Architecture and Design of Customer Support System using Microsoft .NET technologies

Nikolay Pavlov
PU Paisii Hilendarski
236 Bulgaria Blvd.
Bulgaria, Plovdiv 4003
npavlov@kodar.net

Asen Rahnev
PU Paisii Hilendarski
236 Bulgaria Blvd.
Bulgaria, Plovdiv 4003
assen@pu.acad.bg

ABSTRACT

This paper describes the four-tiered architecture, technologies, functionality and electronic services for the participants in the process of customer support with a software system for customer support – Integrated Help-Desk Center (IHDC), based on an object-oriented framework for development of distributed applications.

There exist multiple solutions for customer support management. Many of them do not provide services to end-clients, do not support vertical organizational structure or lack relevant multi-language support for international clients.

The participants in the process of customer support in IHDC are: clients, local partners, local branches, central office and development department. Multilingual support is provided to enable operation over different counties. IHDC consists of: Customer Relationship Management; System for registering and management of tickets; Management of application known issues; Management of application updates.

The system is developed with Microsoft .NET Framework. Its infrastructure is build upon Microsoft Windows Server 2003, Microsoft SQL Server 2000 and Microsoft Information Server. The four tiers are: database server, application server, functional objects and thin client interface – Windows-based and browser-based.

Keywords

Four-tiered architecture, object-oriented framework, customer support, Microsoft .NET Framework.

1. INTRODUCTION

High-quality customer support services have been always identified as an important element of the overall package of software services for customers, crucial for the mission of every software company. Therefore, successful software companies strive to provide increasingly higher quality services to their customers, and seek ways to achieve this by automating and optimizing their processes. One of the necessary elements is to have a centralized repository of all customer-related issues, and to have an established process for handling those issues, to ensure that no problem is neglected or processed inadequately. The dynamic of the modern world creates new economic and cultural environments, thus putting further difficulties before companies, which operate across country boundaries. Some of those problems include multi-level company hierarchical structures and multi-national fields of operation.

There exists a wide range of software solutions for customer support. Many of them are not suited for the latest requirements for quality customer support, because they lack support of certain features. Common disadvantages are:

- Support of horizontal company structure only.
- No multi-language support.
- Insufficient integration with existing office packages.
- Insufficient functionality for Customer Relation Management (CRM), or integration with third-party CRM systems.
- Cannot operate in a distributed environment, for example - over the Internet.
- Do not support built-in declarations of hours and costs.

.NET Technologies 2006
Copyright UNION Agency – Science Press,
Plzen, Czech Republic.

This paper describes a Customer-Support System (CSS), which is aimed to enable multi-national companies provide high-quality support services to their customers by offering an affordable and flexible solution, which overcomes the limitations stated above.

A major advantage of the proposed CSS is the automated translation-request system. This system monitors and assigns translation tasks to appointed personnel, when information is crossing language boundaries within the organization. For example, a call from the client needs attention from a higher level of support, where responsible employees do not speak the client's language. In this case the system assigns all the information, as provided by the client, to a translator. Also, all information, which is to be communicated back to the client, is translated before being made available to the client.

The architecture of the Customer Support System is four-tiered and is based on an object-oriented framework for development of distributed applications. The system infrastructure and is build upon Microsoft Windows Server 2003, Microsoft SQL Server 2000 and Microsoft Information Server. The four tiers are: database server, application server, functional objects and thin client. The system is implemented using technologies, based on Microsoft .NET Framework.

2. ROLES IN CSS

The design of CSS identifies the following roles for participants in the process of customer support:

- Central office (management).
- Branches
- Partners
- Development
- Clients

Central office represents the management of the company, providing the services. This role performs the highest level of support and supervision of the performance of all other levels. Central office is the only instance, which communicates with development, thus providing a centralized and controlled information flow towards development.

Branches are head offices for countries. There is only one branch per country. Branches provide support for all customers from the corresponding country. Branches also serve as a bridge between clients and the Central Office, and enhance communication flow to and from the Central Office by providing translations whether necessary.

Partners are agents within one country, and subordinates of the corresponding branch. Partners provide first level of support, education and other services like installations, configuration on-site, demonstrations. They communicate most actively with existing clients and potential clients.

Development is a department, which provides software services like bug fixing, product extensions, new versions, etc. Issues, which cannot be solved otherwise, are ultimately sent to development by the Central Office. Development never has a direct contact with clients and other levels.

The following diagram represents the structure of the roles, as defined by CSS.

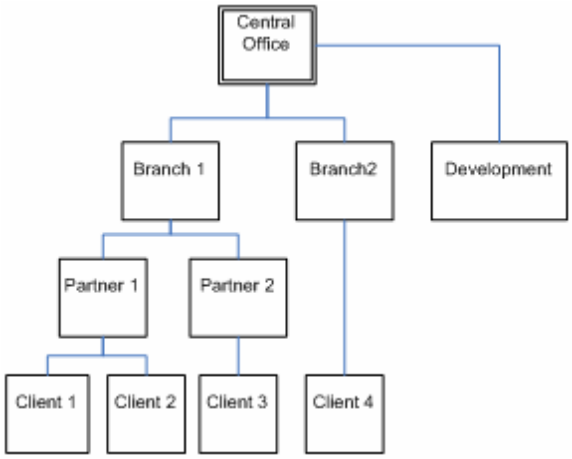

Figure1. Roles Structure in CSS

Figure 1 gives an example of the relations between different roles in CSS. The direct link between Branch 2 and Client 4 means that the role of a Partner is not required and can be skipped in certain cases. For example, in a relatively small market partners may not contribute to the efficiency of the organization and, therefore, will not be established.

Clients are the end-customers of the organization. They contact with the organization always via a Partner or a Local Branch.

3. SERVICES FOR PARTICIPANTS IN CSS

Services for Clients

Clients in CSS are provided with a browser-based Internet application – thin client, with which they can do the following:

- Enter new issues (tickets).
- Attach documents and files to tickets.
- Review status of tickets.
- Provide additional information on existing tickets on request.
- Close tickets when solved.
- Review existing known issues.
- Review new releases, fixes and release notes.

Tickets are entered under pre-defined categories, and with different urgency level. Clients enter a description of their problem / question, and can attach an external file – office document, screenshot, etc to a ticket. Clients can check existing known issues and their solutions to determine if there is a ready answer to their problem. Clients can monitor the progress on each of their tickets, and provide additional information if the customer support asks for it. Certain actions the customer support produce an e-mail to the client to emphasize on issues that would need quick attention. Clients can also see all new releases, patches and fixes, to determine if they should upgrade their software.

Services for Partners

Partners use the desktop-based client of CSS. Partners can see all tickets, entered by their clients, and take actions on them. Partners can provide solutions, request additional information, or send tickets to branches for higher level of support if the solution is beyond their competency.

Partners are provided with access to the database with all their clients, including contact and address information, plus all application releases, fixes, and special consultancy documentation, which is provided by the central office.

Services for Branches

Branches use the desktop-based client of CSS. They see all tickets, entered by clients in the corresponding country. In this way branches can monitor the performance of all their subordinate partners and take necessary measures. Branches see tickets, received directly from their clients, and tickets, for which partners cannot provide a timely solution. Branches can provide solutions, request additional information, or send tickets to the central office, if the solution is beyond their competency. Branches have tools to provide translated information to the central office, if necessary.

Just like partners, branches are provided with access to the database with all their clients, including contact and address information, plus all application releases, fixes, and special consultancy documentation, which is provided by the central office.

Additionally, branches have access to a database with all their partners.

Services for Central Office

The Central Office uses the desktop-based client of CSS. They see all tickets, and thus can monitor the performance at any sub-level. There is an automated system, which automatically escalates tickets, which are not processed timely by the responsible sub-level.

The Central office has all necessary facilities to provide solutions and request additional information. It can also assign tickets to development department, if the problem cannot be solved with other means. There are tools for on-line discussions and solution design. Other tools exist for authoring of release notes, and creating known issues from similar tickets. The Central office can also monitor the performance of the development department.

There is a special tool, which provides summarized information about the status at any level within the organization, with special emphasis on overdue work, or work, approaching its designated deadline. This tool enables management to quickly spot problematic nodes in the company structure and take the necessary measures to resolve the issues.

When a client is updated to a new version of the application, and the update is actually a new application, not simply an upgrade of the old one, all its existing information has to be converted. This includes all tickets, installation information, etc. There is a special tool, which facilitates this process. It archives all data, relevant to the previous version, and creates the necessary structures for the new application.

Services for Developers

Development department uses the desktop-based client of CSS. They see only tickets, assigned to development by the central office. Developers can use the tool for on-line discussions to receive logged additional information. Developers report their work to the Central Office, which is responsible for testing their work before delivering the solution to the customer, and for authoring the necessary release notes for both clients and other levels of support. Development department is isolated from clients, and direct communication between these two parties is not allowed.

Other Services

CSS contains a system for automatic escalation of tickets. It monitors if a ticket is not processed timely at any level below the central office. In such a case, the ticket is escalated automatically to the higher level. This system also monitors ticket deadlines. If a deadline is approaching, the system sends notifications by e-mail to the responsible employees at the current level of support and the central office.

There is and integrated Customer Relation Management system, available to partners, branches and the central office, is. It provides an extensible

data structure for storing client-related commercial information. This system focuses on development of new clients, and management of sales.

Another integral part of CSS is the system for registration of visit reports. Visit reports summarize all agreements and arrangements, negotiated during meetings between clients, company personnel and representatives, and external consultants. Information includes all participants in meetings. The system prepares report documents for each meeting and sends those by e-mail to all participants.

4. PROCESSES IN CSS

CSS defines a schema of sequential processes for handling tickets. It goes in following steps:

1. Ticket is entered by client.
2. The appropriate partner sees the ticket. The partner can solve the problem, and notify the client to approve the solution, or, escalate the ticket to the branch. If no partner is available, this step is skipped.
3. The branch sees the ticket. They can provide a solution, or escalate the ticket to the central office, if they cannot handle the ticket. If a solution is found, the branch can directly implement the solution, or send the ticket back to the partner for implementation. Before escalating the ticket, the branch should translate the ticket into the language of the central office, if necessary.
4. Ticket is received at the central office. If the central office can provide a solution, the ticket is sent back to the branch for localization and implementation. If the problem requires programming, the ticket is assigned to development with additional description and translation, where necessary.
5. Development department receives a ticket with a detailed description, and scheduled dates for start and completion of work on every ticket. When work is completed, the ticket is sent back to the central office for approval.
6. A completed ticket is sent back to the central office. They test the solution and depending on the result, can sent it to the branch for implementation, or revert it back to development.

At every level, except for development, support can request client to send more information, suspending the ticket. The ticket is reopened automatically, when the client gives an answer.

When a ticket is solved, the client is notified and the ticket is suspended. It is the client who does close the ticket.

Suspended tickets are automatically reopened if no action is taken on them for a specified period of time. This logic prevents tickets from being forgotten and stalled.

5. ARCHITECTURE

CSS is developed through the use of an object-oriented framework for distributed business applications. This section describes the key features of the framework.

Four-tier Application Architecture

The architecture of the framework is presented on figure 2.

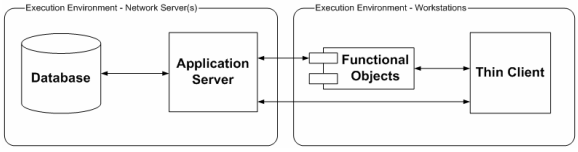

Figure 2: Four-tier architecture

The four-tiers (Figure 2) are: database server, application server, functional objects, and client.

The database is responsible for storing the application data, as well as the internal framework data - application dictionary, security, and customer preferences. Application and framework data are stored within one logical database, which improves encapsulation.

The application server is an intermediate layer between the database and the functional objects. It does not implement business logic; in stead, it provides a number of services to the other layers – services for data access and modification, security services, communication services and system services for initialization, multi-language support and maintenance. It is the only layer that communicates directly with the database server, which enables development of applications for various database platforms by changing only the application server. Access to the database is realized with ADO.NET. The application server is multithreaded – each client is served by a separate thread, which improves the performance on SMP and Hyper-threading systems.

The functional objects are specially designed program modules, which are integrated at run-time within the process of the client application. They provide the functional core of the client application. The functional objects software components, which accomplish a certain task. They realize the business logic of the application and a part of the user interface.

The client application is an environment for execution of functional objects under a common user interface. It provides a number of services for:

- Common graphical user interface
- Load, execute and release functional objects
- Translate all user interface text items (commands, menus, and static texts) towards the active functional object
- Data exchange between running functional objects

The choice of desktop-based architecture for the client application is motivated by the significantly richer features for building of the graphical user interface (GUI), which desktop-based GUI technologies present. The strong support of various infrastructures for distributed applications in Microsoft .NET Framework enables desktop-based applications to access resources and components over local networks and over the Internet - .NET Remoting, SOAP (Simple Object Access Protocol) based web-services, TCP/IP sockets, robust COM (Component Object Model) integration. These are strong foundations for easily building applications with fully-featured, convenient and aesthetic user-interface, while not being limited to client-server application architecture.

Another significant benefit of desktop-based applications is that they make it possible to implement "push" callbacks – events. Such events may be triggered when a user modifies a record in the database, thus notifying all the users, working in the same logical sub-domain, that a relevant data modification has taken place.

The application framework, employed to develop CSS, is using .NET Remoting to realize the communication between the client and the server application. .NET Remoting may be used over the Internet, though its callback features are not suitable for wide-area networks, due to technical and security restrictions. Therefore the application framework implements callbacks via proprietary TCP/IP connections, established by the client application to the application server. This is necessary to resolve issues with firewalls and NAT (network address translation), which "hide' the clients from the server.

Scaling

The architecture of the framework defines two execution environments – servers and workstations. It is possible to run the database and application server on a single server, or scale the environment and have database(s) and the application server running on different server computers. For example, one way to scale up is to run the application server and the database with system data on one box, and the application user data on a separate box. Further scaling is possible by using more than one application servers, each running on a separate computer. This, however, has its drawback – callbacks (events) are not possible across multiple application servers. Proper organization of work may overcome the negative effect, because in large organizations each department has its own area of operation and it is less critical for immediate view of all data modifications, made across the organization. Additionally, in stead of using callbacks, the client is able to use "pull" technology to acquire events from the application server.

Standard Functional Objects

Targeting enhanced code reuse, the framework includes a set of functional objects, which implement security, data browsing and searching, data modifications, reporting, document integration with Microsoft Office and other external documents, and database integrity administration. They are versatile and function according to the specific definitions in the application dictionary.

Data overview: browsing, and sorting data, search for data, filtering data on user-selected criteria. Data is displayed in table format, with options for additional information in addition to the table. Searching and filter can be performed on fields from the table, as well as on other related data – both one-to-many and many-to-many relationships are supported.

Data entry and modification: entry of data, with built-in facilities for client-level data validation. Additional services include copy of data, and edit of multiple records with a single operation.

Reporting services: preview and print of reports. Custom reports can be created on any level, with a What-You-See-Is-What-You-Get (WYSIWYG) editor. Reports can be exported into popular formats like Abode Portable Document Format (PFD), Microsoft Excel (XLS), and HTML.

Integration with Microsoft Office (Microsoft Word): creation, storage and retrieval of Microsoft Word documents, which contain data from the database of the client application. Active links between the documents and the data is maintained. Microsoft Word document files are stored on an especially designated storage folder, which allows access to them even in case of system failure, and provides an organized depository of office's files.

Attachment of external documents (files) to existing application data. An essential part is the descriptive definitions of relevant application data, which

external files can be attached to. Links between application data and the files attached are created. Attached files are stored on an especially designated storage folder, which allows access to them even in case of system failure, and provides an organized depository of office's files.

Security management: application administrators can assign access policies to application users and user groups. There are two types of access policies: on application level, and on data level. Application level security policies determine which screens and functions are available to a user, while data level access policies determine which data is accessible by a user.

Data Integrity management: application administrators can overview all modifications, made by users, and take necessary actions to sustain the logical integrity of the application data.

Those functional objects allow development with minimum, even no code pre-compilation by using the application dictionary. The application dictionary is the "heart" of the framework – it is a centralized repository of logical, functional and business definitions. It describes the hierarchical structure of the application, the user interface – icons, menus, toolbars and forms, and the access security roles on both application and data level. It contains all parameter definitions for the functional objects and thus determines their behavior in every part of the application.

Application dictionary is created in a special descriptive language, based on Extensible Markup Language (XML). The application dictionary is stored in the database of the application and is always interpreted on application startup. The client application parses only the structure, to build the menus and screens, and the security policies for the current user. Functional object parse their designated parameters on loading.

An integral part of the framework is the special tool for authoring of application dictionary contents. It features a schematic presentation of the application structure, plus syntax-highlight editor for the parameters. Authoring of application dictionaries requires understanding of the client database structure, Structured Query Language (SQL), and of the specifics of parameter definition schemas for every functional object used. As a result, application development and support can be performed by non-programmers.

Multi-language support is handled with a tool, which enables the application administrator to translate all text items in the application into virtually any number of languages. It provides convenience facilities: incremental searching, filters for not-translated items, searching for similar translations, etc

CSS is developed on Microsoft .NET Framework. The communication between the database server and the application server is done via ADO.NET. The communication between the application server and the functional objects and the thin client is done via .NET Remoting over TCP/IP. Client front-end is realized as a ASP.NET, installed on Microsoft Information Server; it uses .NET Remoting to communicate with the application server.

Microsoft SQL Server 2000 is used as a relational database server for CSS.

6. REFERENCES

[1] Object-Oriented Application Frameworks, Fayad M., Schmidt D., Communications of the ACM, Special Issue on Object-Oriented Application Frameworks, Vol. 40, No. 10, October 1997

[2] Ralph Johnson and Brian Foote, "Designing Reusable Classes", Journal of Object-Oriented Programming. SIGS, 1, 5 (June/July. 1988), 22-35

[3] Pavlov N., Rahnev A., Framework for Application Development, Scientific works of Plovdiv University, Bulgaria, vol. 35, book 3 - Mathematics, 2006