NORSE TECHNOLOGIES

A Description of the

TECHNICAL ENVIRONMENT

of

NORSE SYSTEMS INC. APPLICATION SOFTWARE

D. L. Hopper
January 2009

I. INTRODUCTION

Norse Systems Inc., in order to provide our clients with extraordinary business success, provides management and administrative tools of extraordinary capabilities. At the root of these capabilities lies our unique technologies.

There are certain basic tenets common to our technology. These include:

- Enterprise Database – Everybody in the organization works out of the same data set. Show us an organization that has people running business critical functions on their PCs and we'll show you an organization where the right hand does not know what the left hand is doing.
- Accessibility – Whether in the office, at home, or on the road, all personnel who can get access to the Internet, can get to the enterprise with an identical security profile as from the office.
- Real-Time Transactions – When events happen, their implications must be public immediately to all personnel. While certain processes, such as posting journals to the General Ledger, may best be processed in 'Batch/on-demand' fashion, most transactions should be reflected immediately. The instant a Charter is booked, it should be visible to Dispatch.
- Knowledge rather than data – Data entry/maintenance processes to be sure must capture structured information. In the case of customer records, Name, Address, Contact People, Phone numbers, email, etc. Many of our competitors attempt to go beyond these minimal requirements by allowing for limited 'Notes' fields, but we let you attach an unlimited set of objects to the record. Correspondence documents, Credit Evaluation Spreadsheets, Photographs, etc. When your personnel open a customers folder (or Student, or Supplier, or Whatever) in the Norse environment, they have at their finger tips ALL of the information that may be relevant to providing proper service.
- Security – While the system is a fully integrated 'enterprise', not everyone get access to it all. Full, easily maintained profile specifications allow you to control of who can see what, and who can do what.
- Respect for Users – Some like graphical screens with mouse navigation while other prefer a 'Green Screen' approach. Each individual user gets their own preference. Likewise some want spreadsheets which the system prepares for them to be MSOffice formats, while others may prefer OpenOffice documents. Again, each person gets to choose.
- T/P Efficiency – The 'tenets' of “Enterprise Database”, Internet accessibility and Real-time transactions, cannot be attained via the sacrifice of 'responsiveness.' If screens as complicated as those needed for Charter Booking, were implemented in '.html', the 'Paint-time' would be so painful your team could not get their days work done.
- T/P Efficiency – Optimized Browsers. Search capabilities which produce sub-second lookup capabilities against tables numbering millions of rows.
- I/T Efficiency – Any/All customization of 3rd party software is done at the server. All user PCs can be configured completely via download/install from public sources.
- I/T Efficiency – Customization and extensions. Powerful and extremely simple ad hoc query capabilities to tabulate data into new reports and/or spreadsheets. Ability to

incorporate such queries into user menus. And easy extension of 'User Fields' in data entry screens.
- Uniformity of Operation – Once you leard to operate one menu or one screen, you know how to operate them all.

Achieving all these 'tenets of design' is not a trivial matter. As stated above, extraordinary business success requires extraordinary tools. And these in turn require extraordinary technologies. The key technology components are as follows:

- EasyCo – Secure/reliable VPN and Virtual Server provider.
- UniVerse – Proprietary multi-dimensional database from IBM.
- AccuTerm – Terminal Emulator/GUI Interface from AccuSoft.
- ScreenDriver – Proprietary Application Service of Norse Systems Inc.

On the pages that follow, we describe each of these technologies, their unique capabilities,and how they combine to provide functionality unheard of (even unimagined) by our competitors.

II. EASYCO TIMESHARE GATEWAY

http://www.easyco.com

Many system designs which centralize around a (remote) corporate database are susceptible to any number of 'opportunities' (read as 'Problems') associated with Communications Reliability and/or Security.

In the case of communications, lesser environments can suffer from line drops, power outages, Internet provider issues, and other anomalous events. Problems range from relatively innocuous 'having to re-do the work', to severe issues such as 'corrupted data' or 'unreleased database locks' which can have the effect of not letting users get back to work.

In the case of Security, lesser environments can be susceptible to hacker attacks, viruses, unauthorized access, or even unauthorized manipulation of data.

The EasyCo Timeshare gateway prevents these problems. It provides 'software-based' (no equipment required) VPN (Virtual Private Network) security as well as insulation from interruption issues. With EasyCo, you cannot, in normal circumstances, get access with a Web Browser, with TelNet, or with FTP services. (Although these can be made available if needed.)

The gateway architecture comes in two parts. The first is local to your branch or office. The second is on the server. At your branch, one machine on your LAN launches the local side of the gateway. This process then contacts your server via the Internet and negotiates one or more communications 'channels'. The number of channels is predefined and the two machines (your LAN gateway machine and your corporate server) agree upon the encryption algorithms they will use. Only after the secure gateway session is active, can it be used to connect to the server.

Users within your branch can now log onto your corporate server. They do so by opening their terminal emulator/graphical interface application (AccuTerm) and establishing a connection (unsecured via your LAN) from their PC to the local end of the gateway consuming one of its channels.

This configuration assures that all communication via public links (the Internet) are secure, but it does much more than that. The part of the gateway running on the server is very intelligent. It can distinguish different types of communication interrupts which might happen and automatically sets up for the re-start process. When communication is restored, and the user logs back in, they are returned to the same screen or menu where they were when the failure happened. If they were within a data entry screen, they are restored to the exact same screen, field, AND CHARACTER, where they were. Database update locks are preserved, and transaction can proceed to 'commit' just as though there had never been an interruption.

III. UNIVERSE

www-306.ibm.com/software/data/u2/universe

At the core of your corporate server lies an extraordinary database product from IBM named uniVerse. In contrast to RDBMS type databases (e.g. Oracle or Sequel Server) it is simple, inexpensive, intuitive, and efficient. Your IT team will NOT have 'job security for life' with this environment. On the performance measure, uniVerse out-performs RDBMS products by a factor of 100 to 1. And for SQL/ODBC fans, “It's in there.”

The 'Tables' within this database are NOT a jumble, within a black box, intelligible only to esoteric software products, licensed annually at outrageous costs, and to DBA staff salaried at equally outrageous costs. Each Table corresponds directly to a single file in the OS directory. Each has associated with it a data-dictionary to support SQL and other query tools.

Each record in each table is accessed directly via a record key. For example 'Part' records are retrieved from the 'InventoryFile' via a 'PartNo.' Customer records are retrieved from the 'CustomerFile' via a 'CustomerNumb', etc.

This is NOT necessarily the case in RDBMS environments. In some RDBMS schema, 'CustomerInvoice' records might be accessible ONLY via a 'Pointer Chain' from a 'parent' 'Customer' record. -- handy for running 'Customer Statements' but horrendous for processes not envisioned by the original schema designer. What if we want to query invoices evincing sales of peanuts to anyone?

In addition to discrete files, and discrete access keys, uniVerse provides other extraordinary capabilities. It is well beyond the scope of this document to describe the database in detail so we'll highlight just a few others.

- Variable record and field sizes. -- The 'body' of any data record in any database table is just a 'string' of characters. 'Fields' within the records are delimited by a special character called an 'Attribute Mark'. (Represented by an '@' symbol in examples shown in this document.) If we are talking about a customer record, and the first field in the record is the customer name field, then the customer name is all of the characters from the beginning of the string up to the first attribute mark. If the customer name is 'ABC SUPPLY' then we are storing 10 characters. If we enroll a new customer whose name is 'ALPHA BETA CONGLOMERATE AND WAREHOUSEING SERVICE CORPORATION, INC.' that's just fine. The database doesn't care. (As you'll see in the SCREEN.DRIVER section below, the data entry screens don't care either. The system imposes no size limitations. Note that the dictionary of the customer file may specify that the 'Name' field has a width of 20 characters, but even this is no problem. If ALPHA BETA gets listed as a line on some report, it will actually be shown in its entirety, in a column 20 bytes wide. It will simply consume 3 lines.
- Multi-Dimensional. -- This is also known as tables-within-tables. Just as 'Fields' within a data record are delimited by attribute marks, there can be multiple values WITHIN a field. They are simply separated by a different special character called a 'Value Mark'. (Represented by a '\' symbol in examples shown in this document.) Fields containing multiple values may or may not be 'correlated' with multiple values in other fields. As

an example lets think about an Inventory record. The record key is a part number. Field 1 is the part description, field 2 is the warehouse code, and field 3 is the quantity on-hand. An entire 'table' of part availability can exist within the single part record, as follows:

10W30 SAE OIL@A\B\C@10\20\5

Part 10w30 is engine oil, and we have 10 units in warehouse 'A', 20 units in warehouse 'B', and 5 units in warehouse C.
In our architecture, with one 'database fetch' we have the entire view of the situation relative to this part. Not only warehouse availability, but also pricing by customer class and quantity breaks, Sales history by period, suppliers and supplier prices, etc. In an RDBMS schema it would likely take 100 or more 'database fetch' operations to assemble the comprehensive view.

- Dynamic Table 'Joins' – In both RDBMS and uniVerse environments, data is frequently split between files. For example, an Invoice file may have a key of InvoiceNumber, Field 1 is InvoiceDate, 2 is CustomerNumber, 3 is InvoiceAmount. In either environment, if we wish a query, or report, which lists InvoiceNumber, CustomerNumber, CustomerName, and InvoiceAmount, then we have a problem. We are reporting from InvoiceFile and 'CustomerName' is NOT there. We have to do what is called a 'Join' of InvoiceFile, with CustomerFile, keying on CustomerNumber. RDBMS/SQL does this by copying data from both tables into a workspace and then reporting from the workspace. The memory consumed in these processes frequently leaves nothing left for regular users thus causing the entire system to run intolerably slow. In uniVerse, by contrast, a 'Word' exists within the InvoiceFile data dictionary for all four of our fields INCLUDING 'CustomerName'. Query processing does NOT consume memory and T/P operations are consistently fast.
- Locking Strategy – It almost seems silly to mention this, but we have seen would-be competitor packages, which still manage contention by file-level locks. These are supposedly centralized enterprise databases, frequently '.NET' based, with a single enterprise database somewhere accessible on the LAN. True, the database is accessible by all, *however*, some packages have been known to produce messages like “Charter Booking is being run by someone else, Please try later.” The wealth of lock-tools provided by uniVerse enables us design applications such that contentrion can occur ONLY at the unit record level.
- Secondary Access Keys – The strategic key behind our browsers ability to provide sub-second search results, across millions of records, stems from this intrinsic uniVerse tool. Most key tables have several secondary indicies enabling instantaneous view by 'non-key' access methods.
- Scalability – While there are not yet any Norse clients with requirements to do so, the uniVerse/ScreenDriver combination has been shown to scale efficiently to levels at the hundreds of millions of records, being accessed 10's of thousands of users. (3000+ concurrently active in a highly complex medical records application.)

IV. ACCUTERM

www.asent.com

The user's interface to the Norse System application is via a Terminal Emulator/GUI Interface product named 'AccuTerm'. The product is from AccuSort Enterprises and provides a wealth of capabilities. First it provides for session initiation and TelNet connection to the local gateway, and thus to the corporate server. Secondly, it provides access to the M/S API (Windows DNA) on the users PC. This enables host-based uniVerse programs to manipulate windows, devices, and services. In other-words AccuTerm becomes the basis of the user interface per se.

V. SCREEN.DRIVER

ScreenDriver is a proprietary product used and marketed exclusively by Norse Systems Inc. It represents years of evolution and was the first 'Application Services' product, in the multi-dimensional database world, to embrace the philosophy of Object Oriented (OO) design. This is the philosophy which says to the programmer, 'If you did a copy and paste, you wrote it wrong."

There is only one data entry program – SCREEN. This 'Method' can be applied to any properly defined database table. There is only one menu program – MENU. This 'Method' can be applied to any properly defined item in the MENUS file.

Obviously, these programs/methods are smart. They look at, and react to, elements and preferences which the user defined at sign-on time, they look at enterprise environment variables set by system administrators, and they look at security profile settings set for the user by administrators. Not all uses necessarily see the same Maintenance screen the same way.

The OO implementation here is simplistic. Objects do not have a class hierarchy, and they cannot inherit behaviors. These 'methods' have a one-to-one correspondence to database tables where there are definitions which they can instantiate. As such these methods can be equally thought of as 'utilities' in legacy programmer parlance.

The following table lists principle 'Methods/Utilities' which ScreenDriver offers, along with the 'Container/DataBaseTable' names which codify objects which can be instantiated.

Container(Table)	Method(Utility)	Description
SCREENS	SCREEN	Structured Data Entry Screens
MENUS	MENU	Application Software navigation menus
UMENUS	MENU	User defined (or modified) navigation menus
FileSelect	GenFileSelect	Generic Data Extraction (sub-set) table retriever
Browse	Browse	Find and Item (record) without knowing it's key
Menuobjects	MOX	Menu Object Xecutive – all selections, all menus
Operators	OperatorExport	Implement UserID Maintenance at the OS level
DDD	BuildDict	Implement revised Detail Database Definitions
DDD	BuildINC	Reconfigure Table Handlers
DDD	BuildIndex	Reconfigure Secondary Table Keys
ScrForms	FormsReport	Forms Printing with or without '.jpg' image overlays
FontSpec	<Program Only>	One class of Form Elements
FileName	<Program Only>	Name xRef table used in Table Handlers
ParaLib	ParaRun	Scripted 'Paragraph' procedures

AppImportCtrl	AppImport	Update client Corp server from Norse Corp server
CtlEntryDoc	Help	Documentation re System setup tables
<Program Built>	GetArgs	Self designing User Argument collector
GetArgsHelp	Help	Documentation re 'GetArgs' events
EZQ	EzqGuiBuild	GUI Query Designer (Dict Word Drag/Drop)
EZQ	EZQ.EXEC	MenuObject Query call method
ItemNotes	NotesManager	New/Edit/Link/De-Link text items attached to records
ObjIndex	Attachments	New/Edit/Link/De-Link DOS items attached to records
Tickler	JournalManager	New/Edit/Link/De-Link/Comply w/ ToDo assignments (or make LogNotes) attached to records.
Various	ListLog	Review Application, Admin, and Software logs
SysMail	SendSysMail	Active Window e-mailer

As with any OO structure, simple events take the form a sequence of steps, most of which are implementation of generic methods. For Example, within the Accounts Payable module, there are menus which include selections to 'Analyze Cash Requirements', 'Print A/P Aging Report', and 'Print A/P Checks' If our user selects the line that says 'Analyze Cash Requirements', then the MENU method which is in control and will respond to the Operator 'Click' event may start a chain. The MENU method would instantiate a MenuObject named 'APCashReq' and then trigger it's launch 'MOX' method. The MOX Method may call for a ParaLib object to run ParaRun. The first step of the ParaRun may be to invoke 'GenFileSelect' against the AccountsPayable Table. The first Step of the GenFileSelect may call for a 'GetArgs' event, whereby the operator can specify the company, bank code, range of dates, etc. While the GetArgs method is waiting for parameters, the operator may click the 'GetArgs' window 'Help' button. The 'Help' method is then invoked against and object instantiated from the 'GetArgsHelpTable' a so a help window appears relative the FileSelect Object from which GetArgs launched. After the Operator run-time arguments are entered, The 'GenFileSelect' method resumes control and makes a list of all payable invoice numbers which meet the Operators criteria. Once the list is complete, control returns to ParaRun method which calls for instantiating an object from EZQ and invoking it's EXQ.EXEC method. The EZQ Object definition may call for finding a pre-existing InvoiceList which the ParaObject now has access to. Then the Date sorter. Then the SpreadSheet Tabulator, Then the SpreadSheet Transmit to the client PC. Then then the appropriate spreadsheet program invoke on the client., etc. etc. etc.

If instead of choosing 'Analyze Case Requirements' many of the steps and methods would be identical. A MenuObject, MOX, ParaRun, GenFileSelect, GetArgs, ...

Application Development and maintenance in this architecture then becomes a process of configuring Object Definitions. In almost all cases, the object definition items found in these various containers, are maintained with the SCREEN method.