즉, 임상규칙 표현에 사용된 ArdenML, Visual Basic Syntax의 차이점을 분석하여 XSLT를 개발하고, 이를 컴파일하고 기능을 검증하는 것이다. 기능검증을 위하여 처방전달시스템(OCS)과 연동하는 Interface를 개발하며 고혈압 룰을 수행하여 결과를 확인하였다. <그림 9>참조.

## 2. 분석방법

### 가. CDSS 구문분석

ArdenML을 Visual Basic언어로 변환하는 XSLT를 개발하기 위해서 Arden Syntax의 문법과 Visual Basic 문법간의 비교분석을 하였다. 이 비교분석을 통해 단순 변환해야 할 부분과 변환이 불가능할 경우 Visual Basic Library로 개발해야 할 부분을 정의하였다.

ArdenML을 Visual Basic으로 변환하기 위해서 Arden Syntax문법에 대한 이해와 구조분석, 그에 따르는 Visual Basic과의 대응 Syntax 및 함수의 개발이 필요하다. 따라서 가장 최근 버전인 Arden Syntax(2.6), Visual Basic(2005)의 문법 구문을 분석하여 구조, 연산자, 데이터 타입, 제어문, 지시문의 차이를 비교하였다. 이 과정에서 Visual Basic의 기본 함수 MLM의 단위와 동일하게 하나의 함수로 분리했고, 분리된 함수를 하나의 Class로 정의 하였다.Class는 Main Logic, Library, Mainteinance 이며 이 세가지 Class 중에서 변환에 핵심역할을 하는 것은 Main Logic인데 이를 함수로 처리하고 나머지 부분은 대부분 주석처리 하였으며 Main Logic을 표현하는 Data Slot, Logic Slot을 모두 구문 전환하였다.

### 나. XSLT 개발방법

MS사의 Visual Studio 2005와 XSLT Engine 6.0을 이용하여 ArdenML에 포함된 277개 임상규칙에 대하여 구조, 연산자, 데이터 타입, 제어문과 지시구문을 변환하였다. XSLT를 적용하기 위해 변환된 구문이

임상규칙문을 처리하도록 Compile하였다.

ArdenML의 임상규칙과 내용은 아래 <그림10>사이트에서 참고했다.

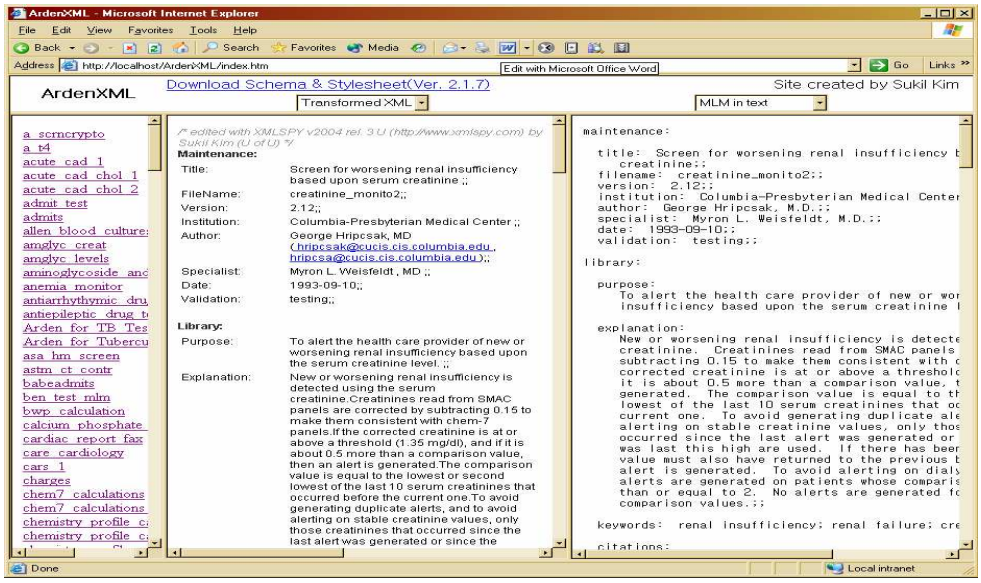


**그림 10. Site for ArdenML**

자료원: http://61.78.109.24/ArdenML [WebSite]

개발툴은 Microsoft의 Visual Studio 2005를 사용하였고, 사용언어는 Visual Basic과 .NET Framework 2.0이다. 동작수행에 관해서 Visual Basic Compiler의 오류메세지 여부로 검사하였다.

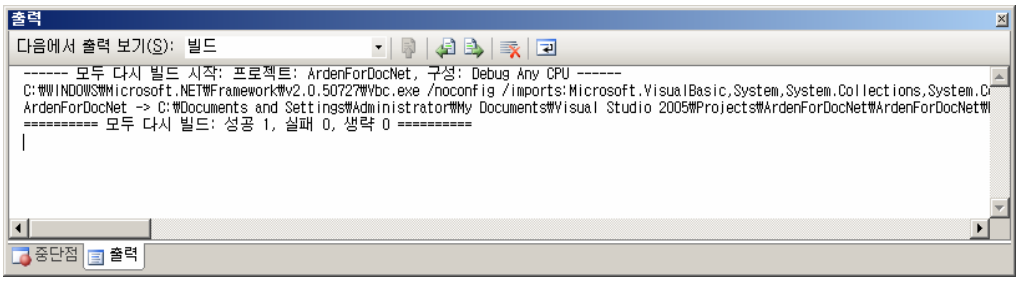


**그림 11. Visual Studio 2005 Compile Result**

개발된 Source코드 및 프로젝트의 솔루션은 <그림12>와 같이 Visual Studio 2005의 Visual Basic을 사용하였고, 그 구조는 다음과 같다.

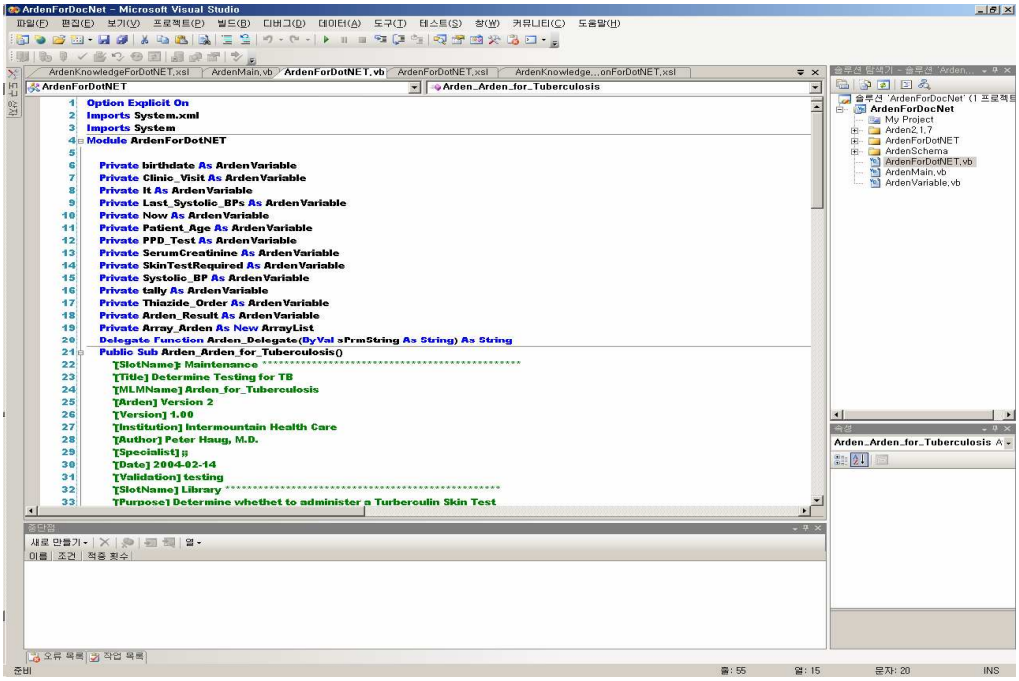


**그림 12. Visual Studio 2005 개발화면**

컴파일은 개발Tool자체의 기능을 이용하였고, 콘솔명령어를 사용해도 동일한 Visual Basic Dll Library를 생성하였다.

그 명령어는 다음과 같다.

```
Vbc.exe /rootnamespace:ArdenForDocNet
        /out:obj\Debug\ArdenForDocNet.exe
        /target:exe ArdenForDotNET.vb ArdenMain.vb ArdenVariable.vb
```

## 다. XSLT 검증방법

XSLT를 검증하기 위해 임상규칙처리를 위한 Interface를 개발하였으며 고혈압 판단룰을 적용하여 결과를 판정하였다.

### 1) 고혈압 룰

```
maintenance:
  title:  본태성hypertension 에 관한 진단;;
  filename:  hypertension_disease;;
  version:  1.01;;
  institution:  Yonsei University Graduate School of Public Health;;
  author:  Kangsoo kim(kskim@bit.co.kr);;
  specialist:  Kangsoo kim;;
  date:  2007-06-10;;
  validation:  expired;;
library:
  purpose: 본태성 고혈압에 관한 진단 및 치료에 관한 정보 제공;;
  explanation:
    본태성고혈압에 관한 기본판단은 수축기혈압과 이완기 혈압을 2~3회 측정한 값을 평균치로 한다.
    그때의 혈압이 140/90 mmHg 이상시 상병으로 인정.;;
  keywords: hypertension;;
  citations:  ;;
knowledge:
  type:  data-driven;;
  data:
      hypertension_storage := event {'32309','32404','32507'};
      hypertension_low := read last
              {'evoking','dam'="PDQRES2",'constraints'="C****"; ; '32309'};
      hypertension_high := read last
              {'evoking','dam'="PDQRES2",'constraints'="C****"; ; '32404'};
      Cheif_Complaint := read last
              {'evoking','dam'="PDQRES2",'constraints'="C****"; ; '32507'};

  evoke: hypertension_storage ;;
  logic:
      /* BUILD DISPLAY.  INCLUDES ONLY NON-NULL RESULTS */
     if any (Cheif_Complaint are in target_diagnoses) then
       conclude true;
     endif;
      /* 이완기 혈압값이 90이상이면... */
      if hypertension_low > 90 then
          conclude true;
      endif;
      /* 수축기 혈압값이 140이상이면... */
      if hypertension_high > 140 then
          conclude true;
      endif;
      ;;
  action:
     write "혈압이 140/90mmHG 이상의 고혈압 환자입니다.";
     ;;
end:
```

그림 13. 고혈압 룰

XSLT 검증을 위해 범용적이고 간단한 고혈압 판정문을 선정하였다. 혈압은 심장의 수축하는 운동과 혈관의 저항 양쪽사이에서 생기는 것으로 혈관벽을 미는 힘(압력)으로 정의할 수 있다. 고혈압은 일반적으로 동맥성 혈압 측정을 의미하는것으로 이삼일 간격으로 두번이상 외래방문시 측정값이 140 / 90 mmHg이상일때 통상적으로 인정한다. 이때 종합적 판단을 위해 문진과 더불어 기본검사, 안저검사, 신장검사, 심장검사 등을 시행하여 위험요소나 건강상태를 평가한다.

따라서 고혈압을 판단하는 룰에는 다양한 정보와 복잡한 의사결정 단계가 존재하지만 여기서는 가장 단순한 룰을 선정하였다. 즉 환자의 혈압 측정값을 기준으로 이완기 혈압이 90이상이고 수축기 혈압이 140이상이면 고혈압으로 판단하는 규칙(Rule)을 사용하였다. 개발된 규칙은 다음 <그림13, 14>와 같다.

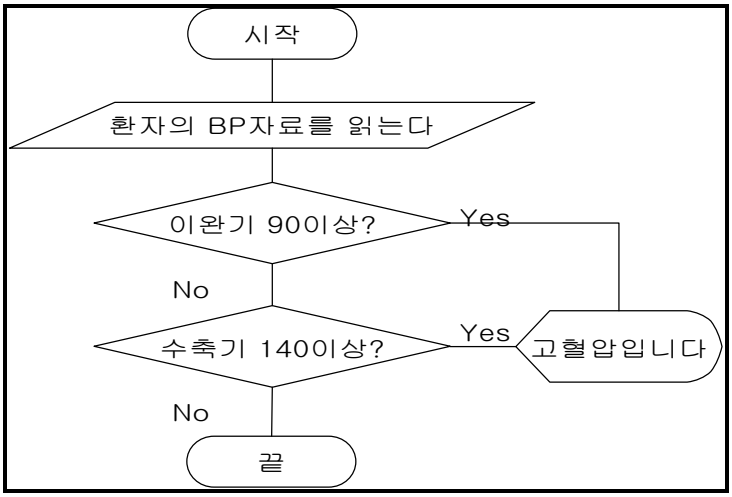


**그림 14. 고혈압판단 룰의 순서도**

### 2) Interface개발방법

기존에 사용하는 처방전달시스템과 연동하기 위해 2005년에 B사에서 개발된 HIS시스템에 포함된 bitnixEMR 7.0 시스템의 OCS Interface 를 개발하였다<그림 15>.

처방전달시스템과 ArdenML 엔진을 연동하기 위해서는 처방전달시스템에 사용되는 Arden Service Interface, Aren Service Agent, Arden DataBase 등의 컴포넌트개발이 필요하다. 즉, Service Interface는 기존의 시스템과 연동하기 위한 함수들이 포함된다. 또한 Service Agent는 연동된 자료를 전달, 처리하는 함수가 포함되고, Arden DataBase는 ArdenML에 관련된 Rule을 저장하고 관련된 데이터를 처리하는 함수가 포함된다. 이러한 함수들을 먼저 개발하고 Entry point기능을 하는 연동 Interface 함수를 기존 시스템에 삽입하여 임상규칙 사용자 프로그램을 완성였다.

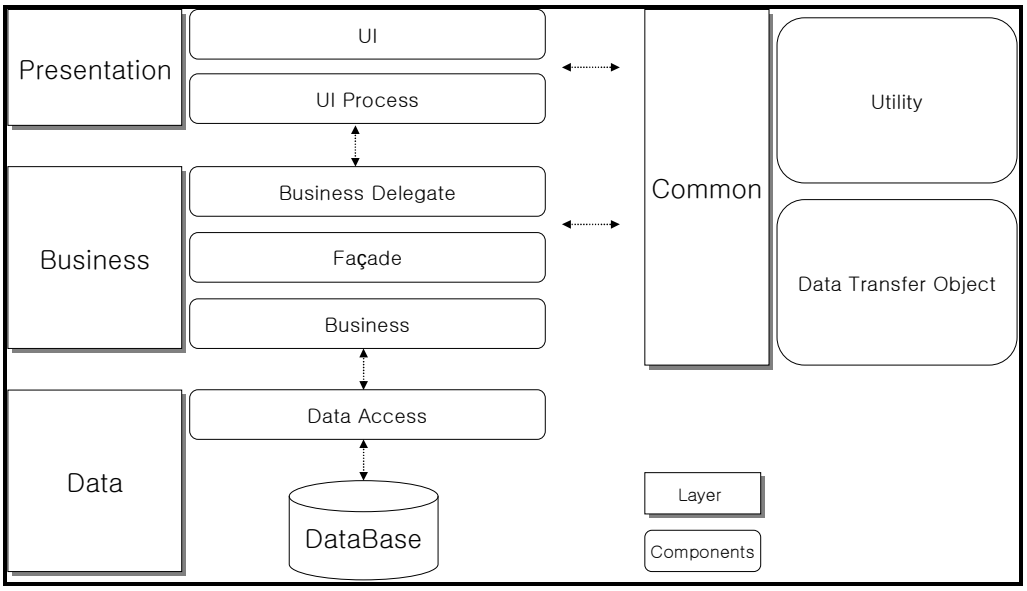


그림 15. bitnixEMR 7.0 System Architecture

이 임상규칙 사용자 프로그램에 고혈압 판정에 필요한 조건에 해당하는 실제 값을 입력하고 작동결과를 확인하였다

# IV. 연구결과

## 1. CDSS 구분분석

**표 3. ArdenML와 Visual Basic.NET 의 구조 비교분석 결과**

| 구문 | Arden Syntax* | ArdenML** | VB.NET*** |
|---|---|---|---|
| | Maintenance(C) | <Maintenance> | Comment |
| | Title(S)* | <Title> | |
| | Mlmname(S) | <MLMName> | |
| | Syntax version(S) | <Arden> | |
| | Version (S) | <Version> | |
| | Institution (S) | <Institution> | |
| | Author (S) | <Author> | |
| 차이 | Specialist (S) | <Specialist> | |
| | Date (S) | <Date> | |
| | Validation (S) | <Validation> | |
| | Library(C) | <Library> | |
| | Purpose (S) | <Purpose> | |
| | Explanation(S) | <Esplanation> | |
| | Keywords (S) | <Keywords> | |
| | Citations (S) | <Citations> | |
| | Links (S) | <Links> | |
| | Knowledge(C) | <Knowledge> | |
| | Type (S) | <Type> | |
| | Data (S) | <Data> | Function |
| | Priority (S) | <Priority> | Comment |
| | Evoke (S) | <Evoke> | Events |
| | Logic (S) | <Logic> | Class |
| | Action (S) | <Action> | Procedure |
| | *Urgency*(S) | <Urgency> | Comment |

*: Category(C), Slot(S), **: tag, ***: type

위의 <표3>은 ArdenML과 Visual Basic 구문의 구조적 차이를

분석한 결과로 Arden Syntex 은 category, ArdenML은 Tag, VB.NET은 Comment, Function , Events, Class, Procedure로 구성되어 있었다 <표3>.

Arden Syntax는 주어진 상황에서 임상적인 의사결정을 어떻게 할지 표현해 주는 임상규칙을 표현하는데 사용하는 일종의 컴퓨터언어 이다. 따라서 Visual Basic이나 Java, C#과 같은 컴퓨터언어가 가지고 있는 대부분의 특징이 정의되어 있다. 따라서 Visual Basic으로의 정확한 변환을 위해서 컴퓨터 언어적인 측면에서의 비교를 해보면 다음 <표4>와 같다.

**표 4. 데이터 타입 분석결과**

| 구문 | Arden Syntax | VB.NET | Description |
|---|---|---|---|
| | Null | Nothing | 특정하지 않은 값을 나타내는 특수한 타입 |
| | Boolean | Boolean | True/False을 갖는 형식 |
| | Number | Integer | 숫자형식 |
| 차이 | Time | Date | 날짜형식 |
| | Duration | Date | 날짜형식 중 기간을 표시하는 형식 |
| | *String* | String | 문자형식 |

대부분의 형식이 동일하나 Time 및 Duration같은 경우는 Visual Basic의 Date형식과 비슷하다. 하지만 Arden Syntax에서 이야기하는 Time과 Duration은 지정된 형식이 있어서 Visual Basic의 Date형식에 일정형식을 지정해주어야 한다. 해당 형식은 표준 ISO 8601:1988 (E)에 따르는 yyyy-mm-ddThh:mm:ss 이다.

**표 5. 연산자 분석결과**

| 구문 | Arden Syntax | VB.NET | 차이 | 설명 |
|---|---|---|---|---|
| 차이 | Unary | Unary | 비슷 | 단항 연산자 |
| | Binary | Binary | 비슷 | 이항 연산자 |
| | Ternary | Ternary | 없음 | 삼항 연산자 |
| | List | Array | 비슷 | 주어진 갑을 목록으로 처리하는 연산자 |
| | Where | – | 없음 | SQL문의 Where조건과 유사하고, 조건이 True인 경우에 만 값이 전달되는 연산자. |
| | Or/And/Not | Or/And/Not | 동일 | 논리 연산자 |
| | =,<>,<,>,<=,>= | =,<>,<,>,<=,>= | 동일 | 관계 연산자 |
| | Is,Are,Was, Were | – | 없음 | 다양한 조건에 맞는 경우에 True,틀리면 False를 전달하는 연산자 |
| | Occurs/Occurred | – | 없음 | 특정 시간을 비교하는 조건문을 동반하여 조건이 True일 때 만 값을 전달하는 연산자 |
| | String | – | 비슷 | 문자열에 대한 다양한 처리를 위한 연산자 |
| | +,–,*,/,** | +,–,*,/,^ | 비슷 | 산술 연산자 |
| | After/Before/Age/From | – | 없음 | Duration 데이터 타입의 값을 처리하기 위한 연산자 |
| | Year/Month/Week/Day/Hour | – | 비슷 | Duration 데이터 타입의 값에서 특정 일자 형식을 취하기 위한 연산자 |
| | Count/Exist/Sum/Last/First/Any/No/All/Latest | – | 비슷 | 집합 연산자로 주어진 값들에 대한 합계, 평균 등의 집합 연산을 위한 연산자 |

Arden Syntax의 연산자는 일반적인 컴퓨터 언어에서 지원하는 논리, 산술, 대입, 비트, 비교 등의 연산자를 지원하면서 의료정보의 특성상 자주 사용되고 많이 쓰이는것들을 미리 정하여 하나의 연산자로 자동수행 될 수 있게 다양한 연산자가 정의되어있다.

그중에 특징적인 것 몇가지를 알아보면 다음과 같다.

Occur/Occurs/Occurred 연산자의 경우 BNF는 <n:Boolean> := <n:any-type> OCCUR WITHIN PAST <n:duration> 게 표현되며, 사용법은 true := query_result OCCURED WITHIN PAST 3 days와 같이 사용한다. 그 의미는 query_result에 해당하는 값이 3일 전 이내에 발생한 자료이면 논리값으로 True가 넘어오고 그 이전에 발생한 자료이면 False가 발생하여 해당값의 발생일자의 조건을 주어 필요한 값만을 취하기 위한 연산자이다.

Any 연산자의 경우 BNF는 <1:Boolean> := ANY <n:any-type>에 표현되며, 사용법은 true := ANY (true,false,false)게 사용한다. 그 의미는 Any 연산자 뒤에 주어지는 값들의 논리연산 값들 중에 하나라도 True가 있으면 True를 전달하는 연산자이다. 이는 여러가지 조건이 존재할 때 그러한 조건들을 하나하나 비교하여 조건값을 취하는 것이 아니라 간단한 ANY 연산자로 모든 조건을 비교하여 값을 쉽게 얻을 수 있는 연산자 이다. <www.hl7.org>

**표 6. 제어문과 지시문 분석결과**

| 구문 | Arden Syntax | VB.NET | 차이 | 설명 |
|---|---|---|---|---|
| 차이 | If-Then-Else | If-Then-End If | 거의 동일 | 조건에 맞는 경우에만 실행되는 제어문 |
| | While-Loop | While-End | 거의 동일 | 특정 조건에 맞는 경우 해당 조건이 틀릴 때 까지 반복해서 실행하는 제어문 |
| | For-Loop | For-End For | 거의 동일 | 조건에 상관없이 지정한 횟수만큼 반복해서 실행하는 제어문 |
| | Conclude | - | 없음 | MLM의 결론을 정하는 지시문. 값이 True면 Action Slot을 실행하고 False이면 실행하지 않는다. |
| | Call | Call | 거의 비슷 | 외부의 MLM을 호출할 때 사용하는 지시문 |
| | Read | - | 없음 | 환자의 데이터를 읽어오는 지시문, DB와 연결되는 부분이나 인증에 관련된 부분의 정의없이 원하는 자료를 읽어오기 위함. |
| | Destination | - | 없음 | 원하는 값을 특정한 목적지에 전달하기 위한 지시문, E-mail발송이나 다른 사용자에게 전달 할 수 있다. |
| | *Write* | - | 없음 | 특별한 값을 문장 또는 코드화 된 메시지 형태로 원하는 목적지에 보내는 지시문 |

Arden Syntax의 다양한 지시문은 일반적인 컴퓨터언어에서 지원하지 않는 형태의 문장으로 특수한 목적을 가지고 정의되어 있는 지시문이 대부분이다.

Read문의 경우 Logic Slot의 Rule을 점검하기 위해 필요한 다양한

환자의 데이터를 DataBase에서 얻어올 경우 사용할 수 있는 지시문인데, 일반적인 컴퓨터 언어에서는 얻어 오고자 하는 DataBase의 종류나 연결 문자열, 접속자의 권한을 위한 접속 아이디, 접속암호 등이 지정 되어야 하나 여기서는 그러한 기술적인 내용은 전혀 지정할 수 없고, 단지 원하는 자료가 어디에 존재하던 연결되어 있고, 원하는 위치에 원하는 자료가 존재한다는 전제하에 단순히 읽어오는 지시문으로 표현하는 방식이다.

여기에 나타나는 이러한 문제점은 Curly Brase 문제로 Arden Syntax의 큰 문제점으로 대두되고 있으며, 이에 대한 다양한 연구가 진행중이다. 이 연구에서는 구축되어 있는 처방전달시스템의 구조에 맞게 Query를 구성하여 처리하였다. Conclude문은 Logic Slot에 의해 점검된 결과가 True이면 Action Slot의 지정된 결론을 실행할지 여부에 대한 결론을 지정하는 지시문이다. Conclude True이면 Action Slot이 실행되며, Conclude False로 지정되면 Action Slot을 실행하지 않고, 그냥 종료된다.

Write문은 Action Slot에서 Conclude True에 의해서 실행될 때, 사용자에게 전달하고 싶은 문장이나 코드화된 메시지를 만들 때 사용된다. 여기에는 또한 AT문이 추가되어 원하는 목적지를 지정, E-mail 이나 메시지 형태로 전달할 수 있다.

## 2. XSLT 개발

분석된 Arden Syntax와 Visual Basic의 Syntax비교에 의해 ArdenML로 작성된 XML을 XSLT를 이용하여 Visual Basic언어로 변환한다.

ArdenML은 기본적으로 Arden Syntax를 표현하는 Tag의 구조로 되어 있다. 따라서 해당 구조를 위에서 분석한 Arden Syntax와 Visual Basic의 Syntax에 맞게 변환하는 것이 중요하다.

다음 <표7>은 XSLT를 이용한 ArdenML과 Visual Basic의 변환 기준표 이다.

**표 7. ArdenML과 Visual Basic 비교분석**

| 종류 | 변환차이 | 설명 |
|---|---|---|
| 데이터 타입 | 거의 동일 | Null값을 제외한 데부분의 형식을 그대로 사용 |
| 연산자 | 상이 | 기본적인 논리, 관계, 산술연산자는 동일하나 나머지 연산자는 상이하여 그 연산자의 의미를 수행하는 함수로 작성. |
| 제어문과 지시문 | 상이 | If/While/For 등의 제어문은 비슷하나 문법이 틀리고, 그 외의 대부분 지시문은 없거나 상이하여 그 지시문의 의미를 수행하는 함수로 작성. |
| 기타 | 상이 | Arden Syntax에서 필수로 기재하는 Categories나 Slots에 대한 부분은 의미적으로 필요을 경우 주석으로 처리. |

Arden Syntax의 Category들 중 Maintenance나 Library의 경우 실제 CDSS를 수행하는데 의미있는 자료를 제공하지 못하기 때문에 모두

주석으로 처리했다. 또한 Knowledge Category에서도 Data Slot과 Logic Slot 그리고 Action Slot만을 변환대상으로 선정했다. 그 외의 나머지는 모두 주석으로 처리했다.

<표7>에서 비교분석된 변환결과를 보면 데이터 타입을 제외한 대부분의 연산자, 지시문, 기타 내용등이 모두 상이하여 단순 변환으로 처리가 되지 않았고, 대부분 Visual Basic Library로 변환시킴으로써 문제점을 처리하였다. 상세한 그 처리방법을 정리하면 다음 <표8>과 같다.

**표 8. 전환 결과**

| 전(ArdenML) | 후(VB.NET) | 설명 |
|---|---|---|
| Name | MLM Name | MLM의 이름을 함수의 이름으로 적용해서 Arden_로 시작. |
| Data Slot | Procedure | (MLMName)_Arden 함수의 Procedure로 처리 |
| Logic Slot | Procedure | (MLMName)_Arden 함수의 Procedure로 처리 |
| Action Slot | Arden_Conclude | Action Slot의 내용을 Arden_Conclude함수로 분리하여 Arden_(MLMName)함수에서 Call하는 형래. |
| *Evoke* | Delegate | 위임 연산자를 사용하여 Events가 발생하는 시점을 외부에서 감지하여 해당 함수가 실행. |

MLM의 이름은 Visual Basic의 Function으로 정하였고, Logic slot의 대부분 연산자 및 지시문은 MLMName_Arden형식으로 별도의 Visual Basic Library를 개발하였다.

Action Slot은 Conclude가 True일 경우 별도로 실행되어야 하기 때문에 Arden_conclude로 분리하고 함수내에서 Conclude가 True일 때 호출하여 실행될 수 있게 처리 하였다. 위와 같은 방법으로 ArdenML을

변환하는 XSLT은 다음 <그림10> 와 같다.

```
Arden For DotNET XSLT

<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with Visual Studio 2003 by Kangsoo Kim -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
        <xsl:output method="html" encoding="UTF-8" indent="yes"/>
        <xsl:include href="ArdenMaintenanceForDotNET.xsl"/>
        <xsl:include href="ArdenLibraryForDotNET.xsl"/>
  <xsl:include href="ardenKnowledgeForDotNET.xsl"/>
  <xsl:template match="/">
          <HTML>
          <BODY>
      Option Explicit On
      <br></br>
      Imports System.xml
      <br></br>
      Imports System
      <br></br>
      Module ArdenForDotNET
      <br></br>
      <!--
        VB.NET에서 참조할 모든 변수를 선언한다.
      -->
      <xsl:call-template name="SelectAllVairable">
      </xsl:call-template>
      <br></br>
      Public Sub Arden_<xsl:value-of select="normalize-space(/ArdenMLs/ardenML/Maint
                enance/MLMName)"/>()
      <br></br>
      <xsl:for-each select="/ardenMLs/ardenML">
                <xsl:apply-templates/>
      </xsl:for-each>
      <br></br>
      End Sub
      <br></br>
      <!--
          ACtion관련된 부분은 함수를 별도로 작성한다.
      -->
      <xsl:apply-templates select="//Knowledge/Action"/>
      <br></br>
      End Module
      <br></br>
    </BODY>
          </HTML>
        </xsl:template>
</xsl:stylesheet>
```

그림 16. XSLT For Visual Basic

그 외 ArdenMaintenanceForDotNET.xsl <부록1>과 ArdenLibrar-yForDotNET.xsl <부록2>, ArdenKnowledgeForDotNET.xsl <부록3>에 대한 xsl 파일은 <부록>을 참조한다.

위에서 언급했듯이 Arden Syntax의 다양한 연산자와 지시문은 Visual Basic의 Syntax로는 1:1로 변환이 불가능하다. 따라서 해당 기능을 하는 Visual Basic함수를 별도로 작성하여 Visual Basic용 Libray Class를 별도로 작성한것을 정리하면 다음<표9>와 같다.

**표 9. 변환된 최종결과**

| Visual Basic Library Name | 설명 |
|---|---|
| ArdenMain | Arden Syntax와 동일한 명칭에 _Arden을 붙여서 생성한 Visual Basic함수 모음 |
| ArdenVariable | Arden Syntax에서 사용하는 대부분의 변수는 단순히 값을 가지는 데이터 타입이 아니라 값과 발생한 일시, 또한 여러 List가 할당될 수 있는 복합변수 이다. |
| ArdenForDotNet | 변환된 Visual Basic Source가 저장되는 별도의 Class 파일이다. |

ArdenML을 작성하는 목적중의 하나는 다른 프로그램 개발자의 지식 표현을 전통적인 컴파일러를 통한 수정없이 변환이 가능하게 하는것이다. 그러기 위해서는 ArdenML의 작성을 5단계로 구분하여 작성할 수 있는데, 0단계에서 4단계의 수준으로 진행될수록 더 이상적인 Schema로 발전하는 것이다(Sailors, 2001). 그 단계를 표현하면 다음 <표10>와 같다.

**표 10. ArdenML의 Encoding단계**

| Level | Encoding |
|---|---|
| 0 | 커다란 CDATA 구조안에 존재하는 MLM을 감싸는 단순 처리 |
| 1 | 계층적 태그시스템으로써 존재하는 Category와 Slot을 처리 |
| 2 | Citation and links slot을 위해 상세한 태그시스템을 개발 |
| 3 | Knowledge Category에서 전체문과 제어구조를 구조적 처리 |
| 4 | Knowledge Section의 명시적으로 구조화된 연산자와 피연산자의 표함한 완전한 XML 처리 |

하지만 변환중에 단순한 1:1로 대응되어 Visual Basic으로 변환하는 것은 불가능하였고, 그에 대한 처리방법으로 ArdenML의 문법과 동일한 기능을 하는 Visual Basic Library를 개발하여 ArdenMain로 분리하였다. 그 개발한 함수를 정리하면 <표11>과 같다.

**표 11. Visual Basic Main Library 개발 결과**

| VB.NET | 개발개수 | 설명 |
|---|---|---|
| ArdenMain | 117 | Arden Syntax에서 처리가 불가능한 기능을 VB함수로 분리해서 개발 |
| ArdenVariable | 1 | Arden Syntax에서 사용하는 대부분의 변수는 단순히 값을 가지는 데이터 타입이 아니라 값과 발생한 일시, 또한 여러 List가 할당될 수 있는 복합변수 이다. |
| ArdenForDotNet | 277 | 변환된 Visual Basic Source가 저장되는 별도의 Class 파일이다. |

위의 ArdenMain은 Arden Syntax가 가지고 있는 다양한 함수들, 예를 들면 Any, All, Reverse, Last 등과 같이 Visual Basic의 Syntax에서는 구현할 수 없는 문법의 연산자들이 상당수 존재한다. 따라서 동일한 기능을 갖는 함수 117개를 개발하였고, 원래의 의미를 유지하기 위해 함수명은 기존의 연산자와 피연산자에 Arden을 붙여서 이름을 정하였다.

ArdenVariable은 Visual Basic의 DataType에는 없는것으로 단순한 값을 갖는 형태가 아니라 해당 값의 발생일시, 기간, Duration 등의 복합적인 값을 갖는 변수로 처리되어야 한다. 따라서 Visual Basic에서는 단순값의 형태가 아니 복합변수를 선언하기위해 ArdenVariable Class를 선언하여 다양한 형태의 처리를 가능하게 하였다.

ArdenForDotNet함수는 ArdenML로 개발된 다양한 MLM을 포함하는 하나의 Class로 개발되는 MLM을 별도의 Class로 관리하는 것이 아니라 지속적으로 하나의 Class로 관리하기 위해서 통합Class로 개발하였다.

ArdenML을 Visual Basic Source코드로 변환하기 위해 개발한 XSLT는 ArdenML의 Schema를 기준으로 모든 element를 변환하였고, 그 Source는 위 <그림16>과 같다.

해당 연구 결과물이 인터넷을 통해 제공하기 위해 기본 OUTPUT METHOD가 “HTML”로 정하였다. 함수의 이름은 Arden_MLMName으로 정하고 향후 변환되는 모든 MLM은 상기와 같은 형태로 변환된다.

ArdenML의 Plus(+)연산자 하나에 대해서 Visual Basic의 문법구조로 표현하는 XSLT의 구문의 예시를 들면 <그림17>과 같다.

```
<xsl:template match="Apply/Plus">
        <xsl:for-each select="following-sibling::*">
                <xsl:apply-templates select="."/>
                <xsl:if test="position()!=last()">
                        <xsl:text> + </xsl:text>
                </xsl:if>
        </xsl:for-each>
</xsl:template>
```

**그림 17. Plus 연산자를 위한 XSLT**

XSLT는 기본은 template의 단위로 실행된다. 즉, template rule 혹은 생성규칙으로 불리는 단위로 실행이되고 이는 element에 match속성의 값인 pattern이라고 불리는 표현에 따라서 처리된다. 따라서 위의 첫번째 <xsl:template match="Apply/Plus">은 ArdenML문서내의 여러 element 들 중에서 Apply의 자식노드인 Plus를 만나면 처리되는 Rule이다.

For-each문에서 select구문은 XPath구문으로 원하는 노드를 찾기 위한 검색표현식 정도로 생각하면 된다. 그중 following-sibling::*는 자식 노드 들중에서 형제관계에 있는 모든노드들을 검색하는 Xpath표현 식이다.

위와같이 Loop를 돌면서 지속적으로 plus(+)연산자를 표현하게 되면 Visual Basic의 plus(+)연산자와 동일한 형태의 문법구조로 표현되게 된다.

## 3. XSLT 검증

고혈압으로 작성된 규칙을 위에서 개발한 XSLT로 변환한 Visual Basic Source코드를 Compile하여 실제 변환된 내용의 문법적오류 검사와 실제 OCS시스템과 연동하기 위해서는 OCS시스템에 연동관련 개발이 필요하다 따라서 다음과 같이 OCS시스템에 연동하기 위한 기본적인 개념도를 보면 다음 <그림19>과 같다.

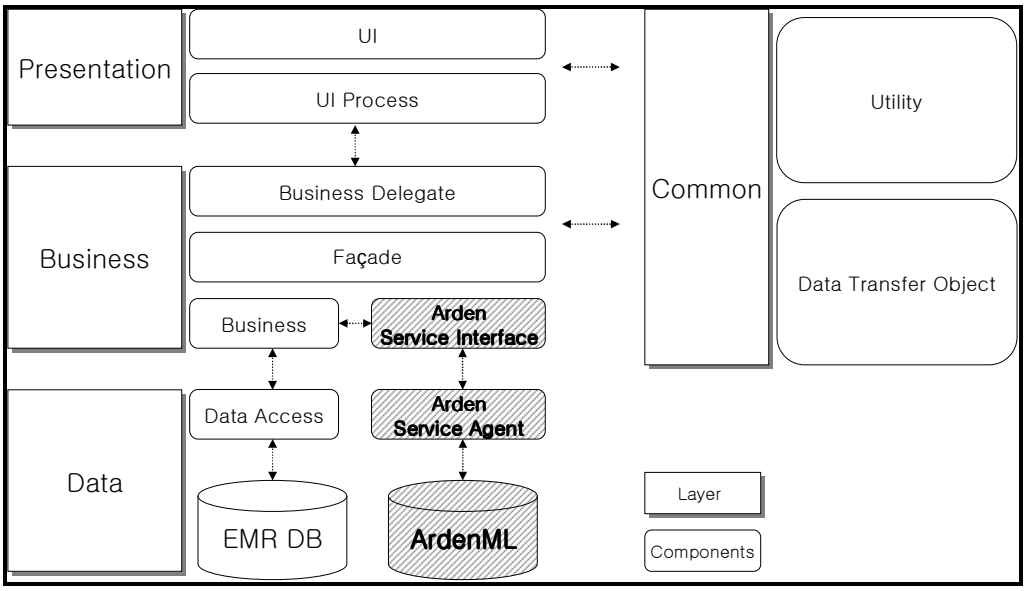


**그림 18. ArdenML 연동을 위한 시스템 구조**

시스템 구성도에는 크게 Presentation Layer와 Business Layer, 그리고 Data Layer로 분리할 수 있는데, 여기서 ArdenML은 Data Layer와 Business Layer에 걸쳐 위치하고 있다. 이유는 ArdenML은 기간시스템인 OCS와 연동하기 위해 많은 부분을 수정하지 않고, 최소한의 개발 및 수정

만으로 기간시스템과 연동될 수 있어야 한다. 따라서 Data Layer의 ArdenML은 Curly Brace 문제를 해결하기 위해서 기존Data와 연동되는 부분을 처리하기 위해서 Data Layer에 위치하고, 수집된 데이터는 Arden Service Agent를 통해서 전달된다. 이때 수행되는 ArdenML의 Rule은 Arden Service Interface를 통해서 기간시스템과 연동된다. 그중 가장 핵심이 되는 부분은 다음과 같이 기간시스템에서 Arden Service Interface로 전달하는 Observer Pattern의 함수이다.

```
ArdenObserver.GetInstance().Notify(CType(Me,
    IArdenObserverable), oPatientItem)
```

이 함수는 향후 추가되는 다양한 Arden Syntax Rule들을 포함하는 접속지점(Entry Point)역할을 하는것으로 향후 추가 수정이나 관리할 필요가 없으며 개발된 ArdenMain.dll과 ArdenVariable.dll을 OCS에 참조를 하였다. 개발된 화면에 159/100의 혈압값을 입력했을 때 결과로 나타난 고혈압 경고 메세지는 <그림 19>와 같았으며 여러가지 혈압값에 대해 오동작은 없었다.

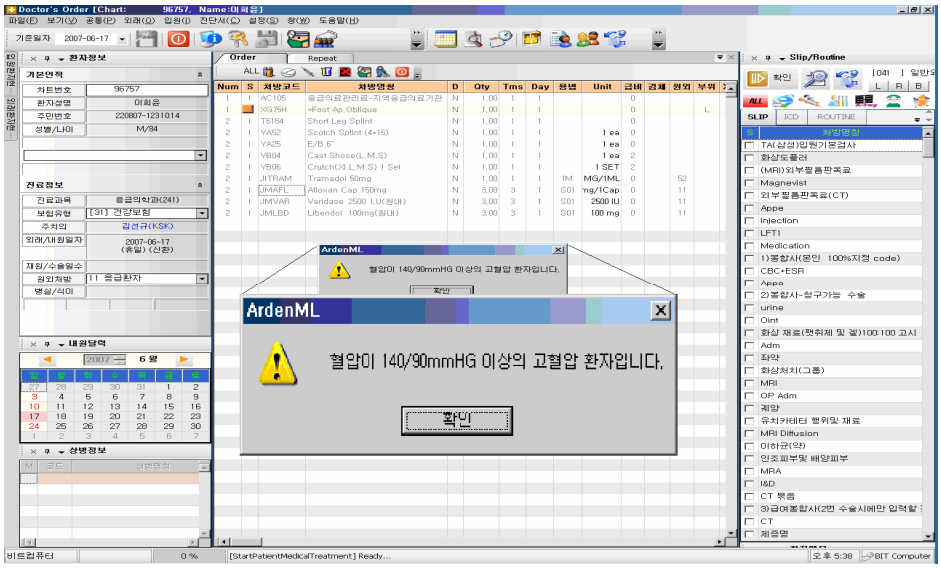


그림 19. OCS와 연계한 ArdenML 화면예시

# V. 고찰

Arden Syntax는 임상의사결정지원시스템(CDSS)의 핵심적인 부분을 담당하고 있지만 그 활용이 매우 저조하다. 많은 이유가 있겠지만 Arden Syntax가 자체적으로 실행될 수 있는 컴파일러를 가지고 있지 못한점과 Curly Brace 문제가 가장 크다고 하겠다. 이러한 문제점을 해결하는 방법으로 Arden Syntax를 XML로 작성하는 ArdenML을 개발하였고, 개발된 ArdenML은 XSLT를 이용하여 범용 컴퓨터 언어로의 전환 이 가능하였다. Arden Syntax가 임상에서 발생하는 다양한 경우를 표현할 수 있도록 개발이 되었지만 이러한 문법들은 컴퓨터 언어로의 변환이 까다로웠지만, 관련된 Library를 개발하여 이러한 까다로운 부분을 해결하였다

Arden Syntax를 작성하는 방법이 운영되는 시스템마다 다른경우 해당 시스템과 연계하기 위해서는 해당 시스템언어 개발이 필수이다.

따라서 Arden Syntax를 Java로 개발하거나(Karadimas, 2002), C++로 개발된(Kuhn, 1994) 연구가 있다. 또는 컴파일된 형태가 아니라 인터프리터 방식의 연구(Gao, 1993)도 있으나 모든 방식이 범용으로 개발되지 못하고 실행환경에 따라 별도의 추가개발을 해야하는 문제가 발생한다. 이는 작성된 임상규칙(MLM)이 지식의 공유와 재사용 이라는 관점에서 많은 문제를 갖고 있으며, 재 작성된 임상규칙이 해석하는 사람마다의 오해와 잘못된 실수로 오류발생율이 높아질수 있다.

본 연구에서는 Visual Basic으로 변환하는 XSLT를 개발하였지만 필요에 따라서는 Java나 C언어로의 변환도 가능할것으로 사료된다. 즉, ArdenML로 작성된 규칙(Rule)은 실행환경이 다른 시스템에도 간단한 XSLT만으로 다양한 Porting이 가능하며, 별도의 추가의 개발없이 사용할

수 있는 장점이 있다.

향후 연구에서는 Visual Basic언어 뿐만 아니라 Java나 C, C#과 같은 범용성이 있는 타 언어로의 변환개발이 필요할 것으로 사료된다.

XSLT의 개발중 ArdenML의 Schema의 견고함에 XSLT개발 작업이 잘 진행되었지만, 일부 Schema의 변경으로 좀더 쉬운 컴퓨터언어로의 변환이 가능함을 발견했다. 따라서 향후 ArdenML이 Arden syntax의 표준 XML로 정해지기 위해서는 이러한 Schema의 변경이 향후 연구되기를 바란다.

# VI. 결론

본 연구는 Arden Syntax가 HL7의 표준의 채택되어 다양한 연구와 개발을 통해 발전해가는 임상의사결정지원시스템의 Rule개발 언어로써 지식의 공유와 진료시점에서 사용될 수 있는 좋은 언어임에도 불구하고 Curly Brace문제와 컴파일러의 부재로인해 널리 사용되지 못하는 문제를 해결하고자 Arden Syntax의 XML버전인 ArdenML을 분석하고, 작성된 임상규칙(MLM)이 재사용되고 공유될 수 있도록 기간시스템에서 많이 사용하는 범용 컴퓨터언어(Visual Basic)로의 변환 XSLT를 개발하고, 다양한 임상규칙(MLM)중 고혈압관련 임상규칙을 B사의 OCS시스템과 연계하여 실행해 봄으로써 동작여부를 평가하였다.

XSLT개발을 위한 Arden Syntax와 Visual Basic언어의 문법을 분석하고, ArdenML의 Schema를 분석함으로써 향후 Arden Syntax가 XML로 진행되어야 하는 당위성을 알수 있었으며, 개발된 XSLT를 응용하면 다양한 컴퓨터언어(JAVA, C, C# 등)로의 변환이 가능함을 알수 있었다. 하지만 Arden Syntax는 발전해 가는 진행중인 언어이기 때문에 개발을 시작할 때 Version 2.5인 것이 현재는 Version 2.6이 발표되었다. 향후 Version 2.6에서 변경된 사항 및 추가사항을 반영해야 할것이며, 또한 ArdenML Schema역시 개발 진행중인 Schema이다. 좀더 컴퓨터언어로의 쉬운 변환을 위해서는 몇가지 수정되어야 할 부분이 있었지만, 연구의 목적상 Schema의 수정부분은 다른 연구를 통해서 좀 더 발전되기를 기대한다.

# 참고문헌

Hripcsak G, Clayton PD, Pryor TA, Haug P, Wigertz OB, Van der lei J. The Arden Syntax for medical logic modules. In Miller RA, editor. Proc. of the Fourteenth Annual Symposium on Computer Applications in Medical Care. New York: IEEE Computer Press, 1990; 200-204.

Ohno-Machado L, Gennari JH, Murphy SN et al. The GuideLine Interchange Format: a model for representing guidelines. JAMIA 1998; 5:357-372.

Jenders RA, Dasgupta B. Challenges in implementing a knowledge editor for the Arden Syntax: knowledge base maintenance and standardization of database linkages. Proc AMIA Symp 2002;:355-359.

Jenders, RA. HL7 Arden Syntax SIG Minutes May, 2001 [Web Page]. Available at http://www.hl7.org/library/committees/arden/minutes/ardensyntax-sig-minutes-0501.html.(Accessed 25 May 2004).

Jenders, RA. The Arden Syntax for Medical Logic Systems [Web Page]. Available at http://cslxinfmtcs.csmc.edu/hl7/arden/. (Accessed 12 June 2004).

Jadhav A, Sailors M. Structuring Healthcare Knowledge Bases: An Analysis of Explicit and Implicit Structures in Arden Syntax and an XML Schema Representation of Arden Syntax. Proc

AMIA Symp 2003; 875.

Karadimas HC, Chailloleau C, Hemery F, Simonnet J, Lepage E. Arden/J: An architecture for MLM execution on the Java platform. J Am Med Inform Assoc 2002; 9(4):359-68.

Kuhn RA, Reider RS. A C++ framework for developing Medical Logic Modules and an Arden Syntax compiler.Comput Biol Med 1994; 24(5):365-70.

Gao X, Johansson B, Shahsavar N, Arkad K, Ahlfeldt H, Wigertz O. Pre-compiling medical logic modules into C++ in building medical decision support systems. Comput Methods Programs Biomed 1993; 41(2):107-19.

Tran NK, Kost GJ. Worldwide point-of-care testing: compendiums of POCT for mobile, emergency, critical, and primary care and of infectious diseases tests. Point of Care: The Journal of Near-Patient Testing & Technology 2006;5:84-92

Sailors RM. Arden Syntax Markup Language ( or Arden Syntax: It's Not Just Text Any More! ). Proc AMIA Symp 2001.

The World Wide Web Consortium (W3C). Extensible Markup Language (XML) [Web Page]. Available at http://www.w3.org/XML/. (Accessed 12 June 2004).

HL7. HL7 Reference Information Model [Web Page]. Available at http://www.hl7.org/Library/data-model/RIM/modelpage_mem.htm. (Accessed 26 July 2004).

Peleg M, Boxwala AA, Bernstam E, Tu S, Greenes RA, Shortliffe EH. Sharable representation of clinical guidelines in GLIF:

relationship to the Arden Syntax. J Biomed Inform 2001; 34(3):170-81.

Hripcsak G, Cimino JJ, Johnson SB, Clayton PD. The Columbia-Presbyterian Medical Center decision-support system as a model for implementing the Arden Syntax. Proc Annu Symp Comput Appl Med Care 1991; 248-52.

# 부록1. ArdenMaintenanceForDotNET.XSL

```
<?xml version=" 1.0"  encoding=" UTF-8" ?>
<xsl:stylesheet      version="1.0"      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="Maintenance">
'[SlotName]: Maintenance ***********************************************
<br></br>
'[Title]    <xsl:value-of select="Title"/>
<br></br>
'[MLMName]          <xsl:value-of select="MLMName"/>
<br></br>
'[Arden]  <xsl:value-of select="Arden"/>
<br></br>
'[Version]             <xsl:value-of select="Version"/>
<br></br>
'[Institution]          <xsl:value-of select="Institution"/>
<br></br>
'[Author] <xsl:for-each select="Author/Person">
            <xsl:value-of select="FirstName"/>
            <xsl:choose>
            <xsl:when test="boolean(MiddleName)">
            <xsl:value-of select="MiddleName"/>
            <xsl:text>. </xsl:text>
            </xsl:when>
            <xsl:otherwise>
            <xsl:text> </xsl:text>
            </xsl:otherwise>
            </xsl:choose>
            <xsl:value-of select="SurName"/>
            <xsl:text>, </xsl:text>
            <xsl:for-each select="Degree">
            <xsl:value-of select="."/>
            <xsl:if test="position()!=last()">
                        <xsl:text>,</xsl:text>
            </xsl:if>
            </xsl:for-each>
            <xsl:apply-templates select="Contact"/>
            </xsl:for-each>
            <xsl:if test="not(boolean(Author/Person))">
                        <xsl:text></xsl:text>
            </xsl:if>
  <br></br>
'[Specialist]
            <xsl:for-each select="Specialist/Person">
            <xsl:value-of select="FirstName"/>
            <xsl:choose>
            <xsl:when test="boolean(MiddleName)">
                        <xsl:value-of select="MiddleName"/>
                        <xsl:text>. </xsl:text>
            </xsl:when>
            <xsl:otherwise>
```

```
                    <xsl:text> </xsl:text>
          </xsl:otherwise>
          </xsl:choose>
                    <xsl:value-of select="SurName"/>
                    <xsl:text>, </xsl:text>
                    <xsl:for-each select="Degree">
                              <xsl:value-of select="."/>
                    <xsl:if test="position()!=last()">
                              <xsl:text>,</xsl:text>
                    </xsl:if>
                    </xsl:for-each>
                    <xsl:apply-templates select="Contact"/>
                    </xsl:for-each>
                    <xsl:if test="not(boolean(Specialist/Person))">
                              <xsl:text>;;</xsl:text>
                    </xsl:if>
  <br></br>
'[Date]             <xsl:value-of select="Date"/>
                    <xsl:text></xsl:text>
<br></br>
'[Validation]       <xsl:value-of select="Validation"/>
                    <xsl:text></xsl:text>
<br></br>
          </xsl:template>
          <xsl:template match="Contact">
          <xsl:text>(</xsl:text>
          <xsl:for-each select="e-mail">
                    <xsl:element name="a">
          <xsl:attribute name="href">mailto:<xsl:value-of select="."/></xsl:attribute>
                    <xsl:value-of select="."/>
          </xsl:element>
          <xsl:if test="position()!=last()">
                    <xsl:text>, </xsl:text>
          </xsl:if>
          </xsl:for-each>
          <xsl:text>)</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

# 부록2. ArdenLibraryForDotNET.XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="Library">
'[SlotName]          Library ************************************************
<br></br>
'[Purpose]           <xsl:value-of select="Purpose"/>
<br></br>
'[Explanation]       <xsl:value-of select="Explanation"/>
<br></br>
'[Keywords]          <xsl:for-each select="Keywords/Keyword">
                     <xsl:value-of select="."/>
                     <xsl:text>, </xsl:text>
                     </xsl:for-each>
                     <xsl:if test="not(boolean(Keywords/Keyword))">
                     </xsl:if>
                     <xsl:for-each select="Citations/Citation">
                     <xsl:if test="position()=1">Citations:</xsl:if>
                              <xsl:value-of select="position()"/>.
                              <xsl:value-of select="CitationLevel"/>
                              <xsl:value-of select="CitationText"/>
                     </xsl:for-each>
         <br></br>
                     <xsl:if test="not(boolean(Citations/Citation))">
'[Citations]
                     </xsl:if>
         <br></br>
'[Links]    <xsl:for-each select="Links/Link">
                     <xsl:value-of select="LinkType"/>
                     <xsl:value-of select="LinkName"/>
                     <xsl:value-of select="LinkText"/>
                     </xsl:for-each>
                     <xsl:if test="not(boolean(Links/Link))">
                              <xsl:text>;;</xsl:text>
                     </xsl:if>
    <br></br>
          </xsl:template>
  <xsl:template name="LineFinish">
    <xsl:if test="position()!=last()">
      <xsl:text>; </xsl:text>
    </xsl:if>
    <xsl:if test="position()=last()">
      <xsl:text></xsl:text>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

# 부록3. ArdenKnowledgeForDotNET.XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo=
"http://www.w3.org/1999/XSL/Format">
<xsl:include href="ArdenKnowledgeExpressionForDotNET.xsl"/>
<!--******************************************************************************* -->
<!--      Knowledge slot main            -->
<!--******************************************************************************* -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************* -->
  <!-- ************************************************************************** -->
  <xsl:template match="Knowledge">
'[Knowledge]*************************************************
    <br></br>
    '[Type] <xsl:value-of select="Type"/>
    <br></br>
    '[Data]
    <br></br>
    <!-- Events Handler를 위한 Define -->
    <xsl:text>Dim Event_Arden As
New Arden_Delegate(AddressOf Event_Arden_Handler)</xsl:text>
    <br></br>
    <xsl:apply-templates select="Data"/>
    <br></br>
    <xsl:if test="boolean(Priority)">
    '[Priority]
    <br></br>
    <xsl:value-of select="Priority"/>
    <br></br>
    </xsl:if>
    '[Evoke]
    <br></br>
    '<xsl:apply-templates select="Evoke"/>
    '[Logic]
    <br></br>
    <xsl:apply-templates select="Logic"/>
    <br></br>
    <!-- 1. If-then-else|If-elseif-else statement -->
    <!-- <xsl:apply-templates select="Action"/> -->
    <!-- <br></br> -->
    <xsl:if test="boolean(Urgency)">
    '[Urgency]
    <xsl:value-of select="Urgency"/>
    <br></br>
    </xsl:if>
          </xsl:template>
<!--******************************************************************************* -->
<!--      Knowledge Slot Common Module          -->
<!--******************************************************************************* -->
 <!-- 1. If-then-else|If-elseif-else statement -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************* -->
<!-- ************************************************************************** -->
  <xsl:template match="If">
          <xsl:text>If </xsl:text>
          <xsl:apply-templates/>
          <xsl:text>End if</xsl:text>
    <br></br>
          </xsl:template>
  <xsl:template match="Condition">
          <xsl:if test="position()>2">
          <!-- more than 1 condition exists: if-elseif statement -->
          <xsl:text>Elseif </xsl:text>
          </xsl:if>
```

```
                    <xsl:apply-templates/>
                    <xsl:text> </xsl:text>
          </xsl:template>
          <xsl:template match="Then|Else">
                    <xsl:value-of select="name()"/>
  <br></br>
            <xsl:apply-templates/>
          </xsl:template>

          <!-- 2. Call statement -->
<!-- 2.1 Data/ Logic -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<xsl:template match="Data//Call|Logic//Call">
          <!-- syntax check -->
          <!-- identifier -->
          <xsl:choose>
          <xsl:when test="count(Identifier)=0">
                    <xsl:text>'[TODO] More than 1 identifier is needed.</xsl:text>
                    </xsl:when>
                    <xsl:when test="count(Identifier)=1">
                              <xsl:apply-templates select="Identifier"/>
          <!-- apply-templates is used to validate identifier against KEYWORDS -->
                              </xsl:when>
                              <xsl:otherwise>
                              <!-- more than 1 parameters are present -->
                                        <xsl:text>(</xsl:text>
                                        <!-- enclose identifiers with parenthesis -->
                                        <xsl:for-each select="Identifier">
                                                  <xsl:apply-templates select="."/>
          <!-- apply-templates is used to validate identifier against KEYWORDS -->
                                                  <xsl:if test="position()!=last()">
                                                            <xsl:text>,</xsl:text>
                                                  </xsl:if>
                                        </xsl:for-each>
                                        <xsl:text>)</xsl:text>
                              </xsl:otherwise>
                    </xsl:choose>
                    <xsl:text>= </xsl:text>
                    <xsl:value-of select="Name"/>
                    <xsl:text> (</xsl:text>
                    <xsl:apply-templates select="With"/>
  <xsl:text>)</xsl:text>
  <br></br>
          </xsl:template>

<!-- 2.2 Evoke -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<xsl:template match="Evoke/Call">
                    <xsl:text>Call </xsl:text>
          </xsl:template>

<!-- 2.3 Action -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<!-- ************************************************************************** -->
<xsl:template match="Action//Call">
                    <xsl:text>Call </xsl:text>
                    <xsl:value-of select="Name"/>
                    <xsl:text> </xsl:text>
                    <xsl:apply-templates select="With"/>
                    <xsl:apply-templates select="Delay"/>
          </xsl:template>
```

```
  <!-- 2.4 Action -->
  <!-- ************************************************************************ -->
  <!-- ************************************************************************ -->
  <!-- ************************************************************************ -->
          <xsl:template match="Delay">
                    <xsl:text> Delay </xsl:text>
                    <xsl:apply-templates/>
          </xsl:template>


  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="Call/With">
                    <xsl:for-each select="*">
                              <xsl:apply-templates select="."/>
                              <xsl:if test="position()!=last()">
                                        <xsl:text>,</xsl:text>
                              </xsl:if>
                    </xsl:for-each>
          </xsl:template>


          <!-- 3. For statement -->
  <!-- ************************************************************************ -->
  <!-- ************************************************************************ -->
  <!-- ************************************************************************ -->
          <xsl:template match="For">
                    <xsl:text>For </xsl:text>
                    <xsl:value-of select="Identifier"/>
                    <xsl:text> In </xsl:text>
                    <xsl:apply-templates select="*[position()=2]"/>
                    <xsl:text> Do</xsl:text>
                    <br/>
                    <xsl:apply-templates select="Do" mode="For"/>
                    <xsl:text>Enddo</xsl:text>
          </xsl:template>
          <xsl:template match="Do" mode="For">
                    <xsl:apply-templates/>
          </xsl:template>

  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<!-- 4. Assignment statement -->
  <xsl:template match="Assignment">
    <xsl:apply-templates select="*[1]"/>
    <xsl:text> = </xsl:text>
    <xsl:apply-templates select="*[2]"/>
    <br></br>
  </xsl:template>


  <!-- 5. Comment statement -->
  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
          <xsl:template match="//comment()">
                    <!--<xsl:text>'</xsl:text>
                    <xsl:value-of select="."/>
                    <xsl:text>'</xsl:text>
    <xsl:if test="not(contains(parent::If,'If'))">
      <br></br>
    </xsl:if>-->
  </xsl:template>
          <!-- 6. Basic elements -->
```

```
            <!-- 6.1 Value -->
   <!-- ************************************************************************** -->
   <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
   <!-- ************************************************************************** -->
<xsl:template match="Value">
                        <!-- Syntax check -->
                        <xsl:if test="@otype='duration'">
            <xsl:if test="not(contains($DurationType,concat('|',@unit,'|')))">
                                    <font class="Error">
                                              <xsl:text>duration type value "</xsl:text>
                                              <xsl:value-of select="."/>
                                              <xsl:text>"   requires one of </xsl:text>
                                              <xsl:value-of select="$DurationType"/>
                                    </font>
                                 </xsl:if>
                        </xsl:if>
                        <!-- Presentation -->
                        <xsl:choose>
                                 <xsl:when test="@otype='string'">
                                    <xsl:text>"</xsl:text>
                                    <xsl:value-of select="."/>
                                    <xsl:text>"</xsl:text>
                                 </xsl:when>
                                 <xsl:when test="@otype='null'">
         <xsl:text> Nothing </xsl:text>
                                 </xsl:when>
       <xsl:when test="@otype='duration'">
         <xsl:value-of select="@unit"/>
         <xsl:value-of select="@otype"/>
         <xsl:text>_Arden(</xsl:text>
         <xsl:value-of select="."/>
         <xsl:text>)</xsl:text>
       </xsl:when>
       <xsl:otherwise>
         <xsl:value-of select="."/>
                                 </xsl:otherwise>
                        </xsl:choose>
            </xsl:template>


            <!-- 6.2 Identifier -->
   <!-- ************************************************************************** -->
   <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
   <!-- ************************************************************************** -->
<xsl:template match="Identifier">
     <!-- Syntax check   -->
     <!-- Keyword check -->
     <xsl:call-template name="KeywordCheck">
       <xsl:with-param name="Identifier" select="text()"/>
     </xsl:call-template>
     <xsl:choose>
       <xsl:when test="contains(text(),'They')">
         <!-- Special purposes -->
         <xsl:value-of select="../../Identifier[text()]"/>.Value
       </xsl:when>
       <xsl:when test=".='ESCn'">
         <!-- Treat CR and LF -->
         <xsl:text>"\n"</xsl:text>
       </xsl:when>
       <xsl:when test="boolean(@index)">
         <xsl:value-of select="."/>
         <xsl:text>[</xsl:text>
         <xsl:value-of select="@index"/>
         <xsl:text>]</xsl:text>
       </xsl:when>
       <xsl:when test="contains(name(following::*),'List')">
         <!-- Special purposes -->
```

```
        <xsl:value-of select="."/>.Values
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="."/>.Value
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <!-- 6.3 List -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template match="List">
                    <xsl:text>New String() {</xsl:text>
                    <xsl:for-each select="*">
                              <xsl:apply-templates select="."/>
                              <xsl:if test="position()!=last()">
                                        <xsl:text>,</xsl:text>
                              </xsl:if>
                    </xsl:for-each>
                    <xsl:text>}</xsl:text>
          </xsl:template>
<!--*****************************************************************************-->
<!--                  Data slot                        -->
<!--*****************************************************************************-->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template match="Read">
                    <!-- 변수에 관한 처리 -->
                    <xsl:choose>
                              <!-- 변수가 (1개)밖에 없으면... -->
                              <xsl:when test="count(Identifier)=1">
                                        <xsl:apply-templates select="Identifier"/>
                                        <!-- 혹시 몰라서 Value-of를 사용하지 낳고 apply-
templates를 사용한다. -->
        <xsl:text>= Read</xsl:text>
        <!-- 집계해서 읽는다면.... -->
        <xsl:apply-templates
select="Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest"/>
<xsl:if
test="not(boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest))">
          <xsl:text>(</xsl:text>
        </xsl:if>
        <!-- for version compatibility -->
        "
        <xsl:value-of select="normalize-space(translate(Mapping, '&quot;', ''))" />
        "
        <xsl:if
test="not(boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest))">
          <xsl:text>)</xsl:text>
        </xsl:if>
        <xsl:apply-templates select="Where"/>
<xsl:if test="boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest)">
          <xsl:text>)</xsl:text>
        </xsl:if>
        <br></br>
                    </xsl:when>
<!-- 담을 변수가 (2개)이상 이면... Array_Arden를 통해서 ArrayList를 사용해야 한다.-->
          <xsl:otherwise>
          <xsl:text>Array_Arden </xsl:text>
        <xsl:text>= Read</xsl:text>
        <!-- Aggregation -->
        <xsl:apply-templates
select="Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest"/>
        <!-- for version compatibility -->
        <xsl:if
```

```
test="boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest)
 and name(*[last()])='Where'">
            <xsl:text>(</xsl:text>
          </xsl:if>
          <!-- for version compatibility -->
          <xsl:if
test="not(boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest))">
            <xsl:text>(</xsl:text>
          </xsl:if>
          "<xsl:value-of select="normalize-space(translate(Mapping, '&quot;', ''))" />"
          <xsl:if
test="not(boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest))">
            <xsl:text>)</xsl:text>
          </xsl:if>
          <xsl:apply-templates select="Where"/>
          <xsl:if
test="boolean(Exist|Sum|Average|Minimum|Maximum|Last|First|Earliest|Latest)
 and name(*[last()])='Where'">
            <xsl:text>)</xsl:text>
          </xsl:if>
          <xsl:text>)</xsl:text>
          <br></br>
          <xsl:for-each select="Identifier">
                                                    <xsl:apply-templates select="."/>
            <xsl:text> = Array_Arden(</xsl:text>
            <xsl:number format="0"/>
            <xsl:text> )</xsl:text>
            <br></br>
          </xsl:for-each>
        </xsl:otherwise>
                    </xsl:choose>
    <br></br>
  </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
            <xsl:template match="Read/Exist|Read/Sum|Read/Average|Read/Avg">
                    <xsl:value-of select="name()"/>
    <!-- Read문에서 괄호 시작을 안해준다. -->
                    <xsl:text>(</xsl:text>
            </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
            <xsl:template
match="Read/Minimum|Read/Maximum|Read/Last|Read/First|Read/Earliest|Read/Latest">
                    <xsl:value-of select="name()"/>
    <!-- Read에서 괄호를 안열어주기로 했으니...무조건 연다... -->
    <xsl:text>(</xsl:text>
    <xsl:if test="boolean(text())">
      <xsl:value-of select="."/>
                            <xsl:text>, </xsl:text>
                    </xsl:if>
  </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
            <xsl:template match="Read/Where">
    <xsl:text>, ( </xsl:text>
    <xsl:apply-templates/>
    <xsl:text>) </xsl:text>
```

```
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      Data문에 있는것들은 별도의 _Arden함수로 모두 분리한다.
  -->
  <!-- ************************************************************************** -->
  <!-- End Resolve Aggregation in Read statement -->
  <!-- Begin Resolve Mappings -->
  <xsl:template
match="Data/Event|Data/MLM|Data/Message|Data/Destination|Data/Interface">
    <xsl:if test="not(name()='Event')">
      <xsl:apply-templates select="Identifier"/>
      <!-- use apply-templates instead of value-of to validate identifier against Keywords -
->
      <xsl:text> = </xsl:text>
      <xsl:value-of select="name()"/>
      <xsl:text></xsl:text>
    </xsl:if>
    <xsl:if test="name()='Event'">
      <xsl:value-of select="name()"/>
    </xsl:if>
    <xsl:choose>
      <xsl:when test="name()='MLM'">
        <xsl:text>_Arden("</xsl:text>
        <xsl:value-of select="normalize-space(translate(Mapping, '&quot;', ''))"/>
        <xsl:text>")</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text>_Arden("</xsl:text>
        <xsl:value-of select="normalize-space(translate(Mapping, '&quot;', ''))"/>
        <xsl:text>")</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
    <br></br>
  </xsl:template>

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
          <xsl:template match="Argument">
                    <!-- Identifier treatment -->
    Dim sArgumentArray() As String = Environment.GetCommandLineArgs
    <xsl:choose>
                              <!-- when only 1 identifier exists -->
                              <xsl:when test="count(Identifier)=1">
        <xsl:apply-templates select="Identifier"/> = sArgumentArray(0)
        <!-- use apply-templates instead of value-of to validate identifier against Keywords
-->
                              </xsl:when>
                              <!-- when more than 1 identifier exists -->
                              <xsl:otherwise>
        <xsl:for-each select="Identifier">
          <xsl:text> = sArgumentArray(</xsl:text>
          <xsl:number format="0"/>
          <xsl:text> )</xsl:text>
          <xsl:apply-templates select="."/>
                                        <xsl:if test="position()!=last()">
            <br></br>
                                        </xsl:if>
                                    </xsl:for-each>
                              </xsl:otherwise>
                    </xsl:choose>
          </xsl:template>
```

```
<!--********************************************************************************** -->
<!--                    Evoke slot                                    -->
<!--********************************************************************************** -->
<!-- Needs more works -->
<!--
          <xsl:template match="Evoke/Apply">
                    <xsl:apply-templates/>
                    <xsl:text>;</xsl:text>
                    <br/>
          </xsl:template>
-->

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
          <xsl:template match="Evoke/Identifier">
                    <!-- syntex check: identifer should be defined in EVENT slot -->
                    <xsl:call-template name="DefinitionCheck">
          <xsl:with-param name="LocationOfDefinition" select="//Event/Identifier"/>
                              <xsl:with-param name="ValueOfIdentifier" select="."/>
                    </xsl:call-template>
                    <!-- presentation -->
                    '<xsl:value-of select="."/>
          <!-- use value-of instead of apply-templates because the identifier was already
validated in data slot -->
                    <xsl:text></xsl:text>
                    <br/>
          </xsl:template>

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
          <xsl:template match="Evoke/Value">
                    <!-- syntex check: type should be time -->
                    <xsl:if test="/@type!='time'">
                              <font class="Error">
                                        <xsl:text> type type is required</xsl:text>
                              </font>
                    </xsl:if>
                    <!-- presentation -->
                    <xsl:value-of select="."/>
                    <xsl:apply-templates/>
                    <xsl:text></xsl:text>
                    <br/>
          </xsl:template>

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template match="PeriodicTrigger">
    <xsl:text>'Every </xsl:text>
    <xsl:apply-templates select="Every"/>
    <xsl:text> For </xsl:text>
    <xsl:apply-templates select="For/*"/>
    <xsl:apply-templates select="Starting"/>
    <xsl:apply-templates select="Until"/>
    <xsl:text></xsl:text>
    <br/>
  </xsl:template>

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template match="PeriodicTrigger/Starting">
                    <xsl:text>' Starting </xsl:text>
                    <xsl:apply-templates/>
```

```
            </xsl:template>
            <xsl:template match="PeriodicTrigger/Until">
                        <xsl:text> Until </xsl:text>
                        <xsl:apply-templates/>
            </xsl:template>

  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="PeriodicTrigger/Starting//Identifier">
    <!-- syntex check: identifer should be defined in EVENT slot -->
    <xsl:call-template name="DefinitionCheck">
      <xsl:with-param name="LocationOfDefinition" select="//Event/Identifier"/>
      <xsl:with-param name="ValueOfIdentifier" select="."/>
    </xsl:call-template>
    <!-- presentation -->
    <xsl:value-of select="."/>
  </xsl:template>


<!--*************************************************************************** -->
<!--                    Logic slot                          -->
  <!--      Expressions are resolved in ArdenKnowledgeExpression.xsl                  -->
<!--*************************************************************************** -->
  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="Conclude">
                        <xsl:text>Arden_Conclude </xsl:text>
                        <xsl:apply-templates/>
                        <xsl:text></xsl:text>
                        <br/>
            </xsl:template>
<!--*************************************************************************** -->
<!--      Action slot                                 -->
<!--*************************************************************************** -->
<!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="Write">
    <xsl:text>Arden_Result.Value = </xsl:text>
    <xsl:apply-templates/>
    <br></br>
    <xsl:text>If bPrmBoolean Then </xsl:text>
    <br></br>
    <xsl:text>Console.WriteLine(Arden_Result) </xsl:text>
    <br></br>
    <xsl:text>End If </xsl:text>
    <br></br>
  </xsl:template>

  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="Action">
    <xsl:text>Private Sub Arden_Conclude(byVal bPrmBoolean as Boolean)</xsl:text>
    <br></br>
    <xsl:apply-templates/>
    <br></br>
    <xsl:text>End Sub </xsl:text>
    <br></br>
  </xsl:template>

  <!--
  <xsl:template match="Write">
    <xsl:text>Console.WriteLine("</xsl:text>
    <xsl:apply-templates/>
```

```
    <xsl:text>")</xsl:text>
    <br></br>
  </xsl:template>
  -->

  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
<xsl:template match="Return">
    <xsl:text>Return </xsl:text>
    (<xsl:apply-templates/>)<br/>
  </xsl:template>


  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
          <xsl:template match="Write/Destination">
                    <!-- syntex check: identifer should be defined in EVENT slot -->
                    <xsl:call-template name="DefinitionCheck">
          <xsl:with-param name="LocationOfDefinition" select="//Destination/Identifier"/>
                    <xsl:with-param name="ValueOfIdentifier" select="."/>
                    </xsl:call-template>
                    <!-- presentation -->
    <br></br>
                    <xsl:text> SendEmail_Arden(</xsl:text>
                    <xsl:value-of select="."/>
    <xsl:text> )</xsl:text>
          </xsl:template>


<!--*****************************************************************************-->
<!--          Syntax Check Common Module          -->
<!--*****************************************************************************-->
          <!-- 1. Identifier-->
          <!-- Keyword declaration -->
  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
          <xsl:variable name="ReservedWords">|ABS|ACTION|AFTER|AGO|ALERT|ALL|
AND|ANY|ARCCOS|ARCSIN|ARCTAN|ARDEN|ARE|ARGUMENT|AS|AT|AUTHOR|
AVERAGE|AVG|BE|BEFORE|BOOLEAN|CALL|CEILING|CHARACTERS|CITATIONS|
          CONCLUDE|COS|COSINE|COUNT|DATA|DATA_DRIVEN|DATA-DRIVEN|
DATE|DAY|DAYS|DECREASE|DELAY|DESTINATION|DO|DURATION|EARLIEST|
ELSE|ELSEIF|ENDDO|ENDIF|END|EQ|EQUAL|EVENT|EVERY|EVOKE|EXIST|EXISTS|
EXP|EXPIRED|EXPLANATION|EXTRACT|FALSE|FILENAME|FIND|FIRST|
FLOOR|FOLLOWING|FOR|FORMATTED|FROM|GE|GREATER|GT|HOUR|HOURS|
IF|IN|INCREASE|INDEX|INSTITUTION|INT|INTERFACE|INTERVAL|IS|
KEYWORDS|KNOWLEDGE|LAST|LATEST|LE|LEFT|LENGTH|LESS|LET|
LIBRARY|LINKS|LIST|LOG|LOG10|LOGIC|LOWERCASE|LT|MAINTENANCE|
MATCHES|MAX|MAXIMUM|MEDIAN|MERGE|MESSAGE|MIN|MINIMUM|
MINUTE|MINUTES|MLM|MLMNAME|MLM_SELF|MONTH|MONTHS|
NE|NEAREST|NO|NOT|NULL|NUMBER|OCCUR|OCCURRED|OCCURS|
OF|OR|PAST|PATTERN|PERCENT|PRECEDING|PRESENT|PRIORITY|PRODUCTION|
PURPOSE|READ|REFUTE|RESEARCH|RETURN|REVERSE|RIGHT|ROUND|SAME|
SECOND|SECONDS|SEQTO|SIN|SINE|SLOPE|SORT|SPECIALIST|SQRT|STARTING|
STDDEV|STRING|SUBSTRING|SUM|SUPPORT|SURROUNDING|TAN|TANGENT|
TESTING|THAN|THE|THEN|TIME|TITLE|TO|TRIM|TRUE|TRUNCATE|TYPE|
UNIQUE|UNTIL|UPPERCASE|URGENCY|VALIDATION|VARIANCE|VERSION|WAS|
WEEK|WEEKS|WERE|WHERE|WHILE|WITH|WITHIN|WRITE|YEAR|YEARS|UNION|
INTERSECT|EXCLUDING|CITATION|SELECT|</xsl:variable>
<!-- now, currenttime, eventtime, triggertime, it, and they  are used as special identifiers. -
->
          <xsl:variable name="DurationType">|year|years|month|months|week|weeks|day|
days|hour|hours|minute|minutes|second|seconds|</xsl:variable>
          <!-- 1. Identifier : reserved words are not allowed to use as identifiers-->
          <xsl:template name="KeywordCheck">
```

```
                    <!-- Syntax check to see if   identifier is in the keywords. -->
                    <xsl:param name="Identifier"/>
                    <!-- Variable ReservedWords is defined in Arden 2.1.7.xsl -->
                    <xsl:if test="contains($ReservedWords,concat('|',translate($Identifier,
'abcdefghijklmnopqrstuvwxyz_',
'ABCDEFGHIJKLMNOPQRSTUVWXYZ_'),'|'))">
                              <font class="Error">
                                        <xsl:text>Keyword "</xsl:text>
                                        <xsl:value-of select="."/>
                                        <xsl:text>"   is not allowed to use.</xsl:text>
                              </font>
                    </xsl:if>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<!-- 1.2 Call/Name, Evoke//Identifier, Write/Destination should be mapped in data slot -->
          <xsl:template name="DefinitionCheck">
                    <xsl:param name="LocationOfDefinition"/>
                    <!-- Where the identifers are defined -->
                    <xsl:param name="ValueOfIdentifier"/>
                    <!-- make a series of identifiers to a string delimited with "|"-->
                    <xsl:variable name="ListOfIdentifiers">
                              <xsl:text>, </xsl:text>
                              <xsl:for-each select="$LocationOfDefinition">
                                        <xsl:value-of select="."/>
                                        <xsl:text>, </xsl:text>
                              </xsl:for-each>
                    </xsl:variable>
                    <xsl:if
test="not(contains($ListOfIdentifiers,concat('|',$ValueOfIdentifier,'|')))">
                              <!-- when the list of identifiers does not include the identier in
testing, it is an error -->
                              <font class="Error">
                                        <xsl:text>'Identifier "</xsl:text>
                                        <xsl:value-of select="."/>
                                        <xsl:text>" is not defined in </xsl:text>
                                        <xsl:value-of
select="name($LocationOfDefinition/parent::*)"/>
                                        <xsl:text> statements.</xsl:text>
                              </font>
                    </xsl:if>
          </xsl:template>


          <!-- 2. Check number of arguments -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template name="CheckNumberOfArguments">
                    <xsl:param name="NumberOfArguments"/>
                    <font class="Error">
                              <xsl:choose>
                                        <xsl:when test="count(following-sibling::*)=0">
                              <xsl:text>'No argument found in "</xsl:text>
                              <xsl:value-of select="name()"/>
                                        <xsl:text>"</xsl:text>
                              </xsl:when>
                    <xsl:when test="count(following-sibling::*)&lt;$NumberOfArguments">
                              <xsl:text>'Too few arguments in "</xsl:text>
                                        <xsl:value-of select="name()"/>
                                        <xsl:text>"</xsl:text>
                              </xsl:when>
                    <xsl:when test="count(following-sibling::*)>$NumberOfArguments">
                                        <xsl:text>'Too many arguments in "</xsl:text>
```

```
                                                    <xsl:value-of select="name()"/>
                                                    <xsl:text>"</xsl:text>
                                            </xsl:when>
                                    </xsl:choose>
                            </font>
                    </xsl:template>


<!--************************************************************************** -->
<!--                    Variable Define templates        -->
<!--************************************************************************** -->
<!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
<xsl:template name="SelectAllVairable">
    <br></br>
    <xsl:for-each select="//Identifier">
      <xsl:sort/>
        <xsl:if test="not(.=preceding::Identifier)">
        <xsl:if test="contains(.,text())">
          <xsl:text>Private </xsl:text>
          <xsl:value-of select="."/>
          <xsl:text> As ArdenVariable</xsl:text>
          <br></br>
        </xsl:if>
      </xsl:if>
    </xsl:for-each>
    <xsl:text>Private Arden_Result As ArdenVariable</xsl:text>
    <br></br>
    <xsl:text>Private Array_Arden As New ArrayList</xsl:text>
    <br></br>
    <xsl:text>Delegate Function Arden_Delegate(ByVal sPrmString As String) As
String</xsl:text>
  </xsl:template>

<!--************************************************************************** -->
<!--      Utility templates      -->
<!--************************************************************************** -->
</xsl:stylesheet>
```

# 부록4. ArdenKnowledgeExpressionForDotNET.XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
        <!-- depricated
        <xsl:template match="Parenthesis">
                <xsl:text>(</xsl:text>
                <xsl:apply-templates/>
                <xsl:text>)</xsl:text>
        </xsl:template>
-->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      대부분 Child Note가 하나짜리인 함수들....
  -->
        <xsl:template match="Apply|Is|Was|Are|Were|Occur|Occurs|Occurred
|IsNot|WasNot|AreNot|WereNot|OccurNot|OccursNot|OccurredNot">
                <xsl:apply-templates select="*[1]"/>
                <!-- transfer control of transformation over operators -->
                <!-- Apply, Is, Occur -->
        </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************************************************************************** -->
  <!-- ************************************************************************** -->
        <!--9.2 List Operators -->
        <!--
9.2.1   , (binary, left associative)
9.2.2   , (unary, non-associative)
9.2.3   Merge (binary, left-associative)
9.2.4   Sort (unary, non-associative)
-->


  <!-- ************************************************************************** -->
  <!-- ************************************************************************** -->
  <!-- ************************************************************************** -->
        <xsl:template match="AddList">
                <!-- has 0 - infinitive number of arguments -->
                <xsl:for-each select="following-sibling::*">
                        <xsl:apply-templates select="."/>
                        <xsl:if test="position()!=last()">
                                <xsl:text>,</xsl:text>
                        </xsl:if>
                </xsl:for-each>
        </xsl:template>

  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      주어진 2개의 목록을 하나의 목록으로 합쳐서 Return해주는 함수
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Merge">
    <xsl:value-of select="name()"/>
    <xsl:text>_Arden(</xsl:text>
                <xsl:if test="name(following-sibling::*[1])!='Identifier'">
                        <xsl:text>(</xsl:text>
                </xsl:if>
                <xsl:apply-templates select="following-sibling::*[1]"/>
```

```
                    <xsl:if test="name(following-sibling::*[1])!='Identifier'">
                              <xsl:text>)</xsl:text>
                    </xsl:if>
                    <xsl:text>, </xsl:text>
                    <xsl:if test="name(following-sibling::*[2])!='Identifier'">
                              <xsl:text>(</xsl:text>
                    </xsl:if>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
                    <xsl:if test="name(following-sibling::*[2])!='Identifier'">
                              <xsl:text>)</xsl:text>
                    </xsl:if>
  <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    주어진 범위에서 해당하는 값들을 일련하게 발생시켜서
    Array로 넘겨주는 함수
-->
<!-- ************************************************************************** -->
<xsl:template match="SeqTo">
  <xsl:value-of select="name()"/>
  <xsl:text>_Arden(</xsl:text>
  <xsl:apply-templates select="following-sibling::*[1]"/>
  <xsl:text>, </xsl:text>
  <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>) </xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    주어진 목록을 기준에 따라서 Sort한 값을 Return한다.
    '[TODO]용예를 확인해야 함.
-->
<!-- ************************************************************************** -->
<xsl:template match="Sort">
                    <xsl:text> </xsl:text>
  <xsl:text>Sort_Arden(</xsl:text>
  <xsl:value-of select="Sort"/>
  <xsl:value-of select="@order"/>
  <xsl:text>, </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[1]"/>
  <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    기본적으로 Assigne할 변수의 값에대해서 조건문이 True일때...
-->
<!-- ************************************************************************** -->
<!--9.3 Where Operator      -->
        <xsl:template match="Where">
  <xsl:text> Where_Arden(</xsl:text>
                    <xsl:if test="name(*[1])!='Identifier'">
                              <xsl:text> (</xsl:text>
                    </xsl:if>
  <xsl:apply-templates select="*[1]"/>
                    <xsl:if test="name(*[1])!='Identifier'">
```

```
                                <xsl:text>) </xsl:text>
                        </xsl:if>
  <xsl:text> , </xsl:text>
                        <!-- logic expressions -->
                        <xsl:apply-templates select="*[2]"/>
  <xsl:text>)</xsl:text>
</xsl:template>


        <!--9.4 Logical Operators -->
<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    VB.NET에서의 And / Or와 동일함.
-->
<!-- ************************************************************************** -->
<xsl:template match="And|Or">
                        <xsl:variable name="Operator" select="name()"/>
                        <xsl:for-each select="following-sibling::*">
                                <xsl:apply-templates select="."/>
                                <xsl:if test="position()!=last()">
                                        <xsl:text> </xsl:text>
                                        <xsl:value-of select="$Operator"/>
                                        <xsl:text> </xsl:text>
                                </xsl:if>
                        </xsl:for-each>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    VB.NET과 동일하지만 위치가 틀리다....
-->
<!-- ************************************************************************** -->
<xsl:template match="Not">
                        <xsl:text> Not </xsl:text>
                        <xsl:apply-templates select="following-sibling::*"/>
        </xsl:template>


<!--9.5 Simple Comparison Operators : higher priority than IsComparison operators
with the same names-->
<!-- They have the same element name as the operators in IS statement. Their priority
is higher than that of IS statement-->
<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    좌측과 우측이 같다....Equal
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/EQ">
                        <!-- syntax check for number of arguments -->
                        <xsl:call-template name="CheckNumberOfArguments">
                        <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                        </xsl:call-template>
                        <!-- presentation -->
                        <xsl:apply-templates select="following-sibling::*[1]"/>
                        <xsl:text> = </xsl:text>
                        <xsl:apply-templates select="following-sibling::*[2]"/>
        </xsl:template>


<!-- ************************************************************************** -->
```

```
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    좌측과 우측이 같지 않다....NOT Equal .... <>
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/NEQ">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> &lt;&gt; </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    좌측이 우측보다 크다....    >
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/GT">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> &gt; </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    좌측이 우측보다 작다....    <
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/LT">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> &lt; </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    좌측이 우측보다 크거나 같다....    >=
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/GE">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
```

```
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> &gt;= </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      좌측이 우측보다 작거나 같다....   <=
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/LE">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> &lt;= </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
          </xsl:template>
          <!-- 9.6 Is Comparison Operators and  -->


  <!-- 9.7 Occur Comparison Operators -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      단항 연산자....
  -->
  <!-- ************************************************************************** -->
  <!--Unary expressions -->
          <xsl:template
match="Present|Null|Boolean|Number|String|Is/TimeType|Duration|
Is/ListType|IsNot/TimeType|IsNot/ListType">
<xsl:if test="not(contains(name(parent::*),'Are')) and not(contains(name(parent::*),'Is'))">
      <xsl:apply-templates select="following-sibling::*[1]"/>
    </xsl:if>
    <!--Verbe with/without nigation-->
    <xsl:choose>
          <xsl:when test="contains(name(parent::*),'Not')">
                    <xsl:text> </xsl:text>
                    <xsl:value-of select="substring-before(name(parent::*),'Not')"/>
                    <xsl:text>Not</xsl:text>
                    </xsl:when>
          <xsl:otherwise>
                    <xsl:value-of select="name(parent::*)"/>
                    </xsl:otherwise>
                    </xsl:choose>
                    <xsl:choose>
                    <xsl:when test="contains(name(),'Type')">
                              <xsl:value-of select="substring-before(name(),'Type')"/>
                    </xsl:when>
      <xsl:when test="contains(name(),'Null')">
        <xsl:text>Nothing</xsl:text>
        <xsl:text>(</xsl:text>
        <xsl:if test="contains(name(parent::*),'Are') or contains(name(parent::*),'Is')">
          <xsl:apply-templates select="following-sibling::*[1]"/>
        </xsl:if>
        <xsl:text>)</xsl:text>
      </xsl:when>
      <xsl:otherwise>
```

```
        <xsl:value-of select="name()"/>
        <xsl:text>(</xsl:text>
        <xsl:if test="contains(name(parent::*),'Are') or contains(name(parent::*),'Is')">
          <xsl:apply-templates select="following-sibling::*[1]"/>
        </xsl:if>
        <xsl:text>)</xsl:text>
      </xsl:otherwise>
                    </xsl:choose>
          </xsl:template>


<!--Binary expressions : lower priority than the same operators after 'Apply'-->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      같다...
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="EQ">
                    <xsl:call-template name="IsBinaryComparison">
                              <xsl:with-param name="Operator">=</xsl:with-param>
                    </xsl:call-template>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      작다...기호로 표현되어야 할때가 있고...함수로 표현되어야 할때가
      있는데....일단....함수로 표현해서...Parameter로 넘긴다.
      해당 함수는 IsBinaryComparison에서 Parameter로 처리한다.
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="LT">
                    <xsl:call-template name="IsBinaryComparison">
                    <xsl:with-param name="Operator">LessThan_Arden</xsl:with-param>
                    </xsl:call-template>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      크다...기호로 표현되어야 할때가 있고...함수로 표현되어야 할때가
      있는데....일단....함수로 표현해서...Parameter로 넘긴다.
      해당 함수는 IsBinaryComparison에서 Parameter로 처리한다.
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="GT">
                    <xsl:call-template name="IsBinaryComparison">
                    <xsl:with-param name="Operator">GreaterThan_Arden</xsl:with-param>
                    </xsl:call-template>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      작거나 같다....기호로 표현되어야 할때가 있고...함수로 표현되어야 할때가
      있는데....일단....함수로 표현해서...Parameter로 넘긴다.
      해당 함수는 IsBinaryComparison에서 Parameter로 처리한다.
  -->
  <!-- ************************************************************************** -->
```

```
<xsl:template match="LE">
                    <xsl:call-template name="IsBinaryComparison">
                    <xsl:with-param       name="Operator">LessThanEqual_Arden</xsl:with-
param>
                    </xsl:call-template>
          </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
     크거나 같다....기호로 표현되어야 할때가 있고...함수로 표현되어야 할때가
     있는데....일단....함수로 표현해서...Parameter로 넘긴다.
     해당 함수는 IsBinaryComparison에서 Parameter로 처리한다.
-->
<!-- ************************************************************************** -->
<xsl:template match="GE">
          <xsl:call-template name="IsBinaryComparison">
          <xsl:with-param name="Operator">GreaterThanEqual_Arden</xsl:with-param>
          </xsl:call-template>
</xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
  GE, LE, GT, LT에 해당하는 함수를 별도로 만든다.
-->
<!-- ************************************************************************** -->
<xsl:template name="IsBinaryComparison">
  <xsl:param name="Operator"/>
  <!-- syntax check for number of arguments -->
  <xsl:call-template name="CheckNumberOfArguments">
    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
  </xsl:call-template>
  <!-- presentation -->
  <!--Verbe with/without nigation-->
  <xsl:choose>
    <xsl:when test="contains(name(parent::*),'Not')">
      <xsl:value-of select="substring-before(name(parent::*),'Not')"/>
      <xsl:text>Not</xsl:text>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="name(parent::*)"/>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:value-of select="$Operator"/>
  <xsl:text>(</xsl:text>
  <!-- for is/occur operators -->
  <xsl:apply-templates select="following-sibling::*[1]"/>
  <!-- it's is/occur operator -->
  <xsl:text> , </xsl:text>
  <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>)</xsl:text>
</xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
  Duration After/Before Standard-Date ==> 기준일자로 부터 Duration간의 날짜이동
  계산한 날짜를 넘겨준다.
  -->
<!-- ************************************************************************** -->
```

```
<xsl:template match="After|Before">
        <!-- it's is operator -->
        <!-- syntax check for number of arguments -->
        <xsl:call-template name="CheckNumberOfArguments">
                <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
        </xsl:call-template>
        <!--Verbe with/without nigation-->
        <xsl:choose>
                <xsl:when test="contains(name(parent::*),'Not')">
                <xsl:value-of select="substring-before(name(parent::*),'Not')"/>
                <xsl:text>Not</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                        <xsl:value-of select="name(parent::*)"/>
                </xsl:otherwise>
        </xsl:choose>
        <!-- Preposition -->
        <xsl:value-of select="name()"/>
        <xsl:text>_Arden(</xsl:text>
  <!--Subject-->
  <xsl:apply-templates select="following-sibling::*[1]"/>
  <xsl:text>, </xsl:text>
  <!--Objective-->
                <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    해당 값이 원하는 변수 목록 또는 기간안에 포함이 되는지 여부를
    판단하는 함수.. -->
<!-- ************************************************************************** -->
        <xsl:template match="In">
        <!-- it's is operator -->
        <!-- syntax check for number of arguments -->
        <xsl:call-template name="CheckNumberOfArguments">
                <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
        </xsl:call-template>
        <!--Verbe with/without nigation-->
        <xsl:choose>
                <xsl:when test="contains(name(parent::*),'Not')">
                <xsl:value-of select="substring-before(name(parent::*),'Not')"/>
                        <xsl:text>Not</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                        <xsl:value-of select="name(parent::*)"/>
                </xsl:otherwise>
        </xsl:choose>
        <!-- Preposition -->
        <xsl:text>In(</xsl:text>
        <!--Objective-->
        <xsl:choose>
                <xsl:when test="name(following-sibling::*[2])!='Identifier'">
                        <xsl:text>(</xsl:text>
                </xsl:when>
                <xsl:otherwise>
                </xsl:otherwise>
        </xsl:choose>
  <!--Subject-->
  <xsl:apply-templates select="following-sibling::*[1]"/>
  <xsl:text>, </xsl:text>
  <xsl:apply-templates select="following-sibling::*[2]"/>
                <xsl:if test="name(following-sibling::*[2])!='Identifier'">
                        <xsl:text>)</xsl:text>
```

```
                    </xsl:if>
    <xsl:text>)</xsl:text>
          </xsl:template>

  <!-- Binary expressions -->
  <!-- ************************************************************************ -->
  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
  <!--
      해당하는 값이 Exp2에 해당하는 기간후의 기간범위 안에 들어가는
      지를 판단하는 Boolean함수로 뺐다. WithinPast_Arden
  -->
  <!-- ************************************************************************ -->
  <xsl:template match="WithinPast">
    <!--Unary operators including Reverse operation-->
    <!-- syntax check for number of arguments -->
    <xsl:call-template name="CheckNumberOfArguments">
      <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
    </xsl:call-template>
    <!--Verbe with/without nigation-->
    <xsl:choose>
      <xsl:when test="contains(name(parent::*),'Not')">
        <xsl:value-of select="substring-before(name(parent::*),'Not')"/>
        <xsl:text> Not </xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="name(parent::*)"/>
      </xsl:otherwise>
    </xsl:choose>
    <!-- Preposition1 -->
    <xsl:value-of select="name()"/>
    <xsl:text>_Arden(</xsl:text>
    <!--Subject-->
    <xsl:apply-templates select="following-sibling::*[1]"/>
    <xsl:text> , </xsl:text>
    <xsl:apply-templates select="following-sibling::*[2]"/>
    <xsl:text>) </xsl:text>
  </xsl:template>


  <!-- Ternary expressions -->
  <!-- ************************************************************************ -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************ -->
  <!--
    Visual Basic에 없는 함수이다보니 의미에 맞게 새로 만들어 주어야 한다.
  -->
  <!-- ************************************************************************ -->
  <xsl:template match="WithinTo|WithinFollowing|WithinPreceding|WithinSurrounding">
          <!--Unary operators including Reverse operation-->
          <!-- syntax check for number of arguments -->
          <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">3</xsl:with-param>
          </xsl:call-template>
          <!--Verbe with/without nigation-->
          <xsl:choose>
                    <xsl:when test="contains(name(parent::*),'Not')">
                              <xsl:value-of                                    select="substring-
before(name(parent::*),'Not')"/>
                              <xsl:text> Not </xsl:text>
                    </xsl:when>
                    <xsl:otherwise>
                              <xsl:value-of select="name(parent::*)"/>
                    </xsl:otherwise>
          </xsl:choose>
    <!-- Preposition1 -->
```

```
  <xsl:value-of select="name()"/>
  <xsl:text>_Arden(</xsl:text>
  <!--Subject-->
  <xsl:apply-templates select="following-sibling::*[1]"/>
  <xsl:text> , </xsl:text>
  "<xsl:apply-templates select="following-sibling::*[2]"/>"
                    <!-- Preposition1 -->
                    <xsl:text> ,  </xsl:text>
                    <!--Objective 2-->
                    "<xsl:apply-templates select="following-sibling::*[3]"/>"
  <xsl:text>) </xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
  Visual Basic에 없는 함수이다보니 의미에 맞게 새로 만들어 주어야 한다.
  Exp2의 일자와 Exp1이 같은 일자이면 True
-->
<!-- ************************************************************************** -->
<xsl:template match="WithinSameDayAs">
                    <!--Subject-->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> </xsl:text>
                    <!--Verbe-->
                    <xsl:value-of select="name(parent::*)"/>
                    <!--Nigation-->
                    <xsl:if test="contains(name(parent::*),'Not')">
                              <xsl:text> Not </xsl:text>
                    </xsl:if>
                    <!-- Preposition1 -->
                    <xsl:text> WithinSameDayAs_Arden( </xsl:text>
                    <!--Objective 2-->
                    <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>) </xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    연속된 문자를 연결해주는 함수
    VB.NET에선 & 가 문자를 연결해주는 기호임.
-->
<!-- ************************************************************************** -->
<!--9.8          String Operators -->
        <xsl:template match="Concat">
                    <xsl:for-each select="following-sibling::*">
                              <xsl:apply-templates select="."/>
                              <xsl:if test="position()!=last()">
                                        <xsl:text> &amp; </xsl:text>
                              </xsl:if>
                    </xsl:for-each>
        </xsl:template>


<!-- 9.9          Arithmetic Operators -->
<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
  Exp1과 Exp2를 서로 더하는 함수...+++++++++++++++++++
-->
<!-- ************************************************************************** -->
```

```
  <xsl:template match="Apply/Plus">
                    <xsl:for-each select="following-sibling::*">
                              <xsl:apply-templates select="."/>
                              <xsl:if test="position()!=last()">
                                        <xsl:text> + </xsl:text>
                              </xsl:if>
                    </xsl:for-each>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
    Exp1과 Exp2를 서로 곱하는 함수... ****************
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/Times">
          <xsl:for-each select="following-sibling::*">
          <xsl:if test="name()='Apply' and contains('Plus|Minus',name(./*[1]))">(</xsl:if>
          <xsl:apply-templates select="."/>
          <xsl:if test="name()='Apply' and contains('Plus|Minus',name(./*[1]))">)</xsl:if>
          <xsl:if test="position()!=last()">
                    <xsl:text> * </xsl:text>
                    </xsl:if>
          </xsl:for-each>
</xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
    Exp1과 Exp2를 서로 빼하는 함수...
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/Minus">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> - </xsl:text>
                    <xsl:if test="name(following-sibling::*[2])='Apply' and
\ contains('Plus|Minus',name(following-sibling::*[2]/*[1]))">(</xsl:if>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
                    <xsl:if test="name(following-sibling::*[2])='Apply' and
 contains('Plus|Minus',name(following-sibling::*[2]/*[1]))">)</xsl:if>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
    Exp1과 Exp2를 서로 나누는하는 함수... ////////////////////////////
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/Divide">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                              <xsl:with-param      name="NumberOfArguments">2</xsl:with-
param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:if test="name(following-sibling::*[1])='Apply' and
```

```
contains('Plus|Minus|Divide',name(following-sibling::*[1]/*[1]))">
        (
    </xsl:if>
                        <xsl:apply-templates select="following-sibling::*[1]"/>
                        <xsl:if test="name(following-sibling::*[1])='Apply' and
contains('Plus|Minus|Divide',name(following-sibling::*[1]/*[1]))">
        )
    </xsl:if>
                        <xsl:text> / </xsl:text>
                        <xsl:if test="name(following-sibling::*[2])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[2]/*[1]))">
        (
    </xsl:if>
                        <xsl:apply-templates select="following-sibling::*[2]"/>
                        <xsl:if test="name(following-sibling::*[2])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[2]/*[1]))">
        )
    </xsl:if>
            </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
    Exp1과 Exp2를 서로 자승하는 함수... ^^^^^^^^^^^^^^^^
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/Power">
                        <!-- syntax check for number of arguments -->
                        <xsl:call-template name="CheckNumberOfArguments">
                                    <xsl:with-param      name="NumberOfArguments">2</xsl:with-
param>
                        </xsl:call-template>
                        <!-- presentation -->
                        <xsl:if test="name(following-sibling::*[1])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[1]/*[1]))">
        (
    </xsl:if>
                        <xsl:apply-templates select="following-sibling::*[1]"/>
                        <xsl:if test="name(following-sibling::*[1])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[1]/*[1]))">
        )
    </xsl:if>
                        <xsl:text> ^ </xsl:text>
                        <xsl:if test="name(following-sibling::*[2])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[2]/*[1]))">
        (
    </xsl:if>
                        <xsl:apply-templates select="following-sibling::*[2]"/>
                        <xsl:if test="name(following-sibling::*[2])='Apply' and
 contains('Plus|Minus|Times|Divide',name(following-sibling::*[2]/*[1]))">
        )
    </xsl:if>
            </xsl:template>


            <!-- 9.10            Temporal Operators -->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
    Duration After/Before Standard-Date ==> 기준일자로 부터 Duration간의 날짜이동
    계산한 날짜를 넘겨준다.
    기준일자를 String이 아니라 변수로 받아야 한다.
  -->
  <!-- ************************************************************************** -->
```

```
<xsl:template match="Apply/Before|Apply/After|Apply/From">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
  <xsl:value-of select="name()"/>
  <xsl:text>_Arden(</xsl:text>
  <!-- presentation -->
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> , </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
  Duration After/Before Standard-Date ==> 기준일자로 부터 Duration간의 날짜이동
  계산한 날짜를 넘겨준다.
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/Ago">
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:text> Ago </xsl:text>
        </xsl:template>


        <!-- 9.11            Duration Operators -->
<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
      Year     :  Return Month
      Month :    Return Month
      Week   : Return Seconds
      Day    :  Return Seconds
      Hour :   Return Seconds
      Minute   : Return Seconds
      Second : Return Seconds
      해서...각가의 함수를 만든다.
-->
<!-- ************************************************************************** -->
<!-- Year|Month|Week|Day|Hour|Minute|Second are already   -->
        <xsl:template match="Year|Month|Week|Day|Hour|Minute|Second">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
  <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates select="following-sibling::*"/>
                    <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
      ExtractYear      :  Return Year
      ExtractMonth :    Return Month
      ExtractWeek   : Return Week
      ExtractDay    :  Return Day
      ExtractHour :   Return Hour
      ExtractMinute   : Return Minute
```

```
        ExtractSecond : Return Second
        해서...각가의 함수를 만든다.
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="ExtractYear|ExtractMonth|ExtractWeek|ExtractDay|ExtractHour|
  ExtractMinute|ExtractSecond">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates select="following-sibling::*"/>
                    <xsl:text>)</xsl:text>
          </xsl:template>


          <!--9.12 Aggregation Operators with Index Extraction Aggregation operators-->
          <!--9.12.18          Element (binary), 9.12.19
Extract characters ... (unary, right associative) are not yet covered -->
          <!--SeqTo is included in 9.2-->
  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
        Count  :  Return 갯수
        Exist :   Return 값이 있으면..
        Average:  Return 평균
        Median : Return 중위수
        Sum  :  Return 합계
        Stddev  : Return 표준편차
        Variance  : Return 분산
  -->
          <xsl:template match="Apply/Count|Apply/Exist|Apply/Average|Apply/Median|
Apply/Sum|Apply/Stddev|Apply/Variance">
                    <!--Unary operators including Reverse operation-->
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates select="following-sibling::*"/>
                    <xsl:text>)</xsl:text>
          </xsl:template>


  <!-- ************************************************************************** -->
  <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
  <!-- ************************************************************************** -->
  <!--
      Any  :  하나라도 True면 True Return
      All   :   모두 True일때만 True Return
      No   :   모두 False일때만 True Return
      Reverse : 주어진 값들의 역순 제공
  -->
  <!-- ************************************************************************** -->
  <xsl:template match="Apply/Any|Apply/All|Apply/No|Apply/Reverse">
                    <!--Unary operators including Reverse operation-->
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:value-of select="name()"/>
```

```
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates select="following-sibling::*"/>
  <xsl:text>)</xsl:text>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    Minimum     :   주어진 값들중 최소값
    Maximum    : 주어진 값들중 최대값
    IndexMinimum  :  주어진 값들중 최소값이 있는 위치값
    IndexMaximum :  주어진 값들중 최대값이 있는 위치값
    Last  :  주어진 값들중 맨 마지막값
    First  : 주어진 값들중 맨 첫번째 값
    Earliest : 주어진 값들중의 시간중 가장 빠른 값
    Latest  : 주어진 값들중의 시간중 가장 늦은 값
    IndexEarliest : 주어진 값들중 시간이 가장 빠른 값의 위치값
    IndexLatest   : 주어진 값들중 시간이 가장 느린 값의 위치값
-->
<!-- ************************************************************************** -->
<xsl:template match="Apply/Minimum|Apply/Maximum|Apply/IndexMinimum|
Apply/IndexMaximum|Apply/Last|
Apply/First|Apply/Earliest|Apply/Latest|Apply/IndexEarliest|Apply/IndexLatest">
                    <xsl:choose>
                              <xsl:when test="count(following-sibling::*)=1">
                              <xsl:value-of select="name()"/>
                              <xsl:if test="name(following-sibling::*[1])!='List'">
                                        <xsl:text>_Arden(</xsl:text>
                              </xsl:if>
                              <xsl:apply-templates select="following-sibling::*[1]"/>
                              <xsl:if test="name(following-sibling::*[1])!='List'">
                                        <xsl:text>)</xsl:text>
                              </xsl:if>
                              </xsl:when>
                              <xsl:when test="count(following-sibling::*)=2">
                              <xsl:value-of select="name()"/>
                              <xsl:text>_Arden(</xsl:text>
                              <xsl:apply-templates select="following-sibling::*[1]"/>
                              <xsl:text>,</xsl:text>
                              <xsl:apply-templates select="following-sibling::*[2]"/>
      <xsl:text>)</xsl:text>
                    </xsl:when>
                    <xsl:otherwise>
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
                    </xsl:call-template>
                    </xsl:otherwise>
        </xsl:choose>
        </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    주어진 값들중에서 조건에 가장 가까운 값을 찾아서 전달한다.
-->
<!-- ************************************************************************** -->
<!-- 9.13          Query Aggregation Operators -->
        <!-- 9.13.2          Nearest ... From (binary, right associative)          62  -->
        <xsl:template match="Apply/Nearest">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">2</xsl:with-param>
```

```
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:if test="name(following-sibling::*[1])='Apply'">
                              <xsl:text>(</xsl:text>
                    </xsl:if>
                    <xsl:apply-templates select="following-sibling::*[1]"/>
                    <xsl:if test="name(following-sibling::*[1])='Apply'">
                              <xsl:text>)</xsl:text>
                    </xsl:if>
                    <xsl:text>, </xsl:text>
                    <xsl:apply-templates select="following-sibling::*[2]"/>
  <xsl:text>) </xsl:text>
          </xsl:template>
          -->

<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    Increase     : 주어진 목록값들 사이의 증가값(exp2 - exp1)
    Decrease     : 주어진 목록값들 사이의 감소값(Exp1 - Exp2)
-->
<!-- ************************************************************************** -->
<xsl:template match="Increase|Decrease">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates/>
                    <xsl:text>) </xsl:text>
          </xsl:template>


<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    PcntIncrease     : 주어진 목록값들 사이의 증가값을 %로 계산(exp2 - exp1)
    PcntDecrease     : 주어진 목록값들 사이의 감소값을 %로 계산(Exp1 - Exp2)
-->
<!-- ************************************************************************** -->
<xsl:template match="PcntIncrease|PcntDecrease">
                    <!-- syntax check for number of arguments -->
                    <xsl:call-template name="CheckNumberOfArguments">
                    <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                    </xsl:call-template>
                    <!-- presentation -->
                    <xsl:text></xsl:text>
                    <xsl:value-of select="name()"/>
                    <xsl:text>_Arden(</xsl:text>
                    <xsl:apply-templates/>
                    <xsl:text>) </xsl:text>
          </xsl:template>


<!--9.16          Numeric Function Operators -->
<!-- ************************************************************************** -->
<!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
<!-- ************************************************************************** -->
<!--
    Arccos  :  Arc-Cosine
    Arcsin   : Arc_Sine
```

```
        Arctan   : Arc_Tangent
        Cosine
        Sine
        Tangent
        Exp      : the power of its argument
        Log
        Log10
        Int    :  가까운 정수
        Floor
        Ceiling
        Truncate
        Round
        Abs
        Sqrt
        AsNumber:  to_number
    -->
    <!-- ************************************************************************** -->
    <xsl:template match="Arccos|Arcsin|Arctan|Cosine|Sine|Tangent|Exp|Log|
    Log10|Int|Floor|Ceiling|Truncate|Round|Abs|Sqrt|AsNumber">
                        <!-- syntax check for number of arguments -->
                        <xsl:call-template name="CheckNumberOfArguments">
                        <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                        </xsl:call-template>
                        <!-- presentation -->
                        <xsl:value-of select="name()"/>
                        <xsl:text>_Arden(</xsl:text>
                        <xsl:apply-templates select="following-sibling::*"/>
                        <xsl:text>)</xsl:text>
            </xsl:template>


            <!-- 9.17 Time Function Operator -->
    <!-- ************************************************************************** -->
    <!-- ************* Generation to VB.NET By Kangsoo Kim ********************** -->
    <!-- ************************************************************************** -->
    <!--
        해당 변수의 작성된 또는 발생된 일자(Time)얻어 오기
    -->
    <!-- ************************************************************************** -->
    <xsl:template match="Apply/TimeOf">
                        <!-- syntax check for number of arguments -->
                        <xsl:call-template name="CheckNumberOfArguments">
                        <xsl:with-param name="NumberOfArguments">1</xsl:with-param>
                        </xsl:call-template>
                        <!-- presentation -->
      <xsl:value-of select="name()"/>
      <xsl:text>_Arden(</xsl:text>
      <xsl:apply-templates select="following-sibling::*"/>
      <xsl:text>)</xsl:text>
            </xsl:template>
  </xsl:stylesheet>
```

# Abstract

## Developing an ArdenML XSLT for Clinical Decision Support System

Kangsoo Kim
Graduate School of Public Health
Yonsei University

(Directed by Professor Young-Moon Chae, Ph.D)

This study was initiated to develop eXtensible Markup Language (XML) module for Arden Markup Language which can be used in Clinical Decision Support System (CDSS). Arden Syntax was chosen as the HL7 standard language because of the advantages such as information sharing and reusability. But, at the same time, it also has problems such as Curly Brace problem and problem of requiring additional compiler. Development of Arden Syntax compiler not only requires a lot of time and efforts, but also requires reorganization of existing systems according to the Medical Logic Module. This is also an erroneous process.

There were several structural differences in three approaches: Arden Syntax is Category; ArdenML is Tag; and Visual Basic is

Comment, Function, Events, Class, and Procedure. XSLT was developed by analyzing three approaches. Based on XSLT, ArdenML was translated into Visual Basic and the interface module was developed to allow communicating with legacy systems.

In this study, we verified that ArdenML can effectively used in developing CDSS by developing the hypertension Medical Logic Module with clinical rules on hypertension and developing interface and XSLT that is capable of translating to various systems using widely used general-purpose computer languages (such as VB, C#, JAVA). Most of the CDSS are independently developed and run, and therefore they require re-entry of rules, but this system makes it easier to interface with the existing systems simply by translating or compiling rules. This enhances development efforts to Rule Based CDSS.