JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2004**

Vol. 3, No. 2
Special issue: .NET: The Programmer's Perspective: ECOOP Workshop 2003

Design of Large .NET Applications: Best Practices

Harald Haller, sd&m AG, Munich, Germany

Abstract

In this paper we present experiences with the .NET Framework and Visual Studio.NET, which we won in two big projects. We describe what standard techniques and architectures can be applied to .NET technology. We give also hints for the practical use of the .NET framework in projects.

1 INTRODUCTION

We designed and implemented two systems:

- A database maintenance system for the internet risk assessor "MIRA" for the Munich Re. Up to 10 developers were included and the project had a size of 8 man years.
- The core application of the real estate investment company Real I.S. of the Bayerische Landesbank Group. The system was build by a team of up to 14 developers in 20 man years.

Both systems were designed as three-tier-applications and implemented in C#. Main features are database maintenance, document management, workflow support, and user management using Active Directory Server.

2 APPLICATION ARCHITECTURE WITH .NET

Technical architecture

Fig. 1 gives an overview over the technical architecture of both software systems. The three-tier architecture consists of thin clients with presentation logic, application server containing the business logic and a database server. This architecture enables scaling, increases reliability and simplifies software distribution.

Cite this article as follows: Harald Haller: "Design of large .NET applications: Best practices", in *Journal of Object Technology*, vol. 3, no. 2, Special issue: .NET: The Programmer's Perspective: ECOOP Workshop 2003, pp. 11-17. http://ww.jot.fm/issues/issue_2004_02/article1

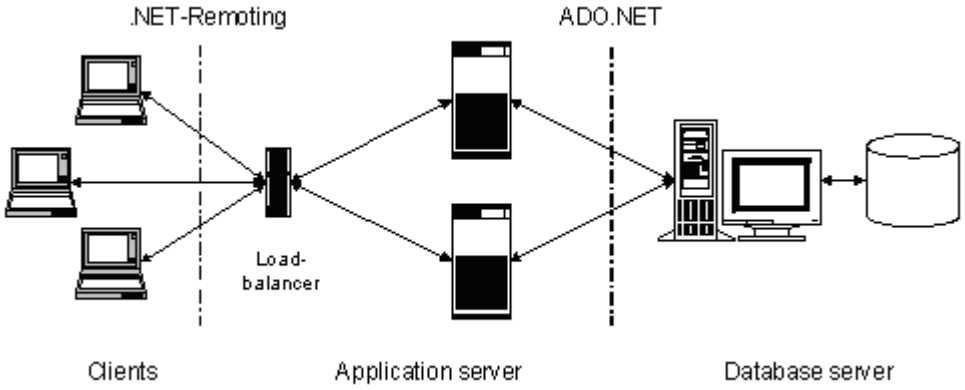


Fig. 1: Technical infrastructure

Architecture

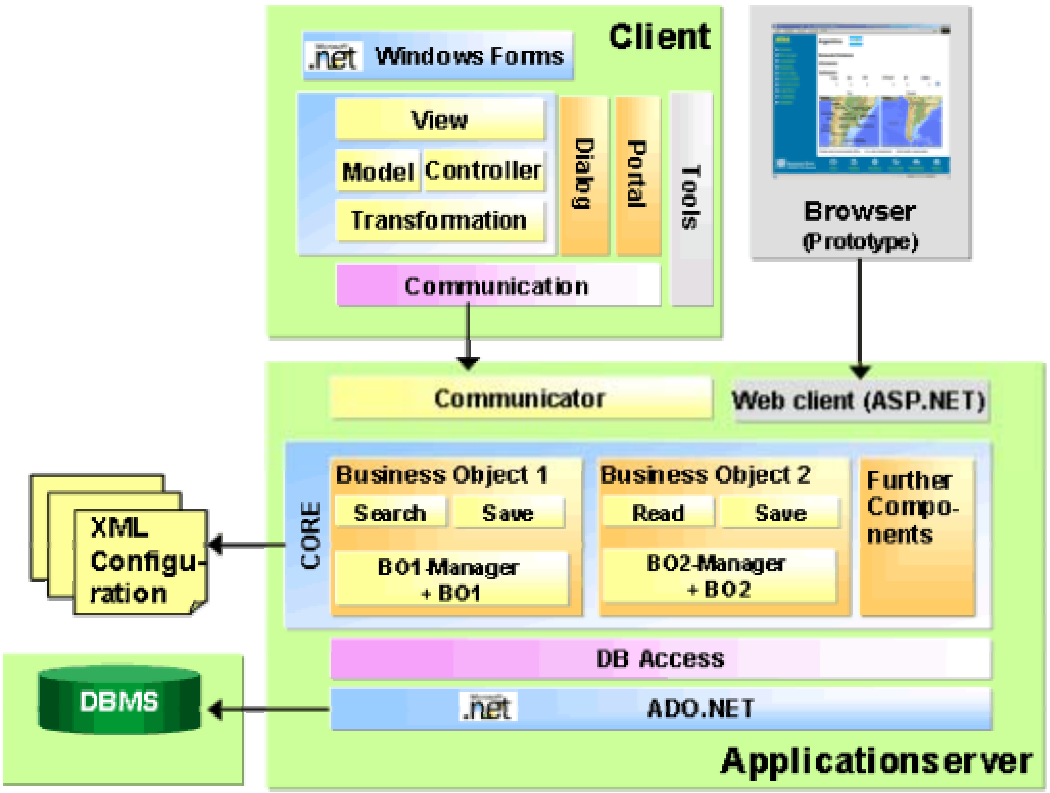


Fig. 2: Technical architecture

The components in fig. 2 have the following responsibilities:

Client

- Our GUI-Controls are based on `Windows.Forms`. Although these classes offer already extensive functionality, expansions were necessary, e.g. for correct output, the interaction between view and model and for copying objects. This was implemented via Wrapper classes. The view uses these Wrapper classes and is responsible for the visible part of the GUI.
- The model contains the data displayed on the surface.
- The component controller processes the user actions, contains the state machine, triggers server actions and opens sub dialogs.
- Transformation maps data between client and server data model.
- Communication is responsible for the data exchange with the server via .NET Remoting.

Server

- Communicator provides server services to the clients.
- Core contains the business logic.
- DB Access performs the access to the database via ADO.NET. Due to performance reasons the DataReader is used and not the DataSet, cp. section 3.1.
- Additionally a web client (as prototype) is implemented on basis of ASP.NET and uses the same core components.

Further additional utility components exist for logging, tracing, exception handling, configurations, security, and data types.

General observations

The features and functions of the .NET framework fit very well in this architecture. But some problems might occur (e.g. performance in DB Access and the transport of data), some components have to be extended (e.g. user controls), some components are missing. But altogether the .NET framework is an excellent basis for the implementation of the design illustrated in fig. 2.

3 EXPERIENCES

.NET Remoting

.NET Remoting is very well-suited for the client-server-communication and can be integrated easily. But the type of communication (binary versus XML/SOAP) and the structure of the data have significant influence on the response time:

In our examples the binary communication was 2-10 times faster than the communication via XML/SOAP.

Further the structure of the data has significant influence on the response time: This will be demonstrated in the following example:

We transfer the following data from server to client:

Country	Capital	Area	Number1	Number 2	Number 3
Germany	Berlin	Europe	8234	3575	634,52
China	Bejing	Asia	2345	243	265,34
...	...	...	...	...	...

There are different possibilities for data containers:

- Dataset:
  Data is read from the database, automatically stored in a DataSet and transported as DataSet to the client. This way is recommended by Microsoft in demos.
- Hashtable:
  The data is stored in a Hashtable. Each row of the table is stored as an object in the Hashtable and referenced by the number of the row.
- 2-dimensional Array:
  The data is stored in a matrix of the form String[][].

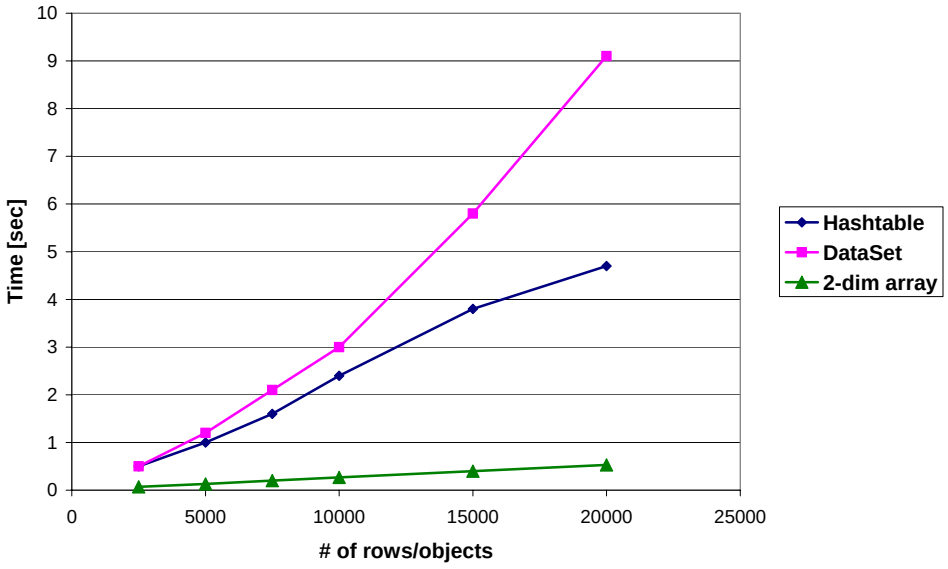
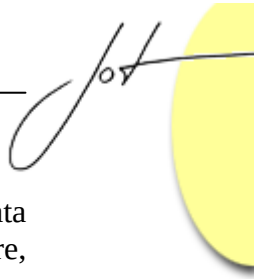


Fig. 3: Serialization

The time necessary for the serialization of the objects, the conversion of the object structure in the data stream, depends on the data container. The two-dimensional array leads to the fastest conversion and the `DataSet` to the slowest. This is not surprising since the `DataSet` contains additional meta information, which is not always necessary. Similar results can be obtained for the net traffic and for the deserialization.



The type of communication (binary versus XML/SOAP), the structure of the data and the objects have significant influence on the response time: The simpler the structure, the faster the communication.

Office-Integration via COM

Frequently .NET-applications have to integrate already existing individual software solutions or software products.

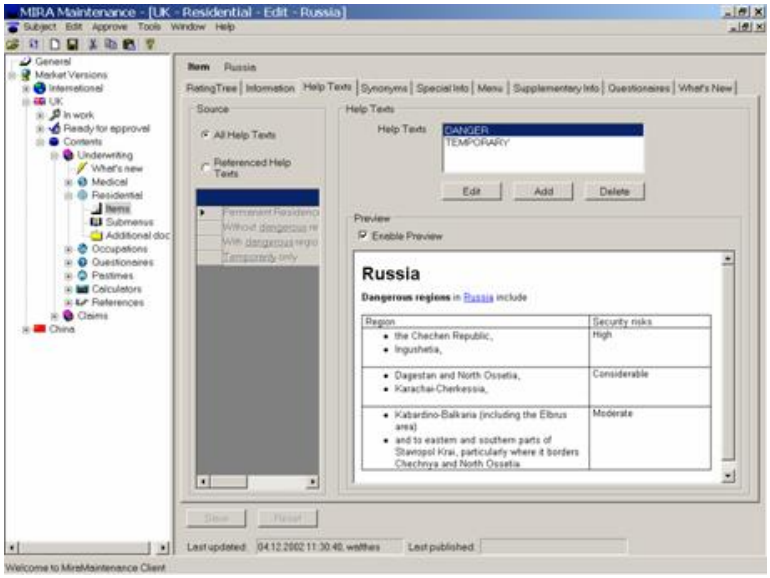


Fig. 4 - Internet Explorer as Plug-In

In our projects we integrated Word, Outlook, Internet Explorer, and Acrobat Reader within our .NET system. This means integration within the GUI and to use many different functions of these products directly by our system, e.g. Microsoft Word was started by our client so that the user could process documents. We reacted to certain actions of the user within Word by showing .NET-dialogues or performing actions in the background. Further the document structure was analyzed when storing the document and the document was automatically transformed to HTML. Hereby the consistency of contained links was tested and keywords for the search were extracted.

All this functions were implemented in .NET via COM in very short time and without any problems[1]. Even the response time was surprisingly good.

Internationalization

.NET promises an extensive support for internationalization of applications. This was a central request in one project with following concrete requirements:

[1] Today the integration of Word as Plug-In within Windows Forms is impossible and the plug-in via the Internet Explorer opening Word restricts Word functionality.

- Data maintained by the system appears in different languages including Russian, Chinese and Hebrew,
- language-dependent sorting rules and algorithms for the string comparison,
- country-specific formatting for numbers, date and time,
- different writing directions,
- several languages simultaneously within the application and even within singular dialogs.

In the namespace `System.Globalization` the .NET framework offers numerous functions, for example:

- location mechanism containing information about language, country, region, and calendar (`System.Globalization.CultureInfo`),
- localized treatment of date, time and numbers (`DateTimeFormatInfo`, `NumberFormatInfo` and `Calender`),
- language-dependent sorting and string comparison (`System.Globalization.SortKey`, `System.Globalization.CompareInfo`).

Resources for the location are available in the namespace `System.Resources` and can be obtained via `System.Resources.ResourceManager`. .NET provides UNICODE-support and conversions between fonts can be performed with the classes `System.IO.StreamReader` and `System.IO.StreamWriter`.

This extensive functionality is used in the project, but nevertheless project-specific expansions may be necessary for international projects:

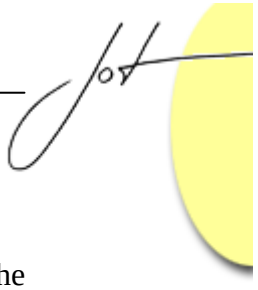
- In our project the assignment of the countries, languages and cultures was not identical with the standard assignment: For example Greece doesn't have any own Greek version at present, but uses an English version.
- Language and culture were not application-specific but dialogue-specific, partially even field-specific. A user could see data in several languages at once. Consequently, the language had to be set explicitly in the Windows.Forms-components.
- Some controls, for example the `DatePicker`, remained in the language of the operating system. Therefore workarounds were necessary.

According to the internationalization the .NET framework relieves much work, nevertheless project-specific adaptations may be necessary.

Development environment

The development environment Visual Studio.NET including Visual Source Safe and Rational XDE is designed for small projects. In order to implement large projects in .NET additional concepts for the utilization of the IDE and additional tools for the global build are necessary. This provides an effective and reliable development process, if the bugs and workarounds are known.

Third-party tools like NUnit as test framework, NAnt as Build-Tool, and NDoc for the generation of online documentation are useful add-ons.



Going Live

The installation of the application could be performed within short time since the assemblies (DLLs) only had to be copied. Our application runs without performance problems with response times usually under 1 second. Respecting the stability of the system, there are no problems. We could not observe any memory leaks, however the server services are restarted daily.

4 SUMMARY

In these two projects we demonstrated that .NET is already a good basis for application development. We do not have any problems with performance and stability according to load and availability. Nevertheless, expansions of the .NET framework and the development of an infrastructure with basis components are useful especially for bigger projects. If the .NET framework shall be used in a concrete project, we recommend to design a reference architecture early and to test it. Furthermore, time for training and workarounds might be necessary. On the other hand, the implementation time is comparable with the development with J2EE as soon as the bugs of the first versions are removed and more stable components of third manufacturers are available. What is more .NET has advantages compared to J2EE in the development of Web applications, Web services and in the integration of Microsoft products.

About the author

Harald Haller is a project manager at sd&m AG - software design & management. Main focuses of his work are the architecture of distributed applications, the design of Java and .NET applications and project management. Currently he is responsible for a .NET-Project at sd&m. He can be reached at harald.haller@sdm.de.