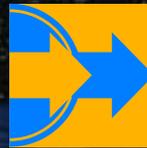




Bridging the Gap Between Internet and Mainframe

Presented by Ruth Stekl
ruth@klst.com



Legacy Wrapper Framework

Abstract



Legacy
Wrapper
Framework



KLEIN + STEKL
Gesellschaft für
Anwendungsberatung mbH

We have built a productive distributed OOP and ORB solution which makes the functionality of existing systems available to new applications thus enabling on-line processing across different systems. This Legacy Wrapper Framework offers a general central function interface to existing programs defined from the business view by means of business requests. This makes possible not only the integration of both existing and new software, but also the integration of old and new technology such as mainframe and internet/intranet.

By the term "legacy applications", we mean existing programs, often coupled with large amounts of data, which work very well and cannot or do not need to be replaced. A legacy wrapper encapsulates such a legacy application thus making it available for new technologies like the internet. The Legacy Wrapper Framework is a framework for implementing such a wrapper which integrates legacy applications running on different systems.

In our productive solution, Unix or PC clients (implemented in languages such as Java, Delphi or C++) transmit function calls to a legacy wrapper residing on a mainframe. This wrapper takes on the role of a server and business request broker. Function calls are sent to the Legacy Wrapper Framework via standard technologies such as IBM MQSeries or Iona Orbix. Depending on the function call, the legacy wrapper controls legacy applications via the best fitting interface (API, IMS program switch, EHLLAPI, direct IMS or DB2 database access) and returns the resulting data to the client. The legacy applications are treated as independent components and do not need to be changed to be integrated.

Due to reuse and synergy, the introduced overall system offers easy access, high stability and security, short response times and a small error rate thus promising a very short pay back period at the same time.

Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000

P. 2

This material is published subject to the usual copyright regulations.

To contact Klein+Stekl on general purposes, please send us an email to info@klst.com. Questions or comments concerning these slides will be appreciated by Ruth Stekl. Please email to ruth@klst.com. If you are preparing a conference on OO business and are interested in a contribution from us, please feel free to write to us. We will get in touch with you as soon as possible.

Thank you!

Overview

- Motivation: What's the Problem?
The Deep Gap Between the Old
and the New World
- Concepts: Which is the Basis we
Build Upon?
 - Component Theory
 - Legacy Wrapping
 - Interface Defined from
Business View
 - Object Oriented Thinking
- Architecture: How Can we Throw
the Bridge?
Accessing Legacy Applications
- Our Implementation: The Bridge
Has Been Built!
Connecting the Mainframe to the
Internet World
- Summary: It's alive!

1. Motivation

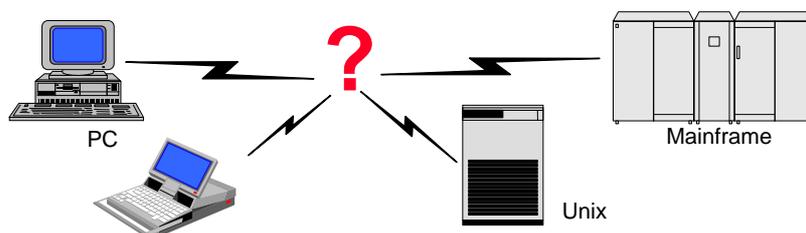
The Deep Gap Between the Old and the New World

Talking about the Old World, we have in mind machines such as a mainframe and languages such as COBOL. Inherited functionalities are often called legacy applications. In many cases, there is no implementation documentation and therefore no deep knowledge of these legacy applications. The same refers to legacy data bases. But in this context, we do not only refer to COBOL applications on the mainframe as "legacy applications", but in general to all already existing applications that are not easily accessible by all new applications.

With the New World, we associate internet or intranet, OO-languages such as Java or OO-concepts such as CORBA. In general, all new implementations are included. In this first part of the presentation, we describe the problem, the deep technological gap between the Old and the New World.

Motivation

- Trend
Standard or customized PC software needs online data access as well as access to existing functionalities, e.g. provided by mainframe
- Problem
New applications must be integrated with existing 'legacy' applications, possibly resident on other platforms



Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000 P. 5

We think there is a global trend in business to buy standard or customized software rather than producing own solutions, as companies used to do. And there are new possibilities like platform-independent languages (Java) and client-server systems including internet clients. But of course the new applications need to communicate with the existing ones, and online access to data or functionalities is highly favoured.

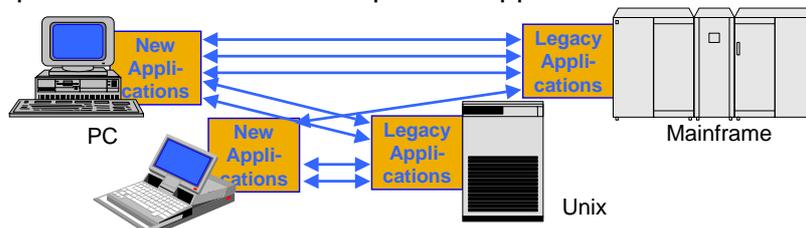
Now the problem arises: How can existing legacy and new applications, possibly located on different platforms, be integrated? And how can this be done at appropriate expenses and in short time?

Usual efforts include the paper interface, but this is not sufficient. It is time consuming and error prone because of the double entry: sometimes it needs days until typing errors are corrected.

To make the process more secure with regard to entry errors, the next step is implementing a file interface. One application prints out to a file, the file is transferred to the target system and processed. A usual way is collecting the results of the day and processing them over night - we are still miles away from online connections.

Motivation

- Trend
Standard or customized PC software that needs online data access as well as access to existing functionalities, e.g. provided by mainframe
- Problem
New applications must be integrated with existing 'legacy' applications, possibly resident on other platforms
- Usual Online Solution
Special interfaces between pairs of applications



Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000 P. 6

To enable online processing, usually special interfaces are built from each new to each existing application. Caused by the lack of synergy, there are n times m interfaces to implement. The implementation efforts are high because the client implementors need to know the legacy application or data to be accessed as well as their system environment. And this highly complex system is very cost intensive to maintain. Additionally, you have to adjust all programs that use one legacy application each time that you change it.

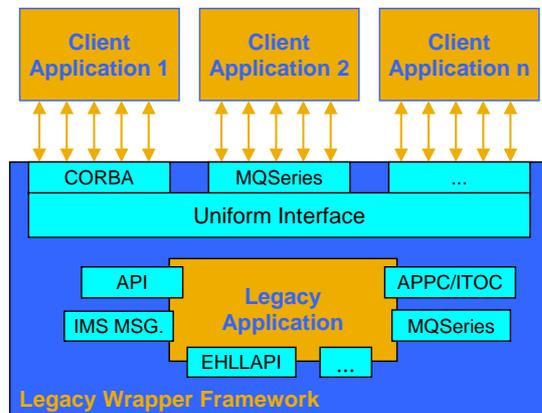
This is a way to achieve online access, but at high expenses.

2. Concepts Which is the Basis we Build Upon?

In this second chapter, we account for the fundamental ideas we have used to reach our goal: On this basis, we have built the bridge that crosses the technological gap.

Concepts: Component Theory, Legacy Wrapper

- Legacy applications are treated as components
- Access to legacy applications according to their abilities
- No changes within legacy applications
- Legacy wrapper provides access via a uniform interface



We treat all applications as single components offering interfaces for communication, at least a user interface.

The point of a legacy wrapper is wrapping technical issues and hiding legacy applications from those applications in need of the legacy functionalities or data. One wrapper can make possible communication with many legacy applications; in order to minimize network traffic, it is useful to install the wrapper on the same machine as the legacy applications. The legacy wrapper may become a server adapter offering the needed functionalities of all legacy applications on its machine.

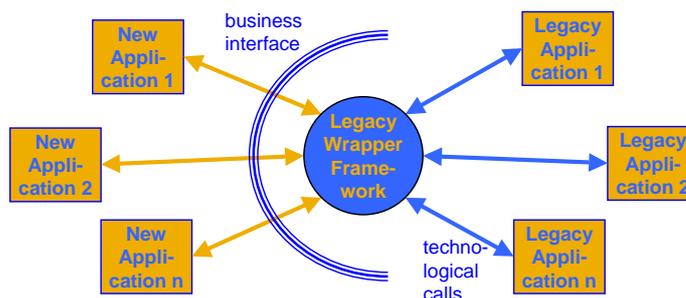
The important feature is that the legacy application is not to be changed to get wrapped. The wrapper uses the already existing interfaces, whatever is available and will suit the needs best.

As a component, the wrapper provides a uniform well-defined interface for client programs. So the client does not need to know anything about the technological details of the legacy application, it does not need to know whether there is a legacy application behind the scenes at all. This uniform client interface is offered by a small number of standard technologies such as CORBA or messaging and queuing (e.g. IBM MQSeries).

Mark the difference between treating new and legacy applications: the legacy wrapper uses the **best interface offered** for the **legacy applications and data** and offers a **uniform interface via technical standard technologies** to the **new application**.

Concepts: Interface Defined from Business View

- Interfaces are defined from business view (Business Request Interface)
- New applications need not care about the technical and implementation issues of the legacy application (application constants, language, platform, location, ...)



Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000

P. 9

Instead of implementing n times m specialized interfaces, we suggest one single server program acting as translator:

The legacy wrapper shields applications behind a **technical** wall. In addition, the Legacy Wrapper Framework offers a **business** wall. The idea is not to defend the legacy applications against access, but to obtain a clearer business view independent of technological details.

The goals are hiding the legacy application and data (LA) as well as possible and uncoupling the applications as far as possible. If the interface of each task is defined in business terms, it becomes independent of the special codes and other implementation issues of the particular legacy application.

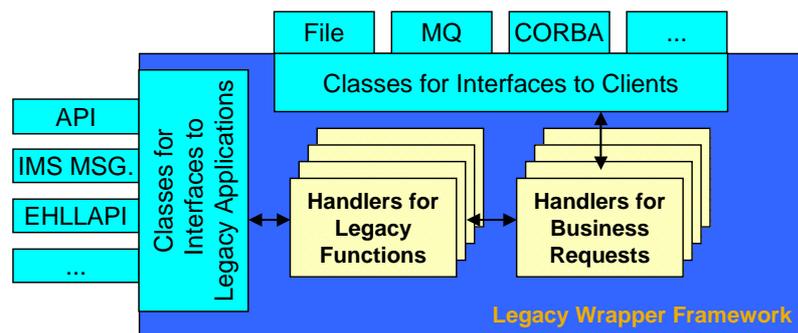
Example: If in one field of an IMS panel dealing with amortizations, the values "D" and "C" are valid choices, in a business oriented interface you will not use those constants. You will rather define constants like `delayOfPayment` and `cessationOfPayment`. This makes the interface definition easier to understand for the programmers, and the LA can change the values without any visible change for the user.

Our proposed server is not necessarily a bottleneck because we describe a logical unit that can be distributed or multitasked. As only data are handed from one application to the other, the computing time is minimal.

Internal OO Structure of the Legacy Wrapper Framework

A general framework

- with special classes for the interfaces
- with special classes for the business request handlers and for the function handlers in form of plug-ins



Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000

P. 10

The internal structure of our Legacy Wrapper Framework consists of a general framework responsible for task management.

In order to enable reading requests and sending results, the LW possesses special classes for each kind of client interface. *) Requests are dispatched to special business request handlers containing the business knowledge of how to deal with the tasks and which LA to call.

Because each functionality of an LA may be used by many request handlers, there are classes for each, e.g. for each API call or each IMS panel.

In order to gain reuse of technological knowledge, there are special classes for handling the technical details of the interfaces to the LAs.

All these classes are plugged into the framework and can easily be extended, adjusted separately or replaced.

*) We include a file interface for offline operation and testing.

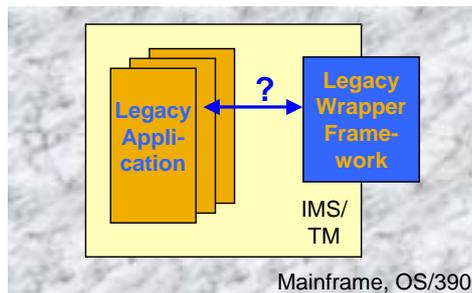
3. Architecture How can we Throw the Bridge?

Now that the foundations are laid out, we describe how to throw the bridge. The most difficult part is actually reaching the Old World, getting access to the legacy applications.

Architecture: Server Interfaces

- If possible: use API of legacy application ...best solution !
- Else, if possible: use IMS message-interface
- Else: use EHLLAPI

...if the legacy application
otherwise needed change



Legacy Wrapper Framework, RS - K+S

© KLEIN+STEKL 2000

P. 12

We show the access to legacy applications on a mainframe under IMS control. (If you are not accustomed to mainframe environments: IMS/TM is a transaction monitor analogous to Tuxedo on UNIX platforms or the MS-Transaction Monitor on PC.) Our legacy applications consist of transactions with panels (analogous to windows) as I/O.

The goal is gaining access to the legacy application without changing it.

If the transaction offers an API, the best way is to use direct function calls.

Other transactions may be able to answer via program-to-program switch. Using the IMS messages interface the caller acts according to the legacy application as a screen. This solution is not easy to implement but very fast.

In the other cases, the legacy wrapper has to use the user interface just like a human user, that is the EHLLAPI interface. Because this is often the only way, we look a little closer at it.

Tools employed: EHLLAPI

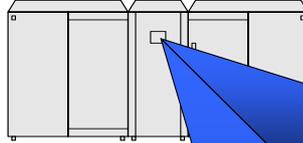
- **Emulator High Level Language Application Programming Interface**
- API for programming a 3270 emulation
- "Automated User"
- Developed by IBM, today part of most 3270 emulations
- Far spread on PC and UNIX 3270 emulators

 now made available within the mainframe
by **INTERSCREEN**

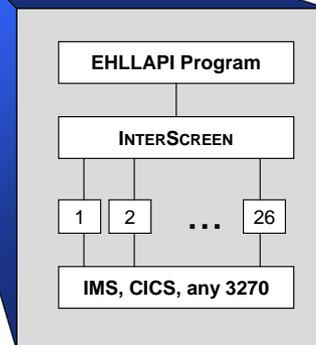
EHLLAPI is the API for programming 3270 emulations and for simulating automated users. This IBM standard is far spread and in common use both on PC and UNIX machines. But to avoid unnecessary network traffic and for security reasons, the Legacy Wrapper Framework should reside on the target system, i.e. the mainframe.

Tools employed: K+S INTERSCREEN

- **EHLLAPI within the Mainframe**



- High speed: full OS/390 power without utilizing LAN/WAN/...
- Security: uncontrolled direct access by PCs avoided
- Multisessioning: up to 26 sessions in parallel per process

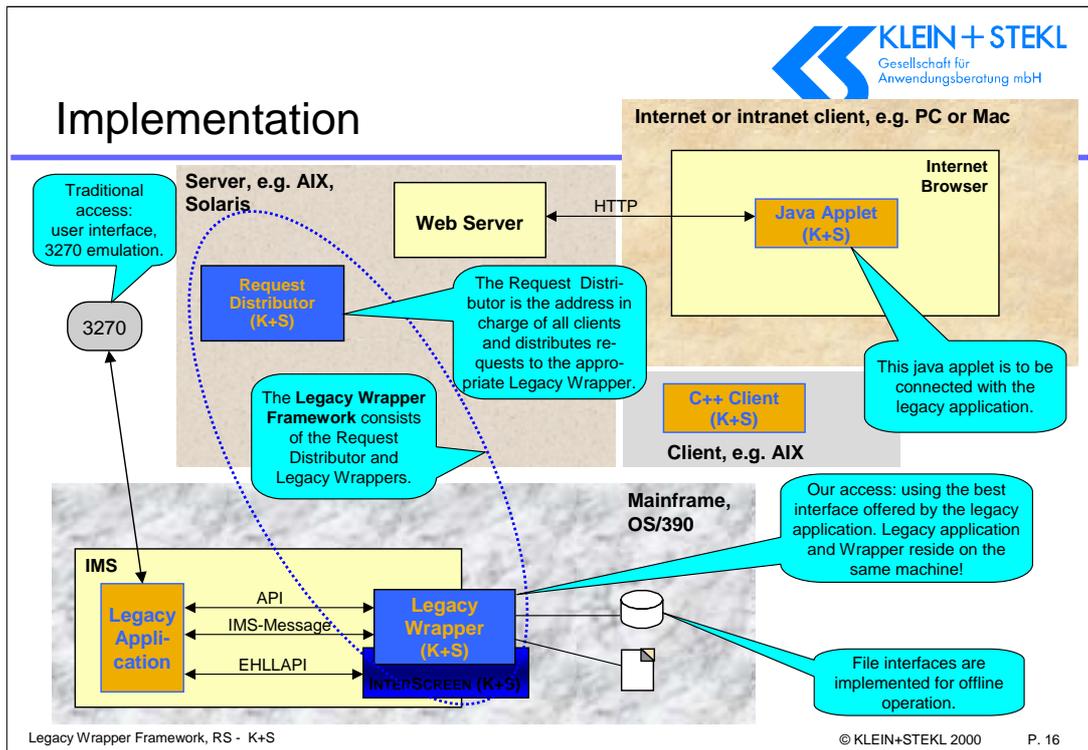


INTERSCREEN is a product developed by KLEIN+STEKL that offers EHLLAPI directly on the mainframe. It consists of the mainframe implementation of the API and several help programs, e.g. for sending a message to an IMS transaction and waiting for response.

4. Implementation The Bridge has been built!

In this last part, we will put all the building blocks together and present our implementation of the legacy wrapper.

We show you how to connect Java internet clients. Gaining access to "normal" standalone PC clients works similarly.



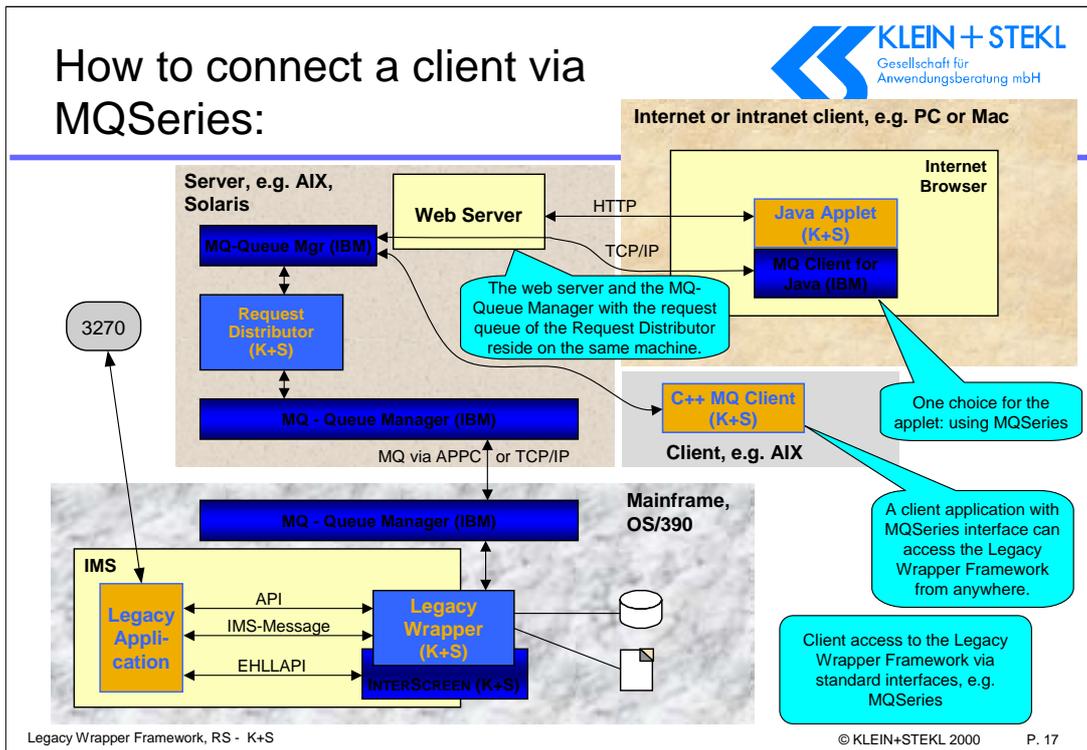
The situation is the following: Our legacy application consists of IMS transactions residing on a mainframe and with 3270 panels as user interface. Of course, they can be accessed by an old-fashioned 3270 terminal or an emulation.

We want to connect a Java applet to this application. Java applets run within an internet browser on any machine offering HTTP contact to a web server. Our web server may reside on an AIX or a Solaris server.

The Legacy Wrapper Framework consists of a Request Distributor and one or more Legacy Wrappers:

The Legacy Wrapper resides on the same machine as the legacy application and uses the best interface offered by the legacy application. The Legacy Wrapper makes direct use of the API and the IMS message interface. The user or EHLAPI interface of the legacy application is accessed via INTERSCREEN; this requires that the Legacy Wrapper runs under control of INTERSCREEN. File and list I/O are established for the legacy wrapper for the purpose of offline operation and testing.

The address in charge of all client calls is the Request Distributor. This Request Distributor offers the business interface via standard communication channels. It distributes the client requests to the appropriate Legacy Wrapper; this is important if you have legacy applications on different machines (see page 20).



The goal of the Legacy Wrapper Framework is integrating many different platforms and languages. Therefore, we employed IBM MQSeries as a standard messaging middleware for accessing the mainframe and for the internal communication between the different parts of the Legacy Wrapper Framework.

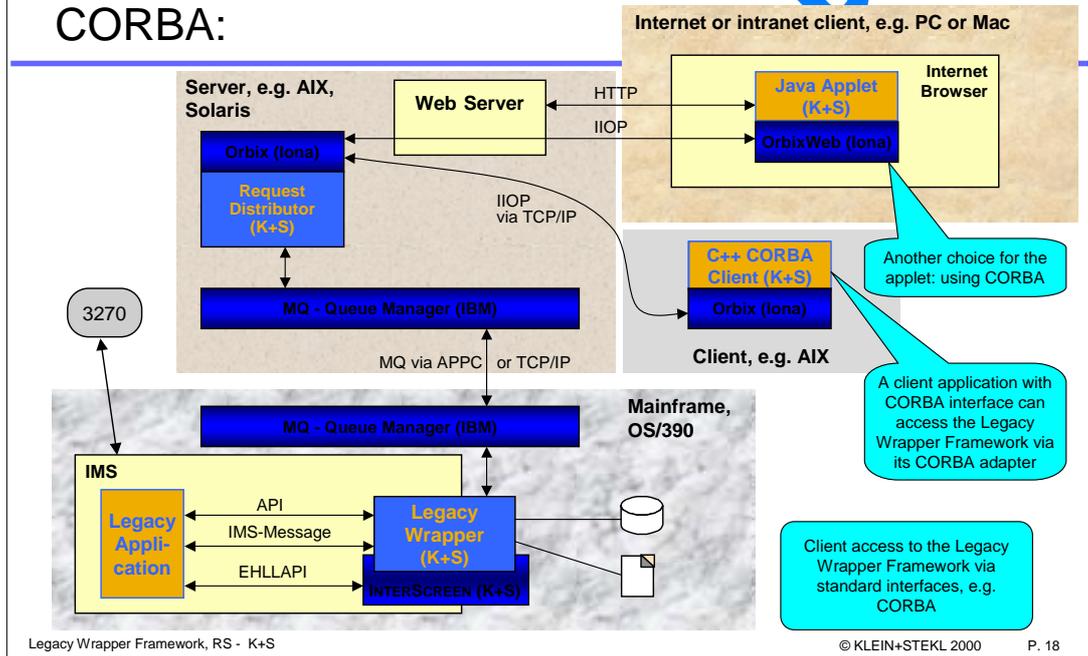
Using the product "MQ-Client for Java", the Java applet can insert request messages into the MQ queues on the web server's machine, whereas XML can be used as one possible message format. The Request Distributor reads the requests and hands them over to the Legacy Wrapper.

Of course one MQ-Queue Manager on the Request Distributor's machine is sufficient. However you have to use different queues for external communication with the clients and for the internal communication of the Legacy Wrapper Framework.

The Legacy Wrapper controls the legacy application, produces a result message and hands this message back to the Request Distributor. The Request Distributor sends the result message to the calling client.

Alternatively every client application with an MQSeries interface can gain access to the Legacy Wrapper Framework by putting messages into the request queue of the Request Distributor. The client in this example is a C++ program residing on an AIX machine.

How to connect a client via CORBA:

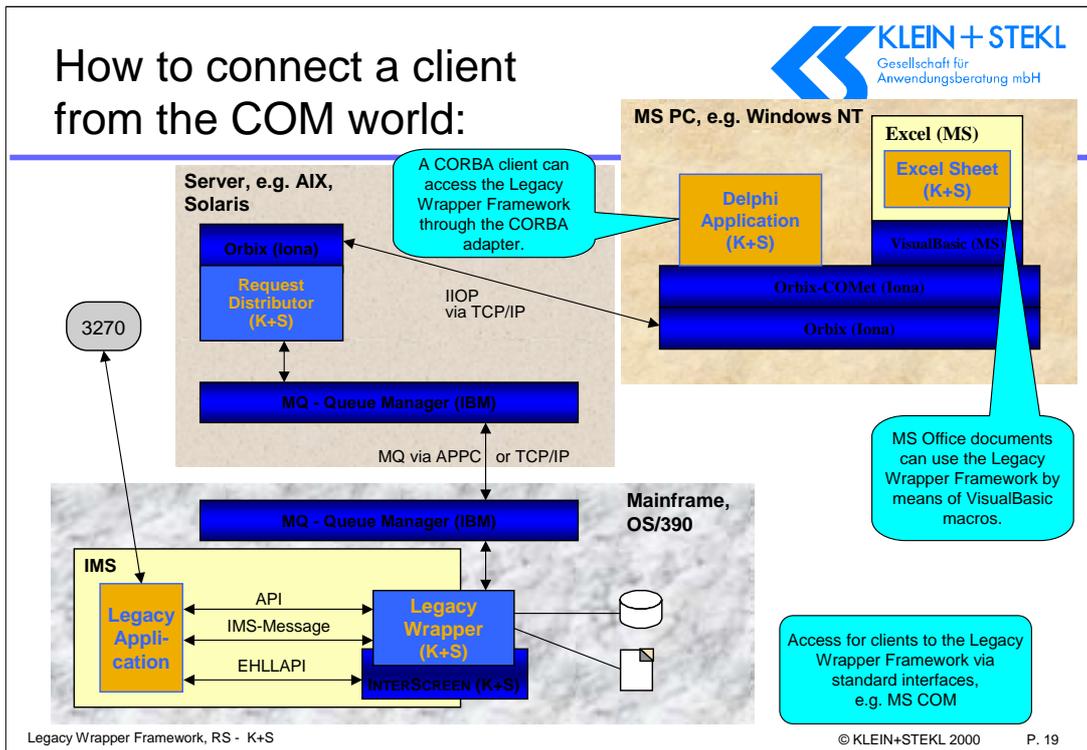


If you prefer using a CORBA interface for your clients, e.g. because you want to use SSL (Secure Socket Layer), the Request Distributor needs an IIOIP interface. We used the CORBA implementation Orbix by IONA.

A Java internet client linked with OrbixWeb sends its requests in the form of function calls to the Request Distributor. The communication between the Request Distributor and the legacy application is as described above.

Alternatively every client application with a CORBA interface can gain access to the Legacy Wrapper Framework by means of IIOIP function calls to the Request Distributor. The client in this example is again a C++ program residing on an AIX machine.

The example shows how easily the Legacy Wrapping Framework can be extended as soon as the infrastructure is set up and the business requests are defined.

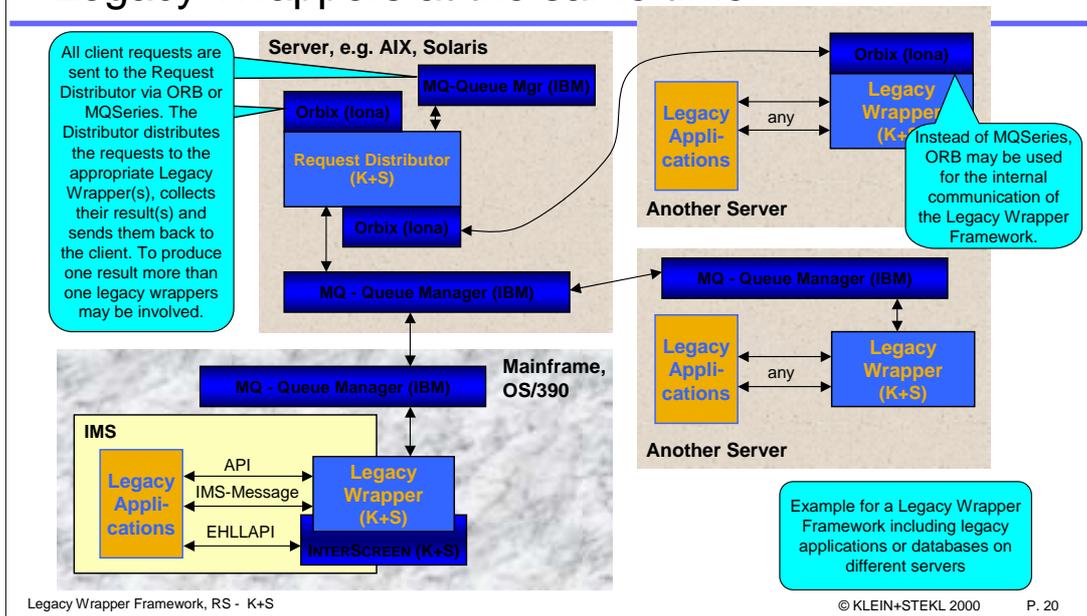


Using a COM bridge, it is easy to gain access to PC clients such as MS-EXCEL macros via the CORBA interface of the Legacy Wrapper Framework. We used the implementation of Iona COMet.

Since Delphi 5.0, CORBA has been integrated into the programming language. Therefore, you can run Delphi applications with CORBA access to the Legacy Wrapper Framework without using the COM bridge.

We have implemented this architecture as shown above. In spite of the great number of layers and products used, the **data transfer time** '1 ping' from an Excel sheet on a Windows NT PC to the legacy wrapper on the mainframe and back again amounts **only to 1/10 sec.**

How to offer services of different Legacy Wrappers at the same time:



In many cases the legacy applications and databases that are to be connected run on different systems. In this situation the best solution is to implement one Legacy Wrapper per machine or per legacy application. The Request Distributor then distributes the client requests to the appropriate Legacy Wrapper(s). It collects the answer(s) from the Legacy Wrapper(s) and sends a result message back to the calling client.

Thus business requests may require services from legacy applications on different systems: If only one Legacy Wrapper is necessary to process the current request, the client request is directly routed to this wrapper. Otherwise, if the services of more than one Legacy Wrapper have to be invoked, the Request Distributor splits the client request into smaller parts which are sent to the appropriate wrappers. Then all the results have to be put together by the Request Distributor to form the actual client result. The knowledge which Legacy Wrapper is to be invoked is encapsulated in the Request Distributor and transparent to the client.

To avoid the Request Distributor becoming a bottleneck, it should use multiple threads and/or process instances. Processing time is minimal because it acts only as a data and function turnstile.

Summary

- Lean organisation: avoidance of paper interface and mess of connections
- Effective reuse of existing functionalities with component technology: using 'best access' for legacy applications
- Client access via standard technologies (CORBA, messaging)
- Independance of technological constraints due to an open architecture, expandable by means of plug-ins
- Common business services encapsulating legacy details
- Legacy wrapping plus idea of business objects

 Productive solution integrating languages and platforms

If you have legacy applications (LA) that should not or can not easily be replaced, there are numerous advantages of using a Legacy Wrapper Framework (LWF) with business request exchange:

Avoiding paper interfaces with multiple data entry, an LWF helps to gain a lean organization and a less fault prone and much faster work flow. Existing functionalities and data can be reused on business level which leads to an independance of technological constraints and therefore to an open architecture.

As soon as the basic structures are implemented, they can easily and quickly be extended to new client interfaces as well as to further legacy applications by plug-ins.

Combining these ideas and advantages, we have shown you a productive solution that integrates programs written in languages from Java to COBOL and running on platforms from mainframe to Windows NT or the internet.

By the way:

If you are interested in

- further information on our concepts,
- implementing your own legacy wrapper,
- our product INTERSCREEN,
- ...,



feel free to get in touch with us! You will find us at:

KLEIN+STEKL GmbH
Heusteigstr.41
70180 Stuttgart / Germany

Tel.: +49 / 7 11 / 96 72-0
Fax: +49 / 7 11 / 96 72-1 30
Internet: <http://www.klst.com>

or: ruth@klst.com