DEPARTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERÍA DEL SOFTWARE

Facultad de Informática
Universidad Politécnica de Madrid

TRABAJO FIN DE CARRERA

Sistema de gestión de la configuración del software y de despliegue para la Plataforma PIO.

Autor
Carlos San Esteban Díez

Tutora
Genoveva López Gómez

Año 2009

A mis padres,
a mi hermano,
y a María

AGRADECIMIENTOS

En primer lugar, quisiera agradecer y hacer una mención especial a los profesores de la facultad que me acompañaron durante la realización de los proyectos en los que he estado involucrado: Dña. Genoveva López Gómez, D. Nicolás García Barcia, D. José Luis Morant Ramón y D. Francisco Javier Soriano Camino, a quienes agradezco el apoyo y la confianza que han depositado en mí.

En la misma medida quisiera agradecer a los jefes de proyecto que desde la empresa, Teldat S.A., Cristian Alonso, Arturo Plaza y Felipe Camacho, se han encargado de orientar nuestro esfuerzo en la dirección adecuada, dándonos la visión empresarial de los proyectos que hemos tenido entre manos. Sin la experiencia y conocimientos de todos ellos, no habría sido posible la realización de los proyectos en los que he participado desde mi entrada en el laboratorio de redes y más en concreto el que aquí se expone.

Quiero destacar y agradecer la ayuda y el apoyo que he recibido de toda mi familia, comenzando por mis padres María Piedad Díez Pérez y Felipe San Esteban Ibáñez, que son los que me han tenido que soportar durante todos estos largos años y que han realizado un gran esfuerzo para que yo pudiera seguir aprendiendo y desarrollando una labor muy intensa en la facultad, aprovechando las oportunidades y los retos que allí se me iban planteando. No puedo dejar de mencionar a mi hermano, Nicolás San Esteban Díez, que es la persona que me ha hecho poner los pies en el suelo y siempre me ofrece su particular visión del mundo. Al resto de mis familiares, mis abuelos, tíos y primos, que son muchos, les tengo que agradecer el calor que he recibido siempre a su lado.

No puedo dejar de nombrar a todos los compañeros que me han aportado sus conocimientos, experiencias y con los que he forjado una gran amistad durante todo el tiempo que he pasado en el laboratorio de redes de teleinformática de la Facultad de informática de la Universidad Politécnica de Madrid, entre los que se encuentran los profesores del Grupo de Redes de Computadores y Sistemas Distribuidos, Agustín

que hace posible poner orden en el papeleo, donde sólo hay caos y por supuesto a todos los compañeros que han pasado o están en el laboratorio, con los que he trabajado, por orden cronológico: Iván, Arturo, Ethel, Leandro, Sebastián, Ramón, Antonio, Eduardo, Vicente, Chema y Jesús, y también los demás compañeros del laboratorio con los que he compartido tiempo, conocimientos y ratos de ocio: Miguel, Elena, Guillermo, Vanesa, Diego, Ana, Rafael, Javier, Álvaro, Juanjo... todos ellos han dejado huella en mi persona.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

INTRODUCCIÓN

Índice

Este trabajo, *Sistema de gestión de la configuración del software y de despliegue para la plataforma* **P***aneles de* **I***nformación en* **O***ficinas (PIO)*, se enmarca dentro de un proyecto de investigación que se ha desarrollado conjuntamente por la Facultad de Informática de la Universidad Politécnica de Madrid y la empresa del ámbito de las telecomunicaciones.

La empresa fue quien tuvo la iniciativa de crear el proyecto **P**aneles de **I**nformación en **O**ficinas, conocido como proyecto PIO, y realizar la labor de investigación y desarrollo de prototipos con la colaboración de la Universidad, más concretamente con la colaboración del Laboratorio de Redes del Departamento de Lenguajes y Sistemas e Ingeniería del Software, debido al carácter distribuido, con redes heterogéneas, en el que se iban a desplegar algunos de los componentes de la plataforma que se iba a desarrollar.

Fruto de esta colaboración se llevó a cabo el desarrollo del prototipo de la plataforma PIO que, tras un periodo de maduración de la propia empresa, se incorporó a los servicios que dicha compañía ofrece a sus clientes. En la actualidad es un servicio que ha sido desplegado a nivel mundial, y que ha sido contratado por clientes de esta compañía tanto nacionales como internacionales.

Figura 1.1: Escudo de la Facultad de Informática

El proyecto PIO nació con el objetivo de difundir todo tipo de contenidos multimedia para transmitir información de interés al público a través de medios gráficos de gran formato y alta definición, como televisores **L**iquid **C**rystal **D**isplay (LCD) o pantallas de plasma, utilizando la infraestructura de red de banda ancha de que disponen la mayoría de las oficinas de cualquier empresa con diversas sedes dispersas.

Para lograr los ambiciosos objetivos que se habían marcado en este proyecto ha habido que enfrentarse, por su complejidad y su carácter claramente distribuido, a algunos de los problemas más importantes del software en la gestión de la configuración y por ende en su posterior instalación y mantenimiento.

Concretamente en este trabajo se aborda la problemática del control de la configuración del software de los equipos encargados de la reproducción de los contenidos multimedia, llamados PIOs, y del despliegue de la plataforma en los mismos, lo que aporta el control de los programas, las librerías y sus respectivas versiones desde la instalación inicial del sistema, hasta sus futuras ampliaciones y actualizaciones.

Por último, cabe señalar que el sistema que se realizó para el despliegue de la plataforma PIO también había de ser desplegado. Aunque este sistema requiere una actualización y mantenimiento mucho menor, exceptuando pequeñas mejoras para soportar el despliegue de la plataforma en hardware más moderno, no es necesaria una gestión de la configuración tan estricta, pero si un proceso de despliegue (la instalación del instalador) acorde a la función que desempeña, que también se aborda en el seno de este trabajo.

1.1. Motivaciones y Objetivos

Este trabajo se ha realizado para cubrir las necesidades que la empresa identificó al realizar junto con la universidad el primer prototipo de la plataforma PIO. Este prototipo se realizó mediante modificaciones manuales en la configuración de

muchos de los programas y librerías que se instalaban en un equipo diseñado para la reproducción de los contenidos en el destino, de ahora en adelante denominado PIO.

La principal preocupación de los responsables de la empresa se centró en la reducción del tiempo y los costes de despliegue de la plataforma en los equipos que se utilizarían para reproducir los contenidos, PIOs. Asimismo, otra de las características identificadas, fue la necesidad de proveer a la plataforma de un mecanismo para que se pudiera personalizar la misma con pequeños cambios para algunos clientes (generar distintas versiones, dándoles la imagen del cliente), e incluso, poder lanzar actualizaciones de la plataforma cuando la misma ya estuviera *online* para añadirle funcionalidades o proveerla de actualizaciones de seguridad. Esta gestión de versiones o variantes presenta el problema del doble mantenimiento.

Para facilitar la gestión de configuración del sistema, se eligió como línea base de la plataforma, la distribución de software libre Debian GNU/Linux. La elección de esta distribución se debió principalmente por la estabilidad y seguridad que aportaba, por el gran número de programas disponibles y las herramientas de gestión con que es mantenida, reconocidas en el mundo del software libre como unas de las más potentes. Todo esto viene avalado por ser la distribución más versionada, dando lugar a múltiples distribuciones derivadas como Ubuntu, Kubuntu, Knoppix, LinEx ...[1]

La ventaja de la utilización de esta distribución se encuentra en que el software de la misma se organiza y distribuye en forma de paquetes binarios que contienen las aplicaciones, librerías y archivos de configuración, e incluyen las referencias a otros paquetes de los que tiene dependencias para su correcto funcionamiento o para ampliar alguna de sus funcionalidades.

Otra de las ventajas de la distribución es la clara diferenciación que se realiza entre el software que es completamente libre y el que tiene algunas restricciones. Esto ayuda a evitar incurrir en infracciones de licencias al utilizar este software en el proyecto.

En un trabajo anterior se desarrolló el prototipo de la plataforma basado en una instalación de Debian GNU/Linux en su versión Sarge. La plataforma consistió en una base con los paquetes estándares mínimos a la que se añadieron los paquetes, programas y librerías de la propia distribución necesarios para la plataforma.

Casi todos los paquetes provenían de los repositorios oficiales de la distribución, pero ciertas aplicaciones no estaban empaquetadas o la versión distribuida era

[1] 133 distribuciones derivadas según http://distrowatch.com/ a 15/06/2011

antigua y carecía de alguna característica necesaria, que sí era provista por alguna versión posterior de esa misma aplicación. Estas aplicaciones fueron instaladas, bien de forma manual sin recurrir al gestor de paquetes, bien recurriendo a la instalación de paquetes de otros repositorios no oficiales, como *backports*, o bien se generaron los paquetes a partir de los paquetes fuentes de otras versiones.

Algunos de los componentes que se debieron instalar manualmente fueron el núcleo del sistema operativo, Linux, debido a la configuración y compilación específica que hubo que realizar al mismo; el reproductor de contenidos multimedia MPlayer y los códecs de audio y vídeo para el mismo. Otro de los programas de la propia distribución que se tuvo que modificar fue el navegador Firefox.

Además de modificar o añadir programas existentes, se desarrollaron otros completamente, que eran responsables de algunas funciones específicas de la plataforma, como el **T**eldat **I**P **D**iscovery **P**rotocol **D**aemon (TIDPD) o el **MO**tor de **VI**sualización (MOVI) que también se instalaban manualmente. Estos desarrollos han sido objeto de otros trabajos de fin de carrera de otros compañeros del proyecto PIO.
.

Todo lo anterior requería la edición manual de los archivos responsables de la configuración de varios de los demonios y programas que la plataforma necesitaba para su correcta ejecución, como Racoon [IPSec, 2011], responsable de los túneles **I**nternet **P**rotocol **Sec**ure (IPSec), el demonio de **S**ecure **Sh**ell (SSH), y todas las opciones de configuración que el programa de instalación de una distribución como Debian requiere durante su instalación interactiva desde el CD de instalación.

Todo esto se traducía en una instalación muy costosa en tiempo y la necesidad de que se ocuparan de su realización técnicos con una mayor experiencia y preparación, lo que se traduce en costes más altos. Por estas razones se decidió automatizar al máximo posible la instalación de la plataforma, evitando en todo momento que el técnico responsable de iniciar la instalación tuviera que utilizar un teclado y a ser posible que tampoco necesitase leer ningún mensaje de la pantalla.

Para otros proyectos similares, la empresa había realizado despliegues manuales desmontando el dispositivo de memoria permanente del equipo a instalar y copiando la información al mismo realizando una imagen uno a uno de otro dispositivo igual, volviendo a montar el dispositivo en el equipo, lo que hacía el proceso lento, costoso y nada versátil ya que las características del dispositivo de almacenamiento debían ser exactamente iguales para realizar la copia.

Recientemente se mejoró el sistema anterior utilizando el software comercial de otra empresa, Norton Ghost [Symantec, 2008], para copiar los datos del dispositivo de almacenamiento por red. Esto mejoraba en coste y tiempo el sistema anterior pues ya no era necesario desmontar el dispositivo, pero se seguían realizando copias exactas del disco completo con lo que el tiempo seguía siendo muy alto y se daban los mismos problemas de versatilidad.

Estos problemas se debían a que no les proporcionaba dicho software un medio para controlar la configuración del software desplegado en los equipos, ya que para cambiar de versión había que realizar el mismo proceso desde el principio, comprendiendo estos pasos: generar de nuevo el prototipo, generar la copia maestra para su replicación y realizar el despliegue en todas las máquinas incluso en las ya desplegadas anteriormente (borrando el contenido anterior).

Dados el estado del proyecto PIO y las motivaciones discutidas en esta sección se definieron los siguientes objetivos del trabajo de fin de carrera que aquí se expone y que se van a desarrollar a continuación.

Automatización del despliegue y la configuración de la plataforma. Se trata de minimizar al máximo la interacción humana en el proceso por los problemas que este tipo de procesos repetitivos sufren, por despistes o errores de los operarios, y el coste en el que se incurre haciendo menos competitiva la explotación de la plataforma PIO.

Reducción del tiempo empleado para realizar el despliegue de la plataforma. La instalación en los equipos del sistema operativo, Debian GNU/Linux, más el conjunto de aplicaciones y librerías que conforman finalmente la plataforma, además de la configuración inicial de todo el software instalado, es un proceso que durante la producción de los equipos tiene que ser lo más ajustado en el tiempo posible, optimizado para poder disponer en poco tiempo del mayor número de equipos preparados para su puesta en producción.

Gestión de componentes software. Se debe elegir el medio de empaquetado y distribución de componentes software que mejor se ajuste a las necesidades de explotación. En primer lugar, es necesaria la clara identificación de cada componente y su respectiva versión, así como los componentes y sus respectivas versiones de los que depende. Asimismo, debe permitir realizar la instalación y desinstalación de forma limpia y automática, por ejemplo, resolviendo las dependencias en tiempo de instalación.

Gestión de múltiples configuraciones. Para soportar el ciclo de vida de la plataforma PIO, es necesario mantener varias configuraciones simultáneamente. Esto es debido a que es posible gestionar versiones personalizadas para algunos clientes y, de forma paralela, las asociadas a los diferentes estados del proceso de desarrollo: versión en desarrollo, pruebas o producción.

Actualización de la configuración de equipos ya desplegados. Una vez ubicados en su destino los equipos con una versión de la plataforma desplegada y dado que el acceso a los mismos está limitado por diseño de la arquitectura del servicio PIO, es necesario un mecanismo que permita mantener actualizada la configuración de estos equipos. Esto implica que los equipos puedan comprobar la existencia de cambios en la configuración y que en caso de existir puedan realizar la correspondiente actualización. Además, se debe obtener únicamente los componentes software nuevos y aquellos a actualizar.

Replicación del sistema de despliegue. Finalmente se añadió como objetivo la realización de un proceso mediante el cual se pudiese replicar el sistema de despliegue de forma que se pudiera obtener, de un modo sencillo, la posibilidad de recuperar en el menor tiempo posible el sistema de despliegue de un fallo en el hardware u otras contingencias, así como poder tener varios de estos sistemas funcionando en distintos lugares.

1.2. Descripción del Trabajo

Durante el desarrollo de este trabajo de fin de carrera se aborda el problema de la gestión de la configuración del software de un sistema complejo, así como la realización de la instalación inicial de los equipos que formarán la infraestructura para la reproducción distribuida de contenidos multimedia.

Para lograr los objetivos que se marcaron al comienzo del trabajo, se estudiaron las diferentes posibilidades de gestión y empaquetado del software, buscando la infraestructura más adecuada. En este proceso se observó que para la plataforma PIO era necesaria la utilización de software libre ya que de esta forma se potenciaba su competitividad por la no inclusión de los costes de licencias de software en el producto final. Por este motivo se descartaron los sistemas operativos de Microsoft.

Dentro del software libre se realizó una comparativa de las diferentes distribuciones más extendidas hasta la fecha, observando principalmente las ventajas

que nos ofrecían las propias herramientas de gestión del software que tenía la propia distribución, la disponibilidad y cantidad de el software de la distribución y las restricciones o licencias que tenía dicho software.

De entre las distribuciones que se propusieron, se eligió Debian como base para el proyecto, debido a que sus repositorios son los que más y mejor organizado tienen el software, como se puede ver en la tabla 1.1 , y por las excelentes propiedades que aportan los paquetes deb para la gestión de la configuración del sistema, junto con el gestor de paquetes dpkg y las herramientas disponibles para la gestión de las dependencias en la paquetería como apt o aptitude.

Distribución	Número de paquetes	Gestor de paquetes	Formato del paquete
Arch Linux	15000	pacman	tar.gz
Debian	25113	apt	deb
Fedora	8000	yum	rpm
Gentoo	80 (13000 fuentes)	portage	ebuild

Tabla 1.1: Distribuciones, paquetes y su gestión. Fuente [Wikipedia, 2009]

Sobre esta base se realizó el estudio de las tecnologías que se utilizaban en otros entornos para el despliegue del sistema operativo y la configuración del software instalado. Finalmente se desarrolló un sistema de instalación a medida de las necesidades del proyecto PIO, basado en otro sistema genérico, distribuido como software libre y utilizado para la instalación de los nodos de máquinas distribuidas como los servidores Beowulf, adaptándolo a las necesidades de este problema en concreto y añadiéndole algunas otras funcionalidades que en ese momento no estaban disponibles, ni en éste, ni en otros sistemas de despliegue similares.

El trabajo desarrollado es capaz de realizar instalaciones masivas en los equipos sin necesitar personal altamente cualificado para satisfacer la posible demanda de un potencial cliente que requiriese una gran cantidad de PIOs para cubrir sus necesidades. Es decir, es el intento de poder poner en el mercado, en el menor tiempo posible, un gran número de equipos con la plataforma PIO.

Para lograrlo, se desarrolló un sistema de instalación automatizada que cumpliese ciertas características: debía ser un proceso rápido, mecánico, con las mínimas intervenciones necesarias por parte de los operarios que asistiesen la operación y concurrente, es decir, que se pudiesen *plataformar* varios equipos simultáneamente desde un único servidor de instalaciones.

Simultáneamente, se consiguió utilizar las herramientas de gestión del software de la distribución, para conseguir que el software instalado en los equipos pudiera ser mantenido y actualizado remotamente, manteniendo el control de la configuración de los programas instalados, sus versiones y dependencias, en los diversos equipos de todos los clientes.

En definitiva, se consigue que, desde la instalación y despliegue inicial de la plataforma, hasta sus posteriores actualizaciones e instalaciones de nuevas características, tener un perfecto control de la configuración de los equipos. La gestión de la misma también se realiza una vez que los PIOs han sido entregados en su destino de forma remota y sin intervención humana en el lugar de su ubicación, evitando desplazamientos costosos.

La reutilización y adaptación de las herramientas de software libre existentes en Debian ha sido la pieza clave para alcanzar los objetivos que se habían propuesto al inicio y llevar este trabajo junto con todo el proyecto a buen puerto. En la actualidad, el sistema PIO, incluyendo la contribución de este trabajo de fin de carrera y de otros que lo precedieron y tras su transferencia tecnológica se ha convertido en un servicio rentable de la empresa.

Capítulo 2

ESTADO DE LA CUESTIÓN

Índice

El trabajo que se desarrolla en este proyecto intenta solucionar, en el contexto del proyecto PIO, dos importantes problemas del software, como se ha comentado anteriormente, la gestión de la configuración del software y el despliegue del sistema operativo junto con las herramientas de software asociadas al mismo para su gestión y para desarrollar el propósito final del sistema.

En el momento en el que se desarrolla este trabajo, los sistemas de gestión de la configuración y de despliegue, se encuentran en una fase de cambios y mejoras continuas. A continuación, se ofrece una breve reseña histórica explicando la evolución hasta la actualidad de este tipo de sistemas, que a lo largo de la corta historia de la informática han ido cambiando, especializándose gradualmente y complementándose. Esto es, la forma en que se distribuye y se gestiona el software que se instala sobre los distintos sistemas operativos, así como los medios y formatos en los que se despliega el sistema operativo.

La distribución y gestión del software está asociada a la evolución en la historia de los sistemas de computación. En la década de los 50 aparecieron los primeros computadores que hasta mediados de los 60, estaban limitados a ejecutar un único proceso y no existía la posibilidad de la interacción en tiempo real. Cada máquina de cada fabricante ofrecía una arquitectura análoga pero incompatible con respecto a las demás y el software de sistemas era un complemento añadido que no se concebía

como un producto independiente. En parte se debía a que su desarrollo y su distribución se realizaba con ficheros en ensamblador y código máquina.

A mediados de los 60 aparece la multiprogramación y el tiempo compartido, dando lugar a la interactividad de las aplicaciones. Simultáneamente se produce una explosión de bibliotecas de código fuente con aplicaciones y utilidades que sacan partido a las nuevas posibilidades. Se hacen patentes los problemas asociados al mantenimiento y distribución de estas colecciones de software, siendo dominantes los costes de mantenimiento. No se gestionaban adecuadamente los *bugs*, no se tenían sistemas de gestión de versiones, no se tenían herramientas para la automatización de la construcción del software o eran rudimentarias, en definitiva, no se tenían las herramientas adecuadas para ninguna de estas y otras tareas relacionadas con el desarrollo y mantenimiento del software.

2.1. Gestión del software

Las aplicaciones que se utilizan habitualmente en los ordenadores personales y que podemos iniciar fácilmente mediante acciones sencillas se han incluido en el sistema mediante alguna de las posibilidades que se presentan a continuación ubicando todas sus dependencias: bibliotecas, archivos de configuración e incluso otras aplicaciones, de una forma adecuada para que la propia aplicación las pueda encontrar y utilizar.

En la actualidad, un usuario final percibe los procesos de instalación y despliegue del software como algo simple puesto que en la mayoría de casos se limita a interactuar con un asistente de instalación. Sin embargo, es un proceso bastante complejo que se adapta al sistema operativo de destino y que debe tener en cuenta las preferencias del usuario o incluso depender del momento en el que se realiza. Un ejemplo claro son los distintos sistemas operativos que existen y otro no tan evidente son los distintos idiomas que soportan dichos sistemas.

Cualquier aplicación software sufre una serie de cambios de forma antes de que se pueda utilizar por el usuario final. Estos cambios se producen en las distintas etapas por las que ha de pasar desde que se crea el código fuente: construcción, empaquetado, distribución e instalación de las aplicaciones.

Este proceso se ha desarrollado de diversas formas adaptándose cada vez más a las necesidades de los usuarios para alcanzar a un mayor número de ellos más

fácilmente. En las secciones siguientes se abordan las distintas aproximaciones al problema, partiendo de las más simples a las más elaboradas.

2.1.1. Manual

Un claro ejemplo de este tipo de entorno de gestión del software es el que se da en Linux From Scratch (LFS), que basa su distribución en la documentación necesaria para generar un sistema operativo básico desde los fuentes de los desarrolladores, con las herramientas de construcción de software integradas en los propios paquetes de desarrollo o en otros paquetes que han de ser construidos con anterioridad y que normalmente son referidos como dependencias del desarrollo.

En este tipo de entorno lo primero que se ha de definir es el *toolchain*, que es la especificación del compilador y su versión concreta, así como el resto de utilidades para la programación, en este caso concreto, GNU Compiler Collection (GCC), GNU make, GNU Binutils, GNU Bison, GNU m4, GNU build sytem (autotools): Autoconf, Automake y Libtool. Por otro lado, también se deben definir las librerías que se van a utilizar como base para el sistema operativo y que posteriormente se pondrán a disposición del resto de aplicaciones que se quieran incorporar al mismo. Esta especificación debe ser muy precisa y la elección de uno u otro conjunto de herramientas de desarrollo se vuelve crítico para poder incorporar más aplicaciones a posteriori.

Abstrayéndonos de otras complicaciones propias de la arquitectura y hardware disponible, así como de la configuración que se haga del mismo, en este caso concreto y en la versión 8.0 de LFS expondré a continuación la lista de herramientas que se han de construir de forma manual para generar una instancia de este sistema operativo: Binutils 2.27, GCC 6.3.0, Linux 4.9.9 API Headers, Glibc 2.25, Libstdc++ 6.3.0, Tcl-core 8.6.6, Expect 5.45, DejaGNU 1.6, Check 0.11.0, Ncurses 6.0, Bash 4.4, Bison 3.0.4, Bzip2 1.0.6, Coreutils 8.26, Diffutils 3.5, File 5.30, Findutils 4.6.0, Gawk 4.1.4, Gettext 0.19.8.1, Grep 3.0, Gzip 1.8, M4 1.4.18, Make 4.2.1, Patch 2.7.5, Perl 5.24.1, Sed 4.4, Tar 1.29, Texinfo 6.3, Util-linux 2.29.1, Xz 5.2.3.

Estos paquetes de software son sólo la lista más básica de las herramientas que se han de construir, para poder tener una base capaz de gestionar e instalar uno de los sistemas operativos más básicos basados en Linux.

Cada uno de ellos, en general, se han de descomprimir, configurar, compilar e instalar de una forma adecuada para su posterior utilización. Cualquiera de estos paquetes de fuentes puede generar una o varias aplicaciones o librerías que serán ejecutadas para poder finalizar el proceso de instalación y después durante toda la vida útil del sistema, necesitando a su vez el mantenimiento y gestión adecuadas para su actualización y reparación de problemas.

2.1.2. Empaquetado de aplicaciones

Las aplicaciones empaquetadas, son aquellas en que la manera de distribuirlas consiste en proporcionarlas en forma de paquetes, llamados en inglés '*software bundle*' o '*application bundle*'. Estos paquetes se encuentran formados por los ejecutables de la aplicación, así como por sus librerías y otros ficheros como imágenes, ficheros de audio, traducciones y localizaciones, etc.

De esta forma se proporciona todo ello como un solo conjunto, simplificando la gestión de la configuración de la aplicación, ya que las librerías de las que depende el programa, que pueden ser tanto dinámicas como estáticas, se integran en un sólo archivo.

El empaquetado de aplicaciones permite evitar los problemas de las dependencias tanto a la hora de instalar la aplicación como a la hora de usarla, ya que cada paquete lleva consigo sus dependencias, y la instalación o desinstalación de otro software no va a afectar a las dependencias de dicho paquete.

Esta es la principal, sino la única, ventaja del empaquetado de aplicaciones, se evitan la problemática de las dependencias, y la aplicación se puede trasladar de un equipo a otro sin necesidad de reinstalarla, ya que el paquete de la aplicación contiene todos los ficheros necesarios para ejecutarla.

Sin embargo, presenta una desventaja, estos paquetes ocupan mucho más espacio en el disco, esta es una de las razones por las que los paquetes MSI o los DMG suelen ocupar mucho espacio en disco.

2.1.3. Empaquetado de componentes

El conjunto típico de una distribución de Linux contiene un núcleo del sistema, herramientas y bibliotecas, software adicional, documentación, un sistema de ventanas, un administrador de ventanas, un entorno de escritorio...

Además, gran parte del software incluido es de fuente abierta o software libre y distribuido por sus desarrolladores tanto en formato binario como mediante un paquete de código fuente, permitiendo a sus usuarios modificar o compilar el código fuente original, adaptándolo a su infraestructura o al API de las librerías que ya están en el sistema, mediante modificadores aplicados durante la compilación.

El principal aporte que una distribución de software libre da a la sociedad es que prácticamente todo el software empaquetado en una versión concreta utiliza la misma versión o ABI de las librerías disponibles en el sistema, de manera que se puedan compartir los componentes y así reducir tanto la memoria de disco consumida al instalar nuevas aplicaciones, como la memoria RAM consumida cuando se utilizan varias aplicaciones que comparten librerías.

En estos entornos los formatos más extendidos son los paquetes '*deb*' en aquellas distribuciones basadas en '*dpkg*' que es gestor de paquetes de *Debian*, o '*rpm*' que es la extensión de los paquetes de las distribuciones basadas en RedHat.

2.2. Instalación de sistemas operativos

La gestión de la distribución de aplicaciones y las distintas formas de organizarlas, no tiene mucho sentido sin la existencia del sistema operativo que es el encargado de homogeneizar los distintos sistemas y arquitecturas hardware, así como de proporcionar herramientas, como por ejemplo llamadas al sistema, que permiten la ejecución de los programas, incluso aquellos que sirven para instalar otros programas, como se explicó anteriormente.

El problema que viene a continuación trata de instalar un sistema operativo, ya que, de forma previa a su instalación, no se dispone de las herramientas que aporta y, por lo tanto, tampoco se dispone de las herramientas de despliegue y gestión de software que se han mencionado.

Es por este motivo que la instalación del propio sistema operativo es una de las operaciones más difíciles y delicada que se debe realizar a cualquiera de los equipos

informáticos, ya que implica, en muchos casos, unos conocimientos de bajo nivel sobre el equipo y su arquitectura. Sin embargo, no es necesario que el administrador de sistemas aplique todo este conocimiento en cada instalación puesto que se han desarrollado aplicaciones que asisten en este proceso y que abstraen de estos detalles.

Estos asistentes han experimentado muchos cambios desde la aparición de los sistemas operativos.

En los 50 los equipos de computación se vendían sin ningún software asociado y eran los clientes quienes desarrollaban su propio software, de esta forma los primeros "sistemas operativos" eran esencialmente colas de planificación de tareas, desarrollados individualmente por los clientes para intentar tener las carísimas máquinas ocupadas sin esperar la puesta en marcha de los operarios la siguiente tarea a ejecutar. Este planificador era desarrollado para una máquina concreta y era desplegado en la misma siempre de la misma forma por el mismo personal. No existía el concepto de distribución software, ni herramientas de despliegue.

Entre 1955 y 1957 se diseñó y desarrolló el primer embrión de un sistema operativo fruto de una cooperación entre General Motors Research y North American Aviation, llamado GM-NAA I/O System. Este sistema se utilizó en unas 40 unidades de IBM 704, a las que proveía de procesamiento por lotes, sin interacción con el usuario, mejorando el número de trabajos realizados por unidad de tiempo. Como se ha apuntado anteriormente los despliegues de este sistema se realizaban manualmente adaptándolo a cada instalación por personal compuesto por los propios desarrolladores y gestionado por operarios especializados.

En los 60 la industria de los ordenadores comienza a distribuir sistemas operativos primitivos junto a sus sistemas, IBM lanzó la serie System/360 de máquinas introduciendo el concepto de un único Sistema Operativo para todas ellas, el OS/360 que se distribuye junto a sus equipos, con la promesa de poder portar cualquier tarea de una máquina a otra, sin necesidad de rediseñarla y compilarla.

El resultado fue un sistema operativo enorme para su época, el tamaño que se planteó en su diseño era de 6KB, teniendo en su primera versión 64KB. Desde esa versión tuvieron lugar múltiples mutaciones y variaciones, todas ellas con los mismos problemas, un tamaño excesivo y defectos en su ejecución. El sistema siguió incrementando su complejidad llegando a convertir su tamaño en un problema como no había surgido anteriormente. El sistema se programó en ensamblador y no se tenían las herramientas adecuadas para gestionar y mantener el software, ni para distribuirlo y adaptarlo a cada instalación.

A partir de ese momento los sistemas operativos fueron creciendo en funcionalidades y en tamaño, necesitando una herramienta específica para su instalación, y aunque esta herramienta en muchos casos es capaz de obtener información descubriendo los dispositivos conectados y la red a la que el equipo está físicamente conectado, siguen existiendo diferentes grados de automatización durante el proceso del mismo, dando lugar dos tipos de instaladores claramente diferenciados, los interactivos y los automáticos.

2.2.1. Interactivo

En este tipo de instalación es imprescindible la actuación de un operador para dotarle de parámetros de configuración que lo adecuen al desempeño para el que se desea que vaya a realizar.

2.2.2. Automático

Esta instalación es la que se puede realizar cuando se conoce perfectamente el resultado de la instalación que se quiere obtener, y por lo tanto se preparan las imágenes, dependencias, o lo que sea para poder instalar varias veces la misma configuración incluso en equipos diferentes.

Capítulo 3

PROYECTO PIO

Índice

Como se ha mencionado anteriormente, este trabajo, se encuentra dentro del Proyecto **P**aneles de **I**nformación en **O**ficinas (PIO), este proyecto se ha realizado en un marco de colaboración con la empresa Teldat, con el objetivo de desarrollar una plataforma sobre la que realizar la distribución de contenidos multimedia.

Tras una fase de transferencia tecnológica, mediante la cual los resultados obtenidos en la universidad fueron trasladados al entorno de producción de Teldat, ésta pasó a ofrecer esta tecnología como un servicio para sus clientes bajo la denominación comercial *Servicio PIO*.

El servicio PIO se constituye como un servicio de valor añadido, ofrecido por Teldat, para la difusión de contenidos multimedia a todas las sedes de una corporación desde un centro de control que determina qué se muestra al público, dónde y cuándo. Citando la descripción publicada por la propia empresa [Teldat S.A., 2005]:

> *El PIO es un servicio abierto de gestión, almacenamiento y difusión segura de contenidos multimedia a cualquier punto remoto de una corporación*

garantizando la entrega y en base a unas políticas de transmisión y emisión preestablecidas.

Estos contenidos multimedia que se distribuyen a los diferentes puntos remotos pueden estar compuestos de imágenes, vídeo, sonido, animaciones, texto, etc. en cualquier formato compatible con los reproductores (PIOs), que soportan los formatos más extendidos. Junto con estos contenidos también se les envía la planificación de la emisión que determina en qué momento se reproducen unos contenidos u otros.

Este tipo de sistema reproduce la información en grandes pantallas que en comparación con el medio estático utilizado en la cartelería tradicional añade muchas más posibilidades derivadas de la experiencia multimedia. Por este motivo se denomina *cartelería dinámica* o *Digital Signage* a estos sistemas que aportan una serie de beneficios a las corporaciones que lo utilicen, especialmente a aquellas que actualmente utilizan otros soportes, como el papel, para publicitar o dar a conocer sus nuevos productos o servicios en sus oficinas.

Algunas de las ventajas que ofrece la utilización de este servicio para los clientes son una imagen moderna y dinámica, agilidad, garantías de seguridad y un bajo coste.

Imagen moderna y dinámica. Se aportan una imagen más moderna y corporativa a las sedes donde se implanta el servicio, teniendo un canal informativo más directo, llamativo y dinámico, que supone un nuevo medio de comunicación y una herramienta de marketing directo para los departamentos de Marketing.

Agilidad. Debido a lo anterior, captan la atención más eficazmente de los usuarios hacia los que va dirigida la información que se distribuye a través del sistema, proporcionando a la entidad, mucha más agilidad para desplegar los nuevos contenidos, mejorando los tiempos de reacción de las campañas de marketing, agilizando y haciendo más eficaz la comunicación con el cliente.

Garantías de seguridad. Además, el servicio incrementa la fiabilidad y no introduce riesgos de seguridad, ya que se hace uso del protocolo IPSec que garantiza la autenticación, confidencialidad y no repudio de las acciones realizadas sobre el sistema.

Bajo coste. Por último, pero no por ello menos importante, reduce los costes de mantenimiento, debido a la reducción en costes de logística y amortiza parte de los costes globales de las infraestructuras de comunicaciones. El servicio permite indicar

las franjas de transmisión de contenidos y estas se pueden adecuar a los tiempos ociosos de las líneas de comunicación ya contratadas.

3.1. Actores del sistema

Para entender mejor el funcionamiento del servicio se identifican a continuación los principales actores que interactúan con el sistema y sus cometidos.

- **Corporación**. Es el cliente que implanta el servicio PIO para la difusión de contenidos multimedia en sus redes remotas.
- **Departamento de Marketing**. Responsable de la programación de las campañas de marketing que se difunden y emiten a través del servicio PIO.
- **Agencia de publicidad**. Colabora con del departamento de Marketing en la elaboración de campañas, elaboración del plan de medios al que agregará el PIO y creación de los contenidos de la campaña.
- **Sede o punto remoto**. Cualquier ubicación remota de la corporación sea esta una oficina, almacén, etcétera, donde se quieran emitir los contenidos de la campaña.
- **Teldat S.A.** Empresa que oferta el servicio PIO mediante el cual se hace viable la emisión de las campañas que la Agencia de Publicidad ha creado en colaboración con el departamento de Marketing de la corporación.

Estos actores se interrelacionan entre sí para hacer posible la existencia del servicio. Se muestra de una forma sencilla el esquema de interrelación de los agentes en la figura 3.1. Tal y como se observa en la misma existen interacciones entre algunos de los diferentes actores.

Se expone, a continuación, la naturaleza de las tres interacciones resaltadas en la figura y las diferentes responsabilidades que asumirán para el correcto funcionamiento de todo el sistema.

1. **Departamento de marketing – Agencia**: Se trata de una comunicación interna entre ambos departamentos o empresas para identificar qué contenidos y cómo se quieren difundir los mismos. El sistema pone a disposición de estos actores una interfaz de comunicación con el servicio:

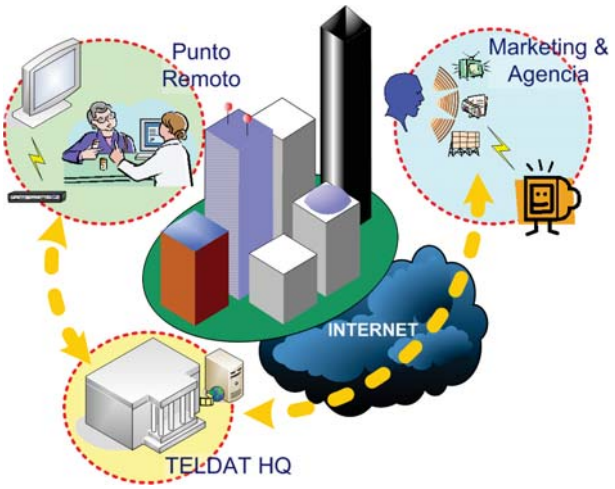

Figura 3.1: Actores del proyecto PIO

a) **Agencia**: La agencia se conectará a través de una aplicación web, que el servicio pone a disposición de este tipo de usuarios, mediante la cual podrá realizar la "subida" de contenidos al servidor destinado a tal efecto por Teldat.

b) **Departamento de Marketing**: Podrá llevar a cabo el establecimiento de las políticas de emisión, es decir, la programación de contenidos para su emisión en franjas horarias determinadas.

2. **Teldat – Puntos remotos**: El servicio tiene mecanismos para identificar la necesidad de transmitir contenidos desde el servidor de almacenamiento a uno o varios puntos remotos cuando estos no están actualizados. Además, esta transferencia se realiza de acuerdo con las políticas de transmisión establecidas por el Departamento de Marketing, esto es, en las franjas horarias en las que se determina que el tráfico o bien no interfiere o interfiere poco con el tráfico corporativo.

3. **Punto remoto – Pantalla**: El punto remoto dispone de toda la programación y contenidos que debe emitir. El software de la plataforma PIO se encarga de presentar los contenidos programados en las pantallas que tenga asociadas (por ejemplo, una pantalla de gran formato LCD) en el momento en que se disparen los eventos de visualización.

Una vez establecidos los roles de los diferentes actores del sistema y las relaciones que se establecen entre ellos, debido a la utilización del servicio PIO, se obtiene una clara imagen del modelo de dominio que se trata en este trabajo y en el que se

adentrará en el siguiente apartado con la introducción de la arquitectura de la plataforma.

3.2. Arquitectura de la plataforma PIO

La plataforma PIO es un sistema de *Cartelería Dinámica* cuyo principal objetivo es el de proporcionar, a las empresas interesadas, una forma de mostrar contenidos en sus oficinas o sucursales a sus posibles clientes y/o usuarios. El carácter dinámico del sistema, implícito en la forma de mostrar la información, proporciona a los contenidos mostrados en el mismo, un atractivo extraordinario debido a la experiencia que ofrecen los contenidos multimedia, lo que mejora la difusión de la información publicitada entre el público objetivo.

Estos sistemas hacen uso de pantallas de grandes dimensiones, como se puede observar en la figura 3.2 para mostrar todo tipo de contenidos: vídeos, texto animado y, en general, cualquier tipo de información multimedia que pueda ser reproducida en un ordenador.

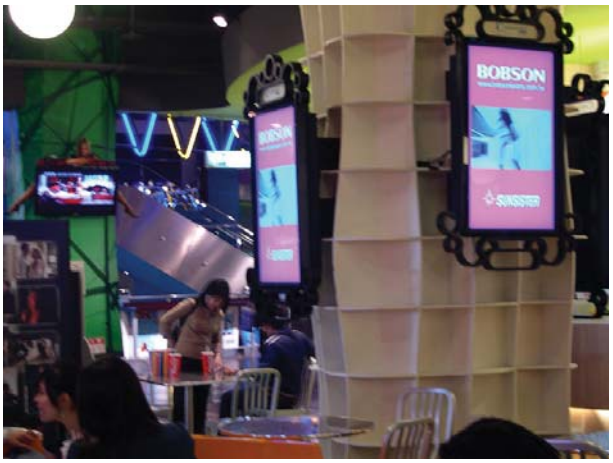

Figura 3.2: Pantallas de Cartelería Dinámica

Debido a que los sistemas de reproducción de los contenidos no quedan a la vista, cabe pensar que el funcionamiento de estas pantallas está basado en la reproducción cíclica de un medio físico estático como un **D**igital **V**ersatile **D**isc (DVD) que contiene los medios a reproducir organizados según el orden en que se desea que se muestren.

No obstante, este enfoque adolece de una serie de problemas, que lo convierte en una solución poco atractiva. La organización de los contenidos en la pantalla y durante

el tiempo de reproducción es un proceso arduo, al cual se añadiría la edición de un DVD, que fijaría el orden en que se han de reproducir los contenidos, imposibilitando cualquier cambio futuro. Además, la distribución de los DVDs implica desplegar una logística cara y pesada para poder actualizar los contenidos de equipos situados en distintas sedes de distintas ciudades.

Frente a esta aproximación, los sistemas de cartelería actuales ofrecen una alternativa de distribución y generación de contenidos ágil que no exige la utilización de soportes físicos de almacenamiento, como DVDs, **C**ompact **D**iscs (CDs), discos duros portátiles o unidades USB, etc. sino que utilizan las infraestructuras de red existentes para este fin. Simultáneamente se puede cambiar la organización de todo el material durante su distribución e incluso alterarlo en función de otros parámetros como la hora o el día en que se reproducen.

En este sistema de cartelería dinámica se pueden identificar cuatro componentes bien diferenciados por las responsabilidades que desempeñan:

- **Reproducción de los contenidos**: encargada de la correcta visualización y sincronización de los contenidos mostrados simultáneamente, así como de las transiciones.
- **Descarga de datos**: gestiona la descarga segura de los contenidos multimedia, las planificaciones y las actualizaciones de software.
- **Planificación de contenidos**: permite organizar los contenidos para su reproducción, dotando al cliente de las herramientas para diseñar la planificación de los mismos en base a parámetros como el horario o el día en que se reproducen.
- **Gestión del servicio**: registra nuevos clientes y los equipos que necesita junto con toda la información necesaria para su correcta instalación y futuro mantenimiento.

La plataforma PIO, que se ha desarrollado en el laboratorio, se ha centrado en los tres primeros puntos, siendo el servicio de gestión un desarrollo de la propia empresa Teldat. El la figura 3.3 se puede observar el esquema de la red en la que la plataforma tendrá que prestar los servicios destacados anteriormente.

En dicha figura 3.3 se puede observar el servidor principal, denominado Central, desde el que se distribuirán los contenidos a todos los PIOs distribuidos por las distintas sucursales de todas las empresas suscritas al servicio.

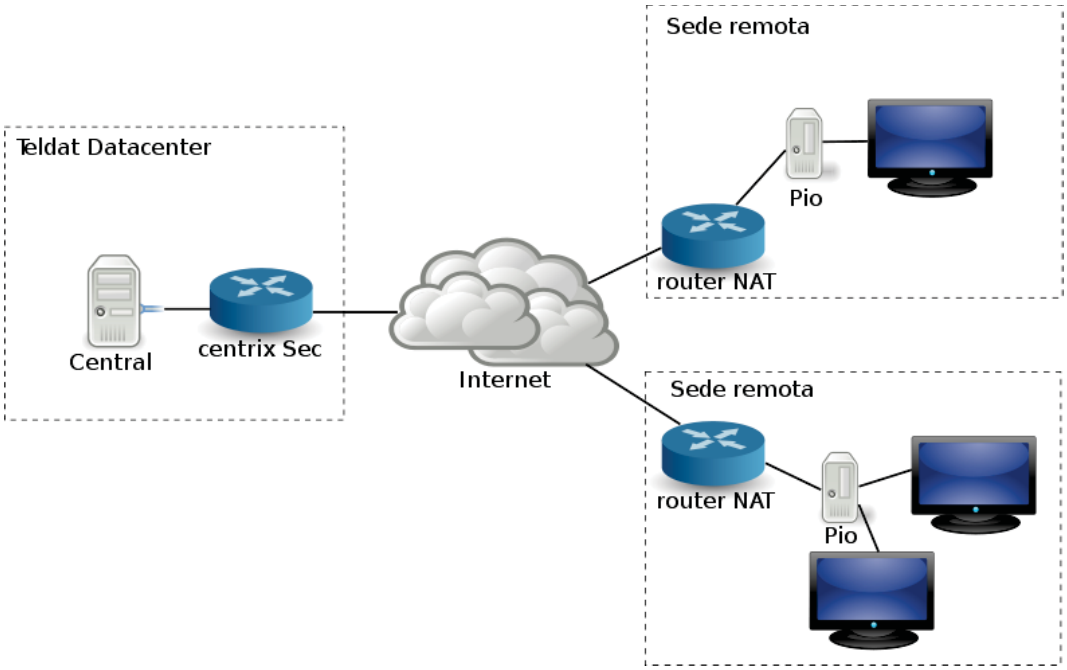

Figura 3.3: Esquema PIO

Esta plataforma intenta aprovechar la infraestructura de red que tiene la empresa cliente para las necesidades propias de su negocio y que en general se encuentra claramente infrautilizada, generalmente conectadas mediante servicios de ADSL o RDSI con conexión permanente, aprovechando, mediante las políticas que se definan en el servicio, los tiempos *ociosos* de la red, generalmente durante el horario nocturno.

La difusión de los contenidos se realiza a través de la red de datos pública, Internet, por lo que se utiliza una VPN para establecer un canal seguro entre los dos puntos, Central - PIO. Este escenario plantea algunos problemas:

- Los PIOs pueden estar en una red privada que realice **P**ort **A**ddress **T**ranslation (PAT), por lo que la iniciativa de establecer el túnel la ha de tener el PIO.
- El túnel no puede estar abierto permanentemente pues el concentrador de **V**irtual **P**rivate **N**etworks (VPNs) puede gestionar un número limitado de ellas.
- Para poder establecer el túnel **I**nternet **P**rotocol **Sec**ure (IPSec), se ha de tener la dirección IP de Central (que es fija) y sería recomendable tener, sobre todo para iniciar cualquier comunicación, la del PIO, pero éste puede tener una IP privada, que sólo tiene sentido dentro de la red en la que se encuentra o una IP dinámica o incluso ambas cosas.

- Para poder informar al PIO que ha de abrir el túnel, ya que este no puede estar siempre abierto, sería necesario abrir uno de los puertos del *router* de la sede del cliente y redirigirlo al PIO. Lo cual es un problema ya que exige la manipulación de los *routers* del cliente, algo que en general plantea muchos problemas.

Para dar solución a estos problemas un protocolo, llamado **T**eldat **I**P **D**iscovery **P**rotocol (TIDP), fue desarrollado. Su funcionamiento consiste en mantener abierto un puerto del *router* mediante el envío reiterado de un paquete muy simple, utilizando el protocolo UDP, con la mínima información necesaria para identificar el equipo en el *router* Centrix.

Mediante este mecanismo se mantiene una puerta abierta para poder, mediante otro sencillo paquete, comunicar al PIO que ha de establecer el túnel IPSec y, de esta forma, poner el equipo en *modo gestión*. Una vez es este estado se puede acceder a cualquiera de sus servicios como si fuera parte de un equipo de la red local.

Finalmente, una vez los contenidos multimedia y las programaciones son cargados en los PIOs, son reproducidas por los mismos, soportando la composición de varios elementos multimedia simultáneos en el/los televisor/es asociado/s. Por lo tanto, en una composición, como se puede observar en la figura 3.4, el PIO puede mostrar simultáneamente un vídeo con diferentes elementos de texto, FLASH, HTML...

Figura 3.4: Ejemplo de composición, imagen obtenida de [Teldat S.A., 2006]

3.3. Empaquetado del software

Como se ha dicho anteriormente se eligió la distribución Debian GNU/Linux por las características de la misma y en especial por su sistema de paquetería. Las

características más importantes de los paquetes binarios de Debian son las siguientes:

- Facilita la instalación y el mantenimiento del sistema.
- Contiene los ficheros y los meta-datos de los mismos: permisos, propietarios y ubicación.
- Registra en base de datos los ficheros instalados, diferenciando los de configuración para una correcta desinstalación o actualización.
- Controla las dependencias y compatibilidades.
- Utiliza mecanismos seguros para la instalación, desinstalación y reemplazo, incluso durante el funcionamiento.
- Contiene información sobre el paquete y autenticación de su autoría.

Otras características propias de esta paquetería son los criterios que se siguen para su nombrado y organización. En el nombre de cada paquete está perfectamente identificado su contenido, la versión del paquete fuente "upstream" (versión liberada por el desarrollador original), la versión de Debian (revisiones realizadas por los mantenedores de Debian a los fuentes originales) y la arquitectura para la que ha sido compilado. Como parte de este esquema de organización, el nombre de los paquetes codifica la información necesaria para identificarlos siguiendo el esquema de figura 3.5. Debido a este esquema, el popular editor *vim* tendría este nombre: `vim_2:7.2.148-2_amd64.deb`, y sus dependencias se pueden observar en la figura 3.6.

❶ ❷ ❸ ❹
`NomPaquete_VersUP-VersDeb_Arq.deb`

❶ Nombre del paquete, normalmente es el nombre con el que se publica pero en minúsculas y evitando caracteres extraños.
❷ Número de versión del software empaquetado.
❸ Número de revisión de Debian. Como parte del empaquetado y adaptación a las políticas de Debian se aplican parches al software original y estas versiones también deben poder diferenciarse. Esto es especialmente importante para las actualizaciones de seguridad.
❹ Arquitectura para la que se ha compilado el paquete binario. La mayoría de paquetes son compilables para todas las arquitecturas soportadas por Debian, que superan sobradamente la decena.

Figura 3.5: Esquema de nombrado de paquetes en Debian

Los paquetes se organizan por varios criterios, como la estabilidad y las pruebas que se han realizado al software del paquete y a la compatibilidad con el resto de software de otros paquetes o la adhesión del software a estándares libres. De esta forma un paquete puede formar parte de la rama *stable*, *testing* o *unstable* dependiendo de la versión y de las librerías que utiliza o contra las que fue compilado. Estas ramas van siendo reemplazadas pasando desde *unstable* a *stable* como se puede ver en la figura 3.7. Adicionalmente cada rama se organiza en áreas de archivo dependiendo de las condiciones de su licencia. El software completamente libre se agrupa en el área *main*, el que siendo libre depende de software que no lo es, en *contrib* y el software no libre en *non-free*.

Con estas características ya expuestas, se decidió utilizar la versión estable de Debian GNU/Linux, que era la versión 3.1 con nombre clave *sarge* en ese momento. En concreto se utilizó sólo el área *main* de sarge, que fue descargada incluyendo toda la paquetería y sus fuentes en un servidor local para mantener la configuración del software y tener controladas las posibles actualizaciones de software que pudiera surgir en la propia distribución. De esta forma establecimos la línea base del proyecto en lo que al sistema operativo respecta.

En este repositorio local se decidió crear un área propia que se llamó *teldat* donde se irían añadiendo los paquetes que se fuesen generando con los distintos programas y librerías que se desarrollaban para los PIOs dentro del proyecto. De esta forma teníamos separado el software que venía directamente de la distribución y el que se desarrollaba en el proyecto que como es lógico se tenía que empaquetar siguiendo los mismos requisitos y aprendiendo a utilizar las herramientas propias de la distribución que se estaba utilizando.

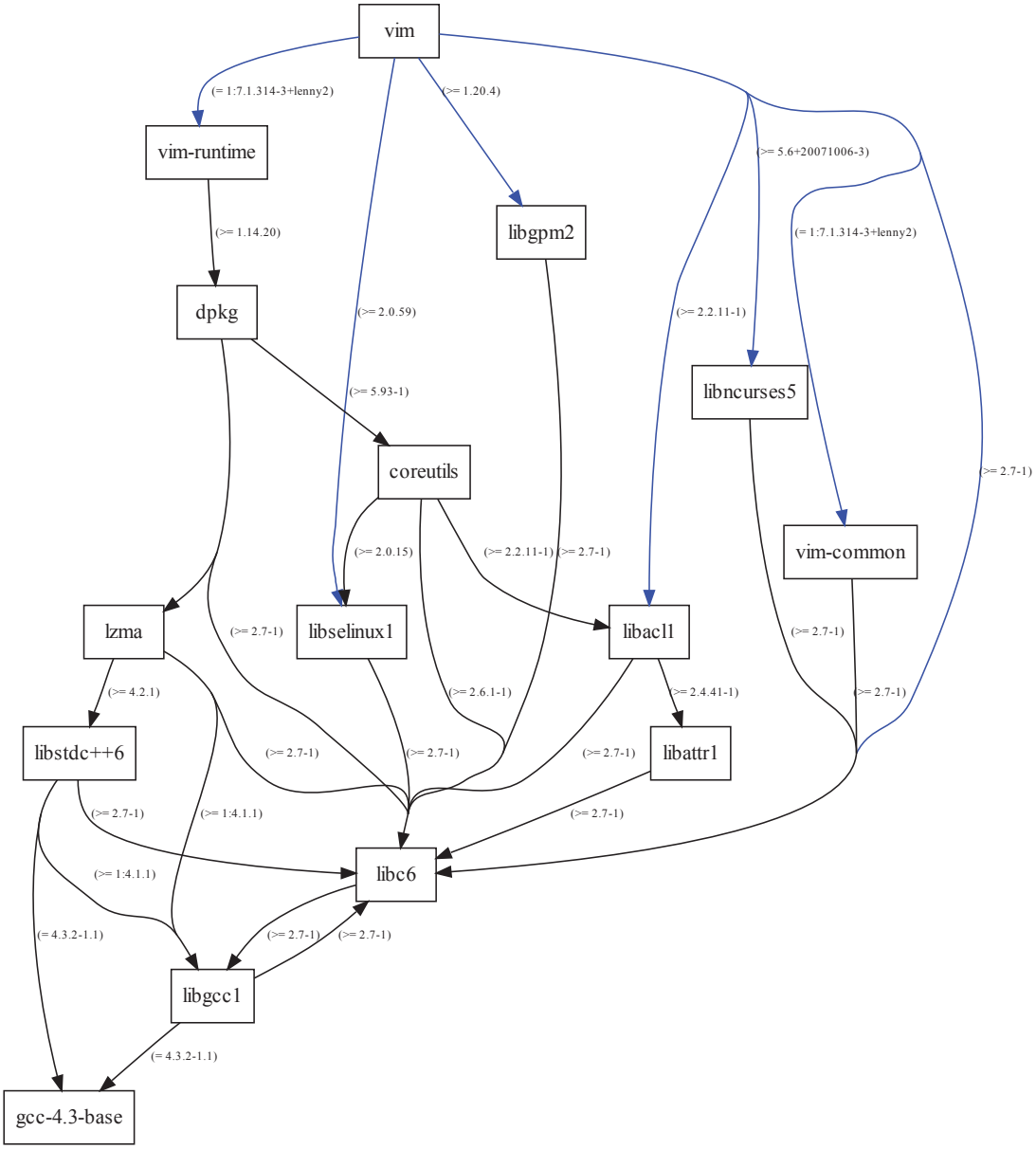

Figura 3.6: Grafo de dependencias del paquete vim

htbp

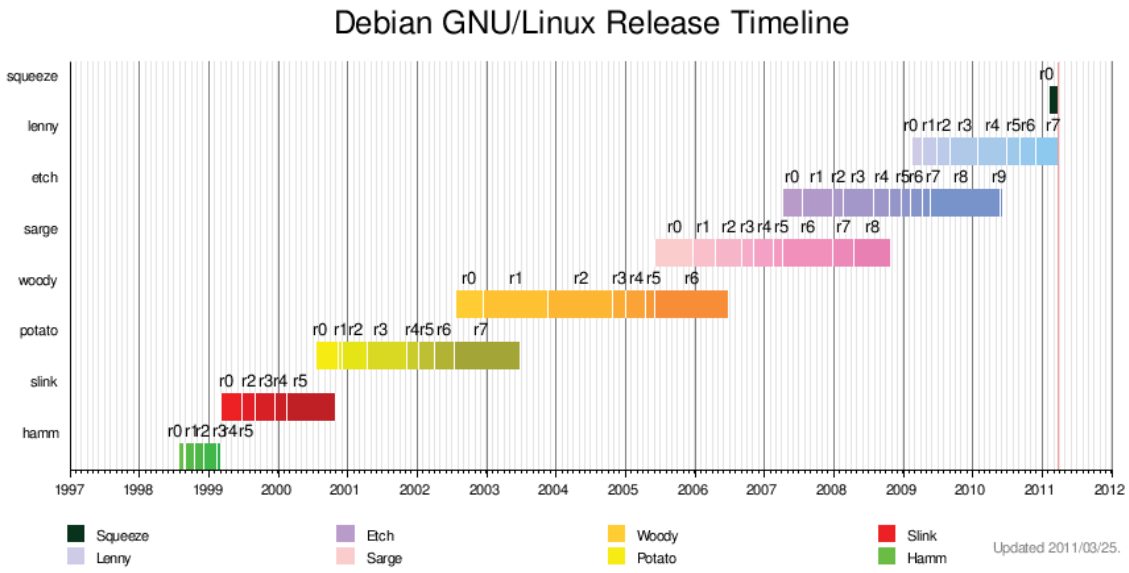

Figura 3.7: Línea temporal de las ramas estables de Debian. Fuente [Wikimedia Foundation, 2011].

3.4. Ciclo de vida de la plataforma PIO

La plataforma PIO se entiende como todo el software necesario para el funcionamiento de los equipos remotos. Esta plataforma tiene que poder adaptarse a cada cliente para cubrir sus necesidades, además de poder ser instalado automáticamente y tiene que tener la posibilidad de actualizarse remotamente.

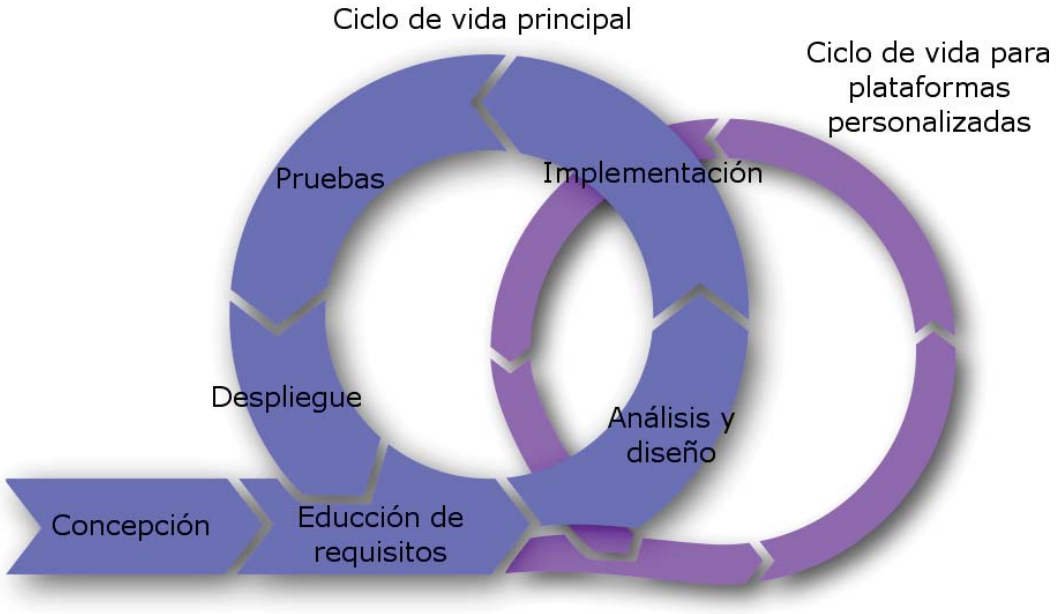

Figura 3.8: Ciclo de Desarrollo de la plataforma PIO

Desde la empresa Teldat se requirió poder desplegar conjuntos de actualizaciones o "service packs" en los equipos remotos para la actualización de los mismos de manera automática ya fuera para corregir *bugs* o para añadir nuevas funcionalidades a estos equipos. Por tanto, había que implantar un sistema de gestión del software instalado en los mismos y el mecanismo para que se actualizasen con las mejoras sin que produjese ningún impedimento a la reproducción de contenidos.

Por lo tanto, el software de estos equipos tenía que actualizarse a través de Internet de forma segura, con garantía de no comprometer la estabilidad e integridad de los mismos, ya que cada vez que se estropease uno, se tendría que enviar a un operario a repararlo, disparando los costes de mantenimiento de la plataforma. Es decir, una actualización con errores de los equipos podía transformarse en cientos de lugares a los que acudir para repararlos, con los costes que eso supondría.

Por todo lo anterior, las actualizaciones se realizan después de realizar pruebas locales y bajo un control de la configuración absoluto sobre qué, cuándo, cómo y a qué equipos se realizan actualizaciones, manteniendo en todo momento la información de la versión de la plataforma que está utilizando cada uno de ellos.

Capítulo 4

SISTEMA DE GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

Índice

El objeto de este proyecto, aunque se puede entender como dos sistemas independientes, como se desarrolla a continuación, también puede ser considerado como un único sistema integrado debido a las restricciones de diseño que el sistema de instalación establece para el resto del proyecto y también a la inversa, ya que las tecnologías que se utilizan por ambos sistemas se interrelacionan, teniendo fuertes dependencias que marcan considerablemente el proyecto y establecen los mecanismos y procesos que se han de utilizar en la gestión de la configuración del software de los equipos ya instalados y la forma en que se instalan inicialmente.

En cierta manera, todo el sistema se podría establecer en un solo servidor que proporcionase todos los servicios simultáneamente, aprovechando las dependencias comunes, principalmente el repositorio de paquetes y el servidor web que los sirve.

Durante el desarrollo fue así, pero en el entorno de producción se separaron los dos servicios por razones de conectividad, ya que el servicio de despliegue ha de funcionar únicamente en una red local por motivos técnicos y de seguridad. Sin embargo, el servicio de actualización y mantenimiento ha de ser accesible por

Internet, a través de túneles seguros IPSec, que permiten administrar y gestionar todos los equipos remotos.

El problema a resolver por este subsistema es facilitar la instalación inicial de la versión deseada de los componentes software, que en todo momento se pueda conocer de manera inambigüa qué versiones están instaladas y que se puedan realizar actualizaciones posteriores de forma automática y óptima (e.g. copiando el menor número de componentes desde el servidor de actualizaciones).

Los objetivos que se persiguen manteniendo la gestión de la configuración de un sistema como este son:

- Llevar el control y registro de los cambios con el fin de reducir los problemas evitando los errores que pueden acarrear una incorrecta sincronización en los cambios.
- Mantener la integridad de los equipos que se instalan a lo largo de la vida útil del sistema.
- Realizar el seguimiento y control de las actividades y de los recursos que intervienen en el desarrollo del sistema.
- Proporcionar un marco común de referencia para la definición y puesta en marcha de planes específicos de aseguramiento de la calidad aplicables a los proyectos concretos que forman parte de este sistema.

Simultáneamente el sistema de gestión de la configuración adoptado nos facilita las siguientes actividades:

- Mantener el sistema, aportando información precisa para poder valorar el impacto de los cambios y reduciendo el tiempo de aplicación del mismo.
- Llevar un control y registro de los cambios que se le aplican al sistema.
- Mantener el sistema de actualización e instalación correctamente alineados para minimizar los tiempos de despliegue del sistema en los equipos y sus actualizaciones.

4.1. Arquitectura del sistema de gestión de la configuración

El sistema que mantiene la gestión de la configuración se encuentra en un servidor y como se puede ver en la figura 4.1 está compuesto de tres elementos diferenciadas:

- Servidor web, en este caso Apache HTTP, común a todos los paquetes gestionados por el sistema, mediante el cual se puede acceder a toda la información de los repositorios de paquetes mediante el protocolo HTTP.
- Un componente enfocado en proporcionar un *mirror* o espejo de parte de la versión base de Debian sobre la que el sistema se desarrolló. Inicialmente fue Debian Sarge, pero fue migrado a Debian Etch.
- Otro componente para mantener un repositorio no estándar con los paquetes empaquetados con software del sistema PIO o software de terceros parcheado o específicamente empaquetado por no estar disponible en la distribución estándar.

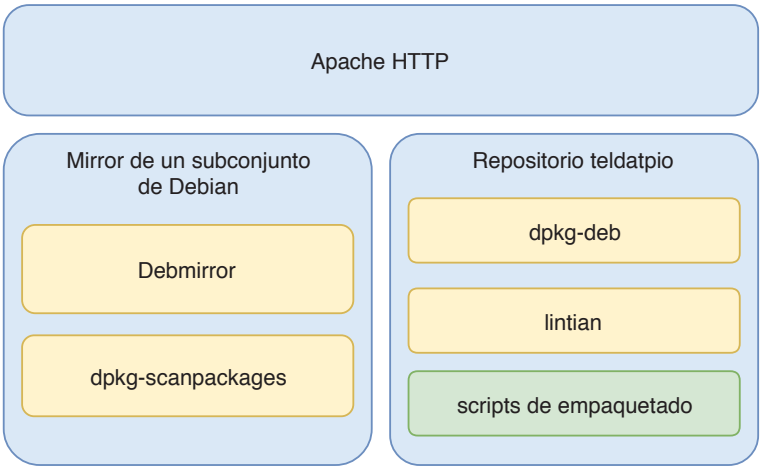

Figura 4.1: Diagrama de componentes

La gestión de la configuración está repartida entre los dos componentes que alojan todos los paquetes que se van a instalar. Ya que cada paquete de que está disponible tiene un apartado de control que define la versión del paquete y las dependencias que este tiene con respecto al resto del sistema.

Así la gestión de la configuración del conjunto de paquetes de Debian, lo mantenemos con las herramientas debmirror que nos permiten crear una imagen de

la situación de toda la paquetería de la distribución seleccionando las arquitecturas que necesitamos. De entre los que se seleccionan sólo aquellos que son necesarios para nuestro equipo y sus dependencias con la herramienta de la distribución *dpkg-scanpackages*. Con este proceso de sincronización y selección de las dependencias manteníamos la linea base del sistema, y se podía controlar los cambias que la distribución va realizando en sus servidores, para poder valorar si se deben incorporar esos cambios también a nuestro sistema.

Durante el desarrollo incluso cambiamos de versión de Sistema Operativo instalado, pues algo después del comienzo con la versión Sarge de Debian GNU/Linux, se obsoleto la versión, convirtiéndose la siguiente versión Etch en la estable de referencia y por lo tanto mantenida por la distribución. Tras un análisis de los cambios, se tomó la decisión de actualizar la línea base del sistema, ya que los equipos eran perfectamente compatibles y nos reportaba beneficios en el mantenimiento y también algunas otras funcionalidades.

El repositorio privado de paquetes denominado *teldatpio* es donde se desplegaban los paquetes con los programas y configuraciones que se habían desarrollado como parte del proyecto PIO para añadir y configurar todas las funcionalidades que necesitaban tener los equipos instalados.

Estos paquetes se desarrollaron idénticos a los paquetes de la distribución manteniendo la misma estructura de control, de manera que se podían desplegar de la misma manera que el resto de la distribución, aunque se mantuvieron en un repositorio independiente para controlar sus cambios y ampliaciones de manera separada.

4.2. Gestión del software basada en paquetes Debian

El repositorio de los paquetes es el elemento que mantiene los metadatos de todos los archivos que componen la distribución. Estos son almacenados siguiendo una jerarquía. De forma que los metadatos de una distribución se encuentran en el repositorio en '*dist/nombre-distribución*'. La estructura de su repositorio se puede recorrer con un navegador web. Existen seis tipos de metadatos clave:

- Release, descripción del repositorio e información de integridad.

- Release.gpg, firma del fichero Realease con la clave privada del repositorio.
- Contents-'*architecture*', listado de todos los ficheros para todos los paquetes en el archivo.
- '*distribution*'/'*area*'/binary-'*architecture*'/Release, descripción utilizada por las reglas de apt_preferences.
- '*distribution*'/'*area*'/binary-'*architecture*'/Packages, todos los debian/control de los paquetes binarios.
- '*distribution*'/'*area*'/source'/Packages, todos los debian/control de los paquetes de fuentes.

Las herramientas de la distribución utilizan estos archivos para en función de la distribución y arquitectura seleccionada obtener la lista de paquetes y sus dependencias para poner a disposición del usuario la posibilidad de instalar cualquier elemento de la distribución.

4.2.1. Funcionamiento de la paquetería Debian

La organización de los paquetes está distribuida en versiones, así siempre está disponible la rama estable o '*stable*', en pruebas o '*testing*' y la inestable o '*unstable*', a las que normalmente también se las identifica con un nombre, que al comienzo de este proyecto era Sarge, Etch y Sid respectivamente.

A lo largo del tiempo la rama en pruebas se convierte en estable, siguiendo la política de versionado que impone la distribución, dando lugar a una nueva rama a la que se le asigna un nuevo nombre y que ocupará la rama de pruebas.

Dentro de las ramas la distribución también se distinguen los paquetes por áreas, existiendo tres componentes diferenciados:

- main, paquetes que cumplen la **D**ebian **F**ree **S**oftware **G**uidelines (DFSG).
- contrib, paquetes que cumplen la DFSG pero con dependencias que no lo cumplen.
- non-free, paquetes que no cumplen la DFSG.

El proyecto se basó única y exclusivamente del area main del archivo de Debian, lo que garantizaba la utilización de software totalmente libre. Para poder valorar el tamaño de las 3 áreas, la versión de Debian Sarge constaba de 15400 paquetes binarios.

4.2.2. Líneas base de Debian

Para el proyecto se hizo necesario realizar una congelación de la distribución, y una imagen de la misma para disponer de todos esos paquetes y de sus fuentes, y no depender de los repositorios externos, que por ser de dominio de la distribución podían cambiar con la evolución de la misma.

Para este fin se utilizó un script de la propia distribución que te permita realizar el mirror mediante este comando: debmirror PATH –host=http://ftp.es.debian.org –root=/debian –dist=sarge –section=main,main/debian-installer –arch=i386,amd64

También se sincronizaba el repositorio de la distribución de actualizaciones de seguridad, ya que en algunas ocasiones se publican con antelación, los paquetes que corrigen vulnerabilidades encontradas para lo que se utilizaba el mismo script pero configurándolo de esta manera: debmirror PATH –host=http://security.debian.org/ –root=/ –dist=sarge/updates –section=main –arch=i386,amd64

Finalmente, con todos estos paquetes se creaban unas imágenes de las que después se obtenían sólo los paquetes que eran necesarios para instalar el sistema **P**aneles de **I**nformación en **O**ficinas (PIO) como se puede observar en el ejemplo de la imagen 4.2

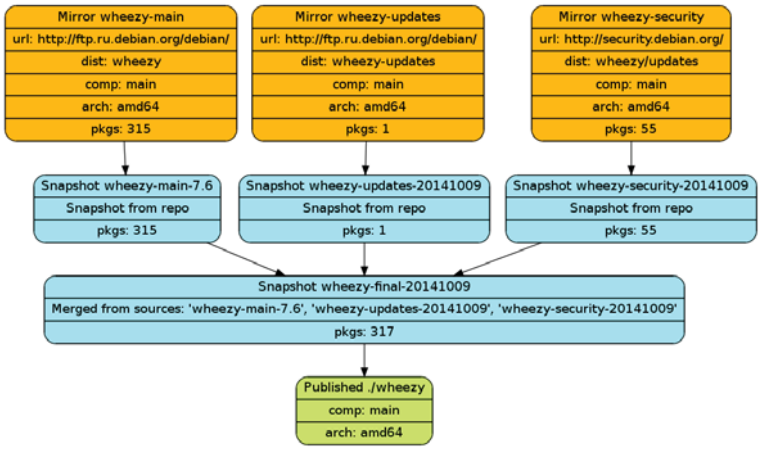

Figura 4.2: Imagen para la instalación

4.2.3. Empaquetado de la plataforma PIO

Los paquetes del repositorio privado que contienen las aplicaciones, librerías y configuraciones del proyecto, está compuesto por un número de paquetes que tienen

Instalación	Núm. paquetes	Núm. ficheros	Tamaño en disco
Sarge estándar	687	81.427	1.165 MB
Sarge *headless*	229	29.969	345 MB
Sarge reducida	180	18548	263 MB

Tabla 4.1: Comparación de los tamaños de Debian Sarge en función del tipo de instalación.

a su vez en debian/control sus dependencias fijadas de manera que una vez desplegados los paquetes adecuadamente en el servidor, sólo es necesario que a las herramientas del sistema se les pida la instalación del paquete teldatpio, ya que este paquete debido a sus dependencias, obliga a instalar todos los paquetes necesarios para la adecuación del sistema.

Además, las dependencias se establecen con la versión precisa de la que depende de manera que se garantiza la correcta instalación de todos los componentes respetando la gestión de la configuración establecida con esas reglas.

En la figura 4.3 se puede observar todo el diagrama de dependencias que genera el paquete mencionado y las dependencias que van surgiendo a su vez de las mismas.

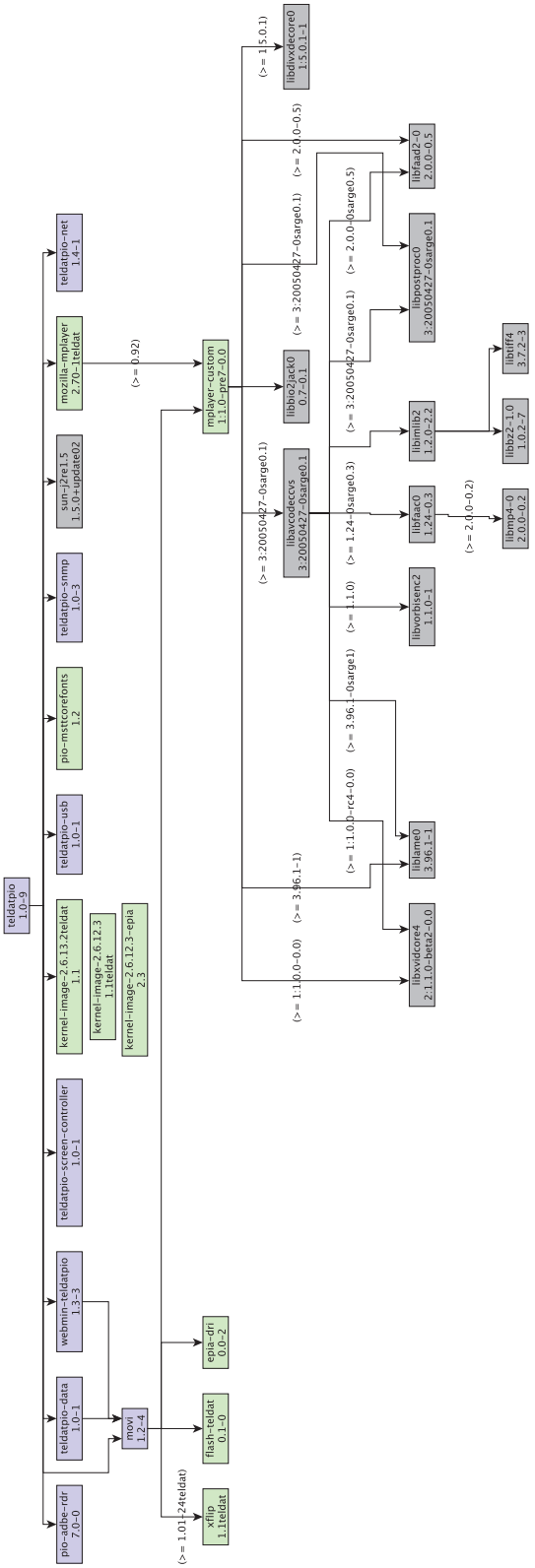

Figura 4.3: Paquetes en el área `teldatpio` y sus interdependencias. Se distinguen tres tipos de paquete: software desarrollado (fondo violeta), software existente pero empaquetado y/o parcheado (en verde) y software existente pero no incluido en Sarge (en gris).

4.3. Soporte al ciclo de vida de la plataforma PIO

El sistema que se ha desarrollado soporta que se puedan desplegar diferentes versiones específicas para los clientes que demanden alguna funcionalidad añadida o especial. Si la diferencia o el cambio es muy grande se puede definir un área para ese cliente en especial partiendo de la que ya existe.

Pero si la adaptación es pequeña o sólo es una modificación a la configuración de la versión estándar, entonces se puede llevar a cabo teniendo una versión específica del paquete virtual `teldatpio` con el cambio introducido en ese paquete o en alguna de sus dependencias.

De esta forma se pueden aprovechar al máximo los componentes comunes y aplicar VCS al código fuente para soportar este modelo.

Capítulo 5

SISTEMA DE DESPLIEGUE DE LA PLATAFORMA

Índice

La plataforma PIO se asienta sobre el sistema operativo Debian GNU/Linux, y para que se pueda instalar sobre cualquier disposición de hardware que cumpla ciertos requisitos, ésta debe tener instalado previamente el sistema operativo, ya que entre las herramientas de Debian GNU/Linux se encuentran las necesarias para gestionar la instalación, desinstalación, configuración y gestión de dependencias de todos los componentes de la plataforma.

El problema al que, por lo tanto, este sistema de despliegue se ha de enfrentar, es a la instalación previa del Sistema operativo además de la posterior instalación de la plataforma.

Una de las características más interesantes del sistema de gestión del software de Debian, es que todo su software puede gestionarse mediante sus herramientas de gestión, desde el núcleo del sistema operativo hasta las propias herramientas. Para esto, en las instalaciones más generalizadas, las que se realizan con un CD o DVD auto arrancable, el despliegue del sistema operativo se realiza una vez inicializado el equipo con un sistema preconfigurado para realizar este despliegue que cuenta con

todas las herramientas necesarias junto con interfaces de usuario para guiar la instalación.

Debido a los requisitos de rapidez y automatización del proceso de instalación, se optó por otra solución basada en la instalación por red. Esto permitía no depender de un medio físico como el CD y de su dispositivo lector. Esto es una gran ventaja porque evita tener que insertar y extraer el medio de instalación por cada equipo, que no solo es costoso en tiempo, sino que está sujeto a fallos (despistes, degradación de los discos ópticos...).

Otra ventaja derivada de la instalación por red radica en la posibilidad de simultanear varias instalaciones sin añadir más medios y utilizando mejor los medios que ya se encuentran disponibles.

5.1. Arquitectura del sistema de despliegue

La arquitectura del sistema de despliegue sigue un modelo cliente-servidor en el que la parte servidora mantiene una serie de servicios para que la parte cliente pueda realizar la instalación automática, siendo el equipo cliente el que inicia el proceso. En primer lugar, el equipo cliente, que dispone de la tecnología **P**reboot e**X**ecution **E**nvironment (PXE), obtiene todo lo necesario para iniciar la instalación, tanto la configuración de red como la ubicación del servidor TFTP del que descargará el software necesario para arrancar el sistema operativo del instalador (*network bootstrap program*).

En este caso, el servidor TFTP ofrecerá el cargador de arranque *PXELINUX*[1] junto con su fichero de configuración que indica qué núcleo, imagen de disco RAM (*initrd*) y sus correspondientes parámetros para iniciar el arranque del sistema operativo.

Este sistema operativo es ejecutado en el equipo cliente solamente durante la instalación y por tanto, no puede depender de sistemas de ficheros almacenados en el disco del propio equipo a instalar. Por este motivo, la raíz del sistema de ficheros del instalador es accedida a través de la red. En concreto, el servidor de despliegue comparte una carpeta mediante el protocolo **N**etwork **F**ile **S**ystem (NFS), (ver figura 5.1).

[1] PXELINUX es una variante del proyecto SYSLINUX que en lugar de cargar desde un medio óptico se carga a través de TFTP.

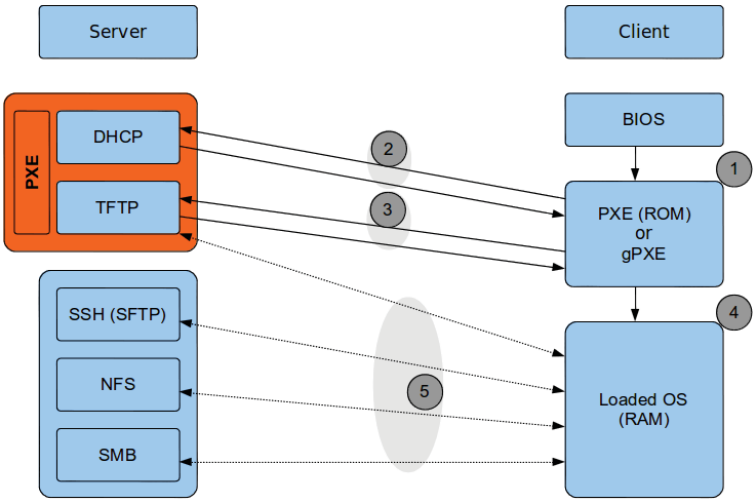

Figura 5.1: Arquitectura del sistema de despliegue

Una vez arrancado el sistema de instalación por red se realiza el particionado del disco duro del equipo cliente y comienza con la instalación de la base del sistema operativo Debian GNU/Linux, mediante la herramienta aptitude y el servidor web donde están ubicados los paquetes, resolviendo automáticamente todas las opciones de configuración necesarias para su correcta instalación.

A continuación, se pasa a instalar el paquete *teldatpio* que es el responsable, por sus dependencias (ver figura 4.3), de instalar todo el software necesario en el equipo cliente y de realizar mediante los scripts de post-instalación la configuración de todos los servicios y programas necesarios para el correcto emplazamiento y funcionamiento del equipo en cualquiera de los destinos donde este servicio ofrecido por Teldat S.A. se desee desplegar.

Finalmente, el equipo al terminar el proceso de instalación se apaga automáticamente para que el operario pueda proceder a su desconexión y embalado. Como se puede observar en todo este proceso, únicamente se ha necesitado la intervención humana para enchufar/desenchufar la corriente eléctrica y conectar/desconectar la red **LAN!** al equipo.

5.2. Entidades

A continuación, se detallarán las entidades que toman parte en el proceso de despliegue de la plataforma.

5.2.1. Servidor de instalaciones: maestro

Este equipo que se denominó maestro durante el desarrollo de este trabajo puede ser un ordenador cualquiera, de hecho, se utilizó durante el proceso de depuración un ordenador similar a los que se utilizaban como clientes. Únicamente ha de tener los servicios de red necesarios para que el equipo cliente pueda comenzar la instalación y un repositorio con toda la paquetería necesaria para completar la instalación de la plataforma.

5.2.2. Switch o conmutador

Para la realización de las instalaciones es necesario un conmutador que permita la interconexión del maestro con varios de los clientes para poder realizar instalaciones en paralelo.

Este dispositivo electrónico que opera en el nivel 2 del modelo OSI (nivel de enlace) mejora sensiblemente el rendimiento del proceso de instalación respecto de un *hub* o concentrador sin afectar al funcionamiento de los protocolos de red que se van a utilizar.

Además, es aconsejable utilizar un switch Gigabit para permitir evitar que el cuello de botella del proceso sea el enlace del servidor con dicho conmutador.

5.2.3. Equipo cliente

Es el ordenador que se desea instalar con la plataforma PIO, a este equipo que después estará en alguna ubicación remota. Este ordenador tiene que tener algunas características básicas para que se pueda realizar la instalación con éxito.

- Procesador Intel x86 o compatible.
- BIOS compatible con el arranque por red
- Tarjeta de red (NIC) con cargador de arranque PXE

5.3. Dinámica del sistema

Para entender el diseño y funcionamiento del sistema de despliegue por red que se desarrolla en esta parte del proyecto, se va a analizar a continuación el arranque de un ordenador personal compatible (PC compatible).

El proceso de arranque comienza al alimentar el equipo presionando el botón de encendido, en ese momento la placa base es alimentada e inicializa el firmware del chipset que intenta iniciar los dispositivos imprescindibles para el ordenador, la CPU y la memoria principal. Si cualquiera de estos dispositivos no se encuentra o no se pueden inicializar la placa base emite un código de error mediante unos tonos mediante un pequeño altavoz, o *speaker*, que integra y queda bloqueada en ese punto.

Si la inicialización es correcta, el procesador comienza a ejecutar, seleccionando uno de los núcleos de procesamiento si el sistema tiene varios, convirtiéndolo en el procesador de arranque, encargado de ejecutar la BIOS y de la inicialización del núcleo del sistema operativo. El resto de núcleos de procesamiento, permanecerán parados hasta que el sistema operativo los active.

En este punto del arranque el procesador se encuentra configurado exactamente igual que los antiguos Intel 8086, por compatibilidad hacia atrás, con el paginamiento de memoria deshabilitado y en modo real de procesamiento. En este estado, el procesador solo puede direccionar un megabyte de memoria y no hay ninguna protección de escritura o gestión de privilegios.

En este punto, la mayor parte de los registros del procesador tienen unos valores predefinidos precargados, que permiten al procesador ejecutar la primera instrucción, almacenada en la dirección `0xFFFFFFF0`, que es el vector de *reset*. Es la placa base la que se encarga de situar en los 16 bytes del vector de reset el salto a la dirección de memoria donde se encuentra el comienzo de la BIOS. Es el chipset el encargado de mantener un mapa de memoria, en el que las direcciones pertenecientes a la BIOS se resuelven con los datos de una memoria flash destinada a tal efecto, ya que la memoria principal aún no contiene información útil.

En esta etapa, el procesador comienza a ejecutar el código de la BIOS, que inicializa algunos de los dispositivos del ordenador e inicia pruebas de autoevaluación de varios de los componentes. A partir de este momento el equipo ejecuta un proceso para poner su tarjeta de red en funcionamiento y utilizando el

protocolo de arranque por red iniciando el proceso de DHCP mediante el cual se le da las instrucciones precisas para que pueda cargar por TFTP la imagen del kernel y el initrd necesario para comenzar la instalación, así como los parámetros que esta imagen pueda necesitar.

Capítulo 6

COMPARATIVA CRÍTICA

Índice

Este capítulo se ha separado en dos secciones para abordar los dos aspectos de carácter técnico que se desarrollan en este trabajo. Hay que apuntar que en la decisión de optar por uno o por otro tuvo más peso el primero, ya que condiciona las posibilidades del segundo. Aun así, se ha querido mostrar que razones sustentan la elección realizada en ambas áreas.

6.1. Sistema gestión de la configuración

La comparación entre el sistema que se va a utilizar y que es libre, Debian GNU/Linux, que está basado en componentes con otros sistemas que se plantearon al comienzo del desarrollo, como Microsoft Windows.

El primero punto a tener en cuenta y más destacado es el coste por licencias, cada uno de los equipos que se instalan y mantienen con el sistema utilizado no tienen que pagar licencia, mientras que si lo basamos en un sistema privado cada uno de los equipos tienen que soportar una licencia del mismo en sus costes.

La adaptabilidad del sistema elegido es muy elevada, se ha podido modificar la presentación de casi todos los elementos, desde el cargador de arranque utilizado,

hasta las pantallas de espera que se presentan al cargar el entorno gráfico. En el sistema privado no se puede modificar estos elementos hasta este punto.

El apartado más importante, es que el sistema está basado en componentes y al estar todos ellos integrados en la misma distribución, garantiza que cualquier programa o librería que se deba añadir de la misma, tendrá total compatibilidad con las librerías que ya estén instaladas, no siendo necesario duplicar las mismas con diferentes versiones, ahorrando costes de mantenimiento.

De igual manera, como la distribución nos provee de todas las herramientas de desarrollo necesarias para añadir o modificar programas y librerías, es muy sencillo crear todo el entorno de desarrollo necesario para crear aquellas aplicaciones o módulos propios que mantengan plena compatibilidad con las librerías ya instaladas.

Para finalizar cabe observar que el basar el sistema en un sistema declarado en su totalidad como libre, nos permite garantizar que no estamos infringiendo ninguna patente o licencia de terceros.

6.2. Sistema de despliegue

El sistema de despliegue que se ha llevado a cabo en este trabajo está basado en FAI, el instalador automático para granjas de ordenadores que utilizando las tecnologías de red y estableciendo una configuración y una pequeña imagen del sistema de instalación (initrd) puede realizar la instalación de varios sistemas simultáneamente basados en componentes. Permite además que el equipo que se ha de instalar sea diverso, siempre que su driver de red este soportado por la imagen que inicia la instalación, y que la instalación pueda ser modificada en función del equipo a instalar o en función del cliente al que vaya destinado.

Otras alternativas que se observaron, como Ghost, WDS, Snap, tenían entre otros inconvenientes. En algunos casos tenían un coste por licencia, que en algún caso era inasumible por ir asociado al número de sistemas instalados, o muy caro, por que constaba de un sólo pago, pero muy elevado.

Por otro lado, estos sistemas alternativos, utilizaban una técnica de despliegue que permitía garantizar la instalación idéntica de todos los equipos, pero a su vez, no permitía la adecuación a que hubiera diferentes modelos con distintas capacidades, ni permitía fácilmente generar diferentes configuraciones para distintos clientes. Por

el contrario obligaba a generar una imagen completa por cada cambio, por pequeño que este fuera.

Por último, la posibilidad de instalar varios simultáneamente se veía mermada por la cantidad de datos que estos sistemas debían transferir, en uno de ellos ni siquiera se podía realizar a través de la red, y era necesario un disco duro USB donde se encontraba la imagen para conectarlo a los dispositivos a instalar, no permitiendo instalar el siguiente sin haber terminado el anterior. Otros permitían la instalación por red, pero al no repartir la carga de la instalación en pequeñas descargas y periodos de configuración, sino todos los datos íntegros tener que transportarlos por la red hacía dos instalaciones simultaneas, ya saturaban las interfaces de red y aumentar más el número de equipos instalados simultáneos incrementaba otros cuellos de botella relacionados con el acceso a disco en el propio servidor de instalaciones.

Capítulo 7

CONCLUSIONES Y LÍNEAS FUTURAS

Índice

Este proyecto culmina los trabajos realizados para el proyecto PIO en la universidad, sentando las bases de su mantenimiento futuro, definiendo las herramientas y procesos que se han de realizar para conseguir alcanzar nuevas funcionalidades, mantener las actuales y realizar el mantenimiento de seguridad de todos los elementos instalados en el sistema.

Este capítulo pretende listar una seria de conclusiones extraídas como fruto de este trabajo y de la participación del autor en el desarrollo de proyecto en general. Además, pretende definir un conjunto de tareas y procesos futuros, que se deben abordar a continuación.

7.1. Conclusiones del Trabajo

La participación en este proyecto de desarrollo y despliegue, desde sus inicios hasta casi llegar al estado de producto final, ha proporcionado al autor y a los compañeros del mismo, la oportunidad de evaluar la evolución de un sistema

comercial, y aprender cuales son los constantes problemas con los que hay que lidiar durante el desarrollo de un proyecto empresarial que está destinado a la venta.

EL desarrollo del sistema se ha realizado sobre la distribución Debian GNU/Linux, lo que ha proporcionado múltiples ventajas para su puesta en marcha, mantenimiento y despliegue sólo durante toda la fase de desarrollo. A continuación, se enumeran algunas de estas ventajas:

Adaptabilidad del sistema: es fácil cambiar el comportamiento del sistema ya que al estar basado en componentes se pueden añadir o quitar fácilmente y configurar así el sistema final.

Disponibilidad del código fuente: disponer del código fuente de otras aplicaciones puede permitir aprender técnicas que se puedan reutilizar en el software que se ha de implementar, así como modificar algunas de ellas para que hagan exactamente aquello que se busca.

Mantenibilidad: todo el sistema de gestión del software utilizado por la distribución puede ser reutilizado, además de poder beneficiarse del trabajo de mantenimiento que realiza la propia distribución para la base del sistema.

La documentación: en general cualquier librería y/o aplicación tiene su documentación asociada al alcance de los usuarios, – y si no, siempre puede disponer del código fuente –.

Por toda esto, el desarrollo de un sistema complejo que opere en las arquitecturas para las que está disponible la distribución, como por ejemplo x86, se ve favorecido por la arquitectura de estos sistemas operativos, cuyas especificaciones son conocidas, y, por otro lado, por la alta configurabilidad de los mismos. Esto hace que a día de hoy gran cantidad de productos del mercado estén utilizando sistemas operativos abiertos.

Además, el presente trabajo ha automatizado y reducido drásticamente el tiempo de despliegue de la plataforma en producción, optimizando el proceso de diversas formas:

- Reduciendo drásticamente la cantidad de software instalado, evitando aplicaciones y librerías innecesarias.
- Utilizando tecnologías de red que permiten la difusión de la información pudiendo desplegar el mismo software en muchos elementos simultáneamente.

7.2. Conclusiones Personales

Desde el punto de vista personal, la realización de este proyecto ha constituido un gran reto y el hecho de tener que desarrollarlo con varios compañeros ha sido muy enriquecedor tanto a nivel personal como a nivel profesional.

7.3. Líneas Futuras

El sistema, tal como se ha comentado anteriormente y como cualquier proyecto de software, estará sometido durante toda la vida útil del producto, especialmente mientras la actividad comercial siga en torno al mismo, deberá someterse a una continua evolución y mantenimiento de seguridad, corrigiendo los problemas que se vayan descubriendo y aplicando los parches necesarios y que vayan surgiendo.

También se está llevando a cabo una nueva línea de investigación y desarrollo, que permite observar y clasificar la cantidad de usuarios y discriminarlos por ciertos parámetros que prestan atención a los contenidos que emite el sistema desplegado en los espacios dispuestos por los clientes. Esto se realiza mediante una cámara que hace capturas cuando detecta algún objeto móvil, y después un software especializado en reconocimiento facial, descarta aquellas en las que la presencia de alguien es baja, enviando las restantes a un servidor, que posteriormente hace un análisis profundo de las imágenes y realiza unos cálculos estadísticos para para calcular el número de personas que ven los contenidos emitidos, e incluso segmentar por la edad aproximada o sexo de los espectadores.

ACRÓNIMOS

B

BIOS **B**asic **I**nput **O**utput **S**ystem

BSD **B**erkeley **S**oftware **D**istribution

C

CD **C**ompact **D**isc

CPU **C**entral **P**rocessor **U**nit

D

DFSG **D**ebian **F**ree **S**oftware **G**uidelines

DHCP **D**ynamic **H**ost **C**onfiguration **P**rotocol

DVD **D**igital **V**ersatile **D**isc

G

GNU **G**nu is **N**ot **U**nix

GPL **G**eneral **P**ublic **L**icense

H

HTML **H**yper**T**ext **M**arkup **L**anguage

HTTP **H**yper**T**ext **T**rasport **P**rotocol

I

IP **I**nternet **P**rotocol

IPSec **I**nternet **P**rotocol **Sec**ure

L

LCD **L**iquid **C**rystal **D**isplay

M

MOVI **MO**tor de **VI**sualización

N

NIC **N**etwork **I**nterface **C**ard

NFS **N**etwork **F**ile **S**ystem

O

OSI **O**pen **S**ystem **I**nterconnection

P

PAT **P**ort **A**ddress **T**ranslation

PIO **P**aneles de **I**nformación en **O**ficinas

PXE **P**reboot e**X**ecution **E**nvironment

R

RAM **R**andom **A**ccess **M**emory

S

SSH **S**ecure **Sh**ell

T

TFTP **T**rivial **F**ile **T**ransfer **P**rotocol

TIDP **T**eldat **I**P **D**iscovery **P**rotocol

TIDPD **T**eldat **I**P **D**iscovery **P**rotocol **D**aemon

U

USB **U**niversal **S**erial **B**us

V

VPN **V**irtual **P**rivate **N**etwork

BIBLIOGRAFÍA

[Feiler, 1991] Feiler, P. (1991). Configuration management models in commercial environments.

[Gkantsidis et al., 2006] Gkantsidis, C., Karagiannis, T., and VojnoviC, M. (2006). Planet scale software updates. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 423–434. ACM New York, NY, USA.

[IPSec, 2011] IPSec (2011). IPSec-Tools Homepage. http://ipsec-tools.sourceforge.net/ comprobado el 15/06/2011.

[Krikhaar and Crnkovic, 2007] Krikhaar, R. and Crnkovic, I. (2007). Software Configuration Management. *Science of Computer Programming*, 65(3):215–221.

[Lange and Glawe, 2009] Lange, T. and Glawe, H. (2009). FAI Guide (Fully Automatic Installation). Disponible en http://www.informatik.uni-koeln.de/fai/fai-guide.html. Actualizado el 19/04/2009. Accedido por última vez el 15/06/2009.

[Manheimer et al., 1990] Manheimer, K., Warsaw, B., Clark, S., and Rowe, W. (1990). The Depot: A Framework for Sharing Software Installation Across Organizational & UNIX Platform Boundaries. *Proceedings of the Fourth USENIX Large Installation Systems Administration (LISA)*.

[Owen and Piper, 2006] Owen, C. and Piper, D. (2006). Using remote installation services for windows to streamline installations in the UTPB computer science research lab. In *Proceedings of the 34th annual ACM SIGUCCS conference on User services*, pages 301–308. ACM New York, NY, USA.

[Symantec, 2008] Symantec (2008). Norton Ghost. Disponible en http://www.symantec.com/norton/ghost. Accedido por última vez el 15/06/2009.

[Teldat S.A., 2005] Teldat S.A. (2005). Descripción del servicio PIO. Disponible en http://www.teldat.es.

[Teldat S.A., 2006] Teldat S.A. (2006). Ficha técnica del servicio PIO. Disponible en http://www.teldat.es/docs/products/ServicioPIO/servicio_pio.PDF.

[Wikimedia Foundation, 2011] Wikimedia Foundation (2011). Wikipedia article on Debian. http://en.wikipedia.org/wiki/Debian accedido por última vez el 15/06/2011.

[Wikipedia, 2009] Wikipedia (2009). Comparison of Linux distributions. Disponible en http://en.wikipedia.org/wiki/Comparison_of_Linux_distributions.

[Wirth, 2008] Wirth, N. (2008). A brief history of software engineering. *IEEE Annals of the History of Computing*, 30:32–39.

Apéndice A

DEBIAN SOCIAL CONTRACT Y Debian Free Software Guidelines (DFSG)

Debian, the producers of the Debian GNU/Linux system, have created the Debian Social Contract. The DFSG part of the contract, initially designed as a set of commitments that we agree to abide by, has been adopted by the free software community as the basis of the Open Source Definition.

"Social Contract" with the Free Software Community

1. **Debian will remain 100% free**

   We provide the guidelines that we use to determine if a work is *free* in the document entitled *The Debian Free Software Guidelines*. We promise that the Debian system and all its components will be free according to these guidelines. We will support people who create or use both free and non-free works on Debian. We will never make the system require the use of a non-free component.

2. **We will give back to the free software community**

   When we write new components of the Debian system, we will license them in a manner consistent with the Debian Free Software Guidelines. We will make the best system we can, so that free works will be widely distributed and used. We

will communicate things such as bug fixes, improvements and user requests to the *upstream* authors of works included in our system.

3. **We will not hide problems**

   We will keep our entire bug report database open for public view at all times. Reports that people file online will promptly become visible to others.

4. **Our priorities are our users and free software**

   We will be guided by the needs of our users and the free software community. We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environments. We will not object to non-free works that are intended to be used on Debian systems, or attempt to charge a fee to people who create or use such works. We will allow others to create distributions containing both the Debian system and other works, without any fee from us. In furtherance of these goals, we will provide an integrated system of high-quality materials with no legal restrictions that would prevent such uses of the system.

5. **Works that do not meet our free software standards**

   We acknowledge that some of our users require the use of works that do not conform to the Debian Free Software Guidelines. We have created *contrib* and *non-free* areas in our archive for these works. The packages in these areas are not part of the Debian system, although they have been configured for use with Debian. We encourage CD manufacturers to read the licenses of the packages in these areas and determine if they can distribute the packages on their CDs. Thus, although non-free works are not a part of Debian, we support their use and provide infrastructure for non-free packages (such as our bug tracking system and mailing lists).

Debian Free Software Guidelines (DFSG)

1. **Free Redistribution**

   The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. **Source Code**

The program must include source code, and must allow distribution in source code as well as compiled form.

3. **Derived Works**

   The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. **Integrity of The Author's Source Code**

   The license may restrict source-code from being distributed in modified form _only_ if the license allows the distribution of *patch files* with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (This is a compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.)

5. **No Discrimination Against Persons or Groups**

   The license must not discriminate against any person or group of persons.

6. **No Discrimination Against Fields of Endeavor**

   The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. **Distribution of License**

   The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. **License Must Not Be Specific to Debian**

   The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

9. **License Must Not Contaminate Other Software**

   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

10. **Example Licenses**

    The *GPL*, *BSD*, and *Artistic* licenses are examples of licenses that we consider *free*.

The concept of stating our *social contract with the free software community* was suggested by Ean Schuessler. This document was drafted by Bruce Perens, refined by the other Debian developers during a month-long e-mail conference in June 1997, and then accepted as the publicly stated policy of the Debian Project.

Bruce Perens later removed the Debian-specific references from the Debian Free Software Guidelines to create *The Open Source Definition*.

Other organizations may derive from and build on this document. Please give credit to the Debian project if you do.