CASE STUDY: Migration Project

One of Sweden's largest banks (in terms of number of customers) also has a leading position in additional markets covering Estonia, Latvia and Lithuania. The bank is a modern inclusive bank, firmly rooted in Swedish savings bank history, serving 7.8 million private customers and more than 600,000 corporate and organisational customers. In 2010 the bank decided to embark on a migration project for their Management Accounting and Reporting System – MARS – from an APL2 Mainframe application to a Dyalog Microsoft Windows Server platform. The responsibility for the migration project was placed with Gothenburg-based Aplensia AB and that company's experienced team of APL programmers. Completed in 2012, the migration not only cut the bank's hardware running costs substantially, it also doubled the performance! Jonas Stensiö takes us through the process of the successful MARS migration project.

"The bank that requested the project has a somewhat atypical organisational structure because the corporate group consists of a large number of companies. More to the point, the bank is connected to a number of other stand-alone banks with whom it shares its IT resources and software applications. This was something we needed to be very aware of and take into consideration before we embarked on the migration project" Jonas Stensiö explains.

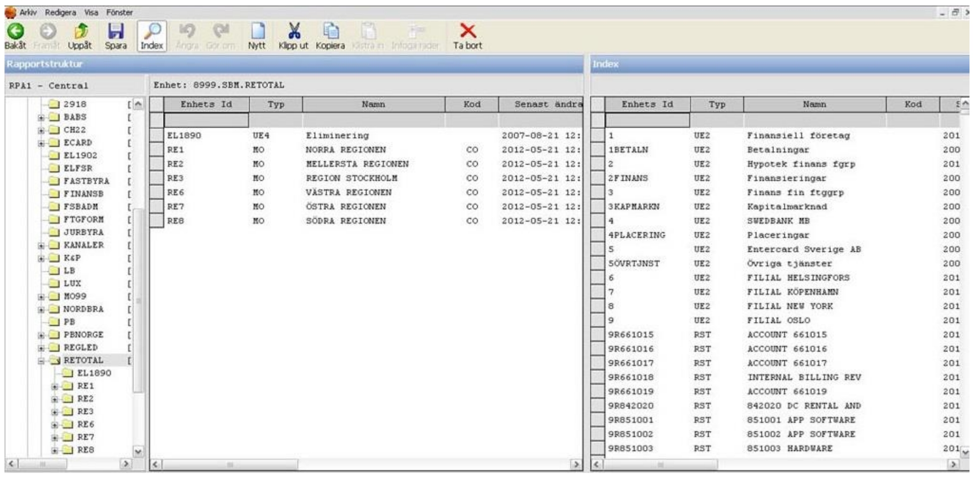
He continues, "The MARS system is used for management accounting and reporting. Most of the reports produced by the system are for internal use, but MARS also delivers data that is used to produce the external financial statements. In addition to this, the system supplies the entire organisational structure, and is used throughout to produce reports for different organisational units and levels. It's a very comprehensive system developed with 260,000 lines of code. It serves 4,500 end users (who use the system to view reports) and 8 administrators (who are responsible for designing and producing reports). It conducts 350,000 transactions per month, and at peak times can reach 35,000 daily transactions."

The complexity of the bank's organisation is indeed very comprehensive. The entire structure is divided into units on several levels, with cost centres at the lowest level. Jonas Stensiö explains "The organisation changes regularly, both because of internal re-organisation and external events. As the organisational structure is used by our reporting system to create reports for different units, as well as reports with drill-down functionality, we need to have a certain amount of flexibility in the system. Add to this that the reporting structure is also used by many other systems in the corporation, for example, access management. In our case, the access management system limits users to *view only* reports for units that they are authorised to evaluate. Beside this, we are maintaining alternate organisational structures that are used in special purpose reports, as not everyone looks at the organisation in the same way. The entire organisational structure is *frozen* for a snapshot picture once a month. Copies of all frozen structures are saved. This is really important, because sometimes an older structure needs to be used later in order to compare reports from different months and our solution makes it really easy to make these comparisons. Finally, we also operate with other types of structures that can be used in reports, for example, account structures. Structures can further be used as sub-structures or even be merged in some situations. So as you can gather, we need to add order to what is otherwise a pretty chaotic picture."

The History of MARS

The old MARS system dates back to the 1980s and was then a purely mainframe based application. In 2001 a Microsoft Excel add-in was created, which enabled end users to get reports directly into Excel, whilst administrators continued to work in APL sessions and create reports. In 2006 it was decided to create a new Microsoft Windows user interface for the administrators. The new user interface contained all parts of the old user interface, although most functions were redesigned with greater usability in mind. Since then, many new functions have been added. By 2010 the bank decided that the time had come to embark on a full migration project to a server-based application. The main reason for wanting to conduct the project was to save costs, and Aplensia was retained to migrate MARS to a Dyalog Microsoft Windows server-based application.

Jonas Stensiö says "We had the advantage that the client application for the new user interface back in 2006 was written in Dyalog. It's a classic Microsoft Windows MDI application, but it consists of many sub-applications that have been developed over the years for various purposes. The idea with the new user interface was mainly to make it easier to edit the organisational structure, and make it look like Windows Explorer. Units can easily be moved from one place to another. Undo/Redo functionality is available everywhere in the application. We also needed to make it easy to design and create reports. As mentioned before, we're working with an ever changing organisation with a variety of reporting requirements, and we needed to be able to find and compare old reports for previous months on occasion. In order to do that, we make heavy use of the Grid control, so anywhere that information can be presented in table form we use the Grid control."



RPA1 - Central

```
2918
BABS
CH22
ECARD
EL1902
ELFSR
FASTBYRA
FINANSB
FSBADM
FTGFORM
JURBYRA
KANALER
K&P
LB
LUX
MO99
NORDBRA
PB
PBNORGE
REGLED
RETOTAL
  EL1890
  RE1
  RE2
  RE3
  RE6
  RE7
  RE8
```

Enhet: 8999.SBM.RETOTAL

Enhets Id	Typ	Namn	Kod	Senast ändra
EL1890	UE4	Eliminering		2007-08-21 12:
RE1	MO	NORRA REGIONEN	CO	2012-05-21 12:
RE2	MO	MELLERSTA REGIONEN	CO	2012-05-21 12:
RE3	MO	REGION STOCKHOLM	CO	2012-05-21 12:
RE6	MO	VÄSTRA REGIONEN	CO	2012-05-21 12:
RE7	MO	ÖSTRA REGIONEN	CO	2012-05-21 12:
RE8	MO	SÖDRA REGIONEN	CO	2012-05-21 12:

Enhets Id	Typ	Namn	Kod	S
1	UE2	Finansiell företag		201
1BETALN	UE2	Betalningar		200
2	UE2	Hypotek finans fgrp		201
2FINANS	UE2	Finansieringar		200
3	UE2	Finans fin ftggrp		200
3KAPMARKN	UE2	Kapitalmarknad		200
4	UE2	SWEDBANK MB		200
4PLACERING	UE2	Placeringar		200
5	UE2	Entercard Sverige AB		200
5ÖVRTJNST	UE2	Övriga tjänster		200
6	UE2	FILIAL HELSINGFORS		201
7	UE2	FILIAL KÖPENHAMN		201
8	UE2	FILIAL NEW YORK		201
9	UE2	FILIAL OSLO		201
9R661015	RST	ACCOUNT 661015		201
9R661016	RST	ACCOUNT 661016		201
9R661017	RST	ACCOUNT 661017		201
9R661018	RST	INTERNAL BILLING REV		201
9R661019	RST	ACCOUNT 661019		201
9R842020	RST	842020 DC RENTAL AND		201
9R851001	RST	851001 APP SOFTWARE		201
9R851002	RST	851002 APP SOFTWARE		201
9R851003	RST	851003 HARDWARE		201

The New User Interface

Reports produced by the system are either simple text files or Microsoft Excel reports. The report designer looks much like Excel and is presented to the user as a grid of cells. Cells can be one of three types; simple text (or numeric), a calculated value (from other cells and simple functions) or a value retrieved from some data source. This last type is the most interesting as the cell is defined to call one of many predefined data retrieval functions. Depending on which function is used, the cell can take different arguments. The arguments can be hard-coded or can be linked to an argument of the report (which has to be specified at report run-time). An argument can also be a link to a structure, or a unit in a structure. More complex possibilities also exist. A special data retrieval function has, therefore, been developed to retrieve data from other reports.

Jonas Stensiö picks up the tale. "The administrators of the system are designing the reports. But the data retrieval functions are maintained by the system developers. New data retrieval functions are added when needed, for example, when new data sources need to be used. For drill-down reports, the value of a cell is an array if the line it is on is drillable. When the report is executed, data for each cell is retrieved in an efficient way. All cells that use the same data source (table) are simply fetched together. Reports can be produced on demand. But, as many reports are used by many users, reports are also pre-produced both to save execution time as well as to be more responsive to the users. Users can subscribe to receive groups of reports. In such cases, e-mails are sent out when the reports have been produced."

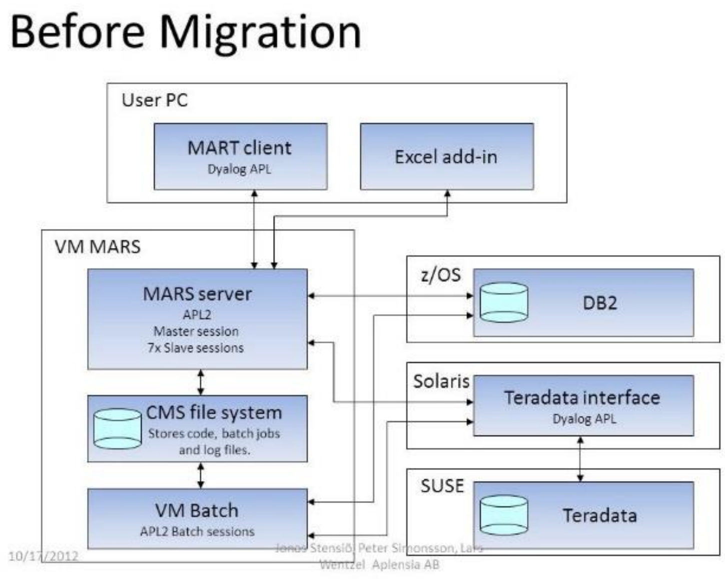
It is the responsibility of the administrators to make sure that reports are pre-produced. To aid the administrators in this task, a function called the *Scheduler* is included in the tool developed by Aplensia. A schedule is a list of batch jobs that are to be executed one at a time. Each batch job can, in turn, contain a list of sub-batch jobs to be executed in parallel. Every month a number of schedules are started by the administrators, some to run at pre-defined times. Many reports are dependent on data being loaded into data tables. This must, therefore, be determined before a schedule can be started. Data tables that have finished loading generate a signal, and schedules can be configured to prompt-start on such signals.

Fokus and Cost Distribution

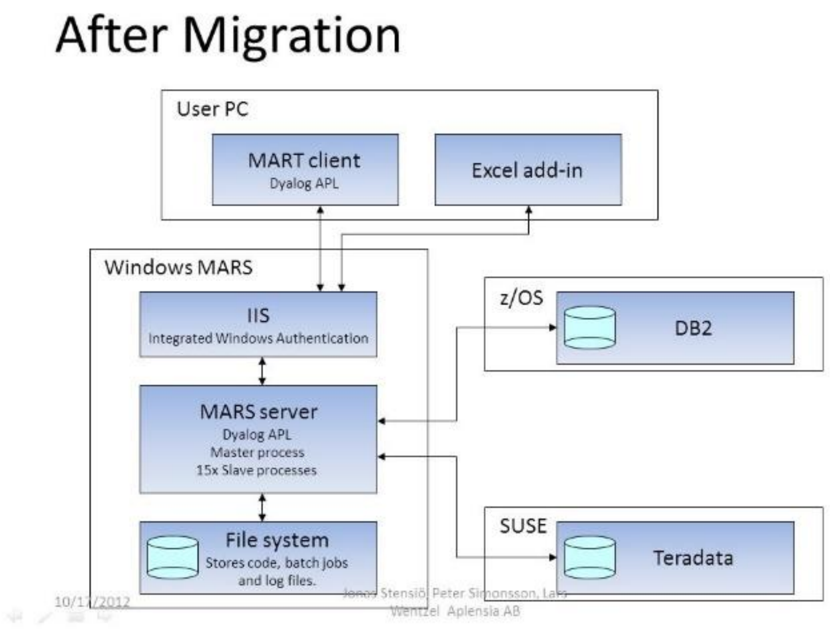
Fokus – which means "Focus" in Swedish – is an application developed for all sales personnel in the Swedish bank as well as the stand-alone banks. As is the case with other sales, staff are measured on all products they sell; Fokus provides an easy means to see the result on both personal and aggregated levels.

"Every Sunday we get the latest numbers for the sales from the previous week, and our batch job for creating all the reports starts. When it's done, the users (sales personnel and managers) get their reports. Every month, a monthly report is further created manually by our administrators. Administrators are also responsible for updating the reports that Fokus is based on. The tool selected for getting the reports is a Microsoft Excel file, where a lot of formatting is done to make the reports look nice and easy to evaluate. As we're creating reports for approximately 3,000 users for the Swedish bank itself and 2,000 users for the stand-alone banks it is no small job – we're literally talking about users fetching approximately 20,000 reports per month! Additionally, every quarter, new Excel files are created – very often a lot changes from the previous quarter," Jonas explains.

Another task that the administrators are involved in is internal cost distribution. The system is used to refine data from the underlying base system. Simply put, costs can be distributed between many units. Tools are available in the system to do this with distribution keys that are maintained by the administrators. Calculation of internal interests and the net interest income is done in the system. Interest costs and incomes are calculated for each unit. The results of these calculations are available to be used in reports. Results are, therefore, also delivered to other systems and other parts of the business group.



Before the migration the client applications communicated with the APL2 server. Requests were sent using the HTTP protocol without encryption. Security was poor because the user ID was sent along with the message and authorisation was based on that. The server consisted of 8 APL2 sessions working in parallel to handle requests. Some requests could initiate batch jobs. Batch jobs were created in the file system and handled by VM Batch sessions. The system used two shared data sources. One of them was a DB2 database that was directly accessible from VM using an IBM SQL workspace. The other was Teradata, which could not be directly accessed from VM. To access Teradata, Aplensia used a Dyalog server. The server was running on UNIX and relayed SQL requests to Teradata using the Dyalog SQAPL interface.



After the migration Integrated Windows Authentication is used to increase the application security. "When we started there was no support for this in Conga. Therefore, we decided to use IIS on the server. Client applications communicated with IIS which, in turn, relayed the requests to the Dyalog server. IIS handles all the security. Messages are encrypted and the user is properly authenticated with their Windows user ID. This is a single sign-on system, so the user does not need to log on again when using the applications. The Microsoft Windows server consists of 16 Dyalog processes. We have implemented a Windows service that is responsible for starting all the processes and keeps them running. Since there is no VM Batch in Windows, we had to implement our own batch handling. We did this in the Dyalog Server so batch jobs are executed just like regular requests. Finally, we used the Dyalog SQAPL interface to access our data sources. This eliminated the need for the UNIX server," Jonas Stensiö says.

NOTE: Conga 2.3, part of Dyalog version 13.2, includes support for Integrated Windows Authentication (IWA). IWA is one mechanism that can be used to provide Single Sign On (SSO) under Windows. With IWA, a user's Microsoft Windows domain credentials are used to authenticate the user to a server without having to enter another set of logon credentials.

Load Balancing and Batch Handling

The server uses the Master-Slave model. One master process controls 15 slave processes. When a request comes in, the header is received by the master. The master enqueues it and assigns it to a slave when one is available. The slave then fetches the header and the body of the message. The body of the message never goes through the master. All valid requests are categorised as being *slow* or *fast* jobs. A *fast* job should not take more than a few seconds to execute. *Slow* jobs, on the other hand, may take some minutes and are more likely to be waiting in queue.

Batch jobs are categorised as a third job type. The master has three job queues, one for each type. Each queue has a dedicated set of slave processes. This ensures that jobs that are categorised as *fast* never get queued up for very long. This also ensures that batch jobs are executed independently of regular requests. The batch queue is stored in the file system. In case the server goes down – or is restarted – the batch queue is restored and the jobs are resumed. A job can create new batch jobs. In this case, the newly created jobs become children of the first one. The parent is not considered to be completed until all the child jobs have been completed.

AXML – XML formats for APL Arrays

Aplensia had to develop an XML format for APL arrays. According to Jonas Stensiö "We call this format AXML and it can serialise any regular APL array. The only exception is arrays containing objects or other types of pointers. The format was developed to simplify messaging between our Dyalog client and the APL2 server. A request may contain an AXML serialised array in the body of the message. When communicating with APL2, one needs to take the different character encodings into account. We chose to implement our own character translation functions for this. This format is also used in communication between our server processes. Furthermore, the format is used to store data in text files. Now that we no longer communicate with APL2 we have replaced the character translation functions with UTF-8 encoding.

"Version control is another issue we needed to address. We use Subversion for version handling of workspaces. For this we have developed a dedicated tool. Code is divided into modules, and in our Subversion tool a module is a directory. A module can contain functions, classes, variables and subdirectories. When loaded into Dyalog each module has its own root namespace. Sub-directories represent sub namespaces. Modules can have dependencies on other modules. When a module is loaded, all of its dependents and their dependents (and so on) are also loaded. Functions and classes are stored as regular text files. This is done in the same way as in SALT."

NOTE: SALT stores data (variables, namespaces) and code (programs) in UTF-8 encoded text files. It can read those files back to redefine the objects in memory.

"Variables are stored as XML files using our AXML format. A session is started by loading a module, the project main module. When this happens, a workspace is created dynamically. Different projects can share the same code modules. When changes are made in the workspace, the tool keeps track of those changes. Saving the workspace means saving those changes. Working with the tool is much like the regular way of loading and saving a workspace. With the tool you can also make commits, full or partial, directly from within the Dyalog session. If a module is loaded, namespace names can be used to specify what to commit, which is very handy," Jonas Stensiö explains.

To Conclude – the Motivation, the Objectives and the Result

"The migration project was primarily initiated and motivated by a desire to cut costs," Jonas Stensiö says. "We also needed to ensure that performance would be at least equally good. Moving into a modern environment was a strong desire, but we still needed to ensure that code could be version handled. Lastly, there was a strong requirement for using secure methods for authorisation and authentication – and messaging.

"We're very pleased to report that the results have exceeded our expectations. The hardware costs were reduced from €1.4 million to less than €200 thousand – that's a saving of a massive €1.2 million! The performance has roughly doubled – which is much better than anticipated. The new server runs in a virtual machine and it's very robust indeed. It has already been moved once from one data centre to another, and the move was performed without any downtime," Jonas Stensiö concludes.