Carrera: Ingeniería en Informática

Cátedra: Ingeniería de Software

Profesores: Marcela Vera y Hernán Camusso

Alumnos: Mariano Schaberger y Gonzalo Onisimchuk

Año: 2015

Punto 1 - Especificación de requerimientos de Software

1 - Introducción

El presente documento describe las necesidades de la Cooperativa Eléctrica de Sunchales, la cual necesita automatizar su gestión comercial. Los principales procesos de negocio a considerar en este escenario son:

- Administración de clientes
- Captura de lecturas de medidores
- Facturación de consumos
- Registración de Cobranzas
- Control de la recaudación

Algunos datos sobre la empresa en consideración:

- ➔ Nombre del cliente: Cooperativa Eléctrica de Sunchales
- ➔ Rubro: Servicios públicos
- ➔ Ubicación geográfica: Sunchales, Santa Fe, Argentina
- ➔ Dimensión:
  - ◆ Cantidad de empleados: aproximadamente 150 empleados.
  - ◆ Sectores: se listan los que se relacionan con el enunciado del presente trabajo:
    - ❖ Atención al cliente
    - ❖ Redes
    - ❖ Mediciones
    - ❖ Facturación
    - ❖ Cobranzas
    - ❖ Contaduría
    - ❖ Otros: resto de los sectores de la cooperativa.

En la siguiente sub-sección de la presente introducción se detallan requerimientos para cada proceso.

<u>Administración de clientes: Registración de un nuevo cliente</u>

El proceso de alta de un nuevo cliente se inicia cuando el mismo presenta en el sector de atención a clientes una solicitud de conexión de nuevo suministro. En dicha solicitud debe consignar:

- ➔ Datos personales del solicitante
- ➔ Domicilio donde se solicita la conexión
- ➔ Declaración jurada de potencia instalada en el domicilio por artefactos
- ➔ Planos de instalación eléctrica del domicilio a efectos de su verificación.

Con la presentación de la solicitud, el personal de atención al cliente genera una orden de servicio de inspección la cual se deriva al sector de redes junto con los planos de instalación para la verificación de factibilidad de conexión.

Realizada la inspección, el personal de redes deriva nuevamente la orden de servicio conformada a atención al cliente indicando la factibilidad o no de la conexión.

Si la conexión es factible, atención al cliente procede a dar de alta el suministro y genera una orden de servicio de conexión que es enviada al sector redes. También se genera la factura por cargo de conexión la cual es entregada al cliente.

El sector de redes procede a conectar un medidor en el domicilio. Una vez realizada la conexión, da por cumplida la orden de conexión, quedando así habilitado el nuevo suministro.

Sectores en consideración:

- ❏ Atención al cliente
- ❏ Redes

<u>Captura de Lecturas</u>

Cada fin de período (duración periodo = un mes) se procede a tomar las lecturas de los medidores y se cargan en el sistema los datos.

La lectura de los medidores se realiza de dos formas:

1. **Manual**: una persona es encargada de recorrer los barrios y registrar por cada usuario su consumo.
2. **A través de un sistema SCADA** (que actualmente no se encuentra implementado en todos los barrios de la ciudad): mediante el cual los medidores transmiten su estado a través de la red de electricidad usando power line communication. Una estación ubicada en la central transformadora concentra todos los datos y los transmite al sistema.

Sectores en consideración:

- ❏ Mediciones

<u>Facturación de Consumos</u>

Tomada y cargada todas las lecturas se procede a la liquidación.

En la liquidación existen distintos tipo de clientes:

- Comerciales
- Domiciliarios
- Fundaciones

- Organismos estatales provinciales y nacionales
- Clientes especiales

Cada uno de ellos tiene un régimen de costos y discriminación impositiva distinta.

Sectores en consideración:

- ❏ Facturación

Registración de Cobranzas y control de recaudación

Lugares de pago:

- **Bancos.** Notificación del pago:
  - Tres días después del segundo vencimiento de la factura ó
  - Al comienzo del período siguiente
- **Casas de cobro** (PagoFácil y Rapipagos). Notificación del pago
  - Tres días después del segundo vencimiento de la factura ó
  - Al comienzo del período siguiente
- **En las oficinas** por medio de cheques a la orden y/o tarjetas de débito o crédito. Notificación del pago:
  - Inmediatamente consumada la operación

Sectores en consideración:

- ❏ Cobranzas
- ❏ Atención al cliente

Estadísticas

El sistema debe permitir la elaboración de estadísticas en cualquier momento y por diferentes criterios.

2 - Ciclo de Vida

Para el desarrollo del presente sistema se emplea el ciclo de vida Proceso Unificado RUP. Elegimos este ciclo de vida ya que recopila las ventajas de los modelos de procesos genéricos (modelos en cascada, desarrollo incremental, ingeniería orientada a la reutilización, prototipos y espiral de Bohem). A la vez implementa muchos de los mejores principios del desarrollo ágil de software. Los puntos a favor que encontramos en RUP son:

- Reconoce la importancia de comunicación directa con el cliente para describir su punto de vista del sistema (casos de uso).
- Destaca la arquitectura de software centrándose en las metas correctas (software comprensible, reutilizable y adaptable a cambios).

- Sugiere un flujo de proceso iterativo e incremental, lo que da una sensación evolutiva esencial en el desarrollo moderno de software.

3 - Plan de Entrevistas

Se realizarán diversas entrevistas por cada proceso de negocio considerado en el sistema. Además, en cada tanda de entrevistas se incluirán consultas específicas sobre las estadísticas necesarias en los diferentes procesos.

También se irá consultando qué integración deberá haber con sistemas pre-existentes, tanto internos como externos.

Todas las entrevistas tendrán como objetivo general corroborar los requisitos que ya contamos, analizarlos, validarlos y captar nuevos aspectos que aclaren o generen nuevos requisitos.

Entrevistas para registración de clientes

- Personas/cargos a entrevistar
  - Jefe o supervisor de áreas de atención al cliente y redes
  - Uno o dos empleados operativos que efectivamente atienden clientes
  - Personal que realiza inspecciones y conexiones de redes.
- Posibles preguntas a formular
  - ¿Qué datos personales y de domicilios se deben cargar en la solicitud de nuevo suministro?
  - ¿Las declaraciones juradas y los planos deberán ser escaneados y adjuntados al sistema?

Entrevistas para lecturas de mediciones

- Personas/cargos a entrevistar
  - Personal encargado de la gestión de las mediciones
  - Personal encargado de realizar mediciones manuales
  - Personal encargado de la gestión del sistema SCADA
- Posibles preguntas a formular
  - ¿Cómo se cargarán las lecturas tomadas manualmente?
  - ¿Qué interfaces se necesitan para leer las lecturas desde la central transformadora que concentra los datos del sistema SCADA?

Entrevistas para facturación de consumo

- Personas/cargos a entrevistar
  - Personal encargado del procesamiento de las mediciones
  - Personal encargado de la generación de facturas

    - Personal encargado de administrar criterios de costos e impuestos para cada tipo de cliente.
- Posibles preguntas a formular
    - ¿Cómo procesar las mediciones?
    - ¿Cómo se configura el proceso de facturación de las mediciones?
    - ¿Cuáles son las variantes a tener en cuenta para facturar a cada tipo de cliente?
    - ¿Es necesaria una pre-visualización de lo que se va a facturar?
    - ¿Qué diferencias habrá en la emisión de las facturas en cuanto a su diseño entre uno y otro tipo de cliente?

Entrevistas para el control de recaudación

- Personas/cargos a entrevistar
    - Personal encargado de cobrar a los clientes en las oficinas
    - Personal encargado de gestionar la recepción de los pagos desde casas de cobro y bancos
- Posibles preguntas a formular
    - ¿Se requieren de procesamientos automáticos?
    - ¿Qué formatos tiene la información proveniente de bancos y casas de cobros?

4 - Herramientas Complementarias

Como herramientas complementarias de ayuda en el relevamiento utilizaremos:

- **Diagramas y tablas**: pueden ayudar a resumir, comprender y validar los requerimientos.
- **Escenarios**: por lo general, las personas encuentran más sencillo vincularse con ejemplos reales que con descripciones abstractas. Son útiles para detallar un bosquejo de descripción de requerimientos. Esto implica trabajar con los participantes para identificar escenarios y captar detalles a incluir.
- **Diagramas y casos de uso**: ayuda a describir la interacción entre el sistema y los diferentes actores involucrados en cada proceso de negocio.
- **Etnografía**: observación directa de los usuarios para entender procesos operacionales. Su fin es:
    - Descubrir requerimientos implícitos del sistema que reflejan las formas reales en que trabaja la gente, en vez de procesos formales definidos por la organización.
    - Descubrir requerimientos que se derivan de la cooperación y el conocimiento de las actividades de otras personas.
    - Tener en cuenta estos aspectos puede mejorar significativamente la productividad adquirida con el sistema.

5 - Propósito y Ámbito de la Presente Especificación

Propósitos de la SRS: El propósito del presente documento es implementar un sistema de software de plataforma web que automatice la gestión comercial de la empresa "Cooperativa Eléctrica de Sunchales".

Ámbito de la SRS: El ámbito del producto se centrará en los siguientes procesos de la cooperativa eléctrica: registración de clientes, lecturas de mediciones del consumo eléctrico, facturación del consumo y recaudación de los cobros.

6 - Requerimientos Funcionales y No Funcionales

Requerimientos funcionales (prioridad de 1 (más alta) a 5 (más baja))

- Personal de atención a clientes deberá cargarlo en el sistema, registrando sus datos personales, datos de domicilios, DDJJ de potencia instalada en el domicilio, planos de instalación eléctrica en el domicilio. *Prioridad*: 1
- Personal de atención a clientes deberá generar una orden de servicio de inspección. *Prioridad*: 2
- Personal de atención a clientes deberá poder derivar la orden de servicio de inspección al sector redes. *Prioridad*: 2
- Personal de redes deberá poder asignar factibilidad de conexión a la orden de inspección y derivarla a atención a clientes. Prioridad: 2
- Personal de atención a clientes deberá dar de alta el suministro al nuevo cliente. *Prioridad*: 1
- Personal de atención a clientes deberá generar una orden de servicio de conexión y la derivará al sector redes. *Prioridad*: 2
- Personal de atención a clientes deberá generar e imprimir una factura por cargo de conexión para entregarla al cliente. *Prioridad*: 1
- Personal de atención a clientes deberá poder cobrar facturas (tanto por cargo de conexión como por consumo mensual) a los clientes. *Prioridad*: 1
- Personal de redes deberá poder dar por cumplida la orden de conexión, una vez efectuada la misma. Esto habilitará el suministro dado de alta previamente. *Prioridad*: 3
- Personal de mediciones deberá poder cargar las lecturas manuales de los medidores en el sistema. *Prioridad*: 4
- Personal de mediciones deberá poder importar las lecturas concentradas por la estación transformadora central, con el fin de incorporar las lecturas realizadas por el sistema SCADA. *Prioridad*: 4
- Personal de facturación deberá poder liquidar los consumos periódicos de los clientes. *Prioridad*: 1
- Personal de facturación deberá poder imprimir las facturas de los clientes. *Prioridad*: 1
- Personal de facturación deberá poder agregar, quitar o modificar tipos de clientes. *Prioridad*: 2

- Personal de facturación deberá poder agregar, quitar o ajustar los criterios de facturación de cada tipo de cliente. *Prioridad*: 2
- Personal de cobranzas deberá poder incorporar información de los pagos que realicen los clientes en bancos y casas de cobro. *Prioridad*: 1
- El sistema deberá utilizar la información de los cobros para saldar la deuda por consumos de los clientes. *Prioridad*: 1

Requerimientos no funcionales

- El sistema debe ser intuitivo, fácil de usar y comprensible para los usuarios. *Requerimiento del producto.*
- El sistema deberá estar disponible la mayor parte del tiempo, pero sí o sí deberá estarlo dentro de la franja horaria de trabajo de las oficinas de la cooperativa. Las actualizaciones al mismo deberán realizarse fuera de ese horario. *Requerimiento del producto.*
- El sistema deberá contar con interfaces con el sistema de mediciones SCADA. *Requerimiento de la organización*.
- El sistema deberá ajustarse a los reglamentos legales vigentes respecto a normas y procedimientos técnicos, de medición y facturación de los consumos. *Requerimiento externo*.

Punto 2: "Plan de Gestión de la Configuración"

1 - Actividades de la Gestión de Configuración del Software

a - Ítems de configuración

Elementos a controlar

A continuación mencionamos los ítems posibles que podemos incluir en la configuración:

- Requerimientos de Usuarios: RU
- Especificación de Requerimientos de usuarios: ER
- Lineamientos de desarrollo: LD
- Código
    - Fuentes de los Programas: FP.
    - Scripts de Base de Datos: SBD.
    - Scripts de Tests Unitarios (pruebas): ST.
- Casos de Pruebas: CP
- Manual de Usuario: MU

Esquema general de versiones y criterios de evolución

Las versiones pueden arrancar del 1 en adelante, con incrementos de a 1 y se anexará la fecha y hora de la misma. Por ejemplo:
Especificación de requerimientos para módulo de impresión de facturas de clientes

- Versión 1: inicial sin modificaciones: V1 con fecha al 25/08/2015 a las 19:18:35 se verá como V120150825191835.
- Versión 2: V220150826171533
- Versión 3: V320150829184549
- Etcétera.

Como criterios de evolución podemos utilizar estados para cada elemento considerado como ítem. Consideramos los siguientes:

- Sin cronograma definido: se relevó una necesidad en una o más tareas pero todavía no se definió (planificó) en qué momento realizarlas.
- En cronograma: quedó estipulado en qué cronograma (mes) será abordada una tarea.
- En desarrollo: tarea en desarrollo.
- En prueba: tarea en pruebas de usuario.
- A instalar en producción: tarea probada por el usuario y pendiente de instalar en producción.
- Instalado en producción: tarea instalada en producción (finalizada)
- Finalizado sin implementar: tema o tarea que nunca llegó a efectuarse, situación que puede darse por diversas razones.

b - Otros elementos de configuración del software

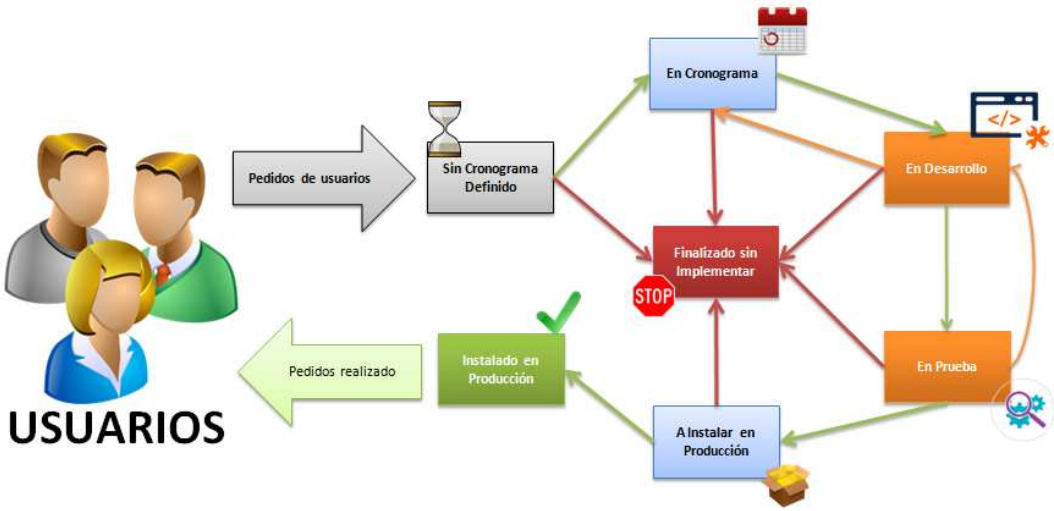
- **Documento de Especificación de Requerimientos**: Cuando surge una necesidad de un usuario (esto puede darse en cualquier momento), sea por proyectos nuevos o mejoras a sistemas actuales, se lo atenderá, y dicha necesidad será registrada en un documento de Requerimientos de Usuarios (RU). Si es la primera vez que se trata esa necesidad, el documento se guardará con versión 1 más un número identificatorio. Por ejemplo, RU + 1234 + V1 (RU1234V1) (Tipo de documento + número de documento + versión del mismo). Si se trata de modificaciones a una necesidad ya cargada como requerimiento, se crea una nueva versión del mismo.
- **Archivos de Código Fuente**: A partir de los requerimientos de usuarios y de la especificación de los mismos, el desarrollador analiza los cambios solicitados y procede con los cambios sobre el sistema para satisfacerlos. A medida que avanza con los cambios los archivos de código fuente se van identificando según su nombre, el nombre del requerimiento, la versión del código y la fecha y hora en que es realizado. Un ejemplo de esto es: ModClientes.cs + ER-015 + V17 + 20150825-190235 (ModClientes.csER15V1720150825190235).
- **Casos de Prueba Unitarios**: Al pasar a prueba una tarea, se involucra a uno o más usuarios para que realicen los casos de prueba unitarias. Un caso de prueba se identifica como CP

(caso de prueba) + V4 (versión + número de versión) + la fecha y hora en que se actualiza el documento del caso de prueba. Por ejemplo CPV520150911093523 (2015/09/11 09:35:23).

- **Manual de Usuario**: A medida que se van agregando nuevas funcionalidades al sistema, se debe actualizar el manual de usuario detallando las nuevas funciones de forma que éste no quede desactualizado a medida que el sistema crece. A medida que va creciendo las distintas versiones se van generando con una nomenclatura similar a la de los anteriores puntos. En ejemplo de esto es ManualModUsuario.exe + MU-2 + V23 + 20150825-190245 (ManualModUsuario.exeMUV2320150825190245).
- **Diagramas de clases**: Cuando se guardan los cambios a una clase que es referenciada por un diagrama de clases, éste se actualiza y se genera una nueva versión del mismo. Por ejemplo, DC (diagrama de clases) + V135 (número versión del mismo) + fecha y hora de versión (es decir, se vería algo como DCV13520150825194835).

2 - Proceso de control de la configuración y cambios en las líneas base

A continuación describimos el proceso de control para la configuración y los cambios en las líneas base mediante la siguiente imagen:



1) Peticiones de usuario: el proceso comienza con una solicitud de cambio por parte de los usuarios. Podemos mencionar algunos de los motivos de cambios:
    a) Gestión de problemas: el tratamiento de errores conocidos puede provocar cambios de infraestructura.
    b) Nuevos servicios: se trata de la ampliación o creación de nuevos servicios de IT

c) Estrategia empresarial: la dirección realiza cambios en la estrategia, lo que puede acarrear cambios de software, hardware y procesos.
d) Actualizaciones de software de terceros: debido a que los proveedores dejan de soportar versiones anteriores de software o hardware.
e) Imposiciones legales: cambios impuestos por la AFIP, por ejemplo.
f) Otros: sugerencias de empleados, clientes o proveedores que pueden ocasionar cambios.

Los pedidos pueden ser rechazados si se considera que el cambio no está justificado, o se puede solicitar su modificación si se considera que algunos aspectos del mismo pueden ser mejorados o re-definirse mejor.

Las prioridades de los pedidos se pueden clasificar de diversas maneras, pero destacamos que los pedidos urgentes tienen un tratamiento diferente al que propone la imagen mostrada anteriormente.

2) Sin cronograma definido: todo pedido que no sea urgente será agregado a una bolsa de pedidos, quedando a la espera de la próxima reunión de planificación, donde se determinará si entra en los cronogramas de desarrollo de los próximos dos meses.
3) En cronograma: de acuerdo a la prioridad de cada pedido, se los asigna a un cronograma mensual de desarrollo para que puedan ser abocados. Los pedidos son insertados a dicho cronograma según su prioridad, fecha de pedido y la capacidad (en horas) que tiene el equipo de desarrollo para abocarse al trabajo.
4) En desarrollo: son pedidos que están siendo procesados por el equipo de desarrollo.
5) En prueba: una vez que se desarrolló un pedido, este pasa a una etapa donde un equipo de testing (conformado por personas abocadas a ello o por usuarios selectos para esta tarea) para constatar de que el trabajo realizado cubre las necesidades del pedido encargado. En caso de fallos, el trabajo es pasado a desarrollo nuevamente para que hagan las correcciones pertinentes.
6) A instalar en producción: los pedidos en esta etapa pasaron con éxito las pruebas de testing y quedan a la espera de un pasaje a producción, el cual puede ser diario, semanal, etc. según el manejo interno de cada empresa o proyecto.
7) Finalizado sin implementar: En todo momento del presente ciclo, un pedido puede ser cancelado. Esto puede deberse a cambios de prioridades en las necesidades del usuario solicitante, como por ejemplo, perder el interés en el cambio solicitado, o volcar toda su atención a otro cambio más urgente, etc.
8) Instalado en producción: se instalan los cambios de software solicitados por el usuario.

A partir de la instalación, el pedido queda cerrado. Todo cambio subsecuente se toma como una nueva solicitud de cambio.

3 - Cambios al PGCS con Desarrollo Rápido de Aplicaciones

Si se aplica un ciclo de vida basado en desarrollo rápido se verían distintas alteraciones en el proceso. Una de las cosas que se trata de evitar, es que un requerimiento quede en estado finalizado sin implementar, ya que al trabajar con iteraciones cortas y a su vez con tareas pequeñas, al momento que se toma un requerimiento el usuario está de acuerdo con ello, de lo contrario tiene la posibilidad de rechazarlo antes de que empiece el desarrollo.

Tomando como ejemplo la metodología Scrum, esta funciona con iteraciones de un determinado tiempo (generalmente 2 o 3 semanas), donde según la importancia de los requerimientos, se van tomando del backlog para realizarlos en las iteraciones. Cada requerimiento no debería durar más de una iteración o debe ser dividido.

En estos casos se vuelve necesario incluir en el versionado de los archivos, el identificador del archivo al que corresponde. Podría ocurrir que al cliente se le entregue un requerimiento donde no todas las funcionalidades estén desarrolladas. Pero sí se entrega algo que agrega valor al producto en vista del cliente.

Además, las metodologías de desarrollo ágil apuntan a usar el mayor tiempo posible en el desarrollo del software, restándole dedicación a la confección de documentación (aclaramos que no es que no se genera documentación en absoluto, sino que se documenta lo mínimo indispensable).

Por otro lado, necesitamos tener en cuenta que se deben identificar distintos roles en el proceso de trabajo, los cuales tendrán distintas funciones y obligaciones. Del mismo modo, es necesario una presencia activa del cliente en el desarrollo del software.

Punto 3

1 – Herramientas disponibles en el mercado

Las herramientas disponibles en el mercado para realizar control de código fuente las podemos encontrar divididas en dos grupos.

Modelo Cliente Servidor

Este modelo consta de un repositorio central al que se accede mediante un cliente instalado en la máquina local. Estos sistemas son:

Subversion (svn): Es uno de los sistemas de control de versión más utilizado. Se utiliza en proyectos como SourceForge, Apache, Python y Ruby, también es utilizado por Google Code para distribuir código. Para sistemas Windows, se suele utilizar la aplicación Tortoise SVN.

Subversion es desarrollado como un proyecto de la Apache Software Foundation y gracias a ello cuenta con una gran cantidad de desarrolladores y usuarios.

Ventajas:

- Gran comunidad de usuarios que lo utiliza.
- Integración con los principales entornos de desarrollo.
- Buena documentación.

Desventajas:

- Puede ser un poco lento.
- Se agrega bastante complejidad al manejar varios branchs.
- No cuenta con demasiados hosting gratuitos.

Concurrent versions system (cvs): fue lanzado en 1986, es uno de los pioneros en este ámbito. Si bien se trata de una tecnología más antigua, sigue siendo útil para cualquier desarrollador.

Ventajas:

- Sistema muy simple.
- Consume pocos recursos.

Desventajas:

- Curva de aprendizaje no muy empinada.
- Tecnología en desuso.

Modelo Distribuido

Entre estos podemos encontrar:

LibreSource: está basado en Java, cuenta con herramientas visuales para facilitar el trabajo en equipo, lo cual lo diferencia de los otros proyectos los cuales son a nivel "línea de comandos". A causa de esto no tiene una gran curva de aprendizaje. Se trata de una herramienta pensada para personas que no quieren aprender técnicas específicas y quieren centrarse más en la comunicación con los miembros del proyecto.

Mercurial: escrito en Python, pretende ser rápido, ligero, portable y fácil de usar. Fue diseñado para proyectos de gran tamaño, por lo tanto no fue pensado para desarrolladores independientes ni diseñadores. Su rendimiento y escalabilidad son una de las características más importantes. Es una herramienta simple de aprender. Mercurial usa un protocolo eficiente, basado en HTTP, que persigue reducir el tamaño de los datos a transferir, así como la multiplicación de peticiones y

conexiones nuevas. Mercurial puede funcionar también sobre ssh, siendo el protocolo muy similar al basado en HTTP.

Ventajas:

- Rápido y escalable.
- Soporta grandes proyectos.

Desventajas:

- Un poco costoso de aprender para principiantes.
- Comunidad no tan grande.

Git es una de las herramientas que está teniendo un rápido ascenso de los sistemas de control de versiones, teniendo un gran impacto para la comunidad de desarrollo web.

Comenzó gracias al núcleo de Linux que es un proyecto de software de código abierto con un alcance bastante importante. Durante los inicios de este, los cambios en el software se pasaron en forma de parches y archivos. Luego, el proyecto del núcleo de Linux empezó a usar controlador propietario llamado BitKeeper. La relación entre la comunidad que desarrollaba el núcleo de Linux y la compañía que desarrollaba BitKeeper se vino abajo, y la herramienta dejó de ser ofrecida gratuitamente. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron durante el uso de BitKeeper.

Algunos de los objetivos del nuevo sistema fueron los siguientes:

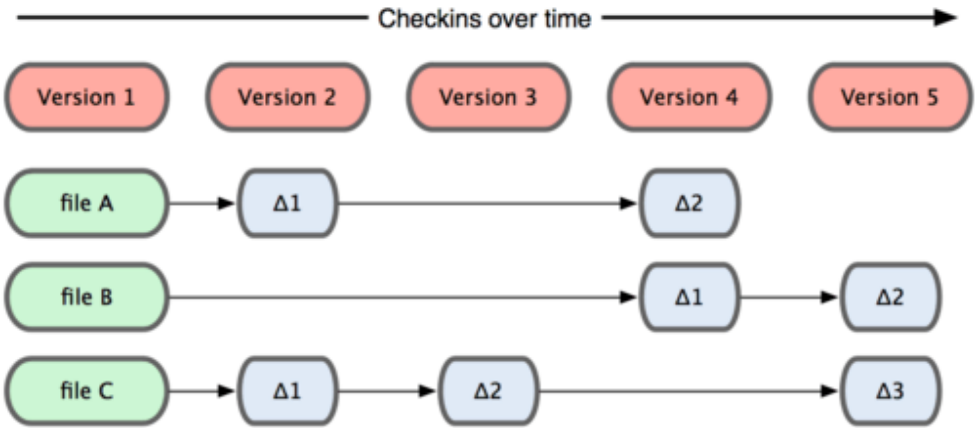
- ✓ Velocidad
- ✓ Diseño sencillo
- ✓ Desarrollo no lineal (miles de ramas paralelas)
- ✓ Completamente distribuido
- ✓ Capaz de manejar grandes proyectos de manera eficiente

Desde su nacimiento en 2005, ha evolucionado y madurado para ser fácil de usar y aún conservar estas cualidades iniciales. Es tremendamente rápido y muy eficiente con grandes proyectos.

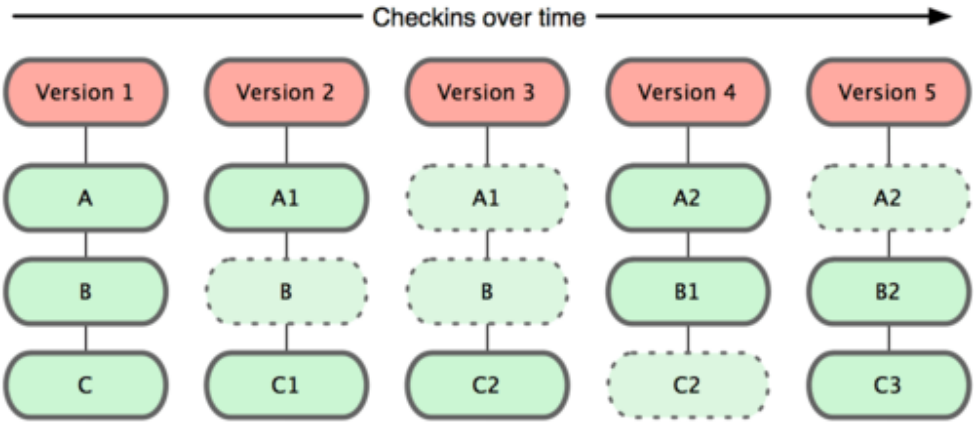
Git se enorgullece de ser un sistema rápido y eficaz, y muchos grandes proyectos de código abierto lo usan para alimentar sus repositorios; Proyectos como: Linux Kernel , VINO , Fedora.

GitHub ha ayudado recientemente establecer Git como un gran sistema de control de versiones, ofreciendo una hermosa parte delantera para muchos proyectos grandes, como Rails y Prototype . Sin embargo, Git no es tan fácil de aprender como CVS o SVN, así que es mucho más difícil de usar para un principiante.

La principal diferencia entre Git y cualquier otro VCS (Subversion y compañía incluidos) es cómo Git modela sus datos. Conceptualmente, la mayoría de los sistemas almacenan la información como una lista de cambios en los archivos. Estos sistemas modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo. Tienden a almacenar los datos como cambios de cada archivo respecto a una versión base.



Git no modela ni almacena sus datos de este modo, los modela más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado. Se modela de la siguiente forma:



Almacena la información como instantáneas del proyecto a lo largo del tiempo.

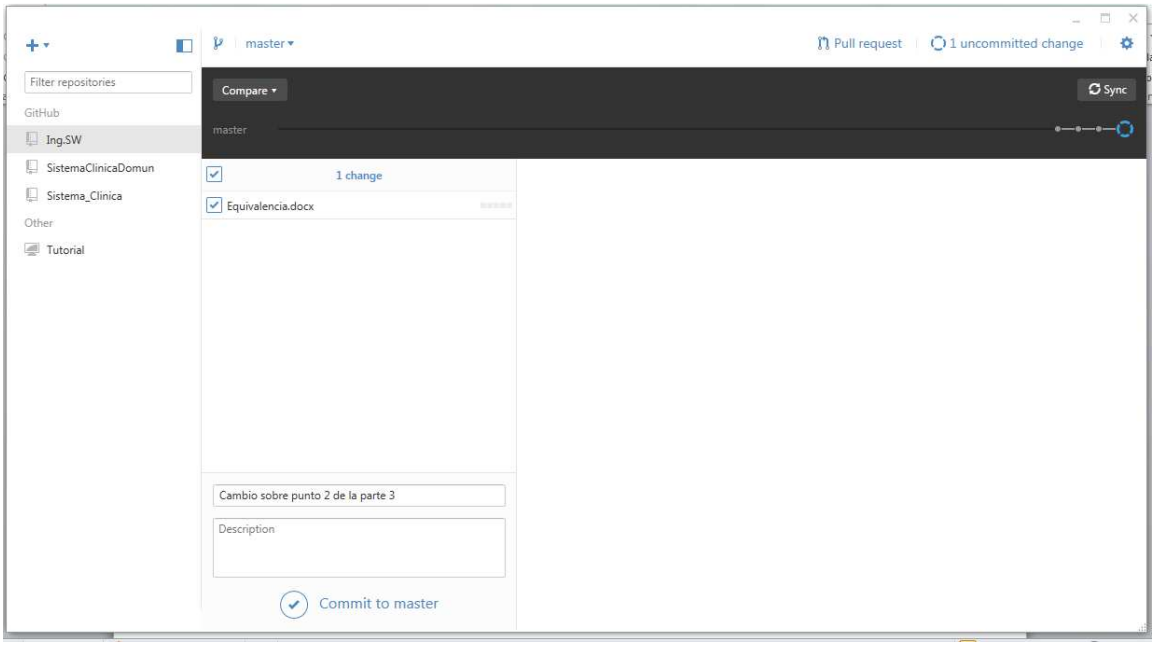
Reconsidera casi todos los aspectos del control de versiones que muchos de los sistemas copiaron de la generación anterior. Esto hace que se parezca más a un mini sistema de archivos con algunas herramientas tremendamente potentes construidas sobre él, que a un VCS.

Alguna de las desventajas que podemos encontrar es que no hay muchos entornos de desarrollo integrados, aunque esto va disminuyendo ya que los principales entornos empiezan a estar integrados. Otra desventaja puede ser que se requiera un poco de documentación a la hora de empezar a utilizarlo, ya que hay algunos conceptos nuevos en él.

2 – Implementación y utilización de la herramienta

Para esta tarea se utilizó la aplicación cliente de GitHub para Windows. Para ello fue necesario crear una cuenta en la página web e instalar la aplicación. Luego, procedimos creando el repositorio “Ing.SW”.

En este punto podemos ver el repositorio creado, junto con el commit de uno de los cambios del documento, siempre trabajando sobre el branch master.



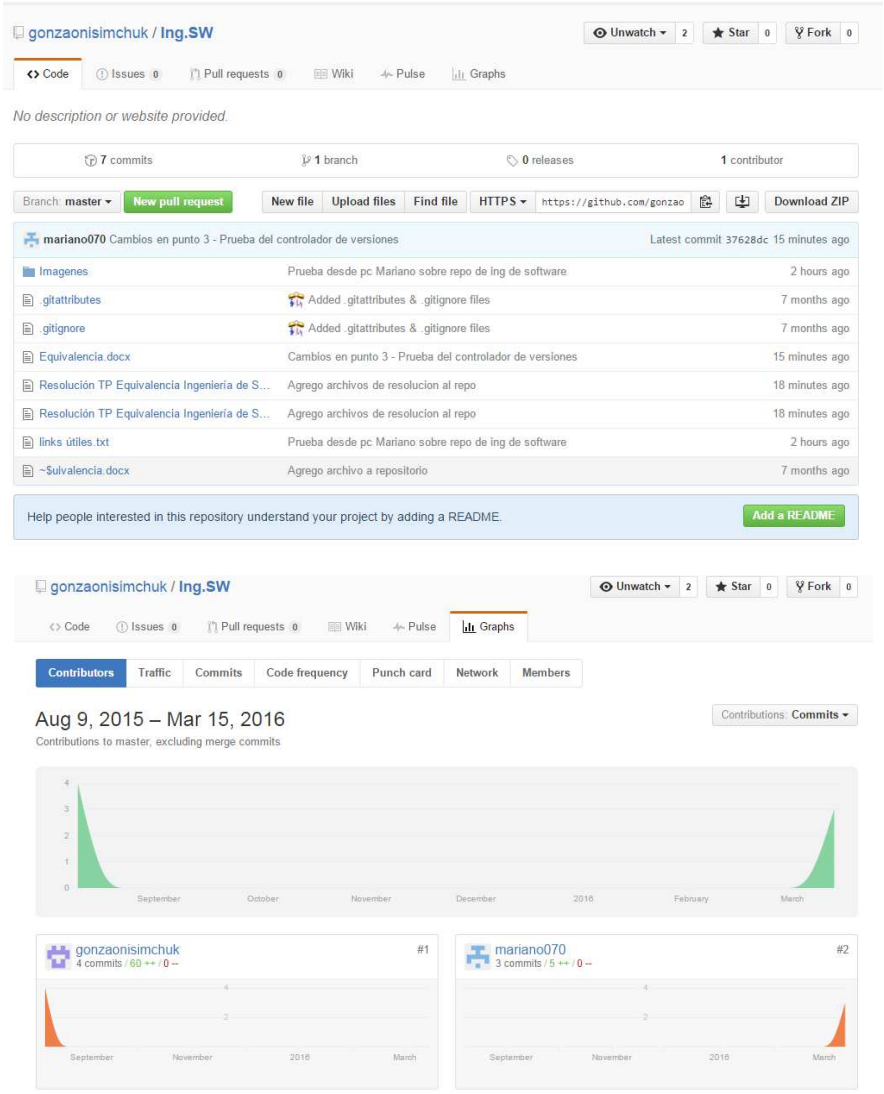
Al utilizar esta herramienta encontramos varias ventajas, una de ellas es que cada vez se vuelve más intuitivo el trabajo con ella, principalmente para los recién iniciados. Otra ventaja es que la mayoría de los principales entornos de desarrollo están empezando a brindar una integración con Git. Además, cuenta con una gran comunidad y prestigio, ya que en alguna ocasión se puede tener distintas percepciones acerca de un desarrollador investigando su perfil en GitHub.

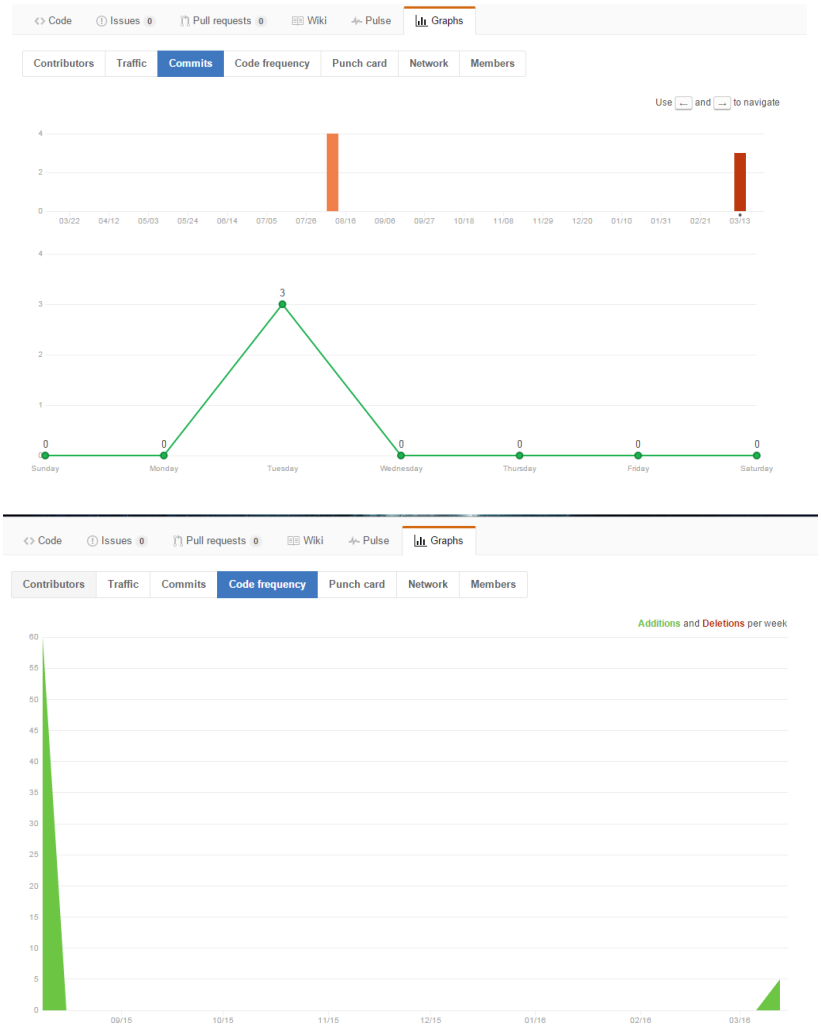
Como desventaja podemos encontrar que si no se utiliza GitHub puede ser un poco complejo al principio. Por otro lado, si se utiliza GitHub, los repositorios privados son pagos.

3- Reporte automatizado y percepciones

En este punto podemos encontrar el reporte automatizado donde se visualiza el historial de cambios del proyecto.

Aquí podemos notar que se trata de 4 changesets distintos, donde en cada uno tenemos su nombre para poder encontrar qué cambios se hicieron puntualmente. En el caso de que se trate de proyecto grande, esto permite ver que archivos se modifican puntualmente y que modificaciones se realizaron.





4 – Ventajas desde un punto de vista práctico

Desde un punto de vista práctico podemos decir que el uso de un controlador de código fuente trae muchos beneficios al ser utilizado. Entre estos podemos encontrar:

- Poder trabajar varias personas sobre un mismo módulo de la aplicación, facilitando el trabajo entre los distintos participantes.
- Poder llevar un historial y auditoría sobre el código fuente.
- Posibilidad de volver a versiones anteriores con gran facilidad.
- Permite realizar distintas ramas de código, posibilitando que se trabaje sobre distintas funciones de una misma aplicación en ambientes distintos y poder unificarlas cuando se desee.

Bibliografía

- Ingeniería de Software-Somerville - Sommerville, 9na Edición, 2011
  - Cap. 2: Procesos de software
  - Cap. 4: Ingeniería de requerimientos
  - Cap. 24: Gestión de la calidad
- Ingeniería de Software: Un Enfoque Práctico - Pressman, 7ma. Edición, 2010
  - Cap. 22: Administración de la Configuración del Software
- http://dis.unal.edu.co/grupos/unbd/manuales/ciclo/cap6_23.htm
- http://sw-notes.blogspot.com.ar/2007/05/items-de-configuracion.html
- http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/vision_general_gestion_de_cambios/vision_general_gestion_de_cambios.php
- https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones