# A New Helpdesk System at UWSP

# Praktikum paper

Tobias Isenberg

e-mail: *isenberg@cs.uni-magdeburg.de*

Stevens Point, May 13, 1998

# Contents

# Praktikum Task

Helpdesks are departments in today's organizations that help keeping hardware and software running by organizing problem solving requests. The current helpdesk software, Star, at UWSP only runs under MS-DOS/MS Windows 3.11. Due to the decision to upgrade the whole campus to the MS Windows 95 operating system in 1997, a new helpdesk software system had to be implemented. In a decision process, it was first agreed upon to buy the commercial product "Applix Helpdesk" and additional components like a Web connection module from Applix, Incorporated. The original task of this Praktikum was to install the product, to configure the software, and finally to provide a working system, including the integration of the Web connection module and a knowledge base.

In the process of the installation, it turned out that there are not only major problems with the software, but also that Applix Helpdesk does not provide all the necessary and promised features. Therefore, the task changed to the implementation of a new workorder system similar to the old Star system. The new system had to work under the new operating system MS Windows 95. The implementation was first planned to be realized using the platform Clarion4. However, due to problems with this programming tool, it was then decided to implement using MS Visual Basic 5.

| | |
|---|---|
| Time frame: | September 1997 – May 1998 |
| Responsible for the task: | Prof. Dan Goulet (University of Wisconsin – Stevens Point, Department of Mathematics and Computing) |
| Advisors: | Prof. Dan Goulet (University of Wisconsin – Stevens Point, Department of Mathematics and Computing) |
| | Dr. Bernhard Preim (Otto-von-Guericke Universität Magdeburg, Institut für Simulation und Graphik) |
| | Dr. Jörg Schirra (Otto-von-Guericke Universität Magdeburg, Institut für Simulation und Graphik) |
| | Debbie Smith (University of Wisconsin – Stevens Point, Information Technology) |
| | Scott Gile (University of Wisconsin – Stevens Point, Information Technology) |

# 1. Introduction

In today's world, there is probably no large enterprise or organization that can exist without the help of computers anymore. And these computers unfortunately do not always work properly. If everybody with problems with his/her computer or the whole system would try on his/her own to find someone to fix those problems. The result would be a chaotic and very inefficient working process.

That is why most large organizations established a centralized troubleshooting system. All troubleshooting requests are centrally collected and are forwarded to the right people with the appropriate knowledge that is required for solving the problem. The process of collecting and storing data about the problem and forwarding the request to information technology experts can be computerized, and thus made much more efficient and effective.

The University of Wisconsin-Stevens Point used Clarion for DOS to write an application to aid this process at their helpdesk. Because of the transition from the MS-DOS/MS Windows 3.11 operating system to the Windows 95 operating system on the campus in 1997, and because of the incompatibility of the old helpdesk program with the new operating system, the implementation of a new system became necessary.

This paper discusses the installation of the commercial product Applix Helpdesk 6.0 and the related components, and the implementation of a new system in Clarion4 and MS Visual Basic 5.

# 2. Helpdesks and Helpdesk Software

In this chapter, the topic, *helpdesk*, is discussed in general. In the beginning, some definitions are given. After that, the necessity for centralized troubleshooting is considered. Then, the typical design of a helpdesk and of helpdesk software is described.

## 2.1. Internal and External Helpdesks

First of all, the actual meaning of the term *helpdesk* has to be defined. According to [Mon97], a helpdesk is a department within a larger organization to which all the computer problems (hardware and software) of this organization are reported, and that helps to find a solution to these problems. If these problems cannot be solved by the helpdesk workers right away, they are forwarded to other people who are able to solve the specific problems. These people try to solve the problem or do their part in solving it. If they cannot solve the whole problem, they forward the problem to other specialists, and so forth, until the problem is finally solved.

Obviously, the term *helpdesk* can also be used in a broader sense and does not necessarily have to be applied only to computer problems. However, because this Praktikum deals with an information technology-related helpdesk, the term will be used in the narrow sense.

In [Mon97], two different kinds of helpdesks are defined. First, there are *internal helpdesks*, which serve only the employees within a certain organization. The second kind of helpdesks are *external helpdesks*. They have to provide solutions for external people. An example for the second kind is a product support department that answers the questions of customers about a certain software product of the company.

Since the specific task for this Praktikum deals with the *internal helpdesk* of the Information Technology department of UWSP, this paper only focuses on internal helpdesks.

## 2.2. Necessity for Centralized Troubleshooting

After defining the term helpdesk as a centralized troubleshooting department, the necessity and advantage of such an entity has to be defended. An organization would not invest into a helpdesk department if there would not be a good reason for doing so.

Normally, if someone has a problem and cannot solve it himself/herself, that person could just call someone who possibly knows of a solution to the problem, and ask for help. Eventually, this leads to a solution for the problem. However, most people usually do not know the right people for solving certain problems. This is especially true in large organizations with many people working on different projects, and therefore, having a variety of knowledge and background about computers.

Furthermore, if there are people constantly calling for help, the people asked are disturbed in their own work and cannot complete their tasks. The interruption would slow down those people considerably in completing their proper work. Therefore, in case of an external help request, this would make the products of the external company more expensive. Internal requests for help would similarly result in inefficiency and misallocation of staff recources.

If, instead, just a limited number of people work for a helpdesk, and problems are reported to them, the whole process of problem solving would become much more efficient. The problems would be solved more quickly and by people with sufficient knowledge. Therefore, a centralized helpdesk gives everybody just one point to address in case of a problem with software and/or hardware. This also eliminates the disadvantage of certain employees who do not know the right people to ask. Additionally, it reduces the disturbance of other people who are working on their own tasks.

Another advantage of a centralized troubleshooting department lies in the knowledge that is accumulated by the helpdesk staff while operating, since different users might encounter the same problem independently from each other. This knowledge can either be accumulated by the staff operating the helpdesk, or can be stored in a knowledge base linked to the helpdesk software. Therefore, after a while, certain common problems can be answered even more quickly.

In [Mon97], a calculation for the justification of an internal helpdesk is given. Supposingly, the cost of an employee lies at least at about $50 per hour (inluding salary, taxes, office space, equipment, training, insurance, etc.). If such an average employee loses one hour to figure out the solution to some kind of a computer problem, the organization loses $50. Such a problem might take a hired helpdesk expert about 10 minutes to figure out. In other words, an investment of $10 for 10 minutes of the time of the helpdesk expert (whose hourly costs would be higher at supposingly $60 per hour) saves $50 by freeing one hour of the time of the normal employee. This investment has suddenly returned 500%.

| | Employee | Helpdesk Expert |
|---|---|---|
| Time to solve problem | 60 min | 10 min |
| Cost | $50 | $10 |

**Table 1:** Comparison regular employee and helpdesk expert

Even if the expert only works for 50% of the time on solving problems, and suppose the expert is five times faster than normal employees, the expert would save $120 every working hour. This would be about 0.25 million dollars net savings a year!

| Helpdesk Expert | Costs | Savings | Revenue |
|---|---|---|---|
| per hour | $60 | $150 | $90 |
| per day | $480 | $1200 | $720 |
| per week | $2400 | $6000 | $3600 |
| per year (50 weeks) | $120000 | $300000 | $180000 |

**Table 2:** Costs and savings by employing an helpdesk expert (working 50% of the time on solving problems)

In summary, a helpdesk serves the purpose of solving computer problems in an organization more quickly and more efficiently. Therefore, it reduces breaks in the working process due to computer problems, and therefore, makes the work more effective.

## 2.3. Components of a Helpdesk

After exhibiting the need for a helpdesk in larger organizations, the components of a typical helpdesk will now be considered.

As already identified, the helpdesk has to be centralized so everybody knows where to ask for help. That means that the helpdesk has to be accessible by phone, e-mail, voice-mail, fax, personal contact, etc. Of course, the helpdesk has to be at a central and well-known location for providing the option of personal contact.

In order to serve an organization efficiently, the helpdesk has to be equipped with good and fast hardware. The hardware provides not only the recording and forwarding of incoming requests, but it also serves as a reference for the most common working environment in the company[1]. The computer has to be equipped with a software and hardware system supporting the operation of the helpdesk. The software part of this system are discussed in more detail in Section 2.5.

In [Mon97], for the hardware part of the helpdesk system, a telephone with a head-set, a fax machine, and sometimes even a call distribution system are named as additional technical requirements for a helpdesk system. This, certainly, is not always necessary and depends to a large extent on the size of the organization and the average number of calls that are received by the helpdesk per day.

[1]As a reference, the helpdesk could have another machine with the normal equipment and software of the organization, so both applications do not interfere with each other, and the reference is closest to the common environment

Besides the technical components, the staff operating the helpdesk also have to be mentioned. There are, first of all, the staff who actually sit at the helpdesk and answer the incoming questions and requests. Besides these people, there is also the technical staff to which the problems can be referred (dispatched) in case the primary helpdesk worker cannot solve the problem right away. These technical staff can consist of people working only for the helpdesk, or they can be a part of the normal information technology staff of the organization.

## 2.4. Procedures of a Helpdesk

After talking about the components of a typical helpdesk, now the procedures at a helpdesk will be described. The process, that typically occurs at a helpdesk, can be broken down into several more detailed steps or procedures. In [Uni97], the process is divided into the following five steps:

1. Case Occurence
2. Case Arrival
3. Case Indentification, Analysis, & Problem Resolution
4. Case Management
5. Case Closure

Because only the last three steps are directly related to the helpdesk itself, only those steps are considered in more detail in what follows.

When a problem is reported to the helpdesk, the helpdesk worker initiates the "Case Indentification, Analysis, & Problem Resolution" step (step 3). In the beginning, the caller and his working environment are identified. After that, the helpdesk worker attempts to analyze the problem by inquiring about the symptoms. In case a problem solution cannot be sufficiently determined, a more detailed analysis by internal or external resources may be initiated (dispatching, see Section 2.5).

The "Case Management" step (step 4) involves monitoring of cases, their reassessment, and customer feedback. The reassessement/feedback can either be initiated by the caller or the helpdesk. In case the problem has to reviewed, step 3 has to be repeated, otherwise the case can be closed (step 5).

The approach at the helpdesk at UWSP is explained in more detail in the list below. It follows, in general, the procedure as described above.

**Case Identification, Analysis, & Problem Resolution**

- Identifying the person
- Identifying the working environment (operating system, etc.)
- Inquiring about the symptoms of the problem
- Attempting to solve the problem
- Entering the workorder and dispatching

**Case Management**

- Monitoring the workorder by the helpdesk manager to ensure timely resolution and appropriate dispatching
- Seeking customer feedback by "follow-up-function": After a workorder is done, an IT[2] staff member ("follow-up") calls the users to make sure that everything is OK. If this is not the case, the process is started over again.

**Case Closure**

- Close workorder

## 2.5. Helpdesk Software: How the System Works

This section explains in more detail what a helpdesk software system has to provide and how such a system works. For this purpose, first the term *workorder* has to be introduced.

A workorder is a data record containing all the information of a particular problem reported to the helpdesk. Some organizations use the term *trouble ticket* or *incident report* instead for the same idea. In this paper the term *workorder* will be used. During the problem solving process, workorders are referred from the helpdesk to a worker or from one worker to another worker[3]. This process is called *dispatching*.

A software system supporting the work of a helpdesk has to implement the following items and processes:

**Workorder:**
Collection and storage of the information about the specific problem in a workorder. Such a workorder contains informatin about the caller, about the problem's symptoms, and about the location of the computer.

[2]Information Technology, department at UWSP of which the responsibilities are computer software and hardware

[3]In the current STAR system, workorders are only referred from the helpdesk to a worker or from a worker back to the helpdesk.

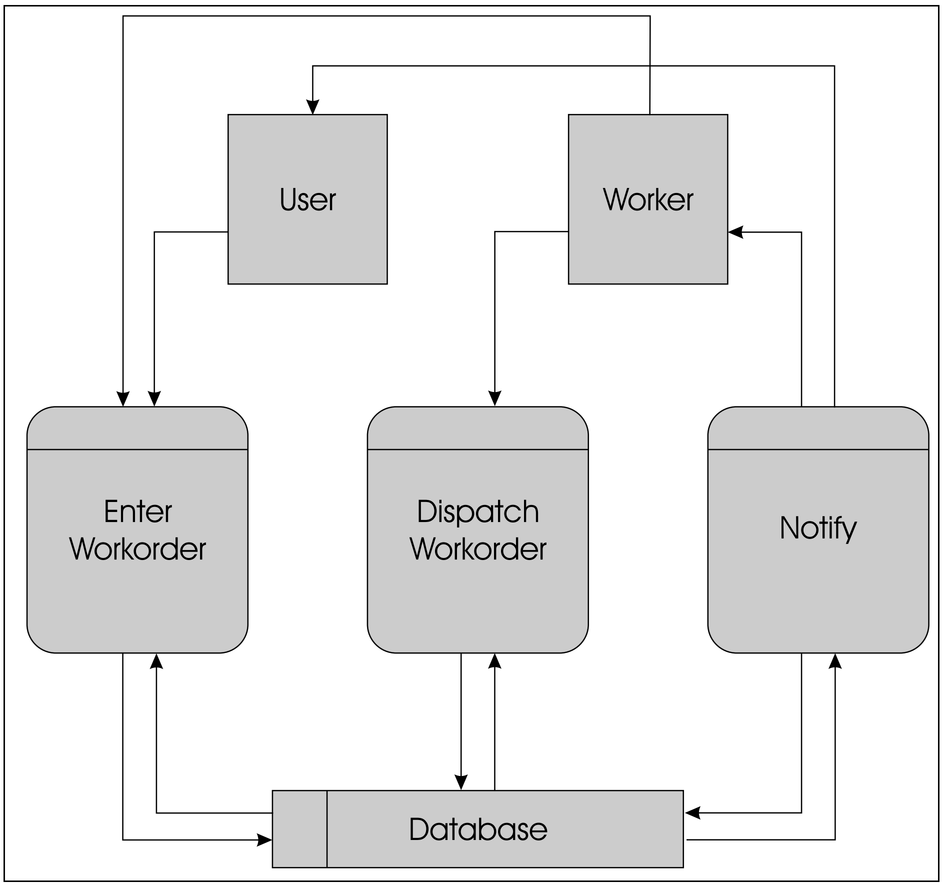


**Figure 1:** Simple helpdesk system: system diagram

**Dispatching:**
Adding additional information about the work that was done or that has to be done to a workorder and forwarding of the workorder to other workers.

**Notification:**
In some cases, certain people have to be notified about certain processes. This can happen, for instance, by e-mail and can be automatically done by the software.

A software package that implements these features can automate the helpdesk work and make it much more efficient and effective. Actually, a helpdesk without any kind of such a workorder software system would probably not make any sense, because it would not be able to serve the organization in an efficient and effective way. The above named items are the very least a workorder system would have to provide. It could be extended to include more features to make the work at the helpdesk more efficient, more effective, and easier.

There are many helpdesk software packages on the market. However, to address all of them or only a selection would be beyond the scope of this report. Nevertheless, some trends of these software packages will be mentioned briefly here as they are discussed in [Bar92].

Generally speaking, the development of helpdesk systems tends to evolve toward integrated and automated systems. This is done by incorporating modern software technologies into the former pure call and case tracking helpdesk systems.

Usually, there are huge amounts of information available electronically, for instance in form of on-line documentation or on CD-ROM. Intelligent Text Retrieval is included in modern helpdesk systems to help in searching this documentation for specific information that is needed to solve a certain problem.

Unlike the this very general technique, Case-Based Reasoning adapts information retrieval especially to the field of helpdesks. It searches the database of previously solved cases to find information about similar cases that might be helpful to solve the current case. This technique becomes very useful after the helpdesk has accumulated a certain number of solved cases or when new workers with less experience join the helpdesk staff.

For both of the just mentioned techniques, the Intelligent Text Retrieval and the Case-Based Reasoning, there is already knowledge for sale in the form of knowledge bases and case bases. The case bases are especially helpful in the first phase of using a new system when only a few cases have been completed.

Also expert systems are used in recent helpdesk systems to support the problem solving process. Some systems store expert knowledge in the system in the form of decision trees. Other systems support the problem specification process by asking certain standardized questions about the problem.

# 3. Helpdesk Software Upgrade at UWSP

This chapter describes the specific problem of the Praktikum. The approach for solving the problem is explained in theory. The old system is considered first. After that, the approach with the helpdesk software by Applix is explained. Finally, the implementation in Clarion4 and MS Visual Basic 5 is discussed.

## 3.1. The Old Star System

UWSP currently uses a software system, called STAR[4], to support the work at their helpdesk. It was written in 1990 by programmers from UWSP's Information Technology in Clarion for DOS and runs only under MS-DOS/MS Windows 3.11. STAR is a simple workorder system which implements basic functionality, such as, entering workorders into the system, dispatching them to other workers (queues), and notifying people, as described in Section 2.5. It does not communicate with other systems or databases on campus.

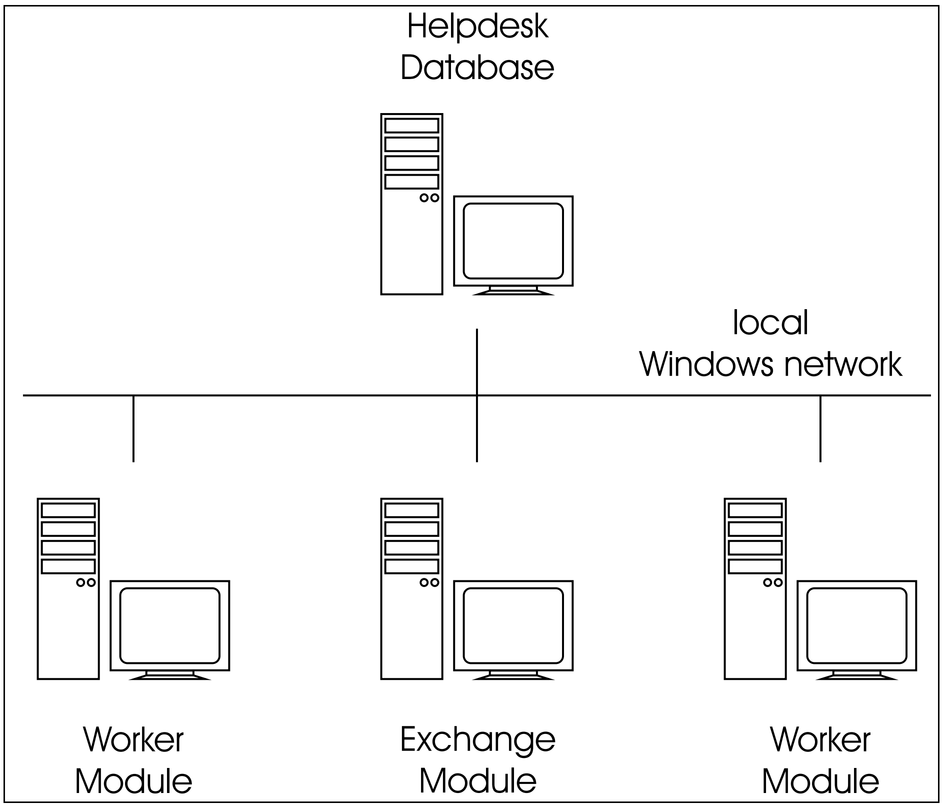


**Figure 2:** Old STAR system: system diagram

[4]Service Tracking And Reporting

The old STAR system uses a central Clarion database to store the workorder data. The different modules access this database to create new records or update old ones. Workorders can be entered using the worker module, or they can be created when users use a certain e-mail form in UWSP's MS Exchange e-mail system. E-mail notifications are initiated, when necessary, during the process.

When a problem is reported to the helpdesk operator by phone, e-mail, or personal contact, the data is first recorded with paper and pen. If the problem cannot be solved by the helpdesk worker right away, the data is entered into a workorder and dispatched to one of the workers in IT. Approximately 850 calls a month are received by the UWSP helpdesk, and additionally, about 358 workorders are entered by users into the system using e-mail forms in MS Exchange.

At any time, a worker can look into his/her work queue (workorders dispatched to him/her), pick one of the workorders, do the necessary work, and document the work on the workorder. After that, he/she either dispatches the workorder back to the helpdesk for further dispatching, in case there is additional work to be done, or dispatches the workorder to the "follow-up" queue, if the problem has been solved. This invokes the Case Management step as described in Section 2.4, where the user is asked if the problem was solved. In case it was not solved, the workorder is sent back to the helpdesk for further dispatching. Otherwise, the workorder is closed.

In the old STAR system, there is no extensive security implemented. Every worker has access to the queues of any other worker. However, this is a desired feature and not a downfall of the system. The primary worker of a workgroup has to be able to log in and have a look at the workorder queues of his/her delegated workers. Also, the helpdesk manager has to be able to check the status of a complaint at any time.

In 1997, the UWSP campus' computer operating system was updated from MS-DOS/MS Windows 3.11 to Windows 95. However, the old STAR system was not compatible with the new system. Therefore, a new solution had to be found. A decision for buying a commercial software system was made, and the new helpdesk software system's main required features were defined by IT (refer to [Smi97], attachment B, for the complete listing) as follows:

- Client-Server-design
- Web interface for other users to enter and check workorders
- Unlimited space to record workorder activity
- E-mail notifications
- Searchable fields
- Connection to other databases (holding, for instance, information about the employees or students)

- Report capabilities
- Interface with HP OpenView to allow for automatic workorder creation for network errors

## 3.2. Applix Helpdesk

According to these features, the software, Applix Helpdesk 6.0 by Applix, Incorporated, was chosen from a list of helpdesk systems, and finally purchased. Applix Helpdesk promised to meet all of these criteria and even some additional features like a knowledge base.

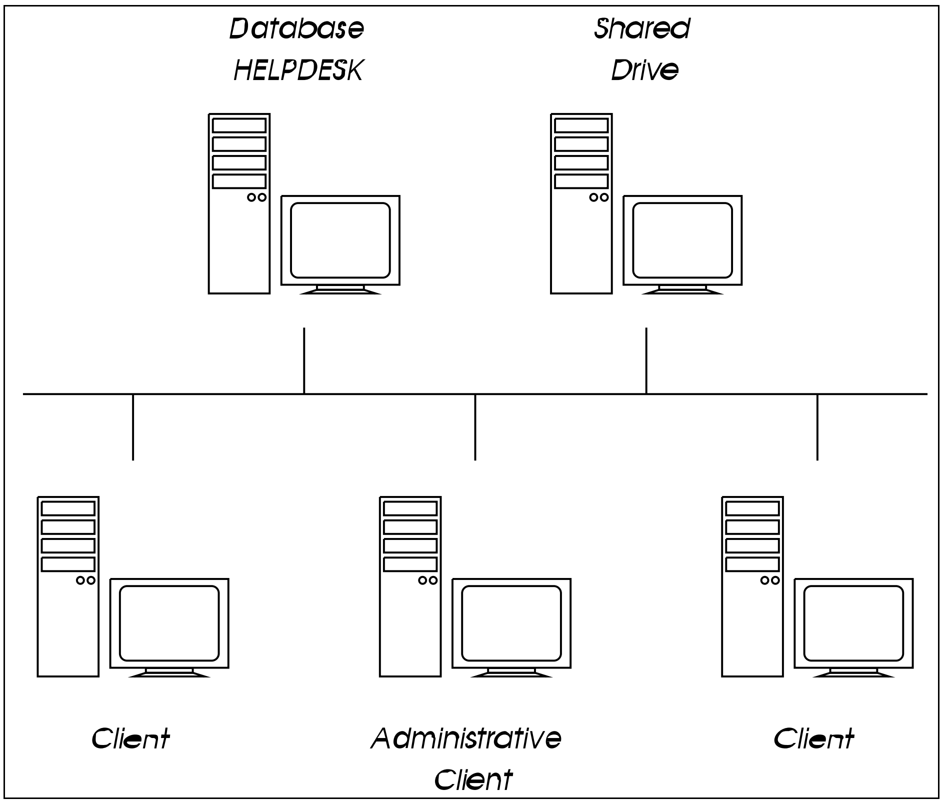


**Figure 3:** Applix: system diagram

Figure 3 shows the design of the Applix system. As can be seen from the figure, the main core of the Applix Helpdesk system is a database residing on a database server. Applix Helpdesk gives the option of working with different database server software systems, such as MS Access, Oracle, Informix, Sybase, or MS SQL Server 6.5. In the local approach, a SQL database residing on a MS SQL Server 6.5 was used. All the workorder data is recorded in this database, along with the views[5] and stored procedures[6] used by the client software.

The clients, which can run on the same machine or other machines, access the data in this database using the ODBC[7] interface. One of these clients is the designated administrative client. This client provides additional functionality for administering the system, for example customization of forms and maintenance of user security.

For ease of installation, the installation files are stored on a shared directory within the MS Windows network. Some standard settings are already created for these installation files during the installation of the Applix system. However, the software also offers the option of creating installation disks.

In addition to the clients, Applix also provides the option of accessing the data via a Web interface using its Applix Weblink component. This is a component residing on another machine which provides the interface to the WWW. Applix Weblink needs one of the Web serving software packages available on the market, like, for instance, MS Internet Information Server.

Besides of the features mentioned, Applix offered to connect their system to a knowledge base they would supply. The supplied knowledge base was created by a third party vendor and contained information about standard hardware and software. In addition, a "self-learning" knowledge base was included, which would collect information about solutions found from earlier problems and provide suggestions for solving new problems.

## 3.3. The New Star System

During the time of the Praktikum, it turned out that the Applix Helpdesk software did not provide all the functions as described and promised in the contract. Thus, a decision was made to return Applix Helpdesk, and to adapt the earlier written Star software to the Windows 95 operating system.

[5]Queries on the database.

[6]Functions that allow the execution of certain commands on the server directly.

[7][Mic98]: "Open Database Connectivity (ODBC) technology provides a common interface for accessing heterogeneous SQL databases. ODBC is based on Structured Query Language (SQL) as a standard for accessing data. This interface provides maximum interoperability: a single application can access different SQL Database Management Systems (DBMS) through a common set of code. This enables a developer to build and distribute a client/server application without targeting a specific DBMS. Database drivers are then added to link the application to the user's choice of DBMS."

## Design

The design of the new STAR system could be approached independently from the actual development platform. Similar to Applix, STAR was designed to use a SQL database on a MS SQL server as the main core of the program. This database is accessed by the three main modules that build up the STAR system:

1. Worker Module
2. Web based forms
3. Report module

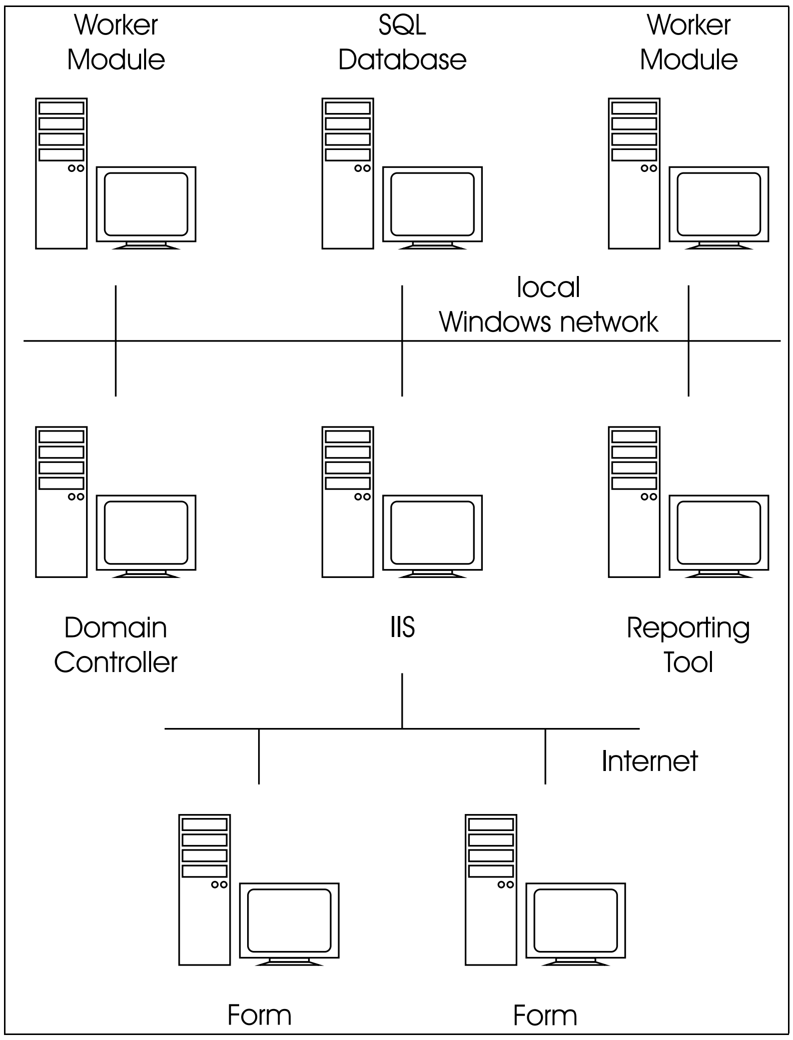


**Figure 4:** STAR: system diagram

As shown in Figure 4, the worker module accesses the database via the local MS Windows network using the ODBC interface. This module provides the workers at IT with the possibility of entering workorders (at the helpdesk itself), of looking at their (and other's) workorder queues, and of dispaching workorders after they finished the work on these. A filter feature enables workers to see only a part of the workorders in the system, for example only their own work queue, or the workorder queue of a category of problems, for instance all hardware related problems.

The Web clients enable faculty, staff, and students of UWSP to enter additional workorders into the Star system. In order to verify that only those groups of people are able to enter workorders, they have to log into the system using their UWSP logon. This information is verified with the domain contoller. The clients are implemented in Active Server Pages[8] and are executed on an Internet Information Server 4.0 (IIS).

The report module finally allows for generating statistical reports on the accumulated data. These reports will include, for instance, reports on the number and the kind of workorders, or on how many workorders a certain worker has worked on. This module is planned to be written in Crystal Reports by Seagate, and will most probably be a Web-based application. However, it is not a part of this Praktikum as explained in section 4.4.

After talking about the design of the Star system, the following part describes the implementation platforms and why they were choosen. It, thereby, focusses specifically on the worker module and its implementation.

### Clarion4

To upgrade the old Star system, which was written in Clarion for DOS, the same programming language in its current release should be used: Clarion4 by Topspeed. Clarion4 is a fourth generetion, object oriented, automated development environment. It is relatively easy to create a running program, and it is not necessary to know much about the programming language of Clarion.

Learning Clarion4 is very easy because the learning curve to use the development tool is very short. Usually, a beginner does not have to learn the actual programming language when starting to use the tools, because they offer the use of wizards. These wizards enable the programmer to create a fairly complex database application without the need of writing code. Clarion4 provides help for beginners in the form of tutorials. These tutorials provide a walkthrough of the main features and show the procedures for creating a simple application.

[8]Active Server Pages (ASPs) are ActiveX scripts and ActiveX server components executed on the server side. They allow the creation of dynamic Web pages. ASPs require Microsoft's Internet Information Server 3.0 or higher to be executed.

The advantage of using a new version of the formerly used software development environment is that it was considered to be easy to just re-compile the whole project in the new version of Clarion. This would have provided a program running under the new environment without additional programming work, and would have given a starting point from which to go on to revise the software and adapt it to the new requirements. Additionally, Clarion4 promised an easy way to just compile a project to the Web. This seemed to be an easy way to provide a program accessible in a variety of different platforms and locations. Workers, for instance, would be able to check their work queue by logging into the Star system from anywhere on campus, or even from home.

However, during the attempts of developing in Clarion4, major difficulties were encountered as described in more detail in Section 10.

### MS VisualBasic 5.0

After encountering the difficulties with the implementation in Clarion4, another plattform had to be chosen. Instead of using Clarion4, the worker module was finally developed in MS VisualBasic 5.0.

This was, first of all, because the IT staff at UWSP is very familiar with the MS VisualBasic development tool. Because IT will have to maintain Star after its implementation is finished, this is an important factor in the decision for the development tool. Also, MS VisualBasic 5.0 is readily available at UWSP since the university has a campus license for this software package.

Another key factor to be considered in the decision for an alternative development tool is compatibility. Because the campus uses mainly software products from the Microsoft Corporation, it would be an advantage if the Star system would be imlemented using a development tool by Microsoft as well.

Two other factors in the decision for MS VisualBasic 5.0 were the ease of coding and the ease of implementing a graphical user interface (GUI) in VisualBasic.

# 4. Development of a New System at UWSP

This chapter addresses the individual practical steps that were taken towards a new implementation of a helpdesk or workorder system for the Helpdesk at UWSP, and the problems that were encountered. First, the approach and problems with Applix Helpdesk and Applix Weblink are shown. Then, the implementation of the new workorder system in Clarion4 and MS Visual Basic 5.0 is explained.

## 4.1. Installation of Applix Helpdesk

According to the documentation of Applix Helpdesk, the recommended platform for the server is MS Windows NT 4.0. Therefore, as a first step, a hardware and software platform was accordingly assembled and configured.

Since the product required the Software MS SQL Server in its 6.0 version or later, this software package had to be running on one machine. First, an existing MS SQL Server 6.5 installation on a separate machine was used. However, Applix Helpdesk caused some crashes of this SQL Server. Because the server was also used and needed to run other applications on campus, a separate SQL server for the test phase of the implementation had to be employed. Therefore, a MS SQL Server 6.5 installation was set up on the same machine that also served as the server for Applix Helpdesk. This also meant an easier access to the SQL server in case of a crash.

The installation of Applix Helpdesk was started according to the description in the Applix Administrator's Guide [App96a]. The installation process ran without problems. The first problem was encountered when the database setup was attempted. The database setup is assumed to create a database on the SQL server and set up initial values. At a certain point, the setup always came up with an error message due to the fact that Applix Helpdesk was only working with 16 bit ODBC connections.

In contrast to that, MS Windows NT 4.0 originally only supports the use of 32 bit ODBC connections. However, the installation of 16 bit ODBC on the Applix server did not prevent the error from occuring. Applix's database setup created and tried to use 16 bit data sources, and therefore, a proper connection to the SQL server, obviously, could not be established. The product support of Applix, Incorporated could not provide a solution either.

Finally, a work-around for this problem was found. For enabling Applix to establish a proper connection to the SQL server, a 32 bit data source with the same name and properties of the corresponding 16 bit data source had to be created manually. After the database setup had access to the 32 bit data source, it worked properly.

The second problem occurred when attempting to enter the license codes that were provided. As happened with the database setup program, the software could not establish a propper connection to the SQL server. Again, the creation of a 32 bit data source with the same entries as the 16 bit data source that was created by Applix' installation program, solved the problem, and the license codes could be entered.

When logging into the system for the attempt of entering the license codes, Applix Helpdesk asked to login as the database server administrator. In the case of MS SQL Server this is the user "sa". This database administrator has all possible permissions on the database server. To ask for a user with this amount of privileges is a rather unusual behavior. It should have been enough to ask for the user of Applix's database on the server, which had to be created before the installation according to Applix' manuals. This user has all permissions necessary for working with Applix' database.

However, after the installation was complete, the software still did not work properly. Applix' client was still sometimes causing the SQL server to crash, as described earlier. The exact cause of the crashes could not be determined. Also, the login into the software did not work properly.

Finally, in contrast to the installation procedures described in the documentation, the installation was tried from a computer running MS Windows 3.11 and the installation procedures worked without problems. This time, apparently, the use of 16 bit data sources was not a problem because MS Windows 3.11 only allows for 16 bit ODBC connections since it is a 16 bit environment. However, according to the installation manual, the installation should have correctly been performed from a computer running Windows NT.

The installation program was performing the following tasks:

1. Copying of an installation directory for the client software to the server machine.
2. Creating the necessary tables and views in the previously created database on the SQL server.
3. Setting up the installation machine as the administrative client by installing the client software.

Further problems with Applix Helpdesk also included that settings necessary for the software were not initially set by the setup program. They had to be obtained from the product support hotline or were given during a training for the software. The installation guidelines provided by different sources (installation manual and training material) were different in the order of the steps that were to be performed during the installation.

The installation order that finally worked was the following:

1. Installation of MS Windows NT Server 4.0 on a computer named “ITHELP” (required by the license code of Applix Helpdesk).
2. Installation of MS SQL Server 6.5 (not necessary on the same computer).
3. Installation of 16 bit clients for MS SQL Server 6.5 on the client computer (the computer the package was installed from automatically became the administrative client).
4. Installation of MS SQL Server 6.5 Service Pack on the server.
5. Creation of a user “Helpdesk” on the SQL server.
6. Creation of new database “Helpdesk” on the SQL server.
7. Installation of the Applix software on the administrative client machine (running MS Windows 3.11).
8. Running of Applix DBSETUP on the administrative client machine.
9. Running the Applix License Manager on the administrative client machine.

## 4.2. Installation of Applix Weblink

After the installation of Applix Helpdesk finally succeeded, the Web interface to the package Applix Weblink had to be installed. The installation program of this package worked without problems.

One prerequisite for the operation of Applix Weblink was the installation of a WWW serving software package. This WWW server can be provided in form of the MS Internet Information Server 2.0, which comes with the standard installation of MS Windows NT 4.0. The MS Internet Information Server was one of the choices of WWW serving software packages that worked with Applix Weblink.

After the installation was complete, it was possible to log into the system. However, the package did not provide a complete menu. The Web pages that were transmitted were incomplete. Only the footer information was sent. The menu items and the beginning of the Web page were missing. Therefore, the system was not working.

Again, the product support service of Applix was contacted to inquire about a solution for the problem. The hotline provided additional settings for the configuration files that were not mentioned in the installation guidelines, and that should have taken care of the problem. Unfortunately, these new settings did not prevent the system from sending incomplete pages.

Another problem with the WWW connection to the server was that the password, which is necessary for logging into the system, was transmitted in clear text. The system was not making use of the features of today's Web browsing software of encrypting the data traffic between browser and server. This could have been done by using the https[9] protocol, which is supported by all major browsers.

Connected to the issue with the password transmission was the problem with the authentification at the Web server. In order to be able to log onto the server, it would have been necessary to set up a special user account for every single user. This would not only have meant setting up an extra account for every faculty member, staff member, and student on campus, but it would also have meant that the users would have to maintain a second password. This was not acceptable.

## 4.3. Return of the Software

During the process of installation, obtaining the newer version Applix Helpdesk 7.0 was considered. At this time, the new version was in the beta development phase. Applix was testing the new version with several customers. For this version, Applix promised to use Windows NT services for several processes. Also, it was to use MS Windows NT 4.0's native 32 bit ODBC connections. Therefore, the new version 7.0 of Applix Helpdesk would have been better integrated into Windows NT 4.0, and most probably the problems encountered would have been solved with this new version.

However, UWSP's IT department decided to return the product. This was because of several reasons. First, the response of the company was very poor. Technical product support could not answer questions as described above. Also, the sales representative did not return calls regarding the new version 7.0 of Applix Helpdesk. Finally, promises made were broken. Applix, for instance, promised to send the new version in October 1997, but the product never arrived.

This caused a poor working relationship between Applix, Incorporated and UWSP. The most important reason for returning the software, however, was that Applix could not interface with HP OpenView for NT. It only interfaced with HP OpenView for Unix. Although the interface to HP OpenView was specified as a requirement (see Section 3.2), Applix could not resolve this problem. Because UWSP does not work with Unix, this situation was not acceptable and the product had to be returned. Therefore, another solution for the Helpdesk had to be found.

[9] A variant of HTTP (hypertext transfer protocol) that supports SSL (secure sockets layer). This allows for encrypted data transfer between browser and server.

## 4.4. The Re-Work of the Star System

After the Applix Helpdesk and its components were returned, the re-work the old STAR system and the conversion of it into a Windows application was initiated. As already described in Section 3.3, because the old STAR system had been implemented with Clarion for DOS, the new version of this platform, Clarion4, was to be used for implementing the new system.

Because of it's large scope, the project was devided into three smaller parts, with Jan Vogt, Dirk Bahle, and Scott Gile as the supervisor working on it as well, and this Praktikum concentrating on one of the three parts as it will be explained below.

### Data Model

The first step that had to be performed was the database design, which was approached as a group. Similar to the approach of Applix Helpdesk, the database for the new STAR system was to reside on a SQL server.

The major problem with this part of the project was that there was no database available containing employment data, personal data, and student data including phone numbers, offices, and other items all together. The workorder system, however, needed access to such a database to give the workers information on how and where the callers could be contacted. This is why the new database did not only have to include the workorder part of the project (STAR), but also the person part (Phonebook).

The central table of the STAR database is the workorder table (*WO*). For every workorder, one record in the workorder table is created. It holds information about the following (for reference see Figure 5):

- A uniquely identifying number for the workorder (*WO_ID*),
- the reporting person (*WO_For_ID*) and the the person entering the workorder (*WO_By_ID*), both being links to the phonebook database,
- the current status and priority of the workorder (*Status_ID* and *Priorities_ID*, respectively), being links to lookup tables *Status* and *Priorities*, respectively,
- the subcategory the problem is classified as (*SubCategory_ID*), linking to the lookup table *SubCategory*, which also contains information about the main category,
- a description of the problem devided into a searchable keywords field (*Keywords*) and a non-searchable description field (*WO_Text*),

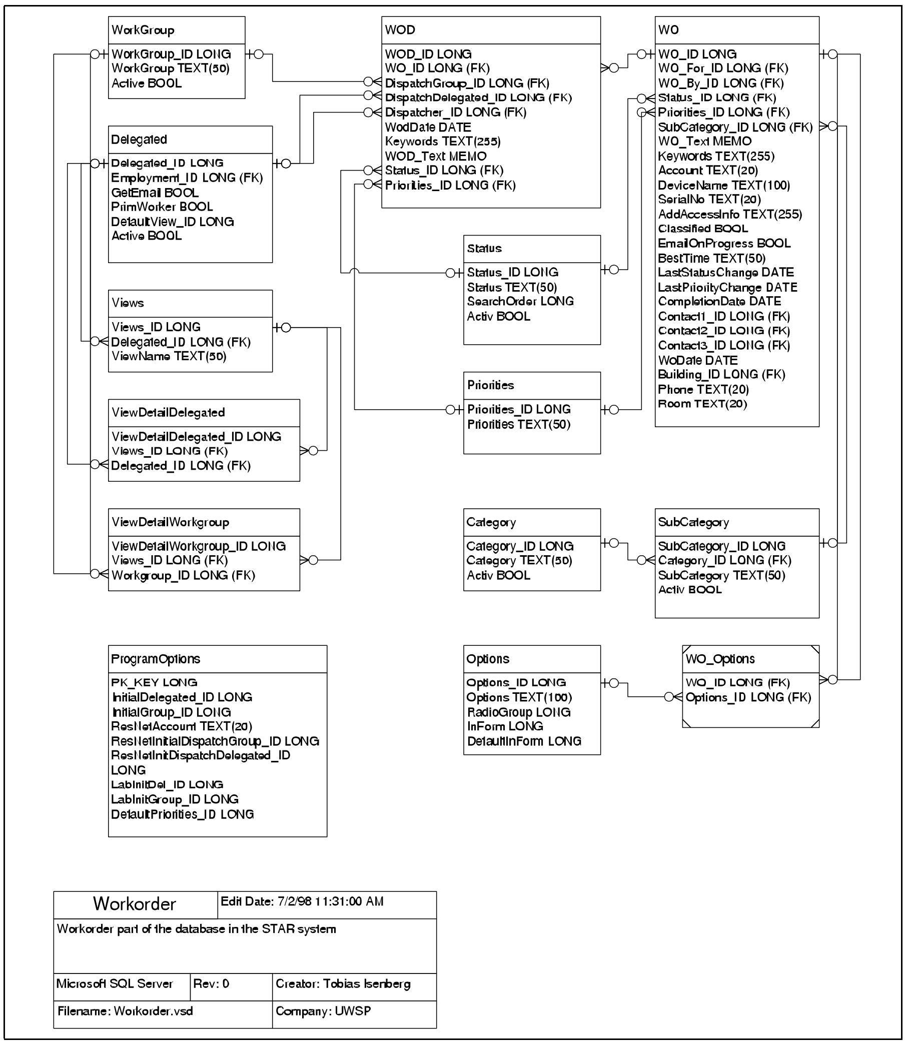


**Figure 5:** STAR: data model

- additional information as account number, device name, serial number, access information, classified status, e-mail request, the time that is best for contacting the person, building name, phone number, and room number (*Account*, *DeviceName*, *SerialNo*, *AddAccessInfo*, *Classified*, *EmailOnProgress*, *BestTime*, *Building*, *Phone*, *Room*, respectively),

- three contact people (*Contact1_ID*, *Contact2_ID*, and *Contact3_ID*), linking to the *Person* table in the phonebook database, and

- time stamps for the workorder creation date, the most recent status change, the most recent priority change, and the workorder completion date (*WoDate*, *LastStatusChange*, *LastPriorityChange*, and *CompletionDate*, respectively).

The second important item is the workorder detail. It represents one dispatching process. For every workorder record there must be at least on record in the workorder detail table (*WOD*). It contains information as follows:

- A uniquely identifying number for the workorder detail (*WOD_ID*),
- the workorder it is associated with (*WO_ID*),
- information about the people connected to the dispatching process: the workgroup the workorder is dispatched to, the delegated worker it is dispatched to, and the person who dispatches the workorder (*DispatchGroup_ID*, *DispatchDelegated_ID*, and *Dispatcher_ID*, respectively), linking to the related tables *WorkGroup*, *Dispatcher*, and *Dispatcher*, respectively,
- similar to the *WO* table, a description of the problem devided into a searchable keywords field (*Keywords*) and a non-searchable description field (*WOD_Text*), and
- a time stamp for the creation of the record (*WodDate*).

Every delegated worker is associated with a workgroup. These workgroups work in one specific field, for instance hardware or networks. Every workgroup has one primary worker who manages the work of the group. Since one employee can be a member of more than one workgroup, there is a different record in the *Delegated* table for every association to a workgroup. To be able to identify a delegated worker, the *Delegated* table contains a link to the *Employment* table, which is a part of the phonebook database. Primary workers are identified by a bit field in the *Delegated* table.

The *SubCategory* table, which is linked to the workorder table, contains a link to the *Category* table. Because of this the workorder is automatically assigned a category as well by assigning a subcategory to a workorder.

The table *WO_Options* associates certain options from the *Options* table with a workorder. Theses options are currently used to specify the operating system of the machine. However, for easily being able to add further options to the STAR system, the options are not hard-coded but kept in a table. There can be multiple options for a single workorder. This is only restricted by the configuration in the *Options* table. Multiple options can be grouped together in a radio button group (*RadioGroup* entry), from which only one can be selected at a time. Also, a default value for each radio button group is stored. The *InForm* entry is a bitmap coding the forms the entries are used in. This is only used in the Web forms part of the STAR system, the workorder module always allows the user to select from all options.

The *Views* table stores the views for the delegated workers. It allows every delegated worker to create personalized views on the whole workorder table customized to the special needs. Every delegated worker can have multiple views, one being the default view (stored in the *Delegated table*). Every view consists of multiple view details (*View Detail* table). These store workgroups and/or delegated workers that have to be included in the view. Therefore, a view is customizable without limitations.

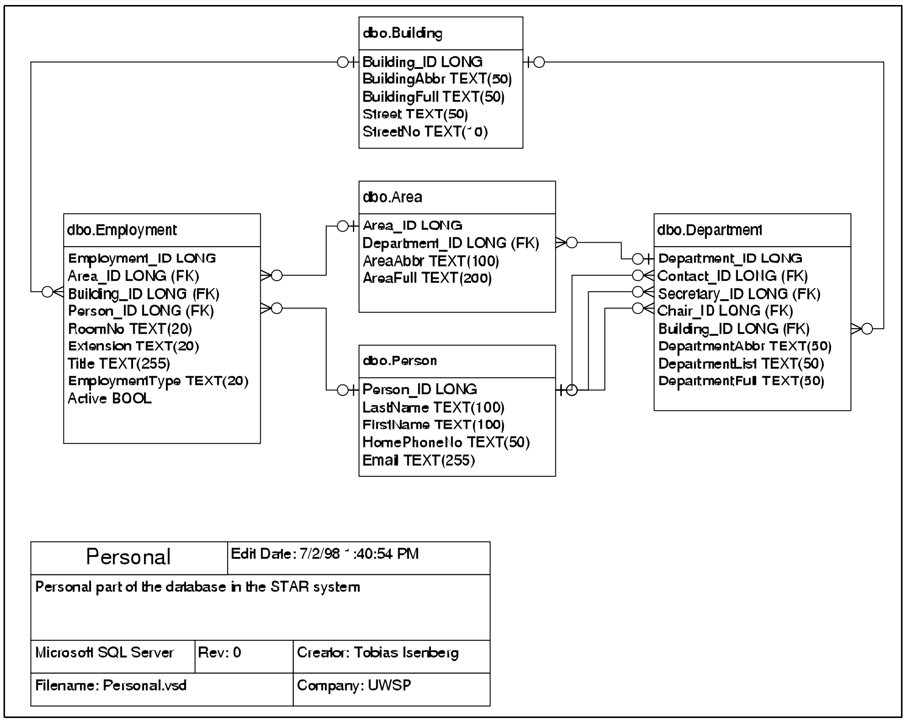


**Figure 6:** Phonebook: data model

On the phonebook side of the database (cf. Figure 6), the main two tables are the *Employment* table and the *Person* table. Since one person can have multiple employments with the university, this division into two tables was necessary. To link with the *Delegated table*, the *Employment_ID* field from the *Employment* table is used.

Furthermore, every record in the *Employment* table is associated with a department, an area within the department, and a building. The department has information about the chair person, the department's secretary, a contact person, and the main building it is located in.

This design of the phonebook database is currently used in the implementation. However, it is not the final design. Currently, the phonebook design is under discussion with the staff working on the phonebook and AIS[10], the department where

[10] Administrative Information Systems, the department at UWSP that hosts employment data as well as student data

this part of the database will finally reside. On the other hand, the final design will probably not differ very much from the current design, as the first meetings have indicated.

### Implementation using Clarion4

As already pointed out earlier, when discussing implementation specifics, this paper focuses on the worker module part of the STAR system.

As a first step in the implementation, the re-compiling of the old code from Clarion for DOS in Clarion4 was tried as described in Section 8. However, this turned out not to be possible. Both versions of the development tool differ in their design too much for just re-compiling the old project. Even the conversion wizard provided by Clarion4 was not of much help. Therefore, the project had to be completely re-developed. This does not necessarily have to be seen only in a negative way. With this approach, the implementation was free from the old code and it could make use of the new tool and its features without limitations.

So first the database description was imported into a Clarion4 data dictionary file using Clarion's ODBC driver. The data dictionary had to be used as the base of the implementation. Being forced to create another data dictionary and use it as a basis for the implementation was inefficient and a cause for problems. The same information was already available on the SQL server. Therefore, a second data dictionary is redundant data storage. But an implementation in Clarion4 without a second physical data dictionary is not possible.

Also, the import feature of Clarion was not very user friendly. It could only import the different tables from the SQL database separately. It also did not import any foreign keys or links between the different tables using foreign keys. Therefore, all foreign keys and the links between the tables had to be re-created in the Clarion data dictionary file.

After the database description was imported, the implementation of the worker module itself was started by implementing the basic features, like workorder queues and viewing, creating, and dispatching workorders.

For the main workorder list (workorder queue) a data control was used that is similar to a data grid. However, this data control did not react as it was supposed to. When clicking on a certain row, the same row was displayed again a few rows further down in the list. This could not be prevented from happening by changing the settings of the data control. It could not be determined whether these problems were caused by the data control or by Clarion4's ODBC driver.

One of the main reasons for deciding on Clarion4 as the develement platform has been it's ability to compile the source code to a Web application very easily as described in Section 8. However, it turned out that the internet compiling module for Clarion that came with Clarion4 was not designed to work with this version, but with a prior version. The next version of the internet compiling module, which would have been compatible with Clarion4, was about to be delivered. However, this version had problems as well. According to reports from other users, this version was producing outputs that were running very slow on browsers except of MS Internet Explorer. This would have been a problem since UWSP uses Netscape Communicator as a Web browser.

### Implementation using MS Visual Basic 5

The implementation of the worker module so far is in the phase of a prototype. It supports major functions, such as viewing workorders, entering new workorders, and dispatching workorders (see screenshots in Appendix A).

In order to access the data at the SQL server, Remote Data Objects are used to connect to the ODBC system data source. This system DSN contains the information about the connection (server name, database name, etc.). To be able to use the data types Remote Data Objects, the library *Microsoft Remote Data Object 2.0* (msrdo20.dll) had to be referenced.

During the startup, a *rdoConnection* is created. This *rdoConnection* is used in the program for creating queries on tables and views of the database, as well as for adding new records to the database. The advantage of using only one connection, instead of Remote Data Controls, is that there is only one location in the program code where the ODBC data source has to be specified. If Remote Data Controls would have been used, the DSN[11] would have to be specified for every control again. Besides that, Remote Data Objects are also faster than Remote Data Controls. Finally, Remote Data Objects do not require Data Bound Conrols to be used for entry fields. Therefore, new records do not have to be created when the window appears, but can be created at any time. This makes the use of auto-numbered fields in the database in connection with “Cancel” buttons much easier.

For querying data, *rdoQuery* and *rdoResultset* objects are used. For every query, a *rdoQuery* object is created from the one *rdoConnection* object that was created at startup time. These *rdoQuery* objects use SQL syntax to query the database and return a *rdoResultset* object. This *rdoResultset* object then contains the result of the query.

So far, four windows are implemented:

- the Main Menu, containing the workorder queue,
- the View Workorder window, displaying the workorder data and the data of the latest dispatch event,

[11]Data Source Name

- the Dispatch Workorder window, allowing to enter the data for a dispatch event, and
- the New Workorder window, allowing the creation of a new workorder.

### Outlook and Future Work

The next step in the implementation will be the filter option. It will allow every worker to apply filters to his/her queue. These filters could be, for instance, only the personal workorders, or only workorders dispatched to a certain workgroup, or a combination of both. Every worker will be able to create personal views, and will have a default view. Since this profile data is stored in the database, the worker will have his own view he/she is used to, regardless from which computer he/she loggs onto the system.

After the implementation of the filter option, two other features will have to be implemented along the way. These are the notification feature and the search option. The notification feature will allow for e-mail notification to workers at specific points in the process, users (the people for whom workorders were created), and the contact person(s).

Finally, the search option will allow for searching the keyword fields of the workorder table and the workorder detail table (dispatch events). This enables workers to look for similar problems that were solved in the past and use these notes to solve current problems.

# 5. Summary and Prospect

## Project Status

As of May 13$^{th}$, the status of the new STAR system is that the design phase is about done and the project is currently in its implementation phase. There is still a lot of coding to be done. However, a first prototype of the worker module supporting basic functionality is working and should be used as a starting point to finish this module. Additional features, such as the filter option, the search feature, and a notification part, have to be implemented. Also, the form module and the reporting module are not finished yet and have to be completed.

After the coding is finished, the system will have to be tested under working conditions with all modules working together. Further tasks include the data conversion of currently active workorders in the old STAR system to allow for a clean cut-over and the documentation of the system.

## Summary

From the beginning of the Praktikum until the end, the task changed very dramatically. In the beginning, a commercial software product had to be installed. At the end, the software needed at UWSP's helpdesk had to be developed and implemented. Because the coding part of the project only took a part of the time, the paper rather discusses the investigation process along the way than the actual coding specifics.

The paper described the process of the attempted installation of Applix Helpdesk and Applix Weblink, and the reasons that lead to the return of the software. Then, the implementation in Clarion4 and MS VisualBasic 5.0, and the design of the underlying database were explained.

Finally, the goal is in sight. The current deadline for the project is July 15$^{th}$. After this date, UWSP will have a new helpdesk system that implements the features of the old system and more. It will be better adapted to the current needs and the current operating system MS Windows 95. It will not have the functionality of a commercial product, but it implements more than the very basic functionality described in the beginning of this paper. It will be much more user friendly than the old system, and will also implement more features.

## Prospect

For the future of the helpdesk system at UWSP some new features would be nice to have. Three of them should be mentioned here.

First of all, it would be nice if the worker module would be available on the Web. This would enable workers to easily access their workorder queues from anywhere on campus or from home, as already described as an advantage in Section 8.

The next one was already specified in the system requirements for the commercial product, as shown in Section 3.2. This was the interface to the HP OpenView system that is running on campus. HP OpenView allows for measuring network performance and generating events when certain customizable conditions occur. In case certain events occur, such as a network failure, an interface to HP OpenView could automatically generate workorders in the STAR system. Also, when the conditions cease to exist, the workorders could automatically be closed.

A third feature would be a knowledge base that is integrated into the workorder system. Such a knwoledge base would extend the search option that is already planned for the STAR system.

# List of Abbreviations

| Abbreviation | Explination |
|---|---|
| AIS | Administrative Information Systems |
| ASP | Active Server Page |
| DBMS | Database Management Systems |
| DSN | Data Source Name |
| e-mail | Electronic Mail |
| GUI | Graphical User Interface |
| HP | Hewlett Packard Company |
| IIS | Internet Information Server |
| IT | Information Technology |
| MS | Microsoft Corporation |
| ODBC | Open Database Connectivity |
| SQL | Structured Query Language |
| STAR | Service Tracking And Reporting |
| UWSP | University of Wisconsin – Stevens Point |
| Web | World Wide Web |
| Windows NT | Windows New Technology |
| WWW | World Wide Web |

# List of Figures

## References

[App96a] Applix, Inc., Westboro, Massachusetts, USA. *Applix Administrator's Guide*, September 1996.

[App96b] Applix, Inc., Westboro, Massachusetts, USA. *Applix Enterprise Enhancement Guide - Version 6.0*, September 1996.

[App96c] Applix, Inc., Westboro, Massachusetts, USA. *Applix Helpdesk Tutorial*, September 1996.

[Bar92] Avron Barr. Software trends at the help desk. In *Intelligent Software Strategies*. Cutter Information Corp., Arlington, MA, September 1992.

[Mic98] Microsoft Corporation. *ODBC 3.0 – Overview*, April 1998. Web site: http://www.microsoft.com/products/prodref/264_ov.htm.

[MM96a] Rita C. Marcella and Iain A. Middleton. Key factors in help desk success: an analysis of areas critical to help desk development and functionality. Technical report, The British Library, 1996.

[MM96b] Rita C. Marchella and Iain A. Middleton. The role of the help desk in the strategic management of information systems. In *OCLC Systems & Services*, volume 12, pages 4–19. MCB University Press, 1996.

[Mon97] Monach Bay Software, Inc. *The Monach Bay Help Desk Handbook*, 1997.

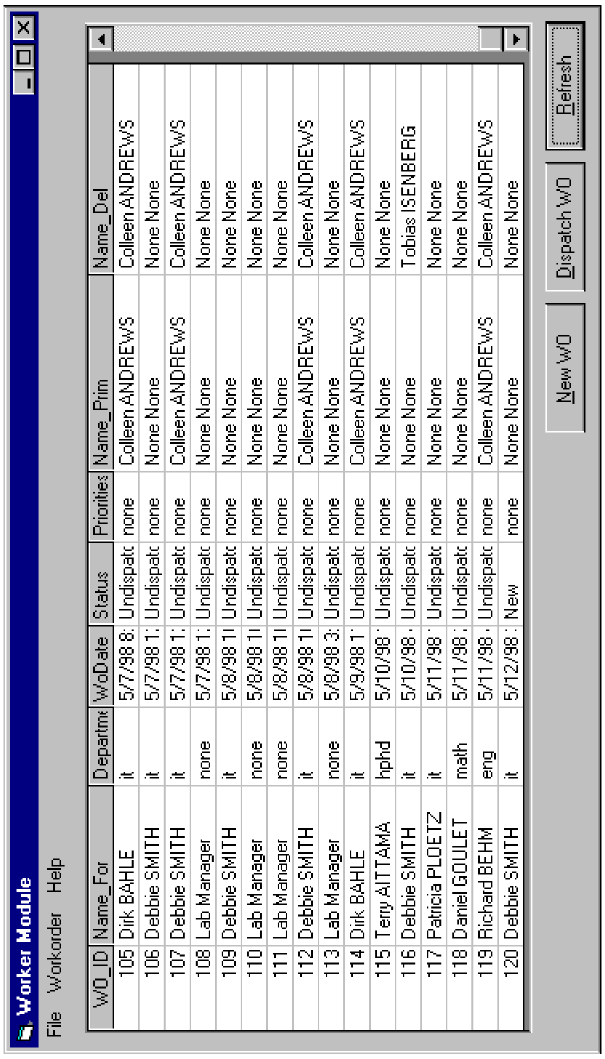
[Smi97] Debbie Smith. Help desk software for information technology search process documentation. Technical report, University of Wisconsin-Stevens Point, Information Technology, 13 August 1997.

[Uni97] University of South Australia, Information Technology Services HelpDesk, Adelaide, Australia. *HelpDesk Process Management*, August 1997. Version 2.1.

[Ver98] Philip Verghis. *Philip Verghis' Help Desk FAQ*, April 1998. Version 4.3. http://www.philverghis.com/helpdeskfaq.html.

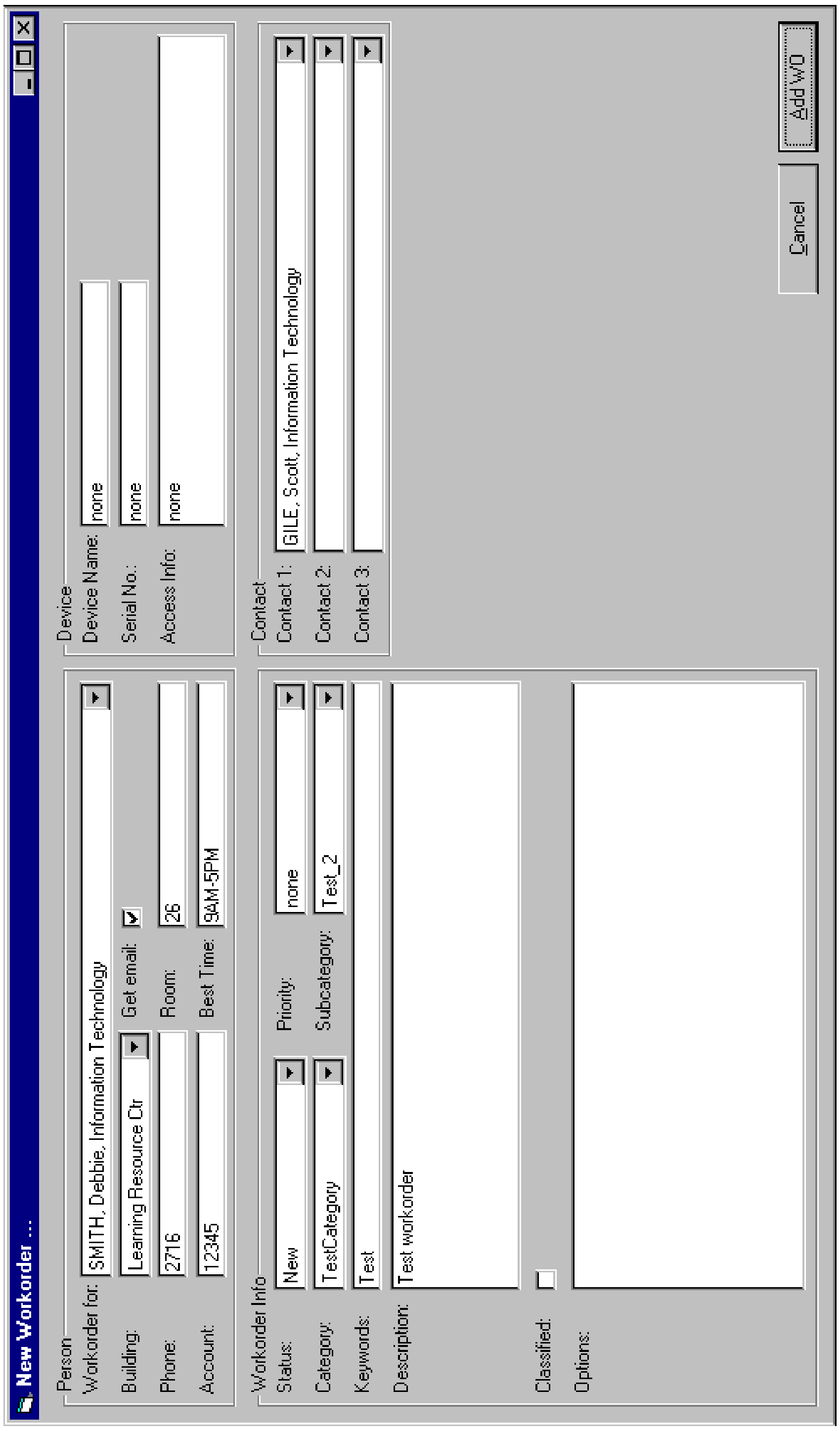
# A. Screenshots

## A.1. Main Menu



| WO_ID | Name_For | Departme | WoDate | Status | Priorities | Name_Prim | Name_Del |
|---|---|---|---|---|---|---|---|
| 105 | Dirk BAHLE | it | 5/7/98 8: | Undispatc | none | Colleen ANDREWS | Colleen ANDREWS |
| 106 | Debbie SMITH | it | 5/7/98 1: | Undispatc | none | None None | None None |
| 107 | Debbie SMITH | it | 5/7/98 1: | Undispatc | none | Colleen ANDREWS | Colleen ANDREWS |
| 108 | Lab Manager | none | 5/7/98 1: | Undispatc | none | None None | None None |
| 109 | Debbie SMITH | it | 5/8/98 1l | Undispatc | none | None None | None None |
| 110 | Lab Manager | none | 5/8/98 1l | Undispatc | none | None None | None None |
| 111 | Lab Manager | none | 5/8/98 1l | Undispatc | none | None None | None None |
| 112 | Debbie SMITH | it | 5/8/98 1l | Undispatc | none | Colleen ANDREWS | Colleen ANDREWS |
| 113 | Lab Manager | none | 5/8/98 3: | Undispatc | none | None None | None None |
| 114 | Dirk BAHLE | it | 5/9/98 1' | Undispatc | none | Colleen ANDREWS | Colleen ANDREWS |
| 115 | Terry AITTAMA | hphd | 5/10/98 ' | Undispatc | none | None None | None None |
| 116 | Debbie SMITH | it | 5/10/98 . | Undispatc | none | None None | Tobias ISENBERG |
| 117 | Patricia PLOETZ | it | 5/11/98 ' | Undispatc | none | None None | None None |
| 118 | Daniel GOULET | math | 5/11/98 : | Undispatc | none | None None | None None |
| 119 | Richard BEHM | eng | 5/11/98 . | Undispatc | none | Colleen ANDREWS | Colleen ANDREWS |
| 120 | Debbie SMITH | it | 5/12/98 : | New | none | None None | None None |

## A.2. New Workorder Window



Person

Workorder for: SMITH, Debbie, Information Technology

Building: Learning Resource Ctr | Get email: ✓

Phone: 2716 | Room: 26

Account: 12345 | Best Time: 9AM-5PM

Device

Device Name: none

Serial No.: none

Access Info: none

Workorder Info

Status: New | Priority: none

Category: TestCategory | Subcategory: Test_2

Keywords: Test

Description: Test workorder

Classified: ☐

Options:

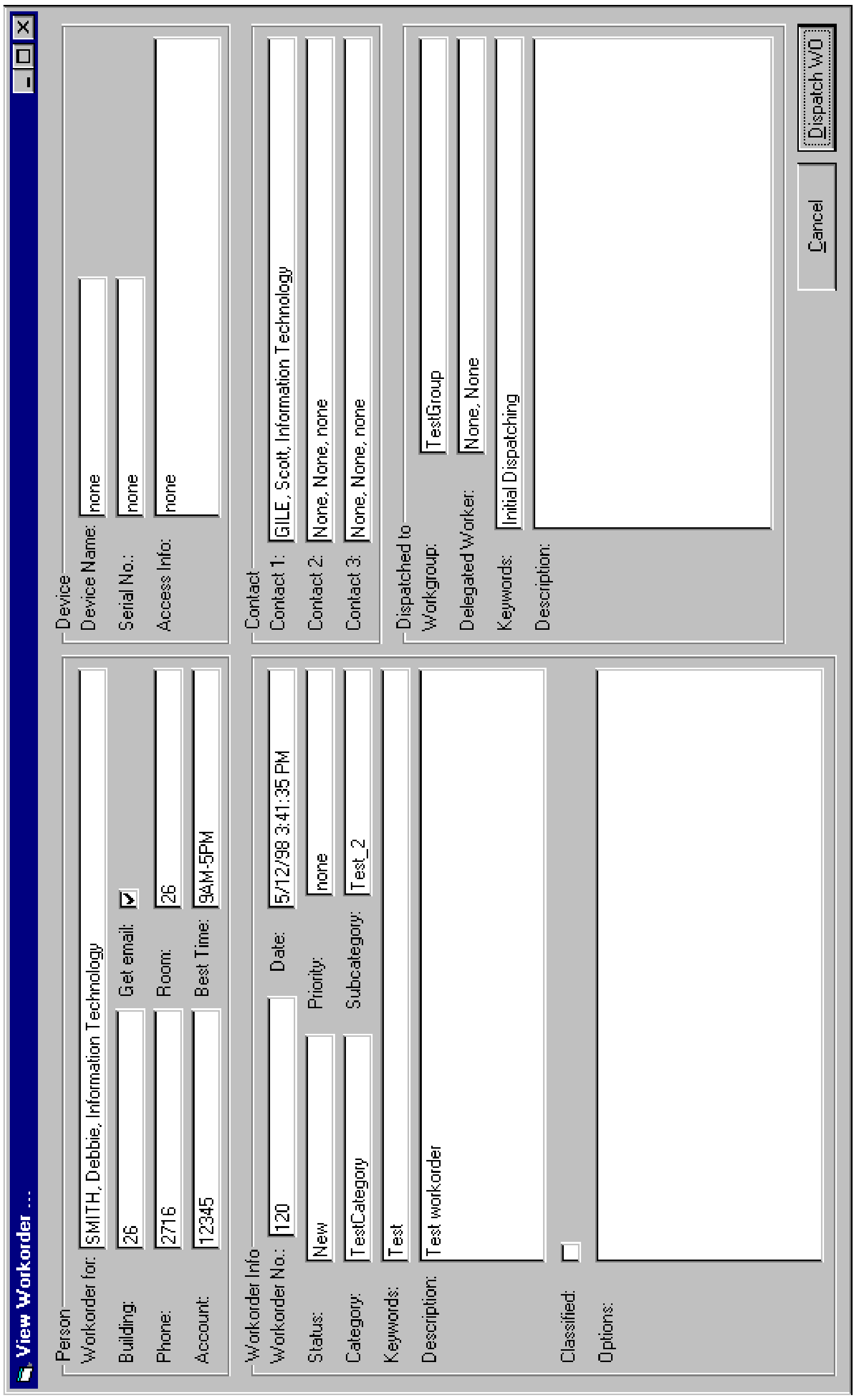
Contact

Contact 1: GILE, Scott, Information Technology

Contact 2:

Contact 3:

## A.3. View Workorder Window



Person

| | | | |
|---|---|---|---|
| Workorder for: | SMITH, Debbie, Information Technology | | |
| Building: | 26 | Get email: | ✓ |
| Phone: | 2716 | Room: | 26 |
| Account: | 12345 | Best Time: | 9AM-5PM |

Device

| | |
|---|---|
| Device Name: | none |
| Serial No.: | none |
| Access Info: | none |

Workorder Info

| | | | |
|---|---|---|---|
| Workorder No.: | 120 | Date: | 5/12/98 3:41:35 PM |
| Status: | New | Priority: | none |
| Category: | TestCategory | Subcategory: | Test_2 |
| Keywords: | Test | | |
| Description: | Test workorder | | |
| Classified: | ☐ | | |
| Options: | | | |

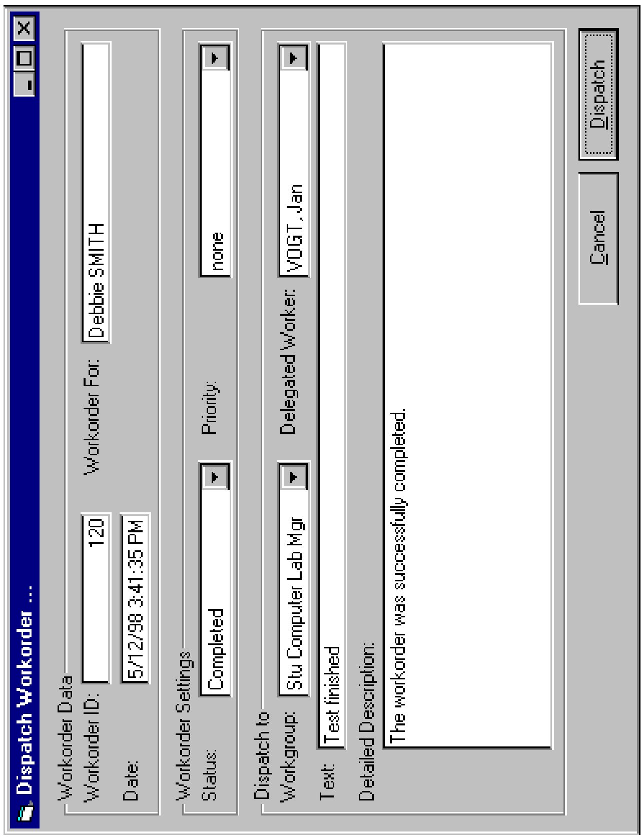
Contact

| | |
|---|---|
| Contact 1: | GILE, Scott, Information Technology |
| Contact 2: | None, None, none |
| Contact 3: | None, None, none |

Dispatched to

| | |
|---|---|
| Workgroup: | TestGroup |
| Delegated Worker: | None, None |
| Keywords: | Initial Dispatching |
| Description: | |

## A.4. Dispatch Workorder Window


Dispatch Workorder ...

Workorder Data
Workorder ID: 120
Workorder For: Debbie SMITH
Date: 5/12/98 3:41:35 PM

Workorder Settings
Status: Completed
Priority: none

Dispatch to
Workgroup: Stu Computer Lab Mgr
Delegated Worker: VOGT, Jan
Text: Test finished
Detailed Description:
The workorder was successfully completed.

# B. MS VisualBasic 5.0 Code

## B.1. Main.bas

```
Private Sub cmdDispatchWo_Click()
    DispatchWoForm.Show
End Sub

Private Sub cmdNewWo_Click()
    NewWoForm.Show
End Sub

Private Sub cmdRefresh_Click()
    On Error GoTo RefErr

    WorkorderDataControl.Refresh
    Exit Sub

RefErr:
    MsgBox "Error:" & Err & " " & Err.Description
End Sub

Private Sub Exit_Click()
    End
End Sub

Private Sub Form_GotFocus()
    Me.WorkorderGrid.Refresh
End Sub

Private Sub Form_Load()
    Dim result As Boolean
    result = InitializeConnection() ' Initialize the
                                    ' connection
End Sub

Private Sub Form_Unload(Cancel As Integer)
    conOdbcConStar.Close
    conOdbcConPersonal.Close
End Sub

Private Sub WorkorderGrid_DblClick()
```

```
    ViewWoForm.Show
End Sub
```

## B.2. NewWoForm.bas

```
Private Sub Category_GotFocus() ' Switch on the timer
    CategoryTimer.Interval = 1
    Me.CategoryTimer.Enabled = True
End Sub

Private Sub Category_LostFocus() ' Switch off the timer
    Me.CategoryTimer.Enabled = False
End Sub

Private Sub CategoryTimer_Timer()
    Dim Status As Boolean

    Status = Me.Category.DataChanged
    If Status Then
        ' DataHasChangedEvent:
        RefreshSubcategoryList
        ' End DataHasChangedEvent
        Me.Category.DataChanged = False
    End If
End Sub

Private Sub cmdCancel_Click() ' Cancel
    Unload Me
End Sub

Private Sub cmdNewWo_Click()
    Dim resWO, resWOD, resDispatcher, resOptions, _
         resLastWo As rdoResultset
    Dim name, response
    Dim dispatcher As Integer

    name = GetUserName()    ' Get the user name

    Set resWO = DoQuery("select * from WO")
    Set resWOD = DoQuery("select * from WOD")
    Set resOptions = _
         DoQuery("select * from ProgramOptions")
```

```
Set resDispatcher = DoQuery("select * from " & _
    "View_Delegated " & _
    "where Email = '" & name & "'")

If IsNull(resDispatcher("Delegated_ID")) Then
    ' Check if dispatching allowed
    response = MsgBox("You are not allowed to " & _
        "enter workorders.", vbOKOnly, "Title")
    Unload Me
    Exit Sub
Else
    dispatcher = CLng(resDispatcher("Employment_ID"))
End If

' Add the new workorder to the database
resWO.AddNew
resWO("WO_For_ID") = _
   Me.WoFor.ItemData(Me.WoFor.ListIndex)
resWO("WO_By_ID") = dispatcher
resWO("Status_ID") = _
   Me.Status.ItemData(Me.Status.ListIndex)
resWO("Priorities_ID") = _
   Me.Priority.ItemData(Me.Priority.ListIndex)
resWO("Subcategory_ID") = _
   Me.SubCategory.ItemData(Me.SubCategory.ListIndex)
resWO("WO_Text") = Me.Description.Text
resWO("Keywords") = Me.Keywords.Text
resWO("Account") = Me.Account.Text
resWO("DeviceName") = Me.DeviceName.Text
resWO("SerialNo") = Me.SerialNo.Text
resWO("AddAccessInfo") = Me.AccessInfo.Text
resWO("Classified") = Me.Classified
resWO("BestTime") = Me.BestTime.Text
resWO("LastStatusChange") = Now
resWO("LastPriorityChange") = Now
resWO("Contact1_ID") = 1
resWO("Contact2_ID") = 1
resWO("Contact3_ID") = 1
If Me.Contact1.ListIndex <> -1 _
    Then resWO("Contact1_ID") = _
    Me.Contact1.ItemData(Me.Contact1.ListIndex)
If Me.Contact2.ListIndex <> -1 _
```

```
        Then resWO("Contact2_ID") = _
        Me.Contact2.ItemData(Me.Contact2.ListIndex)
    If Me.Contact3.ListIndex <> -1 _
        Then resWO("Contact3_ID") = _
        Me.Contact3.ItemData(Me.Contact3.ListIndex)
    resWO("EmailOnProgress") = Me.GetEmail
    resWO("Building") = _
         Me.Building.ItemData(Me.Building.ListIndex)
    resWO("Phone") = Me.Phone.Text
    resWO("Room") = Me.Room.Text
    resWO("WoDate") = Now
    resWO.Update

    Set resLastWo = DoQuery("Select WO_ID from WO " _
        & "where (WoDate in (Select Max(WoDate) ‘‘ _
        & "From WO Where WO_By_ID = " & dispatcher _
        & " Group By WO_By_ID) and (wo_by_id = " _
        & dispatcher & "))")

    resWOD.AddNew
    resWOD("DispatchDelegated_ID") = _
       resOptions("InitialDelegated_ID")
    resWOD("DispatchGroup_ID") = _
       resOptions("InitialGroup_ID")
    resWOD("Keywords") = "Initial Dispatching"
    resWOD("WOD_Text") = "Initial Dispatching"
    resWOD("WodDate") = Now
    resWOD("Dispatcher_ID") = _
       resDispatcher("Delegated_ID")
    resWOD("WO_ID") = resLastWo("WO_ID") ’ per maximum &
                                         ’ eintragender
    resWOD.Update
    Unload Me
End Sub

Private Sub RefreshSubcategoryList() ’ Refresh the
                                     ’ Subcategory list
    Dim resSubCategory As rdoResultset

    Me.SubCategory.Clear

    Set resSubCategory = DoQuery("select * from " & _
        "SubCategory where Category_ID = " & _
        Me.Category.ItemData(Me.Category.ListIndex))

    resSubCategory.MoveFirst
    While resSubCategory.EOF = False
        Me.SubCategory.AddItem _
           resSubCategory("SubCategory")
        Me.SubCategory.ItemData( _
          Me.SubCategory.NewIndex) _
            = resSubCategory("Subcategory_ID")
        resSubCategory.MoveNext
    Wend

    Me.SubCategory.ListIndex = 0
End Sub

Private Sub Form_Load()
    Dim resWoFor, resContact1, resContact2, _
       resContact3 As rdoResultset
    Dim resBuilding, resSubCategory, _
       resCategory As rdoResultset
    Dim resPriority, resStatus As rdoResultset

    Main.Enabled = False

    ' Set up all the dropdown box lists
    Set resWoFor = DoQuery("select * from " & _
       "View_Employment order by Name")
    Set resSubCategory = DoQuery("select * from " & _
       "Subcategory")
    Set resCategory = DoQuery("select * from Category")
    Set resPriority = DoQuery("select * from Priorities")
    Set resStatus = DoQuery("select * from Status")
    Set resBuilding = DoQueryPersonal("select * from " _
       & "Building")

    resWoFor.MoveFirst
    While resWoFor.EOF = False
        Me.WoFor.AddItem resWoFor("Name")
        Me.WoFor.ItemData(Me.WoFor.NewIndex) = _
           resWoFor("Employment_ID")
        Me.Contact1.AddItem resWoFor("Name")
        Me.Contact1.ItemData(Me.Contact1.NewIndex) = _
           resWoFor("Employment_ID")
        Me.Contact2.AddItem resWoFor("Name")
        Me.Contact2.ItemData(Me.Contact2.NewIndex) = _
           resWoFor("Employment_ID")
        Me.Contact3.AddItem resWoFor("Name")
        Me.Contact3.ItemData(Me.Contact3.NewIndex) = _
           resWoFor("Employment_ID")
        resWoFor.MoveNext
    Wend

    resCategory.MoveFirst
    While resCategory.EOF = False
        Me.Category.AddItem resCategory("Category")
        Me.Category.ItemData(Me.Category.NewIndex) = _
            resCategory("Category_ID")
        resCategory.MoveNext
    Wend

    resPriority.MoveFirst
    While resPriority.EOF = False
        Me.Priority.AddItem resPriority("Priorities")
        Me.Priority.ItemData(Me.Priority.NewIndex) = _
            resPriority("Priorities_ID")
        resPriority.MoveNext
    Wend

    resStatus.MoveFirst
    While resStatus.EOF = False
        Me.Status.AddItem resStatus("Status")
        Me.Status.ItemData(Me.Status.NewIndex) = _
            resStatus("Status_ID")
        resStatus.MoveNext
    Wend

    resBuilding.MoveFirst
    While resBuilding.EOF = False
        Me.Building.AddItem resBuilding("BuildingFull")
        Me.Building.ItemData(Me.Building.NewIndex) = _
            resBuilding("Building_ID")
        resBuilding.MoveNext
    Wend
```

```
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Main.Enabled = True
End Sub

Private Sub WoFor_GotFocus() ' Switch on the timer
    WoForTimer.Interval = 1
    Me.WoForTimer.Enabled = True
End Sub

Private Sub WoFor_LostFocus() ' Switch off the timer
    Me.WoForTimer.Enabled = False
End Sub

Private Sub WoForTimer_Timer()
    Dim resWoFor As rdoResultset
    Dim Status As Boolean

    Status = Me.WoFor.DataChanged
    If Status Then
        ' DataHasChangedEvent:
            Set resWoFor = DoQuery("SELECT * FROM " _
                & "View_Employment WHERE " & _
                "Employment_ID = " & _
                Me.WoFor.ItemData(Me.WoFor.ListIndex))
            If Not IsNull(resWoFor("RoomNo")) _
                Then Me.Room.Text = _
                resWoFor("RoomNo")
            If Not IsNull(resWoFor("Phone")) _
                Then Me.Phone.Text = _
                resWoFor("Phone")
            If Not IsNull(resWoFor("BuildingFull")) _
                Then Me.Building.Text = _
                resWoFor("BuildingFull")
        ' End DataHasChangedEvent
        Me.WoFor.DataChanged = False
    End If
End Sub
```

## B.3. ViewWoForm.bas

```
Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdDispatchWo_Click()
    Unload Me
    DispatchWoForm.Show
End Sub

Private Sub Form_Load()
    Dim resWO, resWOD, resWoFor, resContact1, _
       resContact2 As rdoResultset
    Dim resContact3, resSubCategory, _
       resCategory As rdoResultset
    Dim resPriority, resStatus, resWorkgroup, _
       resDelegated As rdoResultset
    Dim wo_text, wod_text As String

    'On Error GoTo RefErr

    Main.Enabled = False

    Set resWO = DoQuery("select * from WO " _
       & "where WO_ID = " & _
       Main.WorkorderGrid.TextMatrix( _
         Main.WorkorderGrid.Row, 0))
    Set resWOD = DoQuery("select * from " _
       & "View_WOD_Last where WO_ID = " & _
       resWO("WO_ID"))
    Set resWoFor = DoQuery("select * from " _
       & "View_Employment " & _
       "where Employment_ID = " & resWO("WO_For_ID"))
    Set resContact1 = DoQuery("select * from " _
       & "View_Employment " & _
       "where Employment_ID = " & resWO("Contact1_ID"))
    Set resContact2 = DoQuery("select * from " _
       & "View_Employment " & _
       "where Employment_ID = " & resWO("Contact2_ID"))
    Set resContact3 = DoQuery("select * from " _
       & "View_Employment " & _
```

```
      "where Employment_ID = " & resWO("Contact3_ID"))
   Set resSubCategory = DoQuery("select * from " _
      & "Subcategory " & _
      "where Subcategory_ID = " & _
      resWO("Subcategory_ID"))
   Set resCategory = DoQuery("select * from " _
      & "Category where " & _
      "Category_ID = " & resSubCategory("Category_ID"))
   Set resPriority = DoQuery("select * from " _
      & "Priorities " & _
      "where Priorities_ID = " & resWO("Priorities_ID"))
   Set resStatus = DoQuery("select * from " _
      & "Status where " & _
      "Status_ID = " & resWO("Status_ID"))
   Set resWorkgroup = DoQuery("select * from " _
      & "WorkGroup " & _
      "where WorkGroup_ID = " & _
      resWOD("DispatchGroup_ID"))
   Set resDelegated = DoQuery("select * from " _
      & "View_Delegated " & _
      "where Delegated_ID = " & _
      resWOD("DispatchDelegated_ID"))

   Me.WoNumber.Text = resWO("WO_ID")
   Me.WorkorderFor.Text = resWoFor("Name")
   Me.Account.Text = resWO("Account")
   Me.BestTime.Text = resWO("BestTime")
   Me.DeviceName.Text = resWO("DeviceName")
   Me.SerialNo.Text = resWO("SerialNo")
   Me.AccessInfo.Text = resWO("AddAccessInfo")
   Me.Building.Text = resWO("Building")
   Me.Phone.Text = resWO("Phone")
   Me.Room.Text = resWO("Room")
   Me.Status.Text = resStatus("Status")
   Me.Priority.Text = resPriority("Priorities")
   Me.Category.Text = resCategory("Category")
   Me.SubCategory.Text = resSubCategory("SubCategory")
   Me.Date.Text = resWO("WoDate")
   If resWO("Classified") Then Me.Classified.Value = 1
   If resWO("EmailOnProgress") _
      Then Me.GetEmail.Value = 1
   Me.Keywords.Text = resWO("Keywords")
```

```
    wo_text = resWO("WO_Text")
    If Not IsNull(wo_text) _
       Then Me.Description.Text = wo_text
    Me.Contact1.Text = resContact1("Name")
    Me.Contact2.Text = resContact2("Name")
    Me.Contact3.Text = resContact3("Name")
    Me.Workgroup.Text = resWorkgroup("WorkGroup")
    Me.Delegated.Text = resDelegated("Name")
    Me.Text.Text = resWOD("Keywords")
    'wod_text = resWOD("WOD_Text") ' Doesn't work
                                  ' - why???
    'If Not IsNull(wod_text) _
       Then Me.Detail.Text = wod_text

Form_Load_Exit:
    Exit Sub

RefErr:
    Dim er As rdoError
       Debug.Print Err, Error
       For Each er In rdoErrors
           Debug.Print er.Description, er.Number
        Next er
       Resume Next
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Main.Enabled = True
End Sub

Private Sub Frame3_DragDrop(Source As Control, _
   X As Single, Y As Single)

End Sub
```

## B.4. DispatchWoForm.bas

```
Private statusValue, priorityValue As Integer

Private Sub cmdCancel_Click()
    Unload Me
End Sub
```

```
Private Sub cmdDispatch_Click()
    Dim response As Integer
    Dim resDispatcher, resWOD, resWO As rdoResultset

    ' Get the Dispatcher's Delegated_ID
    Set resDispatcher = DoQuery("select * from " _
        & "View_Delegated " & _
        "where Email = '" & GetUserName & "'")

    ' Check if dispatching allowed
    If resDispatcher("Delegated_ID") = "" Then
        response = MsgBox("You are not allowed to " _
            & "dispatch " & _
            "workorders.", vbOKOnly, "Title")
        Unload Me
        Exit Sub
    End If

    Set resWOD = DoQuery("select * from WOD")
    Set resWO = DoQuery("select * from WO " _
        & "where WO_ID = " & _
        Me.WoIdField.Text)

    resWOD.AddNew ' Add the record to the database
    resWOD("DispatchDelegated_ID") = _
        Me.Delegated.ItemData(Me.Delegated.ListIndex)
    resWOD("DispatchGroup_ID") = _
        Me.Workgroup.ItemData(Me.Workgroup.ListIndex)
    resWOD("Dispatcher_ID") = _
        resDispatcher("Delegated_ID")
    resWOD("Keywords") = Me.Text.Text
    resWOD("WOD_Text") = Me.Detail.Text
    resWOD("WodDate") = Now
    resWOD("WO_ID") = Me.WoIdField.Text
    resWOD.Update

    resWO.Edit
    resWO("Priorities_ID") = _
       Me.Priority.ItemData(Me.Priority.ListIndex)
    resWO("Status_ID") = _
       Me.Status.ItemData(Me.Status.ListIndex)
    If resWO("Status_ID") <> statusValue Then _
        resWO("LastStatusChange") = Now
    If resWO("Priorities_ID") <> priorityValue Then _
        resWO("LastPriorityChange") = Now

    resWO.Update

    Unload Me ' Exit form
End Sub

Private Sub Form_Load()
    Dim resWO, resWorkgroup, resDelegated As rdoResultset
    Dim resPriority, resStatus As rdoResultset

    Main.Enabled = False
    Me.WoIdField.Text = _
        Main.WorkorderGrid.TextMatrix( _
          Main.WorkorderGrid.Row, 0)
    Me.WoForField.Text = _
        Main.WorkorderGrid.TextMatrix( _
          Main.WorkorderGrid.Row, 1)

    Set resPriority = DoQuery("select * from Priorities")
    Set resStatus = DoQuery("select * from Status")

    Set resWO = DoQuery("SELECT * FROM WO WHERE " & _
        "WO_ID = " & Me.WoIdField.Text)
    If Not IsNull(resWO("WoDate")) Then Me.Date.Text = _
        resWO("WoDate")

    Set resWorkgroup = DoQuery("select * from " _
       & "WorkGroup order by Workgroup")
    resWorkgroup.MoveFirst
    While resWorkgroup.EOF = False
        Me.Workgroup.AddItem resWorkgroup("WorkGroup")
        Me.Workgroup.ItemData(Me.Workgroup.NewIndex) = _
            resWorkgroup("WorkGroup_ID")
        resWorkgroup.MoveNext
    Wend

    Set resDelegated = _
```

```
       DoQuery("select * from View_Delegated")
    resDelegated.MoveFirst
    While resDelegated.EOF = False
        Me.Delegated.AddItem resDelegated("Name")
        Me.Delegated.ItemData(Me.Delegated.NewIndex) = _
            resDelegated("Delegated_ID")
        resDelegated.MoveNext
    Wend

    resPriority.MoveFirst
    While resPriority.EOF = False
        Me.Priority.AddItem resPriority("Priorities")
        Me.Priority.ItemData(Me.Priority.NewIndex) = _
            resPriority("Priorities_ID")
        If resPriority("Priorities_ID") = _
            resWO("Priorities_ID") Then _
            priorityValue = Me.Priority.NewIndex
        resPriority.MoveNext
    Wend
    Me.Priority.ListIndex = priorityValue
    priorityValue = resWO("Priorities_ID")

    resStatus.MoveFirst
    While resStatus.EOF = False
        Me.Status.AddItem resStatus("Status")
        Me.Status.ItemData(Me.Status.NewIndex) = _
            resStatus("Status_ID")
        If resStatus("Status_ID") = _
            resWO("Status_ID") Then _
            statusValue = Me.Status.NewIndex
        resStatus.MoveNext
    Wend
    Me.Status.ListIndex = statusValue
    statusValue = resWO("Status_ID")
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Main.WorkorderGrid.Refresh
    Main.Enabled = True
End Sub
```

## B.5. GeneralCode.bas

```
Public conOdbcConStar, conOdbcConPersonal _
    As rdoConnection

Private Declare Function _
    GetUserName_ Lib "advapi32.dll" _
    Alias "GetUserNameA" _
    (ByVal lpBuffer As String, nSize As Long) _
    As Long

Public Function GetUserName() As String
     Dim lpBuff As String * 25
     Dim ret As Long

     ' Get the user name minus any trailing
     ' spaces found in the name.
     ret = GetUserName_(lpBuff, 25)
     UserName = Left(lpBuff, InStr(lpBuff, Chr(0)) - 1)

     GetUserName = UserName
End Function

Public Function InitializeConnection()
    Dim cn As New rdoConnection
    Dim cn2 As New rdoConnection

    cn.Connect = "uid=;pwd=;server=ITHELP;" _
        & "driver={SQL Server};database=Star;" _
        & "DSN=Star;"

    cn.CursorDriver = rdUseOdbc
    cn.EstablishConnection rdDriverNoPrompt

    Set conOdbcConStar = cn

    cn2.Connect = "uid=;pwd=;server=ITHELP;" _
        & "driver={SQL Server};database=Personal;" _
        & "DSN=Personal;"

    cn2.CursorDriver = rdUseOdbc
    cn2.EstablishConnection rdDriverNoPrompt
    Set conOdbcConPersonal = cn2
End Function

Public Function DoQuery(strSQL As String) As rdoResultset
    Dim rs As rdoResultset

    ' Query the database
    Set rs = conOdbcConStar.OpenResultset(strSQL, _
        rdOpenStatic, _
        rdConcurValues, rdAsyncEnable + rdExecDirect)
    ' Wait for query to finish
    While rs.StillExecuting
        DoEvents
    Wend

    Set DoQuery = rs
End Function

Public Function DoQueryPersonal(strSQL As _
    String) As rdoResultset
    Dim rs As rdoResultset

    ' Query the database
    Set rs = conOdbcConPersonal.OpenResultset(strSQL, _
        rdOpenStatic, _
        rdConcurValues, rdAsyncEnable + rdExecDirect)
    ' Wait for query to finish
    While rs.StillExecuting
        DoEvents
    Wend

    Set DoQueryPersonal = rs
End Function
```

# C. Views on the SQL Database and Stored Procedures

## C.1. View_Delegated

*View_Delegated* connects the *Delegated_ID* with information from the *Person* table.

```
CREATE VIEW View_Delegated AS
SELECT
  Delegated.Delegated_ID,
  Delegated.Employment_ID,
  Person.LastName + ", " + Person.FirstName AS Name,
  Person.Email AS Email
FROM
  (Delegated
  INNER JOIN Employment ON Delegated.Employment_ID =
    Employment.Employment_ID)
  INNER JOIN Person ON Employment.Person_ID =
    Person.Person_ID
```

## C.2. View_Employment – STAR database

*View_Employment* connects the *Employment_ID* with information from the *Person* table. It refers to a temporary table created with the view *View_Employment* and the stored procedure *SP_TempPersonal* in the *Phonebook* database (refer to Section C.3 and Section C.4).

```
CREATE VIEW View_Employment AS
SELECT
  Employment_ID,
  LastName + ", " + FirstName + ", " +
    DepartmentFull AS Name,
  Department_ID,
  RoomNo,
  Extension As Phone,
  PersBuildingFull As BuildingFull
FROM
  Personal.dbo.TempPersonal
```

## C.3. View_Employment – Phonebook database

```
CREATE VIEW View_Employment AS
SELECT
  Employment.Employment_ID,
  Employment.Person_ID,
  Employment.RoomNo,
  Employment.Extension,
  Employment.Title,
  Employment.EmploymentType,
  Employment.Active,
  Person.LastName,
  Person.FirstName,
  Person.MiddleInitial,
  Person.HomePhoneNo,
  Person.Email,
  Department.Department_ID,
  Department.DepartmentAbbr,
  Department.DepartmentList,
  Department.DepartmentFull,
  Area.AreaAbbr,
  Area.AreaFull,
  Building_Pers.Building_ID AS PersBuilding_ID,
  Building_Pers.BuildingAbbr AS PersBuildingAbbr,
  Building_Pers.BuildingFull AS PersBuildingFull,
  Building_Pers.Street AS PersStreet,
  Building_Pers.StreetNo AS PersStreetNo,
  Building_Dep.Building_ID AS DepBuilding_ID,
  Building_Dep.BuildingAbbr AS DepBuildingAbbr,
  Building_Dep.BuildingFull AS DepBuildingFull,
  Building_Dep.Street AS DepStreet,
  Building_Dep.StreetNo AS DepStreetNo
FROM
  ((((Employment
  INNER JOIN Person ON Employment.Person_ID =
    Person.Person_ID)
  INNER JOIN Building AS Building_Pers ON
    Employment.Building_ID = Building_Pers.Building_ID)
  INNER JOIN Area ON Employment.Area_ID = Area.Area_ID)
  INNER JOIN Department ON Area.Department_ID =
    Department.Department_ID)
  INNER JOIN Building AS Building_Dep ON
```

```
    Department.Building_ID = Building_Dep.Building_ID
```

## C.4. SP_TempPersonal

*SP_TempPersonal* creates the temporary table *TempPersonal* using the view *View_Employment*.

```
CREATE PROCEDURE SP_TempPersonal AS

/* Drop temporary table */
DROP TABLE TempPersonal

/* Re-create temporaty table */
CREATE TABLE TempPersonal
(
  Employment_ID int,
  Person_ID int,
  RoomNo varchar(20) NULL,
  Extension varchar(20) NULL,
  Title varchar(255) NULL,
  EmploymentType varchar(20) NULL,
  Active bit,
  LastName varchar(100) NULL,
  FirstName varchar(100) NULL,
  MiddleInitial varchar(1) NULL,
  HomePhoneNo varchar(50) NULL,
  Email varchar(255) NULL,
  Department_ID int,
  DepartmentAbbr varchar(50) NULL,
  DepartmentList varchar(50) NULL,
  DepartmentFull varchar(50) NULL,
  AreaAbbr varchar(50) NULL,
  AreaFull varchar(100) NULL,
  PersBuilding_ID int NULL,
  PersBuildingAbbr varchar(50) NULL,
  PersBuildingFull varchar(50) NULL,
  PersStreet varchar(50) NULL,
  PersStreetNo varchar(10) NULL,
  DepBuilding_ID int NULL,
  DepBuildingAbbr varchar(50) NULL,
  DepBuildingFull varchar(50) NULL,
  DepStreet varchar(50) NULL,
  DepStreetNo varchar(10) NULL
)

INSERT INTO TempPersonal
(
  Employment_ID,
  Person_ID,
  RoomNo,
  Extension,
  Title,
  EmploymentType,
  Active,
  LastName,
  FirstName,
  MiddleInitial,
  HomePhoneNo,
  Email,
  Department_ID,
  DepartmentAbbr,
  DepartmentList,
  DepartmentFull,
  AreaAbbr,
  AreaFull,
  PersBuilding_ID,
  PersBuildingAbbr,
  PersBuildingFull,
  PersStreet,
  PersStreetNo,
  DepBuilding_ID,
  DepBuildingAbbr,
  DepBuildingFull,
  DepStreet,
  DepStreetNo
)
SELECT * FROM View_Employment
```

## C.5. View_WO_List

*View_WO_List* Creates a list of all workorders with information about the workorder number, the name of the person the workorder is for, the abbreviation of his/her department, the date of the workorder, the status of the workorder, the priority of the workorder, the name of the primary worker, and the name of the delegated worker.

```
CREATE VIEW View_WO_List AS

SELECT
  WO.WO_ID,
  Person.FirstName + " " + Person.LastName AS Name_For,
  Person.DepartmentAbbr,
  WO.WoDate,
  Status.Status,
  Priorities.Priorities,
  Person_Prim.FirstName + " " +
    Person_Prim.LastName AS Name_Prim,
  Person_Del.FirstName + " " +
    Person_Del.LastName AS Name_Del
FROM
  (((((((((SELECT WO_ID, Max(WodDate) AS
    Date FROM WOD GROUP BY WO_ID) Date
  INNER JOIN WOD ON WOD.WodDate = Date.Date AND
    WOD.WO_ID = Date.WO_ID)
  INNER JOIN WO ON WO.WO_ID = WOD.WO_ID)
  INNER JOIN Status ON Status.Status_ID = WO.Status_ID)
  INNER JOIN Priorities ON
    WO.Priorities_ID = Priorities.Priorities_ID)
  INNER JOIN Personal.dbo.TempPersonal AS Person ON
    WO.WO_For_ID = Person.Employment_ID)
  INNER JOIN Delegated ON
    WOD.DispatchDelegated_ID = Delegated.Delegated_ID)
  INNER JOIN Personal.dbo.TempPersonal AS Person_Del ON
    Delegated.Employment_ID = Person_Del.Employment_ID)
  INNER JOIN WorkGroup ON WOD.DispatchGroup_ID =
    WorkGroup.WorkGroup_ID)
  INNER JOIN Delegated AS Delegated_Prim ON
    WorkGroup.WorkGroup_ID = Delegated_Prim.Group_ID)
  INNER JOIN Personal.dbo.TempPersonal AS Person_Prim ON
    Delegated_Prim.Employment_ID =
    Person_Prim.Employment_ID
WHERE   ((Delegated_Prim.PrimWorker)=1)
```

## C.6. View_WOD_Last

*View_WOD_Last* connects all workorders with the last relating record in the *WOD* table.

```
CREATE VIEW View_WOD_Last AS
SELECT
  WOD.*
FROM
  WOD INNER JOIN (SELECT WO_ID, Max(WodDate) AS Date
    FROM WOD GROUP BY WO_ID) Date
      ON WOD.WodDate = Date.Date AND WOD.WO_ID =
        Date.WO_ID
```