Chromium OS

Evolución y Gestión de la Configuración

Realizado por:
Alejandro Arciniega
Alberto Rincón
Alfonso Yáñez
Miguel Silva
Pedro González

Grupo 1

Poner una página con control de cambios del documento

Contenido

TODO: Falta mapa de herramientas.

Resumen

A continuación se ofrece un análisis exhaustivo del sistema operativo de software libre Chromium OS.

Deja que lo valore el lector

A la hora de analizar el proyecto tendremos en cuenta una serie de apartados que ofrecerán una explicación avanzada de la gestión de la configuración del mismo. Se partirá del estudio de la gestión del código fuente, su obtención, su modificación y como se trabaja con él.

Igualmente se comentan las formas en la que se gestiona la construcción del sistema mediante el uso de Autotest y scripts, así como las técnicas seguidas.

Además, realizamos una propuesta de cómo se deberían gestionar los entregables, debido a la falta de información sobre ello. Del mismo modo destacamos la forma en la que Chromium gestiona sus incidencias, usando unas plantillas predefinidas, y también la depuración mediante el uso del framework Autotest o de scripts automatizados.

También se propone como el sistema podría abarcar la variabilidad, ya que el proyecto Chromium carece de estrategia para ello, usando el Software Product Line(SPL).

Posteriormente se explican las particularidades del proyecto que dejan entrever comportamientos de integración incremental a la vez que parece seguir los procesos de la integración continua. Para cerrar el documento, incluimos un mapa de conexión de las herramientas que se han analizado y una pequeña conclusión por parte del grupo de trabajo.

Ponerlo en el índice

Introducción

Este documento surge a partir de una propuesta de los profesores de la asignatura de Evolución y Gestión de la Configuración como forma de complementar la docencia de la asignatura.

No aporta

En el presente documento se estudiará la configuración del proyecto de software libre Chromium OS en profundidad, atendiendo a la documentación disponible en la plataforma de desarrollo del proyecto Chromium.

Esto debería ir en el resumen

El sistema operativo Chromium OS surge a partir del proyecto de Google Chrome OS, siendo el primero la versión de código de libre distribución del segundo. El lenguaje de programación utilizado es C.

Para el estudio del sistema operativo ha sido necesaria una extracción exhaustiva de información de los documentos para desarrolladores que el proyecto Chromium proporciona en la página oficial del proyecto. A través del documento proporcionado por la asignatura así como los diferentes temas abarcados durante el curso hemos establecido una división de apartados para establecer unas guías de estudio sobre el proyecto Chromium OS.

De esta manera, el presente documento ha sido dividido en ocho apartados que resumen el estudio que se ha llevado a cabo.

El primero de ellos abarca todo lo relacionado con la gestión del código fuente, exponiendo las herramientas que Chromium OS aconseja utilizar para ello, así como el procedimiento a seguir para realizar cambios en el mencionado código y nuestras valoraciones al respecto.

Da un avance de cuáles son esas herramientas

El siguiente apartado explica la gestión de la construcción, en el cual explicamos la técnica de integración incremental usada por el equipo de desarrollo de Chromium OS para construir el sistema y comentamos las variedades de builds que nos encontramos durante el estudio.

Cuando sea "inglés" en cursiva

Tras ello, el siguiente apartado que se puede encontrar define los tipos de entregables que ofrece Chromium OS y aquellos que propondríamos, ambos bajo nuestra perspectiva.

Lo siguiente que tratamos es la gestión del despliegue que identificamos en el proyecto Chromium OS, la cual se divide en tres procesos fundamentales: petición de cambio y asignación de tareas, despliegue e implementación de cambios, tests de aceptación. Dentro del mismo apartado se explica la estrategia que se sigue para gestionar los releases del proyecto.

Esto no es despliegue es incidencias/cambio

El próximo apartado que aparece en el documento explica el método mediante el cual se gestionan las incidencias que se deseen reportar, el cual sigue el procedimiento genérico en estos casos, que consta de los pasos: creación, aceptación, asignación y resolución, así como también se explican los estados en los que una incidencia puede estar, entre otras cosas. En este mismo apartado se explica el procedimiento a seguir para la depuración del código de Chromium OS, la cual se realiza mediante el framework Autotest para casos de dominio grande o mediante el uso de scripts automatizados proporcionados por el equipo de Chromium para casos de dominio pequeño. Del mismo modo, en ese apartado explicamos cómo afectaría un desarrollo guiado por pruebas a Chromium OS.

Justificar TODO EL TEXTO

Lo siguiente que el documento abarca es la gestión de la variabilidad, la cual no es contemplada en la documentación oficial del proyecto, por lo que valoramos la situación y aconsejamos el uso del Software Product Line para la producción y explicamos como identificar variabilidad en el sistema.

Para continuar, el siguiente apartado explica la existencia de mecanismos de integración continua y despliegue continuo dentro del proyecto, que se dejan ver al comprobarse en la gestión de la construcción el uso de la herramienta BuildBot, la cual gestiona este tipo de mecanismos.

Además, se ofrece un mapa de herramientas que esquematiza la conexión entre las distintas herramientas analizadas durante el estudio de Chromium OS. Para terminar, el documento se cierra con las conclusiones que el grupo de trabajo ha sacado durante la realización del documento.

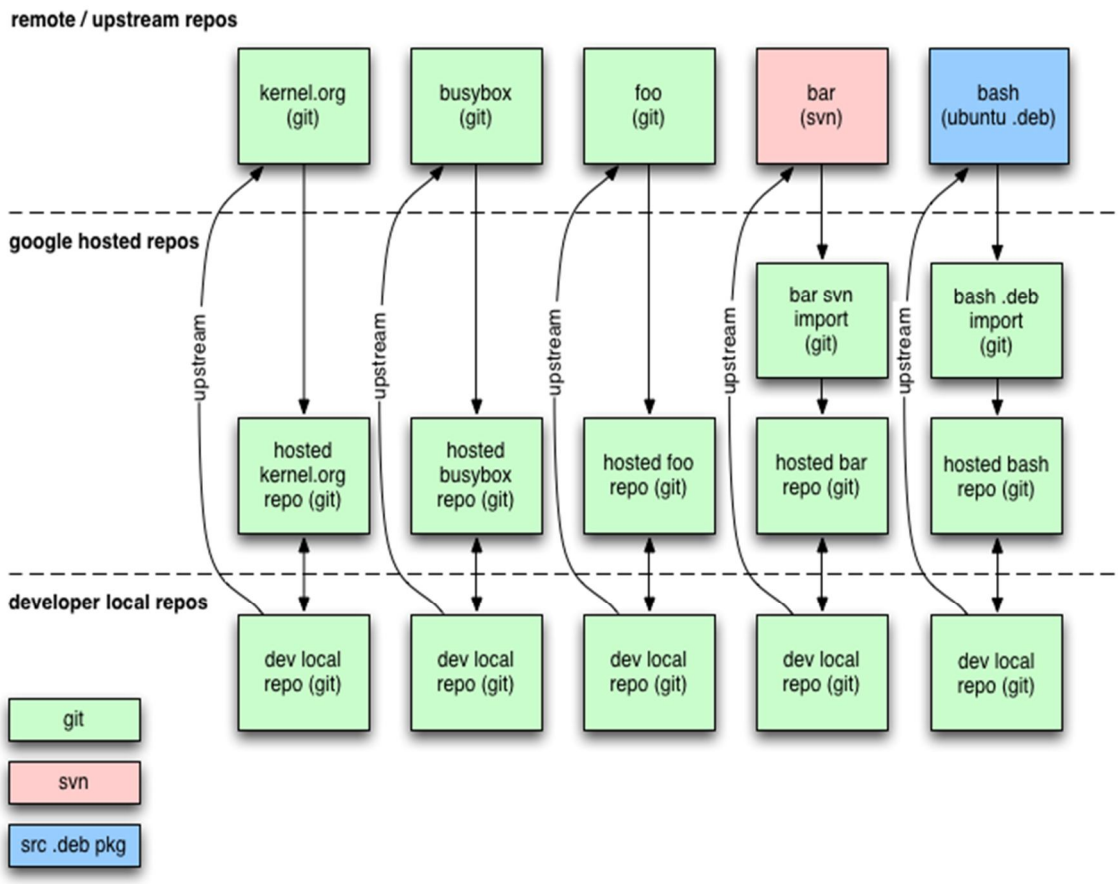
Gestión del Código Fuente

La gestión del código fuente en el proyecto Chromium OS se realiza mediante repositorios Git, ya que es la herramienta más utilizada y, según exponen ellos mismos, la que mejor se adapta a las necesidades del proyecto.

El sistema de repositorios del proyecto consiste en un repositorio Git para cada componente que se esta modificando. Cada vez que se necesite modificar un nuevo componente se creará un nuevo repositorio. Para trabajar se utilizarán copias de estos repositorios superiores, creando una nueva rama de la última versión estable disponible en los repositorios superiores.

TODO: esta parte es MUY interesante pero falta explicarla mejor y poner ejemplos concretos (aquí podéis poner un ejercicio para hacer algo y explicar cómo se solucionaría)

La siguiente imagen explica gráficamente la distribución de los repositorios



1

[1] https://sites.google.com/a/chromium.org/dev/chromium-os/chromiumos-design-docs/source-code-management/repo-arch.png?attredirects=0

Para la gestión del código se definen dos roles concretos: colaboradores y revisores. Se consideran colaboradores aquellas personas que realizan modificaciones del código fuente, aportando posibles mejoras o características adicionales, así como reporte de errores. Por otra parte se encuentran los revisores, desarrolladores cualificados, designados para verificar los cambios propuestos por los colaboradores.

TODO: comentar de cuántas personas hablamos en los distintos roles o por lo menos para algún componente.

La decisión de dividir al personal en dos únicos roles es a primera vista simple, pero en este caso se debe considerar la gran cantidad de desarrolladores que pueden intervenir, lo cual provoca que si asignamos un mayor número de roles, probablemente la gestión del código se vea afectada negativamente en el aspecto temporal, complicándose más cada acción a desarrollar por los niveles inferiores de la jerarquía. Esto no debe ser así en todos los casos, pero si es lo más probable que puede ocurrir. Es por ello que consideramos acertada la distribución de roles, en lo que a la gestión del código se refiere.

¿qué es?

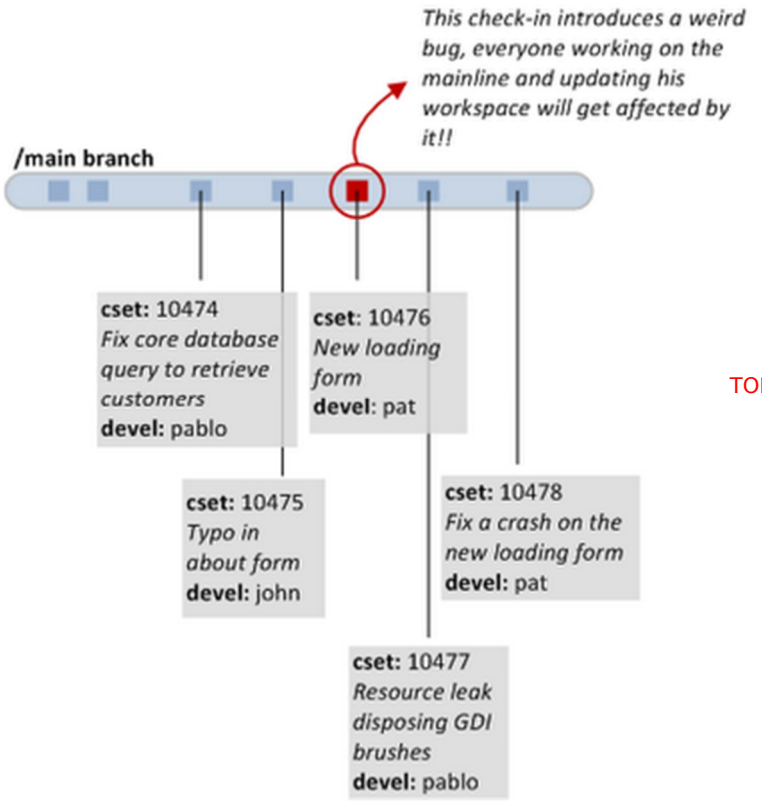
Para comenzar a involucrarse en el proyecto se pide a los colaboradores que trabajen con un repositorio propio en Gerrit, herramienta que se caracteriza por funcionar manteniendo intacta la build a pesar del número de desarrolladores implicados. Estos repositorios se crean mediante una copia del repositorio en el que se encuentra el paquete sobre el que se va a trabajar en la última versión de "Check-point", es decir, el último release.

Explicar, no se entiende

TODO: Todo esto hay que explicarlo mejor!

En la siguiente imagen se explica gráficamente el porqué de usar Gerrit

TODO: Explica la figura

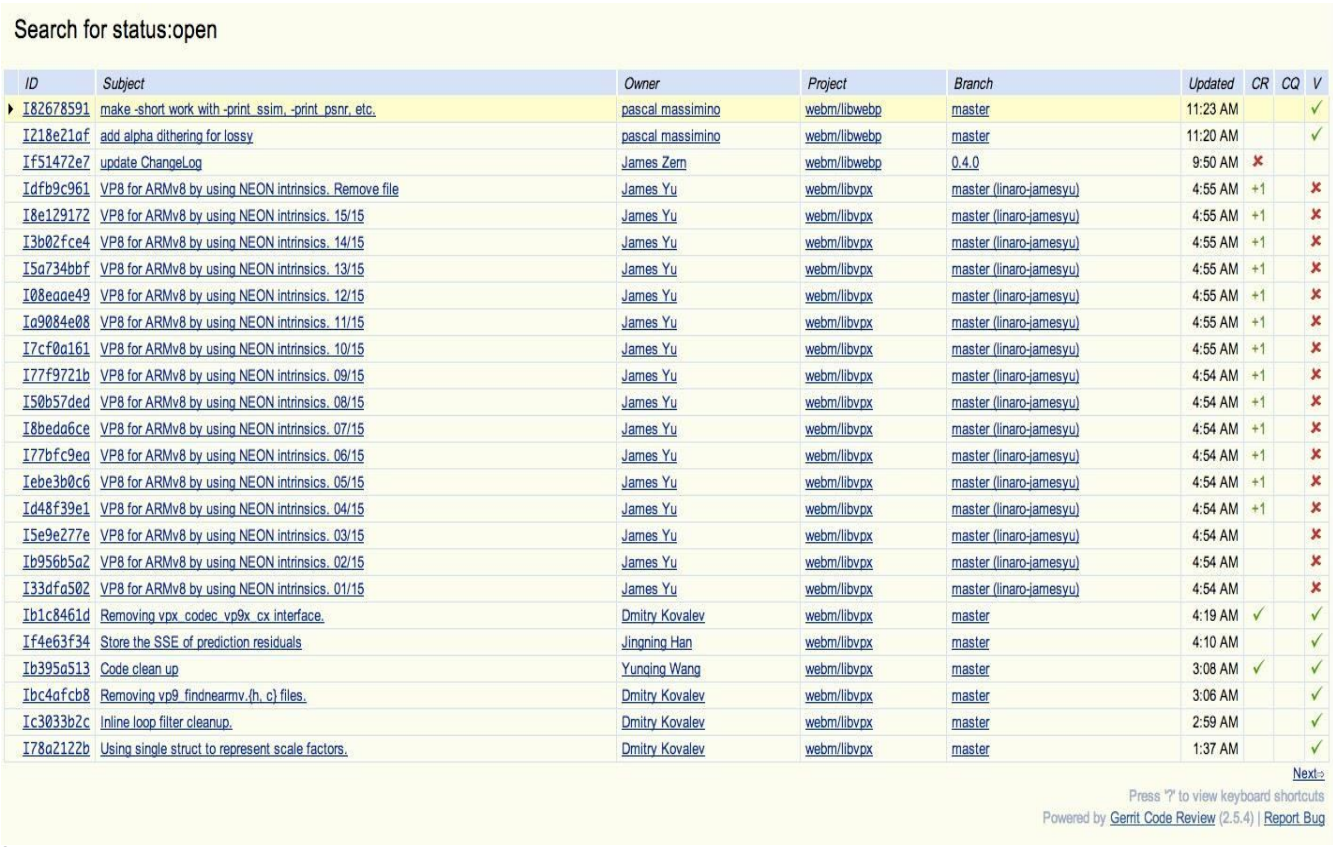


TODO: todo esto hay que explicarlo mejor

http://brigomp.blogspot.com.es/2011/09/gerrit-un-sistema-de-revision-de-codigo.html

Simplemente lo que se quiere explicar es que un error en el branch principal “infectará” los updates de los demás desarrolladores que obtengan el código.

Un ejemplo del uso de la herramienta gerrit con Chromium se puede apreciar en la siguiente captura:



Search for status:open

ID	Subject	Owner	Project	Branch	Updated	CR	CQ	V
I82678591	make -short work with -print_ssim, -print_psnr, etc.	pascal massimino	webm/libwebp	master	11:23 AM			✓
I218e21af	add alpha dithering for lossy	pascal massimino	webm/libwebp	master	11:20 AM			✓
If51472e7	update ChangeLog	James Zern	webm/libwebp	0.4.0	9:50 AM	✘		
Idfb9c961	VP8 for ARMv8 by using NEON intrinsics. Remove file	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I8e129172	VP8 for ARMv8 by using NEON intrinsics. 15/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I3b02fce4	VP8 for ARMv8 by using NEON intrinsics. 14/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I5a734bbf	VP8 for ARMv8 by using NEON intrinsics. 13/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I08eaae49	VP8 for ARMv8 by using NEON intrinsics. 12/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
Ia9084e08	VP8 for ARMv8 by using NEON intrinsics. 11/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I7cf0a161	VP8 for ARMv8 by using NEON intrinsics. 10/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:55 AM	+1		✘
I77f9721b	VP8 for ARMv8 by using NEON intrinsics. 09/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
I50b57ded	VP8 for ARMv8 by using NEON intrinsics. 08/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
I8beda6ce	VP8 for ARMv8 by using NEON intrinsics. 07/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
I77bfc9ea	VP8 for ARMv8 by using NEON intrinsics. 06/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
Iebe3b0c6	VP8 for ARMv8 by using NEON intrinsics. 05/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
Id48f39e1	VP8 for ARMv8 by using NEON intrinsics. 04/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM	+1		✘
I5e9e277e	VP8 for ARMv8 by using NEON intrinsics. 03/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM			✘
Ib956b5a2	VP8 for ARMv8 by using NEON intrinsics. 02/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM			✘
I33dfa502	VP8 for ARMv8 by using NEON intrinsics. 01/15	James Yu	webm/libvpx	master (linaro-jamesyu)	4:54 AM			✘
Ib1c8461d	Removing vpx_codec_vp9x_cx interface.	Dmitry Kovalev	webm/libvpx	master	4:19 AM	✓		✓
If4e63f34	Store the SSE of prediction residuals	Jingning Han	webm/libvpx	master	4:10 AM			✓
Ib395a513	Code clean up	Yunqing Wang	webm/libvpx	master	3:08 AM	✓		✓
Ibc4afcb8	Removing vp9_findnearmv.{h, c} files.	Dmitry Kovalev	webm/libvpx	master	3:06 AM			✓
Ic3033b2c	Inline loop filter cleanup.	Dmitry Kovalev	webm/libvpx	master	2:59 AM			✓
I78a2122b	Using single struct to represent scale factors.	Dmitry Kovalev	webm/libvpx	master	1:37 AM			✓

Next>

Press '?' to view keyboard shortcuts

Powered by Gerrit Code Review (2.5.4) | Report Bug

2

Como podemos comprobar, gerrit proporciona una interfaz simple e intuitiva para gestionar las modificaciones de código realizadas, especificando en que branch, el estado de verificación (V), si el commit está en espera (CQ), etc.

TODO: explicar esto mejor. Esto es un punto crítico

La elección de esta herramienta está ligada a su anteriormente mencionada capacidad de gestionar numerosos desarrolladores sin que la build se vea afectada, y probablemente a su uso libre y gratuito, así como su facilidad de uso. Podría haberse escogido otra herramienta tipo RedMine o trac, pero estas dos extienden su funcionalidad más allá de lo necesario para Chromium OS sin centrarse en lo que realmente necesita Chromium OS en la gestión del repositorio Git.

[2] https://gerrit.chromium.org/gerrit/#/q/status:open,n,z

TODO: crear una subsección o subsecciones para dividir este aparatado

Al iniciar la modificación de un proyecto este debe marcarse como activo. El colaborador puede ver el código fuente, decidir en qué parte trabajara y subir sus cambios, que serán controlados por unos revisores. Cuando se termine de hacer cambios en local, debe hacerse "commit" sobre el branch local mediante Git. Una vez terminado el "commit" local, se deben subir los cambios con un nuevo identificador para que los revisores de código procedan a verificarlo.

Aclarar branch y clone en GIT, parece que hay confusión.

Este equipo de revisores de código se compone de varios especialistas en las diferentes partes del mismo. El hecho de que cada revisor sea experto en parte del código es una buena práctica que va ligada a proyectos de gran envergadura, como pueda ser un sistema operativo. Esta práctica permite la especialización del trabajo.

Es por ello, y por la fiabilidad que ofrece tener un grupo de especialistas con experiencia y conocimientos avanzados, que consideramos el seguimiento y verificación de las modificaciones por parte de revisores apropiado para este proyecto.

Se podría pensar que un gran número de modificaciones podrían sobrecargar a los revisores, pero este problema es siempre solucionable con la contratación de nuevos especialistas si se quiere mantener el coste temporal, o simplemente dosificando el trabajo de estos si el tiempo no es un problema.

La opción alternativa de confiar en las modificaciones de los desarrolladores no sería factible en este tipo de proyectos, pues como se mostró anteriormente, un error en el código se propagaría entre los nuevos updates de los desarrolladores.

Para realizar un "merge" en la rama común se sigue un procedimiento conocido como *"Keep the tree green"*, cuyo objetivo es mantener operativa la funcionalidad básica en las imágenes creadas desde el árbol, de manera que no se desestabilice la construcción.

TODO: explicar MUCHO más todo esto que parece interesante pero está muy poco

Consideramos este método como adecuado pues asegura una base funcional estable en todo momento y, a su vez, reduce el número de merges a realizar. También se establece el uso del comando de git "rebase" para facilitar la sincronización de los repositorios.

¿qué es un patch para vosotros o el proyecto?

Habéis encontrado alguna pega? veis alguna

Si lo que deseamos es realizar un patch en el código de Chromium OS, se debe proceder de la misma forma que si lo que deseamos es añadir alguna funcionalidad. Realizamos los cambios oportunos en local, hacemos commit y posteriormente push para que los revisores aprueben o no nuestro patch y este se aplique.

Cabe destacar, que las directrices de la guía para desarrolladores de Chromium OS aconsejan eliminar la rama local una vez se han terminado de hacer las modificaciones.

Cuándo habláis de rama(branch) queréis decir repositorio local o clone?

Por último, si se ha alcanzado un Check-point, se debe crear una nueva rama para realizar una versión.

TODO explicar est párrafo, ¿qué es un check-point?

Pese a todo, este conjunto de decisiones de gestión del código están en continuo cambio debido a los posibles problemas que se descubran de forma posterior y a las mejoras que se puedan hacer en él.

TODO: se podrían proponer muchos más ejercicios

Ejercicio propuesto (este ejercicio hay que explicarlo MUCHO MAS)

Se pide realizar todos los pasos necesarios para preparar el entorno de trabajo para obtener el código de la rama principal.

Nota: se considera instalado el sistema operativo Ubuntu 12.04 sin actualizaciones y cumpliendo las recomendaciones para la instalación aportadas.

Resolución:

- Abrir un nuevo terminal.
- Escribir sucesivamente los siguientes comandos en el terminal:
  - sudo apt-get install git-core
  - sudo apt-get install gitk
  - sudo apt-get install git-gui
  - sudo apt-get install subversion
  - sudo apt-get install curl
  - git clone ttps://chromium.googlesource.com/chromium/tools/depot_tools.git
  - export PATH="$PATH":`pwd`/depot_tools
  - cd /tmp
  - cat > ./sudo_editor <<EOF
  - #!/bin/sh
  - echo Defaults \!tty_tickets > \$1
  - echo Defaults timestamp_timeout=180 >> \$1
  - EOF
  - chmod +x ./sudo_editor
  - sudo EDITOR=./sudo_editor visudo -f /etc/sudoers.d/relax_requirements
  - git config --global user.email “you@example.com”
  - git config --global user.name "Your Name"

Poner comentarios de los comandos (e.g. para qué curl?) qué se hace en cada paso?

Dónde “you@example.com” es el email que se quiere usar para Git y “Your name” el nombre que quieras usar.

  - cd ${HOME}/chromiumos
  - repo init -u https://chromium.googlesource.com/chromiumos/manifest.git --repo-url https://chromium.googlesource.com/external/repo.git https://chromium.googlesource.com/external/repo.git

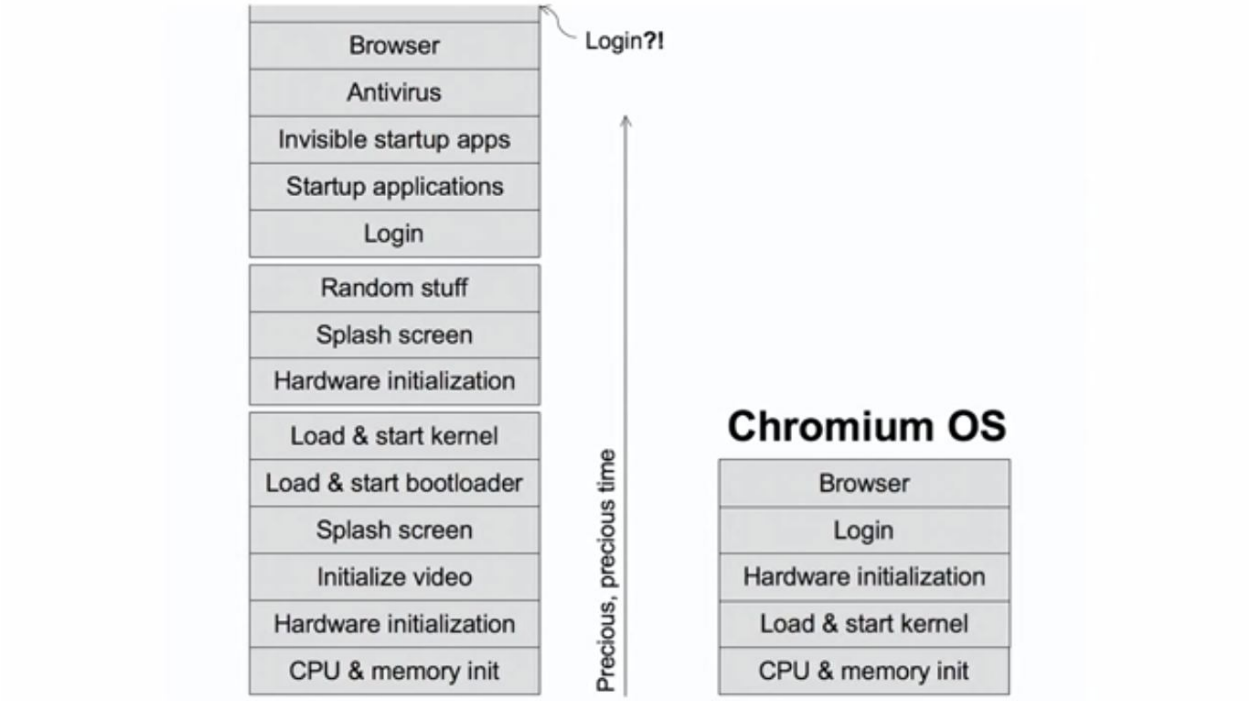
En este paso ya estamos preparados para obtener el código. Si deseamos obtenerlo solo debemos sincronizar con la orden repo:

  - repo sync

Echo en falta en la gestión del código algo que hable del estilo de código que seguro algo tienen definido en el proyecto y si no, se podría definir.

Gestión de la construcción

El origen del sistema operativo es reconocido como una reutilización del Kernel de Linux, al cual se le retiró la funcionalidad y los módulos que no resultaban interesantes para el desarrollo de Chromium OS. TODO: poner una fuente que soporte esta afirmación.



[3]

Según lo visto en teoría, este proyecto parece seguir un estilo de integración Incremental, porque parte de un esqueleto funcional al cual se le van añadiendo funcionalidades que se van diseñando, codificando, probando y depurando antes de añadirlo al código fuente.

Añadir referencia

Esta técnica es un poco costosa en tiempo, ya que es necesario testear/depurar antes de añadir una nueva funcionalidad. Para reducir este tiempo se hace uso de la herramienta BuildBot, que es una herramienta que automatiza la construcción. También se intenta mantener el tiempo entre commits lo más corto posible para obtener la información de los test cuanto antes.

(cuánto es "más corto posible"? días? horas? semanas?)

[3] http://www.youtube.com/watch?v=mTFfl7AjNfI

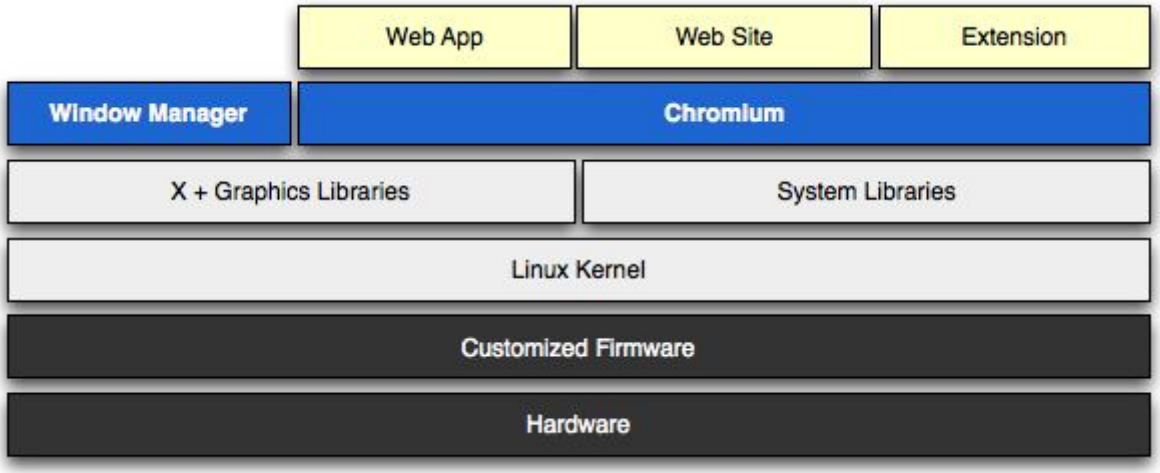
Mejorar

Concretamente se trata de una integración orientada a funcionalidades, en la cual los módulos se integran en grupos que constituyen una funcionalidad identificada. Cabe recalcar que esta técnica es diferente a las conocidas (top-down, bottom-up y sandwich integration) , pero la consideramos correcta, ya que en un proyecto del tamaño de Chromium OS es la técnica óptima , ya que permite minimizar el uso de drivers y stubs, y cada construcción posee una nueva funcionalidad definida.

En lo referente a la arquitectura ChromiumOS se diferencia en tres niveles: el administrador de ventanas y el navegador, el nivel de Software a nivel de sistema y servicios de usuario, y el nivel de Firmware.

¿qué tiene que ver la arquitectura del sistema con la gestión de la construcción?

En la siguiente imagen podemos observar las capas definidas.



[4]

¿qué es esto? explicar mucho mejor

En lo referente a las builds, la documentación sólo hace referencia a las privadas, indicando que para mantener una coherencia en la construcción se debe trabajar con una chroot, un método usado en Unix para comprobaciones. La frecuencia de realización dependerá del desarrollador que realiza la modificación del código.

Respecto a los otros dos tipos de builds, integración y liberación, no nos ha sido posible encontrar información sobre ellos, ya que son procesos internos de Google los cuales son privados y no se divulga información, lo que nos lleva a la conclusión de dar por hecho su uso dentro del sistema Chromium OS. De la misma forma, no es posible obtener información sobre la frecuencia en que se realizan.

Posteriormente se profundizará sobre la posibilidad o existencia Integración Continua y Despliegue continuo en el apartado correspondiente.

Mal redactado

[4] https://sites.google.com/a/chromium.org/dev/chromium-os/chromiumos-design-docs/software-architecture/overviewpng

Ejercicio propuesto: Este ejercicio se puede explicar mucho más.
Realizar una build personal de ChromiumOS, suponiendo que ya tenemos el código fuente en nuestro entorno de trabajo.

Resolución:
Asumiendo que nos encontramos con un terminal en el directorio del código de Chromium OS, procedemos a realizar estos comandos:

- ./chromite/bin/cros_sdk

Ejecutamos esta instrucción dos veces, la primera para crear el chroot y la segunda para acceder a él.

- export BOARD=amd64-generic
- ./setup_board --board=${BOARD}

Con esto quedaría montada la imagen, sin configurar la contraseña de acceso, etc.

Gestión de los entregables

Después de comprobar la documentación proporcionada por Chromium OS, llegamos a la conclusión de que no es posible obtener información de cómo se gestionan sus entregables. Sin embargo, si apreciamos, bajo nuestra consideración, que las distintas versiones que se ofrecen del sistema para descargar son entregables del proyecto, cuya gestión desconocemos, salvo el hecho de que se proporciona el entregable en formato iso para descargar desde varios servidores. Por tanto, vamos a proponer una posible gestión de los entregables.

Partimos de que un entregable es cualquier elemento tangible que se obtiene como resultado de un proyecto que debe ser entregado a un cliente. En el caso de Chromium OS los clientes son los usuarios que se descargan el sistema operativo de forma gratuita. Se podrían considerar otros casos de clientes que soliciten un producto específico. Por tanto vamos a diferenciar estos dos tipos de entregables. Estos entregables se repiten en el tiempo a medida que se crean nuevas versiones de los mismos.

¿a qué se refiera esto?

Por último, tendremos en cuenta un tercer tipo de entregable: documentos de diseño. Este último tipo de entregable, será el más corriente durante la evolución del sistema operativo, ya que las modificaciones más comunes y tangibles serán las correspondientes al diseño de la interfaz de usuario, para la cual se ofrecen unas directrices a seguir en la documentación del proyecto Chromium OS.

Los entregables que se ofrecen a los usuarios que descargan la versión gratuita de Chromium OS serán aquellas versiones del sistema operativo que se ofrecen según el dispositivo en el que se instalará el sistema. Actualmente Chromium OS solo está disponible para ordenadores de mesa y portátiles, pero ya se está trabajando en una versión para tabletas, lo que supondría un nuevo entregable. Los clientes podrán obtener los entregables desde las páginas de descarga proporcionadas por miembros encargados del proyecto.

No es posible por parte de los usuarios de obtener un seguimiento informativo de la evolución de los entregables, salvo lo poco que se comente o adelante en blogs relacionados. En este caso no se distinguen roles, solo existe el cliente, que engloba a cualquier usuario que descargue el sistema. Con respecto a los entregables relacionados con peticiones específicas de clientes, la forma de entrega será la acordada entre las partes.

¿a qué os referís con "seguimiento"?

Refiriéndonos al seguimiento, sería necesario distinguir entre un cliente con conocimientos informáticos, y otro que no los posea. En cualquiera de los dos casos, la entrega de documentación de forma periódica sería suficiente, pero el cliente con conocimientos podría requerir un rol específico dentro de una herramienta de gestión de proyectos (por ejemplo RedMine), para comprobar de primera mano la evolución de su producto.

Pobremente explicado en general

Por último, aquellos entregables en forma de documentación relacionada con la evolución del diseño pueden aparecer en cualquiera de los dos anteriores casos, como parte de ellos si consideramos cierto entregable un conjunto de pequeños entregables (como suele ocurrir generalmente). De la misma forma que ocurre anteriormente, la entrega de esta documentación debe convenirse con el cliente. De no ser posible, como es el caso de producto Chromium OS libre y gratuito, se proporciona estos entregables a través de la web oficial de desarrolladores.

Esto es muy pobre, no aclara casi nada

Tras analizar las situaciones anteriores, propondríamos como plataforma de gestión de los entregables la herramienta RedMine, ya que ofrece un seguimiento del producto en caso de que el cliente lo solicite.

La entrega de los productos debería hacerse en persona dentro de algún soporte físico, para el caso del cliente personalizado, y tal y como se viene haciendo para el proyecto de software libre de Chromium OS, ofrecer enlaces de descarga del producto en formato iso así como el código a través de sus repositorios, o la documentación a través de la web oficial.

TODOs: este apartado tiene muchas posibles mejoras. Se podría explicar cómo se garantiza la integridad de los entregables, cómo se numeran o versionan los entregables y muchas más cosas. Mirad los libros de la bibliografía y referiros a ellos para ver qué cosas podéis poner en este apartado
Este apartado, en general, está bastante pobre.

Interesante pero, ¿por qué no se explica nada de esto dentro del apartado?

Ejercicio propuesto:

Crear un USB booteable mediante una iso del sistema operativo Chromium OS

Resolución:

Desde el terminal de Linux, introducir los siguientes comandos:

1. cd "carpeta raíz de la iso"
2. dd if=nombreiso.iso of=/dev/sdb

Gestión del despliegue

Creo que no tenéis claro qué es el despliegue tenéis que consultar la bibliografía y leer algo más

Aunque no se especifique de forma oficial, a partir de la gestión del proyecto que hemos analizado durante el desarrollo de este documento, suponemos que el equipo de desarrollo de Chromium OS sigue un mecanismo de despliegue dividido en tres procesos fundamentales, que identificamos como: petición de cambio y asignación de tareas, despliegue e implementación de cambios y tests de aceptación.

Este conjunto de procesos resume un procedimiento en el cual existe una motivación para realizar un cambio en el software actual lo que conlleva la modificación del código fuente por parte de algún desarrollador/equipo cualificado.

Una vez se han implementado los cambios solicitados, es necesario comprobar que éstos mantienen el producto dentro de los niveles de calidad exigidos.

Finalmente se libera la nueva modificación del software, ofreciéndose una nueva versión de la forma acordada con el cliente.

Esto parece más algo de gestión del cambio

En la práctica, el mecanismo de despliegue sigue muchos más subprocesos, no por ello de menor importancia, los cuales suponemos que son implementados por parte del equipo de Chromium OS. Una guía sobre cuáles podrían ser esos subprocesos está disponible en MADEJA:

http://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/12

Está bien basarse en MADEJA pero hay que dejar muy claro cómo se hace

En el caso de Chromium OS, los usuarios del sistema son los clientes, y puesto que no tienen autoridad para exigir cambios en el sistema operativo, estos les son ofrecidos de manera periódica basándose en los reportes creados por los propios usuarios o bien por la evolución del mercado.

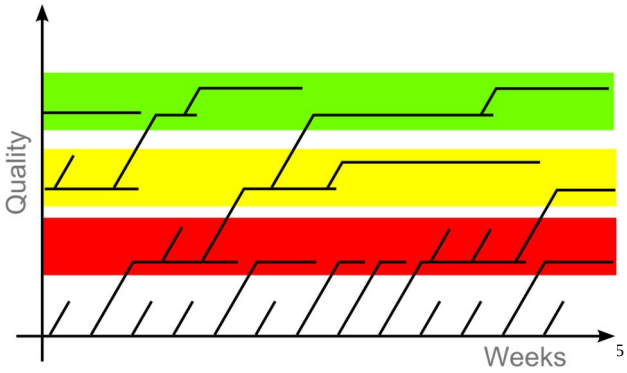
Debemos diferenciar la aportación de los desarrolladores que de forma desinteresada ofrecen cambios en el software, de aquellos realizados por el equipo principal de Chromium, que son los que analizamos en este apartado.

Parece que tenéis confusión entre lo que es gestión de entregables y de despliegue

El punto más crítico durante la generación de una nueva entrega se sitúa en la gestión de las dependencias. Durante la modificación del código, se debe mantener la relación entre los paquetes que componen el producto final, de manera que no se produzcan inconsistencias a posteriori.

Chromium OS enfoca las releases basándose en la idea de lanzar versiones operativas con nueva funcionalidad y mejoras incorporadas cuanto antes. Para realizar este ciclo de versiones de vida corta se utiliza un mecanismo de manera que se inicia el trabajo desde el canal de desarrollador creando una rama del árbol principal sobre la que se trabaja. Si la rama en desarrollo no se ve abortada, pasa al canal de beta y si continúa siendo estable pasa al de lanzamiento.

Esta idea se expresa en la siguiente imagen, donde las líneas representan la evolución de las ramas en el tiempo:



Nuevamente, esto es la entrega, no el despliegue
TODO importante: tenéis que entender bien la diferencia entre gestión de entregas y gestión del despliegue

El tiempo de vida de estas ramas se intenta reducir al mínimo siendo la media de unos 3 meses, por lo que no es necesario atender al código que de mayor antigüedad, lo que ayuda en la refactorización del código. La realización de este pipelining les permite tener mucho más feedback.

Esto son afirmaciones que no está muy sustentadas... no se saca mucho en claro

El ciclo de versiones de vida corta conlleva que los cambios entre ellas sean muy graduales. Esto hace que los usuarios se vayan adaptando gradualmente a los cambios, sin imponer nuevas funcionalidades de forma inmediata y, a su vez, reduce la aversión que los usuarios pueden desarrollar por el cambio.

¿no es esto más bien gestión del código?

Identificamos que los desplegables están clasificados por su versión, pero no encontramos información alguna sobre el ciclo de vida de las versiones del sistema, por lo que proponemos un ciclo de vida con los siguientes estados: desarrollo, pruebas, producción, listo para su despliegue. Este conjunto de estados diferencia de manera clara y concisa el estado de producto, y le somete a una serie de procesos en cada uno de ellos que propician una evolución.

Explicar estos estados

¿O los entregables?

Chromium OS no proporciona información relacionada con la documentación que se genera durante el proceso de despliegue, bien por considerarse información confidencial o simplemente por no disponer de esta funcionalidad. Como desarrolladores opinamos que sería muy útil tener una documentación para comprobar los resultados del despliegue.

Esta sección tiene importantes problemas de coherencia entre despliegue y entrega pues se confunde continuamente una cosa con la otra. Debéis leer la bibligrafía sobre esto y tenerlo mucho más claro.

[5] https://a77db9aa-a-7b23c8ea-s-sites.googlegroups.com/a/chromium.org/dev/developers/tech-talk-videos/release-process/release%20process%20version%202.png

Ejercicio Propuesto

Arrancar Chromium OS a partir de la imagen en un dispositivo de memoria flash USB.

Resolución

1. Encender el sistema y acceder a la BIOS durante el proceso de arranque.
2. Configurar la BIOS para que se intente el arranque a partir de un dispositivo USB en primer lugar.
3. Apagar el sistema.
4. Insertar el dispositivo USB con la imagen de Chromium OS en un puerto USB del sistema.
5. Encender el sistema y esperar a que arranque.

Explicar mejor esto, ¿cómo se genera la imagen (enlazar con un ejemplo de entregables)? ¿qué problemas puede haber en el arranque?
Aquí sí habláis de "despliegue" aunque para hablar de un despliegue más complejo sería interesante algo más que una instalación en un solo puesto.

Gestión de incidencias y depuración

Explicarlo mejor por qué esto es así.

Gracias a la metodología *Keeping the tree green* es difícil encontrar código que cause conflictos a la rama principal ya que cada usuario debe asegurarse en su branch local de que esto no ocurra. Si ocurriera el usuario debe poner una flag indicando que se trata de código conflictivo. Chromium OS gestiona las incidencias con una metodología de auto feedback por parte de los usuarios que prueban Chromium OS.

explicar más a qué se refiere esto
poner las cosas en inglés en cursivas

Estas incidencias siguen el método general de procesado de incidencias, donde en cada paso se ajusta el proceso genérico a una metodología específica:

¿cómo es la plantilla

- Creación: Se realiza mediante un informe a partir de una plantilla predefinida, en función del tipo de reporte a realizar, facilitando los datos para poder analizar la incidencia.
- Aceptación: Un revisor comprueba si la incidencia es relevante y la confirma.
- Asignación: La incidencia considerada relevante se asigna a un desarrollador para solucionarla.
- Resolución: Se utiliza un sistema similar a *Mantis bug tracker*, desarrollado por Google específicamente para este proyecto y similares.

poner ejemplos

¿cómo se llama el sistema? ¿en qué se diferencia? ¿dónde se puede ver?

En Chromium OS, las incidencias se van marcando con varios estados en su ciclo de vida, según esté abierta, es decir, en estudio, o cerrada.

Hacéis varias veces esta referencia a lo "visto en teoría" pero este documento debería ser un proyecto profesional y en esos proyectos no se habla de "lo visto en teoría"

Con respecto al sistema de gestión de incidencias, observamos que utiliza parte del método genérico de depuración estudiados en teoría. Los apartados de reproducir y reparar están implementados ya que al generar el reporte se pide información para poder reproducir el error, y en el apartado de reparar se comparten los análisis en el hilo del reporte. En lo referente al apartado de diagnosticar, cada colaborador lo realiza de forma personal sin llevar un registro. Finalmente en el apartado de analizar no se genera ninguna documentación sobre el error, sino que se usará el hilo de referencia ante posibles dudas.

Las incidencias propuestas son de distintos tipos según la plantilla seleccionada, pero no especifican si lo que está ocurriendo es un fallo, un bug o un error. Recomendamos la inclusión de una etiqueta para poder identificar de una forma más precisa el tipo de incidencia y la gravedad de la misma.

Respecto a la forma de informar de las incidencias surgidas, el sistema de plantillas utilizado es bastante útil y da unas pautas de la información que has de rellenar, con el fin de que el desarrollador que tenga que solucionarla tenga el mayor número de facilidades

A la hora de reproducir las incidencias, solamente podemos concretar que el desarrollador realizará las pautas indicadas para su reproducción. No somos conscientes de la existencia de algún método más, ya que Chromium OS no proporciona información de cómo reproduce las incidencias.

Pues podéis proponerlo vosotros

Proponemos generar una documentación de referencia sobre la incidencia para facilitar la consulta, con el fin de poder tener más información de la incidencia descrita, y de esta forma poder conseguir identificar mejor el error, para poder arreglarlo.

A su vez, destacamos que para la resolución de las incidencias tienen una dinámica parecida a Mantis Bug Tracker, pero adaptada por ellos mismos, en la cual existe la posibilidad de en la misma incidencia de comunicarse mediante comentarios, con el fin de proponer soluciones, obtener resultados de propuestas, etc.

Este sistema es bastante estable y seguro, además nos proporciona toda la información relacionada con la incidencia, como por ejemplo el creador de la incidencia, el tipo, el desarrollador que la tiene asignada, etc. Esto es bastante útil para su futura identificación, lo cual consideramos bastante productivo.

Respecto a la forma en la que Chromium OS se encarga de aceptar, descartar, o retardar la resolución de una incidencia, el sistema no proporciona documentación relacionada con ello, por lo que nosotros proponemos la inclusión de un sistema que controle el estado de las incidencias, gestionado por un administrador del sistema, que a su vez es el encargado de aceptar las incidencias, descartarlas si no las considera útiles, o retardarlas si la prioridad de esta esta por debajo de incidencias abiertas con mayor importancia, siempre y cuando no afecte al funcionamiento del sistema de forma notoria.

En lo referente a la depuración, Chromium OS propone dos herramientas para abordar pruebas sobre su proyecto. Hablamos del framework Autotest y de scripts de automatización de pruebas.
Para pruebas en las que el dominio de problema se incrementa y es necesaria la inclusión de muchas clases, elementos, etc, utilizaremos el framework Autotest, el cual ofrece pruebas completas y automatizadas, que pueden ser creadas por los usuarios o por los propios servidores de la aplicación.

Autotest se usará en conjunto con ebuilds(script de procesamiento en lotes) y deps. Un procesamiento en lotes con autotest consiste en la producción de casos de uso y su instalación de forma automática en una zona de almacenamiento proporcionada, en este caso, por Chromium, para su posterior depuración.

Por otro lado, autotest utiliza deps. "***Conceptualmente, un dep autotest es simplemente otro tipo de archivo que la infraestructura autotest sabe cómo recuperar y desempaquetar***"[6], que nos proporciona la ventaja de que no es necesario de incluir ningún archivo de configuración.

La realización de casos de prueba sobre Chromium OS sigue el estándar de *IEEE Standard for Software Testing Documentation* estudiado en clase, ya que realiza los siguientes pasos durante la creación de un caso de prueba:

1. Creación del caso de prueba identificándolo por un nombre/identificador.
2. Inclusión de las entradas para la realización del caso de prueba mediante el uso de ebuilds o deps, dejando la elección al creador del caso de prueba.

---

[6] http://www.chromium.org/chromium-os/testing/autotest-best-practices

3. Obtención de las salidas esperadas e interpretación de los resultados obtenidos
4. Las posibles dependencias entre los ebuilds o deps para poder comprobar si es necesarios la realización de pruebas previas que se encuentran relacionadas con la que estamos creando.

Chromium OS no proporciona información específica sobre en qué momento se deben realizar las pruebas en su proyecto, por lo que recomendamos la realización de ellas durante la etapa más temprana posible, es decir, una vez el código tenga la suficiente envergadura como para soportar pruebas, ya que conforme más avanzado se encuentre el proyecto, la cantidad de errores es bastante mayor, por lo que el número de casos de pruebas para identificar los posibles errores también se incrementa en gran cantidad, y como consecuencia los costes de realización del proyecto incrementarían a niveles demasiado altos.

En la documentación del proyecto Chromium OS no es posible identificar que tipo de mecanismos han sido usados para la depuración del proyecto durante su desarrollo más temprano. Una opción recomendable sería realizar un desarrollo del sistema guiado por pruebas, ya que de esa forma conforme se desarrolla vamos probando casos de prueba para detectar los errores. Sin embargo, al ser un proyecto de software libre, la obligatoriedad de confiar en la habilidad de los desarrolladores involucrados de manera libre en el proyecto hace que este método sea tal vez impracticable. Por tanto, suponemos que quizás en la etapa más temprana del proyecto el sistema guiado por pruebas ha podido ser usado, pero actualmente se sigue otro procedimiento parecido aunque con algunas particularidades.

Concretamente, se ofrece un depuración en local y otra en remoto. Entramos pues en la segunda opción ofrecida para la depuración por parte de Chromium OS: Los scripts. Para la depuración en local, se aconseja el uso de test unitarios en conjunto con un script proporcionado por ellos, que realiza las pruebas sin necesidad de una máquina virtual, lo cual es bastante útil.

Si por otra parte, necesitamos ejecutar nuestra prueba sobre el sistema, se nos ofrece mediante el uso de otro script, la ejecución de la prueba sobre otro pc que posea Chromium OS, de forma remota. Esto es lo que se conoce como depuración en remoto.

Respecto al diseño de los casos de prueba, queremos recalcar que las pruebas que se realicen con autotest deben ser precisas, identificar el error, no sobrepasar los límites de lo que se pide en la prueba y comprobar correctamente las posibles dependencias de ebuilds y deps. De otra forma el caso de prueba podría no funcionar correctamente.

Cabe destacar que Chromium OS no especifica ninguna de las técnica de diseño para un caso de prueba (particiones equivalentes, valores límite, combinaciones: par-wise,n-wise, errores conocidos o cobertura crud), por lo que se ofrece total libertad al desarrollador.

Para obtener una mejora en el diseño de los casos de prueba, proponemos la realización de cobertura CRUD, en el siguiente orden:

- Cada caso de prueba realizado comienza con el créate.
- Tras el create se realizan todas las update y se termina con un delete.
- Tras la realización de un create, update o delete, podemos ejecutar un retrieve para comprobar que todo esta funcionando correctamente.

Usando esta técnica mejoraría la eficiencia del diseño de los casos de uso, ya que un miembro puede ser el creador, y otro distinto podría volver a usarlo o actualizarlo para obtener resultados distintos. Sin embargo, debemos tener en cuenta que las modificaciones o aportaciones que se realicen sobre el proyecto no tienen porque tener una estructura compatible con CRUD, por lo que es posible usar cualquiera de los métodos anteriormente citados.

Este apartado está también bastante pobre con solo referencias y frases genéricas. Uno lo lee y no aprende ni puede reproducirlo ni abordarlo bien. Debería ser mucho más preciso y concreto. Debería dividirse en sub apartados para ser más claro.

Ejercicio de Gestión de incidencias.

El usuario ha de realizar todos los pasos necesarios para crear una nueva incidencia en Chromium OS. No se tendrá en cuenta la plantilla seleccionada, se deja a elección propia.

El ejercicio es muy básico. No se ha realizado una incidencia real, ¿o sí? ¿es esta incidencia una que habéis "soportado"? si fuera así ¿por que no poner un pantallazo de la incidencia hecha en google code?

Resolución

Seleccionar una plantilla en la dirección web https://code.google.com/p/chromium/issues/entry?template=Defect%20on%20Chrome%20OS, por ejemplo Defect on Chrome OS, y rellenar la información que se pide.
Plantilla rellenada, marcando en negrita la información incluida.

Chrome Version: <From about:version: **31.0.1650.63** >
Chrome OS Version: <From about:version: Platform **1.0**>
Chrome OS Platform: <**Toshiba Satellite L750**>
Network info: <**eduroam**>

Please specify Cr-* of the system to which this bug/feature applies (add the label below).

Steps To Reproduce:
1. **Sign up in to the device**
2. **Insert any SD card in the device**

Expected Result:
The device should recognize the SD card and mount it.

Actual Result:
The device doesn´t mount the SD card

How frequently does this problem reproduce? (Always, sometimes, hard to reproduce?) **Every time i try to mount the SD card in my device.**

What is the impact to the user, and is there a workaround? If so, what is it? **Cannot use any SD card**

Please provide any additional information below. Attach a screen shot or log if possible.
Usb sticks are mounting propertly.

Para terminar, pulsar el botón Submit issue para que se cree con el estado de “unconfirmed”

Gestión de la Variabilidad

Tras un estudio detallado de la documentación del proyecto Chromium OS, no se ha encontrado nada relacionado al tratamiento de la variabilidad para el proyecto. Solo esta disponible una versión en distribución, que se usa en portátiles y ordenadores de sobremesa.

Esto se lee muy mal. "software product line" tiene traducción: líneas de producto software.

Chromium OS no usa el nuevo paradigma a la hora de producción de software, que es el SPL(Software Product Line). Recomendamos altamente su uso, ya que ofrece un abanico de personalización del software, proporcionando una customización del producto, dando lugar a un conjunto de productos software.

Estos productos tendrían la misma base, pero con características especiales que los diferenciaban unos de los otros. Usando este método Chromium OS podría desplegar varias versiones distintas de su sistema operativo, como por ejemplo: Chromium OS home, Chromium OS professional, Chromium OS developer, etc.

Para que Chromium OS pudiera modelar correctamente la futura variabilidad que se incluiría en el sistema, habría que decidir cuales features serían útiles para su uso, con el fin de utilizar el paradigma SPL.

Para identificar la variabilidad concreta de un sistema, la mejor técnica es el uso de un modelo de variabilidad, que consiste en un diagrama con los siguientes elementos:

- Puntos de variación
- Variantes de los puntos de variación
- Relaciones entre los puntos de variación y sus variantes, indicando la cardinalidad utilizada
- Restricciones entre puntos de variación y variantes, o viceversa.

Esto no será verdad. Una cosa es que no la modele explícitamente y otra es que no tenga variabilidad lo cual es imposible

Chromium OS al carecer de variabilidad, tampoco ofrece un modelo de variabilidad, por lo que proponemos una alternativa para poder añadir variabilidad al sistema.

Para poder añadir variabilidad al sistema debemos tener en cuenta las siguiente recomendaciones:

- Identificar la variabilidad a implementar
- Definir sus posibles restricciones
- Implementar la variabilidad
- Administrar la variabilidad implementada

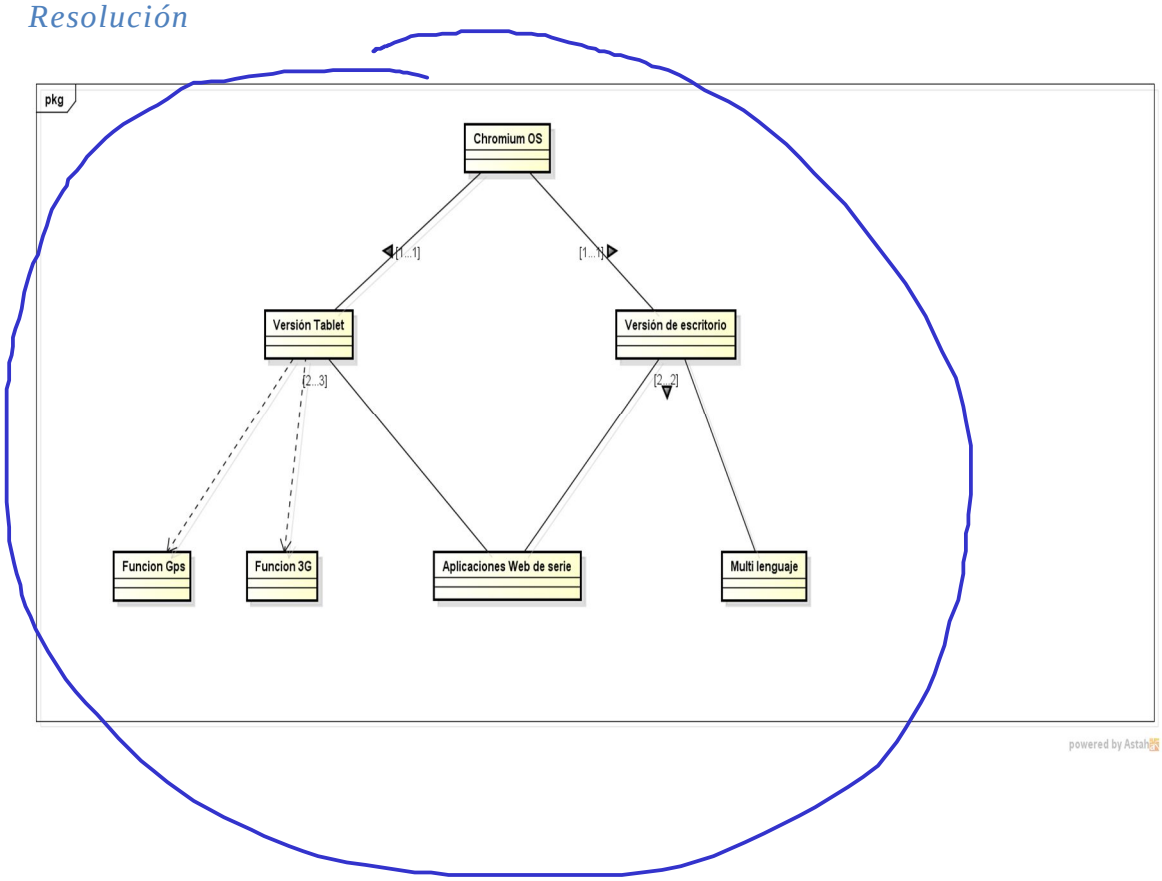
A pesar de la no documentación al respecto, si que hemos encontrado un síntoma de variabilidad en el proyecto Chromium. Se espera que en las próximas fechas, Google saque al mercado una Tablet con Chromium OS. De hecho, en la documentación del proyecto se advierte de este hecho. Por tanto, es probable que si que exista tratamiento de la variabilidad, pero que por razones desconocidas no se haga público.

Este apartado está pobre. Se podrían haber buscado más elementos sobre la gestión de la variabilidad en el código, en los desplegables, etc..., por ejemplo, ¿cómo se gestiona el código de la versión para portátil frente a la de sobremesa?

Ejercicio Propuesto

Proponer un posible modelo de variabilidad para el sistema Chromium OS.

Resolución



Explicar mucho mejor este modelo, ¿qué está modelando?

Integración continua y despliegue continuo

Como ya explicamos anteriormente en el apartado de gestión del código fuente, todo cambio que se desee aplicar al código de Chromium debe ser aprobado por un revisor cualificado, el cual dará vía libre a la subida al repositorio del commit realizado. En la documentación del proyecto Chromium no se alienta a los desarrolladores a hacer commits al menos una vez al día, por lo que sería un primer indicio de que no se sigue un mecanismo de integración continua.

A su vez, tampoco se cumple otro de los diez principios de los método ágiles, el de tener un único repositorio fuente, pues ya hemos analizado que el proyecto se divide en un repositorio por cada componente (está modularizado). Sin embargo, basándonos sólo en la gestión del código no podemos sacar ninguna conclusión sobre la metodología de integración utilizada.

si esto es cierto debería haber quedado claro cómo luego se "mezclan" los repositorios

Es entonces cuando analizando la forma en que se automatizan los procesos de construcción y pruebas llegamos a la conclusión de que efectivamente se utilizan mecanismos de integración continua en la mayor parte de la configuración del proyecto, salvo pequeños detalles que deben ser distintos por el hecho de tratarse de un proyecto de software libre.

¿qué os hace llegar a esta conclusión?

En cuanto a los procesos de construcción, ya hemos comprobado que Chromium OS gestiona la integración con Buildbot, el cual es un framework de Python para gestionar un sistema de construcción centralizado, y que nos asegura soporte para la integración continua. Para ello la herramienta se encarga de automatizar las builds así como las pruebas pertinentes. Este procedimiento es el que nos lleva a pensar que se sigue una metodología basada en la integración continua.

¿lo habéis usado? ¿instalado? ¿probado?

En el apartado de despliegue, la documentación de Chromium no ofrece información sobre qué procesos siguen ni qué herramientas usan. Por tanto no es posible obtener una base sólida en la que asentar el pensamiento de que se están utilizando mecanismos de despliegue continuo, salvo el hecho de que en el apartado de construcción utilizan Buildbot que también ofrece soporte para despliegue continuo.

OJO: Despliegue continuo no es igual a integración continua. Mirad la bibliografía

Entonces cuestionamos si esta posibilidad es factible o no. Si realmente se usase el despliegue continuo en este proyecto, deberían aplicarlo los revisores del código, así como los desarrolladores internos del proyecto, ya que los colaboradores no disponen de autoridad suficiente como para aplicar un commit y desplegarlo directamente hacia producción. Por tanto, el despliegue continuo en este sistema operativo es factible siempre y cuando se hagan desde los estamentos superiores en la jerarquía de roles. Bastaría con seguir utilizando BuildBot para mantener un mecanismo de despliegue continuo.

No se trata de adivinar si se hace o no CI, de lo que se trata es de que propongáis si es bueno hacerlo o no en este proyecto y cómo lo harías, etc..

La gran ventaja que supone esta metodología frente a otras nos lleva a pensar que, aunque no se especifique información al respecto en la documentación del proyecto, realmente se utilizan la integración continua y el despliegue continuo.

A continuación vamos a explicar brevemente qué procesos o mecanismos suponemos son llevados a cabo:
Para un correcto uso del despliegue continuo, cada commit realizado debe ser desplegado instantáneamente, con el fin de identificar los errores más rápidamente.

Respecto a la integración continua del sistema uno de los aspectos más importantes es el de los commits diarios, los cuales son realizados en la rama principal, a fin de reducir los posibles conflictos con el merge, o problemas de integración. Los desarrolladores deberían mantener toda la información en un repositorio simple, a partir del cual se crean las ramas sobre las que trabajar, aspecto que no se lleva a cabo en Chromium.

Para obtener la última versión del repositorio con todos sus recursos ya construidos, recomendamos el uso de un servidor externo de integración continua, ya que ayuda a realizar la tarea mencionada.

Recomendamos la creación de pruebas en una copia del entorno de trabajo, para comprobar que la integración del sistema se realiza de forma correcta, a fin también de corregir posibles fallos encontrados en el sistema. También recomendamos la inclusión de feedback por parte de los desarrolladores, para comentar las posibles mejoras y obtener una integración continua correcta.

Para comprobar que todo funciona correctamente y que la integración se está realizando de una manera adecuada, todos los desarrolladores tienen la posibilidad de obtener el código fuente, para ver las modificaciones realizadas por otros y comprobar en qué cosas se están trabajando.

Ejercicio propuesto

Se propone realizar la instalación y comprobación de la herramienta BuildBot en el entorno de desarrollo.

Nota: Se trabajará con Ubuntu 12.04 y se supondrá instaladas las siguientes herramientas: Python-dev 2.7, virtualenv y Git.

Resolución

Trabajaremos desde un terminal, e iremos introduciendo los siguientes comandos:

- mkdir -p tmp/buildbot
- cd tmp/buildbot
- virtualenv --no-site-packages sandbox
- source sandbox/bin/activate
- easy_install sqlalchemy==0.7.10
- easy_install buildbot

En estos pasos se ha creado un directorio en tmp/buildbot, se ha iniciado un sandbox o zona de trabajo se ha accedido a él y se ha instalado la herramienta.
Ahora crearemos un maestro llamado “master” escribiendo en la terminal actual:

- buildbot create-master master
- mv master/master.cfg.sample master/master.cfg

El archivo “master.cfg” contiene la configuración del maestro. Por ahora usaremos el que viene por defecto. Una vez creado, lo iniciamos y vemos el log para comprobar que todo está correctamente configurado:

- buildbot start master
- tail -f master/twistd.log

Comprobando el log se debe de haber creado el master.

El siguiente paso es crear un esclavo. Para ello abrimos otra terminal y nos colocamos en el directorio de trabajo:

- cd tmp/buildbot
- source sandbox/bin/activate

Instalamos el esclavo:

- easy_install buildbot-slave

Creamos el esclavo:

- buildslave create-slave slave localhost:9989 example-slave pass

Se han elegido esos parámetros para la configuración, ya que son los que vienen por defecto en el archivo “master.cfg”.

- cat master/master.cfg

Pasamos a iniciar el esclavo:

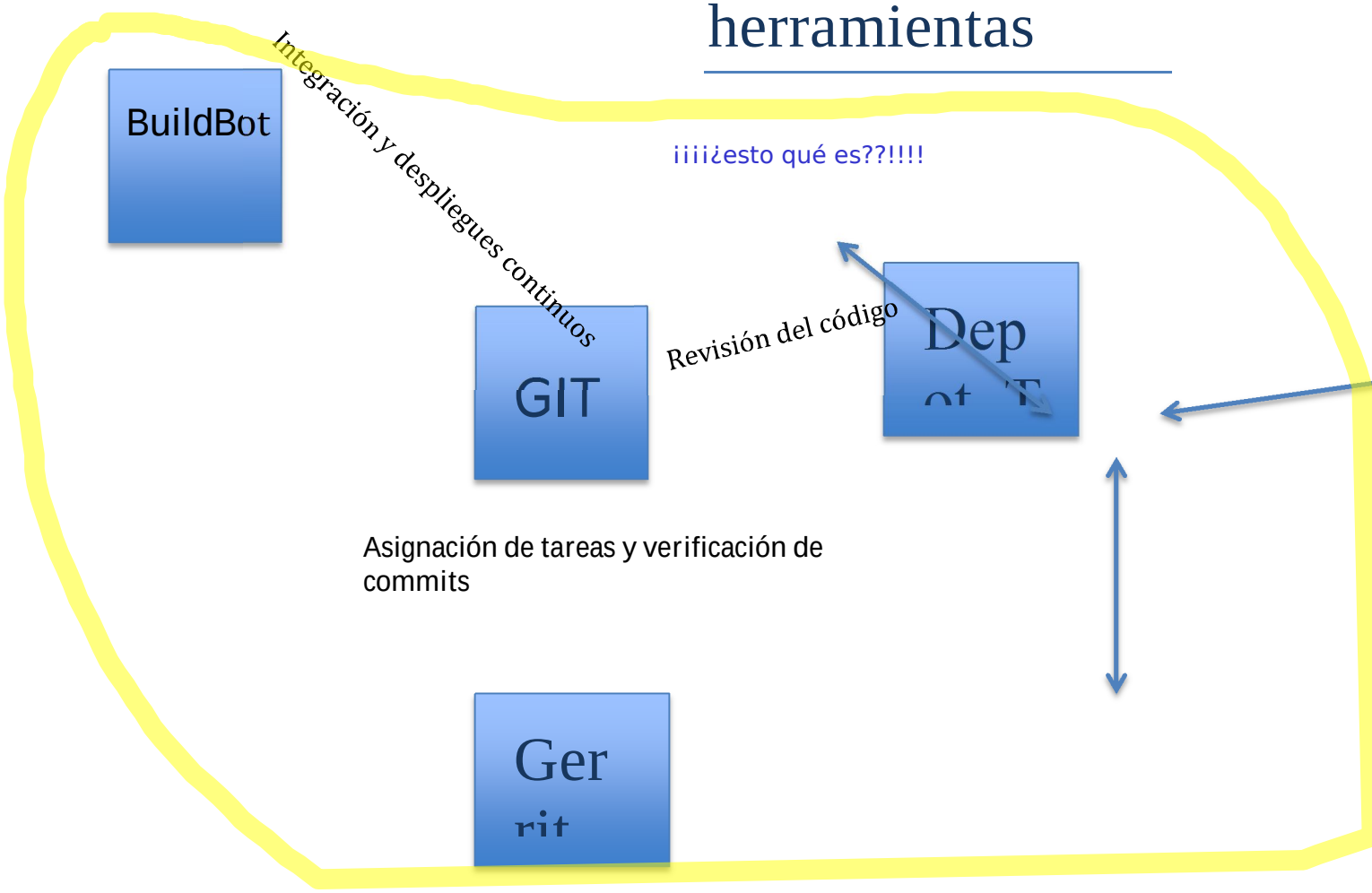
- buildslave start slave

Y miramos el log para asegurarnos de que todo va bien:

- tail -f slave/twistd.log

Con esto ya se ha creado y testeado el entorno de trabajo de buildbot.

Mapa de herramientas



Conclusiones

La numerosa información recabada por el grupo de trabajo durante el desarrollo del documento ha confirmado la dificultad de la gestión de un proyecto de software libre en el que la participación de colaboradores es elevada.

Por otra parte, se han puesto a prueba los mecanismos y estrategias estudiadas en clase, comprobando de primera mano la implementación sobre un proyecto real de gran envergadura.

Se ha conocido de primera mano la importancia de la integración continua y del despliegue continuo, comprobando que son mecanismos que facilitan el trabajo y proporcionan estabilidad.

A si mismo, se han utilizado nuevas herramientas que junto con las proporcionadas por el profesorado nos permiten confeccionar un ecosistema software para la gestión de la configuración de futuros proyectos, con el conocimiento del impacto que tienen sobre estos.

Debido a la falta de documentación proporcionada en numerosos aspectos por el proyecto Chromium OS, el grupo de trabajo ha propuesto bajo su punto de visto las mejores soluciones en cada caso, siendo este un ejercicio de toma de decisiones que nos ha aportado un mayor grado de conocimiento de la asignatura.