CERN-IT-2001-002

Monitoring of Engineering Applications

Authors

S. Ahmad, C. Eck, M. Ferran, B. Flockhart, M. Guijarro, A.M. Osborne[1], P. Saiz, V. Puzynin

Abstract

This paper presents a license monitoring system implemented at CERN[2] in the Computing for Engineering (CE) group[3] of the Information Technology (IT) division. The system gathers statistics on the use of the different software products supported by the group on both Windows and Unix platforms. The web is used for all aspects of the project; data presentation and associated interfaces. The system is completely automated. Manual intervention is required only when there are new products to be monitored. Certain technical details of this work have been reported elsewhere[4].

---

[1] Contact person
[2] http://cern.web.cern.ch/CERN/
[3] http://wwwinfo.cern.ch/ce/
[4] Wrapping procedure for NICE environment. JINR Preprint E11-2000-286, Dubna 2000.

Table of Contents

Introduction

The Computing for Engineering group (CE) supports the use of many (~70) different software packages by the widespread engineering community at CERN. Such packages include computer aided design tools for mechanical and electronic engineering, field calculations, mathematical and structural analysis. These tools are employed to design the CERN accelerators and their associated infrastructure as well as the experimental detectors, which currently consist of literally tens of millions of components. Most, but not all, of these packages are commercial applications, usually having an associated native licensing system. CE group already had significant experience with diverse licensing schemes and had previously tried to solve individual situations by writing application-specific monitoring schemes which were not extendable and were difficult to maintain.

Terminology

First we define some terms commonly used throughout this paper:

A *product* is a software package having a specific version. Products are composed of a set of independent sub-parts or *features* each of which may, or may not, correspond to a different executable file.

A *license* is a file, provided by a software vendor, to control the use of its products and associate features. Licenses are installed on the relevant license server. Then, whenever a user launches the product, a connection is made to the license server to find the appropriate license, and if successful, continues with the execution. Some licenses, known as floating licenses, specify the maximum number of simultaneous or concurrent users of each feature. The latter is also known as the number of *tokens* for that license. Some licenses are site licenses where the number of concurrent users is unlimited. Finally there are node-locked licenses that allow only a set of specific user machines to access the product. For obvious economic reasons, CERN tries to avoid node-locked licenses where possible. The monitoring scheme described in this paper does not consider products with node-locked license, although could, but rather concentrates on products shared by more than one user. Unless the product employs Flexlm licensing technology already, it must be ‘wrapped‘ (see below) before installation on network application servers, on a distributed file system or on the users’ local PC or workstation.

A *license server* contains the license files and is where the vendor license system process (a daemon on Unix) executes. The daemon receives requests to access the products, checks whether there are tokens free for that feature, and finally either grants or denies the user request. The license server also normally holds the *license log-files* of product and feature use.

Requirements

The first step was to make a concise list of the requirements for the system, largely based on the group’s prior experience and also from discussing with other product

managers at CERN.

- With a finite budget for software acquisition, CERN needs to have sufficient, but not a large excess of, commercial licenses for any particular software product.
- Such a scheme should be able to handle and present results in a consistent framework for all products and licenses, regardless of their native licensing system.
- It should work in a consistent way across all flavours of Unix, and also all flavours of Windows supported at CERN.
- The scheme should also be able to monitor CERN 'homegrown' software packages with no licensing scheme.
- In order to be able to perform product version control, the system had to be able to track individual users of different products and the versions employed. This implied a close integration with the Computer Centre Database and CERN's Human Resource database (HR). E-mail addresses for users of different products should be available in such a form as to permit easy use of CERN's e-mail list system.
- The system should automatically inform the managers responsible for product acquisition when licenses or contract agreements were due to expire.
- The system should be based largely on a commercial solution for reasons of long-term maintenance. Nevertheless, some minimal tailoring and interface software would likely have to be written to adapt the chosen solution to the group's needs.
- It had to be as automated as possible. Day to day operations should be managed via pre-scheduled 'cron' jobs that are scheduled by the underlying Unix operating system.
- For reasons of commercial confidentiality, all information generated by the system had to be hidden from the general user community while at the same time being easily accessible to the product administrators. In practical terms this meant that all information, and where possible all interfaces, had to be available on the web via password protected html pages.
- Where possible, information should be available in graphical form in order to show long term trends.
- To permit fast on-line data processing for immediate analysis, the log-files should be limited to contain data for only the previous few weeks (short-term data). All long-term data should be adequately summarized on a weekly basis. The system should be open to allow for easy implementation of other management tools and functions, such as product restriction lists and license 'hogging' management as described below.

The License Database

As noted above, an infrastructure element that was required before the license monitoring scheme was implemented, was a database to store the details of the products and features. The database technology chosen was Oracle – the CERN standard relational database. The basic information consists of the useful addresses of the software provider, of the CE group product management, and the number and expiry dates of the individual licenses for the specific features. In addition URLs for product information and related CERN-specific information were also stored thus enabling the automatic generation of user help web pages.

For all products whose native license scheme is Flexlm, the information in the database is directly and regularly compared with the Flexlm license files as a consistency check. Other license systems do not generally permit easy program access to the license files.

The database interface technology used is SQL*Forms from Oracle. The content of the database is available in read-only mode on the web in the form of dynamic HTML pages (password protected) generated by Oracle ORA*WEB technology.

Managers of individual products in the group are automatically reminded when licenses are about to expire.

The product index web page is shown below in Figure 1.

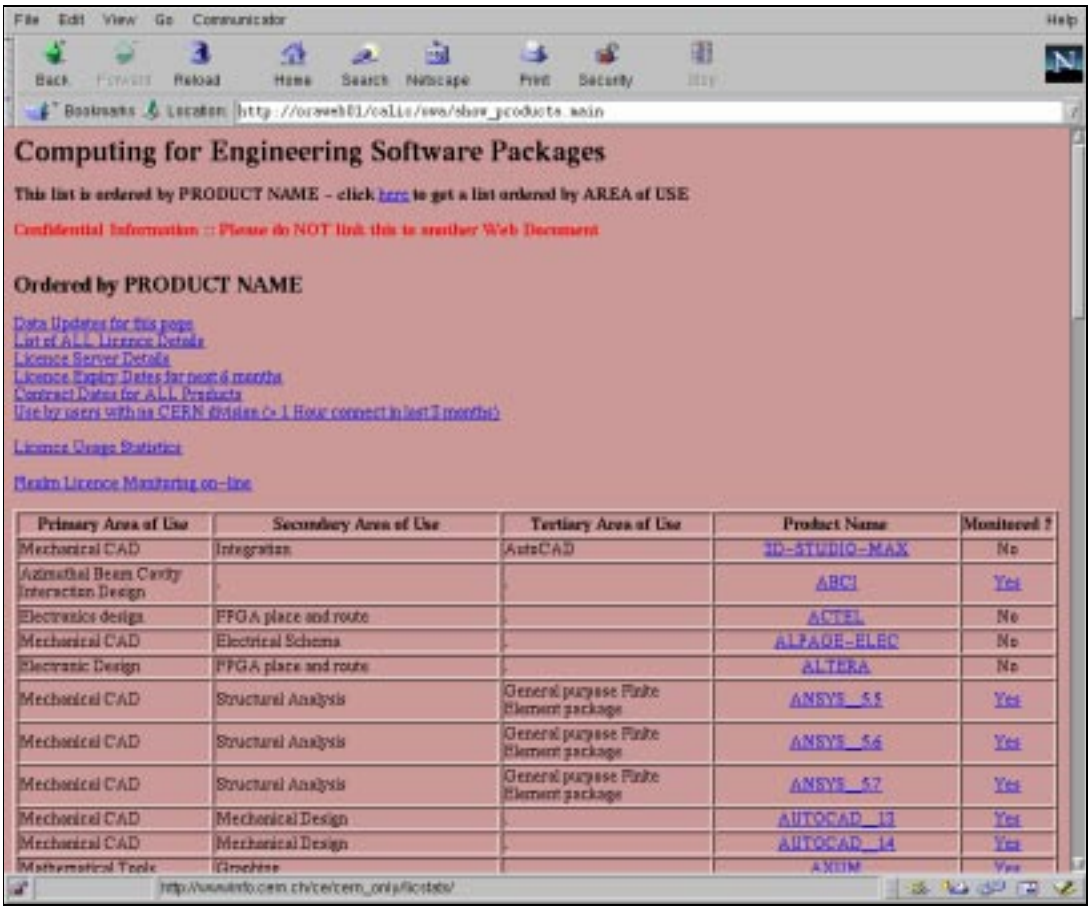

Figure 1 Product Index web page from License Database

Products are categorised by area of use. Links are provided to show individual product details, license server details and products whose licenses will soon expire. Specific product details are given as indicated in Figure 2 below.

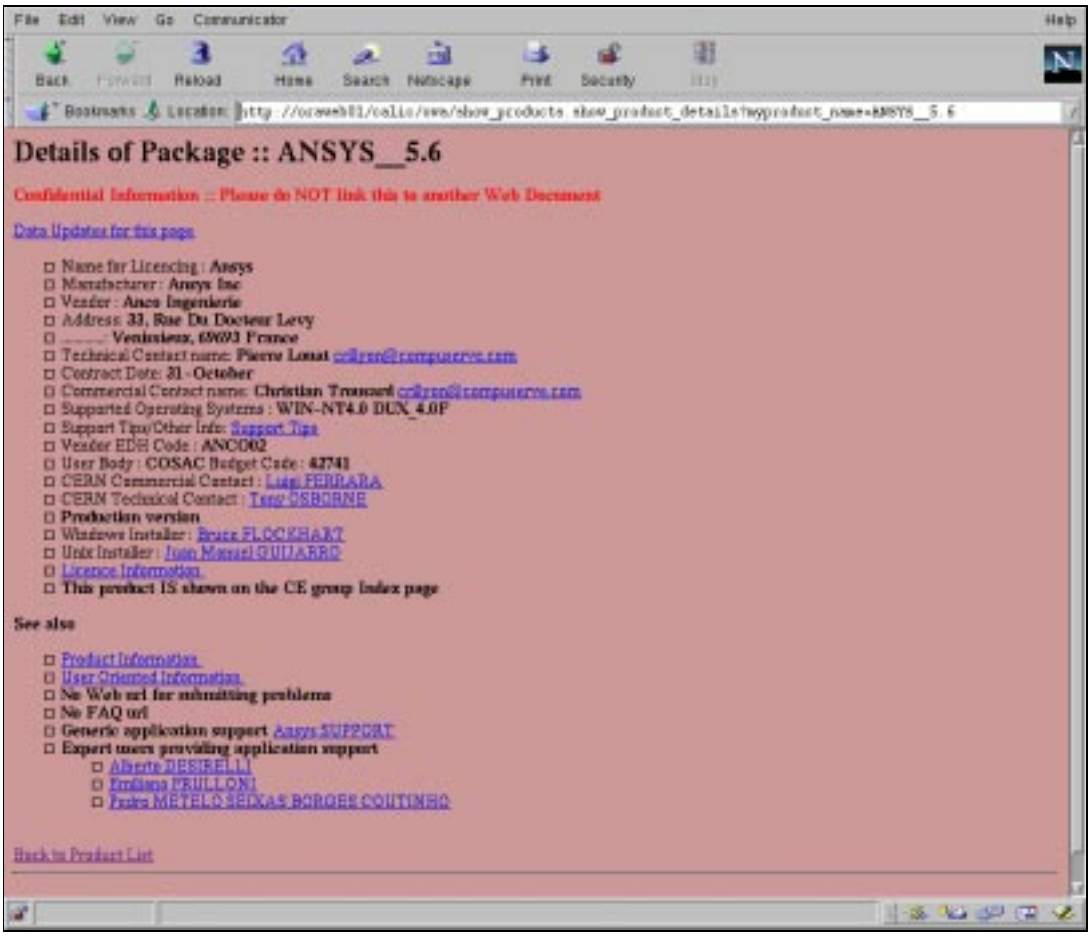

Figure 2 Product details including useful URLs and contact people

Links are provided for contacting both the software supplier and the CE group people responsible for licenses and installation. In addition URLs are provided for installation hints, product and CERN user information.

Typical license details are shown in Figure 3 below

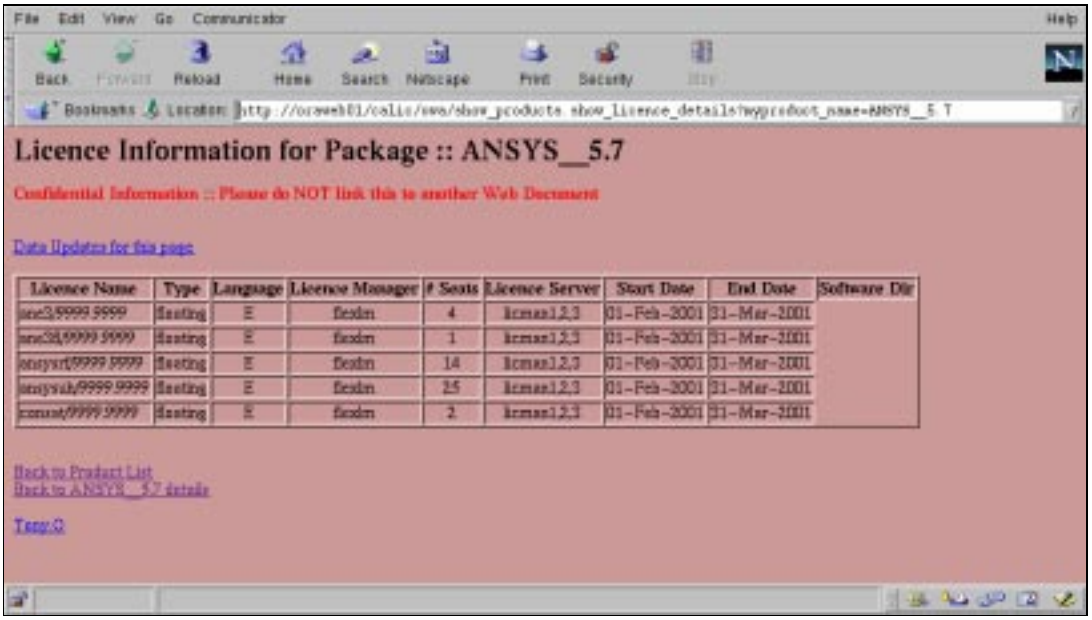

Figure 3 Detailed License information

Details are shown for license feature versions, number of licenses (seats), native license manager, license server and license expiry dates.

License Servers

IT division uses a redundant configuration of three dedicated Solaris based license servers for Flexlm based products. License serving continues if at least two of the three machines are up and running. There are of course other license servers dedicated to other native licensing schemes.

These license servers, licman1, licman2 and licman3, store the Flexlm license files and the Flexlm and Flexwrap log-files. Analysis of the recent product and feature log-files also takes place on these machines. By keeping the size of these log-files to reasonable limits, such analyses do not affect the speedy response to a license request.

The Choice of the Monitoring Software

Based almost entirely on the constraint that the system had to work in a consistent way on both Windows and Unix, a survey of available products rapidly concluded that Samsuite/Flexwrap[5] was the only practical choice. Since Flexlm was already CERN's preferred license system, it was also a natural choice. Furthermore, products already using Flexlm as a native license system were easy to integrate into the system

[5] http://www.globetrotter.com/products.html

and required no further treatment at all. Alternately, applying Samsuite/Flexwrap to products not already using Flexlm as a native license system, produces log-files and statistics identical to the native Flexlm products, thus permitting an overall consistent treatment of the data.

Flexlm, SAMSuite and SAMRreport

The technology used for the license server and associated license monitoring was acquired from Globetrotter. Its license management system, Flexlm, is becoming the de facto standard in the software industry for the type of products that CERN's engineering community requires. The main product was called Flexlm, now known as SAMSuite**.** The same company also offers SAMReport, a tool to analyse and provide textual and graphical reports from the log-files generated by the different Flexlm daemons.

Some of the packages that the group supports already use Flexlm technology as their native license manager. Indeed CE group has long had a policy of trying to persuade software providers to move to Flexlm. For products not using Flexlm, Globetrotter provides another tool, called SAMWrap, which allows the monitoring of products that do not use Flexlm or indeed any license manager. This process is called 'wrapping' and is explained below.

The terms Flexlm and SAMSuite are used interchangeably in this paper, as are the terms Flexwrap and SAMwrap.

Wrapping a Product

Wrapping is the process of enabling a product executable to communicate with the license servers used for monitoring purposes. After wrapping a product behaves in the same way as a native Flexlm product. Thus the log-files are updated for each new request for a product or feature. Analysis of these log-files provides statistical reports for license monitoring. The product end user is unaware of the fact that the product has been wrapped unless two of the three redundant license servers fail. He or she will continue to receive messages directly from the native license manager – notably in the case of a denial or refusal of a request. Clearly, in order to wrap a product, access is required to the product binaries. Thus the standard configuration must first be customised, by wrapping the executable, before making it available to the end users.

It is possible to wrap not only products that employ a floating-license system different from Flexlm, but also those which have no integrated license system. In the first case, Figure 4 shows the situation before and after wrapping a product. Before wrapping, the application contacts only the native license server to obtain a token. After wrapping, the application first requests a Flexwrap (monitoring) token from the Flexwrap license server (which will update the relevant log-file) and subsequently requests a second (product) token from the native license server in order to run the requested application. For the case of products having no native license manager, the process is simplified - only the Flexwrap token is requested.

We have chosen to always Flexwrap our products and features in a *non-blocking mode*. This means that, as long as the Flexwrap license servers are up, failure to

obtain a Flexwrap token will neither affect the granting of the native license token nor the normal execution of the product binary.

Such a failure would occur if one had assigned less Flexwrap tokens to a feature than there were native tokens for that feature-or the licman trio were down. Thus to correctly monitor a feature of a product one has to assign *exactly* the same number of Flexwrap tokens as there are native license tokens. Assigning too few Flexwrap tokens means one record false denials or refusals. Assigning too many Flexwrap tokens means one cannot record a product token being denied or refused. The essential point is that the wrapping process allows the monitoring of the use of the Flexwrap license tokens assigned in the wrapping procedure to each product or feature.

For the case of a product with site license (where the number of concurrent users is unlimited), or for an unlicensed product, the correct number of Flexwrap tokens must be at least as many as the observed maximum number of concurrent users.

To help discern problems of mismatch, the long term statistics plots always display *both* the number of native tokens as well as the number of Flexwrap tokens assigned for any given feature/product. This data comes from both the license database and from the current Flexwrap license file which is read on the fly at the time of the report generation.

Features or products that are wrapped are assigned names which may be chosen freely, although normally one would want to have meaningful and unique names. This is in contrast to products which have Flexlm as their native licensing system. In this case the names are provided by the manufacturer and cannot be modified. In some cases a license file that serves multiple versions of the same product also specifies different feature names for each version. If this is so then one may easily monitor the use of different versions of that product, a possibility that is always available for a product which is Flexwrapped.

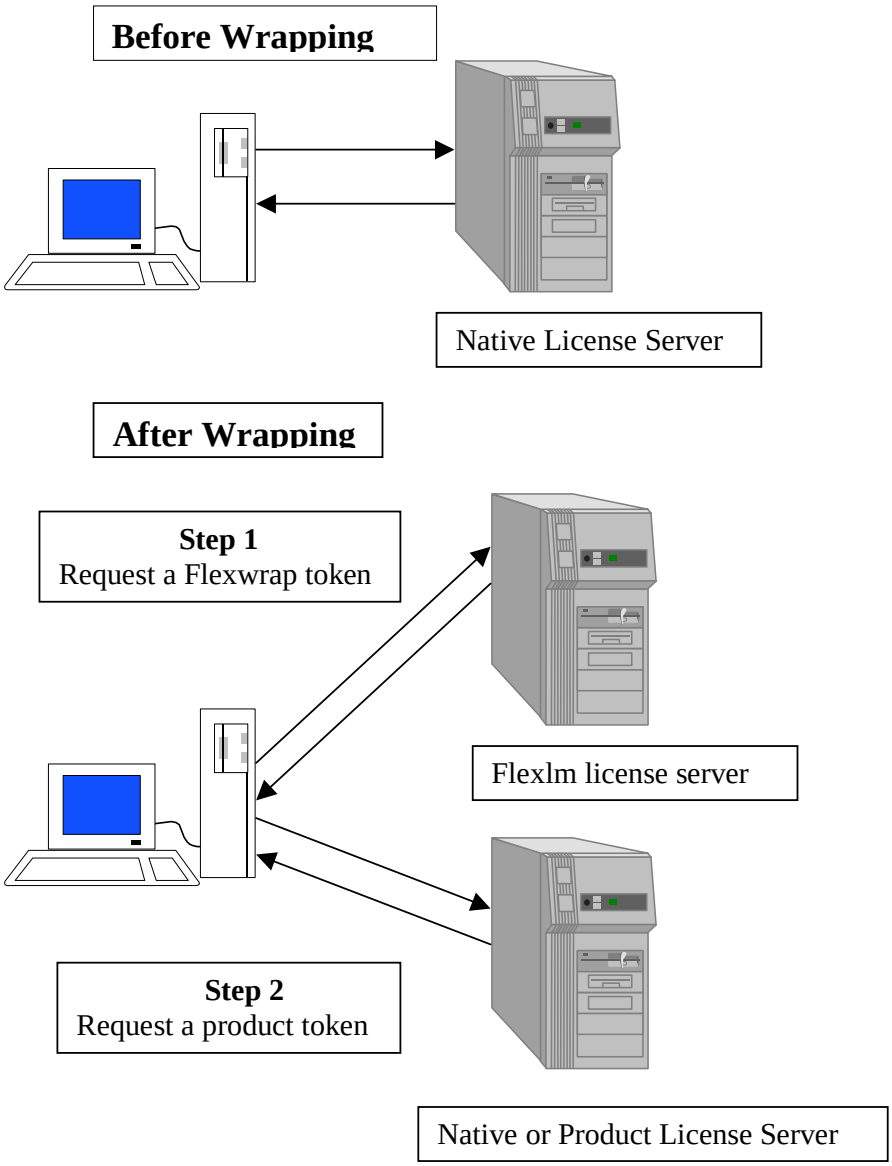

Figure 4 Wrapping Scenario

The complexity of the wrapping process varies significantly from one application to another. Note that it is only possible to wrap executable files or shell scripts and not libraries.

The simplest case corresponds to the situation where to use a product the user has to invoke specifically named binaries or executables. Here one simply replaces the original binaries with wrapped binaries keeping the same name. Alternatively one can rename the original binaries, install small wrapper scripts with the same name as the original binaries, and have these scripts invoke the original (now renamed) binaries. In any event the user is unaware of the wrapping process.

A more complicated situation arises when the application works by selecting a module from within a monolithic single binary according to an environment or an input parameter. Naturally, in general, one would like to monitor the use of all the individual modules. The motivation for this is that for some products specific modules may be very expensive and one would like to know if indeed they are used or not. In this case one has to provide a script or wrapper (which is given the same name as the original monolithic binary) that itself requests a different Flexwrap token for each particular feature (or module) selected according to the input parameter or environment variable value.

In general successful wrapping of an application requires an understanding of which modules within an application *can* be monitored, which of these one *wants* to monitor, and how the selection of a particular module is made by the specific application. Thus some understanding of the internals of the application is unavoidable.

Nested or Double Wrapping

It may happen that one uses different versions of a product, or the same version on different operating systems and the requirement is to monitor the individual use of each version as well as the overall use of the product. Such a case happens when the different versions are served by the same license file. If one only wraps the individual versions one cannot obtain correct statistics for the overall use of the product. More specifically, statistics generated this way for concurrent use and denials will not be correct.

The solution to this problem is to wrap each product twice. In the first wrapping one chooses feature names that are different for each version. This permits the monitoring of the individual versions. Then the products must be wrapped a second time, but now features common to both versions must be given the same feature name. Thus the same Flexwrap token name or type is requested by all versions of each feature. Note that in this case the statistics of concurrent use and denials will be misleading for the individual versions. Denial and concurrent use statistics arising from different product versions can only be combined in real-time and not later, since that would require many time snapshots to be recorded. Nevertheless, statistics gathered for the different product versions are valuable since they correctly indicate their relative use. By using the double wrapping technique the overall product statistics are correct.

Wrapping : Unix Specifics

CERN supports multiple Unix operating systems. The actual distribution of binaries for different Unix platforms is handled by using the CERN adopted standard distributed file system AFS (Andrew File System), although they also could be distributed by NFS or other file systems. Executables are invoked from the different platforms by calling the same named executable with an environment variable that specifies the operating system type.

In some cases we wish to understand the use of applications on the different Unix platforms, and in others we simply require statistics on the overall use on all platforms. This may be solved by either performing single or double wrapping as

necessary. To make wrapping easier for Unix based applications, a script was written to automate the wrapping process. This script requires the name of the (executable) file, the name of the feature and the platforms on which monitoring should be done. Then, the script will copy the original file to another directory, and replace it by a script that would run the executable on the appropriate platform. To reverse the process, another script to unwrap executable files was also written.

Wrapping : Windows Specifics

At CERN there is an overall integrated network environment called NICE[6] for the thousands of networked PC's on the site. Among other facilities NICE provides a set of replicated application servers for the distribution and use of software packages (similar to AFS for Unix users).

The SAMwrap license server is installed on the *sam-srv1* PC operating under Windows NT Workstation. This permits the wrapping of applications and the generation of the appropriate license files.

The user interface for the wrapping procedure is relatively simple and is shown in Figure 5. The interface allows the license administrator to create a newly wrapped application (see Figure 6), to modify the current license information for a wrapped application, or to remove an application from the Flexwrap license file.

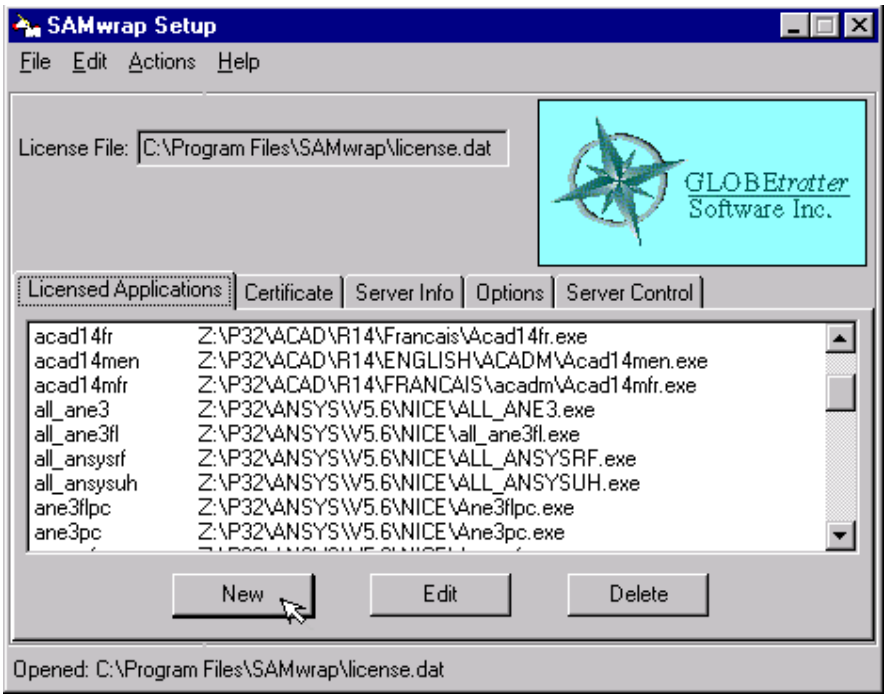

Figure 5 Windows wrapping User Interface

[6] http://nicewww.cern.ch/homepage/home.htm

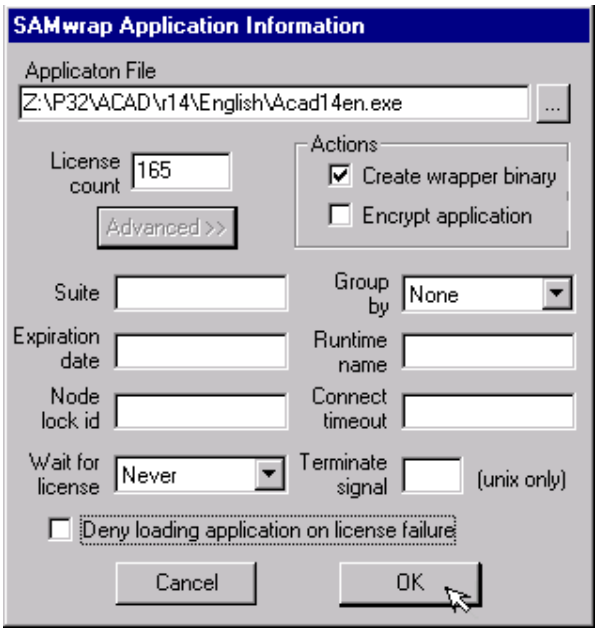

Figure 6 Windows wrapping of a new application

The results of the wrapping procedure are:

- A move of the original executable file *<binary>.exe* to *<binary>.fwr*.
- Generation of the wrapper binary file with the name of the original executable file *<binary>.exe* and the creation of a symbolic link from *<binary>.exe* to original *<binary>.fwr*.
- Creation of a *Swrapper.lf* file in the same directory as *<binary>.exe* file.
- Generation of a FEATURE line for the wrapping application, and adding this line to the SAMwrap license file *license.dat*.

The *Swrapper.lf* file is a plain text file containing the path of the Flexwrap license server. This path points to all three license servers licman1,2 and 3 so that a Flexwrap token may be obtained from any of them in case any one machine is down.

In order to avoid problems should all three machines be unreachable due to network problems, after a given timeout period, the wrapper will read the required license information from a local copy of the license file residing in the same directory as the wrapped binary.

Experience has shown that wrapping is not a trivial procedure. For some applications it turned out that a complex and individual approach was required that was not visible from the outset.

One of the main difficulties encountered was how to handle different product versions

when each of the multiple executables had the same name. As an example, AutoCAD[7] is currently installed on the NICE servers as 8 independent executables – including two language versions for multiple AutoCAD modules – all with the same executable name. A simple renaming of the wrapped binaries will not work on NICE, since there would then be a conflict with the registry information which was updated when AutoCAD was originally installed. One could of course think to modify the registry but experience has shown that such modifications, for a complicated installation such as AutoCAD, are very delicate in a large network environment such as NICE, and should be avoided. Otherwise said, the original AutoCAD executables must not be either changed or moved from their original directory locations.

The solution to this problem was to create 8 different wrapped executables, each of which first contacts licman1,2,3 to get it own uniquely named feature token for Flexwrap, and then invokes the correct AutoCAD.exe file located in the correct file path.

A second example of a specific approach is that of the wrapping for ANSYS[8] on Windows. As on Unix, the application has a single executable file from which a choice must be made to select a particular sub-module corresponding to a specific ANSYS license with its own number of tokens. Since we wished to monitor each individual license, after wrapping the Flexwrap license file has to contain 4 unique feature names. To solve this problem, 4 identical C programs were written and these modified the corresponding *"ansys56_product"* value in the Windows registry. Then the programs invoke the ANSYS executable file as an external process. The standard wrapping procedure was then carried out on these programs, each of which has a unique name.

On-line Displays of Short Term Data

The system generates data in significant detail for the last seven days, with graphs and tables. It makes this data available via web pages which are only visible within the CERN environment and are also password protected. In addition they permit the request for more specific reports from the log-files of the last four weeks.

To present this data, several DHTML pages were written. The first is shown in Figure 7. This page presents all products under control of the monitoring system. Native Flexlm products are indicated in lower case and the wrapped products in upper case. When the user selects a product, a page similar to Figure 8 will appear.

This page presents a summary of the last seven days use of that product. In the upper part of the screen, there is a table with all the features of that product. Then, for each feature, the table shows the number of licenses, number of requests, number of requests issued and denied, percentage of denials, maximum simultaneous use, and the number of hours that the feature was used. In the lower part of the screen, there are three hyper-links. The first displays a graph of the use of the features over the last week (see Figures 9 and 10 below). In these graphs, the red line shows the number of tokens available, and the blue one the number of tokens used. The second link shows a similar graph of the use of the last day. Finally, the third link shows the use of the

[7] http://www.autodesk.com
[8] http://www.ansys.com

product at that particular moment.

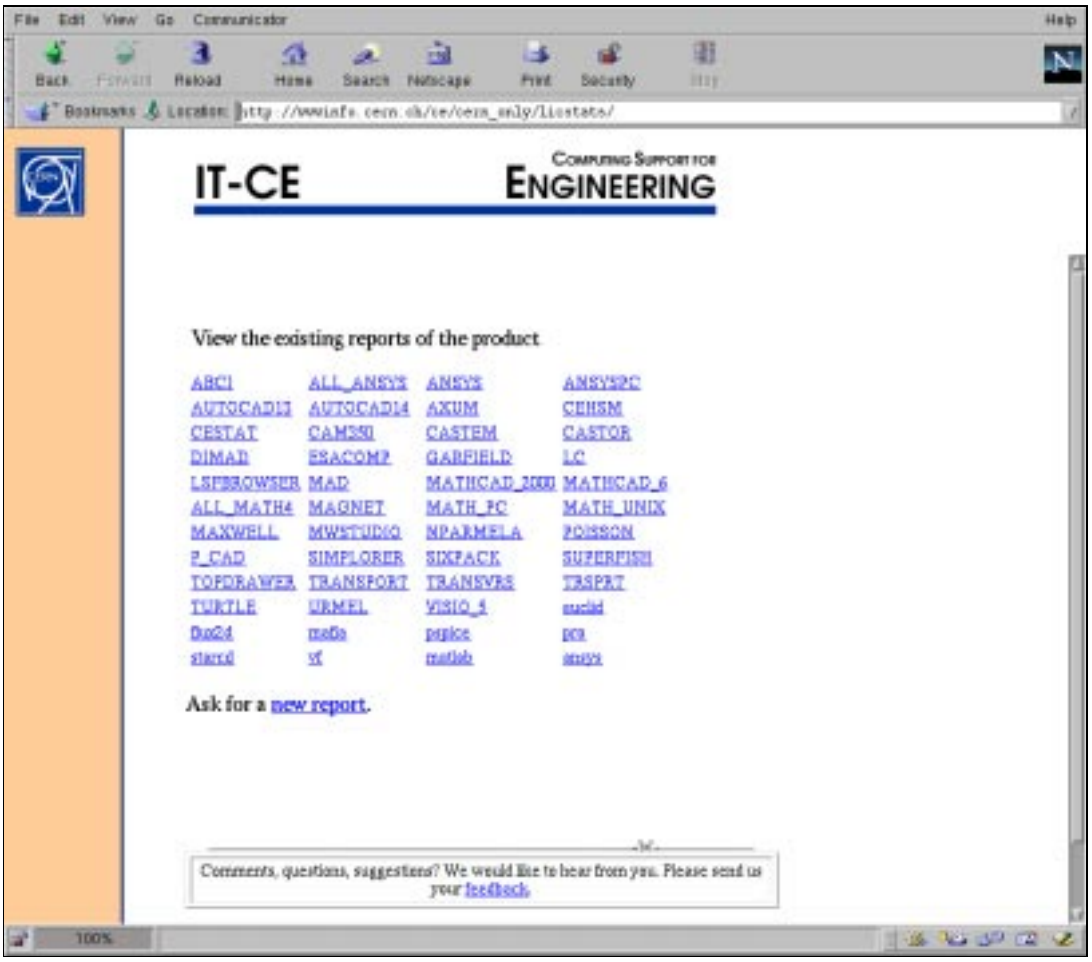

Figure 7 On-line Monitoring index page

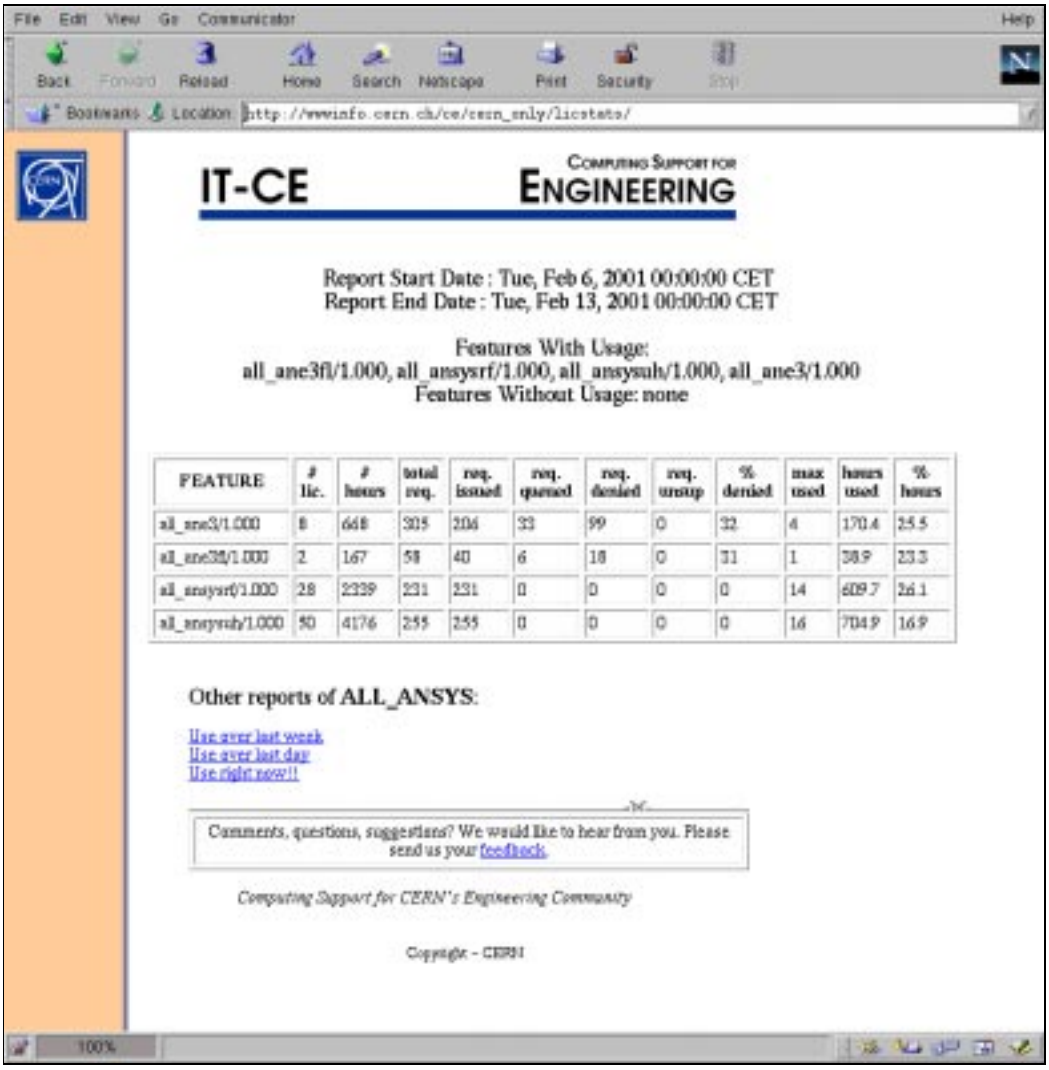

Figure 8 On-line Monitoring License Details

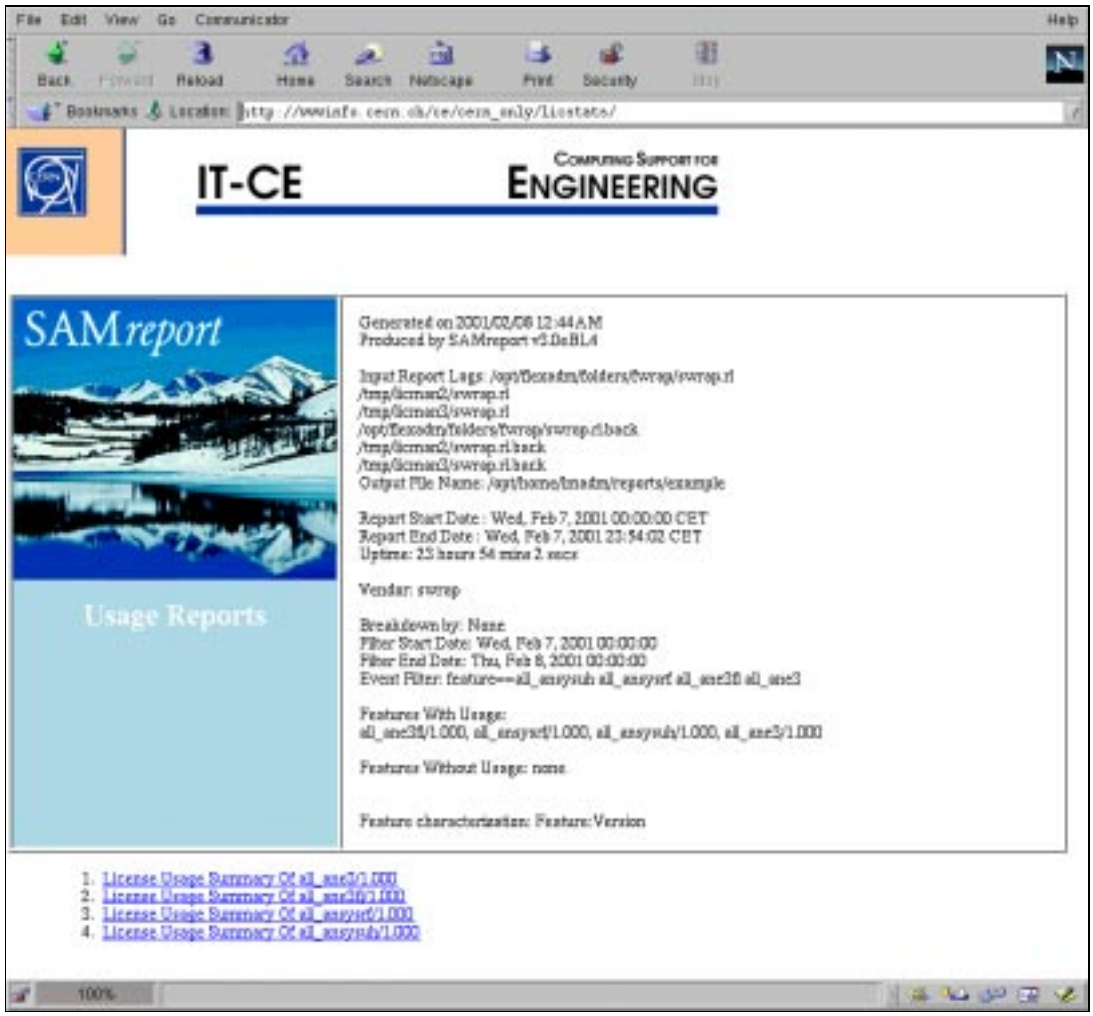

Figure 9 On-line monitoring product front page

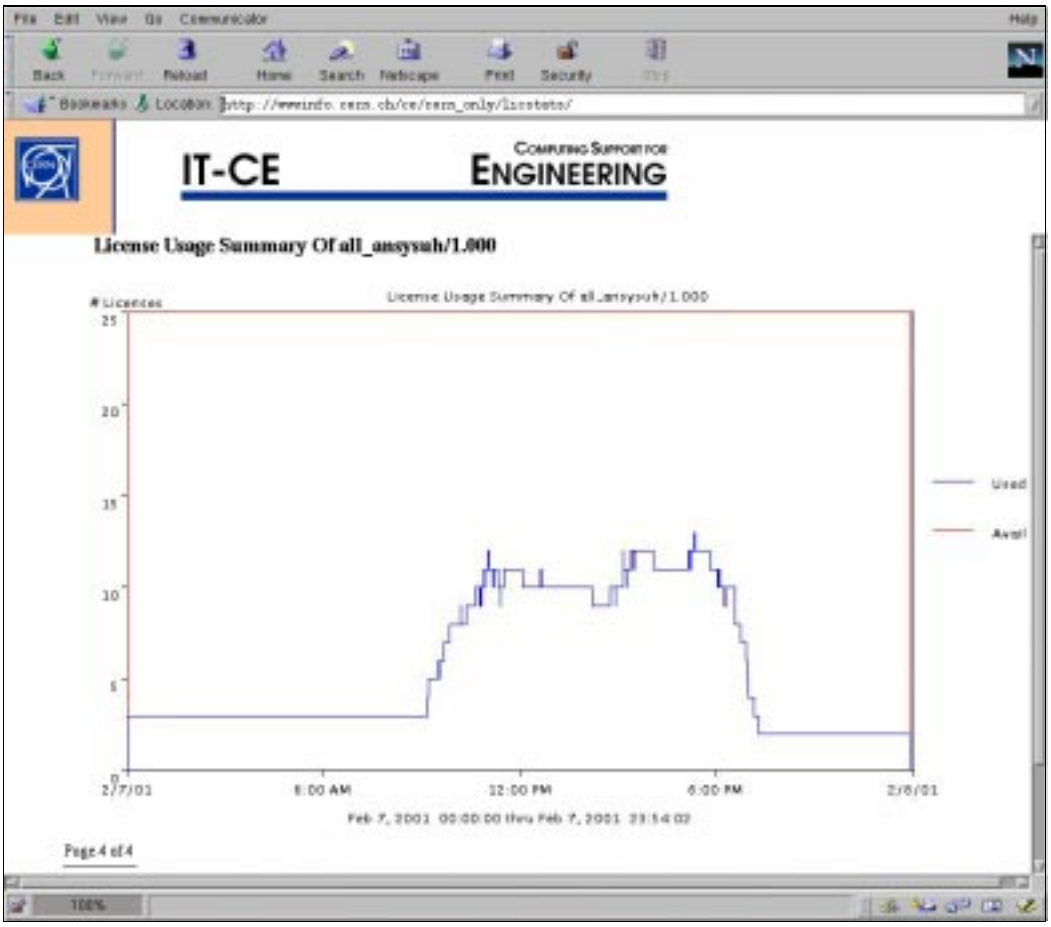

Figure 10 On-line monitoring license use over time

All the above pages are automatically generated every night via cron scripts.

Instantaneous Snapshots of Product Use

The system also allows the querying of who is using a product at a given point in time. This is useful in the case when some immediate urgent action has to be taken and the users should be informed. A typical use of this facility is when a license server must be restarted or rebooted.

Again, the basic information is obtained via the native Flexlm tools, with some post-processing to re-format it. This post-processing was done with a PERL script that generates DHTML. Two more PERL scripts were implemented, the first one to get user phone number and address, and the second one to send all users of a product an instant message via the Zephyr messaging facility that notifies all instances of a userid connected to the CERN network. The type of page available to the user is displayed in Figure 11 below. By clicking on the 'Get userid' button one may see the list of userids currently using a given product. A further option shows the HR information about such users.

In order to send a Zephyr message, authentication via the sender's AFS userid and

password is required.

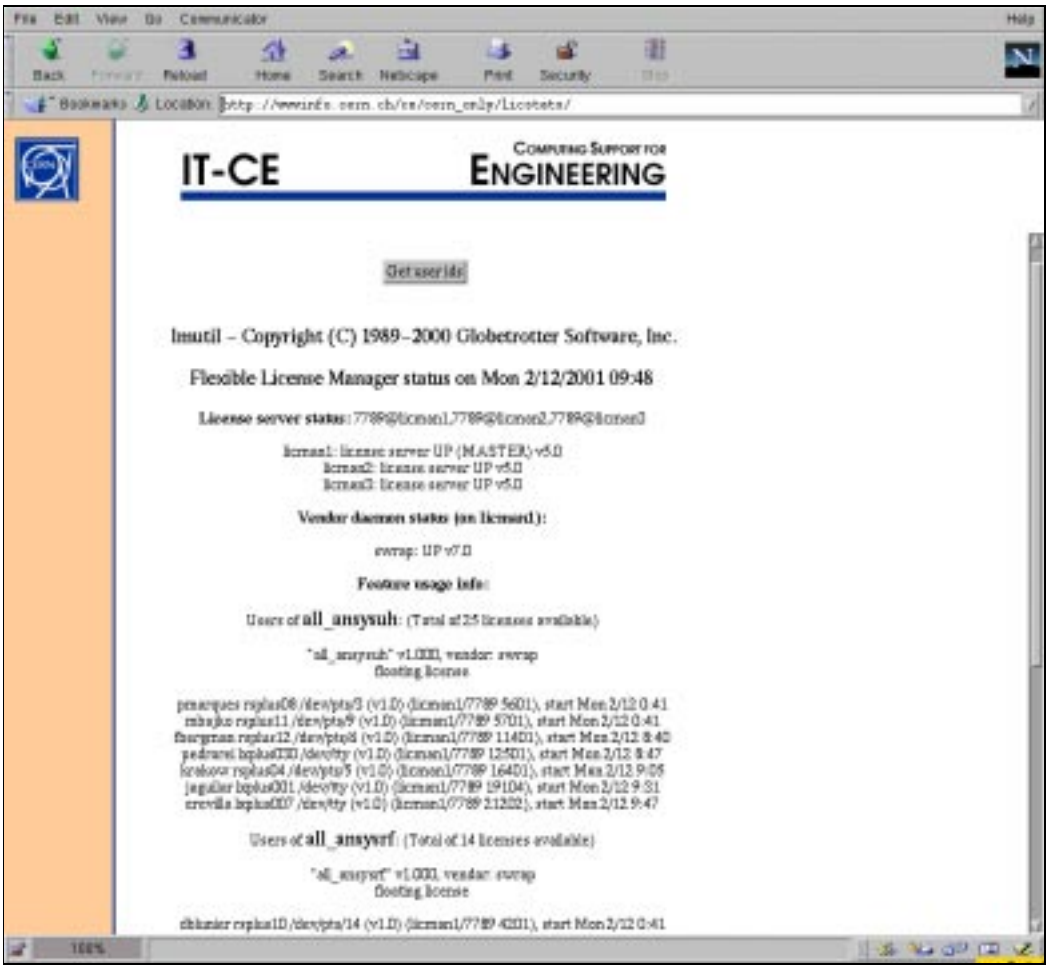

Figure 11 On-line monitoring use of a product now

An End-user Tool for License Management

In order to enable end users to better share a limited number of software licenses, a graphical user interface, based on tk/tcl and the Flexlm tools, was written. A typical screen shot of this tool (**cestat**), available both on Unix and Windows, is shown below in Figure 12.

A list of products is displayed; selecting one displays the features of that product. Selecting a feature then displays the current list of userids that currently have a corresponding license. Finally clicking on a userid one obtains the location and phone number of the corresponding user, thus allowing users to contact each if the need arises. **cestat** works for either native Flexlm or wrapped products. On Unix, **cestat** queries the license servers (licman,1,2,3) via the Flexlm toolkit and displays the result together with data from the HR database. On Windows, a CGI script running on a

web server queries the results on Unix and sends them back to the Windows interface.

cestat is the only tool discussed in this paper that is available to the user community.

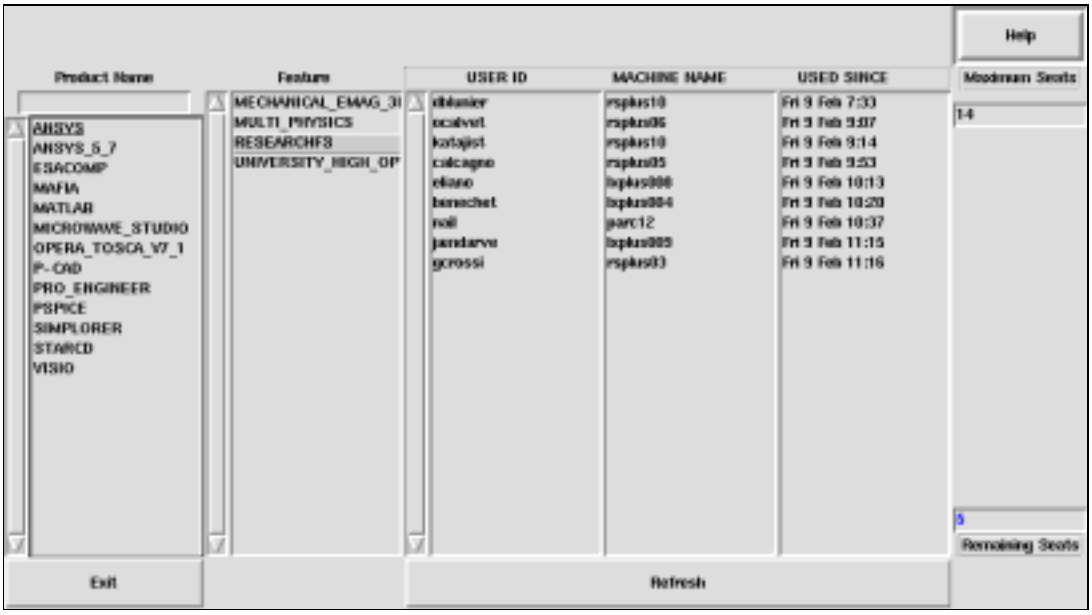

Figure 12 cestat: an end user tool to manage license sharing

Long Term Data and Usage Trends

One of the goals of this project was to reduce the amount of data (log-files) that need to be kept for monitoring many products over a long time period. License monitoring can generate enormous data files, which results in poor performance in any analysis made to extract statistics. Not all information contained in the log-files is needed in the long term. It is interesting to know for how much time an application has been used, but not essential to know in which exact time interval each connection was made.

It was decided to summarise the log-file data on a weekly basis in the following way:

- Users: Connection time by user for each feature of each product.
- Denials/Successes: Number of successful requests and denials of each feature.
- Maximum use: Maximum simultaneous use of a feature per week.
- Average use: Average over five working days of the maximum simultaneous use of each feature per hour.

This data sub-set has proved to be sufficient for our purposes and is also small enough to be kept over a long period of time in a database. Had we attempted to keep the detailed log-files over such a period, they would have amounted to perhaps a few GigaBytes of log-files which in turn would have implied unrealistic data processing times.

For generating the data, SAMReport was used together with a post-processor. The latter was a c-program, which allows one to request report logs of different products, and which outputs only the essential information. Finally the data was loaded into Oracle tables in the same database as used for the product and license data. This data processing was automated with a cron job, running once per week.

Graphical output, produced by Gnuplot, showing long term trends of product and individual feature use was made available on the web using automatically extensible web pages generated by stored PL*SQL procedures. This output step was also automated by a cron job.

Typical long term trend data are shown in Figures 13 a,b,c,d below. These examples are for a double wrapped product which exists in several versions and which is available on Unix and Windows. Single wrapping also shows the use of the individual versions on the different platforms.

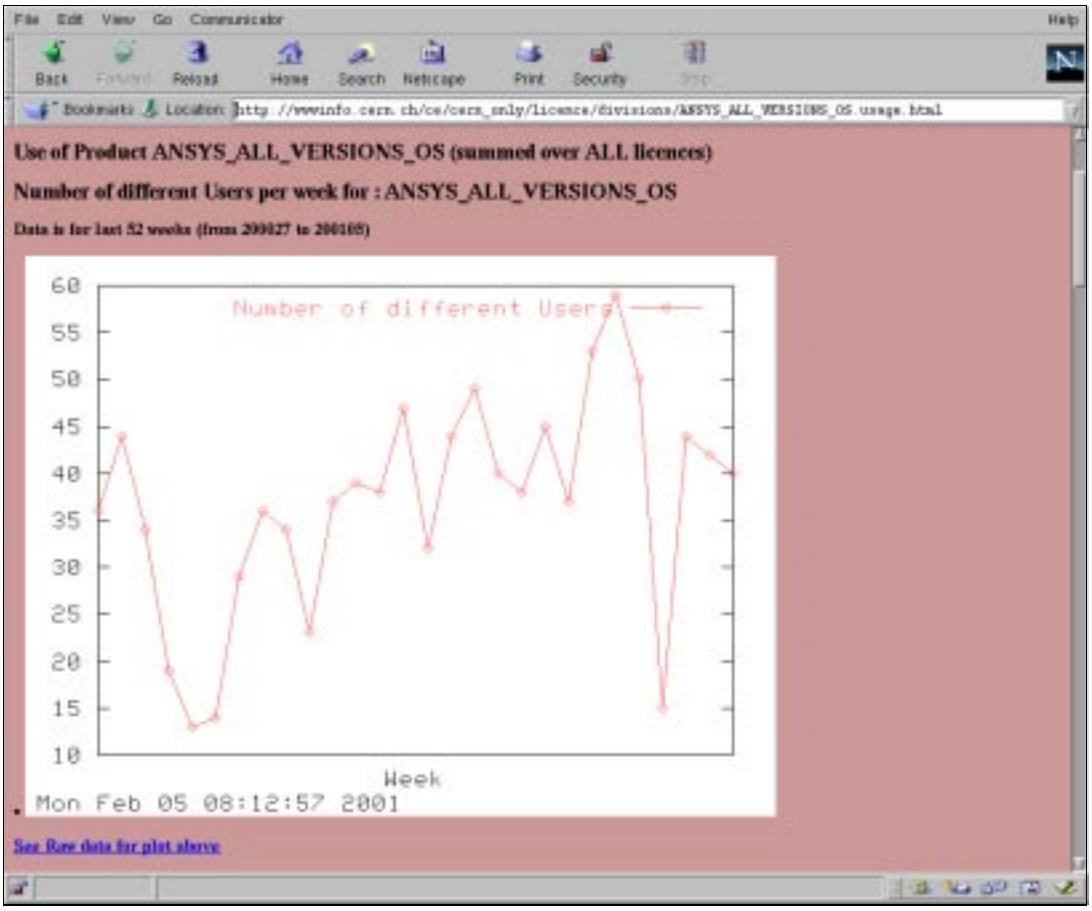

Figure 13a Long term trends for a double wrapped product

The above Figure shows the number of different users, as distinct from userids, per week for a product that is deployed on Windows and Unix and also across several product versions.

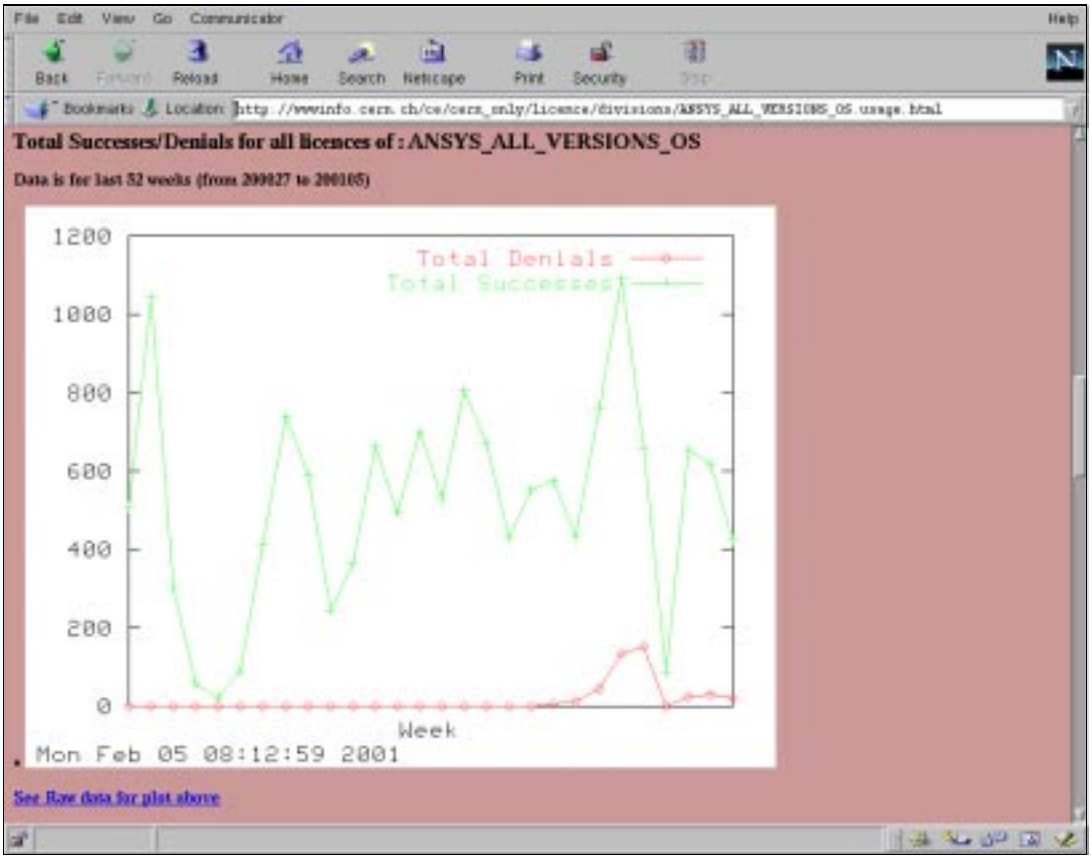

Figure 13b Long term trend for a double wrapped product

The above Figure shows the total successes and denials for a double wrapped product available on Windows and Unix and across several versions. It shows, for the product as a whole, how many times a license was denied.

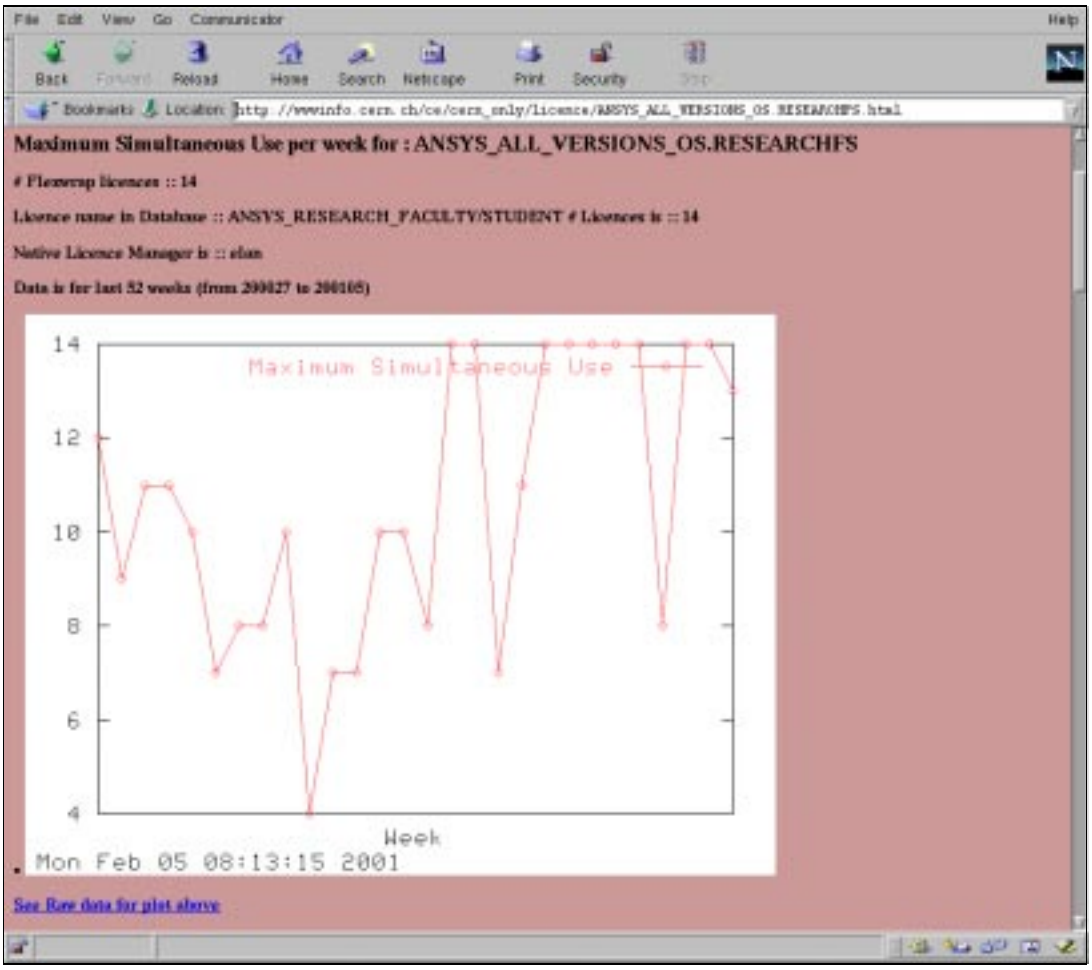

Figure 13c Long term trends for a double wrapped product

The above Figure shows the maximum simultaneous use of a specific feature of a double wrapped product available on Windows and Unix and also across several product versions. The native license manager is elan. In particular it shows how often the maximum number of commercial licenses used is equal to the maximum number available. Note that the number of Flexwrap tokens is equal to the number of available commercial licenses or product tokens.

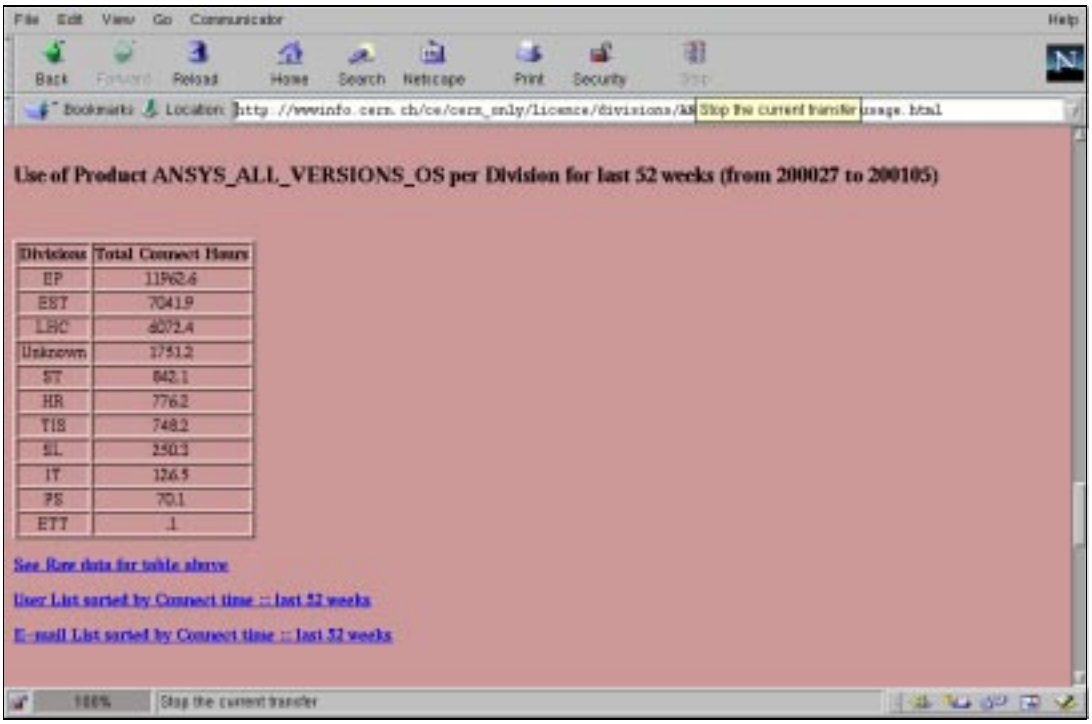

Figure 13d Product use per CERN division

In the CERN context it is interesting to understand the relative use per CERN division of each product. In addition links are provided to lists of users (and their e-mail addresses) sorted by time of use of the relevant product or feature.

Additional License Management Tools

The system has been used to implement several other useful management tools.

License Hogging

A way of dealing with license hogging (users holding onto licenses for an unreasonable amount of time) was also implemented. Early each morning, the status of all the features is examined. Users holding a license for more than a given time period receive an automatically generated e-mail in which they would be kindly asked to log off if they did not need to use the product any more. Often such hogging is inadvertent in that it is caused by a hung or improperly terminated session.

Applications run in batch mode on the Engineering CPU servers can take many days to complete. These are exempted from this type of checking.

Access Control Lists

The possibility of providing access control lists for some products was also implemented. The basic user lists are stored in the Oracle database and the resulting userid lists sent to SAMSuite or to the native license server for restriction control. Updates to the lists are generated automatically (as users arrive at or leave CERN) by cron jobs.

Communication Facilities with the User Community

The system provides automatic construction of E-mail lists of users using products or individual features of a product, in the last month, last 3 months or last year. Parallel user lists ordered by total license connect time (in the same time periods) enable CE group product managers to easily communicate with the relevant user community. These have proved very valuable to get new products or new versions of existing products into production and also to phase out obsolete versions.

As noted above, urgent communications are possible via the built-in Zephyr messaging facility.

Conclusion

We have implemented a comprehensive license monitoring scheme, based on SAMSuite, that provides a consistent handling of software products on both Unix and Windows platforms and which is independent of any native licensing scheme. All interfaces and data presentation are available on the web. The system has been in production for a year. Unless new products are added to the monitoring scheme the system runs automatically, based on the pre-scheduled execution of cron scripts. Introduction of new products is handled by updating a few well documented tables. Currently some 50 products are so monitored. The initial goals and requirements of the project have been fully met and the system has proved to be very useful in managing engineering applications on Windows and Unix at CERN.

Different technologies have been employed, including Oracle, DHTML, PERL scripts, gnuplot, c programs and Flexlm license monitoring products. Comprehensive documentation exists to make maintenance and further development easy.