



INTEGRACIÓN DE GESTIÓN DE PRUEBAS A LA ARQUITECTURA DE INTEGRACIÓN CONTINUA DESARROLLADA PARA EL SOFTWARE CIENTÍFICO TÉCNICO





70 años
IUA

1947-2017



HISTORIA

COMPROMISO

INNOVACIÓN



Esteban Agüero

esteban.u.aguero@gmail.com



Alejandro Biagetti

anbiagetti@gmail.com



Itinerario

1.Introducción

2.Marco contextual

3.Marco teórico

4.Concreción de modelo

5.Marco Práctico

6.Conclusión



1

Introducción



PIDDEF 42/11



Metodología y Framework de Gestión de Líneas
Base de Integración de Aplicabilidad en el desarrollo
de Software para el Proyecto UAV

Automatizar tareas rudimentarias



Gestión de Pruebas





¿Necesidad?

Equipo desarrollo e investigación IUA

Se presenta la necesidad de integración de la Gestión de Pruebas a la Arquitectura de Desarrollo perteneciente al Proyecto PIDDEF 42/11, para mejorar el seguimiento y resolución de incidentes, lograr trazabilidad entre casos de pruebas e incidentes, documentar casos de pruebas, organizar dichos casos de pruebas, y servir de apoyo en los procedimientos de validación y verificación de proyectos

Entorno de desarrollo científico-técnico

- Dominio específico y particular
- Los expertos poseen el conocimiento para llegar a la solución
- Inestabilidad de los equipos
- Desconocimiento de buenas prácticas
- Falta de metodologías
- Presencia de requerimientos emergentes
- Incertidumbre de salidas (outputs)
- Falta de documentación, trazabilidad, etc





Nuestro objetivo principal





Integrar la Gestión de Pruebas al conjunto de componentes que forman parte del Sistema de Integración Continua de la Arquitectura de Desarrollo del Instituto Universitario Aeronáutico, mediante el desarrollo de middleware que interconecte las herramientas open-source que dan soporte a dicha gestión.





Objetivos específicos



- 
- Investigar el Entorno-Científico.
 - Estudiar las posibles soluciones para la incorporación de la Gestión de Pruebas en la Arquitectura de Desarrollo del IUA.
 - Investigar y estudiar las diferentes herramientas disponibles de Gestión de Pruebas open-source.
 - Definir criterios de selección, estableciendo valores ponderados de acuerdo a las necesidades.
 - Comparar herramientas de Gestión de Pruebas utilizando los criterios antes mencionados.
 - Selección una herramienta de Gestión de Pruebas adecuada a las necesidades de la arquitectura existente.
 - Integrar la herramienta elegida a la arquitectura existente. Desarrollar middleware para la interconexión de componentes. Generar referencias de buenas prácticas para los equipos de Desarrollo de Software Científico Técnico.
 - Evaluar el funcionamiento de la herramienta en el entorno de pruebas.
- 



2


Marco contextual



Entorno



Científico-Técnico



Un entorno “particular” - Muchas de las herramientas de la IS no son tenidas en cuenta

Los miembros de un equipo integran sus trabajos con frecuencia. Reduciendo riesgos y tiempos



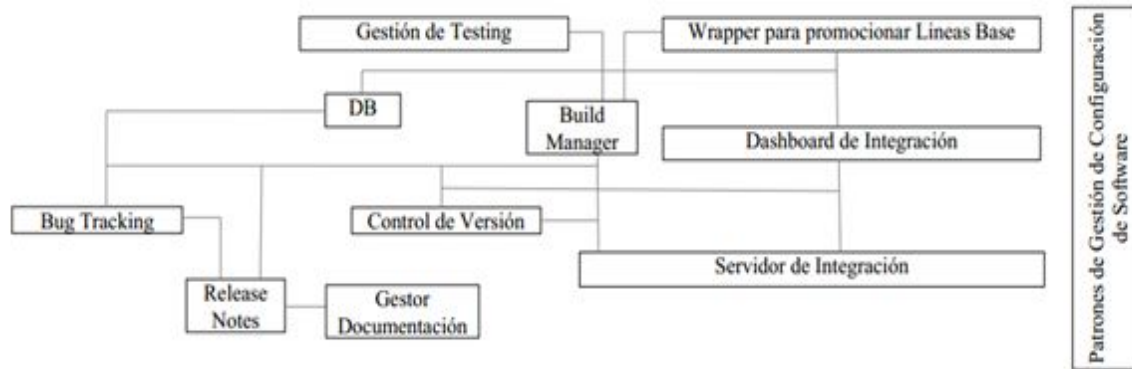
Integración Continua



Arquitectura PIDDEF

Framework para el desarrollo de software C-T, con la misión de automatizar parte de los procesos DS

Arquitectura



Arquitectura PIDDEF

A decorative graphic on the left side of the slide consists of several hexagons of varying shades of blue and cyan. Some hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and four connecting lines. The hexagons are arranged in a cluster, with the largest one in the center containing the number 3.

3

Marco teórico

¿Calidad?

Característica para
comparar virtudes



Se puede calcular



Subjetividad en el tiempo



Mediante observación de
diferencias entre productos
iguales



Aplicación de principios y técnicas
en todas las fases de producción



ISO 9000

“Grado en el que un conjunto de características inherentes a un objeto (producto, servicio, proceso, persona, organización, sistema o recurso) cumple con los requisitos”



En la Ingeniería de Software



R. Pressman



Cumplimiento de los requisitos funcionales, de los estándares documentados



Watts Humphre



El foco principal debería ser la necesidad del cliente. Perspectiva del usuario



Al Davis



Satisfacer las necesidades del cliente, por más que los procedimientos no estén documentados.



IEEE



“El grado con el cual un sistema, componente o proceso cumple con los requerimientos y con las necesidades y expectativas del usuario”

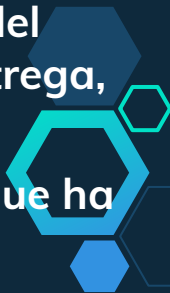




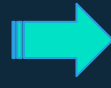
Principios básicos



- ❧ La calidad debe ser una preocupación durante todo el ciclo de vida del software.
- ❧ Sólo se alcanza con la contribución de todas las personas involucradas.
- ❧ Debe ser planificada y gestionada con eficacia.
- ❧ Dirigir esfuerzos a prevención de defectos.
- ❧ Reforzar los sistemas de detección y eliminación de defectos durante las primeras fases.
- ❧ La calidad es un parámetro importante del proyecto al mismo nivel que los plazos de entrega, costo y productividad.
- ❧ Es esencial la participación de la dirección, que ha de proporcionar la calidad.



Testing



Evaluar la calidad del producto, identificar defectos, problemas.



Verificación

Dinámica



Estática



“
IEEE

“Es el proceso de evaluar un sistema o componente de un sistema de forma manual o automática para verificar que satisface los requisitos esperados, o para identificar diferencias entre los resultados esperados y los reales”

¿Qué nos aporta?

Calidad durante todo el proceso



Disminución de costos



Reducción de riesgos



Optimización de recursos



Seguimiento de estándares





V&V

Verificación

El proceso de evaluación de software para determinar si los productos de una determinada fase de desarrollo cumplen las condiciones impuestas en el inicio de esa fase.



¿Construimos el producto correctamente?

Validación

El proceso de evaluación de software durante o al final del proceso de desarrollo para determinar si cumple los requisitos especificados.



¿Cumple las expectativas del cliente?



Aseguramiento de la calidad

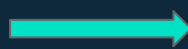
“Una guía planificada y sistemática de todas las acciones necesarias para proveer la evidencia adecuada de que un producto cumple los requerimientos técnicos establecidos.

Un conjunto de actividades diseñadas para evaluar el proceso por el cual un producto es desarrollado o construido”



Misil Atlas

1950



Lanzamiento de satélites y
sondas espaciales



Evolución de la V & V





Técnicas de testing



White Box (estructural)

Black Box (funcional)

Gray Box



Manual

Automático



Estático

Dinámico



Test en entornos ágiles

Objetivo: Mejorar la calidad del software

Responder con mayor rapidez

Desarrollo de a pequeños pasos

Validación con el cliente

Vamos por buen camino?

Entorno Colaborativo





Pruebas según nivel de Componentes

Unidad

Conjunto de uno o más módulos de un programa.

Ej: una clase en un lenguaje orientado a objetos.

Integración

Comprueban que la unidades de prueba funcionan correctamente cuando se integran. Se va probando la arquitectura software.

E: interfaces entre componentes.

Sistema


Relacionadas con el comportamiento de un sistema completo, definido por el alcance del proyecto.

Ej: Desempeño, Stress, Carga, Recuperación, Tolerancia a Fallos.

Aceptación

Conducidas a determinar cómo el sistema satisface criterios de aceptación validando los requisitos.

Ej: Aceptación de usuario, Alfa, Beta.



Test Plan – Plan de Pruebas

Proporcionar la base para llevar a cabo el conjuntos de pruebas de manera organizada

Atributos














El ambiente

- Es una buena oportunidad para detectar la mayoría de los defectos
- Es flexible
- Es ejecutado con facilidad y de forma automática
- Documenta los resultados esperados
- Define claramente los objetivos de la prueba
- Clarifica las estrategias de pruebas
- Define claramente los criterios de las salidas de las pruebas
- No es redundante
- Identifica riesgos
- Documenta los requerimientos de la prueba



Test Plan: componentes

-  Identificador
-  Alcance
-  Elementos a probar
-  Estrategia
-  Categorización de la configuración

-  Entregables
-  Procedimientos especiales
-  Recursos
-  Calendario
-  Manejo de riesgos
-  Responsables



Test Case – Caso de Pruebas

“...Conjunto de entradas de prueba, condiciones de ejecución, y resultados esperados desarrollados con un objetivo particular, tal como el de ejercitar un camino en particular de un programa o el verificar que cumple con un requerimiento específico..” IEEE 610

“



Productividad



Capacidad de prueba



Estimaciones fiables



Formado por

Propósito - Método - **Flujo** - Versión - Datos
de **entrada** - **Resultados** - Documentación



Factores de calidad de los TC

 Correcto

 Exacto

 Económico

 Trazable

 Confiable

 Medible

Formato de los TC



Paso a Paso

GUI



Matrices

Forms



Scripts automatizados





Buenas prácticas



Lenguaje



Describirlos de forma clara y precisa, sin ambigüedades.

Longitud



No contar con muchos pasos, entre 8 y 16. En script no se tiene mucho en cuenta la longitud. Si el mantenimiento y la administración

Casos acumulativos



Casos que dependen de otros. Muchas veces es necesario mantener la autonomía.

Administrar



Incrementa la productividad, la facilidad de consulta, movimiento, escritura.



Automatización

Ejecutar los TC de forma automática, leyendo la especificación del mismo de “alguna forma”, ya sea un lenguaje genérico, una herramienta, hojas de cálculo, etc.



Ventajas

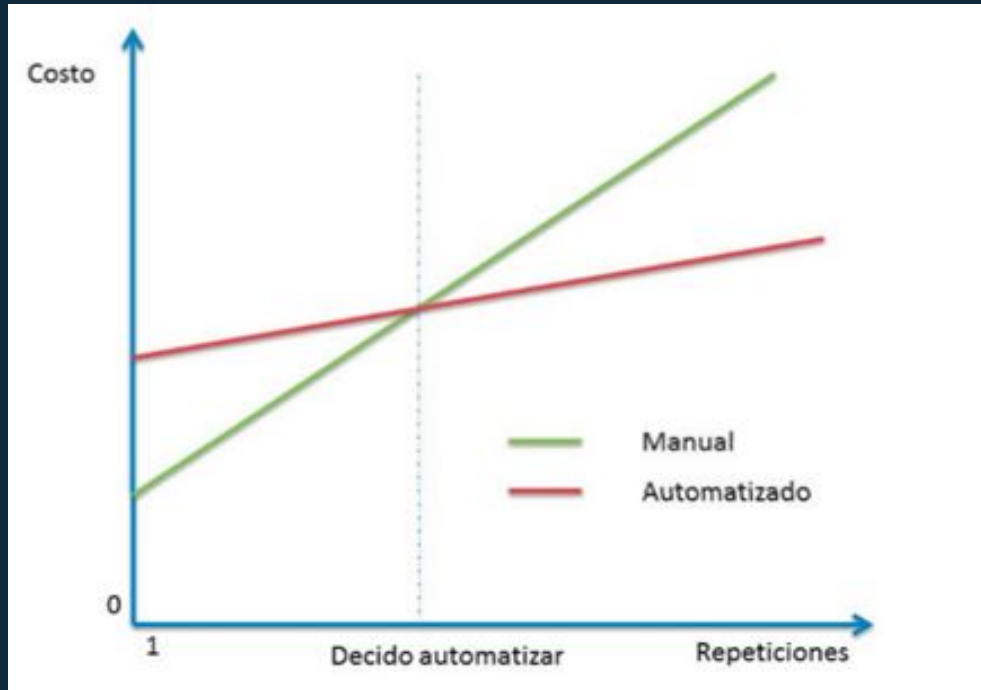
- + Cobertura — Riesgos
- + Ejecuciones — Tiempo
- + Encuentra defectos tempranos
- + Formalización del proceso
- + Reutilización



Desventajas

- + Mayor esfuerzo inicial
- + Costo mantenimiento de los scripts (si los Req cambian)
- + Usabilidad?

¿Cuándo decido automatizar?



Roles



Planea, monitoriza, controla

Test Leader



- Coordina las estrategias de prueba.
- Planifica
- Monitorea resultados
- Realiza acciones de acuerdo a las necesidades
- Introduce métricas para medir el progreso
- Qué debe ser automatizado, en qué nivel, y cómo
- Arquitectura y ambiente de prueba
- Confecionar eeportes

Tester



- Contribuir al Test Plan
- Analizar, revisar, evaluar los Req.
- Configurar ambiente de pruebas
- Implementar las pruebas, y llevar registro, documentar.
- Automatizar los test.
- Monitoreo si es necesario.
- Medir rendimientos de los componentes
- Revisar las pruebas desarrolladas por otros testers.

Métricas

Métricas Base

Métricas Calculadas

Ejemplos



% de casos de prueba ejecutados <No ejecutados>

$R = (\text{N}^\circ \text{ Casos de Prueba Ejecutados} / \text{N}^\circ \text{ de casos de Pruebas Totales}) * 100$



Nº de casos de prueba exitosos <Fallidos><Bloqueados>

(Métrica Base)



Densidad de defectos

$R = \text{N}^\circ \text{ de defectos (Métrica Base)} / \text{tamaño}$



Limpieza de código

$R = \text{test que fallaron alguna vez} / \text{test totales}$





4

Concreción de modelo



1. Investigación de
herramientas

2. Ponderación de
Herramientas

3. Elección de
Herramienta



Herramientas investigadas



TestLink



Testopia



RTH - TURBO



Radi



Salomé




Tarántula




Tabla de ponderación

	Pesos	Testopia	Ponderación	ST	TestLink	Ponderación	ST	RTH	Ponderación	ST
Usabilidad (0-10)	7		7	49		4	28		4	28
Integración Bugzilla (0-1)	8		1	8		1	8		0	0
Requerimiento (0-1)	5		0	0		1	5		1	5
Soporte (0-10)	8		8	64		7	56		5	40
Actualizaciones (0-10)	7		2	14		2	14		0	0
BD (0-2)	6	Mysql	2	12	Mysql	2	12	Mysql	2	12
Usuarios (0-1)	5		1	5		1	5		1	5
Multi-idioma (0-1)	3		1	3		1	3		1	3
Lenguaje		Perl			PHP			PHP		
Interfaz de usuario		Web			Web			Web		
TOTAL				155			131			93





	Pesos	Radi	Ponderación	ST	Salomé	Ponderación	ST	Tarantula	Ponderación	ST
Usabilidad (0-10)	7		4	28		5	35		8	56
Integración Bugzilla (0-1)	8		0	0		0	0		1	8
Requerimiento (0-1)	5		1	5		0	0		1	5
Soporte (0-10)	8		4	32		5	40		6	48
Actualizaciones (0-10)	7		1	7		1	7		2	14
BD (0-2)	6	xml	0	0	Mysql	2	12	Mysql	1	6
Usuarios (0-1)	5		1	5		1	5		1	5
Multidioma (0-1)	3		1	3		1	3		1	3
Lenguaje		Python			JAVA			RUBY		
Interfaz de usuario		Web			Web			Web		
TOTAL				80			102			145



Testopia



Ha sido diseñado para proporcionar un repositorio central que sea de utilidad para la comunidad de testers. Sirve tanto como **repositorio** de TC como un **sistema de gestión**. Su estructura encuadra para satisfacer las necesidades de la comunidad de Tester, independientemente del tamaño del grupo y de la organización a la que pertenezcan.

Aunque Testopia fue diseñado para la prueba de software, puede ser utilizado también para el **seguimiento de cualquier tipo de TC**.

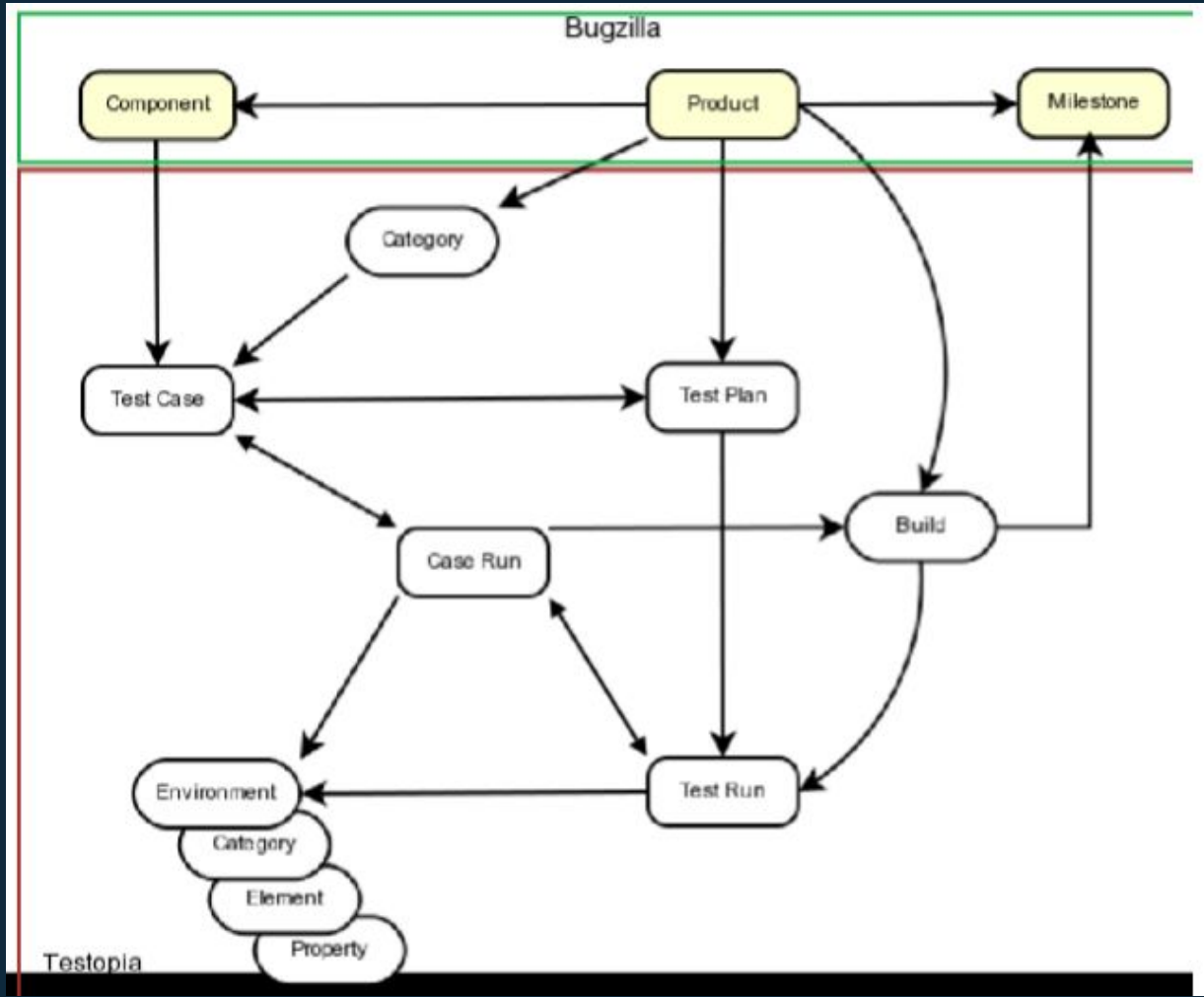
Y, al ser, de código abierto, también permite ser **modificado** de acuerdo a las **necesidades del entorno**.



Requerimientos

- Bugzilla: 3.x
- MySQL: 5.0
- Módulos PERL
- JSON: 1014
- Text::Diff
- Ext JS toolkit: 2.0.1

Arquitectura



GUI

A decorative graphic on the left side of the slide consists of several hexagons of varying shades of blue and cyan. Some hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and four surrounding nodes. The hexagons are arranged in a cluster, with the largest one in the center containing the number 5.

5

Marco práctico



1.Instalación Testopia

2.Diseño
aplicación
Middleware

3.Configuración
de Middleware

4.Cruise
Control

5.Bugzilla/Testopia

6.Prueba de
funcionamiento

Instalación de Testopia



v2.5 Testopia



v4.2 Bugzilla



Middleware

Objetivo: Interconectar los resultados de las distintas ejecuciones de CC con el correspondiente TC configurado en Testopia

- El resultado de la ejecución del TC
- Nuevo Bug, en el caso de que la ejecución no haya sido correcta
- Desactivar la automatización para evitar un exceso de bugs repetidos

Actualiza



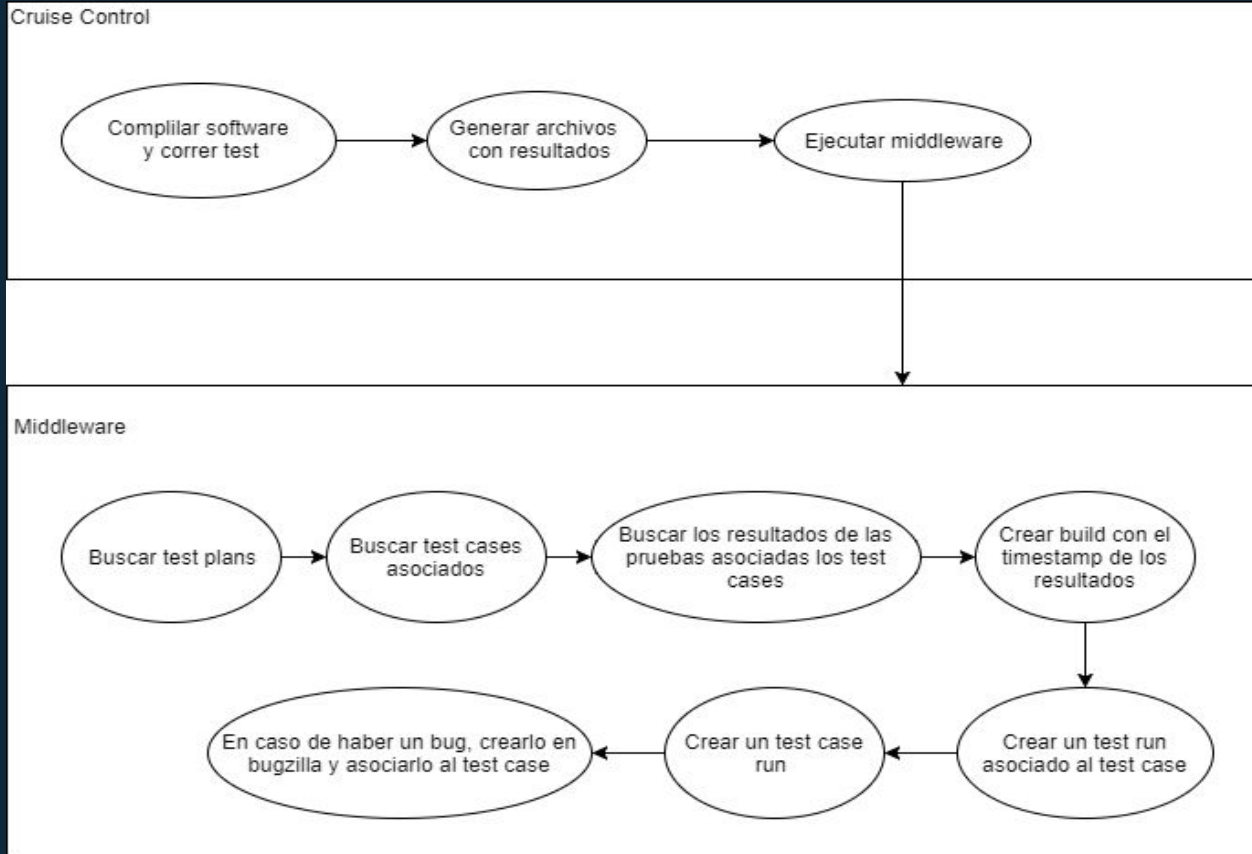
Lenguaje: Java

API: XML-RPC

Clases: BuscarConf - BuscarXML - Reporte -SAXHandler
- UpdateTestCaseRun - Main



Middleware





Configuración Middleware


El programa middleware necesita de parámetros de configuración para poder conectarse tanto con Bugzilla como con Cruise Control.

↓
3 partes

parámetros para realizar las conexiones con Bugzilla y Cruise Control

↓
configuraciones que se le aplicaran a los Bugs que se creen automáticamente

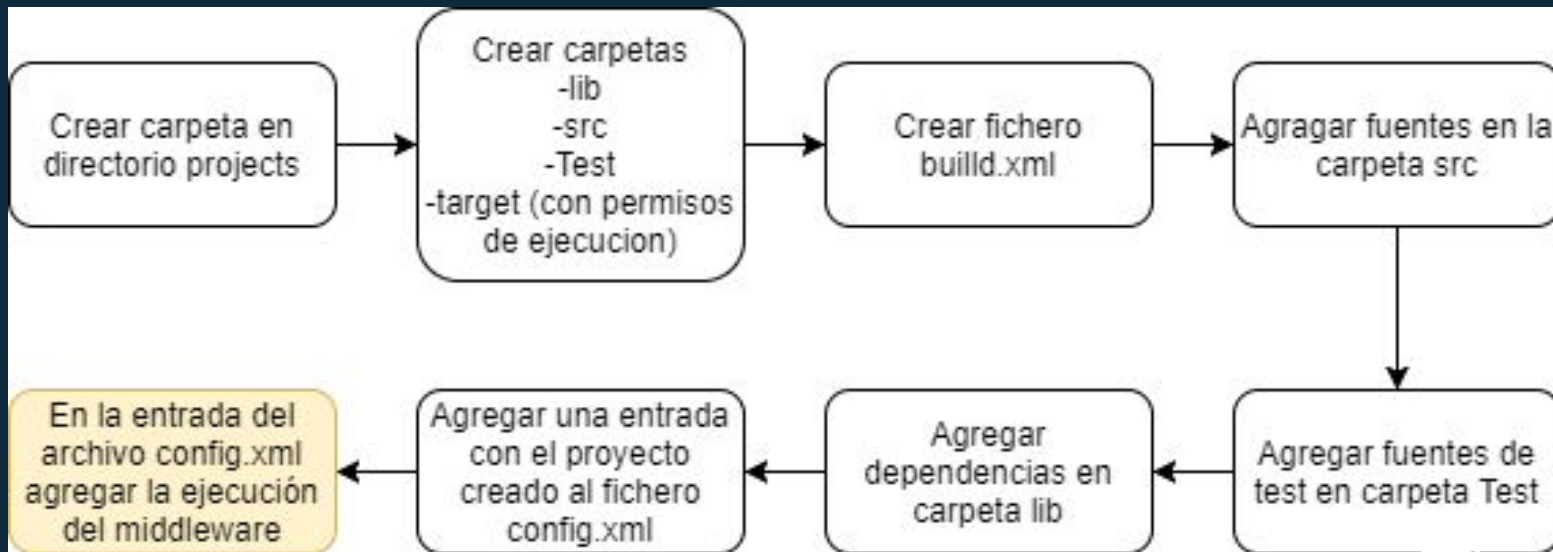
↓
configuraciones de los TestRuns que la aplicación creará (Ej: asignación a usuarios)





Cruise Control

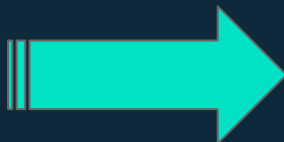
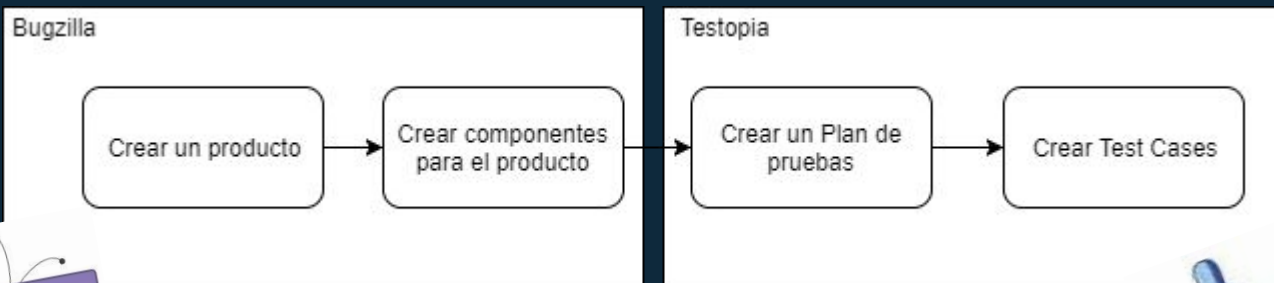
Resumen de pasos





Bugzilla - Testopia

Resumen de pasos





Prueba de funcionamiento

1. Ejecutar el comando “service cruisecontrol start”
2. Ingresar a url Bugzilla/Testopia:
<http://accesodi.iaa.edu.ar/testopia/bugzilla/>
3. Elegir la opción “Product Dashboard” del panel de herramientas.
4. Seleccionar el Producto creado en la sección anterior.
5. Obtener información de los Test Runs (ejecuciones individuales de los casos de prueba): a. Seleccionar la pestaña Test Runs
6. Realizar acciones según sea el resultado del case.





6

Conclusión

Se logró

- ✓ Establecer lineamientos que dan soporte a la validación y verificación al Software desarrollado en el IUA.
- ✓ Investigar y seleccionar herramienta para la Gestión de Pruebas que pueda interactuar con la Arquitectura PIDDEF 4211
- ✓ Desarrollar un componente Middleware para la interconexión de las distintas tecnologías de la Arquitectura.
- ✓ Automatizar el seguimiento de las tareas de prueba. Favoreciendo la documentación, organización, seguimiento y trazabilidad.
- ✓ Automatizar el seguimiento de las tareas de prueba. Favoreciendo la documentación, organización, seguimiento y trazabilidad.
- ✓ Brindar guías prácticas referenciales y de roles para los equipos.



Gracias!

