OpenVMS Technical Journal V11

How Flügger Modernized their OpenVMS Applications

Mogens Porsgaard, Modernization Project Leader, Flügger

Roger van Valen, Seagull Software

© Copyright 2008 Hewlett-Packard Development Company, L.P

Overview

Flügger — one of Scandinavia's leading manufacturers, distributors, and dealers of paint products, wallpapers, paint brushes, and accessories — has depended on the reliability of the HP OpenVMS platform for more than 20 years. When Flügger moved to this operating system in 1986, their IT team developed a number of applications that continue to support the business: logistics, customer relationship management, vendor management, financials, and purchase and production applications. Flügger recently undertook a modernization effort for their OpenVMS applications. All applications run on OpenVMS Alpha server systems and are written in COBOL. They have an HP Integrity server in house for development use and expect to migrate to HP Integrity two to three years from now.

Founded in 1890, Flügger is headquartered in Rodovre, Denmark, and employs 1,400 people. Flügger is expanding its business outside Scandinavia into Poland, the Czech Republic, and China. Business management has outlined a number of requirements to support their global operations: business applications need to be internationalized while the host application remains in Danish; applications need to become more intuitive for users; and application screens, especially those used in the retail shops, need to be modernized with an updated look and feel.

Because the reliability of OpenVMS is fundamental to supporting Flügger's business, migrating to a different platform was not an option. But while experimenting with several modernization tools on the market, Flügger's IT team encountered various limitations that prevented them from enhancing their applications. As they were starting to rewrite their applications, they were introduced to Seagull Software through Benny Nielsen, the Danish ambassador in the OpenVMS Ambassadors program.

Going from "Green Screens" to GUI Panels to Leverage and Extend OpenVMS

We installed Seagull Software's LegaSuite GUI on a Windows PC and the client software on our Windows terminal servers. After a five-day training course, we started working on the migration, and the process has run smoothly ever since. Whenever we needed support from Seagull, they were there within a short period of time, and we have been very pleased with their help.

The setup is very simple. No software has to be installed on the OpenVMS system, which means that we don't have to implement any changes to our existing applications at all. This is a great advantage because we can implement the GUI version in a way that is transparent to users. When the IT staff releases an application, users can start it either from the GUI or from the "old- fashioned" version.

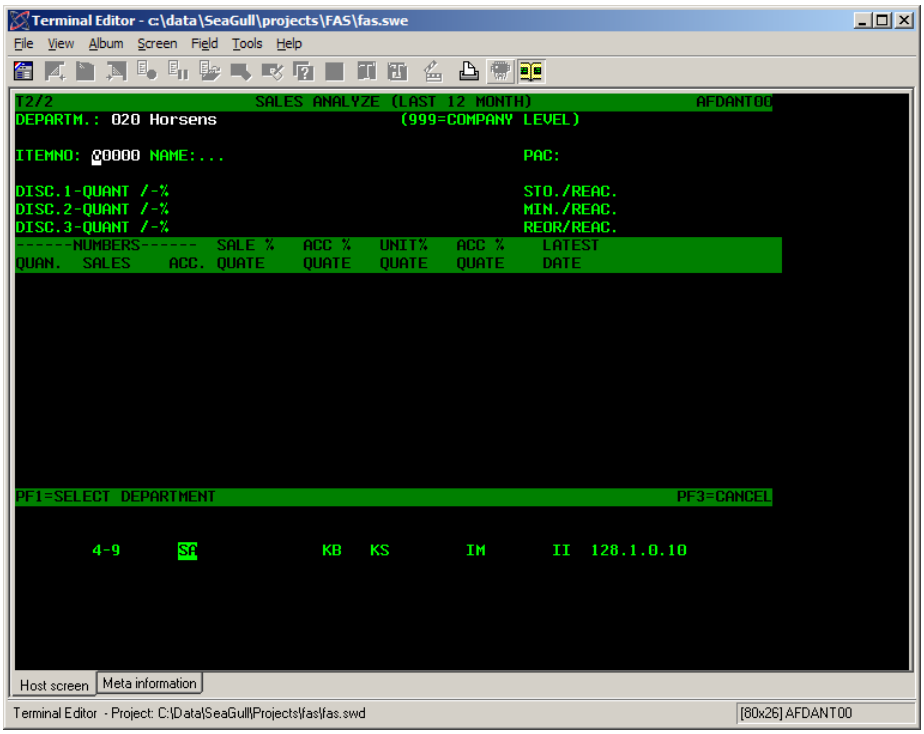
Even though it wasn't necessary, we chose to implement some minor changes in our old applications. The most essential of these changes focused on using a mouse in the GUI panel, so that the OpenVMS application would work essentially as a Windows-based application. Users had to be able to move from one field to another in the OpenVMS application by using the arrow keys. This

challenge was solved by implementing a panel script that sends the correct number of arrow-key sentences when moving from one field to another using the mouse.

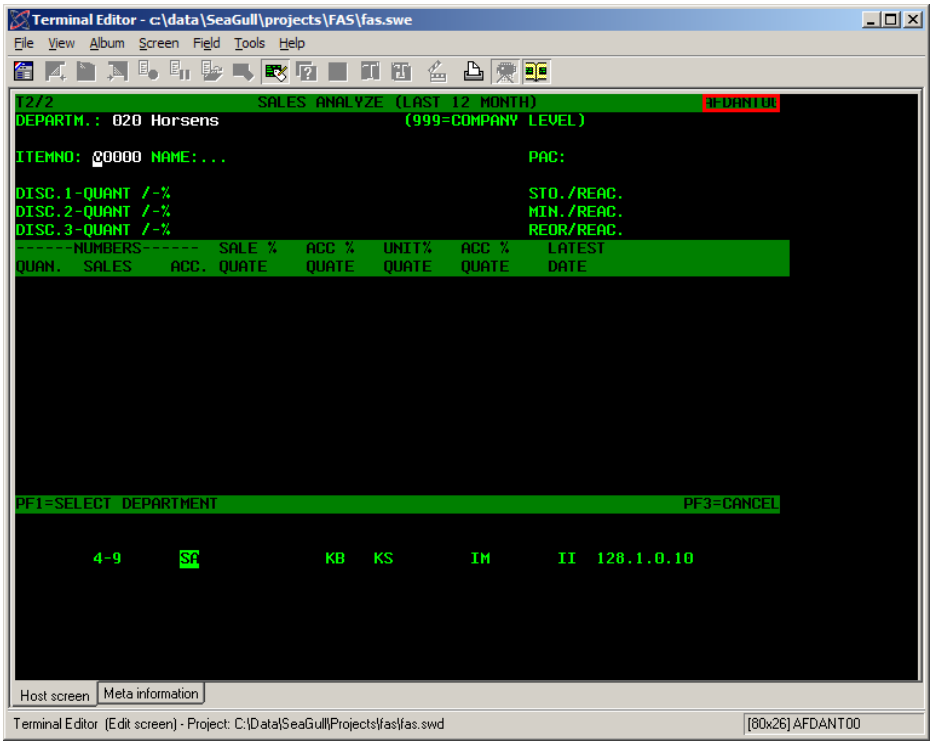
We also took some time to define the look of the graphical panels. We defined different standards for the use of colors, fonts, the look of the buttons, and so on.

The following screen shows a typical OpenVMS application:

© Copyright 2008 Hewlett-Packard Development Company, L.P

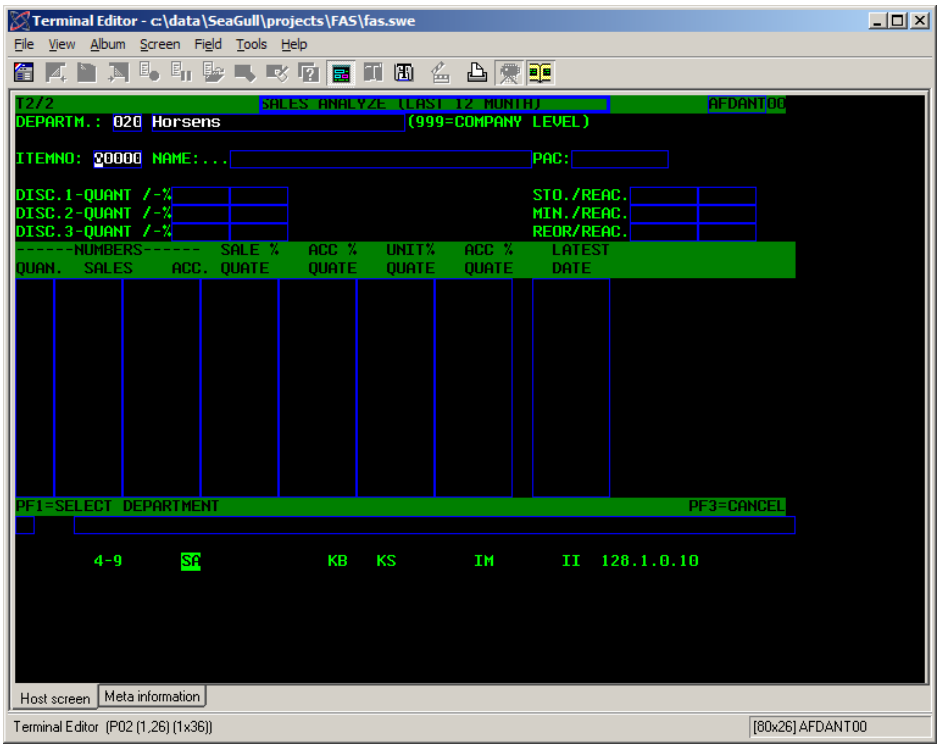


The screen has to be identified to the GUI system. This is done by marking a unique area on the screen, which is shown by the red frame in the upper right area of the following example:



When the application (or screen) is identified, the individual fields on the screen need to be defined, as shown in the blue frames in the following screen:

© Copyright 2008 Hewlett-Packard Development Company, L.P



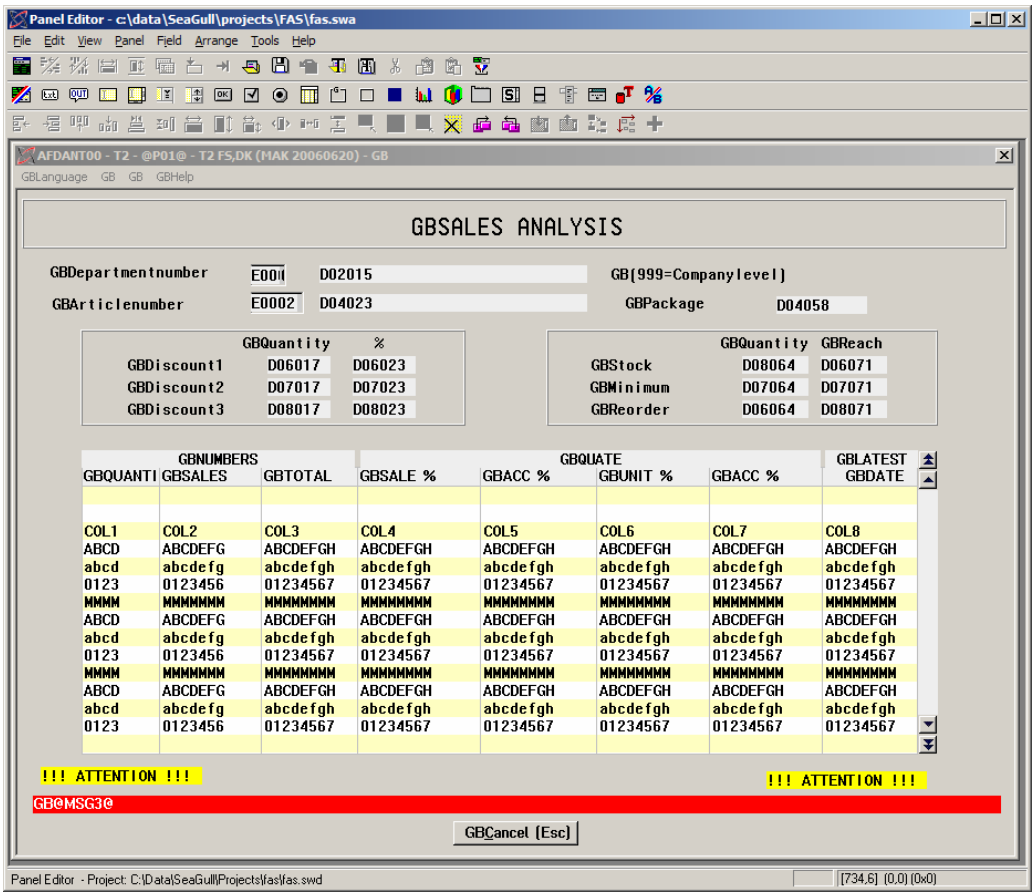
These fields are named uniquely, enabling them to be referenced on the GUI panel.

Then we created a panel in the GUI builder tool of LegaSuite GUI, including the definition of the leading text that explains the fields, as well as the definition of the named fields.

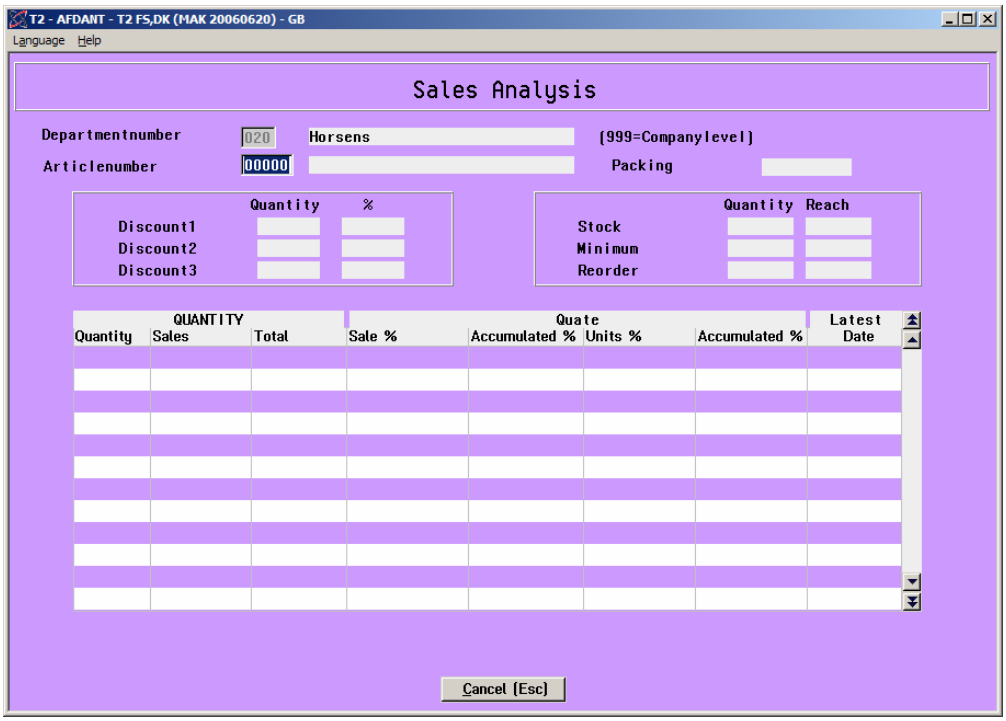
Leading text fields are created with a leading country suffix, enabling the text to be translated to the preferred language during execution of the application.

We had a wide choice of colors, frames, pictures, and other characteristics to include on the GUI panel screens. For example:

© Copyright 2008 Hewlett-Packard Development Company, L.P



When the process was complete, the GUI version of our original OpenVMS application looked like this:



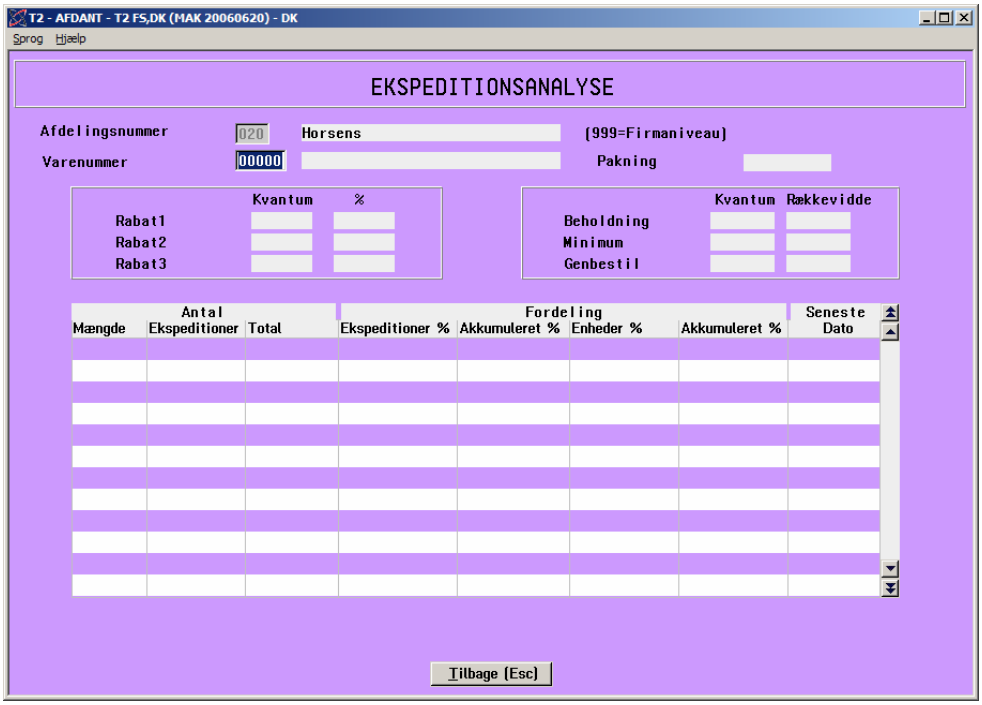
© Copyright 2008 Hewlett-Packard Development Company, L.P

Thanks to the GUI panel menu, we greatly improved the functionality of the application.

We added some features that weren't possible to implement in the original OpenVMS application. For instance, we implemented an HTML document containing a short manual for each application, as well as the option to use different languages.

Each user can change the language as they prefer, any time during the execution. We have already translated, or are about to translate, the system into several different languages, including Danish (of course), Swedish, Norwegian, Polish, Icelandic, and English.

In Danish, the application looks like this:



Technical Implementation

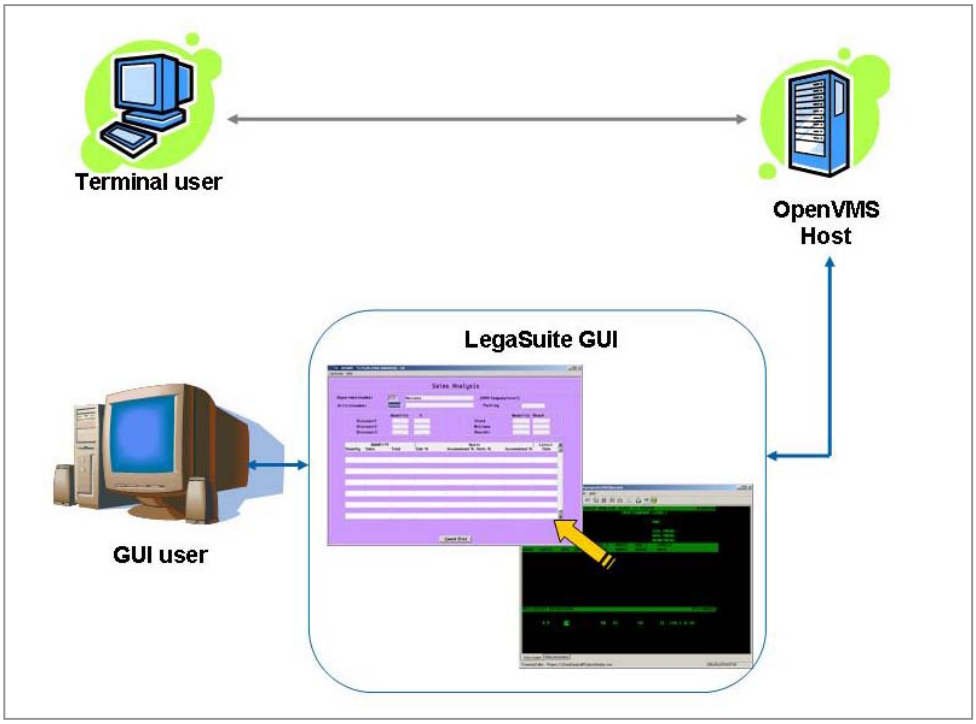
From the OpenVMS host perspective, LegaSuite GUI is a terminal emulator application. LegaSuite GUI connects though Telnet (or Secure Shell SSH) to the OpenVMS host and can work as a VT220, Wyse, or other emulator. Since the LegaSuite GUI application simply drives the host as a regular terminal, the host application does not require changes. However, if the host application is still maintained, changes can be incorporated to make certain functionality easier to implement using LegaSuite GUI.

LegaSuite GUI applies two different concepts to match a GUI layer on top of an OpenVMS application: screens and fields. A "LegaSuite GUI screen" can be defined as a transparent mask on top of a specific emulator view, and a "LegaSuite GUI field" is a container for a specific area on that view. Identification information (screen and field definition) is part of the data stream (for example, a VT220 data stream) received by LegaSuite GUI.

After the screen is uniquely identified, LegaSuite GUI can either show a panel or start some automated steps. A panel is a Windows-style representation to the user. This might resemble the screen (for example, having the same objects as the fields on the OpenVMS application), but it can also be built as, say, one Windows representation (panel) of many screens on the Host application.

The following figure shows how this process might occur:

© Copyright 2008 Hewlett-Packard Development Company, L.P



OpenVMS applications can differ in key handling, but LegaSuite GUI is flexible in adapting its behavior to the host application. Two principal functions can help in this process:

- Metadata: The LegaSuite GUI emulator supports metadata settings, which provide a dynamic way of changing the emulator behavior based on user action (for example, through scripting)
- Scripting: Seagull provides its own script language that has syntax similar to Visual Basic.

A big difference in application handling between a Windows and an OpenVMS application is the level of control. In an OpenVMS application, the host is typically in control. Any key pressed by the user is handled by the host, and the host responses by text echoing, cursor movements, screen updates, and so on. When the user wants to supply text in a certain field, they typically have to use keys to move to that field and then type the contents.

In a Windows application, the user is typically in control. When the user clicks a certain control (such as a box in which to enter text), the control is given focus and the user can type the text. LegaSuite GUI supports a wide range of features such as events, emulator properties, and configuration settings and script functions to move easily through the host application and type text in fields.

Although LegaSuite GUI offers numerous features that help you to overlay the host application with GUI controls, building a GUI application is not only a matter of adding a visual interface on top of a "green screen" application. With LegaSuite GUI, you can redesign the application workflow, consolidate data from different applications, use other (desktop) applications (such as Word, Excel, and so on). Toward these ends, LegaSuite GUI scripting is an important feature.

A typical GUI project follows these steps:

1. Host application investigation. During this phase, answer questions such as the following: How can I tune the host application? Can I standardize specific behavior such as key handing, cursor positioning?
2. Screen and field design. At this point, you must build the "identification" for both LegaSuite GUI screens and LegaSuite GUI fields.
3. GUI building: You now build and customize your graphical panels and their related controls, and then map these controls to the fields designed previously.

© Copyright 2008 Hewlett-Packard Development Company, L.P

4. Scripting: Scripting lets you control the behavior of your GUI application mapped against the OpenVMS application. Scripting also lets you add functionality not available in the host application, such as workflow improvements or integration with desktop applications like email, Word, and Excel.
5. Testing: Testing is an essential step of the development process and is performed in the LegaSuite GUI developer before the application is deployed. The panels are tested to verify the look and feel of the application as well as the data exchange process between the panel and host. The scripts supporting the functionality and navigation of the application are tested using the step-by-step execution capabilities that are built into the LegaSuite GUI developer.
6. Deploying: When deploying the GUI, you have a choice of thin-clients or rich-clients. For example, a Windows® client software or Windows browser plugin achieves the performance characteristics of traditional emulators with the benefits of Win32 code executing natively on the Windows platform.

For more information

To contact the authors, please send email to Mogens Porsgaard, mopo@flugger.com, or Roger van Valen, rvanvalen@seagullsoftware.com.

For more information about Seagull Software's OpenVMS solutions, please see the Seagull website at www.seagullsoftware.com, or send email to info@seagullsoftware.com.

© Copyright 2008 Hewlett-Packard Development Company, L.P