USAID | UGANDA
FROM THE AMERICAN PEOPLE

MICROBANKER WINDOWS EXTERNAL EVALUATION

FINAL REPORT

September, 2006

This publication was produced for review by the United States Agency for International Development. It was prepared by Peter Scott, for Chemonics International Inc.

Rural SPEED

Rural Savings Promotion & Enhancement of Enterprise Development

MICROBANKER WINDOWS EXTERNAL EVALUATION

FINAL REPORT

Rural SPEED
A USAID funded project
Contract No. PCE-I-00-99-00003-00, TO 826

This report submitted by Chemonics International Inc. /September, 2006

TABLE OF CONTENTS

GLOSSARY

ADO – ActiveX data objects. A somewhat obsolete data access library promoted by Microsoft as the connecting layer between applications and databases.

Client application. The component of the application which runs on the user's workstation. This component is responsible for the display and management of the application windows, and is responsible for accepting input from the user and passing the input to the database engine, as well as displaying the data from the database to the user.

Cursor Engine. A component of ADO, this agent is most invisible to the developer, and is often inadvertently activated by a beginning programmer. This component will actually re-write the SQL update/delete statement programmed by the developer in an effort to hide the complexity of selecting the proper record to update/delete and maintaining data integrity in a multi-user environment.

Metadata. Data that describes the structure (not the content) of database structures such as tables and indices. Metadata includes the type of a data column (text, number), length, precision and null ability.

Stored Procedure. A code segment that resides in the database management system, as opposed to residing in the client application.

Transaction. A series of two or more insert/update/delete operations against the database in which all statements must succeed, or fail. If an error or crash occurs after the first intermediate statement in a transaction, it must be rolled back to restore the data to the state before the first statement executed. An example if transferring funds from one account to another, which consists of a debit of one account (an update), immediately followed by a credit (an update) of one account. If the second update fails, the debit from the first update must be rolled back so that the funds do not vanish into the ether.

Wide Area Network (WAN). A network configuration in which a central database is accessed by client application dispersed over a large geographic area. The alternative to the wide area network is the local area network (LAN), in which all clients and servers are located in a small geographic area (such as a building).

EXECUTIVE SUMMARY

This evaluation was undertaken as a preliminary step in determining if the MicroBanker software application is suitable for deployment in the Uganda SACCO environment.

The evaluation has taken place over a period of five days with the development and support staff of the MicroBanker team. The evaluation was performed from a technical perspective, rather than from an examination of features and capabilities. As such, it included examination of the application architecture, client source code, database structure, and communication protocols.

The results of this evaluation yielded the result that the MicroBanker application was very well suited to the Ugandan environment. In particular, the bandwidth constraints of the Ugandan telecommunication network are very well addressed by the attention paid to the development team to the efficiency of the communication protocols between the various application components. Additionally, the choice of a sophisticated database management system and the stateless nature of the application middleware enable it to be operated in the unstable power environment encountered in Uganda.

BACKGROUND

Introduction

In order to achieve the objective of increased rural financial intermediation (saving and lending) and also realized linkages between financial institutions the problem of inadequate Management Information Systems (MIS) for rural SACCOs needed to be addressed. Currently SACCO MIS is manually done or handled by flawed software packages. This encumbers SACCOs' ability to effectively generate operational reports, expand outreach or monitor their portfolios. Further, without the ability to generate adequate, precise, timely and reliable statements, external organizations, such as banks and MDIs are reluctant to enter business relationships with SACCOs and what's worse, the institutions themselves are less capable of detecting corruption and malfeasance internally.

USAID/Rural SPEED has spent considerable effort over the past 18 months considering the MIS problem from both the hardware and software perspective. The hardware issue has been solved to the project's satisfaction but their remains the problem of identifying, testing, piloting and rolling out a package capable of managing SACCOs' accounting and portfolio systems. Rural SPEED has reviewed and discounted several MIS programs suitability for the rural SACCO's over the past 18 months.

GTZ's Financial System Development Programme (GTZ/FSD) recently offered to partner with USAID/Rural SPEED to address this problem with another possible program, the FAO-GTZ Micro Banking System for Windows (MB Win) solution which was developed with FAO and GTZ funding.

In order to ascertain the suitability of the MicroBanker Win system for both the context of the Uganda SACCO environment as well as the ability of the system to address the concerns of the Rural Speed project, a technical evaluation of the application was undertaken.

Scope of Work

The scope of work to be performed under this evaluation is as follows:

- By sampling the MB Win source code determine the following:
  - How effective/efficient is the communication between the application and the database;
  - How scalable (from single work station, to LAN based applications, to WAN based applications) is the software;
  - To what extent can the software be easily customized to meet local requirements for Ugandan SACCOs:
    - What are time implication of customizing the software and
    - What are the cost implications of customizing the software;
  - To what extent is MBWin capable of being integrated with various other software packages:
    - Can it easily integrate accounting transactions from Quick books, Peach Tree, Pastel, etc.?
    - Can it be easily linked to the Uganda standard Performance Monitoring Tool?
- By observing the application itself determine the following:
  - Can the software suitably meet the needs of Uganda's SACCOs?
  - Could a local service provider, in Uganda, be developed to the point where supporting the program from Uganda would be feasible?
  - Has the system been successfully scaled up to LAN and WAN environments?
  - How does the software behave during power loss?
  - How does the software track expenditures within SACCOs using the system?

TECHNICAL EVALUATION OF MICROBANKER

This section of the evaluation addresses the technical evaluation performed on the MicroBanker Windows application. The evaluation was structured into four different evaluations: system architecture, client application, middleware and database.

System Architecture

This section of the report addresses the evaluation of the system architecture. For the purposes of this evaluation, the system architecture concerns the major components of the MicroBanker product along with the communication protocols by which the components pass information.

Evaluation Basis

The starting point for the evaluation was to examine the architecture of MicroBanker. The architecture is critical to effective implementation of an application since it determines the type of supporting software and telecommunications requirements required for effective implementation.

Evaluation Results

This section includes the results of the application architecture evaluation.

Overall Architecture

The architecture of MicroBanker Win is based on a standard 3-tier model, composed of a client, application server (middleware) and a database server. This is depicted in the figure below.

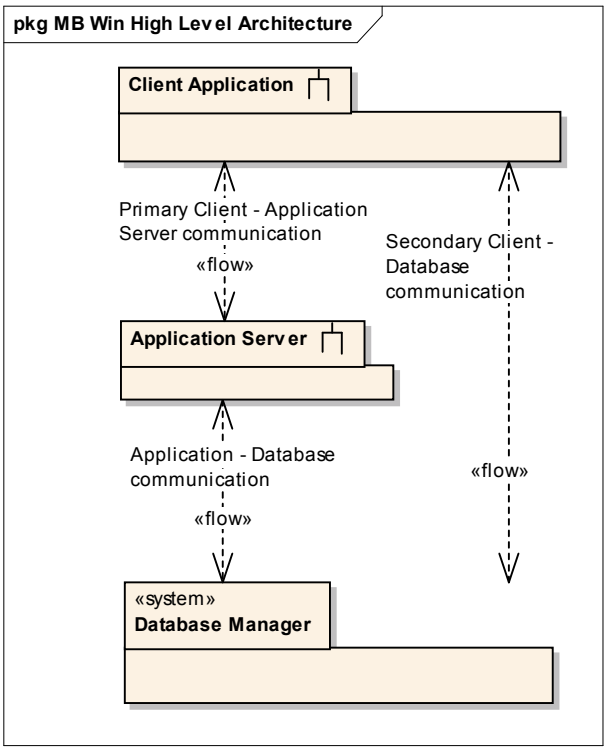


Figure 1 – High Level MicroBanker Win Architecture

One aspect of the architecture is somewhat anomalous to typical 3-tier architecture: the secondary client – database communication flow. In standard 3-tier architecture, the client application only communicates with the application server. No direct communication is enabled between the client and the database. This additional component does have ramifications for MicroBanker Win installation, and will be discussed at several points in this review.

The technology base for each component is described in the following table.

Component	Technology Base
Client application	Borland Delphi 5
Primary client-application server communication protocol	Borland Snapshot protocol, implemented via Microsoft DCOM over TCP/IP, used for the majority of application modules.
Secondary client-database communication protocol	Microsoft ADO protocol, implemented over TCP/IP, used for a limited set of application modules
Application Server	Borland Delphi 5
Database Management system	SQL Server 2000 or 2005
Application server to Database communication protocol	Microsoft ADO protocol, implemented over TCP/IP

The three primary application components (the client, application server and database management) are all standard types of applications whose requirements may be fully met by standard commercially available hardware. The evaluation of each of these components is included in other sections of this evaluation report.

One of the important focuses of this evaluation is to examine if the architecture of the application has an impact on the implementation in Uganda. To this end, the core architectural components to examine are the communication protocols that connect the three primary application components.

In order to better describe the communication protocols and their associated impact on implementation, a more detailed view of the architecture is helpful. This view is provided in the picture below.

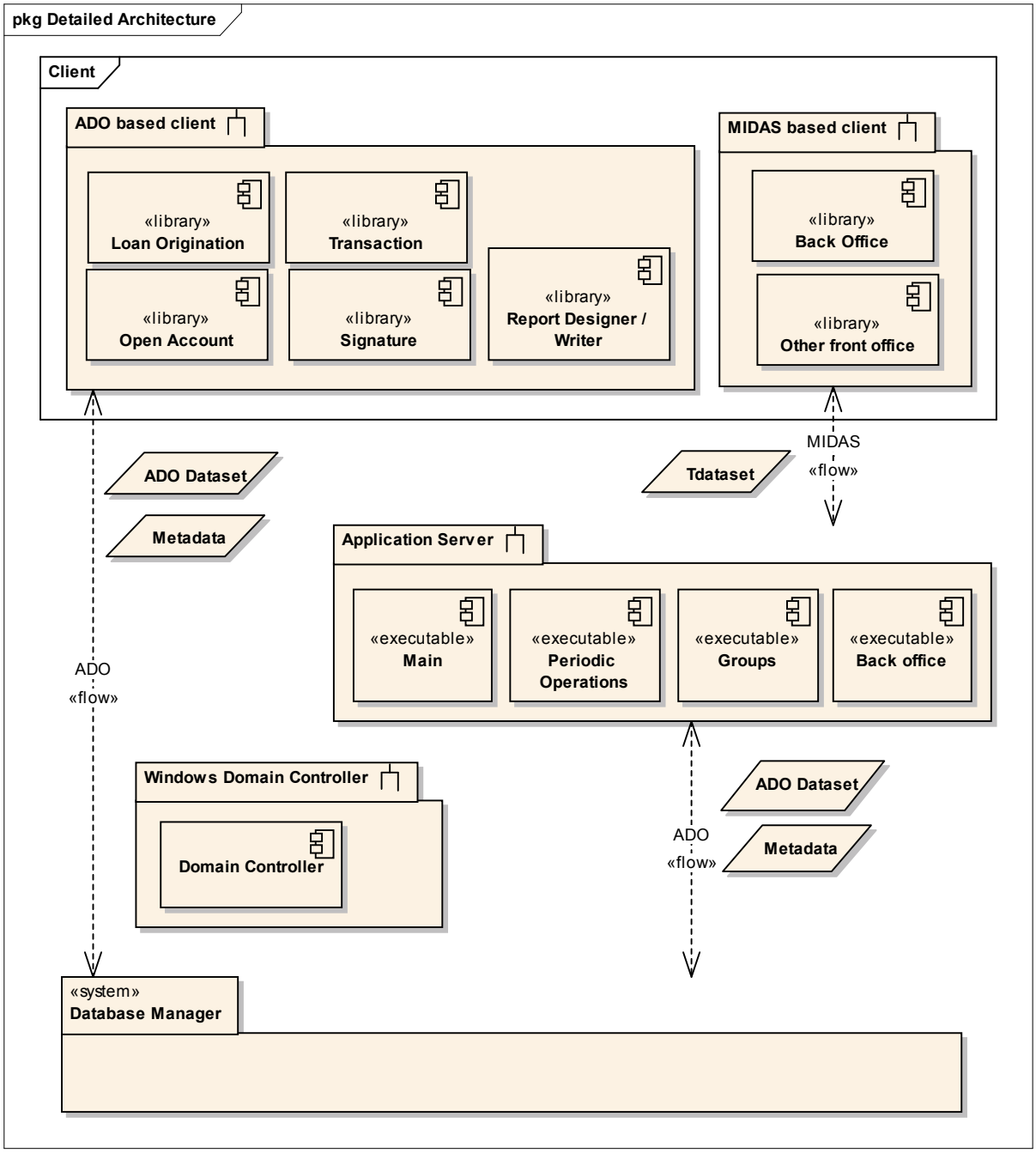


Figure 2 - Detailed MicroBanker Win architecture

Communication Protocols

From the perspective of implementation in the Ugandan environment, the communication protocols are important to examine, since they determine the location and topology of the implementation. This determination is based not solely on the availability of network availability, but also on the bandwidth requirements and associated one-time and recurrent monthly costs.

Borland Snapshot protocol

The first protocol to examine is that between the client and the application server, the Borland Client Dataset. A technical overview of this communication protocol is included in **Appendix A.**

This protocol is proprietary to Borland, and is not in as wide use in the industry as the more common Microsoft protocol. It is a very efficient protocol, having been developed as an alternative to the more common Microsoft protocols. The Microsoft protocols have primarily been developed with a focus more on an ease of programmatic use, at the expense of efficient use of bandwidth.

The use of the Borland protocol between the client and the application server makes the most efficient use of available bandwidth that is possible within the constraints of the development budget, and is a very wise design decision. Discussions with the development team indicate that they plan, in the near term, to use compression technology with this protocol to achieve greater bandwidth efficiency.

Microsoft ADO protocol

The second communication protocol to examine is that between the client and the application server. In order to achieve the most effective use of bandwidth between these protocols, the developers have eschewed the common Microsoft ADO protocol and instead have implemented the Borland Client Dataset binary protocol. This protocol uses a binary (as opposed to text) communication method, which makes optimal use of communication channels. However, this protocol is not used for all the parts of the client application.

Primarily due to resource constraints during the migration from DOS to Windows, several parts of client application bypass the application server, and communicate directly with the database using Microsoft ADO communication protocol.

While the use of this protocol is not, in itself, incorrect, it does have implication for bandwidth requirements when the application is deployed in a Wide Area Network topology, or where the database is located remotely from the client application (an example would be where each branch would have it's own database, but all the branch databases are located in a central site).

The fundamental problem with using the ADO protocol with the amount of metadata that it automatically fetches from the database whenever a SQL statement is executed (this explanation is not completely accurate, but in order to simplify the discussion, is correct enough for this evaluation). See **Appendix B** for a discussion of this issue.

In the Ugandan bandwidth constrained environment, this is a serious shortcoming. While the developers may recommend a standard bandwidth requirement (such as 64 or 128 KB/second for each branch, this recommendation is made based on limited field experience, not on empirical evidence derived from testing. Further, at present, there is only one site (in Honduras) that is operating in a WAN configuration. Further, the Honduran installation was based partly on ADSL technology, which is does not provide symmetrical two-way network traffic, and hence can have an impact on the application. Further, several branches in Honduras are able to operate with 64 Kbps bandwidth only with intelligent multi-protocol routing and direct point to point communication protocol provided by the ISP. Using a normal internet connection at other branches, they have settled on 128 Kbps bandwidth, but even this results in slow perceived performance for front office operations. Faulu Uganda has tried a WAN configuration, but has not performed an analysis of the network traffic to determine an optimal (or even adequate) set of bandwidth requirements.

In addition, there has been no empirical testing done to determine what the bandwidth requirements for the head office should be. In Honduras, the central office currently has a 512 Mbps connection, again with intelligent multi-protocol routing and point-to-point connectivity. One of the issues associated with scalability of the application in a WAN configuration is the incremental growth rate of bandwidth at the head office as branches are added.

The use of ADO to communicate directly between the client and the database (as opposed to the alternate channel of communication between the client and the application server using a non-ADO protocol) is currently limited to the following parts of MicroBanker Win:

- Loan Origination
- Configuration
- Open Account
- Front-office transactions (both credit and debit) for loans, checking and saving products.

It is somewhat of a surprise to find that one of the parts of the application that is most heavily used (transaction processing) is using the ADO protocol, and has not been migrated to the more efficient Borland protocol. The net effect of this reliance is occurs in a WAN based installation, where the communication bandwidth requirements are increased over what would otherwise be required if these parts communicated only to the application server.

Much to their credit, the development team has taken a great deal of effort to minimize the impact of using ADO across a network. However, there is only so much that is within the programmer's arsenal of tools that they may use to avoid this impact. As an example, they have tried to minimize bandwidth requirements by disconnecting from the database as soon as a record is fetched so that bandwidth is not consumed by managing a constant connection between the client and the database. Discussions with the developers show that they fully understand the issue, and intend to move away from the use of ADO in a yet to be determined future release.

Domain Controller

If the application is deployed in a fashion that the application server is remotely located from the clients, a Windows Domain Controller must be implemented. This will require the licensing of a Windows 2003 server and the associated CALs (client access licenses). The controller is necessary for the DCOM process resolution that allows the client component to find the application server.

It is possible to operate the application in a peer-to-peer configuration, which will not necessitate a domain controller. Hence, the institution can forgo the licensing of Windows 2003 and the associated CALs. However, for this configuration to work, the client component and the application server must be installed on each user workstation. The database may be located on a separate server, which should also be located on the LAN.

SUMMARY OF EVALUATION RESULTS

The architecture of MicroBanker Win is the result of a very effective and informed set of developers. It stands in contrast to that of other micro-banking solutions that this evaluator has examined over the course of several years.

From the perspective of implementing the application in Uganda, the single most constraining factor is the communication protocol that has been implemented between the client and database that bypasses the application server. By using Microsoft ADO technology in selected components of the application, the developers have mandated an increase in bandwidth requirements beyond that otherwise required by the core application components when the database is located remotely from the clients.

Unfortunately, the application components that incur this cost will be in common use when the application is implemented at a SACCO. The net result is that for use at a SACCO, the client and the database should be co-located (on the same LAN). The location of the client and the database at different sites will mandate a fairly expensive communication channel, which in the current Ugandan environment, could be prohibitively expensive.

If the institution wishes to locate a single application server (either at a branch or at a central site), a domain controller will be required, which will mandate the licensing of a Windows 2003 server and CALs. Licensing will be required on both an acquisition and a recurrent yearly basis.

Client Evaluation

This part of the report addresses the evaluation of the client component of the MicroBanker Win application. The client component is the Windows interface with which the user interacts.

Technology

The client component is programmed in Borland Delphi 5, and programming language offered as a native Windows 32-bit development environment. Borland's objective in offering this language was enable developers to program applications that consumed less memory and processor resources than similar Microsoft offerings.

Delphi is a COM based language, which is an older Microsoft technology that provides an interface communication protocol between programmed libraries. COM has been depreciated over the last 3 years in favor of .NET. The development team is well aware of this fact, and recognizes that sometime in the near term the application must be ported to .NET.

The most significant drawback to a .NET based application is the requirement for each non-XP operating system to download and install a 23 MB set of .NET library files. Since many of the MicroBanker clients are running Windows 98 and 2000 operating systems, this has been seen as the primary constraint to porting the application to .NET.

Evaluation Basis

The evaluation of the client code included examination of selected portions of the code base. The code was examined in order to determine: the application structure, overall quality of the programming, connection to the database / application server, and the sensitivity of the application to changes in the database structure.

Evaluation Results

This section includes the results of the client evaluation.

Application Structure

It is apparent that a great deal of attention has been paid to the structure of the application. The client consists of a core executable and approximately 18 dynamic link libraries. The libraries have been structured along the lines of business functionality, which yields an application which is readily adaptable to changes requested by the customer base.

Further, several hooks exist in the application which allow for the incorporation of additional functionality without extensive re-writing of the core application.

Quality of programming

From the code modules presented for the review, the quality of the coding is very good. A complete change history is included in each module, as are liberal comments.

It is obvious that the developers have placed a high priority on not only the maintainability of the code, but also the history of changes. This is important considering the relatively dynamic nature of the application.

Connection to Database Server

In a typical three tier application architecture (client – application server – database), the communication pathways do not bypass layers. The client should only communicate with the application server, and the application server should only communicate with the database.

However, there is a fairly large degree of communication between the client and database server, which violates this tenant of n-tier architecture. The developers are fully aware of this, and state that it is a result of a lack of resources required to re-architect during the port to the Windows version of the application.

In practice, this means that the bandwidth requirement for the client to communicate with the other application components is greater that it would otherwise be. This is primarily due to the differences in efficiency between the Borland communication protocol and the Microsoft communication protocol (both are discussed in the section on Application Architecture). This has implications for both the physical placement of the database as well as the bandwidth of the telecommunications link if the server is located remotely from the client application (assuming a standard PC based client configuration).

To mitigate this shortcoming, the developers have tried to minimize the bandwidth requirements. All the SQL 'select' statements (which fetch data from the database) are implemented by stored procedures in the database. This minimizes the amount of data (especially metadata) that is passed across the network. Inserts/updates to the database are performed by SQL statements embedded in the client (see the section on architecture for which processes are located in the client). This methodology is not as efficient as the use of stored procedures, but was mandated by a lack of resources during the upgrade to the Windows version.

Connection to the Application Server

For all back office functions and many of the front office functions, the client communicates only with the application server. The application server is responsible for feeding data from the database to the client. Changes made to the client are contained in a structure called a diffgram, which is passed to the application server. The application server then forms explicit SQL statements with which to update the database.

The client also receives a periodic 'heartbeat' from the application server (a periodic signal meant to inform the client the application server is still active and able to communicate). In case the application server should fail, the client will fail to receive a heartbeat signal and notify the user. This is a very valuable feature, especially in an environment of unstable electrical power.

Dependency of the application program on the database structure

Due to the use of SQL statements embedded in the client application, the client source code is dependent on the structural elements of the database. This results in the structure of the source code being very sensitive to structural changes in the database, which is not a desirable characteristic. As described in the architectural evaluation section of this report, this dependency is localized to four components.

It is obvious that a large degree of informed design effort went into the client component of the application. However, the design effort has been mitigated by the direct communication between the client and the database that bypasses the application server for some of the front office processes. The communication path does increase the network bandwidth requirements for the application. This only matters in a configuration where the database is located remotely from the client, (WAN configuration), as this is the desired implementation configuration for multi-branch organizations, with the net result that the institution is forced to maintain higher initial and recurrent costs for their network communication that would otherwise be required. Also note that in remotely locating the application server, an institution will also need to implement a domain controller, and hence will need to license and manage a Windows 2003 server.

This should be balanced against other competing banking application software however. None of the applications examined to date has been constructed on anything other than a client-server architecture, and so suffer from the higher telecommunications bandwidth requirements (due to the choice of ADO over Midas for all client to database communications). If, in a WAN configuration, the MicroBanker application server is co-located with the database server, an institution may be able to obtain a net savings in communication costs due to the higher efficiency of the communication protocol between the client and the application server.

If the developers are able to remove the direct communication between the client and the database (by passing all communications from the client to the application server) in a future release, MicroBanker would have a clear advantage over all other competing products.

Application Server Evaluation

This part of the report addresses the evaluation of the application server (middleware) component of the MicroBanker Win application. The application server component contains the business logic and data access negotiation between the database and the client components.

Technology

The application server is based on the same technology as the client, Borland Delphi 5. See the previous section of the report for a summary of this technology.

Evaluation Basis

The application server was examined with regard to two aspects: business logic encapsulation and data access management.

Evaluation Results

This section includes the results of the application server evaluation.

Distinct Application Servers

There are actually three application servers in MicroBanker Win: the main server, the periodic operations server, and the group server.

The main application server contains business logic and database connection management. The periodic operations server contains the business and processing logic for begin and end of daily, monthly and yearly operations. The group server contains the business logic for the loan groups.

Business Logic Container

The application servers act to concentrate business logic in a single component, and thereby keep shared logic from being repeated at different points in the client component. The primary advantage here is the reduced amount of coding and testing required, resulting in an ability to more quickly service change requests to the application.

Data Access Manager

One of the strengths of an application server is the ability to concentrate the multiple connection requests from clients to one or a few database server connections. This concentration results in more efficient use of telecommunications bandwidth when operating in a WAN environment. It is of limited use in the context of LAN operations.

Having a data access manager component in an application server is the single most critical factor in determining whether an institution can centrally locate the database server for multiple branch operations. In this respect, MicroBanker is far ahead of the field in similar applications. The inclusion of data access management is what allows a specified branch to connect to another branch database (assuming the branch databases are centrally located).

As mentioned in the Application Architecture section of this report, several core functions have been constructed in a fashion that bypasses this access management functionality, and hence communicate directly with the database. This construction requires more telecommunications bandwidth than would otherwise be required.

The inclusion of a middleware application server is unique in the micro-banking software environment, and offers several important advantages. The most important advantage offered is the ability to offer an efficient communication proxy between the client and the database. This is important only in the context of a WAN configuration. The second major advantage is the ability to keep core business logic out of the client component. By so doing, the management of change requests becomes much more effective and efficient, which allows the MicroBanker team to respond in a timely fashion to customer change requests.

Database Evaluation

This section contains the findings, analysis and summary of the database evaluation phase.

Technology

The MicroBanker Win database is based on Microsoft SQL Server 2005. Two editions are supported: SQL Server 2005 or SQL Express 2005. SQL Server 2005 is a full-fledged licensed database manager that offers support for hundreds of clients, and virtually unlimited size. The Microsoft Express 2005 is a no cost alternative that offers the same functionality as the licensed product, but is limited in both the amount of data it can store (maximum size of the database is 4GB) and simultaneous users (maximum RAM of 1 GB).

Evaluation Basis

The database evaluation examined several aspects of the database features and operations that experience has shown to be critical to successful functioning in power and bandwidth constrained environment.

Evaluation Results

This section includes the results of the database evaluation.

Database Structure

In reflecting the migration of MicroBanker from a DOS based application to a Windows based application, the database contains elements of both applications. Some of the structures in the database represent artifacts from the file-oriented processing design, and some of the newer structures represent more proper relational database tables.

Discussions with the developers confirm that the older file-oriented structures are a result of the lack of design and development resources needed to implement a re-design.

Data Updating

One of the most crucial factors in maintaining the integrity of the data contained in an application database is the manner in which insert, update and delete operations are performed. Fetching data from a database does not have an impact on the integrity of the data, therefore does not have an impact on data integrity.

There are three common methods of performing updates and inserts into the database. They are:

- Updating the Microsoft ADO recordset which hold data fetched from the database, and then allowing the underlying Microsoft Cursor Engine to figure out the best method to use for updating. This method offers the advantage of being the easiest method to program. However, this ease comes at a great cost in a bandwidth constrained environment. That is the amount of metadata that the Cursor Engine must fetch in the background to support its internal operations.
- Explicit INSERT/UPDATE SQL statement coded into the client of middleware application.
- Stored procedures contained in the database management system. Specific parameters required for each operation are passed from the client application to the database, which then executes the stored procedure.

The first method, updating the ADO recordset, is the easiest to program, and in the short term, provides the greatest productivity is the development team is burdened with low-skilled developers. However, this method incurs a hidden cost in terms of passing the meta-data about the specific update/delete operation across the network. In the bandwidth constrained environment of Uganda, use of this programming construct is directly responsible for the overwhelming recurrent communication costs of implementing an application in a wide area network.

The second method, explicit SQL insert/update statements, requires more programming skill than the previous method, but offers a much more reliable mechanism for maintaining the integrity of the database than the previous method. The largest disadvantage to this method is that the SQL code is tightly tied to the database structure. If a change occurs in the structure of the database at some point in the development or maintenance lifecycle that is not reprogrammed in the client application, it is possible for the application to operate without explicit errors, but to damage the data integrity of the database.

The third method, database stored procedures, is the most sophisticated of the three methods. Since the stored procedures are contained and managed in the database, this method offers the best protection of data integrity. Another advantage is that the database will automatically notify the developer if changes in the structure of the database require changes to the update/delete functions.

At this point in the development cycle, MBWin is based on both explicit SQL statements and stored procedures. All fetches of data from the database are based on stored procedures. Most data inserts / updates are performed by explicit SQL statements contained in both the client and the application server (see the architecture section for where the business processes are located). No deletions are ever performed. Discussions with the development team revealed that near term plans include removing the explicit SQL statements from these components and migrating them to stored procedures in the database. Not only will this increase performance, but more importantly will reduce the sensitivity of the application to the database structure.

Back office batch processing

The modality of batch processing for back office functions has been given a great deal of design and programming attention, and sets MicroBanker apart from other banking applications.

Most applications pass batches of data between the client and the database, which is a very bandwidth intensive operation. Changes are made on the client, and are then passed back to the database for

inserts / updates. Since this typically happens at the close of every month, the network is flooded with batch processing traffic. However, this is the same time that the tellers are trying to complete front- and back-office transactions. These functions are greatly impeded due to congestion on the network.

Due to the inclusion of a middleware application server, MicroBanker is able to keep the batch processing concentrated on the application server, and away from the client. All the actual batch generation is performed by stored procedures in the database, so that only instructional and key information are passed from between the application server and the database, greatly minimizing congestion on the network.

Transaction Control

Transaction control is a construct used to ensure that a series of intermediate database operations (insert/delete/update) occur as a single operation (a transaction), or fails as a single operation (in which case any intermediate operations performed prior to the failure are undone). Transaction control is the primary mechanism used to control data integrity.

Transaction control may be done either from the client application or from a database stored procedure. The latter method is the most reliable, and allows the database management systems to actively maintain the integrity of the contained data. Transaction control from the client carries a greater risk to the data integrity (especially in a client-server type of application) since the client may crash or power off in the middle of a transaction, while the server keeps operating.

MBWin primarily uses transaction control outside the database. The exception is the back office batch processing for close of day/month/year. While extra programming effort has been expended to implement controls external to the database to assist in recovering from a client crash during a transaction, this implemented controls are not 100% reliable.

Failed Transaction Recovery

In the situation where either the client application connected to the database, or the database server itself, crashes of the middle of a transaction, a recovery mechanism must be implemented to either complete the interrupted transaction (called roll forward), or to undo the interrupted transaction (called roll back).

SQL Server 2005 offer transaction recovery mechanisms implemented through transaction logs. These are operating system files in which the pending transaction details are logged before the transaction is started in the database. Upon database startup, these logs are checked for interrupted transactions, and the appropriate recovery tactic (roll forward or roll back) are initiated.

The operation of this mechanism is relatively simple. Before a transaction (an insert or update operation) is performed in the database, a before and after snapshot of a record is taken and logged into a simple file outside the database (the transaction log). If the power should be interrupted during the database operation, the database is left in an inconsistent state. Upon subsequent startup, this inconsistent state is recognized, and the database automatically goes into recovery mode. During recovery, the data involved in the interrupted transaction is rolled back to its initial state, and an attempt is made to redo the transaction with the information contained in the transaction log. In most cases, the redo operation will be successful, and the database can start from the point of the last recorded transaction. Very rarely, if the timing of the power loss exactly coincides with writing the transaction log, the log itself will not contain the information required to complete the redo of the transaction, hence the incomplete transaction will be lost.

Error Trapping / Logging

A review of the database stored procedures show that any errors that occur during data updates/inserts are trapped, but not logged by the database. The errors are thrown back to the application client layer via the client application server. The application client is responsible for notifying the user and logging the error.

This has implications in the context of MicroBanker in a Wide Area Network configuration. If during a front-office transaction, the network connecting the application client with the remotely located server is to go down, there will not be propagation of the error notification thrown by the server. Normally, the front-office worker should expect to receive a confirmation message for the transaction. If they do not notice the confirmation is not received, any error occurring in the database will not be noticed by the user, as well as being not recorded.

While the probability of this occurring on a regular basis is very small, it does exist nonetheless.

The Microsoft series of database managers are widely used and supported. They offer many advantages over flat-file management (Dbase, FoxPro) systems used by other micro-banking applications, and more properly suited to this type of application than the competing products from Oracle or Sybase. The Microsoft SQL Server series of databases are very robust, and can operation in an unstable electrical power environment due to their automated transaction recovery mechanisms. Since most of the institutions currently under consideration in Uganda for implementation are relatively small, the non-licensed version of the database will suffice, hence avoiding the cost of database licensing.

The method of database inserts/updates programmed in certain business functions could have been programmed in a more bandwidth conservative method, but this is only applicable in a WAN configuration where the database is remotely located from the client application. Many of the inserts/updates have been programmed with a great deal of effort to minimize network bandwidth utilization. Future changes to the application will increase the efficiency of bandwidth utilization in a WAN configuration, leading to MicroBanker being one of the most bandwidth efficient banking applications on the market.

PMT Integration

Both GTZ and Rural Speed intend for MicroBanker and the PMT 2006 be integrated to ease the transfer of data from MicroBanker to the PMT. The integration would allow the data contained in the MicroBanker database to be transferred to an instance of the PMT running on a user's desktop.

Discussions were held with the MicroBanker development team to assess the feasibility of the integration.

The MicroBanker team has developed a CGAP report in the MIS application, which contains many of the ratios that are generated by the PMT. After a quick review of the PMT beta application, it was agreed that integration would not only be possible, but would be beneficial both the Ugandan microfinance community and the MicroBanker development team.

Integration Approaches

Preliminary discussions were held concerning the different approaches that could be taken to implement the integration.

The first approach considered was to allow the PMT access to the MicroBanker database in order to fetch the data directly from the database. The major advantage of this approach was the minimal amount of resources required by the MicroBanker development team to implement. This approach held several disadvantages, primarily being that the interface to the PMT would be highly dependent upon versioned/customized releases of MicroBanker.

The second approach considered was to allow the MicroBanker application direct access to the PMT database in order to insert data into the PMT. While this approach offered the advantage of minimal resources requirements on the part of the PMT developer, it carried with it similar disadvantages to the first approach.

The third approach considered was the use of a data interchange external file. This file would be written by the MicroBanker application upon a user request, and would then by read by the PMT application upon a similar user request. This approach removed the primary disadvantage of tightly coupling each of the two applications to the data structures of the other application. It would also spread the development task somewhat equally among the two development teams. The primary disadvantage to this approach was that it would require resources on both development teams to implement, as well as a relatively larger amount of coordination in the specification of the external file structure.

At the end of the discussions, it was agreed that the third approach offered the lowest risk approach, and would facilitate data interchange between the PMT and other microfinance software applications in the future. It would also afford the MicroBanker application with an externally audited CGAP-based ratio report.

Data Interchange Format

It was also decided that the format of the data interchange file would be XML-based. This would provide a platform neutral format for exchanging data between the PMT and any banking software solution. Discussions with the developers concluded that the XML document would be element based, rather than attribute based.

Data Interchange Content

After the format issue, the next most important issue is the content of the interchange file. There are two aspects to be considered: periodicity and scope.

The PMT currently allows the user a choice of data entry on either a monthly or quarterly basis. MicroBanker simply maintains a current balance for the General Ledger that is not based on a fixed closing basis; it is feasible to generate a data interchange file on either a monthly or quarterly basis without gathering data from the base transaction tables. Data for the portfolio section of the PMT must be gathered from the transaction tables, since there are no balances kept for most of this data. Since most of their clients operate on a monthly basis (with the exception of Nepalese based institutions), it was decided to offer a monthly export. To mitigate this, the PMT provides the ability to report on a quarterly basis with input data at the monthly resolution level.

The next aspect taken under consideration concerned the scope of the data contained in the interchange file. The two obvious choices were to include either a full years' worth of data (to be incrementally grown with each month), or to include only a single month of data in a series of files. However, due the issues associated with an institution back-dating transactions (entering them into the system after the actual transaction date), there was a very real danger that the general ledger portion of the data could become unsynchronized with the portfolio data. Hence the decision was taken to offer a series of monthly-based export files.

Features to be implemented

In order to implement the interface, several features need to be designed, implemented and tested in both MicroBanker and the PMT.

From the MicroBanker team, he first of these would be a screen in MicroBanker that would allow the user to define the mapping between the Chart of Accounts defined in MicroBanker and the PMT data elements. The specification of this mapping would allow the aggregation of G/L line item data in MicroBanker to the PMT line item data. The second feature required would be the design and development of the data aggregation processing in MicroBanker that would perform the rollup the G/L line item data (as specified by the mapping) prior to the data export. The next feature to be implemented is the data export, which would actually generate and write the XML data interchange file.

Several features would also need to be designed and implemented in the PMT. The first of these would be the data import, which would read and interpret the XML data interchange file generated by MicroBanker. The second feature would be a mapping routine, which would take the data read by the import and place it into both the PMT workbook and the PMT database.

Both teams would need to work on the XML document schema (structure definition) which would guide the other feature designs.

Other Issues

There are several other issues that arose during discussions about the integration. While these are not impediments to the integration, they should be taken into consideration.

The first issue concerns branches. MicroBanker is inherently a single branch application. At present, if an organization has multiple branches, it is often advantageous to co-locate all the branch database files on a single centrally located database server. However, the organization still must manually consolidate the data from the separate branch databases on a monthly basis to report on the institution as a single entity. The MicroBanker team does offer a consolidation module to assist in this integration and reporting, but it is not in wide use. The team is present working on a multi-branch version of the application, but the release date of this is uncertain. The PMT would seem to offer potential to integrate the data from the different databases, and act as a no-cost offering to institutions to perform integrated reporting. However, the PMT Working Group has decided not to offer multi-branch support in the current release of the PMT, hence it's utility to consolidate and report across branches is non-existent unless the institution would undertake to consolidate the databases manually.

The second issue concerns back-dating of transactions. Currently, the structure of the MicroBanker database only allows it to maintain a current balance of the General Ledger. If the organization is to practice back-dating or processing of transactions, the effect of these transactions will not be reflected in the proper accounting period. Worse, the portfolio component of the data would be captured in the transaction tables, not the balance tables. Since the data export for the interchange file will be based on the General Ledger current balance table, it is likely that the general ledger current balance tables could become unsynchronized with the portfolio tables. It was decided by the PMT Working Group that a PMT user would be able to change historical data in the PMT, since this is a practice followed by many institutions on Uganda. Hence the safest manner in which to maintain consistent export data is to limit the data in an export file to a month. If backdating is to occur, a prior months export should be regenerated and imported into the PMT.

MicroBanker allows the institution to define the aging categories for the Portfolio at Risk (PAR). It does not constrain the institution to the aging categories used by the PMT. It was decided that the data export would group the PAR according to categories defined in the PMT. However, it will not be possible for the data to be exported according to user defined age categories, since MicroBanker allows up to six age categories, while the PMT user defined PAR categories are limited to four.

Integration Tasking

The tasking that would be involved in developing the interchange would follow this outline:

Task	Task Description	Resources
Interchange Format Design	The definition of the XML document structure based on a study of both the Microbanker database structure and the PMT database structure	PMT and Microbanker
MicroBanker G/L to PMT mapping definition screen	This task encompasses the design and implementation of screen within MicroBanker that allows a user to define the mapping between the user Chart of Accounts and the PMT line items.	MicroBanker

MicroBanker data aggregation	This task includes the design and implementation of the data processing routines that aggregate the MicroBanker data to the level required by the PMT.	MicroBanker
MicroBanker data Export	This task includes the design and implementation of the routines within MicroBanker that forms the XML document and populates it with the aggregated daily.	MicroBanker
PMT data Import	This task includes the design and implementation of the routines within the PMT required to read the data interchange document	PMT
PMT data mapping	This task includes the design and implementation of the routines required to parse the XML document, extract the data and map the extracted data onto the PMT workbook and database.	PMT

External Accounting System Integration

This aspect of the evaluation addressed the ability of the MicroBanker application to be integrated with other accounting applications. Due to space considerations, a summary of the evaluation is included in the Conclusions and Recommendations section of this report.

Modification for Implementation at Ugandan SACCOs

This aspect of the evaluation addressed the ability of the MicroBanker application to be modified for use with respect to the (unstated) contingencies of the Uganda SACCO environment. Due to space considerations, a summary of the evaluation is included in the Conclusions and Recommendations section of this report.

CONCLUSIONS AND RECOMMENDATIONS

This section of the report contains the conclusions and recommendations of the evaluation. It is structured according to the questions contained in the ToR.

Effectiveness / efficiency of the communication between the application and the database

A great deal of time during the evaluation was spent examining the communication architecture of the application. Details are provided in the architectural evaluation section of this report.

MicroBanker uses a Borland-proprietary communication protocol for most of the data communication between the client software and the application server, as opposed to the more common Microsoft client-server communication protocol (ADO).

In general, the Borland communication protocol is much more efficient in its use of network bandwidth than corresponding Microsoft communication protocol. The utilization efficiency has further been increased by not keeping an open connection between either the client-application server components, or the application-database server components.

Somewhat surprisingly, some operations in MicroBanker do, however, use the Microsoft ADO protocol to communicate directly from the client to the database, bypassing the efficient communication path between the client and the application server (these operations are described in the architecture evaluation section of this report). From an operational perspective, one of the most heavily used functions in the application, teller transaction processing, is one of the operations that bypass the more efficient communication pathway between the client and the application server.

The development team is fully aware of the bandwidth costs of this approach, and is bringing resources to bear on the problem. This bypassing of the application server does have implications in operating MicroBanker to operate in a wide area network configuration. The primary implication is the need for increased communication bandwidth between a branch and the database site. Additionally, the Borland protocol used is based on Microsoft COM technology. If the application server is located outside the client LAN (as in a WAN configuration), the use of COM technology requires the inclusion of a domain controller, which is only available on a Microsoft Windows 2000 / 2003 server, which carry a licensing and maintenance cost.

In summary, the MicroBanker team has spent a great deal of resources on making maximum use of the network bandwidth to ensure reliable use in bandwidth constrained environments. However, the use of network bandwidth is not fully optimized at this point in the application release cycle due to several functions which rely on the more bandwidth intensive Microsoft communication protocol.

Suitably of the software to meet the needs of Uganda's SACCOs

Given the strong SACCO oriented background of the development team, the flexibility of the general ledger configuration and the ability of the application to support custom loan and saving products, it is unlikely that any of the Ugandan SACCO requirements could not be met through the existing configuration options of the current application. Experience has shown that type of changes most likely to be requested are for non-standard interest calculation for 'unique' loan products, which are actively discouraged by the support team.

Customization of the application to meet local requirements for Ugandan SACCOs

The MicroBanker support and development team have a fairly standard approach to customization. If a customer requests a change, the feasibility of that change and its applicability to the base of MicroBanker customers is evaluation. If the change is found to offer advantages to the large portion of the customer base, it is generally offered implemented in the next release as part of the core code base.

The development team will not make client specific changes to the core code base that are only found to be useful to a single institution. However, a mechanism does exist for implementing one-off changes. The architecture of the application allows the integration of a run-time library via the inclusion of a dynamic link library (a DLL). By placing the DLL in a specific folder on the client workstation, the functionality encoded in the DLL will be invoked by one of several hooks included in the main application. This method of extending functionality has been specifically used to afford custom interest calculations, but the customer carries the full cost of design, development and testing.

The rapidity with which these one-off modifications can be implemented depends on the current workload of the development staff. Since the staff is relatively small, they do not have the luxury of devoting resources immediately to any requested change.

The costs for changes are evaluated on a per-change basis, and here is not hard and fast price schedule for change requests.

Integration Potential with Accounting Packages

At present, MicroBanker does not offer any programmatic support for integration with any external accounting packages. Discussions with the development and support team indicated that virtually all their clients use MicroBanker for all accounting purposes within the institution.'

Non-programmatic support is offered via data import routines. Integration based on data import means the integration is only one-way: from external accounting application to MicroBanker. In this scenario, one or more ancillary systems (such as payroll, petty cash vouchering, etc.) export data that constitutes one or more line items in an overall general ledger maintained in MicroBanker. In this scenario, MicroBanker can import one or more subsets of the general ledger on a monthly basis. It is assumed that in this scenario that the account codes in each of the smaller subsystems are aligned with the coding schema configured in the MicroBanker general ledger.

This second scenario allows one-way integration with desktop accounting systems such as Quickbooks and Peachtree. If the multi-currency versions of these systems were to be used, multiple imports, one for each currency, would need to be done on a monthly basis since multi-currency support in MicroBanker is offered by configuring separate databases for each currency.

There are two common scenarios in which MicroBanker does not directly support integration: multi-currency and multi-branch. In both these instances, MicroBanker mandates maintenance of separate databases. Both these scenarios are predicated on data consolidation. That, in turn, requires data export from MicroBanker databases into some form of a consolidated accounting/reporting system. The solution MicroBanker offers for these scenarios is the MIS, which is based on a consolidated database. This database is fed (in non-real time) by backups from the separate MicroBanker databases.

However, if an institution chooses not to license the MIS, they are left on their own to develop data export routines to feed the consolidated accounting system, or to use data backups in a fashion similar to the MicroBanker MIS.

Integration Potential with the Performance Monitoring Tool

As discussed in a previous section of this report, the integration of MicroBanker with the PMT is readily achievable. The approach selected during initial discussions was to reduce the dependency of both applications on internal data structures by utilizing an XML-formatted data interchange document with a well-defined schema for communication.

The definition of the schema should be published as part of the PMT technical documentation in order to avail other software vendors of the ability to interchange data with the PMT.

The major obstacle to data interchange arises from the ability of the MicroBanker end-user to define a customized chart of accounts. Because of this, a mechanism must be incorporated in MicroBanker to provide a mapping between the chart of account line items in the MicroBanker General Ledger and the PMT data element line items. At this point in time, this mechanism does not exist in MicroBanker; hence it exists as a future development task. The mapping must be accurately configured by the user at the time of configuring the general ledger chart of accounts, or the data interchange will be inaccurate. This issue will arise during the integration with other accounting / portfolio packages.

Another integration issue concerns multiple-branches. During the formation of the PMT requirements, it was decided the PMT would only support institution-level information. As such, it is not capable of integrating data from multiple branch data sources (such as the PMT interchange XML document). That means the data exported from MicroBanker must have already been integrated across branches before it is imported into the PMT. However, since MicroBanker does not yet offer a multi-branch solution, a multi-branch organization must either license the MicroBanker MIS (which does integrate multiple branch databases), or rely on some external means of integrating data and then generating the PMT interchange XML document. However, at present, the MicroBanker MIS does not support generation of the PMT data interchange document.

Feasibility of local provision of support services

At present, MicroBanker is not widely installed in East Africa. Consequently, there are no local support services available. The support team in Bangkok feels that it would be beneficial to establish a fledgling support center there that could grow with the market.

Support service should include:

- Implementation
- Training
- Technical training in Microsoft SQL Server maintenance
- MicroBanker release maintenance
- 1st level support
- Issue tracking and forwarding (to the development team)
- Report design and production
- Network design and configuration
- Data replication and integration (to support multi-branch operation)

Of course, any personnel in the support centre must be training in the product in Bangkok by the support team. Training is an essential first step in establishing support.

Perhaps the most important aspect of support is the ability to maintain a SQL Server database. No database will run indefinitely without monitoring and maintenance. Periodically, maintenance operations should be performed to ensure operational stability, especially as the database grows in size. The current release of MicroBanker is installed with SQL Server 2000 or 2005. The 2005 version of the software is significantly different from the 2000 version, and it was emphatically noted by the support team that technical training on the 2005 version was critical.

Due to policy restrictions from the MicroBanker development team, no local modifications of the core source code will be allowed. Hence, customization other than report production will not, in the initial development, be sourced by the local support centre. It is expected that the support centre will vet and forward to the support team valid bug reports and change requests. To support that mode of operation, the MicroBanker team offers a web-based incident and bug tracking system. The support centre should be the interface between the customer and the incident tracking system.

A support resource should have a comprehensive understanding of the system architecture, component configuration and communication protocols that underlie the MicroBanker application. This information is required to assist in proper network design and configuration and change management of the application, especially in a dynamic telecommunications environment such as Uganda where the capacity of the networking provisions are in a rapid state of change.

While it has not been implemented at other support centres, it has become obvious that the ability to model and track network traffic in support of wide-area network configurations is a valuable asset for the support team to offer.

Since at present the multi-branch version of MicroBanker is at least 7 months away from release, current and near term multi-branch installations will require the support team to be versed in data consolidation. Since the development team has already designed and produced a multi-branch consolidation database and MIS with an easy to use interface that allows rapid consolidation of multi-branch data, the support center should be familiar with that component of MicroBanker. However, there is only a relatively few set of reports available from this MIS.

This brings up the last requirement for the support centre: the ability to design and generate reports. In order to do this, the centre should be familiar with the reporting tool provided (which is sourced from Digital Metaphors), or another third party tool (such as Crystal Reports). Knowledge of the reporting tool is not entirely sufficient. The support staff must also be knowledgeable with the structure of the MBWin Core and MIS databases and financial reporting in general.

Scalability potential of the software

In the context of this evaluation, scalability of the software has two dimensions:

- Scalability in terms of the number of users in a local area network environment
- Scalability across a number of branches over a geographically dispersed area

MicroBanker Windows offers excellent scalability over users in a local area network. Since the application is based on a three tier architecture (client / application server / database server), scalability has been addressed from an architectural perspective, ensuring a minimal set of problems will arise from an increased number of users. As an example, in an installation in Sri Lanka, a single installation has 10 tellers performing, on average, 3,000 transactions per day. The design of the application server ensures that each teller session uses a minimal amount of server resources. This allows a standard off-the-shelf physical server to support both the application and the database server.

Scalability over a geographically disbursed area is commonly referred to as a WAN configuration. As evaluated, MicroBanker incurs some restrictions, but offers greater scalability that any other package examined to date by this evaluator.

There are several configuration options available for operating in a WAN environment. These are:

- One or more client components communicating with an application server all located at a branch office with a single database server located at the head office. In this configuration there are multiple database instances running on a single database server located at the head office.
- One or more client components running on PCs at a branch office communicating with a single application server and database server located at the head office. With this scenario, it is possible for the application server and the database server to be running on either separate physical servers or on the same server.
- One or more client workstations running remote desktop protocol (RDP) against a terminal server located at the head office. The terminal server runs individual MicroBanker sessions (one per connected user). Each session communicates with a single application server, which

communicates with a single database server. The terminal server, application server and database server may all be run on a single physical server, or may be hosted on separate servers.

While all these configurations have been tested, the only configuration installed and operating in a WAN environment to date is in Honduras. This site operates with PCs as clients with an application server at each branch, and a single database server at the head (this is the first configuration described). Acceptable performance was achieved with 64 Kb/second telecommunication links at the branch office (but only with intelligent multi-protocol routing and point-to-point switching), with a 0.5 Mb/second link at the head office. It was noted by the development team that the performance of the teller workstations exhibit timing issues that could be resolved by increasing the bandwidth to each of the branches.

This configuration however, does not mean that all the branch data is contained in one database. With the current release of MicroBanker, each branch will have a separate instance of the database running at the head office. This still leaves the organization with the problem of consolidating the branch data at the end of every month.

One issue to be addressed is the choice of database management system when operating in a WAN configuration. The design of the application server allows it to act as a multiplexer of sorts that allows many users to connect to the non-licensed version of SQL Express 2005. However, factors such as performance and size limitations of SQL Express 2005 preclude its effective use in a WAN configuration, requiring institutions to license a full version of SQL Server 2005. At a starting price of approximately $1,500 USD for a license, the cost should not be an inhibiting factor.

Another issue concerns the number and locations of the application server. The client components communicate with a remote application server (one not located on the client workstation) via Microsoft COM. This communication requires a domain controller so that the client components can locate the application server. The implementation of a domain controller requires licensing and maintenance of a Windows 2003 server, with associated CAL licences (both procurement and annual maintenance).

To address the problem of multi-branch consolidation, MicroBanker offers the MicroBanker MIS. This MIS operates against a single consolidated database. This database, used solely for reporting, is a separate database from the main transactional databases. The consolidation is done by importing backup copies of each branch database into the reporting database. At present, approximately 11 different reports are offered. Reports can be exported to multiple formats, including PDF and Excel. However, all the reports examined only report on a single month of data at a time. There are no multiple month or trend reports currently being offered, however a very attractive and flexible series of multi-month/year graphs are offered. This MIS is a relatively new product, and has only been installed in three sites.

Behavior in an environment of unstable power

The single most crucial factor in an unstable power environment of any application software is the database. The simple fact of technology is that databases do not like to be turned off unexpectedly.

There are two types of database management systems commonly used in microfinance solutions. The simplest is a file-sharing database such as Foxpro or Microsoft Access. Access to these databases is managed through the operating system file mechanism. Simply put, these type of databases can only sustain a few interruptions from a power outage before they simply refuse to operate. If the data manager has not made backups of the file on a least a daily basis, the banking solution will fail within a relatively short period of time. Since even faithful backups are only made on a daily basis, an organization utilizing systems based on these databases are at risk of losing an entire days worth of transactions.

The second type of database management system that is in less common use is a full relational database management system. For the African market, this means either an Oracle or Microsoft SQL Server database. The most common product used in the local market is either Microsoft MSDE (a no-license cost scaled down version) or SQL Server 2000/2005. These are proper database management solutions that have provisions for operating in an unstable power environment.

MicroBanker uses the SQL Server database management system, which allows dependable operation in an unstable power environment due to the use of a transaction logging mechanism.

In addition to ensuring the database installation has enabled the transaction logging, the MicroBanker team has included a provision for automatically backing up the transaction log file, ensuring to the largest degree possible the ability of the database to recover incomplete transactions.

The MicroBanker support team stated that *none* of their clients has lost data due to power outages corrupting the database. Some of their installations operate in an environment similar to Uganda, hence the application can be considered fully tested to operate in the intended power conditions.

All that not withstanding, any installation should ensure that the database files are backed up on a daily basis, and an organization have the trained Microsoft technical resources available to recover the database in the event of a disc drive failure (which is a common occurrence under the stress of constant power loss). Disc drive failure may also be minimized by the provision of a properly size uninterruptible power supply (UPS). Further, a RAID configuration for the database data drives is highly recommended.

In addition to the vulnerability of the database server, there is also an issue with the application server software. Unlike the database management system (which operates as a system service, hence will automatically restart after the server reboots), the application server (the middleware) runs as an application. This requires that a person physically log-on to the computer where the application server is running. If the administrator does not log-on after the server re-boot, any client attempting to reconnect after a power failure will receive a ‘connection refused’ message. The development team is fully aware of this issue and intended to address it at some future point.

Additionally, unlike the database server, the application server operates in a stateless manner, meaning that there is no data to be lost or recovered during a power loss.

Ability of the software to track expenditures

MicroBanker allows the General Ledger chart of accounts to be customized by the organization. There are no real limits on the number of sub-accounts that can be defined. There is a limitation of 5 levels of sub-accounts, which is not a realistic constraint. Hence, expenditures can be tracked at any resolution required by the organization.

APPENDIX A: BORLAND DATABASE ACCESS TECHNOLOGY

This appendix describes the Borland database access technology used by MicroBanker Windows. This is the primary communication technology used between the MicroBanker Windows client/middleware and the database server.

DataSnap Technology

The core of the DataSnap technology lies in two components; the TDatasetProducer and the TClientDataSet. The connection components, and their corresponding data modules, in turn provide the medium for these data packets to move in. According to the Delphi help file, under the topic "Deploying multi-tiered database applications (DataSnap)":

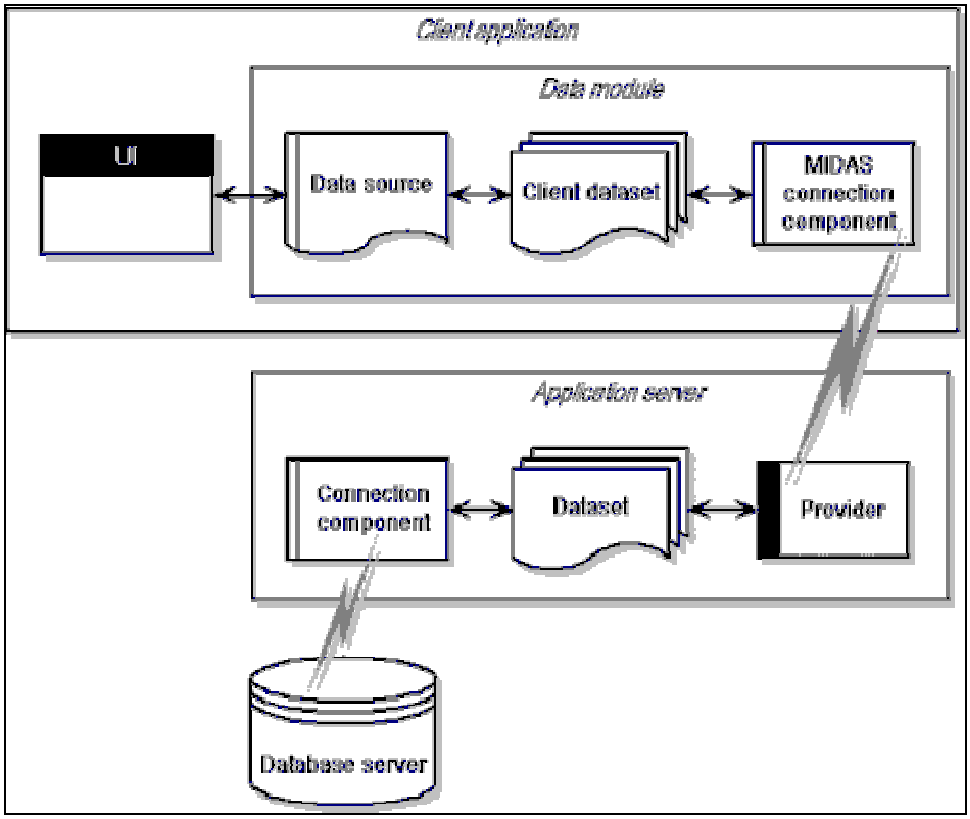
"DataSnap provides multi-tier database capability to Delphi applications by allowing client applications to connect to providers in an application server. Delphi's support for developing multi-tiered applications is an extension of the way client datasets communicates with a provider component using transportable data packets. Once you understand how to create and manage a three-tiered application, you can create and add additional service layers based on your needs."

Understanding Provider-Based Multi-Tiered Applications

Delphi's support for multi-tiered applications use the components on the DataSnap page and the Data Access page of the component palette, plus a remote data module that is created by a wizard on the "Multi-tier" page of the New Items dialog. They are based on the ability of provider components to package data into transportable data packets and handle updates received as transportable delta packets.

The components needed for a multi-tiered application are described in the following table:

Component	Description
Remote data modules	Specialized data modules that can act as a COM Automation object, SOAP server, or CORBA object to give client applications access to any providers they contain. Used on the application server.
Provider component	A data broker that provides data by creating data packets and resolves client updates. Used on the application server.
Client dataset component	A specialized dataset that uses midas.dll or midaslib.dcu to manage data stored in data packets. The client dataset is used in the client application. It caches updates locally, and applies them in delta packets to the application server.
Connection components	A family of components that locate the server, form connections, and make the IAppServer interface available to client datasets. Each connection component is specialized to use a particular communications protocol.



The DataSnap Protocols

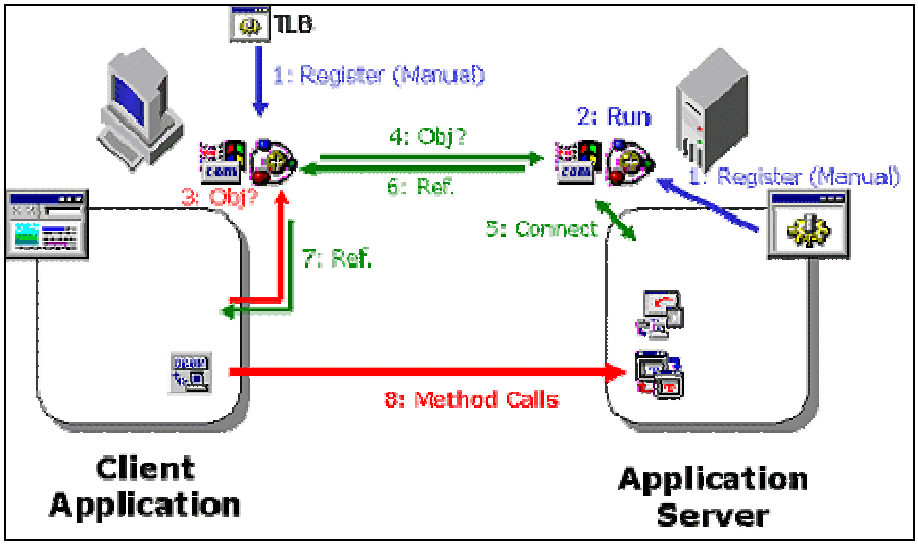
There are several types of connection components that can connect a client dataset to an application server. They are all descendants of TCustomRemoteServer, and differ primarily in the communication protocol they use (DCOM, CORBA, TCP/IP, HTTP, or SOAP).

The connection component establishes a connection to the application server and returns an IAppServer interface that the client dataset uses to call the provider.

This is where we first start discussing design considerations in a DataSnap application. The first concentration for a multi-tiered application is the underlying technology that will provide our remote procedure call abilities. Remote procedure called (RPC) is the term used in our industry to define technology that is capable of calling procedures or functions remotely.

COM/COM+

A developer can create COM/COM+ application servers by creating a "Remote Data Module" or a "Transactional Data Module". We then connect to these application servers by using the "COM connection" component. This methodology uses the Windows underlying COM and COM+ technologies to invoke calls on a Windows NT domain.



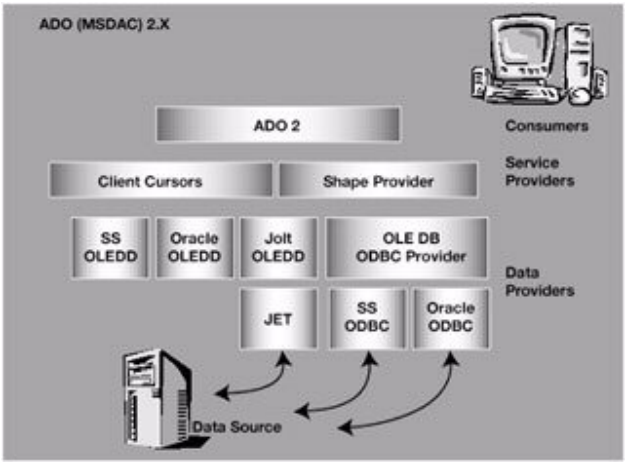
TCP/IP

Note that MicroBanker does not use the TCP/IP protocol since this requires a Borland Database Engine (BDE) to be running on both the client and server computers. The BDE has proven to be less than optimally stable, and requires a larger degree of support than running the COM interf

APPENDIX B – THE IMPACT OF MICROSOFT ADO ON THE COMMUNICATION BANDWIDTH

This appendix is not intended to provide a summary of the Microsoft ADO technology, as the previous appendix does for the Borland technology. Rather, it is intended to explain how ADO adversely impacts telecommunications bandwidth requirements.

The Microsoft ADO engine sits on top of a universal data access mechanism called OLEDB. The ADO engine is multi-layered, with the bottom layer acting as an interface between an OLEDB provider and the application. An optional component used to cache records on the client (as opposed to the database) is called the ADO Cursor Engine. The figure below shows the architecture of ADO. The Cursor Engine is labeled as 'Client Cursors' in the figure.



The ADO Cursor Engine, while primarily responsible for caching fetched records, also includes a component responsible for managing the concurrent updating and deleting of records in the database. It is this component that is responsible for a great deal of the bandwidth consumption in a networked environment.

When the application is running, the ADO Cursor Engine (a component of the ADO protocol) will, during runtime, actually re-write the SQL code written by the developer to ensure that the correct record is actually updated or deleted. In order to do this, it requires a large amount of metadata. In an effort to assist low-skill programmers, the ADO Cursor engine will retrieve a large amount of metadata about the tables, columns and indices in the database correlated to the data requested by the SQL statement.

The processing that results from this 'assistance' is as follows:

1. The client application issues a SQL statement to the database, which is passed as a short text string, such as "Select * from [Transaction] where client code = 'ABCD'"
2. The ADO Cursor Engine, behind the scenes and completely unknown to the developer, issues another statement to the database, requesting all the descriptive information (metadata) about the transaction table. The amount of data.
3. The database sends back the record(s) requested by the statement, which are typically limited in size to 1 or more kilobytes. In addition, it sends back hundreds of kilobytes of metadata. The application receives only the record, while the ADO Cursor Engine captures not only a copy of the record, but also the metadata.

The net result is that while the developer codes and expects the application to only send and receive a small amount of data, typically tens or hundreds times more data is actually passed back from the database. This is the core impediment to implementing the application database remotely from the client application in a wide area network environment.