메시지 기반 인터페이스 공동 개발을 위한 메시지 정의 및 관리 시스템

유제영° 박진희
삼성탈레스(주)
jeyoung.yu@samsung.net, jh91.park@samsung.net

Message Definition and Management System for Cooperative Message-based Interface Development

Je-young Yu°, Jinhee Park
Samsung Thales Co., Inc.

대형 시스템은 일반적으로 수십 명 이상의 개발자가 협력하여 개발한다. 하나의 대형 시스템은 수많은 컴포넌트로 구성되며, 그 컴포넌트의 수는 시스템의 구성에 따라 수십 개에서 수천 개에 이르기도 한다. 각 컴포넌트는 전체 시스템의 기능을 수행하기 위하여 서로 명령이나 데이터를 주고 받는다. 이렇게 각 컴포넌트가 제공하는 서비스나 데이터의 형식과 내용을 정의한 것이 컴포넌트의 인터페이스이다. 컴포넌트의 인터페이스가 정의되는 형식에는 메시지 기반의 인터페이스 정의와 RPC (Remote Procedure Call)나 분산 객체 기반의 인터페이스 정의가 있다. 메시지 기반의 인터페이스 정의는 컴포넌트간 통신이 UDP/IP와 같은 메시지 기반의 통신 서비스나 DDS[1] 같은 미들웨어 상에서 이루어진다. 여기에서 인터페이스의 정의는 컴포넌트간의 메시지 형식과 내용의 정의를 의미한다. RPC나 분산 객체 기반의 인터페이스 정의는 컴포넌트간 통신이 RPC 서비스나 CORBA[2], DCOM[3]과 같은 분산 객체 서비스 미들웨어 상에서 이루어진다. RPC나 분산 객체 기반의 인터페이스 정의는 원격에서 호출할 수 있는 함수 정의나 클래스 정의를 의미한다.

대형 시스템은 각 컴포넌트가 여러 다른 개발자에 의하여 개발된다. 따라서, 개발 과정에서 이러한 컴포넌트 간의 인터페이스 정의 및 관리의 효율성은 전체 개발 효율에 큰 영향을 미친다. 특히, 메시지 기반의 인터페이스 정의의 경우, 개발 과정에서 컴포넌트간의 인터페이스 역할을 하는 메시지가 필요에 의하여 새로 생성되거나, 제거되거나, 변경되는 일이 빈번하게 일어난다. 이러한 변경에 대한 관리가 효율적으로 이루어지지 않는 경우, 관련된 컴포넌트를 개발하는 서로 다른 개발자들이 서로 다른 인터페이스를 기반으로 개발을 진행하는 경우가 발생할 수 있다. 이는 바로 개발 및 시험에 직접적인 영향을 미치며 공동작업의 효율을 떨어뜨리는 직접적인 원인이 된다. 특히 이러한 문제는 컴포넌트의 통합 과정에서 흔히 볼 수 있다.

기존에 소스 코드를 관리하고 형상 변경을 추적/관리하는 시스템은 다수 존재한다. 대표적인 형상관리 시스템으로는 상용 시스템으로 ClearCase[4]가 있으며, 공개 소프트웨어 시스템으로 CVS[5]와 Subversion[6]이 있다. 하지만, 인터페이스 정의를 관리하고 형상 변경을 추적/관리하는 시스템을 찾아보기는 힘들다. 따라서 많은 개발 프로젝트에서 인터페이스 정의에 대한 관리를 문서(Interface Design Document-IDD)를 통하여 수행하거나, 인터페이스 정의를 위한 코드를 생성한 후, 그것을 소스 코드 관리 시스템을 이용하여 간접적으로 관리하는 방법을 이용하곤 한다.

간접적인 인터페이스 변경 관리는 효율성이 떨어지며, 인터페이스 변경 절차의 부재로 관리에 많은 노력이 요구된다. 또한, 간접적인 관리의 경우 정확하게 이루어지지 않으면, 관리가 전혀 이루어지지 않는 것과 동일한 결과를 초래한다. 이러한 문제를 직접적으로 해결할 수 있는 방법은 컴포넌트 간의 인터페이스를 정의하고 관리할 수 있는 별도의 시스템을 이용하는 것이다.

우리는 이러한 문제를 해결하기 위하여 메시지 정의 및 관리 시스템을 개발하였다. 메시지 정의 및 관

리 시스템은 메시지 기반의 인터페이스 관리를 위한 수단을 제공한다. 또한, 메시지 구조와 해당 메시지를 처리하기 위한 기본적인 소스 코드를 자동으로 생성해 주며, 정의된 메시지들을 이용하여 IDD 문서를 생성해 줌으로써 문서작성 작업을 간단하게 만들어 주며, 메시지 변경으로 인한 실제 인터페이스와 문서와의 불일치를 방지한다.

메시지 정의 및 관리 시스템은 전체 시스템의 각 컴포넌트 간 메시지를 관리함으로써 각 컴포넌트 간 메시지 흐름에 대한 종합적인 정보를 수집하는 것이 가능하다. 이러한 정보를 바탕으로 컴포넌트 간 메시지 흐름의 병목 현상의 발생을 미리 방지하거나, 전체 시스템 성능의 조율을 지원할 수 있다.
메시지 정의 및 관리 시스템은 현재 일차 버전의 개발이 완료되어 실제 개발에 적용을 위한 시험을 진행 중이며, 개발에 적용 시 다음과 같은 사항들을 기대할 수 있다.

- 컴포넌트 간의 인터페이스 정의 절차 단순화
- 컴포넌트 간의 인터페이스 변경 절차 단순화
- 인터페이스 변경 내역 추적
- 서로 다른 인터페이스 버전의 사용으로 인한 시간과 노력의 낭비 방지
- 인터페이스 기술서 자동 생성으로 문서화 단순화
- 컴포넌트간 인터페이스 관련 코드 자동 생성으로 표준화된 인터페이스 코드 사용 지원
- 전체 시스템에서 각 컴포넌트간의 메시지 및 컨트롤 흐름 분석

현재까지 개발된 버전의 경우, 표준화된 인터페이스 처리 코드 지원과, 컴포넌트간 메시지 및 컨트롤 흐름 분석 부분은 지원하지 않는다. 따라서 향후 연구 과제로 첫째, 표준화된 인터페이스 처리 코드를 자동으로 생성하여 인터페이스 처리 부분을 표준화 및 단순화함으로써 개발 효율을 증가시키는 방법 연구와, 둘째, 전체 시스템의 메시지 및 컨트롤 흐름을 분석하여, 컴포넌트 간의 네트워크 트래픽 분석 및 컨트롤 흐름에 대한 도식화를 제공함으로써 시스템 분석 도구로 활용하기 위한 방법에 대한 연구를 진행할 예정이다. 또한, 메시지 기반의 인터페이스 정의뿐만 아니라, 분산 객체 기반의 인터페이스 정의에도 적용할 수 있는 인터페이스 정의 및 관리 시스템으로의 발전 역시 향후 연구 과제이다.

참고문헌

[1] "Data Distribution Service (DDS) for Real-Time Systems Version 1.2", OMG, January 2007
[2] "Common Object Request Broker Architecture (CORBA/IIOP) Version 3.0.3", OMG, March 2004
[3] Charlie Kindel and Nat Brown, "Distributed Component Object Model Protocol(DCOM/1.0)," Microsoft Corporation, January 1998
[4] IBM Rational ClearCase, http://www-06.ibm.com/software/awdtools/clearcase/
[5] CVS - Concurrent Versions System, http://www.nongnu.org/cvs/
[6] Subversion, http://subversion.tigris.org/