SDL에서 C로의 번역기의 설계 및 구현

김병건* 최원혁** 김성재* 김승호*
*경 북 대 학 교 컴 퓨 터 공 학 과 **한 국 전 자 통 신 연 구 원
bgkim@borami.knu.ac.kr whchoi@etri.re.kr
sjkim@borami.knu.ac.kr shkim@bh.knu.ac.kr

Design and Implementation of SDL to C Translator

Byung-Gun Kim* Won-Hyuk Choi** Seong-Jae Kim* Sung-Ho Kim*
*Department of Computer Engineering, Kyungpook National University
**Electronic and Telecommunication Research Institute

요 약

현재의 소프트웨어 개발은 구현 중심의 개발에서 설계 중심의 개발로 바뀌고 있다. 설계 중심의 개발은 구현 프로그램의 자동 생성을 바탕으로 시스템 개발 시간을 크게 단축할 수 있고, 정형화된 명세를 검증함으로써 설계와 구현의 일관성을 유지할 수 있으며 유지보수가 효율적이다. 본 논문에서는 설계 중심의 개발 환경을 구축하기 위해, ITU-T에서 권고한 시스템의 명세와 기술을 위한 언어인 SDL에서 범용 프로그래밍 언어인 C로의 자동 번역기를 설계하고 구현하였다.

1. 서론

통신상에서 다양한 서비스가 요구됨에 따라 통신용 소프트웨어의 개발에 대한 중요성이 점차 증대되고 있다. 그러나 이러한 소프트웨어들은 대규모의 개발 인력과 긴 개발 기간을 필요로 하고, 생명주기 또한 다른 소프트웨어에 비해 긴 특성을 가지고 있기 때문에 다양한 서비스 요구에 대한 신속한 대응이나 새로운 기능의 개발이 힘들다. 따라서 현대의 이러한 요구 사항에 대처하기 위해 SDL과 같은 상위 수준 시스템 명세 언어들을 사용한 설계 중심의 개발 방법론이 대두하고 있다. 또한 설계 언어를 통해 직접적인 프로그램 코드의 작성 없이 소프트웨어를 개발하는 방법이 다양하게 연구되어 왔다[1, 2].

ITU-T(International Telecommunication Union - Telecommunication Standardization Sector)에서는 보다 효율적인 시스템의 유지보수를 위해 시스템의 명세 및 기술 언어로 SDL(Specification and Description Language)을 권고하여 시스템을 위한 목적 코드의 개발 전에 목적 시스템에 대한 명세 및 기술을 수행하도록 하고 있다[3-5].

SDL을 이용한 시스템의 명세 및 기술은 시스템의 개발과 검증을 상위 설계 수준에서 정형화된 방법으로 자동적으로 수행할 수 있게 해 준다. 또한 요구자와 개발자 상호간의 정보 교환을 용이하게 하여 전체 시스템에 대한 분석과 시뮬레이션, 시스템의 유지 관리 등에 유용하게 사용될 수 있으며 객체지향 기술 방법을 적용시킴으로써 시스템 구현의 상위 단계에서 재사용성을 향상시킬 수 있다.

대형 시스템들은 10년 이상의 오랜 생명주기와 대규모 개발 및 유지보수 인원의 참여로 인하여 설계 문서와 구현 사이의 불일치가 나타나는 경우가 많다. 따라서 소프트웨어의 개발 시 설계에서 구현까지의 일관성을 유지하고 기능의 추가나 변경, 삭제 등 설계 수준에서의 부분적인 변화가 의미적 변화 없이 구현에 바로 영향을 미치도록 하기 위해서는 소프트웨어의 명세로부터 실제적인 구현 코드를 생성하는 자동 번역기의 개발이 매우 중요하다. 또한, 최근의 교환기 시스템 개발에 기존의 CHILL과 같은 언어보다 범용 언어인 C의 사용이 점점 더 확발해지고 있고, 많은 알고리즘에 관련된 모듈들이 C로 작성되어 있으므로 SDL에서 C로의 번역기의 개발은 필수적이라고 할 수 있다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 SDL과 C에 대해서 설명하고, 3장에서는 SDL에서 C로의 번역기의 설계와 구현에 대하여 살펴보고, 4장에서는 실제로 번역된 예를 들어 본다. 마지막으로 5장에서는 결론을 내린다.

2. SDL과 C

SDL은 시스템의 동작을 확장된 유한 상태 기계 개념을 기초로 정의하고 있는데, 흐름도와 유사한 형태의 그래픽 다이어그램으로 전체 시스템을 표현하는 SDL/GR(Graphical Represen-

tation)과 프로그래밍 언어와 비슷한 형태로 SDL/GR에 상응하는 SDL/PR의 두 가지 형태로 이루어진다. SDL/GR은 시스템의 구조와 행위를 명확하게 보여주고 제어와 자료의 흐름을 시각화하므로 유한 상태 기계에 의해 효과적으로 표현되는 통신 시스템의 명세와 설계에 적합하고, SDL/PR은 시스템의 체계적인 문서화나 다른 이기종 CASE 도구간의 정보 교환에 적합하다. 따라서 SDL은 전자 교환기와 같은 대형 시스템의 명세 및 설계 문서를 작성하는데 많이 이용되며 이외에도 데이터 통신 분야의 프로토콜을 표현하는데 주로 사용되고 있다.

범용 프로그래밍 언어인 C는 기존의 프로그래밍 언어가 가지는 대부분의 기능을 포함하는 크고 복잡한 언어로 다양한 응용 분야에 적용할 수 있는 일반적인 기능에 중점을 두고 설계되었다. C 언어는 구조화된 프로그래밍 및 모듈화 된 설계를 쉽게 할 수 있어서 프로그램의 신뢰도가 높고 이해하기가 쉽다. 또한, 효율적이고 강력하며 융통성, 이식성도 매우 좋아 특정 시스템을 위해 만들어진 C 프로그램을 다른 시스템에서 실행시키기 위해서 그 프로그램의 일부만 수정하거나 전혀 수정하지 않아도 되는 등의 여러 가지 장점을 가진다.

3. 설계 및 구현

SDL에서 C로의 번역기는 SDL로 작성된 원시 프로그램을 분석, 변형하여 C로 작성된 원시 프로그램을 생성할 수 있는 도구를 말한다. SDL에서 C로의 번역기를 통한 개발 환경의 개요는 그림 1과 같다.

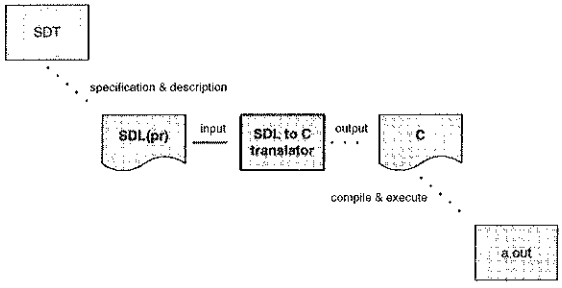

그림 1. SDL에서 C로의 번역기를 통한 개발 환경의 개요

이러한 번역기는 크게 SDL로 작성된 원시 프로그램을 분석하는 분석부, 분석된 정보를 바탕으로 이를 C 정보 형태로 대응시키는 대응부, 대응된 C 정보를 바탕으로 C로된 원시 프로그램을 생성하는 생성부 등의 모듈로 나뉜다.

전체적인 SDL에서 C로의 번역기는 그림 2와 같이 구성된다.

3.1 SDL 분석기(SDL analyzer)

SDL 분석기는 입력 SDL/PR 데이터의 구문 분석을 위해 명세 정보의 확장과 같은 전 단계 작업을 수행한다. 입력 정보들 중에서 코드 변환에 영향을 주지 않는 부분들을 변환에 앞서 미리 제거한다. 또한 SDL에서 사용되는 매크로는 실제 변환시에 해당 코드들로 확장되어야 하므로 구문 분석 단계의 효율성을 고려하여 미리 확장을 수행한다. 이러한 작업 후에 실제적으로 번역에 필요한 정보를 분석, 수집하고 이 정보들을 대응기에 전달하게 된다.

3.2 SDL to C 대응기(SDL to C mapper)

대응기는 분석기에서 넘어온 정보를 각각에 해당하는 C 정보로 대응시킨다. 이 과정에서 언어상의 차이점으로 인해 고려해야 할 사항들이 있다.

SDL에서의 시스템과 블록에는 수행 코드는 존재하지 않고, 프로세스에서 사용할 변수 및 시그널과 관련된 정보만 존재한다. 따라서 시스템과 블록은 변수 및 시그널 정의만 전역 변수로 대응시키고 번역 과정에서 사라진다.

SDL의 프로세스에 대응되는 개념으로 C의 함수를 쓸 수 있으나 프로세스를 함수로 대응시키는데 있어서 고려해야 할 점이 있다. SDL에서 내부 변수로 선언 없이 사용되는 SENDER, PARENT, OFFSPRING 등과 같은 키워드들에 대한 처리가 필요하며, SDL에서의 프로세스 자체가 시그널 수신을 대기중인 상태이므로 이를 나타낼 수 있는 방법 또한 필요하다. 따라서 SDL의 내부 변수는 C코드에서 명시적인 변수로 선언해 주어야 하고, 시그널 수신을 대기중인 프로세스는 while(1)문을 이용하여 표현해 준다. 또한 SDL에서는 분산, 병렬환경을 지원하는데 비해 C에서는 이러한 것들을 지원하지 않으므로 이를 분산, 병렬환경을 지원해주는 SROS(Scalable Realtime Operating Sys- tem)의 프리미티브를 이용하여 해결한다.

분석기에서 넘어오는 정보를 바탕으로 대응시킬 C 정보를 생성하고 C 코드 생성에 필요한 추가 정보들을 생성한다. 생성된 정보들은 C 코드 생성기로 전달된다.

3.3 C 코드 생성기(C code generator)

대응기에서 넘어온 정보들을 이용해서 각각의 C 정보에 해당하는 코드를 생성한다. 이렇게 번역되어 나온 C 언어로 된 원시 프로그램을 컴파일하여 수행하기 위해서는 병렬 수행되는 프로세스를 지원하기 위한 라이브러리를 연결시켜 실행화일을 생성해야 한다.

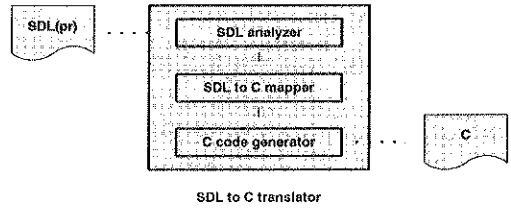

그림 2. SDL에서 C로의 번역기의 구성

4. SDL에서 C로의 번역 예

다음의 표 1과 2는 3장에서 제시된 방법으로 SDL에서 C로의 번역기를 설계하고 구현하여 SDL/PR 파일을 입력받아 C코드를 생성한 예이다.

표 1. SDL/PR의 예

```
 1 : PROCESS prover ;
 2 : FPAR paraNum Integer ;
 3 : DCL myNum Integer ;
 4 : DCL proveIndex Integer ;
 5 : DCL controller_prover Pid ;
 6 : START ;
 7 : task myNum := paraNum ;
 8 : task controller_prover := PARENT ;
 9 : nextstate WaitProve ;
10 : state WaitProve ;
11 : input prove ( proveIndex ) ;
12 : decision proveIndex = myNum ;
13 : ( TRUE ) : output proveYes TO controller_prover ;
14 : ( FALSE ) : output proveNo TO controller_prover ;
15 : enddecision ;
16 : nextstate - ;
17 : input stopPlay ;
18 : stop ;
19 : endstate ;
20 : endprocess prover ;
```

표 2. 번역된 C 코드의 예

```
 1 : void prover ( unsigned int PARENT , int paraNum )
 2 : {
 3 : enum { WaitProve } ;
 4 : unsigned int p_state ;
 5 : unsigned int SENDER ;
 6 : unsigned int OFFSPRING ;
 7 : int prover_ret ;
 8 : int myNum ;
 9 : int proveIndex ;;
10 : unsigned int controller_prover ;
11 : myNum = paraNum ;
12 : controller_prover = PARENT ;
13 : p_state = WaitProve ;
14 : while ( TRUE )
15 : {
16 : switch ( p_state )
17 : {
18 : case WaitProve :
19 : prover_ret = _rcvsigcase ( 2 , &SENDER , 2 , 0 , _prove ,
&proveIndex , _stopPlay , NULL ) ;
20 : if ( prover_ret == _prove )  &proveIndex , _stopPlay , NULL ) ;
21 : {
22 : if ( ( proveIndex == myNum ) == TRUE )
23 : {
24 : ipc_send ( _proveYes , 0 , controller_prover , 0 , 0 , NULL ) ;
25 : }
26 : else if ( ( proveIndex == myNum ) == FALSE )
27 : {
28 : ipc_send ( _proveNo , 0 , controller_prover , 0 , 0 , NULL ) ;
29 : }
30 : p_state = WaitProve ;
31 : }
32 : if ( prover_ret == _stopPlay )
33 : {
34 : process_exit ( ) ;
35 : }
36 : break ;
37 : }
38 : }
39 : return ;
40 : }
```

표 1은 SDL/PR의 한 프로세스이며 표 2는 번역기를 통해 생성한 대응되는 C 함수이다. 표 1의 1~2행은 프로세스의 선언부로 표 2의 1행과 같이 생성된다. 표 2의 3~7행은 SDL의 상태와 내부 변수 등을 나타내기 위해 C 코드에 추가된 부분이다. 표 1의 3~5행은 변수의 선언부분으로 표 2의 8~10행에 대응된다. 표 2의 14행은 시그널 수신을 대기하고 있는 SDL 프로세스의 특성을 나타내기 위한 while문이며 16행은 SDL 프로세스의 상태에 따른 수행을 결정하기 위한 switch문이다. 표 1의 11, 13, 14, 17행은 시그널의 송, 수신에 해당하는 것으로 표 2의 19, 24, 28행과 같이 SROS 프리미티브 형태의 코드로 번역된다.

5. 결론

현재의 소프트웨어 개발은 구현 중심에서 설계 중심의 개발 방법으로 바뀌어 가고 있다. 설계 중심의 개발 방법은 설계 문서에서 자동 생성된 구현 프로그램을 통하여 시스템 개발 시간을 단축할 수 있고, 설계 수준에서 정형화된 명세를 검증함으로써 설계와 구현 사이의 일관성을 가지게 되어 개발 이후의 유지 보수가 효율적이다. 그러므로 설계 중심의 개발 방법을 구축하기 위해서는 설계 문서에서 구현 언어로의 자동 번역기가 필요하다.

본 논문에서는 SDL로 작성된 원시 프로그램에서 C 언어로 된 원시 프로그램을 자동으로 생성하는 번역기를 설계하고 구현하였다. SDL에서 C로의 번역기는 입력으로 들어오는 SDL로 작성된 원시 프로그램을 분석하는 분석기와, 분석된 정보를 해당하는 C 언어 정보에 대응시키는 대응기, 대응된 C 코드를 생성하기 위한 생성기로 구성되어 있다. 이렇게 설계하고 구현한 번역기를 이용하여 실제의 응용에 적용함으로써 앞서 기술한 여러 가지 이점들을 얻을 수 있었다.

향후에는 SDL의 패키지 등과 관련된 아직 지원되지 않는 번역에 대한 연구가 필요하다. 또한 C 코드를 디버깅 할 수 있는 환경과 같은 다른 도구들과의 연동에 관한 연구도 필요하다.

참고 문헌

[1] K. Cheng and L. Jackson, *"MELBA+: AN SDL SOFTWARE ENGINEERING ENVIRONMENT,", The fourth SDL Forum, SDL '89: The Language at Work,* pp.95-103, 1989.

[2] D. Lee, J. Lee, W. Choi, B. Lee, and C. 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.

[3] A. Olsen, O.færgemend, B. Pedersen, R. Reed, and J. Smith, *Systems Engineering Using SDL-92,* ELSEVIER SCIENCE B.V., 1995.

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

[5] *CCITT Specification and Description Language,* ITU Recommendation Z. 100, p.220, 1992.