제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

SDL 패턴을 이용한 통신 프로토콜 개발방법

정기숙*, 이병선*, 변영준**
*한국전자통신연구원 네트워크연구소
**University of Florida
e-mail : kschung@etri.re.kr

A Methodology for Development of Communication Protocols using SDL Patterns

Ki-Sook Chung*, Byung-Sun Lee*, YoungJoon Byun**
*Softswitch Team, Network Research Lab.,
Electronics and Telecommunications Research Institute
**Dept. of Computer and Information Science and Engineering
University of Florida

요 약

소프트웨어 재사용은 개발 노력 및 비용의 절감 등의 효과로 인해 오랫동안 연구 대상이 되어 왔다. 최근에는 코드 재사용의 단계를 넘어 디자인 패턴이나 프레임웍과 같은 보다 상위 레벨에서의 구조 및 설계 재사용에 대한 관심이 늘고 있다. 본 논문에서는 통신 프로토콜 개발 경험을 토대로 추출한 SDL 패턴언어를 소개하고 추출된 SDL 패턴을 통신 프로토콜 개발 단계에서 효율적으로 재사용함으로써 초기 단계에서의 오류를 줄이고 보다 더 안정된 시스템을 구현할 수 있는 방법에 대해 기술한다.

1. 서론

디자인패턴은 특히 객체지향 설계 분야에서 개발 경험을 재사용하기 위해 도입된 개념이다[2,3]. 빈번하게 발생하는 설계 단계의 문제에 대한 해결방법을 일정한 형식에 근거해 기술함으로써 설계 단계에서의 오류를 최소화하고 비용을 단축하여 설계의 재사용성을 높이고 있다.

본 논문에서는 통신 프로토콜이라는 특정 응용 도메인을 대상으로 하는 SDL(Specification and Description Language) 패턴을 소개하고 개발 단계에서 적용하는 방법에 대해 기술한다.통신 프로토콜 개발 시 정형적 언어인 SDL 을 사용하면 효율적인 시스템 설계 및 높은 유지보수성을 얻을 수 있다. 이러한 SDL 기반 프로토콜 개발 시 자주 발생하는 설계 문제 및 해결방법을 패턴으로 추출한 것이 SDL 패턴이다[1]. 제 2 장에서는 추출된 SDL 패턴에 대해 간략히 기술하고 3 장에서 통신 프로토콜 개발 단계에서 SDL 패턴을 적용하는 방법을 살펴본다. 마지막 제 4 장에서는 향후 연구방향을 모색하고 결론을 맺는다.

2. 통신프로토콜을 위한 SDL 패턴

2.1 SDL 패턴 개요

SDL (Specification and Description Language)은 ITU-T 권고안 Z.100 에서 정의된 객체지향 정형언어로써, 복잡한 실시간 시스템을 기술하기 위해 고안되었다. 특히, 통신 프로토콜과 같이 시그널로 서로 상호작용 하는 병행시스템(concurrent system)을 기술하는데 많이 사용되어 지고 있다.

SDL 패턴이란 통신 프로토콜이라는 특정 도메인 내에서 빈번히 발생하는 디자인 문제에 대한 일반적인 해결방법을 기술하는 디자인 패턴이다. 즉, 패턴을 적용하여 구체화한 결과가 SDL 의 형식으로 표현된다. 그러므로 패턴 기술 시에 패턴의 응용을 위해 문법적인 규칙 뿐만 아니라 의미적 특성을 보다 더 세밀하게 기술해야 한다.

SDL 패턴은 기존의 객체지향 디자인 패턴[2]와 비교해 볼 때, 소프트웨어 개발 방법에 있어서 객체지향이라는 점에서는 동일하다. 특정 환경에서 빈번히 발생하는 일반적인 설계 문제를 해결하기 위해 객체와

클래스들을 찾아내고 그들 간의 관계와 역할 및 상호 작용을 명시한다. SDL 패턴이 일반적인 디자인 패턴과 다른 점은 통신 프로토콜이라는 특정 응용 도메인의 테두리에서 SDL 의 정형성이라는 장점을 이용하는 것이다.

SDL 패턴의 기술 형식은 기존의 패턴 템플릿[2,3]을 따르면서 SDL 구현 항목에 중점을 두었으며 (표 1)과 같은 형식을 따른다.

(표 1) SDL 패턴 형식

항목	설명
Name	패턴의 이름, 패턴의 주목적을 기술할 수 있는 의미있는 단어 또는 문구여야 한다.
Context	문제가 발생하는 상황
Problem	해결해야 하는 문제를 명시한다.
Forces	문제를 다양한 측면에서 살펴보고 고려해야 할 제약 사항이나 해결책이 가져야 하는 특성에 대한 요구사항을 제시한다.
Solution	문제의 해결책
Implementation	해결책에 대한 SDL 구현을 보여주며 solution 과 대체적으로 1:1 로 매핑된다.
Example	패턴을 사용한 예제를 보인다.
Variants	특수화시키거나 변형된 패턴을 보여준다.
See also	다른 시스템에서의 사용법이나 유사한 패턴을 기술한다.

2.2 통신 프로토콜을 위한 SDL 패턴언어

패턴언어(Pattern language)는 주어진 도메인에서 문제를 해결하기 위해 사용되어지는 패턴들의 집합을 말한다. (표 2)에 나열된 SDL 패턴언어는 통신 프로토콜 개발 경험을 토대로 추출[1]한 것으로 통신 프로토콜 개발 시 재사용되고 있다[5].

(표 2) SDL 패턴언어

Category	Patterns	Variants
Structural patterns	Protocol layer	Split protocol layer
	Mux	
	Dynamic handler	Split dynamic handler
Behavioral patterns	Basic CEFSM	Predicate CEFSM
		Predicate after action
		Source merge
		Target merge
		Sequential merge
	Timer	
	Repeated events	Timed repeated events
		Timed repeated trials
	Message transfer	Simple sender
		Simple receiver
		Confirmed sender
		Confirmed receiver
		Timed simple receiver
		Timed confirmed sender
		Repeated receiver
		Repeated sender
		Repeated comfirmed sender
		Repeated confirmed receiver
		Timed repeated trial receiver
		Timed repeated trial confirmed sender
		Timed repeated trial confirmed receiver
		Message transfer in middle layer

(표 2)에서 보는 바와 같이 SDL 패턴들은 용도나 특성에 따라 structural pattern 과 behavioral pattern 이라는 두 가지 그룹으로 분류된다.

- ***Structural Patterns***: 통신 프로토콜의 구조(architecture)에 대한 패턴이다. 통신 프로토콜 소프트웨어는 그 특성 상 여러 블록(block)들과 그들 간의 통신 통로(communication path)로 구성되어 질 수 있다. 블록은 시스템을 개발하는 구성요소로서 또다시 서브 블록들로 이루어질 수 있으므로 트리 구조를 형성하게 된다. 상위 레벨 시스템 구조 설계 단계에서는 블록은 블랙박스로 표현되어 통신 통로나 메시지와 같은 외부 인터페이스는 정의하지만 블록 내부의 세부적인 사항은 고려하지 않는다.

- ***Behavioral Patterns***: Structural pattern 을 이용하여 식별해낸 블록들의 내부 행위를 설계하는데 사용된다. 각 블록의 인스턴스는 입력 메시지등과 같은 이벤트에 의해 변경되는 상태(state)를 가지며 이 입력 메시지로 인해 또 다른 출력 메시지를 발생시키는데 이러한 동작들을 정형적으로 기술하기 위해 CEFSM(communicating extended finite state machine)을 사용한다. 또한 조건이나 시간과 같은 제한 요소를 표현하기 위해 predicate 와 타이머 개념을 도입하여 다양한 시스템 동작을 기술할 수 있도록 확장할 수 있다.

3. SDL 패턴을 이용한 통신 프로토콜 개발 방법

3.1 SDL 패턴언어의 구조

설계 단계에 패턴을 적용하기 전에 (표 2)에 나열된 SDL 패턴언어의 특성을 살펴보면 (그림 1)과 같이 나타낼 수 있다. 추출된 두가지 종류의 패턴은 통신 프

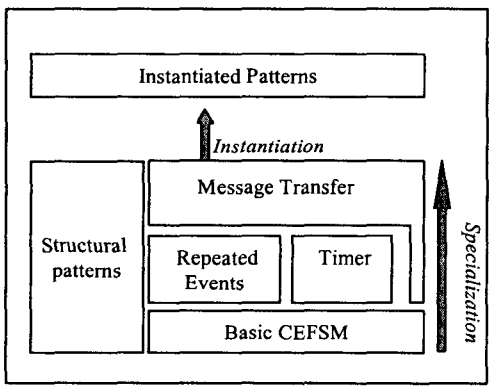
로토콜 개발 초기 단계에서 시스템 구조 측면과 동작 측면을 설계할 때 발생하는 설계 문제를 다루고 있으며, 이들 structural pattern 과 behavioral pattern 이 실제 시스템 설계 단계에서 적용되기 위해서는 구체화(instantiation) 과정과 특수화(Specialization)과정이 필요하다.

- *Instantiation*(구체화): SDL 패턴을 적용하면서 개발 대상 시스템 내에서의 역할과 위치에 맞게 메시지, 상태, 동작, 타이머 등 SDL 패턴 파라메터에 실제 이름과 값을 부여하는 것이다.

- *Specialization* (특수화): behavioral 패턴에 적용되는 방법으로 기본적인 패턴에 다른 제약 조건, 즉, predicate, 타이밍 제약, 이벤트의 반복 등을 반영함으로써 더 복잡하고 특수화된 동작을 나타낼 수 있도록 패턴 자체를 진화시키는 과정이다. 이 과정을 통하여 보다 더 복잡한 동작을 표현할 수 있는 새로운 패턴을 만들어 낼 수 있다. 특수화의 가장 일반적인 방법으로는 basic CEFSM 과 같은 기본 패턴들의 variant 를 만들어 이들을 합병하는 방법이 있다. 합병하는 방법에는 순차적 합병(sequential merge), 소스 합병(source merge), 타겟 합병(target merge) 등이 있다[7].

구체화 과정 및 특수화 과정을 거치면서 패턴은 보다 실제 시스템에 적용하기 쉬운 형태를 갖추게 되며, 패턴언어 자체가 확장될 수 있다.

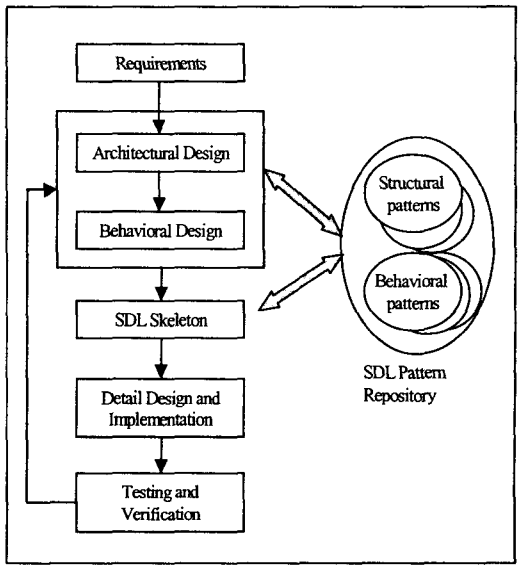
(그림 1) SDL 패턴언어의 구조



3.2 SDL pattern 을 이용한 개발 절차

지금까지 소개한 SDL 패턴언어를 기반으로 하는 개발 절차는 (그림 2)에서와 같이 간단하게 도식화 할 수 있다.. 일반적인 시스템 개발 방법과 다른 점은 시스템 요구사항 분석 후 초기 설계에서 적당한 SDL 패턴을 검색하여 적용하고 그 결과 SDL skeleton 을 결과물로 얻는다는 점이다. 그리고 시스템 설계 단계를 시스템 구조 설계와 동작 설계의 두가지 측면으로 나누어 순차적으로 진행하여 각 과정에서 해당하는 SDL 패턴을 적용할 수 있도록 한다. 즉, 구조와 동작에 관한 SDL 패턴을 재사용하기 위해서는 설계 단계에서 시스템 구조 설계 시 먼저 structural 패턴을 적용하여 시스템의 구성 요소인 블록들과 통신 경로를 찾아내어 설계하고, 각 블록의 내부 동작을 설계할 때 behavioral 패턴을 적용한다

(그림 2) SDL 패턴기반 개발방법



패턴의 재사용을 위해서는 다음과 같은 과정이 필요하다.

1) SDL 패턴 검색: SDL 패턴언어 즉, 패턴 repository 에서 각 패턴의 내용 중 context, problem 부분을 읽어보고 비슷한 문제나 환경에 적합한 패턴을 검색한다.
2) SDL 패턴의 구체화 및 특수화: 적당한 패턴을 찾은 다음에는 force 부분과 example 을 검토하고 실제 구현 대상 시스템에 적당한 형태로 적용한다. 패턴에서 제시한 solution 에 구체적인 이름과 값을 부여하여 구체화하고 특수화 정도를 살펴 적당한 레벨의 패턴을 선별한다.
3) SDL skeleton 생성: 패턴의 SDL implementation 부분을 참고하여 SDL skeleton 을 생성한다. 이 때 상용화 도구를 이용하여 편집 및 문서화를 하면 유지보수성을 높일 수 있다. 패턴에서 식별한 블록은 실제 SDL 상에서는 블록 또는 프로세스에 매핑이 되며, 신호 경로(signaling path)는 채널 또는 시그널 루트에 매핑이 된다.

위와 같은 패턴 적용 단계에서 기존 패턴에 대한 수정이나 새로운 패턴을 발견하여 추가할 수도 있다. 전체 절차가 반복되어 이루어 지면서 SDL 패턴언어에 대한 비정형적이지만 평가를 내릴 수도 있고 개발 과정에서 보다 높은 품질의 설계를 얻을

수 있다.

전체적인 상위 설계가 끝나면 기존 방식과 같이 세부 설계 및 구현을 거쳐 시험 및 검증 단계를 거친다. 이 과정도 물론 SDL 이 기반이 되므로 상용화 도구를 사용하여 사전에 시뮬레이션을 통하여 시스템의 동작 오류 등을 미리 발견할 수 있으며 코드 생성기를 이용하여 자동으로 코드를 생성할 수 있다.

4. 결론

급속하게 발전하고 있는 통신 환경에서 통신 프로토콜 개발에 대한 비용이나 시간 단축은 필수적인 요구사항이 되고 있다. 본 논문에서는 통신 프로토콜 개발 시 SDL 패턴을 적용하여 통신 소프트웨어의 재사용성을 높일 수 있는 개발 방법에 대해 소개하였다. 통신 프로토콜을 기술하기에 적합한 정형 언어인 SDL 을 기반으로 하는 SDL 패턴언어를 소개하고 실제 전체 개발 단계에서 어떻게 적용할 수 있는지 살펴 보았다.

패턴 추출 방법이나 검색 및 적용 방법에 있어서는 아직 체계적이거나 자동화된 방법이 나오지 않았기 때문에 개발자의 경험과 능력에 의존할 수 밖에 없다. 향후 개발 초기 단계에서 비용과 노력을 줄이기 위한 패턴 재사용에 관한 연구는 이런 문제에 중점을 두어야 할 것이며 추출된 패턴에 대한 검증 및 평가 방법을 찾아내는 것도 중요한 연구 방향이 될 것이다.

참고문헌

[1] Y. J. Byun, B.A. Sanders, and K.S. Chung, "A pattern language for communication protocols," in *Proceedings of the 9th Conference on Pattern Languages of Programs*, September 2002.

[2] E. Gamma, R. Helm, R.Johnson, and J.Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

[3] F. Buschmann, R.Meunier, H.Rohnert, P.Sommerland, and M.Stal. *Pattern-oriented Software Architecture: A System of Patterns*, John Wiley & Sons, 1996.

[4] B. Geppert and F. Roβler. The SDL pattern approach-reuse-driven SDL design methodology. *Computer Networks*, 35(6), 1996

[5] C.S.Keum and et al. "SDL pattern-based SIP-T Protocol in a softswitch system", *in the proceeding of CSN-2002* , September 2002.

[6] J. Ellsberger, D.Hogrefe, and A.Sarma. *SDL:Formal Object-oriented Language for Communicating Systems*. Prentice Hall, 1997.

[7] Y. J. Byun, C.S. Keum, et al. "Design patterns of Communicating Extended Finite State Machines in SDL", in *Proceedings of the 8th Conference on Pattern Languages of Programs*, September 2001.