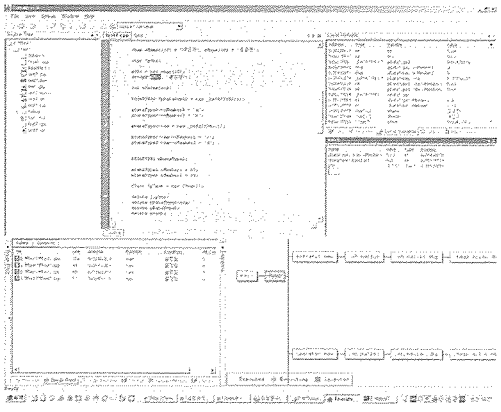
Tomato "Software Analysis Program"

OpenGL로 구현된
Sequence Level Debugging 환경



이 프로그램은 소프트웨어 개발 과정에서 유용하게 사용 될 수 있는 Debugger이다.
프로그램의 전반적인 기능과 UI는 기존의 Debugger 사용자들이 불편함 없이 다룰 수 있도록 하기 위해 Visual Tool에서 제공해주는 Debugger와 거의 동일하게 구현되었으며, Tomato 프로그램의 핵심이라 할 수 있는 기능으로 일반적인 Debugger의 Debugging 환경 보다 한 단계 더 추상화된 Debugging 환경을 제공해 주기 위해 Sequence Level Debugging 환경을 구현 하였다. 이 환경은 개발자가 프로그램을 Debugging하는 과정에서 현재 어느 시점에서 Debugging을 시작 했는지 그리고 CPU 입장에서 앞으로 어떤 절차를 거쳐 프로그램이 수행 되었는지를 OpenGL로 구현된 Sequence Viewer를 통해 한눈에 알아 볼 수 있는 비주얼한 환경을 제공해준다. 프로그램 사용자는 함수 단위로 다루어지는 Tree Node로 표현된 Sequence Viewer에서 원하는 Node를 마음대로 선택 할 수 있으며, 선택된 Node에 해당하는 함수에 대한 소스코드 상에서의 위치를 검색해준다.
또한 수많은 Node들이 펼쳐 져있을 경우 화면을 마음대로 이동할 수 있으며 원하는 Node를 찾는데 걸리는 시간을 단축시키기 위해 Sequence Viewer에 대한 자체적인 검색기능을 제공해준다.
위 개념은 기존 Debugger 환경에 Source Insight 프로그램의 장점을 포함한 형태로 Source Insight에서의 Source Code 수준에서 처리되는 정적인 구조가 아닌 CPU의 실행 단계에서 처리되는 동적인 구조로 구현되었다.
또한 Debugging 과정에서 현재 운영체제 내에서 수행중인 모든 프로세스 정보, 스레드 정보, Heap Memory정보, 페이지 테이블 정보 등을 살펴 볼 수 있는 기능을 제공해 줌으로써 개발자들에게 많은 도움을 줄 수 있을 것이다. 현재의 소프트웨어 개발 과정은 추상화 개념의 발전으로 과거에 비해 개발 과정이 쉬우며, 앞으로는 더욱더 쉬운 환경이 제공 될 것이다.
소프트웨어 개발 뿐 만이 아니라 소프트웨어 유지 및 보수 도구 역시 비슷한 과정으로 발전 될 것 이라고 생각하며, Tomato 프로그램은 이 부분에 있어서의 초석이 될 것이다.

정보통신부장관상

Software Analysis Program(Tomato)

1. 작품명 : Software Analysis Program(Tomato)

2. 제작자 : 조선대학교

대표자 : 대인기
개발 참여자 : 김근영, 조금현
주소 : 광주광역시 동구 서석동 375번지 조선대학교
공대 2호관 11층 컴퓨터공학부
전화 : 062) 230-7381
휴대폰 : 016) 750-6468
email : daeinki@msn.com

3. S/W 요약설명

소프트웨어 개발 과정에서 소스코드 분석 및 디버깅 환경 제공

3.1 개발 배경

Application 개발과정에서 수행하는 디버깅 과정 그리고 개발 완료 후 프로그램의 완성도를 높이기 위해 발생할 수 있는 장기적인 오류를 수정하기 위한 Testing과정 그리고 마지막으로 개발된 프로그램에 대한 문서화 작업등 개발 진행부터 완료까지 몇 단계의 과정을 수행하게 된다.

디버깅 과정은 어플리케이션을 개발 하는 과정에서 개발자가 작성한 소스코드를 메모리 수준에서 정확한 수행결과를 검증해보는 과정

으로써 개발 툴 상에서 제공해주는 디버깅 도구 및 기타 여러 디버깅 도구들이 이용되고 있다.

Testing과정은 가장 많은 시간과 비용을 필요로 하는 과정으로써 개발완료 후 완성된 프로그램을 수행해 보면서 문제가 발생되는 부분을 체크하여 보다 완성도 높은 프로그램을 만들어 내기 위한 과정이다.

문서화 작업은 개발이 완료된 후 개발된 프로그램에 대한 요구사양서 및 상세 설계서등의 문서를 작성해 둠으로써 차후에 개발된 프로그램을 수정 및 갱신 할 때 쉽게 접근 할 수 있도록 하는 과정이다.

위 3가지 과정은 어느 하나 소홀이 할 수 없는 과정으로써 각 과정을 수행함에 있어서 도움이 될 만한 도구들이 수없이 존재하고 있다.

실제로 현재 개발자들은 디버거 외에 UML툴과 같은 추상화 객체를 다루는 툴들을 많이 이용하고 있다.

이런 툴은 소프트웨어 개발 전에 프로그램 설계 도구로 많이 사용되고 있으며 앞으로 개발자들은 더욱 향상된 개발 환경을 원하게 될 것이다.

이런 개발자들의 요구의 요구를 충족시키기 위해서 다음과 같은 방향으로 개발을 결정하게 되었다.

수행중인 프로그램의 함수수준에서 사용한 메모리 현황을 표현 수행중인 프로세서를 일시정지 시키고 현재 수행되었던 각각의 함수에 의해 사용되어진 메모리 현황을 변수 수준에서 확인 및 변경 가능하도록 함으로써 **수행중인 상태에서 Testing이 가능**하게 한다.

또한 현재 운영체제에서 수행되고 있는 모든 프로세스 목록, 대상 프로세스에 대한 스레드 정보, 힙 메모리 정보, Page Table Entry 정보, 적재된 모듈 정보를 개발자가 알아볼 수 있도록 하여 개발하는데 참고가 되도록 하며, **수행중인 프로그램의 실행과정을 함수 수준의 시퀀스 단위로 그려냄**으로써 **개발자들이 소스코드를 분석하는데 드는 시간과 노력**을 줄일 수 있도록 구현한다.

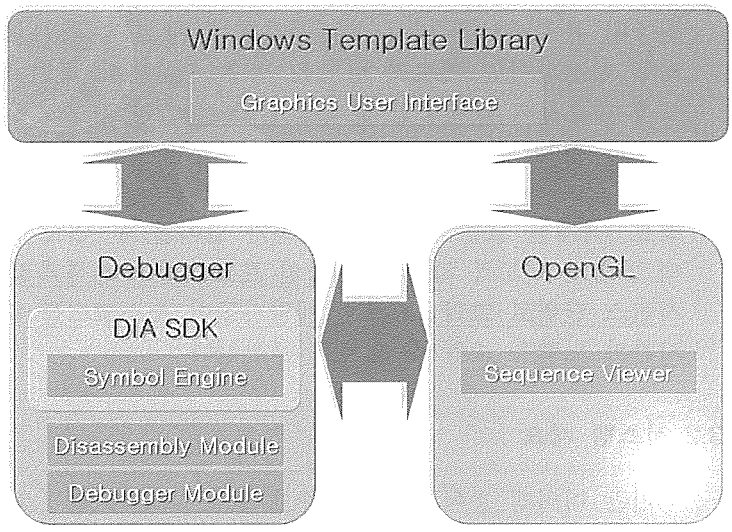
3.2 시스템 개요

기존 디버깅 툴에서 제공해주고 있는 디버깅 환경을 보다 개선하여 한 단계 더 추상화된 수준에서 디버깅 환경을 제공해 줌으로써, 소프트웨어를 개발하는 개발자들로 하여금 개발 시간 및 개발 비용을 최소화 할 수 있도록 개발된 소프트웨어 분석 도구이다.

3.3 시스템 특징

이 프로그램의 대부분의 기능은 기존 Visual C++, Visual Studio.NET에서 제공해주고 있는 디버거와 기능적으로 모두 동일하다. 이는 개발자들에게 익숙한 디버깅 환경을 제공해주기 위해서 기본적으로 구현한 기능이며, 추가적으로 소스코드 보다 한 단계 더 추상화된 개념까지 디버깅 환경을 끌어올려 소프트웨어 개발 시에 상당한 도움이 되도록 개발 되었다. 현재 소프트웨어 개발자들은 자신의 개발 능력을 최대로 끌어올리기 위해 디버거 뿐 만이 아니라 설계 도구를 포함한 가능한 모든 도구들을 동원하고 있다. 이 프로그램은 이러한 개발자들의 요구를 만족시키기 위해 일반적으로 설계 도구에서 표현되는 Sequence 형태를 프로그램을 실행하는 과정에서 실시간으로 분석하여 제공해 줌으로써 개발 시간 및 비용을 최소화 할 수 있을 것이다.

또한 Tree 형태의 Sequence Node들을 현재 수행중인 노드, 수행되었던 노드, 사용자가 소스코드를 검색하기 위해 선택한 노드들로 구분하여 보여줌으로써 Sequence Viewer의 활용을 극대화 하고 있다.



■ GUI

- MFC를 사용하지 않고 WTL기반으로 제작되어서 MFC에 비해 상대적 으로 가볍게 제작되었다.

■ Debugger

- 디버거는 Symbol Engine, Disassembly Module, Debugger Module로 다시 나눠지며, Symbol Engine은 MS에서 제공해주는 DIA SDK를 wrapper로써 라이브러리로 제작 되었다.

■ Sequence Viewer

- Sequence Viewer는 OpenGL기반으로 제작되었으며, 이 모듈 역시 라이브러리화 하였다.
  GUI와 Debugger모듈간의 인터페이스는 FACADE 패턴을 적용 GUI모듈 내에서의 복잡성을 최소화 하여 제작되었다.

3.4 제품 구성

.Net Framework
Symbol Engine, Sequence Viewer Library
실행 파일 이미지

3.5 프로그램 구성 및 주요기능

1) Debugger

■ Multi Break Pointer 기능

- 디버깅을 원하는 소스코드의 원하는 라인에 여러 개의 브레이크 포인터를 지정 및 편집이 가능하다.

■ 3단계 Debugging 환경 지원

- Assembly code, Source code Debugging 환경뿐 만이 아니라 Sequence Level Debugging 환경지원

■ Step Over, Step Into 기능

- Assembly code, Source code 그리고 Sequence level에서 한 단계씩 디버깅 할 수 있는 Step Over 기능과 호출하려는 함수의 내부로 CPU를 이동 시킬 수 있는 Step Into기능을 지원

■ Function 단위의 Local variables 정보 표현 및 Watch Window 지원

- 분석중인 프로그램의 Function단위의 Local variable names들

을 Structure, Class Object의 맴버 변수 이름을 개발자가 작성하는 형태로 재가공 하여 표현하며, 객체 또는 변수 명에 대한 메모리 내용을 분석과정에서 실시간대로 확인 할 수 있다.

■ Register 정보 표현
- 분석 과정에서 범용 레지스터 및 32비트에서 추가된 모든 레지스터 정보를 확인 할 수 있다.

■ Memory 정보 표현
- 분석 과정에서 원하는 메모리 영역을 Hex code 및 Ascii code형태로 확인 할 수 있다.

■ Back Tracing 및 Auto Tracing 기능
- 현재 정지된 위치까지 함수가 호출된 과정을 확인 할 수 있으며, 사용자가 원하는 위치 혹은 프로그램이 종료 될 때까지 CPU내에서 처리되는 함수의 호출 과정을 한 눈에 확인 할 수 있다.

2) Sequence Viewer

■ Sequence Viewer 기능
- 분석중인 프로그램이 수행 중에 호출되었던 함수들을 OpenGL로 제작된 Viewer를 통해 Tree구조 형태로 표현 할 수 있다.

■ Tree노드에 대한 함수 명 검색 기능
- 디버깅 진행 과정이 표현된 Viewer에서 전체 함수 목록 중 원하는 함수를 빠르게 검색할 수 있다.

3) 부가적인 기능

■ 현재 운영체제에서 수행중인 모든 프로세스 정보 및 가상메모리 정보, 라이브러리, 스레드 그리고 Heap memory 정보 등을 확인 할 수 있다.

4. 개발단계별 기간 및 투입인원수

개발단계	개발시간	인원	비고
시스템 설계	04.04.01 ~ 04.04.30	3	기본/상세설계
프로그래밍	04.05.01 ~ 04.06.30	3	소스코드 작성
GUI 수정	04.07.01 ~ 04.07.31	3	사용자 인터페이스 개선
Testing	04.08.01 ~ 04.08.31	3	버그 수정 및 기능 추가
문서화 작업	04.08.01 ~ 04.08.31	3	디버거 개발 방법 상세 문서화
계	5개월	3	

5. 사용 또는 개발언어, TOOL

Microsoft Visual Studio .Net 2003
Source Insight 3.5
Photoshop 7.0

6. 사용 시스템

사용OS	Microsoft Windows 2000/XP
CPU	펜티엄Ⅱ166MHz 이상
모니터	15인치 이상
메모리	64MB 이상
FDD	1.44MB
HDD	1GB 이상
VGA	SVGA 이상
Sound Card	Sound Blaster 호환