CHILL 기반의 SDL 심벌 디버거의 구현

최원혁° 김성재 김승호
경북대학교 컴퓨터공학과
{whchoi, sjkim}@borami.ce.knu.ac.kr
shkim@bh.knu.ac.kr

Implementation of SDL Symbol Debugger based on CHILL

Won-Hyuk Choi Seong-Jae Kim Sung-Ho Kim

Department of Computer Engineering, Kyungpook National University

요 약

현재의 소프트웨어 개발은 구현 중심에서 설계 중심으로 변하고 있다. 설계 중심의 개발은 구현 프로그램의 자동생성을 바탕으로 설계 수준에서 정형화된 명세를 검증함으로써 설계와 구현 사이의 불일치를 없앨 수 있고, 유지보수가 효율적이다. 본 논문에서는 설계 중심의 개발 방법을 구축하기 위한 SDL과 CHILL의 통합된 개발환경에서 디버깅 환경을 제공하기 위한 도구를 제안하고, 정합, 추적, 제어 인터페이스를 통하여 실행 중인 CHILL 프로그램의 수행을 시스템 구현의 전단계 설계 문서인 SDL/GR의 심벌로 디버깅하는 SDL 심벌 디버거를 구현하였다.

1. 서론

최근의 통신 시스템들은 점점 대형화되고 있기 때문에 긴 생명주기(life cycle)를 가지고 대규모 개발 인력을 필요로 한다. 이런 현상은 통신 시스템뿐만 아니라 다른 대형의 실시간 분산 시스템에서도 발생하며, 이로 인해 시스템의 초기 개발 비용보다는 유지보수에 더 많은 비용이 투자된다.

따라서, ITU-T(International Telecommunication Union - Telecommunication Standardization Sector)는 대형 시스템의 개발시간과 비용을 줄이고 시스템의 질적 향상을 위해 생명주기의 각 단계에 따라 적용할 수 있는 언어로, 시스템의 명세(specification)와 기술(description)을 위해 SDL (Specification and Description Language)을, 구현을 위해 CHILL(CCITT High Level Language)을 권고하였다[1,5,7].

소프트웨어 개발에서 구현 단계 이전에 SDL을 통한 시스템의 명세는 명세 및 기술 문서로부터 소프트웨어의 정형적인 검증과 확인을 가능하게 하고, 이를 바탕으로 구현 프로그램의 자동 생성을 가능하게 한다. SDL의 자동 번역은 시스템 개발 기간을 축소시키고, 개발된 시스템의 유지보수를 용이하게 할 수 있는 환경을 제공할 뿐만이 아니라, 패턴과 코드의 재사용, 생산성의 향상 등의 장점도 제공한다. 이를 통하여 사용자의 요구 사항에 따른 설계 문서와 구현 프로그램 간의 불일치를 해결하고 일관성을 유지할 수 있다[2,3].

Telelogic과 Verilog 같은 CASE 도구 개발 업체에서는 통신 및 실시간 분산 시스템의 개발을 위해 SDL을 기반으로 하는 시스템 명세 및 기술의 생성에서 SDL 수준의 디버깅과 C나 C++로 수행코드의 생성까지 일관성을 유지할 수 있는 다양한 CASE 도구들을 제공한다[3,4]. 그러나, 대형 교환기와 같은 실시간 병렬 분산 시스템을 개발하는데 적합한 CHILL언어와 통합된 CASE 도구는 지원하지 않는다.

본 논문에서는 SDL과 CHILL의 통합된 개발환경을 제공하기 위한 도구인 SDL 심벌 디버거를 구현한다. SDL 심벌 디버거는 실행 중인 CHILL 프로그램의 수행을 시스템 구현의 전단계 설계 문서인 SDL/GR의 심벌로 디버깅하는 CASE 도구이다. 심벌 추적을 위한 정보는 SDL로 기술된 명세에서 구현 프로그램인 CHILL로 변환할 때 SDL/GR 정보로부터 정합 인터페이스로 추출되어 생성 프로그램에 함수의 형태로 삽입된다. SDL 심벌 디버거는 실행 프로그램으로부터 전송된 정합 인터페이스를 추적 인터페이스로 해석한 후, PostMaster[8]를 거쳐 SDT(SDL Design Tool)의 SDL 편집기를 제어하여 CHILL 코드의 수행을 시각적인 SDL 심벌의 수준에서 추적한다[4]. 여기에 제어 인터페이스를 추가함으로써 다양한 디버깅 기능을 제공한다. 제어 인터페이스를 통하여 SDL 심벌 디버거는 SDL 심벌 수준 실행 추적(SDL Symbol level execution trace), SDL 심벌 수준 단일 스텝 추적(SDL Symbol level single step trace), SDL 심벌 단위 중지점 설정(SDL Symbol level break point establishment), SDL 심벌에서 CHILL 소스코드로의 참조(SDL Symbol to CHILL source referencing) 그리고 CHILL 소스코드에서 SDL 심벌로의 참조(CHILL source to SDL symbol referencing) 등의 기능을 제공한다. SDL 심벌 디버거의 구현은 설계 수준에서 프로그램을 검증할 수 있게 하고, 설계와 구현언어의 개념 차이에서 발생하는 오동작을 제거할 수 있게 함으로써 전체 개발 시간과 비용을 줄일 수 있다.

2. 개발환경의 통합

2.1 개발환경 통합의 필요성

구현 중심의 소프트웨어 개발 방법은 명세 및 설계 단계와 일관성 없이 소프트웨어를 개발하기 때문에 사용자의 요구사항 변화에 대하여 민감하게 대처하기가 어려우므로 유지보수에 많은 비용이 들어간다. 이와 같은 이유로, 현재의 소프트웨어 개발은 구현 중심에서 설계 중심의 개발 방법으로 변하고 있다. 설계 중심의 개발 방법은 분석과 실행이 가능한 명세와 설계를 통한 구현 프로그램의 자동생성을 통하여 소프트웨어를 개발한다. 설계 수준에서 정형화된 명세를 검증함으로써 설계와 구현 사이의 불일치를 없애고, 사용자의 요구 사항에 민감하게 대처할 수 있으므로 개발 이후의 유지 보수비용을 줄일 수 있다.

이와 같은 설계 중심의 개발 방법을 위한 개발환경은 설계 환경과 구현환경을 연결할 수 있는 새로운 인터페이스 도구들을 추가함으로써 이루어진다. 설계문서를 설계 수준에서 검증할 수 있는 디버깅환경과 설계 문서에서 구현언어로의 자동 번역기가 추가되는 인터페이스 도구들이다[2,3,6].

2.2 SDL 심벌 디버깅 환경

본 논문에서 제안하는 SDL 심벌 디버깅 환경은 그림 1과 같이 이루어진다. SDL로 작성된 시스템 명세로부터 CHILL 실행 프로그램을 생성하는 번역기, CHILL 소스 수준 병렬 디버깅을 지원하는 TECH(Testing Environment for CHILL), 그리고 실행중인 CHILL 프로그램과 연계하여 SDL 심벌 디버깅을 수행하는 SDL 심벌 디버거와 이를 시각적으로 보여주는 SDT의 SDL 편집기로 이루어진 개발 환경이다[3].

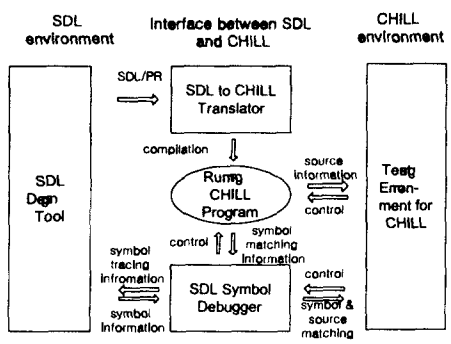

그림 1 SDL 심벌 디버깅 환경

SDL 심벌 디버깅 환경은 기존의 TECH만을 사용한 변환된 CHILL 프로그램의 소스 수준 디버깅이 SDL 시스템 명세와 연계되지 않은 채로 수행됨으로써, 프로그램의 이해도가 떨어지고 설계 문서와의 일관성을 저하시키는 단점을 보완한다. 이러한 변환된 프로그램의 이해도와 설계문서와의 일관성 문제는 CHILL 프로그램의 디버깅시에 프로그램의 수행과 디버깅 작업을 SDL 명세의 심벌 수준에서 수행함으로써 해결한다.

3. 심벌 디버거

SDL 심벌 디버거는 실행 중인 CHILL 프로그램의 수행을 시스템 구현의 전단계 설계 문서인 SDL/GR의 심벌로 디버깅하는 CASE 도구다. 이는 SDL에서 CHILL로의 자동 번역기와 더불어 설계 중심의 개발 방법 구축을 위해 설계와 구현의 개발 환경을 통합하기 위한 필수적인 도구이다.

3.1 인터페이스의 설계

SDL 심벌 디버거를 구현할 때 생성된 CHILL 프로그램과 SDL 편집기를 연결하고 심벌의 추적과 각 기능들을 제어하기 위해서 정합, 추적, 제어의 세 가지 인터페이스가 필요하다.

정합 인터페이스는 CHILL 코드의 자동 생성시에 생성된 코드와 정합하는 SDT에서의 SDL 심벌 이름과 위치 정보를 가지는 함수의 형태로 삽입된다. 생성된 정합 인터페이스는 SDL 문서 중에서 프로세스에 포함된 부분에만 한정되며 시스템과 블록의 기술 부분은 심벌 추적에 관여하지 않으므로 포함되지 않는다. 이렇게 생성된 정보는 CHILL 프로그램의 수행시에 시그널의 형태로 SDL 심벌 디버거로 전송되고, 심벌 디버거 내에서 추적 인터페이스로 변환됨으로써 심벌의 추적을 가능하게 한다[4].

추적 인터페이스는 SDL 심벌 디버거의 실시간 통신기가 실행 프로그램으로부터 수신한 정합 인터페이스를 변환한 정보인데, 이 정보가 PostMaster를 거쳐 SDT의 SDL 편집기를 제어하여 CHILL 코드의 수행을 시각적인 SDL 심벌의 형태로 보여준다[4].

제어 인터페이스는 사용자의 입력을 받아 심벌 추적의 고유 기능을 바탕으로 심벌 디버거의 다양한 기능을 제공한다. 이러한 제어 인터페이스는 외부 인터페이스가 PostMaster로부터 수신한 응답 정보를 실시간 통신기로 전송하는 것을 제어함으로써 SDL 편집기에서 단일 스텝 추적 등의 다양한 디버깅 기능을 제공한다.

3.2 심벌 디버거의 구성 및 기능

SDL 심벌 디버거의 구성은 그림 2와 같이 네 개의 프로세스로 구성된다.

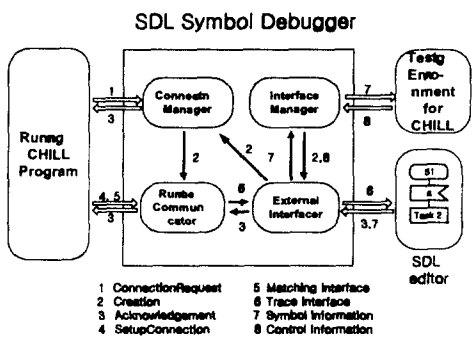

그림 2 SDL 심벌 디버거의 구성

수행 중인 CHILL 프로그램 내의 하나의 프로세스와 SDL 심벌 디버거의 실시간 통신기 프로세스 하나를 연결하는 기능을 수행하는 **연결 관리기(Connection Manager)**, 실행중인 CHILL 프로그램의 프로세스로부터 정합 인터페이스 정보를 수신하고, 이를 해석하여 추적 인터페이스로 변환한 후 외부 인터페이스로 전송을 담당하는 **실시간 통신기(Runtime Communicator)**, 실시간 통신기로부터 수신된 추적 인터페이스를 PostMaster로 전송함에 따라 심벌 추적을 제어하는 **외부 인터페이스(External Interface)**와 사용자의 입력을 받아 제어 인터페이스를 통하여 SDL 심벌 디버거의 각 기능들을 제어하는 **인터페이스 관리기(Interface Manager)**로 구성된다.

SDL 심벌 디버거의 기능은 수행중인 CHILL 프로그램,

TECH와 사용자의 인터페이스간의 통신을 거쳐 SDL 편집기를 통하여 시각적으로 보여진다. SDL 심벌 디버거의 각 기능은 첫째로 가장 중요한 기능인 생성된 CHILL 프로그램의 수행을 SDL 심벌의 형태로 시각화 해주는 **SDL 심벌 수준 실행 추적**, 둘째로 심벌의 단계적인 수행을 가능케 하는 **SDL 심벌 수준 단일 스텝 추적**, 셋째로 중지점 설정을 통해 부분 디버깅을 가능하게 하는 **SDL 심벌 단위 중지점 설정**, 넷째로 TECH와의 통신을 통하여 생성된 코드 수준에서 디버깅을 할 수 있게 하는 **SDL 심벌에서 CHILL 소스코드로의 참조**, 마지막으로 **CHILL 소스코드에서 SDL 심벌로의 참조**로 이루어진다. 그러나, 본 논문에서 구현한 심벌 디버거는 TECH와의 통신의 문제로 인해 CHILL 소스코드에서 SDL 심벌로의 참조 기능은 구현하지 않았다.

3.3 심벌 디버거의 구현

구현된 SDL 심벌 디버거의 사용자 인터페이스는 아래의 그림 3과 같다. 심벌 디버거는 CHILL언어로 작성된 심벌 디버거의 주 프로그램과 TCL/TK로 작성된 사용자 인터페이스로 나누어진다. 사용자 인터페이스의 오른쪽 디버깅 프레임을 통해 심벌 디버거의 기능을 제어하고, Communicator프레임과 Running state프레임을 통하여 심벌 디버거의 동작 상태를 보여준다.

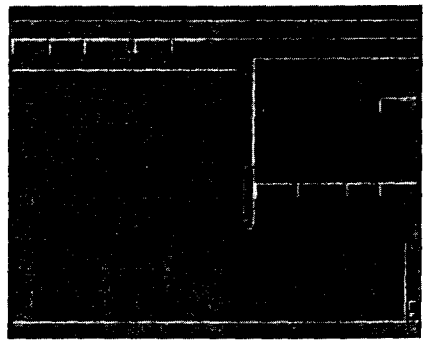

그림 3 SDL 심벌 디버거의 사용자 인터페이스

그림 3은 구현된 심벌 디버거의 사용자 인터페이스를 통해 SDL 심벌 수준 단일 스텝 추적을 수행하는 과정을 보여주고, 그림 4는 SDL 편집기를 통하여 심벌 추적의 예를 보여준다.

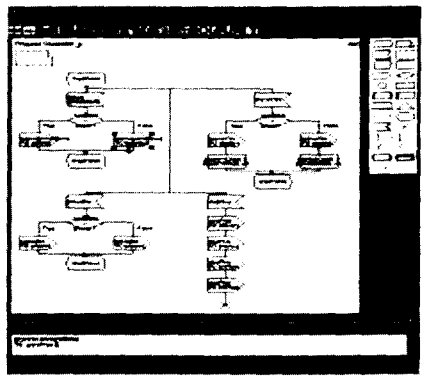

그림 4 SDL 편집기에서의 심벌 추적

4. 결론

현재의 소프트웨어 개발은 구현 중심에서 설계 중심의 개발 방법으로 변하고 있다. 설계 중심의 개발 방법은 설계 문서에서 자동 생성된 구현 프로그램을 통하여, 설계 수준에서 정형화된 명세를 검증함으로써 설계와 구현 사이의 일관성을 가진다. 그러므로, 개발 이후의 유지보수가 효율적이다. 설계 중심의 개발 방법을 구축하기 위한 방법은 설계환경과 구현환경사이를 연결하는 설계 수준에서 설계 문서를 검증할 수 있는 디버깅환경과 설계 문서에서 구현 언어로의 자동 번역기를 통해 이루어진다.

본 논문에서는 SDL과 CHILL의 통합된 개발환경에서 디버깅 환경을 제공하기 위한 도구인 SDL 심벌 디버거를 구현하였다. 구현된 SDL 심벌 디버거는 실행 중인 CHILL 프로그램의 수행을 정합, 추적, 제어 인터페이스를 이용하여 시스템 구현의 전단계 설계 문서인 SDL/GR의 심벌로 디버깅한다. 본 논문에서 구현한 SDL 심벌 디버거는 SDL 심벌 수준 실행 추적, SDL 심벌 수준 단일 스텝 추적, SDL 심벌 단위 중지점 설정 등의 기능과, TECH와의 연계 수행을 통한 SDL 심벌에서 CHILL 소스코드로의 참조 기능을 제공한다. SDL 심벌 디버거의 구현은 SDL에서 CHILL로의 번역기와 더불어 SDL과 CHILL의 개발환경을 통합시키고, 설계 수준에서 프로그램의 검증, 설계와 구현언어의 개념 차이에서 발생하는 오동작을 제거함으로써 전체 개발 시간과 비용을 줄일 수 있었다.

그러나, TECH와의 연계 수행시 TECH로부터 SDL 심벌 디버거로의 통신 채널 미비로 인해 CHILL 소스코드에서 SDL 심벌로의 참조 기능을 제공하지 못하는 문제점을 가지고 있기 때문에 향후 SDL 심벌 디버거와 TECH와의 통신 방법에 관한 연구가 필요하다.

5. 참고 문헌

[1] 최완, 박항구, 홍진표, 강석열, 송영기, 최고봉, 김환철, 신영승, 조준희, 이근구, 정찬진, *SDL 환경 : TDX-10 총서 제 6권*, pp.1-11, 한국전자통신연구소, 1994.

[2] S. Y. Lee, D.G. Lee, J. K. Lee, and S. H. Kim, "Conceptual Transformation from SDL-92 to CHILL-96 using Signal Subordination," *RTCSA'99*, pp. 484-487, 1999.

[3] 이시영, "개념 변환과 직접 사상을 이용한 SDL-92에서 CHILL-96으로의 자동 번역," 박사학위논문, Dec., 1999.

[4] 원준석, "SDL 심벌 추적기의 설계 및 구현," 석사학위논문, Dec., 1999.

[5] A. Olsen, O. færgemend, B. Møller-Pedersen, R. Reed and J. R. W. Smith, *Systems Engineering Using SDL-92*, pp.15-30, ELSEVIER SCIENCE B.V., 1995.

[6] D. G. Lee, J. K. Lee, W. Choi, B. S. Lee, and C. M. Han, "A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Large-scale Switching Systems," ETRI journal, Vol. 18, No. 4, 1997.

[7] 송영기, 박항구, 홍진표, 최고봉, 박경숙, 백의현, 이동길, 이준경, 정영식, 정찬진, 조철희, *CHILL 프로그래밍 언어 : TDX-10 총서 제 7 권*, pp.i-ii, 한국전자통신연구소, 1994.

[8] Telelogic AB., *Telelogic Tau 3.5 Manual*, 1998.