# El modelo de testeo de gvSIG

*Victoria Agazzi – Manuel Madrid*

# El modelo de testeo de gvSIG

## Índice.

# 1. Introducción

## 1.1. Objetivos del área de testeo.

*Los proyectos de software libre están basados en la participación de la comunidad en distintas áreas.*

*Una de las áreas donde más pueden aportar los usuarios es en la de testeo (usuario = tester).*

*Existe un gran número de usuarios a los que les interesa participar en la depuración de errores y en la mejora y que están dispuestos a colaborar de manera más o menos comprometida.*

*Nuestro objetivo es: "* ***hacer de gvSIG una aplicación confiable utilizando como principal recurso la comunidad de usuarios".***

***Se contrata, inicialemente, una consultoría con Dimensión Informática (hoy Indra).***

# 1. Introducción

## 1.2. Líneas de trabajo.

- Sistematización de las pruebas funcionales (común a toda clase proyectos).
  - · Evolución:
    - · Del testeo exploratorio a los planes de prueba.
  - · Propuesta:
    - · Introducción de PDP en nuevos desarrollos.
    - · Generación de PDP para funcionalidades actuales.

- Infraestructura colaborativa (específica proyectos libres).
  - · Evolución:
    - · De las listas de correo al reporte directo.
  - · Propuesta:
    - · Herramientas
    - · Procedimientos
    - · Coordinación

# 1. Introducción

## 1.3. Modelos de testeo.

*- Modelo 1: desarrollo en base a contrato.*

*El desarrollo es fruto de un contrato por el que un promotor contrata a un proveedor para implementar un desarrollo oficial.*

*- Modelo 2: desarrollo donado.*

*El trabajo no se ha desarrollado con la intención de que sea oficial pero una vez hecho público a alguien le interesa integrarlo. El interesado puede ser cualquiera: administración, universidad, empresa e incluso la propia DGg.*

# 2. Sistematización de las pruebas

*2.1. Conceptos sobre planes de pruebas (PDP).*

*2.2. Gestión de PDP.*

*2.3. Ejemplo de pruebas sobre gvSIG.*

*2.4. Documentación disponible.*

*2.5. ¿Quién diseña los PDP?*

*2.6. ¿Quién ejecuta los PDP?*

# 2. Sistematización de las pruebas

## 2.1. Conceptos sobre PDP

*Un plan de pruebas es un conjunto de **estrategias** y **recursos** para llevar adelante una **metodología de pruebas sistemática y planificada**.*

***¿Qué queremos conseguir?***

- ✓ *Tener las pruebas de forma escrita y estructurada*
- ✓ *No dejarnos "huecos" sin testear, o ser conscientes de ello*
- ✓ *Contar con la comunidad de usuarios para la ejecución de las pruebas*
- ✓ *Ser capaces de monitorizar la estabilidad de lo que desarrollamos*
- ✓ *Ser capaces de identificar errores generados por errores corregidos*
- ✓ *Validar comportamiento de gvSIG o de una extensión*

# 2. Sistematización de las pruebas

## 2.1. Conceptos sobre PDP

### Estructura jerárquica

*Esta estructura facilita la **cobertura del plan**, consta de **varios niveles**, con **relación de 1---> n**. Además es simple identificar **zonas no testeadas**.*

***Aplicación:** gvSIG*

- ***Subsistemas:** plugins/extensiones. Lo más independientes posibles.*
  - ***Módulos:** funcionalidades concretas del subsistema*
    - ***Casos de uso:** acciones que los usuarios podrán ejecutar en cada módulo*
      - ***Casos de prueba:** sucesión de pasos para ejecutar los casos de uso.*
        - ***Escenarios:** juegos de datos entrada-salida de cada caso de prueba. Condiciones de comprobación.*

# 2. Sistematización de las pruebas

## 2.1. Conceptos sobre PDP

### ¿Qué tipos de pruebas son incluidas en un PDP?

- ✓ *Pruebas mínimas de instalación (capacitación)*
- ✓ *Pruebas de nueva funcionalidad (alfa)*
- ✓ *Pruebas de persistencia (misma versión, versión anterior)*
- ✓ *Pruebas de regresión (funcionalidades existentes)*
- ✓ *Pruebas transversales, aporte de la comunidad*

*El "secreto" está en tener las **Alfas y Persistencia**, ya que las demás son las mismas pruebas pasadas en diferentes entornos o momentos.*

### ¿Cumplimos estándares con esta forma de estructurar el plan de pruebas?

*Métrica3:* *Planes de prueba para todo el ciclo de desarrollo de software.*

*ISO 9646:* *Normas tests de aceptación. Metodología y estructura. [1998]*
*Salomé TMF implementa la estructura de esta ISO.*

# 2. Sistematización de las pruebas

## 2.1. Conceptos sobre PDP

*Históricamente hemos caminado hacia la* ***sistematización de las pruebas***

***Los tests*** *que hemos generado desde el grupo de testeo de gvSIG* ***son una forma de planes de prueba.***

***Ahora sabemos cómo rediseñarlos para:***

- ✓ ***minimizar la cantidad de casos de prueba,***
- ✓ ***crear escenarios para diferentes tipos de datos,***
- ✓ ***hacer las pruebas de forma más eficiente para cubrir todas las funcionalidades que tenemos.***

# 2. Sistematización de las pruebas

## 2.2. Gestión de PDP

***Salomé TMF***

- ✓ *Gestor de planes de prueba con licencia GNU/GPL*
- ✓ *Acceso a través de un navegador web*
- ✓ *Facilita la definición de la estructura funcional de las pruebas*
- ✓ *Permite adjuntos para cartografía, otros datos, pantallazos, etc*
- ✓ *Facilita la selección de las pruebas a pasar*
- ✓ *Facilita la asignación de los testers que ejecutarán las pruebas*
- ✓ *Facilita la ejecución ordenada y coordinada de las pruebas*
- ✓ *Resume los resultados de las pruebas ejecutadas*

# 2. Sistematización de las pruebas

## 2.2. Gestión de PDP

***Salomé TMF***

*Se ha **modificado la aplicación** para una **mejor gestión** de las pruebas.*

*Se han redactado los **procedimientos paso a paso**, ampliando la documentación existente, para el **diseño, asignación, ejecución e interpretación de resultados**.*

# 2. Sistematización de las pruebas

## 2.3. Ejemplo de pruebas sobre gvSIG

*Las últimas* ***estabilizaciones de extensiones*** *han sido "aprovechadas" para* ***generar pruebas basadas en planes de prueba***

*Para* ***nuevas funcionalidades*** *sobre nuevas versiones se pretende* ***estabilizarlas sobre planes de prueba***

visualizar | editar | propiedades | compartir | versions | traducir a | acciones

### Relacion y estado del los planes de prueba para gvSIG Desktop

por Javier Galán Sánchez Última modificación 26/03/2009 10:55 **Histórico**

Subir un nivel | Anterior Reuniones sobre Planes de Pruebas

**Lista de los subsistemas y el estado en el que se encuentra su plan de pruebas**

| subsistema | estado | comentarios |
|---|---|---|
| Main gvSIG | En proceso | |
| jCRS | Terminado | |
| Symbology | | |
| Editing | | |
| Geoprocess | En proceso | Están terminados solo falta comentar los escenarios y subirlos a Salomé |
| DDBB | | |
| Web Services | | |
| Georeferencing | | |
| Network | | |
| Raster | | |
| Topology | | |
| Normalization | En proceso | |
| Internacionalización | Comprobando | Todavía no está subido a Salomé |

| leyenda | |
|---|---|
| valor | descripción |
| En proceso | Se ha comenzado a realizar el plan de pruebas |
| Comprobando | Se ha terminado la redacción y se está pasando a un build para comprobar la redacción |
| Terminado | Terminado |

# 2. Sistematización de las pruebas

## 2.3. Ejemplo de pruebas sobre gvSIG

***Veamos un ejemplo de diseño, asignación, ejecución y resultados...***

# 2. Sistematización de las pruebas

## 2.4. Documentación disponible

inicio | organización | documentación | descargas | desarrollo | noticias

usted está aquí: inicio → producción → documentación → procedimientos, faqs y howtos → procedimientos → procedimientos de testing → testeo → por metodología de testeo → testeo mediante planes de prueba

### Testeo mediante planes de prueba

Subir un nivel

**Este testeo se basa en las pruebas organizadas según una estructura funcional determinada.**

**¿Qué es un plan de pruebas?** — por Victoria Silene Agazzi — Última modificación 28/01/2009 11:36
Descripción de los planes de pruebas que hacemos en gvSIG

**Consejos útiles para generar planes de prueba** — por Victoria Silene Agazzi — Última modificación 12/03/2009 11:36
Detalle de cada nivel del plan de prueba y ejemplo sobre la extensión JCRS de gvSIG.

**Consejos útiles para priorizar un plan de pruebas** — por Victoria Silene Agazzi — Última modificación 14/10/2008 18:07

**Plan de prueba piloto gvSIG** — por Victoria Silene Agazzi — Última modificación 14/10/2008 18:07

**Gestor de planes de prueba** — por Victoria Silene Agazzi — Última modificación 08/05/2009 15:12
Cómo definir nuevos casos de prueba y ejecutarlos en campañas

# 2. Sistematización de las pruebas

## 2.4. Documentación disponible

### Procedimientos – Salomé TMF

- ✓ *Cómo diseñar las pruebas*
- ✓ *Cómo ejecutar las pruebas*
- ✓ *Cómo monitorizar las pruebas*

**4.2 Creación de los pasos que componen el caso de prueba.**

El primer problema para la gestión de un caso de prueba es que éste puede tener n escenarios, es decir, n combinaciones de valores entrada-salida para una misma ejecución de pasos. La manera de plasmar esta variabilidad en SALOME TMF es mediante lo que llama parameter. Por lo tanto, en la definición de un caso de prueba, cuando se detecte una entrada o una salida que supone una variación del mismo caso de prueba será sustituida por un parámetro. De forma posterior, es decir, tras la inserción de una variable en un paso se deben definir los valores que puede tomar.

Pasando a la gestión de los pasos del caso de prueba mediante SALOME TMF se va a presentar mediante dos apartados, el primero, en el que se define cómo crear un paso de un caso de prueba (action) y el segundo, en el que se especifica cómo insertar una variable (parameter) en un paso.

**4.2.1 Creación de un paso de un caso de prueba.**

Las acciones para la creación de un paso de un caso de prueba son las siguientes:

1. Acceder al gestor de pruebas SALOME, concretamente al proyecto o subsistema con el que se vaya a trabajar.

# 2. Sistematización de las pruebas

## 2.5. ¿Quién diseña los PDP?

*Los PDP se diseñarán en **fase de desarrollo** de las nuevas funcionalidades,*

*Los PDP incluirán principalmente **pruebas alfa** de nuevas funcionalidades,*

*Los PDP serán **un entregable más** del proceso de desarrollo*

*Los PDP tendrán un **seguimiento por parte del área de testeo** del proyecto,*

*Los PDP **se diseñarán sobre Salomé- TMF***

*Las pruebas no son sólo los pasos a seguir: **todos los datos necesarios** (cartografía, parámetros, etc) deberán ser de **libre distribución.***

# 2. Sistematización de las pruebas

## 2.5. ¿Quién diseña los PDP?

*Los desarrollos de funcionalidades que tengan origen en la colaboración, deberemos* ***evaluar la necesidad de desarrollar su PDP*** *o incluir las puebas de dichas funcionalidades en un PDP ya existente.*

***Procedimiento de diseño:***

- ***Reunión inicial*** *para definir estructura funcional, y funcionalidades a cubrir*
- ***Definición de las pruebas*** *en Salomé TMF*
- ***Seguimiento del avance*** *de la pruebas*

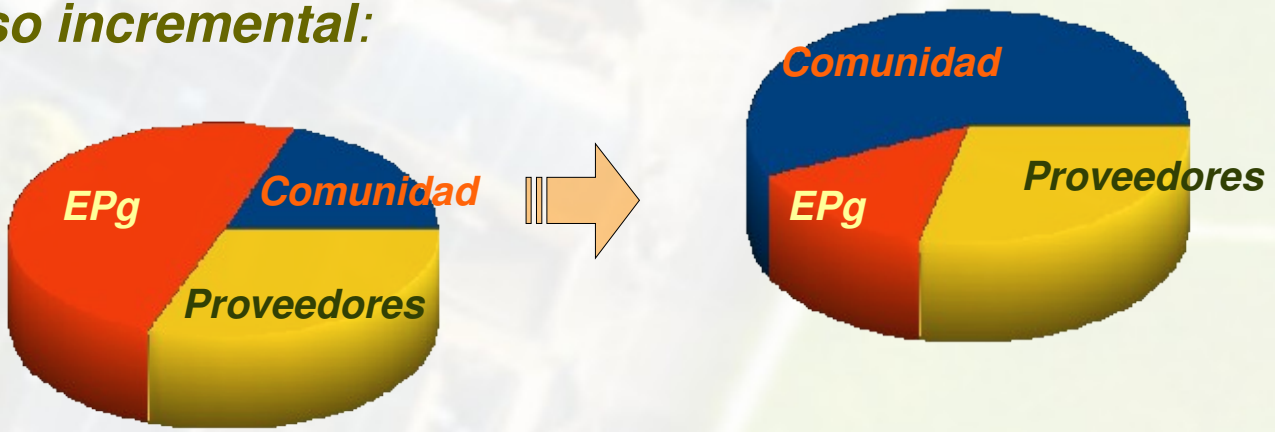
*Una vez se decida entrar en estabilización, se* ***evaluará la cobertura del PDP*** *en función de lo acordado en la reunión inicial.*

# 2. Sistematización de las pruebas

## 2.6. ¿Quién ejecuta los PDP?

*Una vez se empiece la estabilización de un desarrollo, se hará una* ***convocatoria a la comunidad*** *para la ejecución de las pruebas, reporte de errores y comienzo de corrección de los mismos,*

*Los* ***testers ejecutores*** *serán de la* ***comunidad****, de los* ***proveedores*** *y del* ***EPg*** *(Estructura Profesional de gvSIG)*
*Será un* ***proceso incremental****:*




*Se crearán usuarios de* ***Salomé TMF*** *para los ejecutores y se hará* ***seguimiento de las pruebas*** *que realicen y de los* ***resultados de dichas pruebas****.*

# 2. Sistematización de las pruebas

## 2.6. ¿Quién ejecuta los PDP?

*De las sucesivas ejecuciones tendremos* ***identificados los errores a corregir***

*Estos* ***errores serán dados de alta en bugtracking*** *de cara a su corrección y posterior validación*

*Este tipo de testeo es* ***complementario al exploratorio****, pero nos garantizará una* ***estabilidad mínima que queremos cumplir***

# 3. La infraestructura colaborativa

## 3.1. Herramientas.

### 1) Página de descargas de builds de desarrollo.

· Build generado, build publicado.

· Consulta de cambios entre builds consecutivos.

· Consulta de cambios entre versiones

· Importancia de recibir reportes de error desde el principio.

**Subproj. gvSIG Sin determinar** Subproj. build number **1233**

Builds

Changes in this build.

Report a bug

| id | summary | priority | status | resolution | resolve bn |
|---|---|---|---|---|---|
| 1137 | Falta el rumano y el polaco en los idiomas de gvSIG | blocker | closed | fixed | 1233 |
| 1864 | Faltan las traducciones al inglés de las Herramientas de Castilla y León | blocker | closed | fixed | 1233 |
| 452 | Error al exportar a Postgis con nombre de esquema incorrecto | critical | closed | fixed | 1233 |

# 3. La infraestructura colaborativa

## *3.1. Herramientas.*

*2) Formulario de comunicación de errores y sugerencias.*

· *Página web en preparación.*

· *Previsto también reporte desde el propio gvSIG.*

· *Nueva gestión de ficheros de registro de error. Se recogerán de forma automática la mayor parte de los datos relevantes de cara a su traslado a un ticket y a la reproducción del error por parte del desarrollador. Por otro lado esto aligeraría la carga de trabajo en la parte de aceptación de reportes.*

· *Capacidad para realizar búsquedas entre los bugs ya dados de alta antes de reportar uno nuevo.*

# 3. La infraestructura colaborativa

## 3.1. Herramientas.

*3) Seguimiento de incidencias.*

· *El usuario tendrá capacidad para hacer seguimiento de las incidencias que comunique.*

· *El usuario percibe que su reporte es tenido en cuenta.*

· *Posibilidad de participar en la acotación y solución del error.*

# 3. La infraestructura colaborativa

## 3.1. Herramientas.

*4) Catálogo de colaboradores.*

*· Se trata de un registro de posibles colaboradores a los que previamente se les ha propuesto serlo.*

*· Se recogen datos sobre perfil de usuario, es decir, sobre las funcionalidades que puede, prefiere o les interesa testear, S.O., etc.*

*· El objetivo es poder realizar campañas de testeo óptimas además de establecer una relación estrecha con los miembros destacados de la comunidad.*

*· El nivel de compromiso es variable y lo fija en cada momento, como es lógico, el propio colaborador.*

# 3. La infraestructura colaborativa

## 3.1. Herramientas.

*5) Gestor de planes de prueba.*

· *Se trata de la herramienta que se ha presentado en el apartado anterior, imprescindible para coordinar las campañas de testeo, así como para ejecutar y monitorizar las pruebas.*

# 3. La infraestructura colaborativa

### *3.2. Procedimientos.*

*- Son las normas que rigen un determinado flujo de trabajo. Deben indicar claramente cosas como a quién afectan, en qué momento se aplican y qué tareas conlleva. Son sobretodo el "cómo" pero también el "quién" y el "cuando" entre otros.*

*- Disponer de procedimientos documentados es crítico sobretodo de cara a trabajar con colaboradores.*

*- Actualmente todavía faltan procedimentos por documentar.*

*- Por otro lado se esta trabajando en la creación de un sistema de gestión de procedimientos que facilitará en gran medida la ejecución de los mismos (by Model Driven Development).*

# 3. La infraestructura colaborativa

### 3.3. Seguimiento y coordinación.

*- Es en esta parte donde probablemente recaiga la mayor carga de trabajo, al ser una labor horizontal a todo el testeo y se cubrirá normalmente con recursos de la Epg.*

*- Cuanto mayor y más heterogéneo es un grupo de trabajo mayor importancia tiene la coordinación.*

*- Algunas de las tareas de seguimiento y coordinación son:*

- *Gestión de campañas de testeo en las distintas fases.*
- *Coordinación de fases de estabilización.*
- *Validación de correcciones de error.*
- *Filtrado y validación de reportes.*
- *Coordinación y soporte en la generación de PDP.*
- *Etc.*

# 4. Ciclo de vida del testeo

## 4.1. El testeo como actividad transversal

## 4.2. Principales características por fase

- Fase de desarrollo
- Fase de estabilización
- Fase de mantenimiento

# 4. Ciclo de vida del testeo

## 4.1. El testeo como actividad transversal

*El **testeo como una actividad que acompaña el ciclo de vida** de gvSIG*

> *Participación de la comunidad, **cada vez en fases más tempranas**: en 1.1.2 con RC1, en 1.9 con Alpha, en 2.0 desde el 1º build.*

*Las fases del ciclo de vida del testeo están **muy condicionadas por el ciclo de vida de gvSIG o sus extensiones***

*Dependiendo del orígen de desarrollo se requerirá una **coordinación diferente por parte del equipo gvSIG**, pero siempre con el objetivo de **garantizar la calidad** de lo que distribuimos.*

# 4. Ciclo de vida del testeo

## 4.1. El testeo como actividad transversal

| | | Interés incorporación a línea oficial | Diseña las Pruebas | Ejecuta las Pruebas | Coordina las Pruebas |
|---|---|---|---|---|---|
| Modelos de desarrollo | Desarrollo en base a contrato | Promotor | Proveedor | Proveedor + Comunidad + (EPg) | EPg |
| | Desarrollo donado a gvSIG | Administración, Universidad, Empresa, DGg, etc. | Interesado (o su proveedor) | Interesado (o su proveedor) + Comunidad + (EPg) | EPg |

**EPg**: Estructura Profesional de gvSIG.
**DGg**: Dirección General de gvSIG

# 4. Ciclo de vida del testeo

## 4.2. Fase de desarrollo

***Definición PDP:***

*Las **pruebas a nivel usuario** se definirán en función de los **requerimientos del desarrollo***

*Dichas pruebas **se diseñarán antes de llegar a la estabilización** del desarrollo*

*En esta fase nos centraremos en **pruebas alfa**, de nueva funcionalidad y de **persistencia** (misma versión, versión anterior)*

*Los **errores encontrados gracias al diseño del PDP** se darán de alta de cara a su corrección*

***Testeo exploratorio: (¡testers extras en fase de desarrollo!)***

*Recibiremos reportes de error de testeo exploratorio, dichos reportes se filtrarán de cara a mantener la **calidad de la BBDD de bugs** (tareas de mantenimiento)*

*Las **sugerencias** reportadas podrán tenerse en cuenta en la misma versión que se está desarrollando o en futuras versiones*

# 4. Ciclo de vida del testeo

## 4.2. Fase de estabilización

*Al iniciar la fase se **evaluará la cobertura del PDP** diseñado para las funcionalidades a estabilizar. Identificación de áreas en donde el **testeo exploratorio puede ser más efectivo/complementario**.*

*Se prepararán las pruebas del PDP que se quieran pasar que sean de nueva funcionalidad: **pruebas alfa***

*Se prepararán las pruebas del PDP que se quieran pasar que No sean de nueva funcionalidad: **pruebas de regresión***

*Se avisará a la comunidad del inicio de fase: **ejecución del PDP existente y testeo exploratorio***

*Se pasarán las **pruebas de persistencia** (misma versión, versión anterior)*

***Actualización de tickets** dados de alta en fase de desarrollo + tickets de error de versiones anteriores*

# 4. Ciclo de vida del testeo

## 4.2. Fase de estabilización

*Se **evaluará la estabilidad** en función de TODOS estos errores y se **priorizarán qué tickets es necesario abordar***

*Se llevará a cabo una **reunión de coordinación** para dar comienzo a las correcciones de bugs.*

***Se inicia el proceso iterativo de depuración de errores**: corrección, generación de build, validación de corrección.*

*En cada nueva distribución **se validan las correcciones** hechas*

*Con una frecuencia menor se pasarán las **pruebas de regresión** cuyo objetivo es **identificar errores que son consecuencia de correcciones***

***Repriorización de errores** conforme avanza la estabilización*

*Alcanzado el **nivel de estabilización** acordado en la reunión de coordinación, se **publica la versión estable del desarrollo**.*

# 4. Ciclo de vida del testeo

## 4.2. Fase de mantenimiento

*Posibles errores no tenidos en cuenta que merezcan la* ***publicación de una actualización*** *(parche)*

*Del uso de la aplicación,* ***casos de uso reales****, se detectarán* ***errores que se reportarán y filtrarán*** *de cara a mantener la* ***calidad de la BBDD de bugs*** *(tareas de mantenimiento)*

> *Los* ***errores reportados*** *se intentará tenerlos* ***actualizados sobre la última versión en desarrollo****, pero no siempre será posible*

*Las* ***features*** *serán reportadas y evaluadas a la hora de* ***cerrar funcionalidades para la siguiente versión***

***Actualización del PDP****, versionado del mismo para siguientes versiones del desarrollo*

# 5. Conclusiones

## Modelo basado en la participación de la comunidad

### - Repercusiones:

· Los builds deben hacerse públicos desde el primer momento (fase de desarrollo).
· Los cambios entre builds también. Rigurosidad a la hora de documentarlos.
· La infraestructura necesaria para recibir y coordinar la colaboración es compleja.
· La mayor carga de trabajo quizá esté en la coordinación (Epg).

### - Beneficios:

· Gran fuente de recursos aprovechables para el proyecto.
· Exposición a infinitos casos de uso y de prueba imposibles de cubrir mediante PDP.
· Detección temprana de errores al hacer el software público desde la fase de desarrollo.
· Contar con el usuario final es clave de éxito de cualquier producto.

# 5. Conclusiones

***Implantación de la sistematización de las pruebas.***

*- Repercusiones:*

· *Tareas que hasta ahora no se realizaban y que se deberán realizar a partir de ahora.*
· *En desarrollos en base a contratos los PDP se considerarán como un entregable más.*
· *En desarrollos donados la realización de los PDP deberá asumirla la DGg u otro interesado.*
· *Desde el área de testeo deberemos dar soporte en la generación de PDP.*

*- Beneficios:*

· *Mayor control sobre la calidad y estabilidad de la aplicación.*
· *Imprescindible para programas de gran tamaño.*
· *Reducción tiempo estabilización, más palpable cuanto mayor se hace la aplicación.*

# Preguntas y comentarios...

# El modelo de testeo de gvSIG


## ¡Muchas gracias por vuestra atención!

*Victoria Agazzi – Manuel Madrid*