Proceedings of the 2004 Informing Science and IT Education Joint Conference

Universal Help Desk Database Access Tool Using C# and WEB Services on the .NET Platform

Samuel Sambasivam
Azusa Pacific University, Azusa, CA, USA

Aleksandar Bulajic
Maegrsk Data A/S, Denmark

ssambasivam@apu.edu

lanb@45.dk

Abstract

There is a continuous need for a tool that can access limited sets of data through remote program interface. The current technology development offers Internet and Web based application as the obvious choice. Web Help Desk application which accesses a data source and displays contents to the users is usually developed using Internet, together with target application.

This paper offers a different approach. By using SQL database meta-data a generic Web Help Desk application can be created to access different database systems, independent of the implemented technology and data model changes. By using XML Web Service as the layer where all access to the remote data sources are located, a system can be created that can access to data source across networks and operating systems boundaries. If any part of a data model is changed, application should still be able to access a data source.

Keywords: C#, Help Desk, JAVA, .NET, Relational Database, SQL, Web Service, XML

Introduction

In the process of the software development and maintenance, there is continuous need for monitoring database changes and investigating data changes and data consistency. Discovered errors are usually reported in the separate report and corrected by a development/maintenance team.

The next section "Background of the Problem" discusses the problem background in more details. The sections from "Relational Database System" to "Alternative Development Tools" present an overview of the currently available technologies and methodologies for project implementation.

The "Architecture and Design" section presents project architecture, design and database model overview.

Background of the Problem

In the development environment, tools delivered as a part of the database software and installed on each of the developer's machines usually access to the database data that are located somewhere on the local area network (LAN). Accessing data in the development environment is not an issue. Different support teams make necessary LAN and tools settings and help the development team to accomplish the tasks.

In the production environment, end user usually reports data inconsistency through Help Desk. Help Desk reports errors to the project contacts or development/maintenance team and the process of the error correction starts. The first addressed problem and one of the most important parts of the error correction process is collection of the relevant information and error recreation in the controlled environment.

For this purpose, more and more applications are using Web interface, browser, for monitoring limited set of the most important database tables. Using ASP or JSP technology within advance known monitored tables, columns and search conditions usually develops these applications as a part of the current project. If the data model is changed, then these applications shall be changed too; coded, built, deployed and tested again and again. This process is a waste of time and money and in some cases when error occurs is usually too late for a fast answer to the changed requirement.

Using another database access tool such as Enterprise Managers, TOAD, ORACLE DBA Studio, DB2 Command Centre, PL/SQL or online SQL queries are limited to the current LAN configuration. Dislocated databases, different networks, different customers and security reasons seriously limit the use of these tools. In the case when application data and development/maintenance team are located in the different companies or are divided by networks boundaries, data is extracted in one place and sent to another location by using mail or another data transport layer. If it is not sufficient, then it requests another data extraction and data sending process is repeated until cause of error is discovered and corrected. This process is time consuming and error prone too. And it can cause loss of profits in the case of the serious application errors.

Another problem is that in the production environment, access to the data is restricted. There are several different reasons. One of the important reasons is the protection of the sensitive business information, even in the case when the Help Desk or development/maintenance team is part of the same company. Statistics shows that 80 percent of the security breaches are done from inside of the company. The other reasons may be intentional and non-intentional production data corruption or deletion, dislocation of the databases and restriction to the data access across network boundaries.

Accessing the databases in the dislocated networks, by using standard database tools delivered by database vendors or third parties, could be a problem even in the Internet world of the global network communications, because these databases are usually located behind the firewalls and proxy servers. To be able to access this data there has to be additional ports open in the firewalls and provided authorization for accessing internal networks, which can cause serious security breaches and increase maintenance expenses. Those tools usually need database administrator connection privileges and are very rich with database administration commands too. It can be used to exploit data in another project located in the same database server too.

Development of the dynamically configurable User Interface to the SQL database tables independent of the database changes could solve the first problem and provide a tool that can be developed once and used in all projects. Development of the tool that it is using Internet connection and http protocol to access database could solve data access problem in the case of the dislocated data location.

Restricting functionality of this tool to the read-only access, limiting access to only the current project database objects and recording whole process of data analyses could reduce chance of data corruption, deletion and also helps to discover and identify security breaches.

Relational Database System

Since Dr. E. J. Codd in 1970 presented his ideas about relational database model in the article "A Relational Model of Data for Large Shared Data Banks" (Codd, 1970), there has been a large number of books written on the same subject. In the meantime, relational database systems became a major database, industrial standard used in a large number of applications. In the Computer World, in October 1985, Codd published two articles "Is Your DBMS Really Relational?" and "Does Your DBMS Run by the Rules?" that present a list of the 12 rules for evaluating a relational system. According to these rules, there is no fully relational system available yet ("Codd's 12 rules," 2003). In 1974-75, Raymond Boyce and Don Chamberlen, working in the IBM System R project group (McJones, 1997, p. 15), have been inspired by Codd's work (Codd, 1970). They created a new language called Structured English Query Language (SEQUEL), for extracting data from the relational data system (McJones, 1997). Later on, this has been abbreviated to SQL ("On Relations," 2003). In 1979, Larry Ellison's company called Oracle has created the first commercial product based on the relational model and using SQL. Later on, IBM has developed SQL/DS relational system, which later became known as DB2 database (McJones, 1997). The first version of the Microsoft SQL Server for OS/2 platform has been developed by Microsoft and Sybase together and released in 1988. In 1990, Microsoft decided to develop SQL Server database for Windows NT only. The SQL Server 4.2 for Windows NT 3.1 was released in 1994. Microsoft ended partnership with Sybase in 1994. Microsoft released SQL Server 6.0 in 1995. After releasing version 6.5, in 1997, Microsoft released version 7.0 and in 2000, version 2000 (Spenik & Sledge, 2003). Microsoft database is delivered with user-friendly GUI administration interface called Enterprise Manager. The fourth Codd's rule has been described as "Active, online relational catalogue -- The description of the database and its contents are represented at the logical level as tables and can therefore be queried using the database language" ("Codd's 12 rules", 2003).

In the current relational database systems, this rule is actually described as a set of database system tables. These tables contain information about relational objects, keys, constraints and can be queried by using ordinary SQL statements. In the DB2 database, the system tables SYSTABLES, SYSCOLUMNS and SYSDATATYPES can be used to collect information from the target database about user created tables, columns and column types. In the Microsoft SQL server, these tables are called SYSOBJECTS, SYSCOLUMNS and SYSTYPES. These tables contain enough information to create Universal Help Desk Database Access Tool. It should be possible to read any relational database model that implements Codd's rule 4. However, there can be differences in the table names, table columns and data types that shall be handled separately for each particular case. If the program code contains embedded SQL statements, system tables could be used to test SQL statements correctness. Existence of the table name, column names, column types, stored procedures and syntax of the SQL statements can be checked during program compilation and avoid runtime errors in the case when error is caused by calling non-existing objects, name is misspelled or the wrong type is used.

The SQL becomes an ANSI and ISO standard (Howe, 1993). The first standard was made in 1986. The major standard revision has been done in 1989 known as SQL-89, in 1992 known as SQL2 or SQL-92 and 1999, known as SQL3 or SQL-99. Even though most of the authors will state that embedded SQL is a cross-platform language (Spenik & Sledge, 2003); our experience is that cross-platform compatibility is actually achieved by pre-compiler. The pre-compiler deals with differences between platforms and make compiled code platform compatible (FFE Software 1996). Another problem related to the SQL standardization is extension added to language, for example by Microsoft, called Transact SQL or by Oracle, called PL/SQL. Implementation of the stored procedures also differs from one database system to another. These differences could cause portability problems.

XML Web Services

*"Definition: A **Web Service** is a software application identified by a URL [IETF RFC 2396], whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols." (W3C 2002a).* In the case of the Intranet applications running on the trusted networks, it is not a problem to make necessary system setting and allowed access to the remote database. Access to the underlying hardware and software infrastructure is easily controlled and usually owned by the same company.

This problem is not simple in the Internet application. Internet applications are protected from the outside world by firewalls and proxy servers that actually act as firewalls too. Multiple companies own the hardware and software infrastructure. A network setting in the Internet environment becomes more complex because of more firewalls, proxy servers, and networks owned by different companies that can be direct market competitors. In the process of creating hardware and software infrastructure and implementing necessary setting, the communication overhead is large and expensive too. Application to application communication and software integration is limited if it is not implemented as standard for message exchange. These are some of the reasons why the large consortium of the major IT companies agreed about common standard for communications in the Internet environment. The standard called Web Services or XML Web Services is actually a collection of related technologies.

The XML Web Services are the current answer for the requirement for Application-to-Application communication, wide software integration, existing code and application reusing. This requirement is not a new one. *"The World Wide Web is more and more used for application to application communication. The programmatic interfaces made available are referred to as Web services*." (W3C, 2002b). The XML WEB Services are reusable pieces of software that could be accessed by using network and API. The communication is accomplished through the set of sent and received messages. These messages are based on the XML as pure text files and could be easily transferred across different platforms by using standard Internet protocols (HTTP, TCP/IP). The XML Web Services are the collection of the related technologies. The Web Services implementation depends directly of the following technologies: XML, SOAP, WSDL, and UDDI.

These technologies will be described and used in the project implementation. It will be presented through real examples used in the project design. The Web Services are standardized through World Wide Web Consortium (W3C) and wide accepted from the world leading IT companies such as Microsoft, IBM, SUN, Oracle, and BEA Systems. That is an important precondition for widely technology deployment and future success. These companies developed a set of tools, based on the common standard for documents, data and messages handling. The software developers are not depending on a single vendor. There are different vendors that implement standard solution even in the not compatible technologies as for example .NET Framework and J2EE. The resources are free and accessible through the Internet (MSDN Microsoft 2003a; Java, 2003). The Web Services, as other emerging technologies in the past, creates a lot of expectation in the IT professional community. But from experiences we can learn that new standards do not solve IT problems by default. The IT problems origin is much deeper than technology limitation. There are a lot of different factors as for example competition, fight for markets share, market dominance, politics, etc. These factors and many other similar sometimes can be a power that speed up development, but sometimes can waste a lot of resources and slow down development. Different, not compatible, products and different standards make choice and development more complex too.

There is no doubt that Web Services will be an important part of the future application development, especially in the case of the software interoperability and integration. (Chappell, 2002, p.

74) But to experience the limitations of the new technology it is necessary first to implement it and use it for some time. There is also consideration related to the Internet network bandwidth. The Internet network is still a bottleneck. Could it get even worse in the case that large number of the application starts to communicate with each other by using http protocol? In the current situation it is the limitation factor. For wide Web Services deployment, it is necessary to provide more bandwidth and probably some new protocols too. (WS-I 2003a). Though this technology is still not mature and there are still important issues as for example security issues (WS-I, 2003a; W3C, 2003c; W3C, 2003d), it could be already used in the development of the application where the requirements are: "Programmatic access to application accessed via Internet, Application to Application communication (A2A) and Business to Business (B2B) communication" (Chappell, 2002, p. 46).

C# and ASP.NET

Microsoft .NET platform is announced as the largest Microsoft investment in the future. The .NET brand is actually applied to several different technologies (Chappell, 2002, p. 2). This platform offers framework and sets of the different technologies for wide Enterprise application integration, Web development and Web Services implementation. *"The* .NET platform *is one over which Web-based applications can be distributed to a great variety of devices (even cell phones) and to desktop computers. The platform offers a new software-development model that allows applications created in disparate programming languages to communicate with each other."* (Deitel et al., 2002, p. 9)

In the .NET platform, Microsoft has created Common Language Runtime (CLR) to support multiple languages. All languages that are based on the implementation of the Common Type System (CTS) and CLR are translated in the same Microsoft Intermediate Language (MSIL). MSIL removes differences used by C# or Visual Basic.NET or J# or any other language. These are translated in the same MSIL code. There are significant advantages in using this approach, for example portability, type safety verification when it is loaded in memory, better security and reliability (Chappell, 2002, p. 88). Disadvantage is slower code; even MSIL is always compiled before execution (Chappell, 2002, p. 89). The C#, developed at Microsoft by a team led by Anders Hejlsberg and Scott Wiltamuth (Deitel et al., 2002, p. 9) is one of the most important Microsoft .Net platform for CLR based language. This language is based on the C++ syntax, but the language structure and the large library called Class Library are new. It is explicitly created as object-oriented language (Chappell, 2002, p. 120). The error prone C++ language features, such as pointers and multiple inheritances, are removed. Automatic garbage collection is introduced to prevent memory-leak errors. These are also part of the .NET Framework. The .NET Framework delivers command line C# compiler. Microsoft Visual Studio.Net includes C# language too.

The C# is a new programming language and one of the most important Common Runtime Language (CLR) based languages (Chappell, 2002, p. 119). "*C# also enables a new degree of language interoperability: Software components from different languages can interact as never before. Developers can package even old software to work with new C# programs. In addition, C# applications can interact via the Internet, using industry standards such as the Simple Object Access Protocol (SOAP) and XML, which we discuss in Chapter 18, Extensible Markup Language (XML). The programming advances embodied in .NET and C# will lead to a new style of programming, in which applications are created from building blocks available over the Internet.*" (Deitel et al. 2002, p. 10)

According to the Microsoft sources and other authors (Chappell, 2002; Deitel, et al., 2002), C# is a suitable language for implementing a proposed project. It is delivered as a part of the powerful IDE (Visual Studio.Net), for Rapid Application Development (RAD). This environment contains different tools for application development, debugging, testing and deployment. The ASP.NET is the next generation of the Active Server Pages (ASP), Microsoft technology for creating and managing dynamic Web pages. As part of the .NET platform, using any .Net platform language can develop ASP.NET (Microsoft Visual Studio.NET 2002a). Differences between ASP and ASP.NET are significant. In the ASP.NET object-oriented languages such as C# and Visual Basic.Net replace scripting languages. The code is separated from the presentation in a separate file by using code-behind technique. It permits more presentation files to share the same code-behind file. The ASP.NET also introduces Web Forms. These forms can be built in the Visual Studio.Net by using visual editor and drag and drop components in the form. The ASP.NET ensures that these components are mapped to the corresponding HTML components (Chappell, 2002, p. 270). The ASP.NET pages are event-driven. Each ASP.NET page is represented by a class (Chappell, 2002, p. 270). This class is compiled into the MSIL code and written to the disk (Chappell, 2002, p. 269). Every time when the page is requested this code is used. When a certain event occurs, the ASP.NET calls corresponding method from the code-behind file (Chappell, 2002, p. 275, Deitel et al., 2002, p. 949). The session state management is significantly improved. In the ASP.NET session can be stored in the same process as application, in another process that runs locally or on another machine, or the SQL server database. These characteristics and common IDE such as Visual Studio.Net makes ASP.NET and C# suitable tools for development of the proposed project.

"The advantage of using Visual Studio is that it provides tools that make application development much faster, easier, and more reliable. These tools include: Visual designers for Web pages with drag-and-drop controls and code (HTML) views with syntax checking, Code-aware editors that include statement completion, syntax checking, and other IntelliSense features, Integrated compilation and debugging, Project management facilities for creating and managing application files, including deployment to local or remote servers."(Microsoft Visual Studio.Net, 2002a).

Architecture and Design Pattern Recommendation

The home page of Microsoft MSDN (MSDN Microsoft 2003a) provides links to the series of articles, white papers and guidelines for Application Architecture and Design recommendations. A large number of people have been included in writing and reviewing these materials (Microsoft 2002a). Microsoft primary focus in the .Net Platform is wide Enterprise Integration and distributed application development. The Web Services are playing the main role in most of the scenarios. XML Web Services ability to expose interface to any kind of the client makes this technology suitable for middle tier components. The browser-based clients, rich Windows application client, component, another Web Service or another distributed application running on the different platforms can access XML Web services component (Microsoft Visual Studio.NET, 2002b). The articles and white papers offer detail discussion about architecture and design patterns. Online documents are available for high level architecture and design pattern and practices (Microsoft 2002a) and links to more detailed discussion about recommended solutions. Distributed application architecture is a set of interacting components. Each component encapsulates functionality and exposes public interface for interaction with another components. As long as components provide required functionality, the internal component implementation is not important. The important part is what data needs to be sent to the component and what response will be received from the component (Microsoft, 2002a: 5). In the case of the XML Web Services, these components are loosely coupled. "By identifying the generic kinds of components that exist in most solutions, you can construct a meaningful map of an application or service, and then use this map as a blueprint for your design." (Microsoft, 2002a, p. 7). The components that provide similar functionality can be grouped into layers (Microsoft, 2002a, p. 5). By identifying common application layers can be created reusable architecture pattern. *"By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than rewriting the entire application over, if they decide to change technologies or scale up."* (Webopedia, 2003)

Today it is widely accepted and meaningful to use expressions as 2-tier, 3-tier or n-tier application design. It is meaningful to call these tiers Presentation Layer, Web Layer, Business Layer or Data layer. These layers make up logical application decomposition. The physical implementation can be different, so the single computer can contain one or more layers or all of them or each layer could be implemented on the different computer even with different operating systems. Applications and services can also be built by using the same components (Microsoft, 2002a, p. 5). The used components inside of the service could have different role and responsibility. The services are usually communicating by exchanging messages. These communications can be synchronous or asynchronous. The asynchronous communication is the preferable one, especially when using store-and-forward mechanism, because they allow a more loosely coupled approach (Microsoft, 2002a, p. 109). Asynchronous design has to deal with special considerations such as message correlation, optimistic data concurrency management, business process compensation and external service unavailability (Microsoft, 2002a, p. 5). Mackenzie (2001) discusses different approach to asynchronous workflow management and benefits of asynchronous processing. Microsoft Pattern & Practice (Microsoft, 2002a, p. 110) provides a list of design consideration and disadvantages in the case of the asynchronous communication such as: Deterministic outcome, Message correlation, Message delay, Transaction flow, Repeated messages, and Message sequences. We have used the layered architecture for our work. The number of layers, architecture and design details are discussed in Architecture and Software Design section.

Another important part of the software development process is service or component internal design and implementation. Design Patterns (Gamma, Helm, Johnson, & Vlissides, 1995) provide a list of the common patterns used in the object-oriented software design process. These are another level of the common solutions identification. The primary goal is to reuse design patterns and good practice in the new developed components or application. Another important reason is also to improve common application design understanding and reducing communication overhead by reusing the same generic names for implemented solution. For example, it is much easier to understand words such as Single Pattern, Factory Method, Mediator Pattern, and Facade Pattern than description of the actual implementation. Design Patterns book (Gamma et al., 1995) describes 23 design patterns divided into three types (Cooper, 2000): Namely, Creational patterns, Structural patterns, and Behavioural patterns. Cooper (2000) provides a tutorial with examples programmed in Java in his book. *"Design patterns are a common way to organize objects in your programs to make those programs easier to write and modify.*" (Cooper, 2000, p. xv). Another important design issue is separating presentation, business logic and data access code. It is another common pattern that originates from Smalltalk (Cooper, 2000; Esposito, 2000) called Model-View-Controller (MVC). By separating view from business logic and data access, application is able to easily present the same data in a different form. Microsoft removes the controller and replaces word model with document (Esposito, 2000). It becomes Microsoft document/view model (Esposito 2000), assuming that data is always rendered for Windows. In the .NET Framework, code-behind technique is used to separate code and Web Forms in the .aspx file. The code that resides in the separate file, usually dll, is called in the case when the Web Forms are displayed. The MVC is usually implemented by creating one or more View classes that presents data in different forms. In the case of the XML documents, XSL is used to create different data views. Design patterns improve program design and make it easy to implement changes. Drawback of using Design Patterns is the reduction in application performance. Grady Booch, Ivar Jacobsen, and James Rumbaugh have developed the Unified Modeling Language (UML) by combining their own methods for software modelling in one common method and notation. Since the Object Management Group (OMG) ratification of the UML, in 1997, it becomes a standard widely accepted tool for Object Oriented Analyze and Design (OOAD). We have used the UML notation for software system modelling and application design presentation for our work. The UML creators Grady Booch, Ivar Jacobsen and James Rumbaugh have written in the foreword (Fowler & Scott, 2000, p. xiii):

"*When we began to craft the Unified Modeling Language, we hoped that we could produce a standard means of expressing design that would not only reflect the best practices of industry, but would also help demystify the process of software system modeling. We believed that the availability of a standard modeling language would encourage more developers to model their software systems before building them. The rapid and widespread adoption of UML demonstrates that the benefits of modeling are indeed well known to the developer community.*" (Fowler & Scott, 2000, p. xiii)

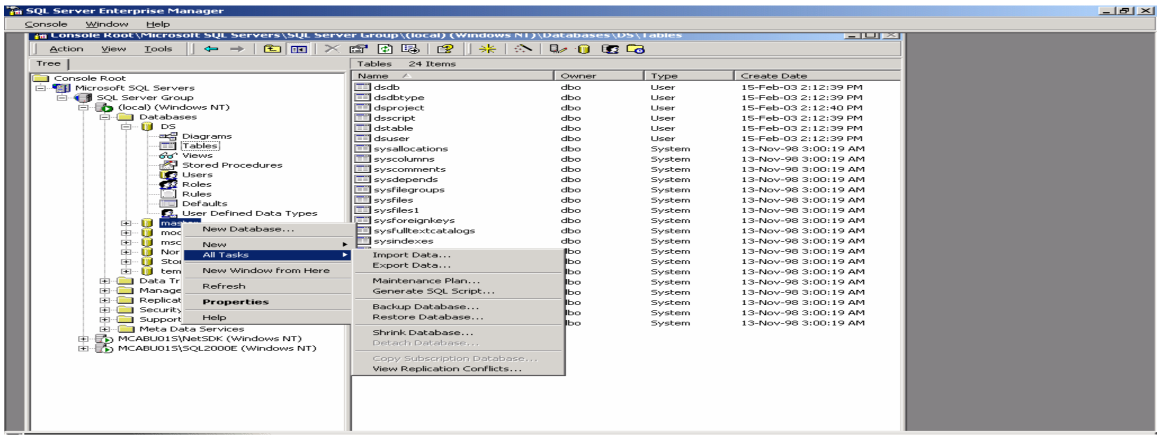
The UML has been used in the computer based software development process tools. One of them is Rational Unified Process (RUP) developed by Rational (Rational, 2003b). The Use Cases are an important part of the UML and the process of the software system development too. The Use Cases have been originally invented by Ivar Jacobsen in the late 1960s and introduced to the object-oriented community in the late 1980s and have been filling a significant gap in the software requirements collection and management process (Cockburn, 2001). In this project the Use Cases are used for describing project requirements such as business requirements and also for project testing. In the case of the small and middle size projects there is the Extreme Programming software development approach ("Extreme Programming," 2003). This approach is best known by recommendation that the source coding should first start by test case development and then actually source code. This method is recommended in the case of the "risky projects with dynamic requirements" ("Extreme Programming," 2003).

Existing Tools for Accessing SQL Database

A product called User Friendly Interface (UFI) was created to administer Oracle environment. The name has been changed to SQL *Plus after Oracle version 4. The SQL *Plus is command line character interface to the database. The end user types SQL statements interactively in SQL * PLUS. Oracle 7 and 8 introduce new Graphical User Interface (GUI) for database administration called Enterprise Manager (Advanced Information System, 1996, p. 290). The SQL *Plus is still a part of the standard Oracle database installation. IBM has created the first portable programming interface to the SQL database as part of the DB2 implementation. It was actually semi-portable Embedded SQL. The portability has been provided by pre-compiling source code and embedded SQL statements (FFE Software, 1996). Embedded SQL has become ANSI SQL standard in 1986. In 1990, the SQL Access Group (SAG) defined a portable API for SQL. This became a standard for SQL APIs and large number of the drivers for different database systems has been developed. Microsoft uses SAG specification as basis for Microsoft ODBC and later on adds access to the system catalog (FFE Software, 1996).

"*ODBC (Open Database Connectivity) is an industry standard programming interface that enables applications to access a variety of database management systems residing on many different platforms. ODBC provides a large degree of database independence through a standard SQL syntax, which can be translated by database-specific drivers to the native SQL of the DBMS.*" (Advanced Information System, 1996, p. 1312).

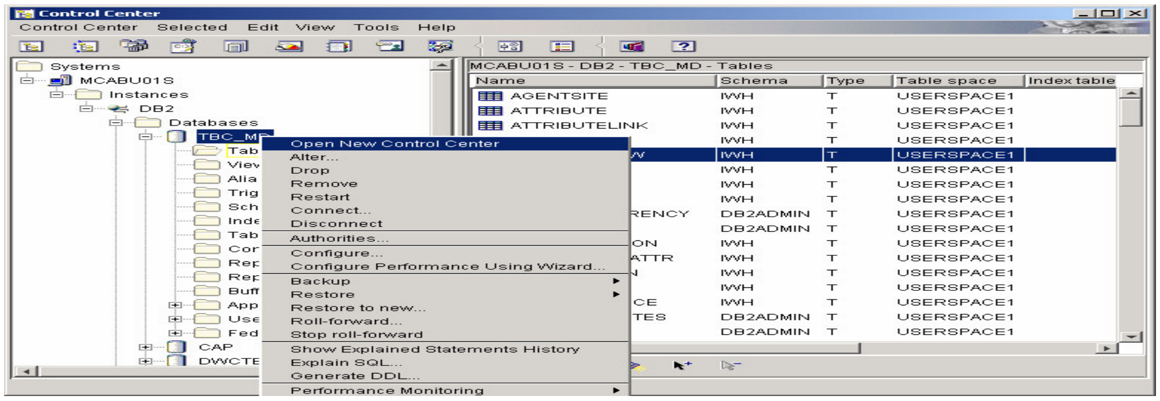
ODBC is designed to be independent of the particular database implementation. By using embedded SQL, through ODBC Application Programming Interface (API), it is possible to access different databases and hide complexity from the application. ODBC is made from different components: standard API, driver manager component (responsible to use right database driver), and other components (for processing SQL statements and returning results to the application). ODBC is able to access different databases located on different platforms. In the client/server environment and dislocated database systems, ODBC includes also any network software necessary for communication with remote database system (Advanced Information System, 1996, p. 1312). Network communication software is provided by database vendors and is specific for each database system. ODBC is actually standard middleware responsible for communication between application and target database. For remote database access, ODBC is using TCP/IP network protocol. Database administration through Graphical User Interface is today a standard in all major database products. It is called Enterprise Manager, DBA Studio, Command Center, etc. MS SQL Enterprise Manager is a user-friendly, easy to use interface to the MS SQL Server database. It is a part of the server installation and it is not available on the client's computers. This interface helps to accomplish any administration tasks easily. By using this tool, it is possible also to access database tables and browse through table contents. Access and command execution are simplified through a set of menus and submenus, where by simply clicking on a menu item or a form displayed where necessary, parameters can be inserted or just selected.



Name	Owner	Type	Create Date
dsdb	dbo	User	15-Feb-03 2:12:39 PM
dsdbtype	dbo	User	15-Feb-03 2:12:39 PM
dsproject	dbo	User	15-Feb-03 2:12:40 PM
dsscript	dbo	User	15-Feb-03 2:12:39 PM
dstable	dbo	User	15-Feb-03 2:12:39 PM
dsuser	dbo	User	15-Feb-03 2:12:39 PM
sysallocations	dbo	System	13-Nov-98 3:00:19 AM
syscolumns	dbo	System	13-Nov-98 3:00:19 AM
syscomments	dbo	System	13-Nov-98 3:00:19 AM
sysdepends	dbo	System	13-Nov-98 3:00:19 AM
sysfilegroups	dbo	System	13-Nov-98 3:00:19 AM
sysfiles	dbo	System	13-Nov-98 3:00:19 AM
sysfiles1	dbo	System	13-Nov-98 3:00:19 AM
sysforeignkeys	dbo	System	13-Nov-98 3:00:19 AM
sysfulltextcatalogs	dbo	System	13-Nov-98 3:00:19 AM
sysindexes	dbo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM
	bo	System	13-Nov-98 3:00:19 AM

Figure 1. MS SQL Enterprise Manager

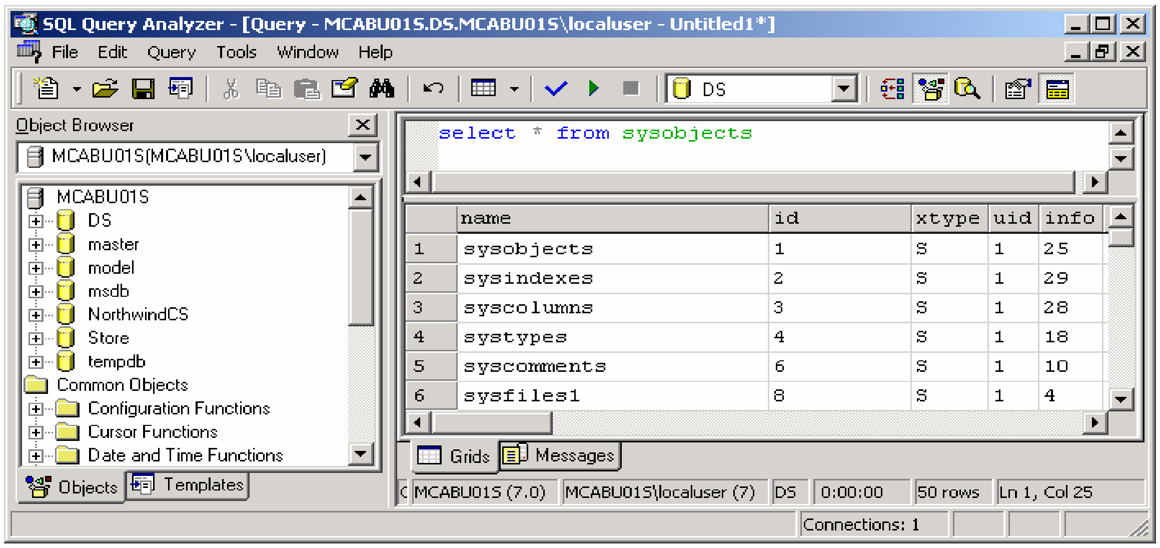
Figure 1 illustrates typical Enterprise Manager Screen. In the left part of the screen there are databases and database objects such as Tables, Views, Stored Procedures, etc. In the right part of the screen there is a list of these objects, sorted by name and additional info about owner, creation date and time. Menus and submenus contain different administrative functions. Even this interface to the database is very rich with functionality and easy to use, in the case when Help Desk Team uses it, there can be a number of reasons why it is not appropriate: It requests database administration privileges, it is too complex for average users, it also exposes other databases and sensitive data to the user, it can intentionally or non intentionally delete the whole database, it requires database administrator privilege, it does not keep track of who accesses the database tables. Another similar tool is IBM's DB2 Command Center.



Name	Schema	Type	Table space	Index table
AGENTSITE	IWH	T	USERSPACE1	
ATTRIBUTE	IWH	T	USERSPACE1	
ATTRIBUTELINK	IWH	T	USERSPACE1	
	IWH	T	USERSPACE1	
W	IWH	T	USERSPACE1	
	IWH	T	USERSPACE1	
	IWH	T	USERSPACE1	
	IWH	T	USERSPACE1	
RENCY	DB2ADMIN	T	USERSPACE1	
	DB2ADMIN	T	USERSPACE1	
ON	IWH	T	USERSPACE1	
ATTR	IWH	T	USERSPACE1	
N	IWH	T	USERSPACE1	
	IWH	T	USERSPACE1	
CE	IWH	T	USERSPACE1	
TES	DB2ADMIN	T	USERSPACE1	
	DB2ADMIN	T	USERSPACE1	

Figure 2. IBM DB2 Control Center

Similarity is obvious, and the reason why using this tool may not be appropriate for Help Desk database access. Oracle database tools called Enterprise Manager and DBA Studio and Microsoft SQL Server Enterprise Manager look almost the same. There are also client tools for accessing a database remotely. Such as Oracle's SQL *Plus and Microsoft's Query Analyzer. SQL *Plus is an ad-hoc, character based, command line tool. The SQL statements are typed at the command line and executed one by one against database. This tool has a number of limitations in typing, editing, and presenting results. Microsoft Query Analyzer provides GUI interface and different screens for typing SQL statements and presenting results.



```
select * from sysobjects
```

	name	id	xtype	uid	info
1	sysobjects	1	S	1	25
2	sysindexes	2	S	1	29
3	syscolumns	3	S	1	28
4	systypes	4	S	1	18
5	syscomments	6	S	1	10
6	sysfiles1	8	S	1	4

Figure 3. Microsoft SQL Server Query Analyzer

This tool is also an ad-hoc tool; even though it is possible to save SQL statements for future use. It also requires SQL syntax knowledge. Third party tools such as TOAD, PL/SQL look similar. These are also rich with additional database administration functionalities. In this particularly case, additional functionality provided by these tools cannot be seen as an advantage. Security and data protection is the main reason. Statistics show that 80% of all security breaches are coming from inside companies. So exposing minimum of the necessary data can be a part of the solution. Protecting data from non-intentional and intentional deletion and corruption is also an important part of the data security. Enterprise tools are also part of the Enterprise Editions and additionally increase Overall Costs of Ownership.

Alternative Development Tools

JAVA and .Net Framework are competitive, but not compatible technologies. These technologies offer corresponding set of tools for the application development. The similarity between Java and C# is obvious. These are object-oriented languages. The language structure is the same, and everything is packed in a class module. The both compilers produce half-compiled code. In the case of Java it is called byte-code and C# compiler produces Microsoft Intermediate Language (MSIL) code. Both of the languages use syntax similar to C/C++. Java is executed inside Java Virtual Machine (JVM) and C# is executed inside .NET Framework. Java offers operating system independency and cross platform portability. The C# project, called mono, is developing .Net Framework under UNIX. Comparing Java and C# and commenting C# reliability, productivity and security, James Gosling, Java inventor says: *"You find stuff in it that has essentially loopholes for everything. They had this problem in their design rules that they had to support C and C++, which means you have to have a memory model where you can access everything at all times. It's the existence of those loopholes that is the source of security, reliability and productivity prob-*

lems for developers. So on the one hand, they copied Java, and on the other hand, they added gratuitous things and other things that are outright stupid. That's amusing." (Wong, 2002).

There are also differences between the two languages. Java is a cross-platform portable and platform independent language. Even though C# can be used on any platform that implements .Net Framework and Common Language Runtime (CLR), it is currently connected closely to Windows platforms only. Java language is interpreted but the C# is compiled. Java did not implement pointers, but C# still supports pointers in the unsafe code. JVM executes only java byte code. The CLR executes any code compiled in the MSIL. It creates a system that is language independent. Any language that implements Common Type System (CTS) can be executed in the .NET Framework and CLR. The major reasons to choose between Java and Microsoft platform could be application architecture requirements. If the application is platform independent, then, as it is now, Java platform has an advantage. If the applications are closely tied to the Windows platform, then Microsoft platform has an advantage. Java can also be used for current project implementation, but then Java Server Pages (JSP) must be chosen for Web development too. Java Server Pages (JSP) is a corresponding technology to the Active Server Pages (ASP) pages and ASP.NET too. ASP.NET is the next generation of the Microsoft ASP. These technologies provide rich set of the functionality to implement Web applications that create dynamic content. JSP is built on the top of the Java Servlets technology. Both are built to handle dynamic page contents easily. Both are splitting presentation and business. Both technologies replace Common Gateway Interface (CGI) programming and make Web pages development much easier. Main differences are in the supported platforms, Web Servers and languages used. The JSP are portable across different platforms. JSP is a part of the J2EE. As it is now, ASP.NET is mainly connected to the Windows platform. Almost any Web servers such as BEA Web Logic, IBM Web Sphere, Apache, Microsoft IIS, support JSP. ASP.Net is supported only by Microsoft IIS; even third party products are available to support ASP.NET on the UNIX platform. JSP uses Java programming language. The ASP.NET can be programmed by using C#, VB.NET and JScript.Net and any other .NET language. ASP.NET introduces a rich set of new functionality not available in the previous version and significant improvement. One of the most important is real object oriented language support, session management, Web Forms and event driven Web application. These two technologies are competitive and mutually exclusive. It means if JSP is used for development, then Java language and Java platform compatible tools (software) are used in the process of web page development. In another case, when ASP.NET is used, then .NET languages and Microsoft tools are used in the process of web page development.

Architecture and Software Design

Purpose of the Architecture and Software Design

The purpose of the Architecture and Software Design is to provide overview of the technical architecture and guidelines for Universal Help Desk Database Access Tool software implementation. This section describes the content and the interfaces of the different software tiers and layers.

Scope of the product

The primary software solution is providing easy, flexible and user-friendly access to the target database objects. The tool should be able to provide access to the dislocated database systems and independent of the database model changes.

Technical Architecture

Technical Architecture (Figure 4) shows typical n-tier, layered system architecture.

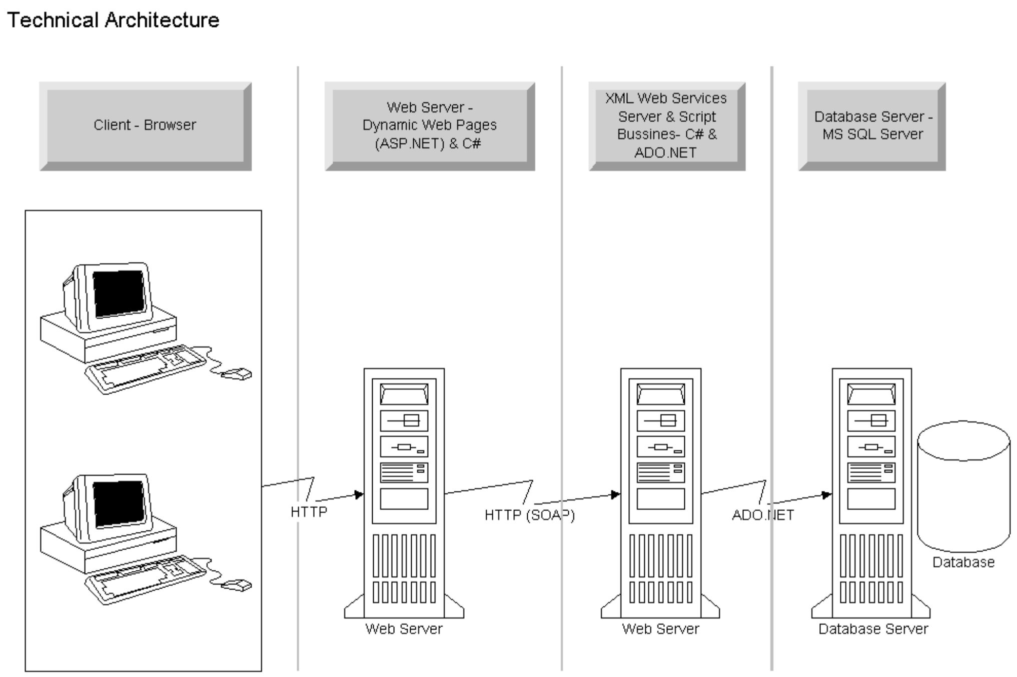

Technical Architecture
Client - Browser
Web Server - Dynamic Web Pages (ASP.NET) & C#
XML Web Services Server & Script Bussines- C# & ADO.NET
Database Server - MS SQL Server
HTTP
HTTP (SOAP)
ADO.NET
Web Server
Web Server
Database Server
Database

Figure 4, Technical Architecture

The high level of the system design is divided into Client and Server layers. Client – Browser layer is using a thin client, browser and HTTP protocol to access Server layers. Server layer is divided into Front-End part, Web Server - Dynamic Web Pages layer and Back-End, XML Web Services & Script Business Server and Database Server layer. Communication through Front-End and Back-End are achieved through HTTP protocol. The Front-End and Back-End are communicating by exchanging XML messages. User can access the Database Server layer only through Web Server - Dynamic Web Pages layer and through XML Web Services & Script Business Server layer. XML Web Service Server layer communicates with Database layer by using Microsoft ADO technology. XML Web Services Server layer and Database layer do not need to be installed on the different servers. However, Web Server –Dynamic Web Pages should be installed on separate servers in their own Demilitarised (DMZ) zone and dislocated from the rest of the system because of the security reasons.

System Context

Context Diagram (Figure 5) describes the Front-End Server component, Web Interface, and the stakeholders.

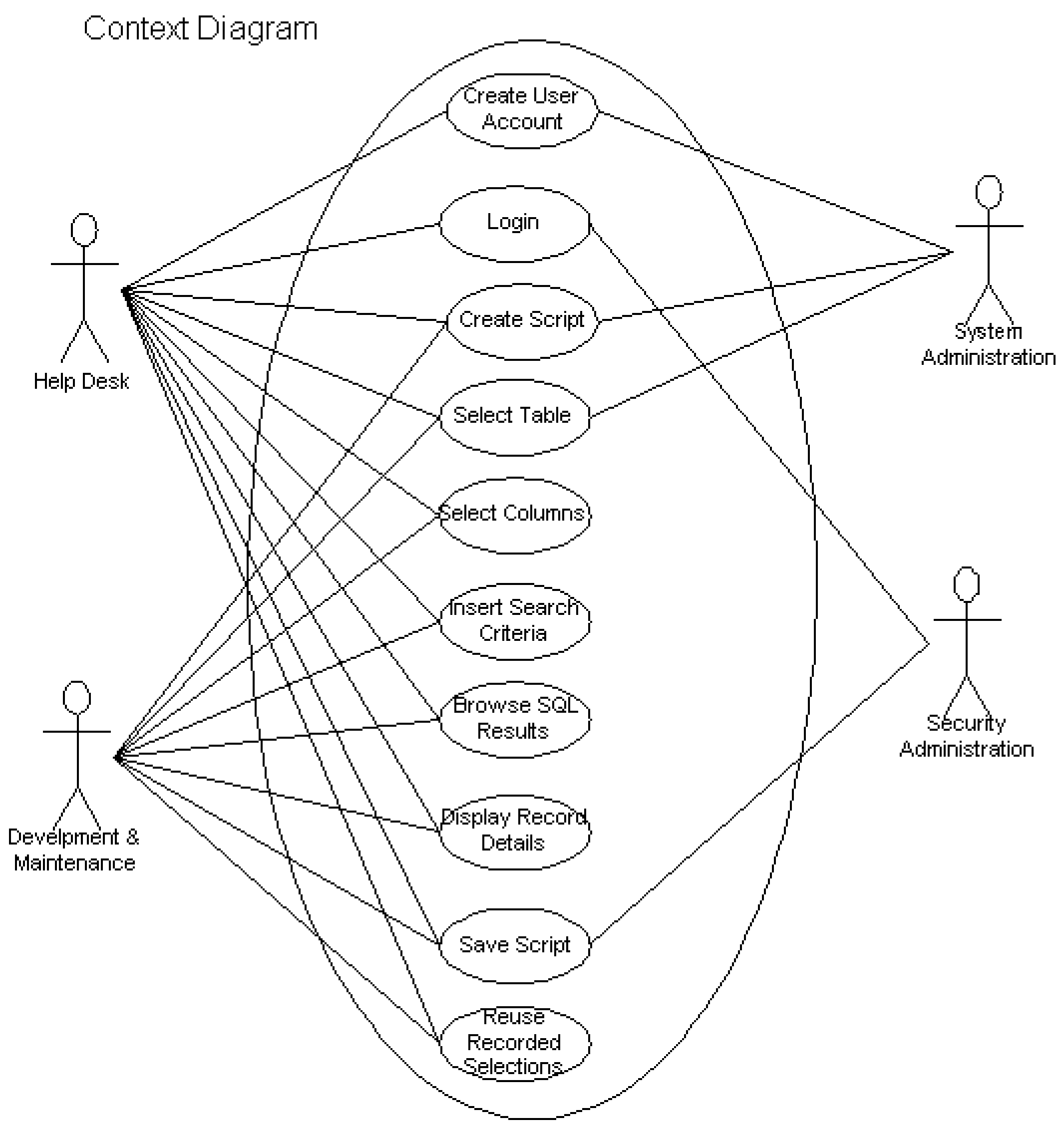


Figure 5, Context Diagram

On the left side of the system context are end users Help Desk and Development and Maintenance. On the right side of the system context are System Administration and Security Administration. The main role of the system administration is to provide server access and database access to the end users.

Application Architecture Design

Layered Architecture (Figure 6) presents application logical layers. Physical implementation can differ and is described in Figure 4, Technical Architecture.

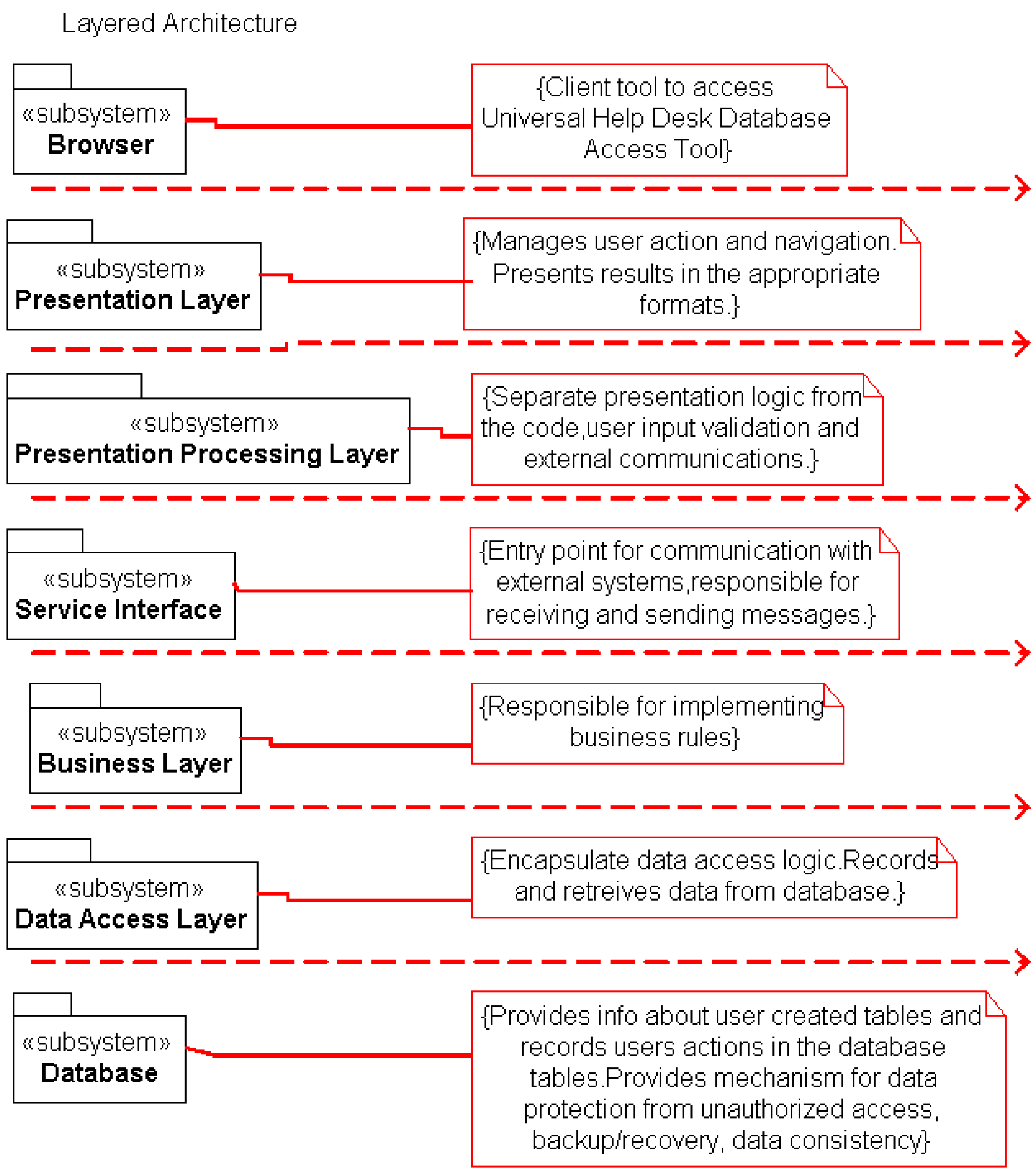

Layered Architecture
«subsystem» Browser
{Client tool to access Universal Help Desk Database Access Tool}
«subsystem» Presentation Layer
{Manages user action and navigation. Presents results in the appropriate formats.}
«subsystem» Presentation Processing Layer
{Separate presentation logic from the code,user input validation and external communications.}
«subsystem» Service Interface
{Entry point for communication with external systems,responsible for receiving and sending messages.}
«subsystem» Business Layer
{Responsible for implementing business rules}
«subsystem» Data Access Layer
{Encapsulate data access logic.Records and retreives data from database.}
«subsystem» Database
{Provides info about user created tables and records users actions in the database tables.Provides mechanism for data protection from unauthorized access, backup/recovery, data consistency}

Figure 6, Layered Architecture

Presentation Layer and Presentation Processing Layer shall be located on the same server. Business Layer and Data Access Layer shall be also located on the same server because of the performances reasons. The other layers can be installed on different servers.

Principal Object Identification

Principal Object Identification (Figure 7) identified principal objects in the different layers.

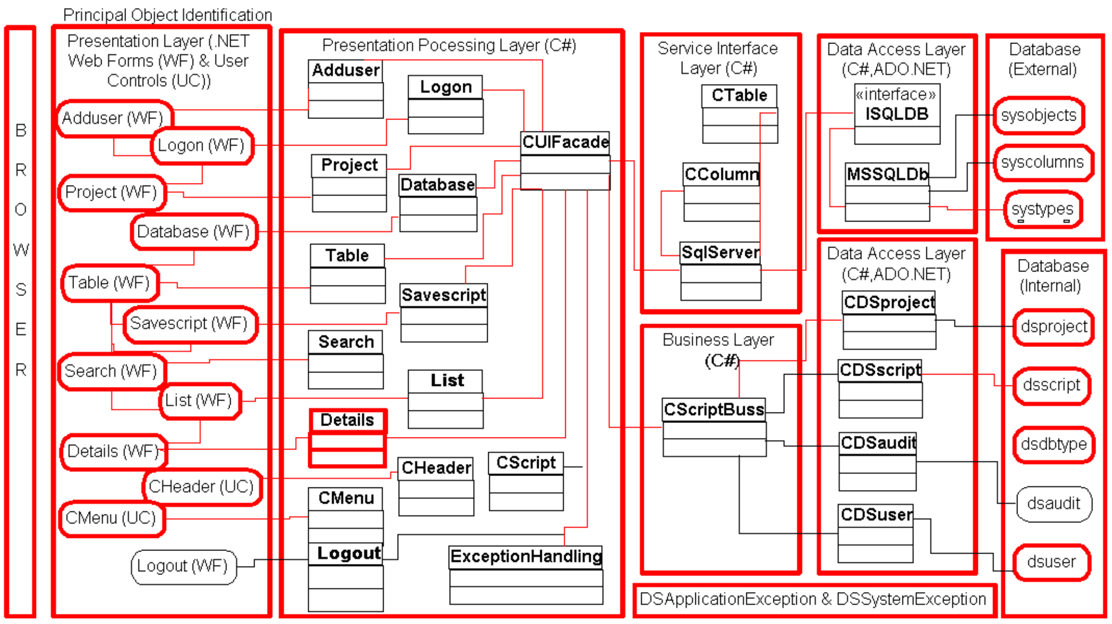


Figure 7, Principal Object Identification

Database

This project is using two kinds of tables. System tables are used to collect information about user created tables, table columns and column types. These tables are called SYSOBJECTS, SYSCOLUMNS and SYSTYPES in MS SQL Server database. This project is implemented by using MS SQL Server database and it is a reason why these names are used. In the case of another SQL database system these tables might have different names; such as, the corresponding tables are called SYSTABLE, SYSCOLUMNS and SYSDATATYPES in DB2 database. Data Access Layer, Facade and Builder pattern will provide methods that deal with database specifics.

Another group of tables are project tables where information about projects, databases, tables and scripts are stored. Business rule describes how these tables are used. Entity-Relationship data model describes relationship between these tables.

Database Model

Database Model ER-Diagram (Figure 8) represents data model. Relational database stores meta-data in the system tables. By browsing certain tables it is possible to get information about available tables, table columns and columns types. Universal Help Desk Database Access Tool application uses this information to dynamically create user interface for accessing certain data table.

Below model presents a set of the system tables called sysobjects, syscolumns and systypes that contain SQL database meta-data. The rest of the tables are used to store the user created scripts. Even in the Figure 8, these two data table sets are drawn in the same space; in the real world these are usually dislocated. System tables provide information about remote database system and user created tables. DS tables are used to record access to the remote database and reuse it in further investigation or repeating the old ones.

In the case of the MS SQL Server database, information about created tables are stored in the 'sysobjects' table. Each table has a unique 'id' number. Unique 'id' number can be used to find information about table columns in the table 'syscolumns'. Each column has information about name, type, length, precision, etc. Information about column type can be found in the table 'systype' by using 'xusertype' as a search key. In this model system tables contains only columns that are used by this project. These tables contain large number of other columns that are not presented in above model. Using database tools such as Enterprise Manager in the case of the MS SQL Server database can see these.

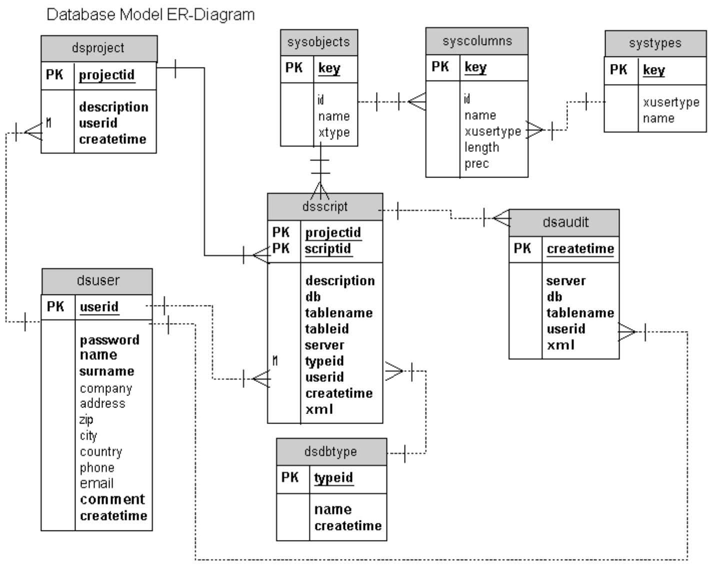


Figure 8, Database Model ER-digram

In the case of the MS SQL Server database, information about created tables are stored in the 'sysobjects' table. Each table has a unique 'id' number. Unique 'id' number can be used to find information about table columns in the table 'syscolumns'. Each column has information about name, type, length, precision, etc. Information about column type can be found in the table 'systype' by using 'xusertype' as a search key. In this model system tables contains only columns that are used by this project. These tables contain a large number of other columns that are not presented in above model. Using database tools such as Enterprise Manager in the case of the MS SQL Server database can see these.

The user created tables can be selected from this table by following query:

SELECT name, id FROM sysobjects WHERE xtype = 'U'

This query selects all user created tables from target database. It actually selects table name and table Id. Table Id is a number that it is used as a key to access another table to get information about columns in the selected table.

The following query selects all table columns:

SELECT name, xusertype, length, prec FROM syscolumns WHERE id = <id>

By using selected 'xusertype' it is possible to get information about column data type. The following query selects column data type:

SELECT name FROM systypes WHERE xusertype = <xtype>

The sequences of these queries can provide all necessary information to build a user interface to the database table.

Another group of tables, with the names starting with 'ds' are used for recording user scripts and activities.

'DSPROJECT' table records created projects.

'DSSCRIPT' table records created scripts. The 'xml' field in this table contains XML document that contains all application script settings. When stored script is loaded, this XML message is parsed and recreated the saved CScript object that is used in the current application.

'DSAUDIT' records all SQL scripts executed against the database. The 'xml' field in this table contains a whole SQL statement that is used to access remote database system. This information together with other table fields can be used to analyze security breaches and other kinds of analyses.

'DSUSER' table records all users related information and can be used to validate user account information in the case that XML Form based authentication is changed from using User.xml file to the SQL database table.

'DSDBTYPE' is a lookup table for SQL database type. This is currently implemented for the MS SQL Server database. When the other database access is implemented, and then must be a corresponding entry in this table.

Summary and Conclusion

The primary goal of this project has been development of the tool for accessing different database systems that it is independent of the data model changes. This tool shall be able to access a target database even though the database model has been changed. Inserting a new table column, deleting the old one or changing column name will change the database model. This goal has been achieved by using SQL database meta-data to dynamically create database calls that are satisfied by the current database model. The other goal is to access different database systems; it has been achieved by XML Web Service design and by using Builder Pattern to make it easy to add the new class that implements necessary interface methods and provides physical access for the target database.

This approach has a number of advantages over developing static application for accessing remote database system. The first and obvious one is that it offers significant savings and reduces overall application development cost. This approach offer generic application that is independent of the target database model changes. It is not limited to a certain set of tables but rather accessing all user created tables, but still allows database administrator to limit table access for certain users or user groups. Access to database table(s) is enabled by using minimum of necessary granted options. The audit log provides historical information of each database access. This tool could be easily connected to Change Management and track all actions when database needs to be analyzed.

The project has been developed as a Web application by using XML Web Services. The XML web services layer encapsulates database access for the remote database systems. The XML Web Services technologies provide software-to-software communication across systems and network boundaries. The standardization and wide acceptance by leading IT market companies make the XML Web Services technologies an attractive choice for distributed applications development. However, there are still some drawbacks when using the XML Web Services such as missing transaction management and common security standards, even though there is a continuous work to close those technology gaps. The security model implemented in this project and based on the XML Forms authentication shall be supported by current security standards. Even though the security issues are out of the scope of this project, for real production implementation Secure Socket Layer (SSL) should be used, to encrypt login data during network transport.

The other changes related to the currently used authentication model can be improved by using SQL database table for storing user account information. Currently these are stored in the XML file. The main reason is the more sophisticated isolation level system in the case of the SQL database and SQL database integrated security. The data model already contains a database table that corresponds to the XML file used for storing user account information. In the case of .Net Framework, it is very easy to switch from XML to Dataset format and vice versa. The Dataset can be presented as a XML message, or populated from the XML message. The other limitation in this project is the SQL statements that are built from the user input in the search forms. In the case when more form fields contain search criteria logical 'and' is always used in the conditional part of the SQL statement. It can be improved by letting the end user to specify conditions by presenting the choice of the 'or', 'not', 'like', 'in' and other SQL operators. However this is left for further development and it is out of the scope of the current project.

The technology used for project implementation is based on the Microsoft .Net Framework platform. The Visual Studio.Net has been used as a development tool and the MS SQL Server database has been used for the data model. Even now there are still some unpleasant bugs in the Visual Studio. Net, such as 'Build is corrupted' and problem with the debugging web forms, it is a great development environment that provides a lot of useful tools and rapid application development. The built in support for XML and XML Schema, together with visual Web Forms design, debugger, documentation tools an utilities for transforming XML Schema to C# class and vice versa and tools for creating application installation files, makes it the right choice for development in the Microsoft environment. The drawback is that, as it is now, it is not platform independent. There is a project called 'Mono' that should provide .Net Framework also under Unix/Linux platform.

The differences between unit testing and integration testing are difficult to see in the case of the online applications development. For example, when the unit test on the Web Form is performed, it is at the same time integration test for current module and if Web Form request or store data in the remote database, it is already complex integration test. There are tools available on the market for unit testing. One of them is NUnit (NUnit 2003), the tool for creating unit test during development and also for automatic test execution and regression testing. This tool is used in the current project, primary for testing the XML Web Services. The NUnit (NUnit 2003) is a free-ware and can be freely downloaded, distributed, used in commercial application and even can be extended with your own code. This tool unfortunately has not been used in the current project from the beginning.

The patterns are everywhere. This project has been used Architectural and Design patterns recommended by Microsoft .Net Architecture Center (MSDN Microsoft, 2003c). The current project architecture is designed by using layered, n-tier architecture. The project design implements MVC Pattern, Facade Pattern, Data Access Pattern and Builder Pattern. The current project scope does not include deployment project. Development of the deployment project needs own resources and it is usually developed as parallel project. This project scope of the deployment is the network infrastructure, server's configuration, fail-over configuration, load-balancing, vertical and horizontal scalability, clustering and deployment in the Web farms. The current project can be deployed to satisfy above-mentioned development project scope. The layered architecture and XML Web Service make it possible to deploy application on the dislocated servers. This application can be easily connected to the Configuration Management Change Management procedure. If the data model contains reference to the Change Management call ticket, then all database access related to the current call ticket can be tracked. The audit table records SQL statements that are used to access the database tables. The inputs in this table are recorded chronologically. If the Change Management calls ticket number is a part of these recordings then it can be easily reconstructed and later on all SQL statements that are used in the process of the error recreation/investigation can be analyzed. This can help in the case of similar or the same errors to find what procedures for error recreating are used in the previous case.

References

Advanced Information System. (1996). Oracle Unleashed, Second Edition. SAMS Publishing. Retrieved 5 March 2003 from http://www.netti.hu/doc/Oracle.Unleashed.Second.Edition/ch13/0289-0293.html

Armstrong, E., Bodoff, S., Carson, D., Fisher, M., Fordin, S., Green, D., Haase, K., & Jendrock, E. (2003). "The Java Web Services Tutorial". Retrieved 7 March 2003 from http://java.sun.com/webservices/docs/1.1/tutorial/doc/

Bulajic, A. (2003). Master's Thesis. The University of Liverpool and KIT e-learning.

Chappell, D. (2002). *Understanding .NET: A tutorial and analysis*. London: Addison-Wesley.

Cockburn, A. (2001). *Writing effective use cases.* London: Addison-Wesley.

Codd, E. F. (1970). A relational model of data shared for large data bank. *Communications of the ACM.* Retrieved 3 March 2003 from http://www.acm.org/classics/nov95/toc.html

"Codd's 12 rules" for a fully relational DBMS. (2003). Retrieved from http://newton.uor.edu/FacultyFolder/CKettemborough/Codd12R.html

Cooper, J. W. (2000). *Java design patterns A tutorial.* Boston: Addison-Wesley.

Deitel, H. M., Deitel, P. J., Nieto, T. R., Yaeger, C. H., Zlatkina, M., & Litfield, J. (2002). *C# How to program.* Upper Saddle River, New Jersey: Prentice Hall.

Esposito, D. (2000). Creating and optimizing performance for XML document/view Web applications. *MSDN Magazine*. Microsoft Corporation. Retrieved 17 March 2003 from http://msdn.microsoft.com/msdnmag/issues/0600/cutting/default.aspx

Extreme Programming: A gentle introduction. (2003). Retrieved 22 June 2003 from http://www.extremeprogramming.org/

FFE Software. (1996). Open Database Access and ODBC. Retrieved 8 March 2003 from http://www.martinscholl.com/html/mhc/odbc_history.html

Fowler, M., & Scott, K. (2000). *UML distilled second edition: A brief guide to the standard object modeling language*. Boston: Addison-Wesley.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns elements of reusable object-oriented software.* Boston: Addison-Wesley. Retrieved 17 March 2003 from http://c2.com/cgi/wiki?DesignPatternsBook

Howe, D. (1993). The Free Online Dictionary of Computing. Retrieved 6-Mar-03 from http://foldoc.doc.ic.ac.uk/

Java. (2003). Web Services. Retrieved 7 March 2003 from http://java.sun.com/webservices/

Mackenzie, D. (2001). Architectural options for asynchronous workflow. Microsoft Developer Network, Microsoft Corporation. Retrieved 16 March 2003 from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/bdadotnetarch12.asp

McJones, P. (1997). The 1995 SQL reunion: People, projects, and politics. Retrieved 5 March 2003 from http://www.mcjones.org/System_R/SQL_Reunion_95/index.html

Mercury Interactive. (2003). TestDirector. Retrieved 17 June 2003 from http://www-svca.mercuryinteractive.com/products/testdirector/

Microsoft. (2002a). Application architecture for .NET: Designing applications and services. Microsoft Corporation. Retrieved 14 March 2003 from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp

Microsoft Visual Studio.NET. (2002a). Introduction to ASP.NET Web application in Visual Studio. Microsoft Development Environment 2002, Online Documentation, Microsoft Corporation

Microsoft Visual Studio.NET. (2002b). Programming the Web with XML Web services. Microsoft Devel opment Environment 2002, Online Documentation, Microsoft Corporation

MSDN Microsoft. (2003a). XML Web services developer home. Microsoft Developer Network, Microsoft Corporation. Retrieved 7 March 2003 from http://msdn.microsoft.com/webservices/default.aspx

MSDN Microsoft. (2003b). Microsoft patterns & practices: Complete List. Microsoft Developer Network, Microsoft Corporation. Retrieved 14 March 2003 from http://msdn.microsoft.com/practices/type/CompleteList/default.asp

MSDN Microsoft. (2003c). .NET architecture center. Microsoft Corporation. Retrieved 18 June 2003 from http://msdn.microsoft.com/architecture/

Nunit. (2003). .NUnit. Retrieved 18 June 2003 from http://www.nunit.org/default.htm

On Relations and Relationship. (2003). Retrieved 4 March 2003 from http://accesshelp.net/content/view.aspx?_@id=53437

Plew, R. R., & Stephens, R. K. (2000). SAMS teach yourself SQL in 24 hours (2nd ed.). SAMS Publishing. Retrieved 5 March 2003 from http://members.tripod.com/er4ebus/sql/ch01.htm

Rational. (2003a). IBM Rational RobotUnified. Retrieved 14 June 2003 from http://www.rational.com/products/test.jsp

Rational. (2003b). Rational unified process. Retrieved 14 June 2003 from http://www.rational.com/products/rup/index.jsp

Spenik, M., & Sledge, O. (2003). *Microsoft SQL Server 2000 DBA survival guide*. SAMS Publishing.

System R. (1970). Home Page. Retrieved 5 March 2003 from http://www.mcjones.org/System_R/index.html

W3C. (2002a). Web services descriptions requirements, Working draft 28 October 2002. Retrieved 6 March 2003 from http://www.w3.org/TR/2002/WD-ws-desc-reqs-20021028/#definitions

W3C. (2002b). Web service activity. Retrieved 6 March 2003 from http://www.w3.org/2002/ws/

W3C. (2003c). XML encryption WG. Retrieved 6 March 2003 from http://www.w3c.org/Encryption/2001/

W3C. (2003d). "XML signature WG. Retrieved 8 March 2003 from http://www.w3.org/Signature/

Webopedia. (2003). N-Tier application architecture. Jupitermedia Corporation. Retrieved 15 March 2003 from http://www.webopedia.com/quick_ref/app.arch.asp

Wong, W. (2002). Why Microsoft's C# isn't. Retrieved 9 March 2003 from http://news.com.com/2008-1082-817522.html

Web Service Interoperability Group. (WS-I) (2002a). Resources and guidelines for WEB services interoperability. Retrieved 7 March 2003 from http://www.ws-i.org/

WS-I. (2002b). Promoting Web services interoperability across platforms, applications and programming languages. Retrieved 7 March 2003 from http://www.ws-i.org/docs/20021017.introduction.mht!20021017.introduction_files/frame.htm

Biography

Dr. Samuel Sambasivam is the chairman of the Department of Computer Science of Azusa Pacific University. Professor Sambasivam has done extensive research, publications, and presentations in both computer science and mathematics. His research interests include optimization methods, expert systems, Fuzzy Logic, client/server, Databases, and genetic algorithms. He has taught computer science and mathematics courses for over 20 years. Professor Sambasivam has run the regional Association for Computing Machinery (ACM) Programming Contest for six years. He has developed and introduced several new courses for computer science majors. Professor Sambasivam teaches Database Management Systems, Information Structures and Algorithm

Design, Microcomputer Programming with C++, Discrete Structures, Client/Server Applications, Advanced Database Applications, Applied Artificial Intelligence, JAVA and others courses. Professor Sambasivam coordinates the Client/Server Technology emphasis for the Department of Computer Science at Azusa Pacific University.

Aleksandar Bulajic is living in Denmark. He got his Master of Science degree from Liverpool University. He has been working as an IT developer, software specialist and consultant for more then 20 years in different companies in Europe. He is currently working for Maersk Data A/S. Maersk Data is a member of the A.P. Møller group, the largest Danish concern. He has been working on MVS, UNIX and Windows platforms with different developer tools and developed applications by using different programming languages such as COBOL, PL/I, C, C++, Visual Basic, C#, Java and .NET. The fields of his special interests are User Requirements Gathering, Project Organization, Integrated Development Environments (IDE) & Development Tools, Web Applications & Web Services, Testing and Software Development Automation. He is using Java and .NET technologies in current project developments.