CASE STUDY: Automotive – Production Schedule

Hercules is a system used by the Volvo Car Corporation (Volvo) for planning car production. Its main output is the monthly Master Production Schedule (MPS) – a detailed plan on how many cars are to be produced per market, factory and week during the following 13-15 months. Migrating such a comprehensive and vital system from an APL2 Mainframe to a Dyalog Microsoft Windows platform is no small task. Working with experienced and expert APL programmers from Aplensia in Gothenburg together with Mahindra Satyam, the project was completed in record time, with no disruption to the users or the operations, saving a substantial amount in the process. Peter Simonsson from Aplensia takes us through the process of the successful Hercules migration project, which was completed in the autumn of 2012.

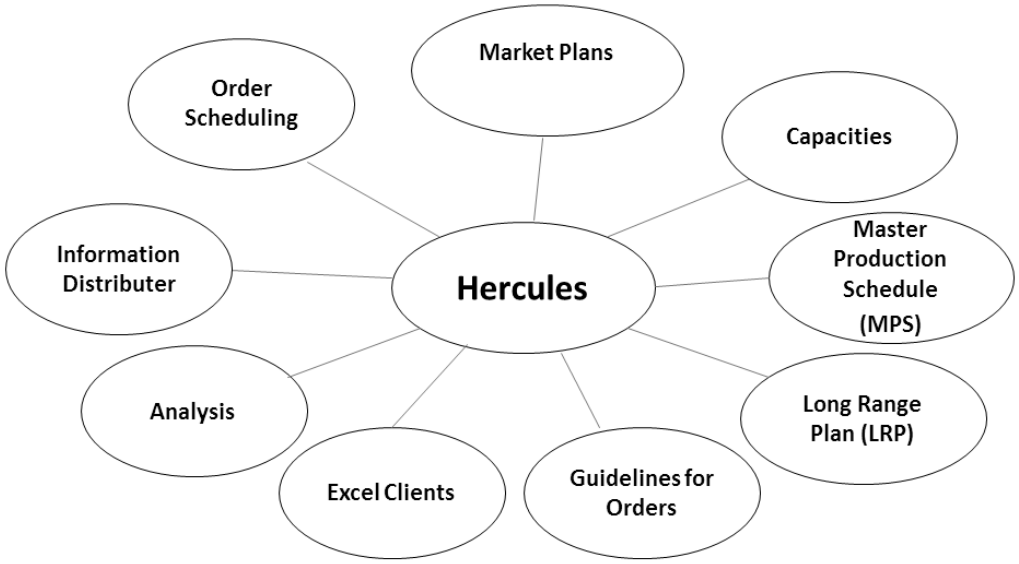
"Outside of producing the monthly Master Production Schedule (MPS), based on market requirements, available production capacity and material supplier capacity on vital components, Hercules also performs a number of other key functions such as long range planning (up to 9 years), handling of production and material capacities used for order slotting. The system also performs checks on the forecasted volumes. The MPS is subsequently broken down into Bills of Materials, which is used to send out forecasts to the suppliers," Peter Simonsson explains.

When you have an annual output of around 450,000 cars (2011) from production facilities in Gothenburg, Ghent (Belgium) and Chengdu in China, it takes very little imagination to understand the potentially catastrophic consequences of a Hercules system failure.

"Hercules is a very large system. Not so much in terms of users – the solution is only used by around 300 expert users plus a few analysts. However, with more than 425,000 lines of code, 468 user screens, 290 tables, integration with 17 other systems, use of 25 online web-services towards the product data system on orders for handling all the car configuration rules, as well as collecting information from other systems such as sales statistics, you could say that we're talking about the heart and the virtual soul of Volvo Cars. It was, therefore, very obvious that we had to plan the migration project very carefully to ensure we didn't accidentally cause any disruption to the system," Peter explains.

The Need to Migrate

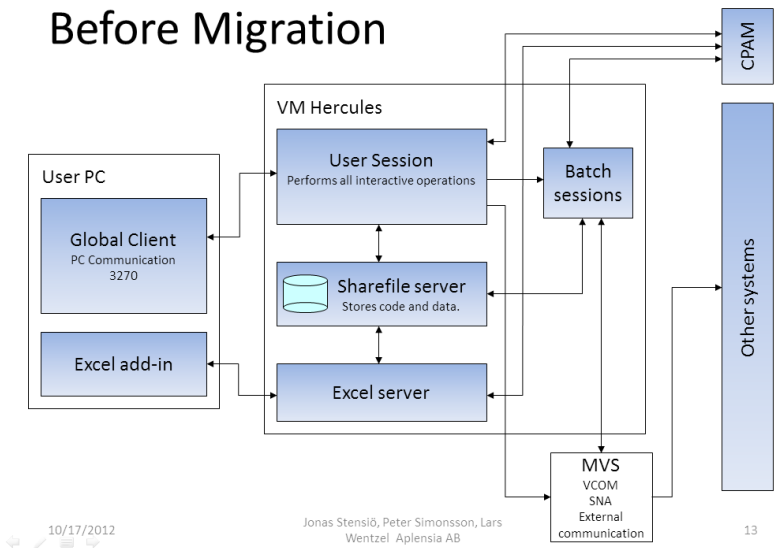
Hercules is an old system that has worked admirably for more than 20 years. It was originally based on IBM Mainframe VM and APL2. Over the years, the system has been enhanced and updated continuously, but by 2012 the technology it was based on was declining rapidly, which made support more and more difficult.

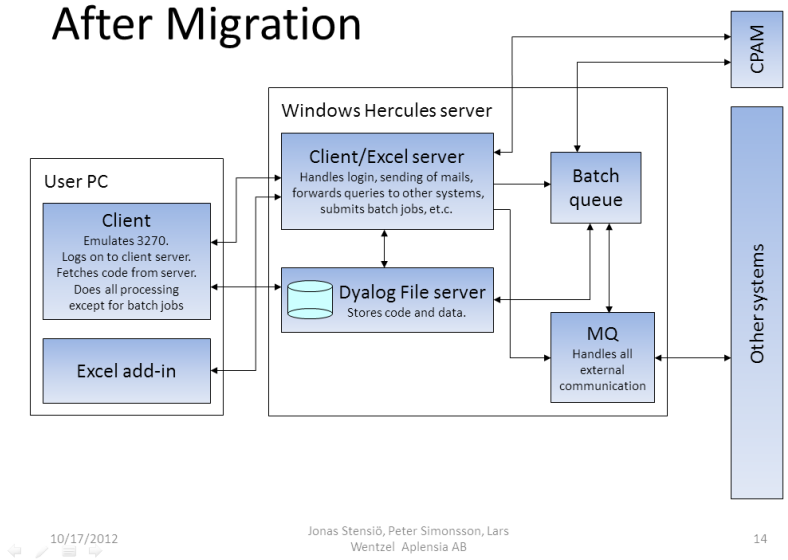


Peter says "The main background for wanting to undertake the migration in the first place is one of declining technology and support. The upgrades of APL2 were few, and the system was using several components for which we were getting poor or non-existent support. Coupled with this was the fact that running costs in the mainframe were getting rather high, and we knew from experience that we could run an APL system much more cheaply on a Windows server. Volvo Cars also wanted to harmonise the APL platform used in Hercules and other important APL applications throughout the company in order to better support them in the future. However, there were a couple of important things we needed to take into consideration. First and foremost, Volvo Cars didn't want a new system as such as the functionality of Hercules was working well and thus needed to be kept. Also, there was no question of changing the Graphical User Interface (GUI), so we would need to retain GDDM (IBM Graphical Data Display Manager) and 3270 screens. The decision, therefore, quickly focused on securing the platform and migrating the system, rather than re-writing it from scratch – and to do it quickly!"

The Road to Moving Code

Aplensia decided to propose a project where they would migrate the code from APL2 to Dyalog, although this is not completely straightforward as there are syntax differences. The team also wanted to fully utilise some of the powerful modern facilities in Dyalog. To move forward quickly, the team decided to keep the more than 400 screens, so they needed a 3270 emulator in Microsoft Windows. The rationale behind this decision was prudent, as it is always possible to introduce a new GUI step-by-step in the future when required – and when the budget allows for it.





"We decided to use thick clients, which means that lots of things are processed on the users own work stations. The old Hercules system was using an old Sharefile system from Manugistics so we needed a replacement and decided to use Dyalog File Server. Although we did look at using Dyalog component files directly, the reason for not going down that route was mainly that we needed to protect the files," Peter says.

He continues "We also had to find a new way of solving authentication. For the integrations in Hercules which are sending and receiving files, we wanted to get out of the in-house VCOM system and move to the new standard WebSphere MQ, and this required a bit of extra work. Lastly, we needed to solve the batch process handling, so we copied and adapted a solution we'd done for another Dyalog system, which we knew worked."

Before the migration the users logged on to their own user session in VM using IBM Personal Communications. For some specific tasks there were Excel add-ins that allowed the user to pull data from Hercules into Excel, work with them there, and then send the data back to Hercules. The Excel add-ins were serviced through HTTP-queries by four server sessions running in VM. There were also a number of sessions to handle batch jobs and processing incoming files. Communication with other systems went through MVS using a mix of different technology; however, most of them used the in-house VCOM. All data storage, as well as the major part of the application code, was stored in Sharefile.

After the migration users log on by starting an executable application installed on their PC. The executable fetches all the required code from the server, thus it has no application logic saved locally and only needs to be replaced for Dyalog updates. The users' fat client processes all data (except batch jobs) locally, and the server is only contacted to perform tasks that the client cannot do itself such as sending e-mails, sending queries to other systems, submitting batch jobs and so on. Sharefile

has been replaced with the Dyalog File Server. A solution for batch queue handling has been implemented and all external communications go through WebSphere MQ.

Getting Serious with Code and Data Transfer

Aplensia used SCAR (Self-Contained Array) to transfer code and data from VM to Windows. This meant creating a set of functions to read the Sharefile component files, covert them to SCAR using the ATS function supplied by IBM and writing the result into CMS files. The files were then sent over to Microsoft Windows through FTP, picked up on the Windows side, converted into Dyalog using the SCAR functions in Dyalog's SQAPL workspace and saved into Dyalog component files.

"The configuration file – **aplunicd.ini** – had to be tuned somewhat to get all the characters correct, but once we'd done that, this turned out to be a very quick way of moving data. Total time to transfer 30 GB of production data and converting it into Dyalog component files was about 6 hours," Peter says with a satisfied smile.

"For the transfer of code we made use of the same functions, but we added an additional step where we converted the code into transfer format using the `⎕Tf` function in APL2. The transfer format of the code was written into the component files that were transferred and read into Dyalog, where variables and functions were created by simply doing an execute on the transfer format strings. In order to save time on the "go live" weekend, we did a full transfer one week ahead, saving a record of files transferred and the update timestamp each file had when transferred. This meant that on the actual "go live" date we only needed to re-transfer files that had been changed since the complete transfer one week earlier."

Migrating the Code, Code Tooling and Testing

According to Peter "We wanted to migrate the code as automatically as possible. However, there are several language differences between APL2 and Dyalog which we needed to take care of. The three main differences are; selective assignment in Dyalog is more limited, there are differences in some of the functionality such as formatting and error trapping, and we wanted to use namespaces for groups of functions. The latter meant that all our basic functions needed to be re-named and all calls to them needed to be changed. But once we had that in hand, we developed a tool that automatically changed the code. For some things this was easy – for example, selective assignment. For formatting and error trapping it was more difficult. The tool we developed, therefore, aimed to find all these special situations and give diagnostic messages for them, and then they were written manually. We also re-wrote the code generator to generate code with correct Dyalog syntax and use the new namespaces and naming conventions."

Testing was a bit special. The basis for the testing was that the original code was correct and that the logic was there from the old system. But since the team had to change part of the code manually, it was necessary to test that this was done correctly, and they basically ran through all or most lines of codes to be on the safe side.

"At the real low level of testing we did actually run through every single line of code," Peter says. "This was done with our basic and standard functions – all in all 600-700. We used a small set of functions utilising the `⎕MONITOR` in Dyalog. On a higher level we concentrated on a core set of

interactive user functions – dialogs. These were tested using an extended and more interactive tool also using ⎕MONITOR. System integration was obviously important since we had changed all files sending and receiving data to use MQ. This meant a lot of work with the other systems and with the Integration Centre at Volvo Cars IT department. A large part of this work was testing. Finally, we conducted user testing with a reference group. This group also prepared the remaining users for the switchover. The users tested the 3270 emulator and the performance and they also tried to catch any remaining bugs. The last two weeks the old and the new system were running in parallel with the same data.

"As I mentioned earlier, we decided to keep the more than 400 screens using a 3270 emulator. However, the users were also used to working extensively in Excel via Excel add-ins in the old system, so there was a need to get data from the new application into Excel. One way is to use Excel add-ins; another is export of data from the application screens to Excel. In VM it was done by sending the data in Excel format (or actually tab-separated values saved as .XLS) as an email attachment to the user's inbox. With the move to Windows we could instead make use of the Excel OLE object, thereby adding an option to directly open an Excel sheet filled with data. This also added the possibility to do formatting on the produced Excel sheets."

The Dyalog File Server – DFS

Aplensia needed a replacement for Sharefile/AP and, whilst the component files in Dyalog provide almost the same functionality, there was a need for a secure and robust client/server setup. Aplensia turned to Dyalog Ltd who helped with the development of the Dyalog File Server (DFS).

"Once we had DFS we only needed minor code changes in the application. There were changes in the calling of the F-functions (FTIE, FREAD and so on) but we also needed to make some adaption of the code that made excessive amounts of data accesses. In VM the access of data was local and hence large amounts of small reads took no time at all. But since DFS sends data over the network, those small reads gets costly. It was necessary to test extensively, because DFS was being developed in parallel with our migration project, and partly based on our requirements. The testing included functional and performance testing, as well as stability and load," Peter explains.

"For authentication we took a lesson from Dyalog Ltd's development of the DFS. Dyalog Ltd decided to implement IWA (Integrated Windows Authentication) as part of Conga. Based on their code we implemented the same authentication method both between clients and server as well as between Excel add-ins and the server. The solution makes use of SSPI (Security Support Provider Interface) to establish an authenticated connection between client and server using a challenge-response protocol (NTLM). The big advantage of using IWA is that the users don't have to log on to the specific application. Instead, the user credentials of the current logged-on Windows user is used. Thus, you only have to enter username and password if you need to log on as a different user."

"As for the integration with other systems, all of them were changed to use WebSphere MQ, which is the current standard at Volvo Cars. All of the integrations were prepared and tested in advance in both test and QA environments. In actual production, the migration was scheduled in two steps:

1. Before "go live", all outbound integrations were set up and tested. From the new production environment simple ping tests were done by sending a small test file to each downstream

system, just to verify the connection. Up until this point – with one exception – no changes were required in the receiving system's production environment. During "go live", the old system was shut down and instead configured to automatically forward any incoming file to the new system.

2. After the successful "go live", the inbound integrations were switched over one by one. Over a period of two weeks one system at a time had changes installed to start sending to the new system.

A few other systems were already using FTP, and those we have decided to keep for the moment. Changing them will require no more than a simple change of FTP parameters. The same applied to web-services, all we had to do was change the configuration," Peter concludes.

Results

The entire migration project was conducted over 10 months. The new Hercules system went into operation in October 2012. It is currently estimated that the migration to Dyalog for Windows will save Volvo Cars 3,000,000 SEK (approximately €350,000) annually.