Architecture Notebook

COM2CORBA

Date: 2009-05-18
Version: 1.0.1 (revision 424)

This document

Summary This document presents the architecture for the COM2CORBA project.

Author Daniel Forsberg

Date 2009-05-18

Version 1.0.1 (revision 424)

Document history

Version	Date	Changes
0.1.0	2009-02-06	Document created.
0.2.0	2009-02-12	Document converted to LyX.
0.3.0	2009-02-22	Added key abstractions.
0.4.0	2009-02-25	Changes to layers and key abstractions.
0.5.0	2009-02-26	Added glossary.
0.6.0	2009-02-26	Updated chapter five.
0.6.1	2009-03-02	Removed inclusion of the glossary document.
1.0.0	2009-03-09	First version.
1.0.1	2009-05-18	Corrected version.

Contents

Chapter 1

Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements and any other overarching aspects of the system that shape the design and implementation.

Chapter 2

Architectural goals and philosophy

The main architectural goal is to provide a well defined architecture that facilitate maintenance and future development. Future upgrades on underlying tools should be considered in the design process. The COM interface has by tradition been most common on Windows platforms while many CORBA services are running on other operating systems. COM2CORBA will not be dependent on any specific hardware or operating system. At the current state there are no demands on performance of operation. COM2CORBA is a open source project and thus applies open source philosophy according to the community.

Chapter 3

Assumptions and dependencies

3.1 Assumptions

- The users of COM2CORBA wish to have a code generator that supports the latest versions of TAOs CORBA implementation.

3.2 Dependencies

The following dependencies applies given the assumptions in section 3.1.

- **COM:** Changes/additions to the standard might require changes in the COM backend.
- **IDL:** Changes/additions to the standard might require an update of TAO to a newer version.
- **TAO:** Future releases of TAO with changes in TAO's IDL compiler (more specific, the TAO frontend) will require upgrades of COM2CORBA. Changes to the CORBA standard requires an updated version of TAO but does not affect COM2CORBA in any other way. Additional changes to the TAO backend will not affect COM2CORBA.

Chapter 4

Architecturally significant requirements

For a more in depth description of the architecturally significant requirements, review the system wide requirements document.

- Use TAO for the CORBA C++ code generation.
- Separate frontend and backend.
- Generate similar output as the earlier version.
- Easy to upgrade and maintenance.

Chapter 5

Decisions, constraints and justifications

5.1 User interface

The user interface will consist of solely command line operations. This makes COM2CORBA easy to integrate in automated processes. A well defined command line interface makes a solid base for easy developement of a graphical user interface frontend for COM2CORBA if the need arises.

5.2 Reuse of the TAO code generator

The decision to reuse of TAO is based on the system-wide requirements. It simplifies future upgrades for the user and it also reduce development cost. TAO is a well known CORBA middleware framework and have seen years of testing in operating environment.

5.3 Performance requirements

There exists no performance requirements for code generating.

Chapter 6

Architectural mechanisms

6.1 Error management

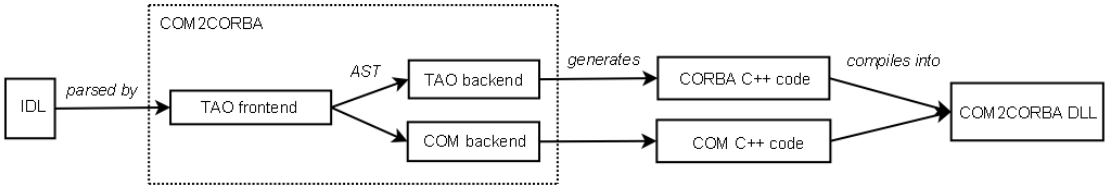
Allows errors to be detected, propagated, and reported.

6.2 Debugging

Provides elements to support application debugging.

Chapter 7

Key abstractions



7.1 TAO frontend

The TAO frontend reads the input (IDL) and process this information, creating a binary internal representation called AST.

7.2 TAO backend

The TAO backend generates C++ classes based on the AST that encapsulates calls to a CORBA service.

7.3 COM backend

The COM backend generates COM-specific C++ code such as interfaces based upon the AST.

Chapter 8

Layers

COM2CORBA uses a two-layered architecture. Follow the visibility pattern while designing each layer to ensure that elements may depend only on components in the same layer and the next-lower layer.

- Layer 1 includes the TAO frontend, also called the frontend layer.
- Layer 2 includes the TAO backend and the COM interface backend. This layer is also called the backend layer.

Chapter 9

Architectural views

9.1 Use cases

From an architectural point of view we have two distinct use cases. Each one of them should be considered while making design decisions.

- **User:** A person or program that provides a IDL description and want COM2CORBA to generate the necessary output to build a CORBA DLL-file.
- **Developer:** A person that does maintenance to the system (e.g. upgrading to a new version of TAO) or adding new features.