ASIS, manage and distribute application software in the HEP community.

Ph. Defert [a] G. Cancio M. [a] L. Cons [a] E. Fernandez D. [a] M. Goossens [a] S. Gouache [a] C. Lecommandeur [b] O. Le Moigne [a] M. Ouwehand [b] A. Peyrat [a] A. Possoz [b] I. Reguero [a]

[a] *CERN, CH-1211 Geneva 23*

[b] *Ecole Polytechnique Fédérale de Lausanne*

We describe a project for distributing application software in the worldwide High Energy Physics (HEP) community. Hundreds of packages are maintained centrally and users can access them directly through the network. Access performance and reliability are optimised by specifying in installation scripts which packages are to be copied locally, or accessed remotely. Tools exist to generate packages from sources, the central file-base can be replicated manually or automatically and can be limited to propagation of changes only.

Key words: Software Generation, Management and Distribution.

ASIS (the Application Software Installation Server) provides an environment with a large number of freely available UNIX applications. End-users need not get, configure, generate, install or manage the software themselves. System maintainers are provided with tools to generate, install and manage packages.

The ASIS repository covers a large variety of application domains, such as High Energy Physics (HEP) data analysis software, most GNU packages, the latest TEX/ LATEX setup, MIT X-windows contributions, TCL/TK based software. Presently, ASIS offers executables for more than 600 products on eight different platforms for SUN, DEC, IBM, SGI, HP and PC computers.

The end-user accesses software of ASIS in any computer on site. The only change needed in the system setup is to run a single script, whenever the repository is updated. A tools suite helps product maintainers keep their packages up-to-date by automating the generation and installation process, paying special attention to quality and reliability control. Finally, remote site managers can replicate the ASIS repository on their own sites with the help of easy-to-use, efficient and configurable software.

1 Organisation of the Repository

In a heterogeneous environment, i.e., one with more than one architecture/operating system (O.S.) present, each package is partitioned in a **specific** part, dependent upon that O.S. and containing binary files and libraries, and a **sharable** part, containing scripts, include files, fonts, startup files, man pages, etc. which are usable on all platforms.

Each specific part contains references to the shared part.

In the repository, multiple versions of a package can coexist, but generally, only one is "InProduction", i.e., formally supported. A version of a product which is not "InProduction" is stored in a specific directory that contains all specific and shared data such as in the NIST depot [3]. "InProduction" products for a platform are all stored in the same directory tree structure. The correspondence between files and products is kept in the ASIS data base.

2 User Access to ASIS

A data base with description of all product and version description is stored in ASIS together with state and history information for each package in the repository. Each day, an HTML copy of the data base is generated using the `ASISinfo` library. In addition, the HTML files contain anchors pointing to the actual documentation distributed with the packages.
`ASISwsm`, the ASIS WorkStation Manager, is a perl script which builds on the local computer the directory(ies) where applications reside, usually /usr/local, so that all packages have the correct execution environment. Links are created from the directory /usr/local to a distributed file system, for instance at CERN: /afs/cern.ch/asis (AFS), /nfs/cern.ch/asis (NFS) or /:/asis (DFS). On top of that, a local system administrator can customise `ASISwsm` to ignore packages, to obtain a local copy or to choose specific versions of a product. A TK-based configuration editor (`tkwsm`) helps system maintainers to configure their workstation.

3 Product Maintainer's Environment.

Software Processing Model

The various steps that packages go through, from the moment that the sources are released until when the programs are delivered to the users are described by a "Software Processing Model" whose state/transition diagram shown in Figure 1.

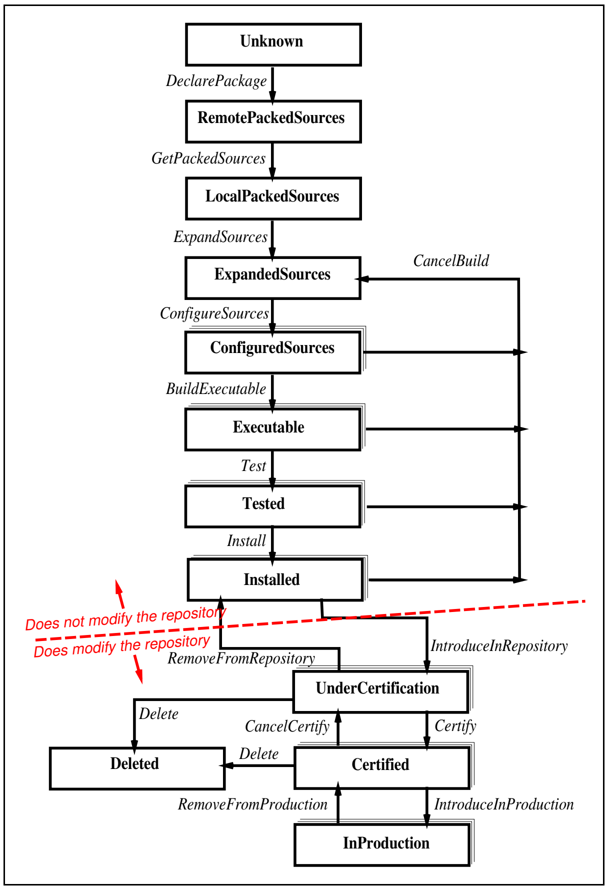

Fig. 1. The "Software Processing Model". The dashed line delimits the set of active transitions, which modifies the repository and thus the users' environment, and the set of passive transitions.

A complete description of the states and transitions can be found on the ASIS Web home page `http::/consult.cern.ch/dci/asis`.

Generation of Software

A tool called `happi`, which stands for "Heterogeneous Automated Product Processor and Installer", performs all passive transitions in the ASIS "Software Processing Model", `happi` can execute a single transition, change a product state or display the current state and its associated internal information. It does all necessary operations and dispatches the generation tasks to dedicated computers, each associated with a supported platform.
`tkhappi` is a TK based interface to `happi` which increases its usability and the readability of its output.
Generally, `happi` is automatically started when a new version appears in the LocalPackedSources directory. `tkhappi` is used to view logs or repair run-time errors. The actual work of a "product maintainer" consists in providing `happi` with a "configuration file", written in perl, describing all transitions.

Managing Versions in the Repository

The Software Processing Model satisfactorily

describes the actual operations performed on a package. However, modelling repository changes requires the introduction of a new concept. Indeed, when a product maintainer changes the version of a product which is "InProduction", two state transitions take place: (1) one version is *Removed*d*FromProduction*, (2) the new one is *Introduced*d*InProduction*. These operations should be executed in the right order and in an atomic way, and therefore a transaction system was introduced in ASIS. A transaction is a sequence of operations (transitions and/or specific actions) which are either run to completion or not the state of the repository remains untouched. For a transaction to be committed, many checks are made verifying access rights, file name conflicts, file system quotas, the available temporary space, etc. The program `tktes` is used by product maintainers to edit and submit ASIS transactions to `ASISctm` (the ASIS Central Transaction Manager). These transactions change the availability of products and versions. They are built as a sequence of transitions or special operations and submitted to `ASISctm` for approval. When all checks are successful, the transaction is committed.
The actual modifications of the repository are performed by `ASISlcm` (the ASIS Local Copy Manager) after the transactions are committed. `ASISlcm` takes the list of committed transactions and performs them.

Replication.

The different replicas are updated from a master which is not necessarily the "Reference Copy". `ASISlcm` reads the master ASIS copy to get all the transactions which were executed since the last update and performs them on its own repository. It is the system administrator of the replicating site who customises `ASISlcm` and decides when it is run.

4 The Status of the Project.

`ASISwsm` is part of the standard CERN installation procedure on all UNIX systems.
`happi` and `tkhappi` are used for maintaining most of public domain packages. `tktes` is not yet ready and presently replaced by an extended `tkhappi`.
`ASISlcm` is being tested with some remote sites and will be put in production as soon as they are validated.
EPFL, the Swiss Federal Institute of Technology of Lausanne recently joined the project.

References

[1] William Bliss, Jonathan Streets, Lourdu Udumula, and Margaret Votava. *UNIX Product Distribution, User's Guide*. Fermilab, FNAL, 1992.

[2] Ph. Defert et al. Managing and distributing application software. In *LISA'96 Proceedings*, pages 213–226, 1996.

[3] Kenneth Manheimer, Barry Warsaw, Stephen Clark, and Walter Rowe. The depot: A framework for sharing software installation across organizational and unix platform boundaries. In *LISA IV Proceedings.*, pages 37–46. Usenix, 1990.