W/S 기반의 원격 공동연구 플랫폼 설계 및 구현

신영미, 김명섭, 홍원기
포항공과대학교 전자계산학과
경북 포항시 효자동 산 31번지 ㉾ 790-784
{dry, mount, jwkhong}@postech.ac.kr

서영호, 김 용
시스템 공학 연구소 감성공학연구부
대전시 유성구 어은동 1번지 ㉾ 305-333
{syh, ky}@seri.re.kr

Design and Implementation of a W/S-based Collaborative Research Platform

Young-Mi Shin, Myoung-Sup Kim,
Won-Ki Hong
Dept. of Computer Science and Engineering,
POSTECH

Young-Ho Suh, Yong Kim
Sensitivity Engineering Department
Systems Engineering Research Institute

요 약

공동연구에 대한 필요성이 증대됨에 따라 이의 문제점으로 지적되는 시공간의 제약을 극복하기 위한 원격 공동연구 시스템 개발의 필요성은 더욱 증대되고 있다. 이 논문은 다자간 통신, 실시간 처리, 미디어 동기화 등의 기능들의 제공을 목표로 하는 MAESTRO라 불리는 객체 지향 분산 멀티미디어 시스템을 기반으로 개발된 원격 공동연구 및 실험시스템 플랫폼인 원격 화상회의, 전자 공책, 화이트 보드, 채팅 도구, 응용 공유, 세션 관리 도구 등 원격 공동연구 플랫폼의 설계와 구현을 설명한다.

1. 서론

원격 공동연구 및 실험시스템 구축을 위해서는 원격 화상회의, 화이트 보드 등의 기능뿐만 아니라 전자 공책, 대화형 메세지 전달 도구, 원격기기 모니터링, 원격 정보검색, 분산 공유 멀티미디어 D/B 등이 통합된 플랫폼이 필요하다.

1990년대 들면서 많은 멀티미디어 응용 프로그램이 개발되고 있으며, 원격 공동연구를 할 수 있는 환경에 대한 관심도 높아져 실제로 상품화된 경우[4, 5, 6, 7]도 있지만, 아직 이들 응용 프로그램들은 원격 공동연구와 실험시스템에서 필요한 도구들을 모두 제공해 주고 있지는 않다.

이 논문에서는 MAESTRO [1, 2, 3]를 기반으로 설계하고 구현된 원격 공동연구 플랫폼에 대해 설명한다. MAESTRO는 다양한 분산 멀티미디어 응용 프로그램들을 쉽게 개발하고 운용할 수 있는 시스템으로, 통신 서비스, 세션 서비스, 멀티미디어 데이터베이스 서비스, 네임 서비스, 관리 서비스, 품질 관리 서비스, 보안 서비스 등을 제공한다. 이들은 멀티미디어 응용 프로그램을 쉽게 개발할 수 있고 응용 프로그램들이 안정적이고 효율적으로 운영될 수 있는 환경을 제공한다. MAESTRO의 분산 멀티미디어 서비스들은 OMG CORBA[8, 9]를 이용하여 설계되고 구현되었다.

개발한 원격 공동연구 플랫폼에는 원격 화상회의, 전자 공책, 화이트 보드, 채팅 도구, 응용 공유, 세션 관리 도구 등이 있다.

2. MAESTRO 개요

이 장에서는 MAESTRO에 대해서 개략적으로 설명한다.

2.1. MAESTRO의 구조

그림 1은 MAESTRO의 계층적 구조를 보

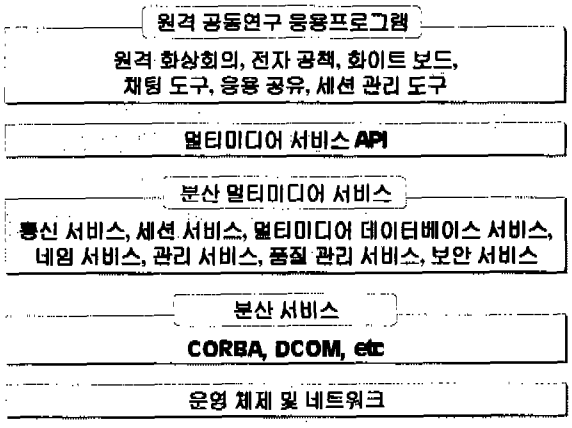

그림 1. MAESTRO의 계층적 구조

여 준다. 맨 위층에 원격 화상회의, 전자 공책, 화이트 보드, 채팅 도구, 응용공유, 세션 관리 도구 등 다양한 멀티미디어 응용 프로그램들이 존재한다. 멀티미디어 응용 프로그램들은 MAESTRO 상에서 여러 구성 객체들로 분산될 수 있으며 분산된 각 객체들이 서로 상호 작용하면서 하나의 응용 프로그램을 구성할 수 있다.

응용 프로그램의 아래층은 멀티미디어 서비스 API를 제공한다. 이 API 층은 객체 지향 방법론을 이용해서 설계되었으며 여러 클래스 정의들로 구성된다. API 아래층에는 분산 멀티미디어 서비스를 제공해 주는 여러 분산 객체들이 존재한다. 이 층의 분산 객체들이 제공해 주는 서비스들로 통신 서비스 (Communication Service), 세션 서비스 (Session Service), 멀티미디어 데이터베이스 서비스 (Multimedia Database Service), 네임 서비스 (Name Service), 관리 서비스 (Management Service), 품질 관리 서비스 (QoS Management Service) 그리고 보안 서비스 (Security Service) 등이 있다. 이 층의 객체들은 CORBA [8, 9]나 DCOM [10] 등과 같은 분산 객체 기술을 이용해서 형성될 수 있고 원격 공동연구 응용 프로그램들은 이 객체들이 제공하는 여러 서비스들을 run-time시에 사용할 수 있다.

분산 멀티미디어 서비스 층의 아래층에는 분산 서비스를 가능하게 해주는 미들웨어가 존재하는데 분산 객체 기술에 해당되는 CORBA나 DCOM 등의 기술이 이용될 수 있다. 가장 아래층에는 다양한 운영체제 (Solaris, HP-UX, AIX, Digital Unix, IRIX, Linux, Windows NT 등)와 네트워크 기술 (ATM, FDDI, SONET, Fast Ethernet, Token Ring)이 존재할 수 있다.

위 구조에서 MAESTRO를 구성하는 핵심은 멀티미디어 서비스 API 층과 분산 멀티미디어 서비스 층이다.

2.2. MAESTRO 멀티미디어 서비스 API

이 절에서는 MAESTRO의 멀티미디어 서비스 API 층에 대해서 설명하는데 멀티미디어 서비스 API를 정의하는 목적은 다음과 같다.

- 멀티미디어와 관련된 여러 클래스 정의들을 제공함으로써 멀티미디어 응용 프로그램들이 멀티미디어와 관련된 문제들을 쉽고 유연하게 사용할 수 있도록 한다.
- 멀티미디어 응용 프로그램 개발자들에게 공통된 인터페이스를 제공함으로써 새로운 멀티미디어 응용 프로그램들을 쉽게 개발할 수 있도록 한다.
- 여러 멀티미디어 응용 프로그램에서 중복적으로 사용되는 코드들을 멀티미디어 서비스 API에 포함시킴으로써 코드의 재사용 정도를 높인다.
- 비디오, 오디오 등을 포함하여 다양한 미디어들을 처리하고 화면에 출력할 수 있도록 함으로써 멀티미디어 응용 프로그램들이 다양한 미디어들을 쉽게 다룰 수 있도록 한다.
- 멀티미디어 응용 프로그램의 개발 시간을 단축시킨다.

2.3. MAESTRO 분산 멀티미디어 서비스

이 절에서는 분산 멀티미디어 응용 프로그램들을 수행하고 관리하는데 중요한 역할을 차지하는 MAESTRO의 분산 멀티미디어 서

비스들에 대해서 설명한다. 현재 통신 서비스, 세션 서비스, 멀티미디어 데이터베이스 서비스, 네임 서비스, 관리 서비스, 품질 관리 서비스, 보안 서비스의 7가지 분산 멀티미디어 서비스가 정의되어 있다. 각 분산 멀티미디어 서비스는 분산 객체들로 구성되며, 이 분산 객체들이 해당 서비스를 제공해 준다.

2.3.1. 통신 서비스

통신 서비스는 멀티미디어 응용 프로그램들이 다양한 종류의 미디어들을 서로 주고받을 수 있도록 해 준다. 다양한 종류의 멀티미디어 응용 프로그램들을 지원하기 위해서 통신 서비스는 다자간 통신, 흐름 제어, 신뢰성 있는 통신, 대역폭 제어, 미디어 동기화, 실시간 통신 등의 기능을 제공한다.

2.3.2. 세션 서비스

만약 여러 사용자들이 공동 작업을 하고자 한다면 이 사용자들은 여러 종류의 응용 프로그램들을 동시에 사용하여 서로 공동 작업을 하게 된다. 이럴 경우 공동 작업을 하는 사용자들을 하나의 그룹으로 만들어서 각 사용자들에 대한 정보와 각 사용자들이 사용하는 응용 프로그램들의 정보를 관리하여야 한다. 세션 서비스는 이와 같이 공동 작업을 하고자 하는 사람들을 하나의 그룹으로 묶어주는 서비스이다. 즉 세션 서비스를 이용해서 하나의 세션을 생성하거나 원하는 세션을 찾거나 세션에 참여하거나 참여한 세션을 떠나거나 생성한 세션을 없앨 수 있다.

2.3.3. 멀티미디어 데이터베이스 서비스

멀티미디어 서비스 API 층에서 다양한 종류의 미디어 객체들이 정의되었다. 멀티미디어 데이터베이스 서비스는 기본적으로 멀티미디어 서비스 API 층에서 정의된 여러 미디어 객체들을 저장하거나 저장되어 있는 미디어 객체들을 인출할 수 있도록 해 준다. 멀티미디어 데이터베이스 서비스는 이와 같은 기본적인 서비스 위에서 Video On Demand (VOD)나 디지털 라이브러리 서비스와 같은 보다 상위 레벨의 서비스들을 제공할 수 있다.

2.3.4. 네임 서비스

분산 멀티미디어 서비스들을 제공해 주는 객체들은 네트워크 상에서 분산되어 있다. 이런 환경에서 멀티미디어 응용 프로그램들은 자신이 원하는 서비스를 제공해 주는 객체들을 쉽게 찾을 수 있어야 한다. 네임 서비스의 역할은 특정 객체에 특정 이름을 달아서 응용 프로그램들이 이름만을 가지고 특정 객체를 찾을 수 있도록 하는 것이다. 따라서 어떤 객체가 서비스를 제공하고자 할 때는 네임 서비스를 이용해서 자신의 이름과 자신을 등록해야 하며 응용 프로그램들은 객체의 이름만을 가지고 해당 객체를 찾을 수 있다.

2.3.5. 관리 서비스

관리 서비스 [11]는 MAESTRO에서 분산 멀티미디어 서비스를 제공하는 분산 객체들과 응용 프로그램들을 관리하기 위해서 필요하다. 이 서비스는 분산 객체들과 응용 프로그램들을 모니터하여 그 정보를 제공해 주거나 분산 객체들과 응용 프로그램들의 장애를 해결하거나 분산 객체들과 응용 프로그램들을 제어할 수 있는 기능을 제공한다. 관리 서비스는 분산 객체들이 보다 효율적이고 안정적으로 서비스를 제공할 수 있도록 하여 응용 프로그램들이 큰 문제없이 수행될 수 있도록 한다.

2.3.6 품질 관리 서비스

지금 사용되고 있는 대부분의 네트워크들은 멀티미디어 응용프로그램의 요구사항을 충족시키지 못하고 있다. 품질 관리 서비스 [12]는 이러한 응용프로그램의 요구를 보장하기 위한 서비스이다. 즉, 사용자가 원하는 품질만큼의 서비스를 보장한다는 것이다. 물론, 사용자가 최상의 품질을 요구한다고 해서 그것을 다 들어줄 수 있는 것은 아니다.

2.3.7 보안 서비스

보안 서비스는 MAESTRO 시스템 이용 환경의 보안을 유지하여 준다. 사용자의 등록을 관리하여, 미리 등록되지 않은 사용자에게 접근 권한을 막고, 특별히 공개되지 않은 공동연구를 위해 암호 체크 등을 담당한다. 또한 저장된 멀티미디어 데이터의 접근

에 대한 권한도 제어한다.

3. MAESTRO 시스템 구현

MAESTRO 서비스를 구성하는 분산 객체들은 CORBA와 C++ 언어를 이용해서 구현되어 있다. 각 분산 객체들의 인터페이스는 CORBA IDL을 이용하여 정의되었다. 그리고 CORBA 제품으로 현재 가장 널리 사용되고 있는 IONA의 Orbix 2.3 [15]을 이용하였다.

또한 MAESTRO 멀티미디어 서비스 API의 미디어 장치중 카메라로부터 비디오 데이터를 읽어 들이기 위한 장치로 Sun Video Card [13]를 이용하였으며 Sun Video Card를 이용할 수 있게 해 주는 Solaris XIL 라이브러리 [14]를 이용하였다.

4. W/S 기반의 원격 공동연구 플랫폼

이 장에서는 앞서 설명한 MAESTRO를 이용하여 W/S 기반으로 개발된 원격 공동연구 플랫폼의 설계와 구현을 설명한다.

개발한 원격 공동연구 플랫폼은 다자간 회의를 가능케 하여 주는 **원격 화상회의**, 기존 연구노트의 기능을 대체할 뿐만 아니라 정보의 공유 및 원격 접근 기능등을 제공함으로써 연구자에게 편리한 환경을 제공하여 주는 **전자 공책**, 간단한 그림을 그리거나 문자를 입력할 수 있는 **화이트 보드**, 문자를 이용해 간단한 의견 교환을 가능하게 하여 주는 **채팅 도구**, 원격지의 응용프로그램을 실시간으로 공유하여 사용하는 기능을 제공하는 **응용 공유**, 그리고 세션의 생성, 삭제, 참가, 떠남 등의 기능을 제공하여 주는 **세션 관리 도구** 등 이다. 이 응용 프로그램들은 MAESTRO의 서비스를 이용하여, 쉽게 구현할 수 있다.

각 응용 프로그램들은 통신 서비스에서나 세션 서비스 등의 서비스를 받을 수 있는 인터페이스를 이용하여, 데이터를 보내거나 받을 수 있고, 임의의 공동연구에 합류할 수 있다. 각 원격 공동연구 플랫폼은 각 분산 멀티미디어 서비스와의 인터페이스를 정의하는 class를 상속받아 정의한다. 분산 멀티미디어 서비스의 통신 서비스와의 인터페이스를 정의한 CSO_API, 세션 서비스의 SSO_API, 멀티미디어 데이터베이스 서비스의 MDBSO_ API, 네임 서비스의 NSO_API, 관리 서비스의 MSO_API, 품질 관리 서비스의 QMSO_ API, 보안 서비스의 SeSO_API 중 필요한 서비스의 API를 상속받아서 그것이 제공하는 서비스를 이용한다.

부록에는 7가지 서비스 중 통신 서비스와 세션 서비스를 이용하여 C++로 구현된 인터페이스의 헤더 파일이 실려 있다.

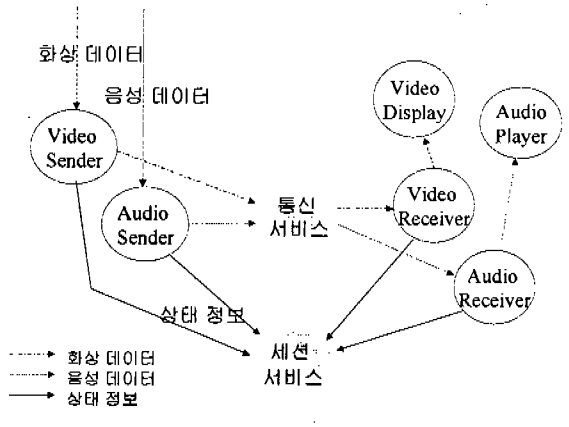

그림 2. 원격 화상회의의 데이터 흐름도

그림 2는 원격 공동연구 플랫폼 중 하나인 원격 화상회의의 데이터 흐름도 (Data Flow Chart)를 나타낸다. 원격 화상회의는 통신 서비스와의 인터페이스인 CSO_API를 상속받은 후, 통신 서비스를 해 주는 CORBA 객체인 *Communication Factory*에 bind하여 응용 프로그램이 사용할 *Port*를 할당받은 후, 데이터 통신 품질이 *q*인 *Channel*을 만들어 자신이 공동연구의 개설자가 되거나, 다른 *Channel*에 접속하여 다른 사람이 만들어 놓은 공동연구에 참여 할 수 있다. 사용자로부터 입력 받은 데이터는 채널을 통해 통신 서비스에 보내지고, 현재 참여하고 있는 사용자에게 보내어진다. 또한 통신 서비스로부터 다른 참여자의 데이터를 받는다. 또한 원격 화상회의는 SSO_API를 상속받아서, 세션 서비스를 수행하는 CORBA 객체인 *sso*에 bind한 후에 자신의 정보를 세션 서비스에 등록하거나, 응용 프로그램이 끝나기 전 세션 서비스에 등록되어 있는 자신의 정보를 삭제한다. 이와 같이 설계하고 구현된 원격 화상

회의를 이용하여 주고 받는 화면을 다음 그림 3에서 볼 수 있다.

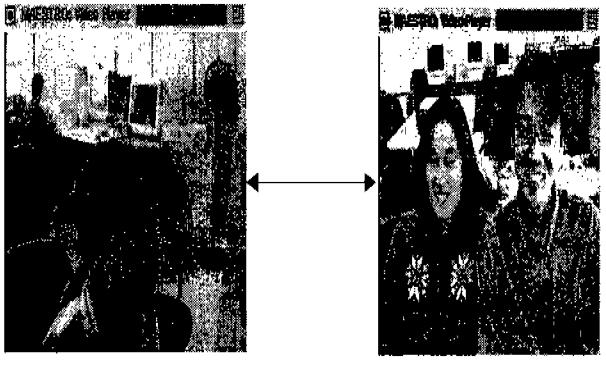

그림 3. 원격 화상회의

다른 원격 공동연구 플랫폼은 MAESTRO에서 제공하는 서비스를 이와 같은 방법으로 이용하여 쉽게 구현이 가능하다.

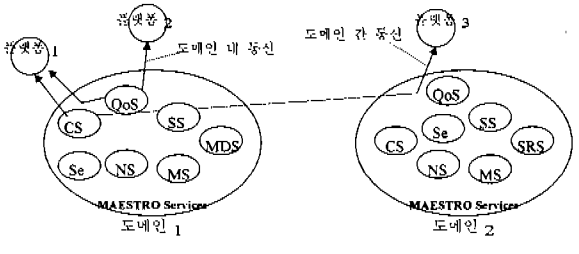

그림 4. 플랫폼 간 통신

그림 3은 분산 멀티미디어 서비스를 이용하여 원격 공동연구를 진행할 때 나타나는 각 분산 멀티미디어 서비스 객체들과 원격 공동연구 플랫폼이다.

도메인이란 분산 멀티미디어 서비스들이 멀티미디어 응용 프로그램들에게 제공되는 영역으로 정의되는데 도메인의 개념은 분산 멀티미디어 서비스들의 관리를 쉽게 하고 확장성을 높이기 위해서 사용된다.

각 플랫폼은 자신의 분산 멀티미디어 서비스 객체를 이용하여 다른 플랫폼과 도메인 내 통신이나 도메인 간 통신을 통해 데이터를 전달하고 받을 수 있다.

그림 5의 세션 관리 도구는 세션 서비스가 제공하는 기능을 쉽게 이용할 수 있도록하는 도구로서 사용자는 세션 관리 도구를 통해 공동연구에 참가하여 그 공동연구에서 사용중인 플랫폼을 실행하여 데이터를 주고 받는다. 따라서 세션 관리 도구는 다른 플랫폼을 실행할 수도 있다.

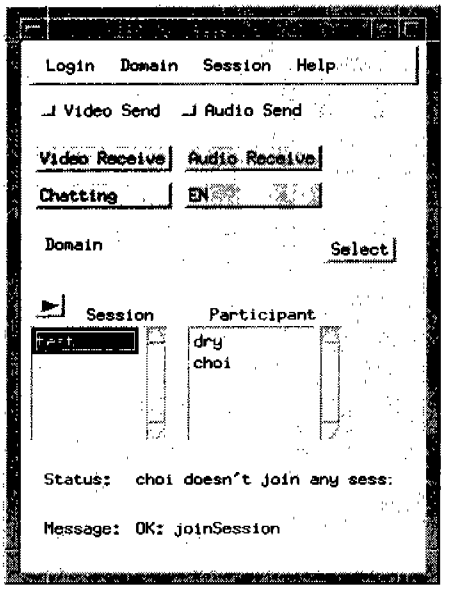

그림 5. 세션 관리 도구

원격 공동연구 플랫폼은 C++ 언어를 이용해 구현하였으며 MAESTRO에서 제공하는 서비스를 호출함으로써 다자간 통신, 기존 세션 참여 등 여러 가지 서비스를 이용할 수 있게 된다.

그림 4, 5는 개발한 플랫폼 중 원격 화상회의와 세션 관리 도구의 화면이다.

5. 결론 및 향후 과제

이 논문에서는 W/S 기반의 원격 공동연구 플랫폼을 설계하고 구현하였다. 플랫폼은 분산 멀티미디어 응용프로그램을 개발할 수 있는 환경을 제공하는 MAESTRO를 이용하여 쉽고 효율적으로 개발하였다. 개발한 플랫폼은 모두 6가지 인데, 이들 외에 더 필요하다고 생각되는 응용 프로그램들은 추가로 개발되어질 수 있을 것이다.

[참고문헌]

[1] J. W. Hong, T. H. Yun, J. Y. Kong, and Y. M. Shin, "A Flexible and Reliable Distributed Multimedia System for

Multimedia Information Superhighways", *Malaysian Journal of Computer Science*, Vol. 10, No. 2, December 1997, pp. 1-16. .

[2] T. H. Yun, J. Y. Kong and J. W. Hong, "A CORBA-based Distributed Multimedia System", *Proc. of 1997 Pacific Workshop on Distributed Multimedia Systems*, Vancouver, Canada, July, 1997, pp. 1-8.

[3] T. H. Yun, J. Y. Kong and J. W. Hong, "Object-oriented Modeling of Distributed Multimedia Services", *Proc. of IEEE International Conference on Communications*, Montreal, Canada, June, 1997, pp. 777-781.

[4] Sun Microsystems, ShowMe, http://www.sun.com/prodcts-n-solutions/sw/ShowMe/products/ShowMe_Overview.html.

[5] Silicon Graphics, InPerson 2.2, http://www.sgi.com/Products/software/InPerson/ipintro.html.

[6] Intel, ProShare, http://www.intel.com/proshare/conferencing/.

[7] Cornell University, Cu-SeeMe, http://www.visc.vt.edu/succeed/videoconf.html.

[8] OMG, The Common Object Request Broker: Architecture and Specification Revision 2.0, July 1995, OMG TC Document.

[9] OMG, CORBAservices: Common Object Services Specification, March 1995, OMG Document Number 95-3-31.

[10] Microsoft, DCOM: A Business Overview, http://www.microsoft.com/ntserver/info/dcom.htm, August, 1997.

[11] J. Y. Kong, J. W. Hong, J. T. Park and D. J. Kim, "A CORBA-Based Management Framework for Distributed Multimedia Services and Applications", *Proc. of the Distributed Systems: Operations and Management*, Sydney, Australia, October 1997, pp. 132-144.

[12]홍원기, 김종서, 강영민, 신영미, "응용 서비스 품질 관리를 위한 관리 대상 요소 선정에 관한 연구", 최종연구보고서, 한국전자통신연구원, 1998

[13] Sun Microsystems, Sun Video User's Guide, August 1994.

[14] Sun Microsystems, Solaris XIL 1.1 Imaging Library Programmer's Guide, November 1993.

[15] IONA, Orbix 2, IONA Technologies Ltd., November 1995, Release 2.3.

부록: 통신 서비스, 세션 서비스 인터페이스

● cso_api.h

```
class CSO_API {
  private:
    Communication::QOS_Parameter q;
    Communication::CommunicationFactory_var
        cf;
    Communication::Port_var port_a;
    Communication::Channel_var ch_a;
  public:
    CSO_API(char* domain);
    ~CSO_API() {};
    void set_port_number(int portNum);
    int get_port_number();
    void create_channel();
    void destroy_channel();
    void connect_channel(char* dst_domain,
        int portNum);
    void disconnect_channel();
    char* recieve_message();
    void send_message(char *message);
};
```

● sso_api.h

```
class SSO_API {
  private:
    SSO_var sso;
    char* _discrimination;
  public:
    SSO_API(char* discrimination,
        char* domain);
    ~SSO_API();
    void set_port_number(int portNum);
    void destroy_application();
    void leave_application();
};
```