얼굴 표정을 이용한 코딩 스타일 점수 시각화
(Coding Style Score Visualization Using Facial Expression)

지 정 훈 † 이 윤 정 †† 우 균 †††
(Jeong Hoon Ji) (Yun Jung Lee) (Gyun Woo)

요 약 본 논문에서는 소스코드의 코딩 스타일을 검사하고 코딩 스타일 점수를 얼굴 표정을 이용해 시각화하는 시스템인 *StyleVisualizer*를 제안한다. *StyleVisualizer*는 스타일 평가 점수에 따라 몇 가지 얼굴로 표현한다. 웃는 표정은 소스코드가 코딩 표준을 잘 준수했음을 의미한다. *StyleVisualizer*의 효과를 알아보기 위해 두 분반의 실용 컴퓨터 교과목 수강생을 대상으로 실험을 진행하였다. 실험에서는 *StyleVisualizer*를 사용한 분반과 사용하지 않은 분반에 대해 코딩 표준 준수에 대한 오류 비율을 비교했다. 실험결과, *StyleVisualizer*를 사용한 경우 30% 이상 오류 비율이 감소하였다. 논문에서 제안하는 시스템은 학생들에게 자신들의 프로그램에 대한 시각화된 결과를 피드백해 줌으로써 코딩 표준을 준수하고 가독성이 높은 프로그램을 작성하는 능력을 기르는 데 도움을 줄 수 있을 것으로 기대된다.

키워드 : 시각화, 코딩 표준, 코딩 스타일, 프로그램 가독성, 얼굴 표정

Abstract This paper presents an automated visualization system, called *StyleVisualizer*, which checks the coding style of source codes and visualizes the coding style score using facial expression. Our system represents some kinds of facial expressions according to the evaluated score of the code style: A smile face means that the source code follows coding standards correctly. To measure the effectiveness of the *StyleVisualizer*, some experiments have been conducted on two class students in an applied computer course. In the experiments, we have compared the error ratio for obeying the coding standards when the *StyleVisualizer* was used or not. According to the experimental results, the error ratio with the *StyleVisualizer* was reduced above 30% than that without it. We expect that our system can encourage the students to obey the coding standards by providing the feedback of the visualized faces corresponding to their programs, resulting in high readable programs.

Key words : Visualization, Coding Standard, Coding Style, Program Readability, Facial Expression

1. 서 론

소프트웨어 개발에 있어서 생산 비용을 절감하는 것은 중요한 요소이며, '프로그램 구현'과 '유지보수' 단계는 생산비용에서 많은 부분을 차지한다[1]. 소프트웨어 개발과정에서 코딩 표준(coding standard)을 준수하면 개발 생산성을 향상시킬 수 있다. 그리고 코딩 표준은 개발 단계에서 팀원들 간의 코드 리뷰(code review)와 의사소통을 원활하게 해준다.

코딩 표준에 따라 개발된 프로그램은 가독성이 높아지며, 이는 소스코드를 이해하기 쉽게 하고, 오류가 발생했을 때, 빠르게 수정할 수 있도록 한다[2]. 또한, 프로그램에서 중요한 정보를 더 빨리 찾을 수 있어 새롭게 추가되는 요구사항을 빠르게 반영할 수 있게 한다.

이로 인해, 코딩 표준은 실제 소프트웨어를 개발하는 연구소나 회사에서 연구자들에 의해 개별적으로 제안되어 사용되었다[3-5]. 그러나 코딩 스타일에 대한 정의가 모호하고, 경험적이고 주관적인 요소들이 많이 작용하므로 동일한 코딩 요소에 대해 서로 다른 규칙을 제시하

· 본 연구는 지식경제부 및 정보통신연구진흥원의 IT산업원천기술개발사업의 일환으로 수행하였음[2007-S-015-01, SaaS기반 이동형 개인맞춤 사무환경 구축 기술 개발]

† 학생회원 : 부산대학교 컴퓨터공학과
jhji@pusan.ac.kr
†† 정 회 원 : 부산대학교 컴퓨터공학과
leeyj01@pusan.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
woogyun@pusan.ac.kr
(Corresponding author임)

논문접수 : 2009년 11월 18일
심사완료 : 2010년 4월 20일

정보과학회논문지: 소프트웨어 및 응용 제37권 제7호(2010.7)

는 경우도 많이 있다. 다시 말해서, 개발자의 코딩 스타일은 오랜 습관이 많은 영향을 미친다.

Li의 연구에서는 대학의 C나 자바와 같은 프로그래밍 언어 교육에서 학생들에게 코딩 표준을 준수하도록 교육하는 것이 중요하며, 학생들이 규모가 큰 고급 프로그램 성공적으로 작성하는데 도움을 준다고 하였다[6]. 하지만 학생들은 코딩 표준을 지키는 것이 개발에 도움이 된다는 것은 잘 알지만 이를 지키면서 프로그램을 작성하는 것을 어렵게 느끼고 있다. 그러므로 프로그램을 처음 배우는 단계에서부터 코딩 표준을 지키면서 올바르게 프로그램을 작성하는 습관을 길러주는 교육이 필요하다.

본 논문에서는 프로그램 소스코드에 대해 코딩 스타일 지침들의 준수 여부를 검사하고 결과를 시각적으로 보여주는 *StyleVisualizer* 코딩 스타일 시각화 시스템을 제안한다. 그리고 코딩 표준의 여러 항목들 중에서 프로그래밍 언어를 처음 접하는 학생들에게 쉽게 익힐 수 있는 소스코드의 텍스트 및 시각적 레이아웃에 관한 편집 스타일(typographic style)을 중심으로 한 코딩 스타일 지침을 제시한다. 본 논문의 목적은 프로그래밍 수업에서 코딩 스타일에 대한 지속적인 평가와 피드백을 제공함으로써 학생들로 하여금 가독성 있는 프로그램을 작성하도록 유도하는 데 있다.

2. 관련연구

2.1 코딩 표준

코딩 표준을 준수하면서 작성된 프로그램은 확장이 용이하며 재사용이 뛰어나다는 장점이 있다. 이 때문에 프로그램 설계 및 구현과 관련하여 코딩 스타일에 대한 연구가 이루어졌다[7-9]. C 언어와 관련된 코딩 스타일 예로는 AT&T 스타일, Gnu 스타일, MISRA 스타일이 있다.

Indian Hill C 스타일[3]은 1990년에 AT&T사의 벨 연구소에 의해 발표되었다. 그리고 Gnu 코딩 표준[4]은 공개 소스를 기반으로 하는 Gnu 개발 프로젝트를 진행할 때, 준수해야 할 사항들을 정의하고 있다. 마지막으로 MISRA(Motor Industry Software Reliability Association) C 가이드라인[5]은 자동차 산업용 소프트웨어 개발을 위한 표준 지침으로 정의되었으며, 현재에는 영역을 확장하여 자동차뿐만 아니라 모든 산업 분야에 적용시킬 수 있도록 재정의 되었다. 이밖에도 C++ 및 자바를 비롯하여 여러 가지 프로그래밍 언어에 대해 코딩 시 지켜야 할 사항들을 정의하고 있다.

C 언어를 위한 코딩 표준들은 주로 문법적 구성요소들을 기준으로 세부 규칙들을 분류하고 있다. Indian Hill C 스타일은 총 11개의 범주(파일, 주석, 함수선언, 일반선언, 공백, 단순문, 복합문, 연산자, 이름규칙, 상수 메크로)로 분류하고 있다. 그리고 MISRA C 가이드라인에서는 총 17개의 범주로 분류하고 있다. Gnu 코딩 표준에서는 명시적으로 범주를 분류하고 있지는 않지만 각 요소들에 대해 세부규칙들을 정의하고 있다.

2.2 코딩 스타일 지원 도구

Li의 연구[6]에서는 프로그래밍 수업에서 코딩 표준을 학습하고 수용하는 것에 대한 학생들의 의견 조사를 통해 코딩 표준에 대한 인식과 교육이 중요함을 언급하였다. 그의 연구에서 학생들은 소스코드의 가독성에 대해 중요하다고 인식은 하고 있으나 가독성 높은 프로그램을 작성하기 위해 표준을 준수하는 것은 어렵게 느끼고 있으며, 잘 지키지 않는 것으로 조사되었다.

황준하의 연구[10]에서는 C 언어 교육과 정적 분석 도구 개발을 위해 코딩 표준들을 종합하여 새로운 C 코딩 스타일을 제안하고 CStyler라는 자동화된 C 코딩 스타일 검증기를 lex와 yacc를 이용해 개발하였다. 이 연구에서는 코딩 규칙의 분류 기준으로 문법적 요소뿐만 아니라 가능적 요소를 추가적으로 고려하여 1차적으로 문법적 요소를 기준으로 10개의 범주를 나누었으며, 2차적으로 기능에 따라 레이아웃(layout), 이름(naming), 제어흐름(control flow), 데이터 흐름(data flow)과 같이 4가지 범주로 분류하였다. CStyler는 새롭게 정의된 규칙을 포함하여 총 137개의 코딩 규칙들을 검사한다.

문양선의 연구[11]에서는 인지심리 이론[12,13]에 근거한 객체지향 설계 기법 및 코딩 스타일 지침을 제안하였다. 이 연구에서는 인지심리 이론 중 7±2 청크(chunk) 이론과 학생들을 대상으로 한 인지 실험을 통해 유도된 내포구조(nested structure) 한계 수 3을 바탕으로 객체지향 소프트웨어의 성분들(메소드, 클래스, 메시지, 클래스 상속구조)의 구성 및 관계에 대한 지침들을 정의하고 이를 지원하는 도구를 개발한 바 있다. 7±2 청크 이론은 G. Greeno[12]와 R. Mayer[13]가 주장한 프로그램 의미의 인지과정에 바탕을 두고 있다. 이 주장은 프로그램 이해과정을 덩이화(chunking)로 보고 있다. 이들 연구진들은 인지심리 이론에 바탕을 둔 코딩 스타일 지침들을 이용하여 역공학 시각화 도구도 개발하였다[14]. 역공학 시각화 도구는 프로그램의 이해도 증진을 통해 소프트웨어의 유지보수를 쉽게 한다.

3. 코딩 스타일 지침

본 절에서는 프로그래밍 교육에 활용 가능한 코딩 스타일 지침들을 정의한다. 이를 위해 관련 연구에서 소개한 코딩 표준들과 CStyler 연구에서 정의한 코딩 스타일 지침들을 기반으로 프로그래밍 언어를 처음 시작하는 학생들에게 적합한 코딩 스타일 지침을 새롭게 정의한다.

먼저, Oman과 Cook를 살펴보면 코딩 스타일을 구성하는 요소들을 기능적인 범주로 분류하는 방법을 제시하고 있다[15]. 이들은 코딩 스타일 요소를 일반 범주(general practices), 편집 스타일(typographic style), 제어 구조 스타일(control structure style), 정보 구조 스타일(information structure style)의 4가지 범주로 나눈다. 그리고 각각의 범주에 대한 정의와 분류 기준, 하위클래스 목록, 스타일 요소의 응용 예제들을 정의하고 있다. 표 1은 각 범주에 대한 정의를 보여준다.

표 1 Oman과 Cook의 스타일 범주 정의

범주	내용
일반	스타일에 직접 영향을 끼치는 프로그래밍 프로세스와 관계된 규칙
편집	프로그램 실행에 영향을 주지 않고, 코드 레이아웃과 코드의 주석에 영향을 주는 스타일 특성
제어구조	제어흐름 생성, 프로그램 또는 시스템을 구성하는 알고리즘 구현 방법에 관련된 스타일 특성
정보구조	데이터구조와 데이터흐름 기법들의 선택과 사용에 관련된 스타일 특성

지금까지 제시된 코딩 스타일 표준들은 들여쓰기, 명명법, 주석과 같은 편집 스타일 규칙들뿐만 아니라 제어흐름이나 데이터 흐름 등과 같은 고수준의 프로그래밍 개념이 필요한 규칙들도 많이 포함하고 있다. 앞서 언급된 Li의 연구에서처럼 이러한 고수준의 규칙들은 프로그래밍 언어를 처음 시작하는 학생들의 수준에는 다소 어려울 수 있으며 이로 인해 코딩 스타일 지침을 준수하는 것 자체를 기피하게 만들 수도 있다. 따라서 본 논문에서는 코딩 스타일 분석 항목을 편집 스타일에 관련된 요소들로 제한한다. 표 2는 본 논문에서 정의한 코딩 스타일 지침에 대한 몇 가지 예를 보여준다.

표 2 코딩 규칙의 예

분류	코딩 지침
배치규칙	복합문의 내부 코드들은 1 탭 씩 들여쓰기 함수 끝난 후 1줄 이상 공백 줄 삽입
문장규칙	if 문의 조건식에 대입문 사용하지 않기. do-while 문에 항상 중괄호가 나타나도록.
명명규칙	함수, 변수명의 길이는 3~31 글자로 제한 변수명의 첫 문자는 소문자
공백규칙	할당문 '=' 양쪽에 공백 추가 이항 연산자는 피 연산자와 공백으로 분리

4. 코딩 스타일 시각화 시스템

본 절에서는 프로그램 소스코드로부터 코딩 스타일 분석 결과를 얼굴 이미지를 이용해 시각화하는 코딩 스타일 시각화 시스템인 *StyleVisualizer*를 제안한다. *StyleVisualizer*의 코딩 스타일 시각화는 크게 세 단계

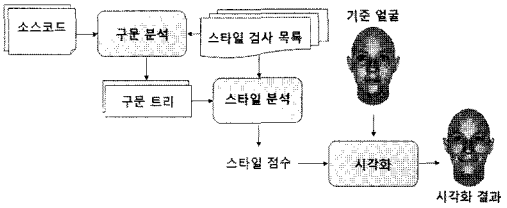

그림 1 *StyleVisualizer* 시스템 구성도

로 진행된다. *StyleVisualizer*는 먼저 프로그램 소스코드를 입력으로 하여 구문 분석과 스타일 분석을 거쳐 소스코드에 대한 스타일 점수를 계산한다. 마지막으로 스타일 점수를 이용해 얼굴 이미지로 시각화한다. 그림 1은 Style-Visualizer의 시스템 구조도를 보여준다.

그림 1에서 구문 분석 모듈은 C/C++ 프로그램 소스코드를 입력으로 하여 구문 트리를 생성한다. *StyleVisualizer*는 OpenC++ 파서를 이용해 구문트리를 생성한다. OpenC++는 C++의 MOP(Meta-Object Protocol) 지원을 위해 제작된 파서로서 새로운 C++ 문법(syntax)이나 및 행동(behavior)을 정의할 수 있도록 함으로써 C++를 확장한 메타프로그래밍을 가능하도록 한다. *StyleVisualizer*에서는 OpenC++를 확장하여 편집 스타일에 관련된 요소들을 검사한다. 여기서 편집 스타일은 구문 트리를 생성하는 과정 중에 어휘분석 모듈에서 검사한다. 그 외의 규칙 검사는 파서를 통해 생성된 구문 트리를 이용한다.

스타일 분석 모듈에서는 구문 트리와 코드 스타일 검사 목록을 이용하여 해당 코드가 코딩 규칙에 맞게 작성되었는지를 검사하고 이를 이용해 스타일 분석 테이블을 만든다. *StyleVisualizer*의 스타일 검사 목록은 코딩 스타일 지침에 해당하는 것으로 3절에서 설명한 것처럼 편집 스타일에 속하는 21가지 항목으로 구성된다. 스타일 분석 테이블은 코딩 규칙, 검사 횟수, 에러 횟수로 구성되며, 검사 횟수는 입력된 소스코드에서 해당하는 코딩 규칙이 나타난 횟수를 나타내며, 오류 횟수는 해당 규칙을 지키지 않은 횟수를 의미한다. 전체 구문 트리의 탐색이 끝나면 스타일 분석 테이블이 완성되고 이것을 이용해 스타일 점수를 계산한다.

마지막으로 시각화 단계에서는 전 단계에서 계산된 스타일 점수를 이용해 기준 얼굴의 표정을 변형함으로써 최종 얼굴 이미지를 생성한다. *StyleVisualizer*는 FaceGen Modeller 2.0을 이용해 시각화에 사용되는 얼굴 이미지를 생성한다. 여기서 얼굴 표정은 무표정한 얼굴을 기준으로 보간을 통해 생성된다. 시각화 결과는 이미지의 얼굴 표정이 웃는 모습에 가까울수록 지침을 잘 따랐음을 의미하며, 찡그린 표정일수록 지침에 따르지 않았음을 의미한다. 그림 2는 C언어로 작성된 소스코드

```
void calcstyle(void)
{
  int i, g, c=0;
  float v=0;
  for (i=0;i<RULE_NUM;i++) {
  gstStyle[i].calcErrorRate();    c+=gstStyle[i].isFound();
  }
  for (i=0;i<RULE_NUM;i++) value+=gstStyle[i].getErrorRate();
  if(c==0) return;
  else {
  g=(int)((v/c)*100.0);
  g=100-g;
  showFace(g);
  }
}
```

(a) 코딩 지침을 잘 지키지 않은 예

```
void CalcStyle(void)
{
  int   i;
  int   nGrade;
  int   nCount = 0;
  float fRate = 0.0;

  for (i = 0; i < RULE_NUM; i++)
  {
    gstStyle[i].calcErrorRate();
    nCount += gstStyle[i].isFound();
  }

  for (i = 0; i < RULE_NUM; i++)
  {
    value += gstStyle[i].getErrorRate();
  }

  if(nCount == 0)
  {
    return;
  }
  else
  {
    nGrade = (int)((fRate / nCount) * 100.0);
    nGrade = 100 - nGrade;
    ShowFace(nGrade);
  }
}
```

(b) 코딩 지침을 잘 지킨 예

그림 2 소스코드 스타일 분석 및 StyleVisualizer를 이용한 시각화 결과

와 *StyleVisualizer*를 이용해 시각화된 얼굴 이미지를 보여준다.

그림 2(a)는 코딩 스타일 규칙을 제대로 지키지 않은 가독성이 낮은 소스코드의 예로 소스코드에서 이항 연산자의 좌우에 공백삽입, 들여쓰기 등이 제대로 지켜지지 않았음을 볼 수 있다. 그림 2(b)는 그림 2(a)의 소스코드를 코딩 스타일 규칙에 따라 수정하여 시뮬레이션한 결과를 보여준다.

본 논문의 연구에는 포함이 되지 않았지만, 본 연구팀에서는 프로그래밍 과제에 대해 웹을 통해 소스코드를 제출하고 자동으로 채점을 할 수 있는 온라인 자동 과제 채점 시스템을 구현하여 C/C++ 및 자바 강의에서 사용하고 있다. *StyleVisualizer*는 프로그래밍 관련 교과목을 수강한 학생들이 제출한 소스코드에 대해 코딩 스타일 점수에 대한 시각화 결과를 피드백 한다. 학생들은 *StyleVisualizer* 시각화를 통해 가독성 높은 소스코드를 작성할 수 있다.

5. 실 험

코딩 스타일 시각화 시스템의 교육적 효과를 알아보기 위해 *StyleVisualizer*를 이용한 비교 실험을 진행하였다. 실험은 부산대학교 실용 컴퓨터 교과목 수강생들을 대상으로 진행하였다. 그리고 실험데이터는 한 학기 동안 수강생들이 제출한 실습 프로그램을 수집하여 생성하였다.

실용 컴퓨터 교과목은 컴퓨터 및 전산학 전공이 아닌 학생(학부 1~3학년)을 대상으로 C언어 프로그래밍을 주제로 이론과 실습 강의가 진행된다. 그러므로 수강하는 학생들은 대부분 프로그래밍을 처음 배우는 학생들로 C언어 문법에 익숙하지 않다. 표 3은 코딩 스타일 지침 실험 대상자 및 실험진행에 대한 정보를 보여준다.

표 3 코딩 스타일 검사 실험대상

항 목	내 용
교과목	실용 컴퓨터(C언어 프로그래밍)
기간	2010. 03~2010. 06(1학기)
대상	2 분반(C113과 C114)
전공	컴퓨터 및 전산학 비전공
학년	1~3 학년
인원	30명(C113), 35명(C114)
수준	초급(프로그램을 처음 배움)

코딩 스타일 지침에 대한 실험은 다음과 같이 진행하였다. 먼저, 두 실험 그룹에 대해 코딩 스타일과 *StyleVisualizer*에 대한 교육 없이 실습을 진행하였다(C113-A 및 C114-A 그룹). 다음으로 C113 그룹에 대해서는 코딩 스타일 지침에 대한 교육만 진행했으며(C113-B), C114에 대해서는 코딩 스타일 교육과 더불어 *StyleVisualizer*를 사용하도록 하였다(C114-B). 실험은 코딩 스타일에 대한 교육 이전에 두 번의 실습을 진행했으며, 교육 후에는 다섯 번의 실습을 진행하였다. 코딩 스타일에 대한 평가는 두 경우에 대해 코딩 스타일 검사를 진행하여 개선 정도를 비교하였다. 표 4는 실험결과를 보여준다.

표 4 *StyleVisualizer*를 이용한 코딩 스타일 교육 후 검사 결과 : 실험 대상은 두 분반(C113, C114)을 대상으로 진행했으며, C113-A와 C114-A는 코딩 스타일 교육 전을 나타내고, C113-B는 코딩 스타일 교육 후 실험결과를 나타내고, C114-B 그룹은 교육과 함께 *StyleVisualizer* 사용 후의 실험결과를 나타낸다.

프로그램 그룹	프로그램 수	문장			공백			명명규칙		
		빈도	오류	비율	빈도	오류	비율	빈도	오류	비율
C113-A	58	117	59	50.43	588	484	82.31	952	83	8.72
C113-B	139	122	20	16.39	278	164	58.99	2376	530	22.30
C114-A	69	88	72	81.81	624	508	81.41	1146	99	8.64
C114-B	166	105	17	16.19	342	158	46.2	2900	517	17.83

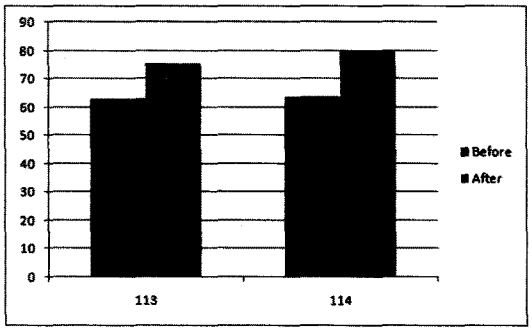

그림 3 과제별 코딩 스타일 평균 점수

실험결과 코딩 스타일 지침 오류 비율은 *StyleVisualizer*를 사용하지 않고 교육만 진행했을 때 약 15% 감소한 반면, *StyleVisualizer*를 적용했을 경우 약 30% 이상 감소하였다. *StyleVisualizer*를 적용했을 때, 각 카테고리 별로 문장 규칙의 경우 약 65%, 공백 규칙의 경우 약 35% 이상 감소하였다. 하지만 명명 규칙의 경우는 약 11% 정도 오류 비율이 증가하였다. 명명 규칙이 증가한 원인을 살펴본 결과, 프로그램 소스코드에서 루프(loop)에서 'i' 나 'j' 와 같이 길이가 3이하인 지역변수의 사용이 많아 오류 발생 비율이 증가하였다.

그림 3은 과제별 코딩 스타일 평균 점수를 보여준다. C113과 C114 분반 모두 코딩 스타일 지침 교육과 *StyleVisualizer*를 적용했을 때, 평균 점수가 높아진 것을 볼 수 있다. 또한, *StyleVisualizer*를 이용한 실습을 마친 후, 학생들에게 코딩 스타일 지침을 고려하여 소스코드를 작성했을 때와 고려하지 않았을 때에 대하여 설문조사를 진행하였다. 학생들은 코딩 스타일에 따라 소스코드를 처음 작성할 때에는 시간이 오래 걸렸지만, 디버깅에서는 프로그램 수정이 용이했다고 느꼈다. 그리고 *StyleVisualizer*의 얼굴 이미지를 이용한 시각화 결과에 대해 학생들은 텍스트 형식의 컴파일 오류 레포팅보다 쉽고 거부감이 없었으며, 실습 과제에 대한 해답을 찾은 후에도 웃는 표정의 시각화를 만들기 위해 코딩 스타일 지침에 맞게 수정한 학생들이 많았다는 것을 알 수 있었다. 실험결과 *StyleVisualizer*의 얼굴 이미지 시각화 기법은 프로그래밍을 처음 접하는 학생들에게 자신의 소스코드에 대한 분석 결과를 피드백 해줌으로써 올바른 프로그래밍 습관을 기르고 양질의 소스코드를 작성하도록 유도하는데 교육적 효과가 있음을 알 수 있었다.

5. 결론 및 향후 연구

본 논문에서는 프로그램 소스코드의 코딩 스타일을 분석하여 시각화하는 시스템을 제안하였다. 제안된 코딩 스타일 시각화 시스템은 C/C++ 언어로 작성된 프로그램 소스코드를 입력받아 코딩 표준을 얼마나 잘 준수하는지에 대해 평가한다. *StyleVisualizer*에서는 프로그래밍 언어를 처음 시작하는 학부 학생들의 수준에 맞추어 들여쓰기, 이름규칙과 같은 편집 스타일에 대한 항목만으로 코딩 지침을 정의하였다.

코딩 스타일에 대한 분석 결과는 수치나 여러 가지 측정 항목으로 제시하는 기존의 평가 프로그램들과는 달리 얼굴 표정 이미지로 시각화하였다. 결과 얼굴 이미지가 웃는 얼굴일수록 코딩 표준을 잘 따랐음을 의미하고, 찡그린 얼굴일수록 코딩 표준을 지키지 않았음을 의미한다. 사람들과 친숙한 얼굴 이미지를 이용해 코딩 스타일을 시각화함으로써 학생들로 하여금 자신의 코딩 습관을 좀 더 직관적으로 파악할 수 있도록 하였다.

향후 연구로는 기능적으로 분류된 코딩 규칙들을 반영한 시각화 기법에 대한 연구와 프로그래밍 언어의 구문과 실행시간 스택과 연관된 코딩 스타일에 대한 시각화에 대하여 연구를 진행할 것이다.

참 고 문 헌

[1] H. Cho, M. Hwang, "A Study on the C Source Code Restructuring for Effective Maintenance," *Proc of the KIISE Korea Computer Congress 1996*, vol.23, no.2, pp.1573-1576, 1996. (in Korean)

[2] T. Tenny, "Program Readability: Procedures Versus Comments," IEEE Transactins on Software Engineering, vol.14, no.9, pp.1271-1279, 1988.

[3] H. Sutter, A. Alexandresuc, "C++ Coding Standards: Rules and Guidelines for Writing Programs," Addison Wesley, 2004.

[4] R. Stallman, et al., "GNU Coding Standards," 2007.

[5] L. Hatton, "Language Subsetting in an Industrial Context: A Comparison of MISRA C 1998 and MISRA C 2004," *Information Software Technology*, vol.49, no.5, pp.475-482, 2007.

[6] X. Li and C. Prasad, "Effectively teaching coding standards in programming," In *Proceedings of the 6th Conference on Information Technology Education*, pp.239-244, 2005.

[7] L.W. Cannon, et al., "Recommend C Style and Coding Standards," AT&T Bell Labs, 1997.

[8] X. Fang, "Using a Coding Standard to Improve Program Quality," In *Proceedings of the 2nd Asia-Pacific Conference on Quality Software (APAQS'01)*, pp.73-78, 2001.

[9] S. Cho, Y. Moon, C. Yoo, Y. Kim, O. Chang, "Object-Orented Design and C++ Programming Style Guidelines Offer Tool," *Proc of the KIISE Korea Computer Congress*, 1996, vol.23, no.2, pp.1445-1448, 1996. (in Korean)

[10] J. Hwang, "Design and Implementation of a C

Coding Style Checker," *Journal of KSCI*, vol.13, no.2, pp.31-40, 2008.
[11] Y. Moon, C. Yoo, O. Chang, "Object-Orient Design and Programming Style Guidelines based on Congnitive Psychology Theories," *Journal of KIISE(B)*, vol.25, no.3, pp.530-542, 1998.
[12] G. Greeno, "The Structure of Memory and the Process of Problem Solving," *Contemporary Issues in Cognitive Psychology*, pp.23-45, 1973.
[13] R. Mayer and B. Shnejderman, "Syntactic/Semantic Interactions in Programmer Behavior: Model and Experimental Result," *International Journal of Computer and Information Sciences*, vol.8, pp.213-238, 1979.
[14] Y. Moon, J. Kim, H. Cho, C. Yoo, Y. Kim, O. Chang, "A Reverse Engineering Visualization Tool for Increasing Understandability of C++ Programs," *Journal of KIISE(C)*, vol.1, no.2, pp.160-171, 1995
[15] P.W. Oman, C.R. Cook, "A Taxonomy for programming style," *Proceedings of the 1990 ACM Annual Conference Cooperation*, pp.244-250, 1990.

지 정 훈
2003년 경성대학교 컴퓨터공학 학사. 2005년 경성대학교 컴퓨터공학 석사. 2010년 현재 부산대학교 컴퓨터공학과 박사과정 관심분야는 프로그래밍언어 및 컴파일러, 프로그램 표절검사, 자바가상기계, 프로그램 시각화

이 윤 정
1995년 부경대학교 전자계산학 학사. 1999년 부경대학교 전산정보학 석사. 2008년 부경대학교 전자계산학 박사. 2010년 현재 부산대학교 U-Port 사업단 박사후연구원. 관심분야는 얼굴 애니메이션, 웹 시각화

우 균
1991년 한국과학기술원 전산학 학사. 1993년 한국과학기술원 전산학 석사. 2000년 한국과학기술원 전산학 박사. 2010년 현재 부산대학교 정보컴퓨터공학부 부교수 관심분야는 프로그래밍언어 및 컴파일러, 함수형 언어, 그리드컴퓨팅, 소프트웨어 메트릭, 소프트웨어 테스팅 등