Evolution of CORBA Framework: An Experience Study

Ruslan Shevchenko ¡Ruslan@Shevchenko.Kiev.UA¿
Grad-Soft ltd, Kiev, Ukraine
http://www.gradsoft.kiev.ua
Anatoliy Doroshenko ¡dor@isofts.kiev.ua¿
Institute Of Software Systems
National Academy Of Science of Ukraine

Abstract

In this experience study some evolution of an IT Enterprise distributed component framework based on CORBA architecture is presented. Some methodological and pragmatical aspects of software maintaince and evolution are discussed. Some additions to CORBA standards are proposed along with roadmap of future development.

1 Introduction

The task of building and maintance of integrated enterprise system is quite different from traditional software development. The main pecularities are:

- Lifecycle issues
  - We have different business processes which have different lifecycle requirements to appropriate software systems. On one hand there exist business processes which must be changed very quickly reflecting dynamic management behavior and on the other one — there are also business processes that must be more detailed and more stable to reflect long term production issues. So lifecycle of integrated software system consists of number of shorter and longer lifecycles of application and system software.

  - Evolutionary changes: we must be able to adopt systems to new technology or to new platform in extensive way as changing whole system requires significant investments. So software components must have isolated well-defined external interfaces and these interface of software component must have longer lifecycle than component itself. Therefore theyt must be standardized.

- Complexity issues: enterprise-wide systems are quite complex. Existing modeling tools does not allow us to split this complexity in reasonable way. Yet another problem is knowledge transfer inside development team. Development in such systems requires domain specific knowledge which usually does not exist in written form.

- Integration issues: we must have standard interfaces for common infrastructure of enterprise software system such as relational databases and intraweb.

In this paper we analise how this peculiarities can be handled in CORBA-based object-oriented framework and what adoption is required to standard development technique and software engineering methods.

2 CORBA as object-oriented framework

CORBA is a vendor-neutral standard for distributed object architecture, supported by Object Management Group - consortium from over 700 companies. CORBA gives us language independent object oriented framework for distributed computing based on open standards [4]. The CORBA itself defines following major parts:

- ORB core — framework which provides infrastructure for CORBA Object model. Interfaces for different parts of distributed systems must be described in CORBA IDL (interface definition language) and then can be translated into implementation language stubs or loaded to interface repository. Appropriate language entities (servants) which implement CORBA objects can be written on a programming language of choice.

- CORBA Services — a set of interfaces which provide API to application programmers for some 'horizontal' common functionality. Among them are Naming Service, Event Service, Transaction Service and so on (about 15 services).
- Vertical Domain Interfaces — a set of services specific to some vertical domains such as financial or airline scheduling.

Future development of standard is concentrated on building component framework [5] and integration IDL with software modeling techniques.

3 Experience Study

The framework of CORBA presented above looks good in theory. We implemented the first CORBA based system in 1997 and are maintaining and extending it up to today. Along these years we realised that CORBA is a nice framework for splitting system into distributed parts. But now the scope of CORBA usage is limited to this functionality. Some lessons we learned in these years are following:

- Lifecycle issues:
  - IDL is ideal for specifying system interfaces. But today we can not use automatically generated code for defining of fine-grained object isolation levels. The gap between CORBA objects and implementation servants is too big in the latest standards. Call of local object requires call of interceptors and determination of object in POA active object map. Usage of collocation optimization like [17], which was used in CORBA-2.0 ORB-s is limited in the latest standard. CORBA-2.4 have concept of local interfaces and valuetypes but process of building and deploying of local interfaces is not formalized. Recently adopted CORBA-2.5 specifications [6] will provide technique for creating local interfaces by application programmer.
  - Components with different lifecycle issues can be managed as separate processes. In most cases this is acceptable from pragmatic point of view, but CORBA does not provide standard way for automatic management of object implementations. Most ORB-s provide proprietary API for this.

    CORBA Components Framework will provide deployment containers.
  - Semantics of versioning interfaces in CORBA is too weak to be usable.

- Complexity issues: IDL allow greatly reduce complexity by splitting system into loosely-coupled parts and separate interface from implementation. Combining IDL with API documentation generator and UML tool give us nice framework for documenting system design.
- Integration issues: we discovered that CORBA standard does not provide interfaces to well-known elements of IT infrastructure such as access to relational databases and well-known authentification frameworks. Even worse, CORBA standards set define Query Service which can be used for relation database access, but semantics of this service too weak for direct usage. Most interesting that interfaces defined in Query Service does not allow effective implementation. So we invested a few months of work into useless standard, then did special research for CORBA performance issues [2], [3] and implement own framework for database access [15] which allows efficient queries evaluations.
- Software administration issues: CORBA has reached set of API for process-level software management such as implementation repositories [8] and trading service. From other side, library-level management layer is totally absent in general framework.

4 What is needed in CORBA to be ideal OO Framework

Today's CORBA purpose is organization of interfaces between remote parts. Extending its usage for modeling and logical interface definition framework which can be used in real life large scale projects requires additional efforts.

- Support of fine-grained object model: IDL is near ideal language for architectural description of system. But now in practice, it can be used only for describing connection between remote subparts because semantic of CORBA call require too much overhead which can not be fully eliminated by collocation optimization without breaking semantics of CORBA object model. One of CORBA features of 'location transparency' — local call is not differ from remote call on object level model — in real life can not be reached. Local and remote call have completely different overhead and software engineers must to distinguish remote and local objects during design process. More detailed, when we work with complex distributed systems, we have four levels of granularity:

1 the same host, the same process space and the same language. We have no overhead during call of local interface. This is semantics of CORBA local interface.

2 the same host, the same process space but another language. We have some marshalling overhead but not data transfer overhead. This semantics currently can not be expressed in CORBA model, the nearest analog is OS COM interface.

3 host is in the same LAN, another process space. Data transfer becomes the biggest source of call overhead. This is semantics of traditional CORBA interface.

4 host is in WAN. In this case we have not only data transfer overhead but also various proxy and firewall issues. This granularity is usually behind the scope of CORBA model.

Integration of components of granularity level 1 or 3 can be done using CORBA IDL for interface definitions. For granularity level 4 we have WSDL mapping.

We have no way to express granularity level 2 in CORBA model. On other hand, components of such granularity are often used in any big system for data access components integration and embedding of script languages.

Todays way is to use some low level tools such as SWIG [1] which breaks unified object model, or emulates granularity level 2 by granularity level 3 components which add process boundaries overhead for each inter-subsystem call. So some explicit notation for showing granularity level 2 components (we can name it as 'local foreign' can be useful.

Note that for making it possible we do not need in a inter-language binary interface standard as usually pointed.

Now almost all interpreted language are supplied with:

– shared library which contains language interpreter;

  - some API for interaction with C lanfuage such as Java JNDI or TCL forign language interface.
  - some access to dlopen() (or LoadDLL) functionality in system library.

  Compiled languages typically have some tools for linking with C libraries.

  So, almost for each language we have C binding and representation of language entities as C data types. They are different in each languages but can be unified by providing inter-language stubs. There exist such tools as Swig [1] which automatically generates such integration stubs for interconnections between 9 the most popular languages. Therefore, we do not need a standard for binary compability of components, we need binary compability for C stubs which already exists.

- Support of fine-grained packaging: we must be able to use CORBA local binary modules. This means, that:
  - application programmer can write implementation language file which will be compiled as separated entry of compilation;
  - there is a way to show ORB map of entry points and appropriative object or bytecode files;
  - ORB must be able to dynamically load appropriative object or bytecode file by providing some portable API.

  Currently standard technique for implementing deployment containers exists only for Java language. OMG raise RFP for this issue, C++ API prototype implementation is build by GradSoft [11].

- Support of relation database access: specifications of CORBA Query service is too weak to be usable. But it is not mean, that access to relation database is not needed. Persistent State Service does not cover all functionality of Query Service: it is intending only for object storage. As a result, each database vendor usually provide own implementation of some form of Query Service [9], [10]. We build implementation of 'better' Query Service in UAKQuery [15] which can be extended for full replacement of original Query Service, but with well-defined semantics and interfaces designed to allow high-performance implementation [16], [3].

- Support of web access: web access now is standard element of enterprise infrastructure. So there must exist a standard way for CORBA/Web interconnection. Note, that this must be not limited to IDL/WSDL mapping, but include more general issues, such as building presentation layer [2].

  Today two main directions of CORBA/Web integration are known, one is based on dynamic scripting [12] and another provides static API [14]. We believe that both approaches are suitable in different situations and must be standardized.

- Support of reflection and dynamic scripting: using a scripting language as a 'glue' between components in combination with reflective interface descriptions usially reduce project time and cost. But during applying this approach in CORBA environment we discovered, that exists technical limitations caused (again) by granularity of CORBA object model. Using refection in CORBA means using of remote object 'Interface Repository' (for local interfaces also). So, any using of refective techniques become ineffective. This problem is partially addressed in CORBA CCM specifications [5].

- Quality of specifications: obsolete specifications must be removed from standard set. Software engineers must sure that technology standards can be applied to real-life tasks. Also applying of formal methods to specification [7], [13] can be used for verification purpose.

- Versioning: built-in support of versioning of IDL specifications can not be used. Note that correct semantics definition of versioning on interface level is very difficult task itself. We need correct mechanism for fusion between versioning semantic and possible interface inheritance in another component. So we can recommend pragmatical approach: does not use build-in CORBA versioning, instead include version information to names of modules or interfaces.

5 Comparison of object frameworks

Following is a comparison of most popular technologies in context of future requirements for developments of enterprise-wide systems:

	$CORBA$	EJB	$DCOM$
$Designfeatures$			
$Isolated\ Interfaces\ Definitions$	Yes	$Partially$	$Partially$
$Formal\ methods\ applicability$	Yes	$Partially$	$Partially$
$Fine-Grained\ Object\ Model$	$Future$	Yes	Yes
$Coarse-GrainedObjectModel$	Yes	Yes	No
$Components$	$Future$	Yes	Yes
$Integration\ Feautes$			
$RDB\ Access$	$Third-Party$	Yes	Yes
$Web\ Access$	$Third-Party$	Yes	Yes
$Transactions\ Access$	Yes	Yes	Yes
$Scripting$	Yes	NO	Yes
$Interobility\ with\ other\ models$	Yes	Yes	No
$Maintance\ Feautures$			
$Versioning$	NO	NO	NO
$Deployement\ Containers$	$Future$	Yes	$Partially$
$Platform-Independence$	Yes	Yes	NO

So as we seen none of existing OO frameworks provide full set of features necessary for large-scale projects. Today preferred architecture of enterprise software development are: EJB-based for long-lifecycle projects, COM-based for projects with short lifecycle. Ironically, there are no standard-based technologies for automation connection of EJB and COM. CORBA in its todays state can not be used as general concept framework, but from other side it is most suitable to be extended to full-featured technology.

6 Conclusion

CORBA can be transformed from framework for description of distribution architecture into general object-oriented environment which would be suitable for modeling and developing of large-scale software complexes. In this paper the necessary extensions to object model are described. They can be implemented as extensions of IDL and CORBA Component Model. Some of the necessary prototype extensions to services standard set are build by Grad-Soft [11] team and proved to be applicable in real-life projects. Some software is available from Grad-Soft web page, `http://www.gradsoft.kiev.ua`, free for non-commercial usage.

7 Akcnowledgements

I want to thanks Michi Henning - Chief Scientist of IONA Technologies, for review early version of this paper and discussion. This work was impossible without effort of Grad-Soft software engineers: Alexandr Yanovec, Vladimir Joss, Sergey Krisanov and others.

References

[1] Dave Beazley. *Swig: Simplicified Wrapper Interface Generator*, 1996-2001. http://www.swig.org.

[2] Ruslan Shevchenko. Anatoliy Doroshenko. A method of mediators for building web interfaces of corba distributed enterprose applications. *Lecture Notes in Informatics V. 4 - Proceeding of Information Systems Technology and its Applications 2001*, 2001. ISBN 3-88579-331-8.

[3] Ruslan Shevchenko; Anatoliy Doroshenko. Tecniques for increasing performance of corba based distributed applications. *Parallel Computer Technologies proc. 6-th int, conf. PACT2001 LNCS vol 2127 pp 319-328*, 2001. Novosibirsk, Russia.

[4] Object Management Group, editor. *The Common Object Request Broker: Architecture & Specification v. 2.3*. OMG, 1999. formal/99-10-07.

[5] Object Management Group, editor. *CORBA Components*. OMG, 1999. ccm/99-07-01.

[6] Object Management Group, editor. *The Common Object Request Broker: Architecture & Specification v. 2.5*. OMG, 2001. formal/2001-09-01.

[7] Remi Bastide; Philippe Palanque; Ousmane Sy; David Havarre. Formal specifications of corba sevices: Experience and lessons learned. *ACM SIGPLAN Notices Vol. 35 Num. 10*, 2000. Processing OOPSLA-2000.

[8] Michi Henning. Binding, migration and scalabilty in corba. 1998. http://www.ooc.com.au/ michi.

[9] Oracle Inc. *Oracle8i Enterprose JavaBeans and CORBA developers guide.*, 1999.

[10] Sybase Inc. *Sybase and CORBA Whiteparer*, 1998.

[11] Grad-Soft ltd. *GradSoft Home Page*, 1999-2001. http://www.gradsoft.com.ua.

[12] Phillip Merle and Others. *CORBAWeb home page*, 1996-2000. http://corbaweb.lifl.fr.

[13] Matteo Pradella; Matteo Rossi et al. A formal approach for designing corba based applications. *Processing of the 2000 International Conference on Software Engeneering June 4-11 Limerik Ireland*, 2000.

[14] Sergey Krisanov Ruslan Shevchenko. *mod_cbroker: Programming Guide*, 2000-2001. GradSoft-PR-e-05.12.2000-2.0.0.

[15] Ruslan Shevchenko. *UAKGQuery: Programming Guide*, 1999-2001. GradSoft-PR-e-16.06.2000-1.5.

[16] Ruslan Shevchenko. Analysis of efficiency enhancing methods of corba based distributed applications. *Proc. 2-nd Int. Conf. on Programming, May 23-26, 2000, Kiev, Ukraine*, 2000. pp 226-240 (in Russian).

[17] S. Vinoski. Collocation optimizations for corba. *C++ Report*, 1999. Vol. 11 No 9.