디지털산업정보학회 논문지 제14권 제3호-2018년 9월
http://dx.doi.org/10.17662/ksdim.2018.14.3.021

다중 뷰 편집환경을 위한 점진적 다중진입 지원 파서에 대한 연구*

염 세 훈** · 방 혜 자***

A Study of Incremental and Multiple Entry Support Parser for Multi View Editing Environment

Yeom Saehun · Bang Hyeja

〈Abstract〉

As computer performance and needs of user convenience increase, computer user interface are also changing. This changes had great effects on software development environment. In past, text editors like vi or emacs on UNIX OS were the main development environment. These editors are very strong to edit source code, but difficult and not intuitive compared to GUI(Graphical User Interface) based environment and were used by only some experts. Moreover, the trends of software development environment was changed from command line to GUI environment and GUI Editor provides usability and efficiency. As a result, the usage of text based editor had decreased.

However, because GUI based editor use a lot of computer resources, computer performance and efficiency are decreasing. The more contents are, the more time to verify and display the contents it takes.

In this paper, we provide a new parser that provide multi view editing, incremental parsing and multiple entry of abstract syntax tree.

Key Words : Abstract Syntax Tree, Incremental Passing, Multi View Editing Environment, Multiple Entry

Ⅰ. 서론

컴퓨팅 환경이 사용자 친화적으로 변화하면서 통합개발환경 사용자 친화적으로 변화하고 있으며 사용자들의 다양한 요구에 의해 형태와 기능이 다양화되고 있다. 이러한 다양화로 시스템의 복잡성이 증가하면서 화면의 갱신등과 같은 시스템 성능과 관련된 문제점들이 나타나고 있고 이런 문제점들은 시스템 성능의 저하를 가져온다.

특히 사용자에게 다양한 뷰를 제공하는 편집환경일수록 성능저하의 가능성은 간과할 수 없는 문제이다. 이에 본 논문에서는 성능저하를 개선하기 위해 역파싱을 활용한 점진적 파싱으로 점진성

* 이 연구는 서울과학기술대학교 교내연구비의 지원으로 수행되었습니다.
** 동서울대학교 컴퓨터소프트웨어과 부교수(주저자)
*** 서울과학기술대학교 컴퓨터공학과 교수(교신저자)

(incrementality)과 일관성(consistency)을 제공하여 작업의 효율과 시스템의 성능을 높이고자 한다. 점진성은 어떤 작업의 결과에 대한 변화가 생겼을 때 전체 작업을 재실행하지 않고 변화된 부분만 재실행하는 것으로 시스템의 성능을 향상시킬 수 있다.

일반적인 파서의 경우 문서가 모두 작성이 되고 난 후 문서의 적합성을 검사하며 적합성 검사 시 문서 전체에 대한 적합성 검사가 이루어진다. 이런 전체문서의 적합성 검사는 문서의 갱신이나 수정이 많을 때 검사시간을 증가시켜 작업의 효율을 떨어뜨린다.

본 논문에서 제시하는 점진적 파싱방법[1]은 사용자가 문서의 일부분을 수정하였을 때 전체 문서를 검증하지 않고 수정된 부분만 재검증하도록 설계하여 검증속도를 높여주었다. 편집기에서의 지원하는 점진성은 편집화면의 갱신에 대한 점진성이다.

편집환경 중 그래픽 편집환경은 그래픽 요소를 사용하여 작업자가 작업내용을 직관적으로 파악할 수 있게 하여 작업의 효율을 높여준다는 장점이 있다. 하지만 그래픽 편집기는 컴퓨터 자원을 많이 요구하는 그래픽 요소를 사용하기 때문에 텍스트 편집기보다 많은 컴퓨터 자원을 요구하며 문서의 수정내용이 많거나 수정횟수가 많을 때 수정 내용에 대한 화면의 갱신속도가 느려지므로 작업효율이 낮아질 수 있다. 특히 작업량이 많아지거나 문서가 복잡해지면 성능 차이는 더욱 크다. 이러한 단점 때문에 그래픽 편집기의 경우 화면의 갱신속도가 매우 중요하며 시스템 성능 향상을 위한 효과적인 화면 갱신이 요구된다.

이러한 화면 갱신 방법 중 수정된 문서의 검증을 문서 전체로 하지 않고 수정된 문서부에만 적용함으로써 검증속도를 높일 수 있으며 수정된 내용을 화면에 반영하기 위해 편집화면 전체를 갱신하지 않고 수정된 화면만 변경하는 방법으로 작업 효율과 시스템 성능을 높여 작업시간을 단축하였다.

편집환경에서 점진성과 함께 요구되는 것이 일관성이다. 다중 뷰 개발환경[2,3]은 작성하는 편집대상에 대한 다양한 정보를 서로 다른 출력 방법을 이용하여 사용자에게 보여줌으로써 편집대상의 이해를 높여서 최적의 문서를 작성할 수 있도록 도와준다. 특히 XML응용언어들의 편집환경[4-8]에서는 문서의 다양한 정보는 더욱 중요하다. 그러나 개발환경의 여러 가지 문서생성 방법들이 일관성을 유지 못하고 작업 중인 내용과 상이한 내용을 다른 출력화면에 나타낸다면 사용자는 문서에 대한 이해도가 저하되어 다중 뷰 편집환경의 의미가 없어지게 된다. 이러한 단점을 막기 위하여 편집환경에서는 문서생성방식간의 일관성을 제공하였다.

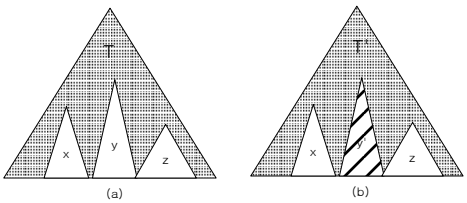
Ⅱ. 점진적 파싱을 통한 문서생성

점진적 문서생성은 문서나 프로그램이 수정된 후 전체적인 문서생성을 수행하지 않고 수정된 부분만 재생성하여 빠른 응답시간을 보장하는 문서 생성방법이다.

점진적 문서생성 방법은 다음과 같다. 그림 1의 (a)와 같이 파스트리 T= xyz라고 가정하자. 이때 y → y′로 수정되면 파서는 파스트리 T 전체를 생성하지 않고 (b)와 같이 y′에 해당되는 서브트리만 재생성하여 파스트리에 추가하여 새로운 파스트리 T′를 구성한다. 이렇게 부분 트리만 수정함으로써 파서는 전체를 다시 생성하여야 하는 부담이 줄어 전체적인 문서생성속도가 증가하며 수정내용에 대한 빠른 적합성 검사를 보장받을 수 있다.

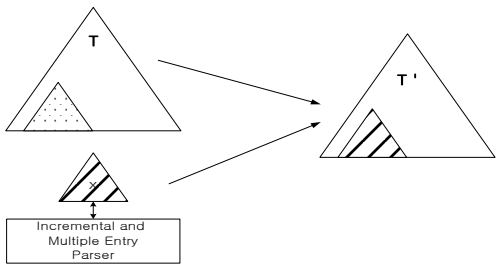
점진적 문서생성으로 작업 중 입력과 오류의 탐지 사이의 지연을 감소시킴으로써 작업의 전체적인 효율과 신뢰도가 증가된다. 특히 구문지향 편집환경

에서는 점진적 파싱은 필수적이다. 점진적 파서를 가진 편집환경은 문서를 작성하면서 바로 문서에 대한 검증이 이루어진다. 이렇게 함으로써 문서 작성자는 언제나 적합성이 보장된 문서만 작성하며 적합성이 보장된 환경에서 최적의 구조를 가진 문서의 작성만 전념할 수 있게 하여 작업 효율을 높일 수 있도록 한다.



<그림 1> 점진적 파싱방법

점진적 파서는 작성하는 문서의 적합성을 검사하고 문서트리를 생성하기 위한 점진적(incremental)이고 다중 진입(multiple entry)을 지원하는 파서로 추상구문트리를 이용하여 점진적 파싱을 제공한다. 공용 파서는 문서가 새롭게 생성되거나 일부분만 수정되거나 상관없이 문서의 루트에서부터 하향식(top-down) 방식으로 문서전체를 생성한다. 그러나 점직적 파싱을 지원하는 파서는 사용자의 불완전한 입력이 발생하거나 문서 일부만 수정되었을 때는 문서의 전체에 대한 검증이 실행되는 것이 아니라 수정된 문서의 일부분만을 생성한다. 공용 파서는 문서를 검증할 때 잘못된 입력이나 불완전한 입력이 들어오면 문서부분생성이 불가능하여 전체문서를 생성해야 하므로 실시간 문법 검증은 불가능하며 문서의 일부부만 편집되더라도 전체 문서 생성이 이루어져야 하므로 문서의 크기가 커질 경우 문서 검증에 많은 시간이 걸려서 작업효율이 떨어진다. 이에 반해 점진적 파싱을 지원하는 파서는 입력 스트링의 맨 처음 입력이 정의된 요소일 경우 해당되는 요소를 루트로 하는 서브트리를 생성하고 편집단계에서 해당되는 요소의 입력이 종료되었을 때 편집하고 있는 전체 문서의 트리에 삽입하여 전체 트리를 확장한다. 이렇게 입력 스트링의 일부를 루트로 하여 생성하는 것을 다중 진입 이라고 한다. 다중 진입은 불완전한 입력에 대한 검증이나 문서의 일부분만 검증할 수 있도록 하여 점진적 문서생성을 도와준다. 특히 다중 진입은 문서의 특정부분만 따로 작성하고 이에 대한 검증을 할 수 있게 함으로써 미리 문서의 일부분을 만들어 놓고 문서 작성 중 원하는 곳에 삽입할 수 있게 하였다. 점진적이고 다중진입을 지원하는 파서를 이용한 문서생성은 그림 2와 같이 문서 전체를 구축하지 않고 일부분만 작성하고 검증하여 문서에 삽입할 수 있도록 하였다. 이때 파서는 작성된 문서의 일부분이 전체 문서를 구성하는 트리에 삽입가능한지를 검사하고 적합성이 보장되었을 때 전체 문서트리에 부분 트리를 삽입한다.



<그림 2> 점진적이고 다중진입을 지원하는 파서를 이용한 문서생성

본 논문의 점진적 다중진입 파서는 트리조작기를 이용하여 문서를 생성한다. 트리조작기는 편집기에서 사용자가 트리조작 명령을 내리면 이에 따라 트리를 재구성하면서 파서와 연동하여 적합성을 검증한다. 사용자가 편집기에서 생성할 문서 일부의 첫

요소를 주면 파서는 문서의 완성도와 상관없이 사용자로부터 주어진 요소를 루트로 하는 트리를 생성하면서 문서의 적합성을 검사하고 적합성이 검사된 문서의 일부분은 전체문서에 삽입함으로써 전체적인 문서의 적합성을 보장받게 한다.

점진적 다중진입 문서생성은 추상구문트리 구성방법을 이용한다. 다중진입 문서생성은 문서의 일부분만을 추상구문트리로 구성하는 것이다. 일반적인 추상구문트리의 경우 문서를 생성할 때 항상 추상문법의 시작요소부터 시작한다. 그러나 다중진입 문서생성의 경우 추상문법의 시작요소가 아닌 요소를 루트로 하는 트리를 생성하여야 한다. 이를 위해 현재 노드를 루트로 하는 문서생성을 원하는지를 검사하고 추상문법의 시작요소가 아닌 요소를 루트로 하는 다중진입을 원하는 경우 현재 노드를 루트로 하여 구문 분석기에서 새로운 토큰을 받아 [알고리즘 2]를 이용하여 추상구문트리를 구성한다. 점진적 다중진입 문서생성을 위한 방법은 [알고리즘 1]과 같다.

[알고리즘 1] 점진적 다중 진입 문서생성 알고리즘

입력 : 부모 노드

출력 : 추상구문트리

방법 :

1. 현재노드가 다중 진입을 요구하는지 검사하고 다중 진입을 원하는 경우 구문 분석기에서 새로운 토큰을 읽고 2를 수행한다.
2. 알고리즘 2를 수행한다. □

[알고리즘 2] 추상구문트리의 구성

입력 : 입력 요소의 추상문법 생성규칙 번호

출력 : 추상구문트리

방법 :

1. 알고리즘 4-1로 트리의 루트를 생성한다.
2. 생성된 트리의 맨 마지막 노드의 위치를 저장한다.
3. 구문 분석기 호출하여 입력토큰을 받는다.
4. 입력된 토큰에 따라 다음 과정을 수행한다.
   a) 저장된 맨 마지막 노드에 현재 읽은 입력토큰 노드를 추가를 위해 맨 마지막 노드를 리스트 노드로 변환한다.
   b) 변환된 리스트 노드의 왼쪽에 입력된 토큰의 값을 갖는 노드를 추가한다.
   c) 입력된 토큰과 동일한 레벨의 입력을 고려하여 리스트 노드의 오른쪽에 Nil노드를 추가한다.
   d) b)에서 추가된 노드를 루트로 하는 트리를 생성하기 위해 Nil노드를 추가한다.
   e) 입력된 토큰을 루트로 하는 트리를 생성하고 d)에서 생성한 Nil노드에 추가한다.
   f) 구문 분석기를 호출하여 다음 토큰을 읽고 a)부터 반복한다.

Ⅲ. 점진적 파싱을 통한 화면갱신

컴퓨팅 환경에서 편집환경은 가장 많이 사용되고 있으며 중요한 역할을 차지하는 시스템 소프트웨어 중의 하나이다. 편집기는 프로그래밍이나 문서, 숫자로 구성된 정보를 생성하고 수정하는 역할을 수행하였다. 그러나 최근에는 컴퓨터에서 다루는 정보의 다양화로 편집기의 역할이 프로그래밍을 하거나 텍스트 정보를 생성하는 역할에만 국한되지 않고 영상이나 음향과 같은 다양한 정보를 구성하고 다루는 역할을 요구받게 되었다. 이에 따라 편집기는 초창기 단순한 텍스트 정보만을 작성하고 저장하는 형태에서 영상이나 음향과 같은 다양한 정보를 다룰 수 있는 형태로 발전하였다. 이런 편집기 형태의 다양화는 문서를 작성하기 위한 문서 편집기에서도 나타

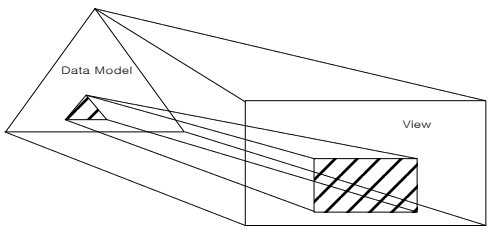
났다. 초창기 문서 편집기들은 단순히 정보를 보관하기 위해 문자나 숫자만을 저장할 수 있도록 단순한 형태를 취하였다. 그러나 최근 문서의 형태와 내용이 다양해짐에 따라 이를 효과적으로 편집하고 작성하기 위한 다양한 형태의 편집기들이 나타나고 있다. 그러나 편집기의 형태가 다양해지고 다양한 형태의 문서를 작성함에 따라 컴퓨터 성능과 연관된 문제점이 나타났다.

컴퓨터 성능과 연관된 문제 중에서 문서 편집기와 연관되어 있는 것이 작성하는 문서의 정보를 화면에 나타내는 것이다. 문서 편집기는 작성하는 문서의 내용을 화면에 출력해야함으로 출력하는 문서의 내용이 복잡하거나 문서의 형태가 다양할 경우 많은 컴퓨팅 시간을 요구하여 작업시간이 증가한다. 그러므로 최근 문서 편집기는 효율적인 화면제어로 작업시간의 단축이 필요하다. 작성물의 내용을 화면에 부적절하게 출력하는 경우 사용자는 잘못된 문서를 작성하거나 적절하지 못한 문서를 작성할 수 있다. 또한 잘못된 화면제어는 컴퓨터 성능을 떨어뜨릴 수 있다. 특히 최근 사용되는 컴퓨팅 환경이 그래픽 환경임에 따라 효과적인 화면제어의 중요성은 더욱 커지고 있다.

편집기 문서생성환경과 같이 여러 편집기를 제공하는 환경은 다양한 편집화면으로 구성된다. 이러한 다양한 편집화면을 가진 환경은 편집을 수행하는 사람에게 여러 가지 문서정보를 제공하여 문서에 대한 이해도를 향상시키고 최적의 문서를 만드는데 유용하다. 그러나 문서 내용의 변화에 따라 화면의 많은 부분을 갱신하여야 한다는 부담이 생기고 이에 따라 시스템 성능의 저하를 가져올 수 있다. 시스템 성능 저하의 원인은 최근 편집기가 작성 문서에 대한 사용자의 이해도 증가와 사용의 편이성 이유로 그래픽 요소를 이용한 그래픽 편집기 형태로 제공하고 있기 때문이다. 그래픽 편집기는 텍스트 편집기에 비해 더 많은 컴퓨터 자원을 요구한다. 텍스트 편집기는 단순한 텍스트 문자만을 화면에 출력하면 되지만 그래픽 편집기는 화면에 나타난 그래픽 요소의 위치와 순서를 기억하고 있다가 화면에 출력하여야 하므로 더 많은 양의 메모리와 계산시간을 요구하게 된다. 이러한 점에서 화면의 갱신은 단순한 문서내용의 화면변경이 아니라 시스템 성능을 크게 좌우하는 결정적인 요소가 되었다.

편집기 문서생성환경에서도 다양한 편집기를 제공하고 있고 다수의 그래픽 편집기를 사용하기 때문에 화면갱신으로 시스템 성능이 낮아질 수 있다. 이러한 시스템 성능저하를 보완하기 위하여 본 편집환경에서는 점진적 화면갱신(incremental update)을 사용하였다.

점진적 화면갱신은 문서를 편집하는 동안 문서의 내용이 변경되고 화면을 갱신할 때 화면 전체를 갱신하는 것이 아니라 그림 3과 같이 변경된 문서 부분만을 화면에 반영하여 화면에서 변경되는 부분을 최소화하는 것이다. 또한 사용자가 문서를 작성하는 입력 시간에 변경된 내용을 화면에 즉각적으로 출력하게 함으로써 시스템 성능의 저하를 최소화하였다. 점진적 화면갱신 방법으로 화면 전체를 갱신하는 부담을 줄이고 입력 시간을 이용한 즉각적인 화면 갱신에 따라 시스템 성능의 저하를 막을 수 있다.



<그림 3> 점진적 화면갱신

본 편집환경에서 사용되는 점진적 화면 갱신방법은 [알고리즘 3]의 역파싱 알고리즘으로 제공한다. 화면이 수정되면 [알고리즘 3]에 따라 수정된 현재 노드를 기준으로 역파싱 알고리즘이 수행되어 수정된 문서부분만 화면에 출력하여 점진적 화면갱신을 보장한다.

[알고리즘 3] 역파싱 알고리즘

입력 : 현재노드, 부모노드

출력 : 역파싱된 출력화면

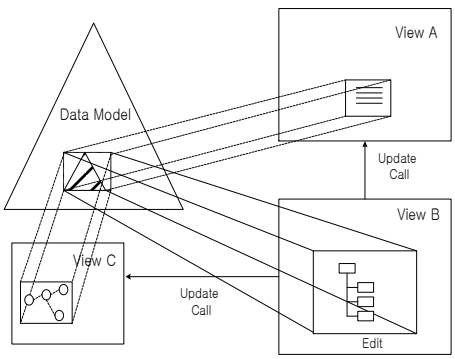
방법 :

1. 현재 노드가 리스트 노드인 경우 추상 문법에서 현재 노드를 좌측기호로 하는 생성규칙에 대한 역파싱 규칙을 읽고 생성규칙에서 하위노드가 있음을 표시하는 @가 나타나면 하위노드에 대한 역파싱을 다시 한다.
2. 현재 노드가 리스트 노드가 아닌 경우 현재 노드를 좌측기호로 하는 생성규칙에 대한 역파싱 규칙을 읽고 역파싱 규칙의 문자열을 출력하고 하위 노드가 존재하는 경우 @가 나타나면 하위노드에 대한 출력할 문자열을 가지고 있으면 출력하고 그렇지 않으면 역파싱을 다시 호출한다.

Ⅳ. 일관성

편집환경은 하나의 문서에 대해서 여러 다양한 뷰를 제공함으로써 사용자가 작성하는 문서에 대한 이해도를 높일 수 있어 작업 효율을 높일 수 있다. 그러나 편집환경은 다양한 편집화면에 의해 발생하는 단점도 가지고 있다. 이런 단점으로 문서내용의 화면 출력에 대한 문제가 있다. 하나는 앞에서 기술한 다양한 편집화면을 제공하여 사용자가 작성하는 문서에 대한 인지도는 높일 수 있으나 이에 따른 화면 갱신의 문제가 발생하여 시스템 성능을 저하시킬 수 있었다. 또 다른 단점은 다양한 편집화면간의 불일치성으로 의해 발생하는 인지도의 저하이다. 편집기 문서생성환경은 여러 편집화면으로 사용자가 다양한 문서정보를 제공하여 문서에 대한 이해도를 높여주어 최적의 문서를 만들 수 있도록 도와줌으로써 작업 효율을 높이는데 그 목적이 있다. 하지만 여러 편집화면이 서로 다른 편집진행 상황을 나타낸다면 사용자는 현재 작성되는 문서의 편집진행상황을 파악하지 못하여 문서작성 효율을 떨어뜨릴 수 있다. 이러한 단점을 보완하기 위하여 역파싱을 이용한 일관성을 제공하였다.

일관성(consistency)이란 여러 편집화면 중 하나를 이용하여 문서가 수정되었을 때 한 편집화면의 수정된 내용이 다른 편집화면에 즉각적으로 반영되어 여러 개의 편집화면들이 항상 동일한 내용을 출력하여야 하는 것을 지칭하며 편집하고 있는 상황을 여러 편집화면들이 동일하게 나타내줌으로써 사용자가 어떤 편집화면을 보던지 현재 편집하고 있는 위치를 파악할 수 있도록 하는 것이다.



<그림 4> 편집기 간의 일관성

일관성을 제공하기 위해서 그림 4와 같이 View

B에서 문서의 편집을 하면 추상구문트리가 수정이 되면서 View A와 C의 화면이 점진적으로 갱신된다. 이때 점진적 갱신을 하기 위해 View B는 현재 수정된 노드에 대한 View A와 C의 역파싱을 호출하고 편집이 발생한 노드를 알려주어 View A와 C가 [알고리즘 3]에 의해 점진적 화면 갱신을 할 수 있게 한다.

Ⅴ. 결론

컴퓨팅 환경이 사용자 중심으로 변화하면서 다양한 사용자 소프트웨어들이 제공되고 있다. 특히 사용자의 문서편집환경이나 통합개발환경은 사용자에게 다양한 뷰를 제공하여 사용자에게 다양한 정보를 제공하고 사용자의 작업 이해도를 높여준다. 그러나 사용자에게 동시에 많은 정보를 제공하면 제공된 정보의 갱신 시 문서전체에 대한 적합성을 검사하기 때문에 정보갱신이 많을 때 검사시간이 증가하며 작업효율이 떨어진다. 이러한 단점을 보완하기 위해 작업내용 전체의 적합성검사가 아닌 갱신된 부분에 대한 적합성 검사만 수행하여 효율저하를 감소할 수 있다.

이를 위해 문서의 내부구조인 추상구문트리를 구성할 때 다중진입이 가능하도록 구성한다. 다중진입이 가능한 추상구문트리는 입력이 발생하는 부분을 루트로 서브트리를 구성하고 입력이나 수정이 끝났을 때 서브트리를 전체트리에 삽입한다.

이러한 점진적 파싱을 통하여 전체 문서 검증에 따른 성능저하를 막아줄 수 있지만 문서 갱신에 따른 화면의 갱신도 전체화면에 대한 갱신이 발생할 시 작업효율이 저하될 수 있다. 이를 보완하기 위하여 화면 갱신도 전체가 아닌 수정된 내부구조에 해당되는 부분만을 역파싱하여 수정된 부분만을 출력함으로써 성능저하를 최소화할 수 있으며 동일한 추상구문트리를 통해 수정된 화면 부분을 다른 화면에 즉각적으로 표시해줌으로써 일관성을 유지할 수 있다.

이러한 점진적이고 다중진입을 허용하는 파서를 이용한 수정방법은 XML 응용언어를 작성하기 위한 편집환경 등에 매우 유용하게 적용될 수 있다. XML 응용언어 문서편집환경은 문서의 내용만 중요한 것이 아니라 문서의 구조 또한 중요한 부분이다. 그러므로 텍스트 편집기에서 입력이나 수정이 될 때 문서트리 구조를 동시에 보여주면 사용자의 작업 이해도를 향상시킬 수 있다.

참고문헌

[1] Carlo Ghezzi and Dino Mandrioli, “Incremental Parsing,” ACM Transactions on Programming Languages and System, Vol. 1, No. 1, July, 1979, pp.58-70.

[2] Jingwei Wu and Margaret-Anne D. Stoey, “A Multi-Perspective Software Visualization Environment,” Proceedings of the 2000 Conference of the Centre for Advanced Studies on Collaborative Research, November, 2000.

[3] Kang Zhang, Da-Qian Zhang and Jiannong Cao, “Design, Construction, and Application of a Generic Visual Language Generation Environment,” IEEE Transactions on Software Engineering, Vol.27, No.4, April 2001.

[4] 김효정, 민미경, “규칙기반 문서 분류기를 이용한 XML 문서의 자동생성,” 한국멀티미디어학회 학술논문집, 2000. 11, pp.125-128.

[5] 박상기, 박만곤, “XML DOM을 이용한 웹기반 전

자교과서 자동생성 시스템," 한국멀티미디어학회, 한국멀티미디어학회 학술논문집, 2004. 11, pp.571-574.

[6] 김민지, 이정아, 이종학, "교수-학습자료의 XML 문서 생성 시스템 설계 및 구현," 한국멀티미디어학회, 2003년도 한국멀티미디어학회 추계학술발표논문집, 2003. 11, pp.852-855.

[7] Martin Erwig, "A Visual Language for XML," Proceedings of the 16th IEEE Symposium on Visual Language, 2000.

[8] 염세훈, 방혜자, "템플릿 기반 XML문서 생성기의 설계 및 구현," 디지털산업정보학회 논문지, 제8권, 제4호, 2012, pp.73-81.

[9] 이재건, 염세훈, 방혜자, "Binary XML을 이용한 전자출결시스템 설계 및 개발," 디지털산업정보학회 논문지, 제11권, 제3호, 2015, pp.11-19.

■ 저자소개 ■

염 세 훈
(Yeom Saehun)

2002년~현재 동서울대학 컴퓨터소프트웨어과 교수
2005년 2월 숭실대학교 컴퓨터학과 (공학박사)
1994년 2월 숭실대학교 컴퓨터학과 (공학석사)
1992년 2월 서울과학기술대학교 컴퓨터공학과 (공학사)

관심분야 : 컴파일러, 프로그래밍언어, HCI, XML

E-mail : shyeom@du.ac.kr

방 혜 자
(Bang Hyeja)

1985년~현재 서울과학기술대학교 컴퓨터공학과 교수
2017년~현재 서울과학기술대학교 대학원장
1993년 숭실대학교 컴퓨터공학과 (공학박사)
1983년 5월 Univ. of North Texas 전자계산학과 (이학 석사)
1977년 2월 숭실대학교 컴퓨터공학과 (공학사)

관심분야 : 컴파일러, 프로그래밍언어, 형식언어론, XML

E-mail : hjbang@seoultech.ac.kr

논문접수일 : 2018년 08월 20일
수 정 일 : 2018년 08월 30일
게재확정일 : 2018년 09월 04일