Planificación y Pruebas de Sistemas Software

Curso 2015-16

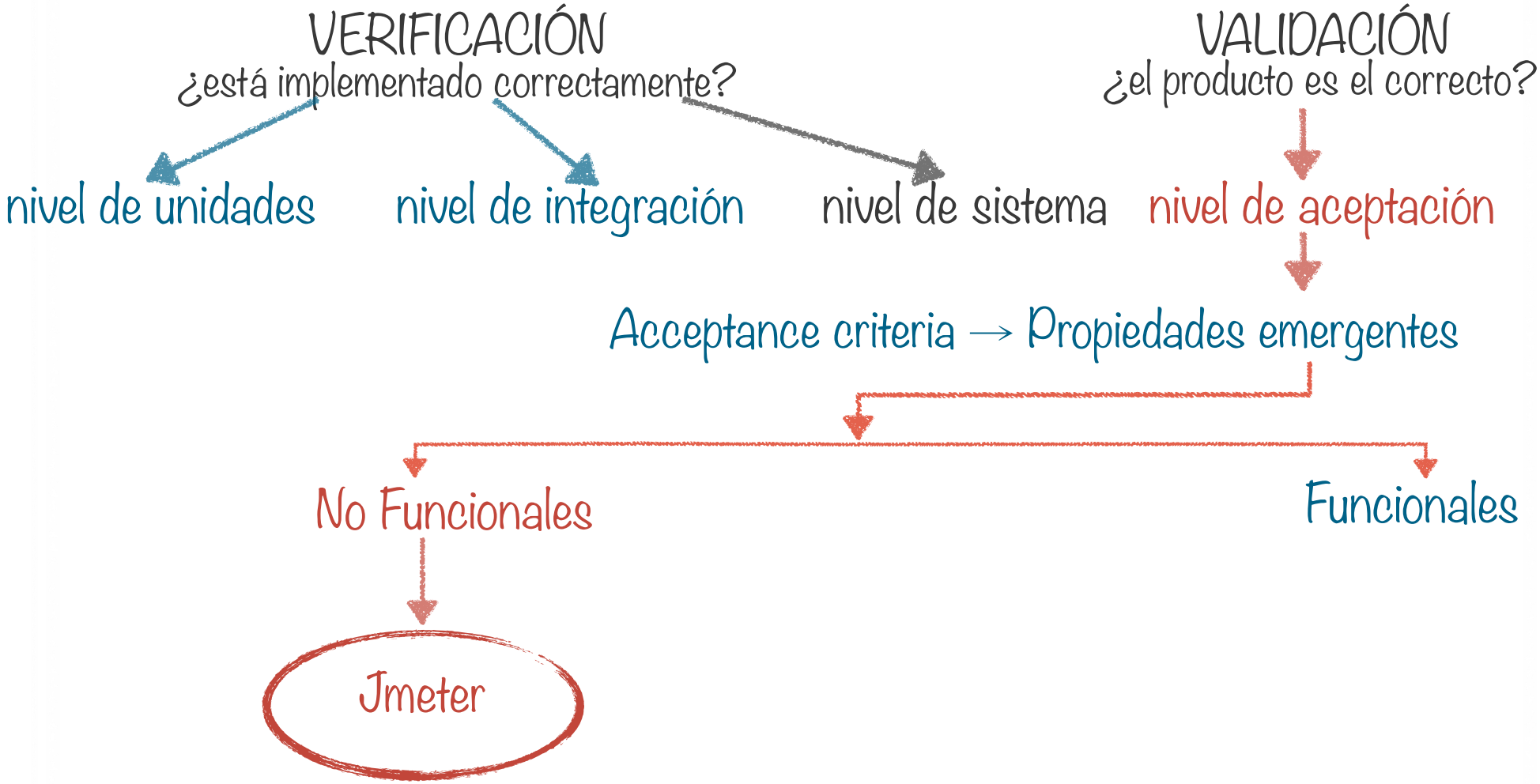
Sesión 10: Pruebas no funcionales

Propiedades emergentes no funcionales
El proceso de pruebas
Automatización de las pruebas: JMeter

María Isabel Alfonso Galipienso
Universidad de Alicante eli@ua.es

Niveles de pruebas

- Las pruebas se realizan a diferentes niveles, durante el proceso de desarrollo del software



Propiedades emergentes no funcionales

- Hay dos tipos de propiedades emergentes:
  - Funcionales: describen lo **que** el sistema hace, o debería hacer (*does view*)
  - No funcionales: hacen referencia a "how to well a system perfom its functional requirements"
- Muchas de las propiedades emergentes NO FUNCIONALES se categorizan como "-ilidades":
  - fiabilidad: probabilidad de funcionamiento sin fallos durante un tiempo determinado en un entorno específico
  - disponibilidad: tiempo durante el cual el sistema proporciona servicio al usuario. Suele expresarse como: hh/dd (p.ej 24/7: 24 horas al día, 7 días por semana)
  - mantenibilidad: capacidad de un sistema para soportar cambios. Hay tres tipos de cambios: correctivos, adaptativos y perfectivos.
  - escalabilidad: hace referencia a la capacidad de mantener el tiempo de respuesta ante cambios en el número de usuarios que utilizan el sistema.

Algunas métricas utilizadas

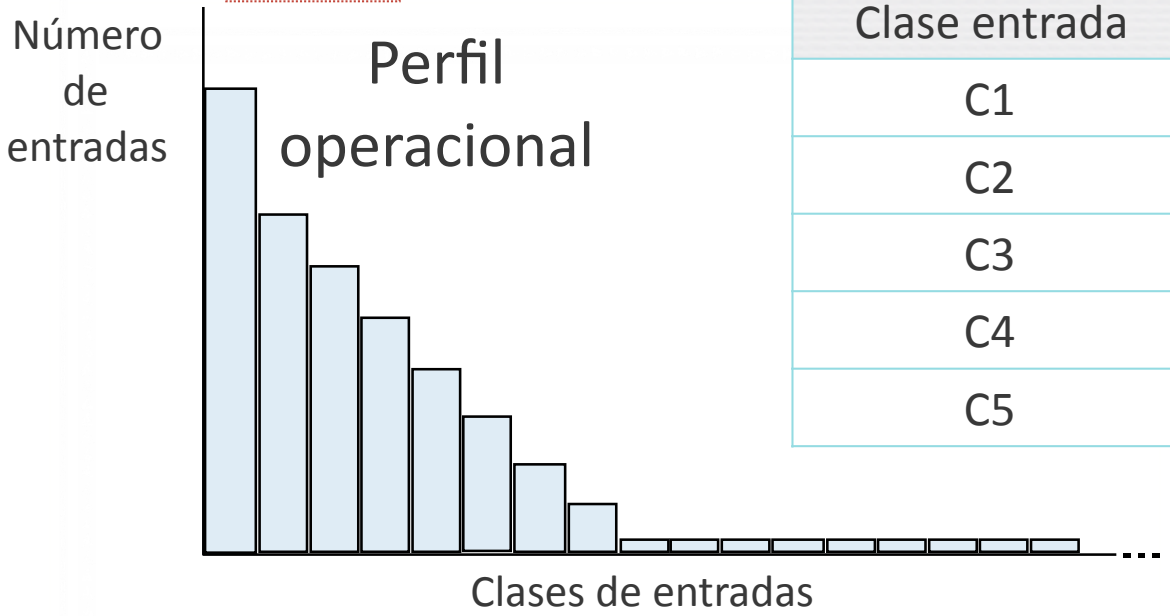
- Los criterios de aceptación deben incluir propiedades emergentes "cuantificables"
- Para juzgar en qué grado se satisfacen los criterios de aceptación se utilizan diferentes métricas:
  - Para estimar la fiabilidad se utilizan pruebas aleatorias basándonos en un perfil operacional. Se utilizan las métricas MTTF (Mean Time To Failure), MTTR(Mean Time To Repair), y MTBF=MTTF+MTTR (MTBF: Mean time between failures)
  - Para estimar la disponbilidad se utiliza la métrica MTTR para medir el "downtime" del sistema. La idea es incluir medidas para minimizar el MTTR
  - Para estimar la mantenibilidad se utiliza la métrica MTTR (que refleja el tiempo consumido en analizar un defecto correctivo, diseñar la modificación, implementar el cambio, probarlo y distribuirlo)
  - La escalabilidad del sistema utiliza el número de transacciones (operaciones) por unidad de tiempo. Los sistemas suelen poder incrementar su escalabilidad siempre y cuando no sobrepasen limitaciones de almacenamiento de datos, ancho de banda o velocidad de procesador.

Ejemplos de pruebas

- Responsiveness, fiabilidad, carga... son ejemplos de propiedades emergentes no funcionales. Todas ellas influyen en el **rendimiento** del sistema.
- Las pruebas de stress consisten en "forzar" peticiones al sistema por encima del límite del diseño del software. Por ejemplo si el sistema se ha diseñado para permitir hasta 300 transacciones por segundo, comenzaremos por hacer pruebas con una carga de peticiones inferior a 300 e incrementaremos gradualmente la carga hasta sobrepasar los 300 y ver cuándo falla el sistema
- Para evaluar la fiabilidad de un sistema podemos utilizar lo que se denominan **pruebas estadísticas**, que consisten en:
  - construir un "perfil operacional" (*operational profile*), que refleje el uso real del sistema (patrón de entradas). Como resultado se identifican las "clases" de entradas y la probabilidad de ocurrencia asumiendo un uso "normal"
  - generar un conjunto de datos de prueba que reflejen dicho perfil operacional
  - probar dichos datos midiendo el número de fallos y el tiempo entre fallos, calculando la fiabilidad del sistema después de observar un número de fallos estadísticamente significativo

Ejemplo de generación de pruebas estadísticas

Paso 1



Paso 4

Clase entrada	Distrib. Probab.	Intervalo
C1	50 %	1-49
C2	15 %	50-63
C3	15 %	64-78
C4	15 %	79-94
C5	5 %	95-99

Paso 2

Se generan números aleatorios entre 1 y 99, por ejemplo:

13-94-22-24-45-56-81-19-31-69-45-9-38-21-52-84-86-97-...

Paso 3

Se derivan casos de prueba según su distribución de probabilidad:

C1-C4-C1-C1-C1-C2-C4-C1-C1-C1-C3-C1-C1-C1-C1-C2-C4-C4-C5-...

Resumen del proceso de pruebas

- Ya hemos visto que, en general, las propiedades emergentes no funcionales determinan el rendimiento rendimiento de nuestra aplicación
- Para evaluar el rendimiento, necesitamos realizar las siguientes actividades:
  1. Identificar los criterios de aceptación: identificar y cuantificar las propiedades emergentes no funcionales que determinan cuál es el rendimiento aceptable para nuestra aplicación (p.e. tiempos de respuesta, fiabilidad, utilización de recursos...)
  2. Diseñar los tests: deberemos conocer el patrón de uso de la aplicación (perfil operacional) para que nuestros casos de prueba estén basados en en escenarios reales de nuestra aplicación
  3. Preparar el entorno de pruebas: es importante que el entorno de pruebas sea lo más realista posible
  4. Automatizar las pruebas: utilizando alguna herramienta software, "grabaremos" los escenarios de prueba, y ejecutaremos los tests
  5. Analizaremos los resultados y realizaremos los cambios oportunos para conseguir nuestro objetivo

Pruebas de carga y stress

- Las pruebas de carga y stress forman parte de las pruebas para comprobar que el sistema cumple con los requisitos no funcionales
- Las pruebas de carga validan el rendimiento de un sistema en términos de "tratar un número específico de usuarios manteniendo un ratio de transacciones" (p.ej. "una petición del sistema se debe tratar en menos de 2 segundos cuando existen 10000 usuarios dentro del sistema")
  - El rendimiento se fija mediante el ratio de número de transacciones por número de usuarios
  - Hay que tener mucho cuidado con criterios de rendimiento ambiguos, como "Las peticiones se tienen que servir en un tiempo razonable". Dichas sentencias son imposibles de cuantificar y por lo tanto, imposibles de probar con precisión
- Las pruebas de stress comprueban la confiabilidad y robustez del sistema cuando se supera la carga normal
  - confiabilidad = fiabilidad
  - robustez: capacidad de recuperación del sistema ante entradas erróneas u otros fallos

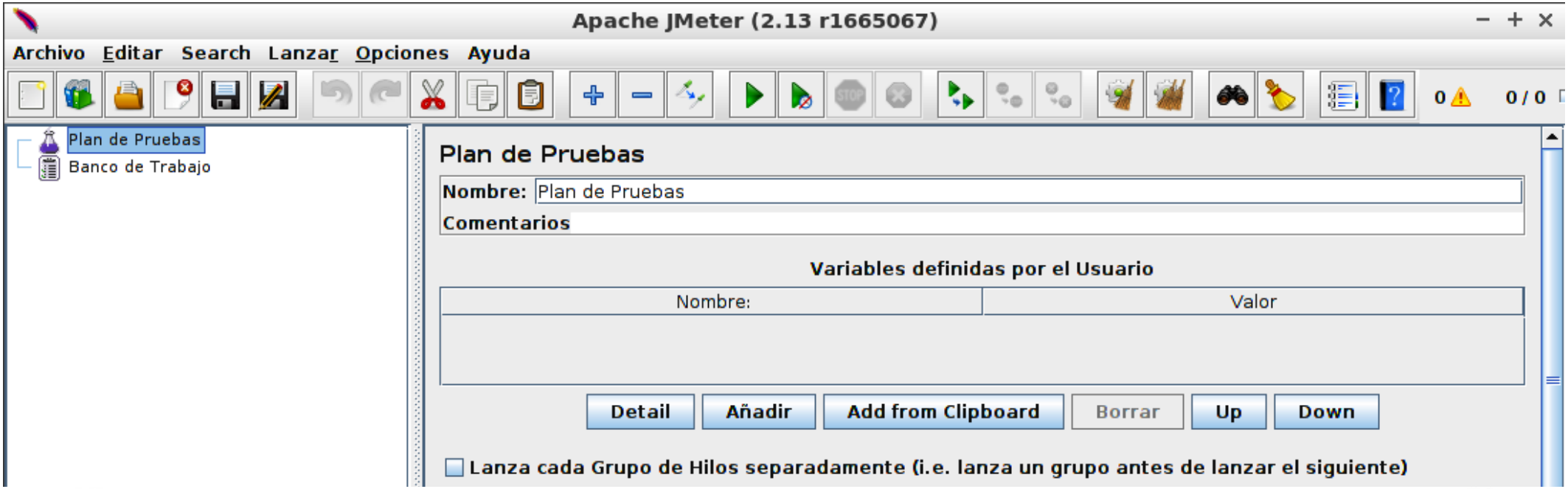
Consideraciones sobre el Rendimiento

- ¡¡¡ No hay que dejar las pruebas de rendimiento para el final del proyecto !!!
- Posibles fallos relacionados con el rendimiento pueden conllevar el retocar mucho código
  - Los esfuerzos de los últimos minutos ponen la calidad y estabilidad de la aplicación en un riesgo alto e innecesario.
- Evitamos el peligro mediante una buena estrategia de pruebas combinada con una arquitectura software que considere el rendimiento desde el inicio del desarrollo
  - Las iteraciones iniciales del proyecto son para construir prototipos exploratorios y comprobar todos los requisitos no funcionales.

JMeter

Ver http://jmeter.apache.org/usermanual/index.html

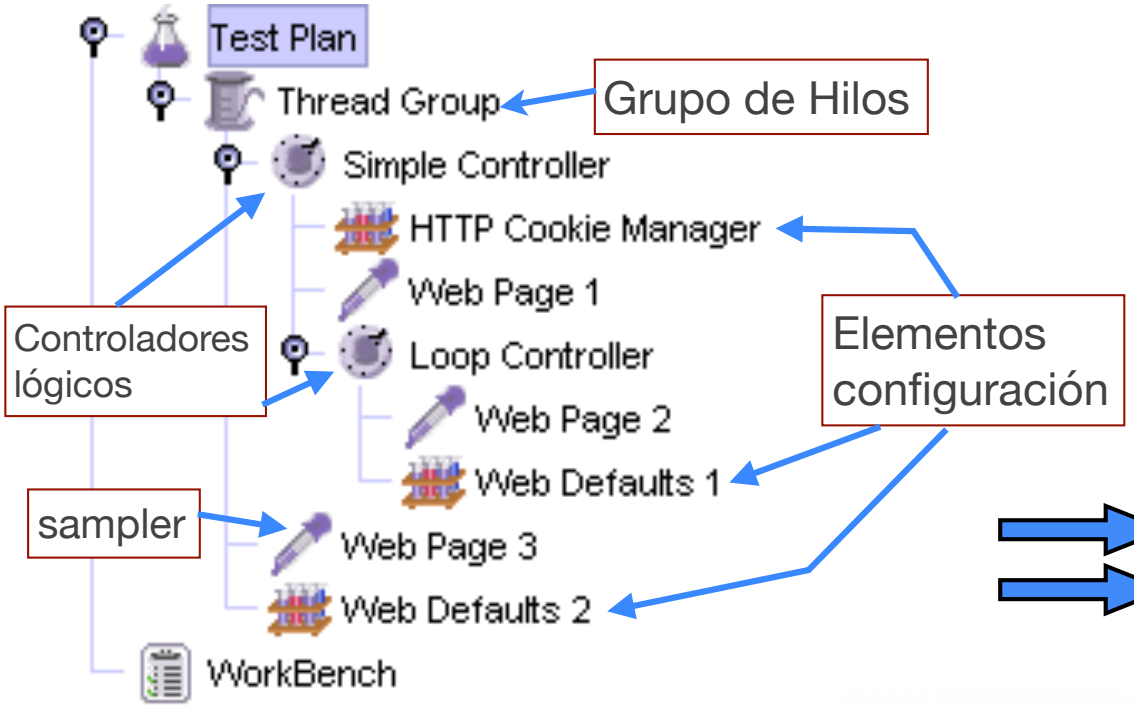
- Apache JMeter (http://jmeter.apache.org/) es una herramienta de escritorio 100% Java diseñada para medir el rendimiento mediante pruebas de carga
- Permite múltiples hilos de ejecución concurrente, procesando diversos y diferentes patrones de petición
  - JMeter permite trabajar con muchos tipos distintos de aplicaciones. Para cada una de ellas proporciona un *Sampler* o *Muestreador*, para hacer las correspondientes peticiones



JMeter: Plan de pruebas (I)

- Un plan de pruebas JMeter describe una serie de "pasos" (acciones) que JMeter realizará cuando se ejecute el plan
- Un plan de pruebas está formado por por:
  - Uno o más grupos de hilos (*Thread Groups*)
  - Controladores lógicos (*Logic Controllers*)
  - *Samplers*, *Listeners*, *Timers*, *Assertions*, *Pre-Processors*, *Post-Processors* y
  - Elementos de Configuracion (*Configuration Elements*)

Nos centraremos en los elementos que hemos marcado con una flecha azul

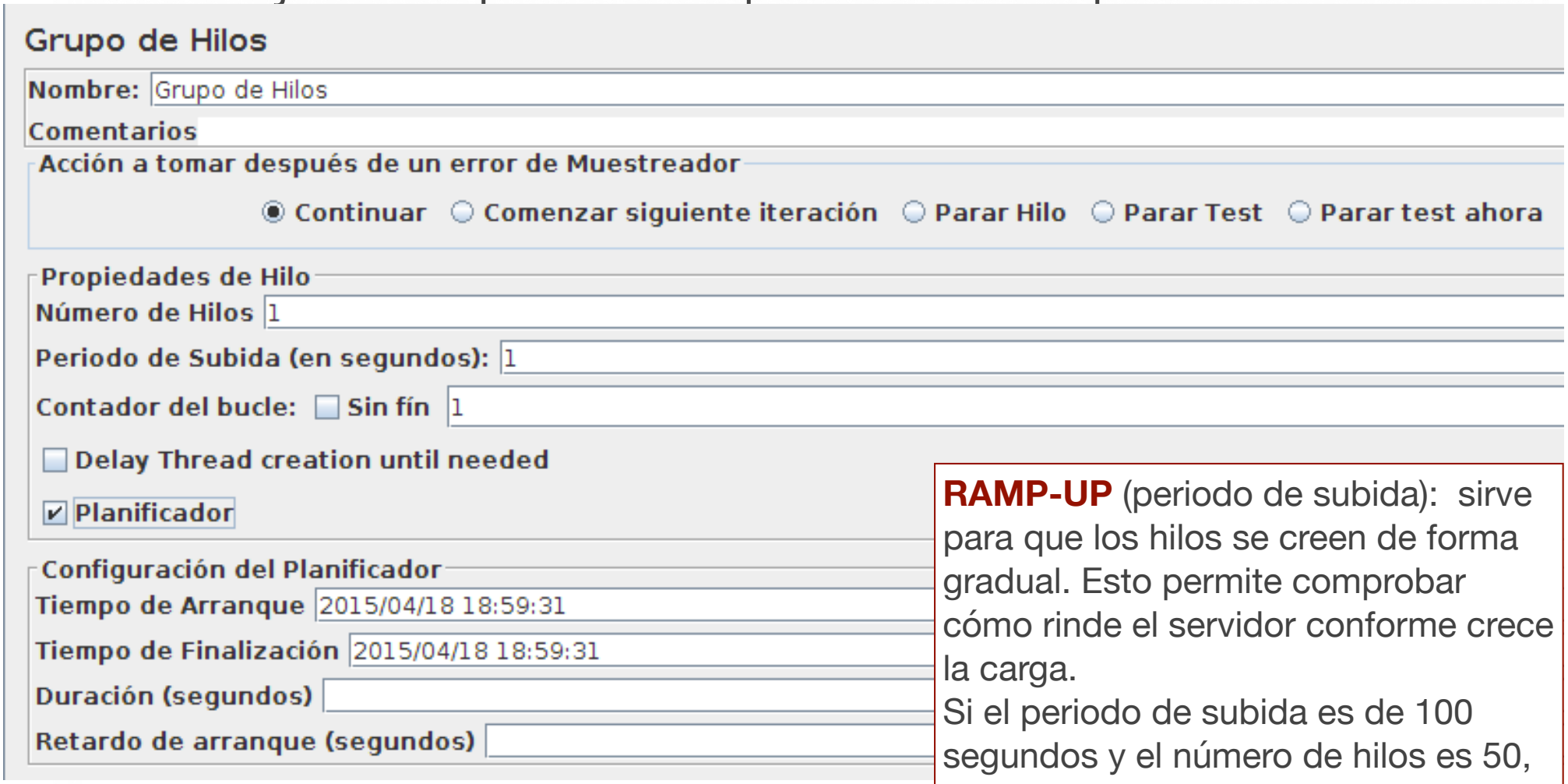


El orden de ejecución de los elementos de un plan es el siguiente:

1. Configuration elements
2. Pre-Processors
3. Timers
4. Sampler
5. Post-Processors (unless SampleResult is null)
6. Assertions (unless SampleResult is null)
7. Listeners (unless SampleResult is null)

JMeter: Hilos de ejecución

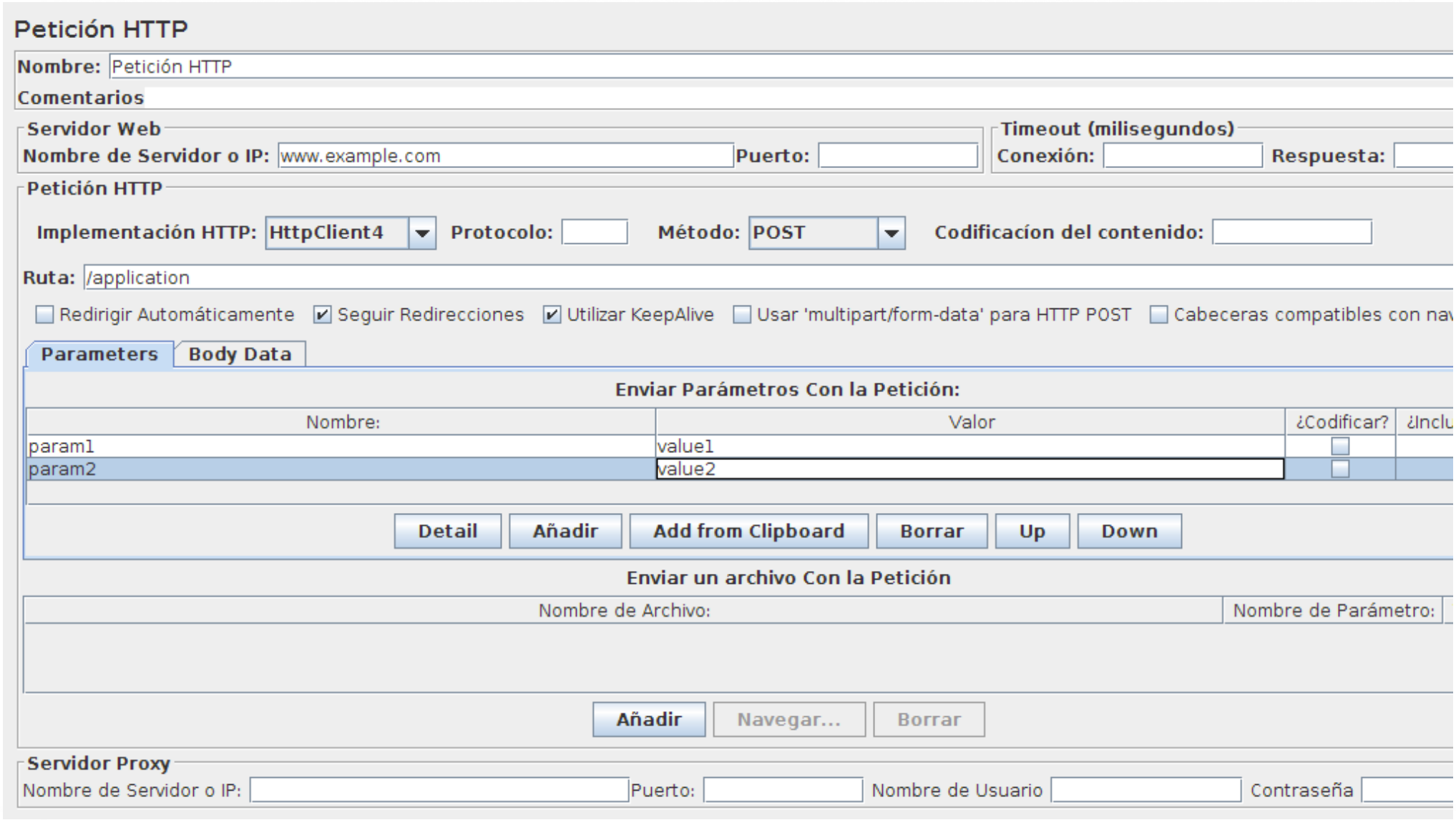
- Un hilo de ejecución es el punto de partida de cualquier plan de pruebas
- Cada hilo ejecuta completamente el plan de forma independiente de otros hilos



RAMP-UP (periodo de subida): sirve para que los hilos se creen de forma gradual. Esto permite comprobar cómo rinde el servidor conforme crece la carga.
Si el periodo de subida es de 100 segundos y el número de hilos es 50, significa que el servidor tardará 100 segundos en crear los 50 hilos, es decir, un nuevo hilo cada 2 segundos

JMeter: Samplers

- Los Samplers (muestreadores) envían peticiones a un servidor. Ejemplos de samplers:HTTP request, FTP request, HDBC Request,... Se ejecutan en el orden en al que aparecen en el árbol



JMeter: Controladores lógicos

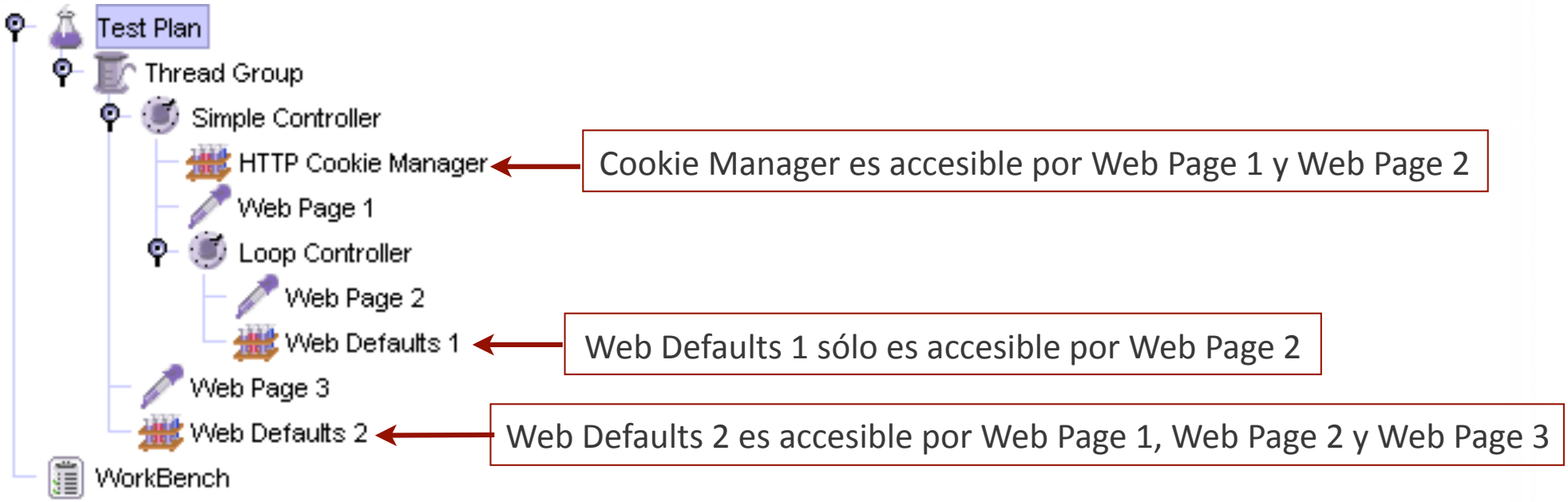
- Determinan la lógica que JMeter utiliza para decidir cuándo enviar las peticiones (orquestan el flujo de control). Actúan sobre sus elementos hijo
- Ejemplos de controladores
  - Simple controller: No tiene efecto sobre cómo procesa JMeter los elementos hijos del controlador. Simplemente sirve para "agrupar" dichos elementos.
  - Loop controller: Itera sobre sus elementos hijos un cierto número de veces
  - Only once controller: Indica a JMeter que sus elementos hijos deben ser procesados UNA ÚNICA vez en el plan de pruebas
  - Interleave controller: ejecutará uno de sus subcontroladores o samplers en cada iteración del bucle de pruebas, alterándose secuencialmente a lo largo de la lista

Ejemplo: supongamos que el grupo de hilos está formado por 2 hilos, y 5 iteraciones

Loop Iteration	Each JMeter Thread Sends These HTTP Requests
1	News Page
1	Log Page
2	FAQ Page
2	Log Page
3	Gump Page
3	Log Page
4	Because there are no more requests in the controller, JMeter starts over and sends the first HTTP Request, which is the News Page.
4	Log Page
5	FAQ Page
5	Log Page

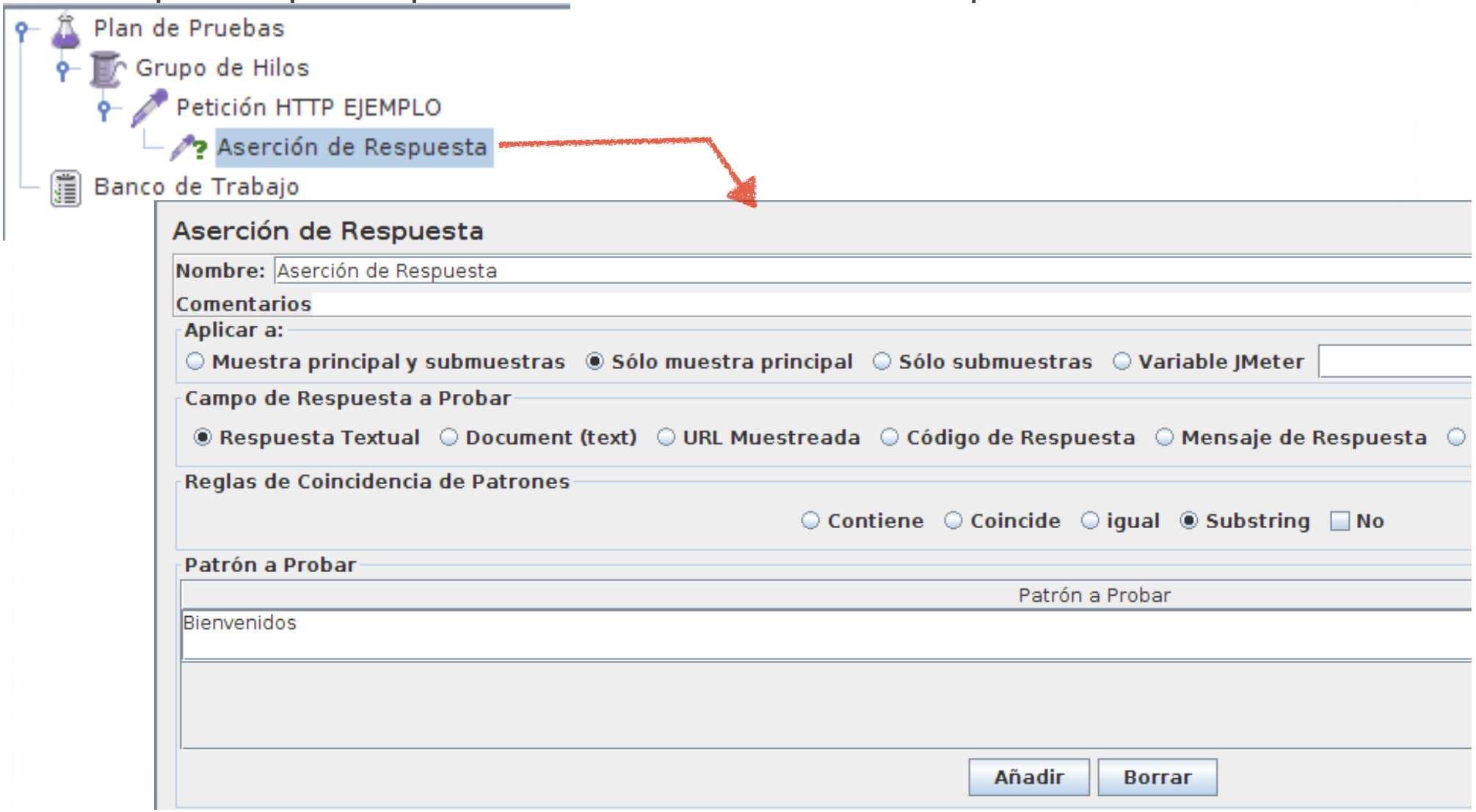
JMeter: Elementos de configuración

- "Trabajan" conjuntamente con un sampler. Si bien no realizan peticiones (excepto HTTP Proxy Server: es un elemento solamente disponible para el banco de trabajo), pueden modificar las mismas a través de los atributos correspondientes
- Un elemento de configuración es accesible sólo dentro de la rama del árbol (y sub-ramas) en la que se sitúa el elemento
- Un elemento de configuración dentro de una rama del árbol tiene mayor preferencia que el mismo elemento (mismo tipo de elemento) en una rama padre



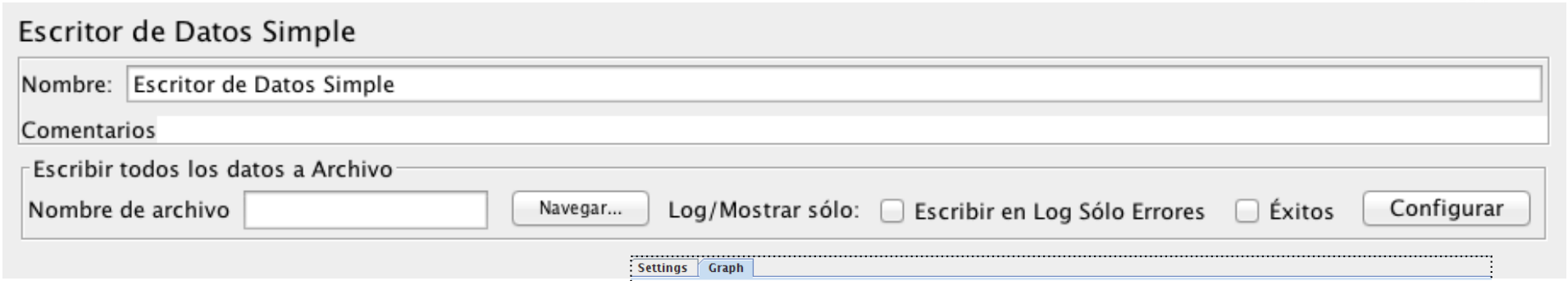
JMeter: Aserciones

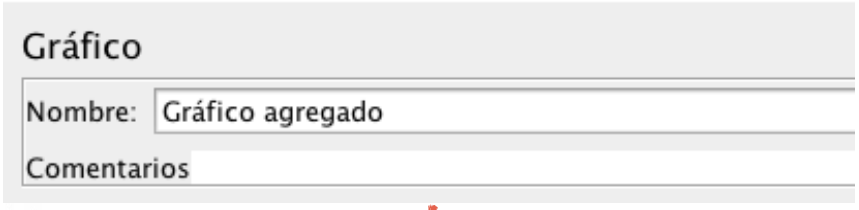
- Las aserciones permiten hacer afirmaciones sobre las respuestas recibidas del servidor que se está probando. Podemos añadir aserciones a cualquier sampler. Se trata de probar que la aplicación devuelve el resultado esperado

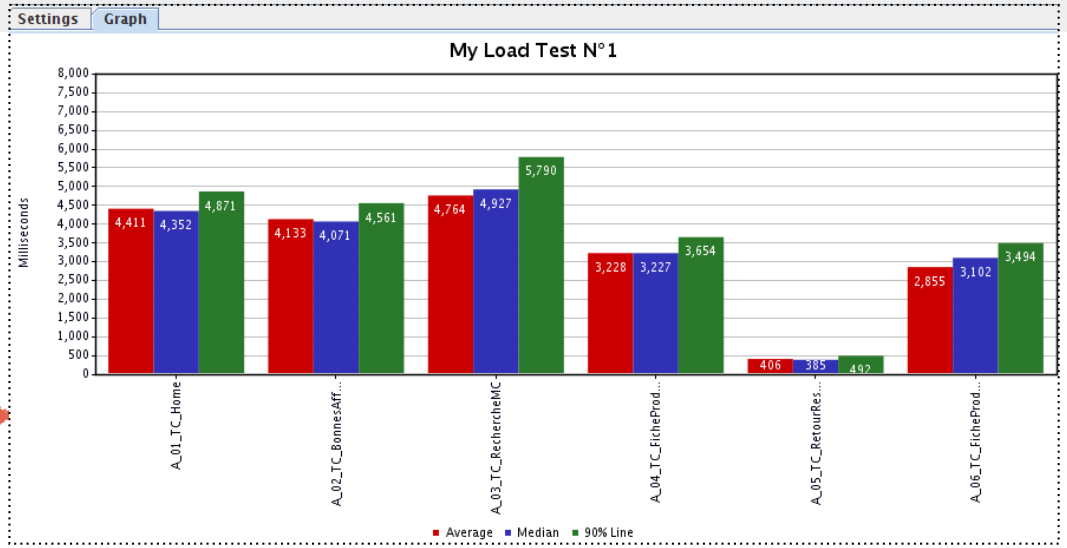


JMeter: Listeners (I)

- Los listeners (receptores) se utilizan para ver y/o almacenar en el disco los resultados de las peticiones realizadas. Proporcionan acceso a la información que JMeter va acumulando sobre los casos de prueba a medida que se ejecutan los tests
  - TODOS los *listeners* guardan los MISMOS datos; la única diferencia es la forma en que presentan dichos datos en la pantalla
  - Ejemplos de *listeners*: Escritor de Datos Simple, Gráfico, Gráfico de resultados, Informe agregado, Reporte resumen, Response Time Graph, Resultados de la Aserción,...







JMeter: Listeners (II)

- Más ejemplos de listeners

Rendimiento = num_peticiones/segundo

Informe Agregado

Nombre: Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición HTTP Login	211	4	4	6	2	31	0,00%	61,9/sec	75,3
Petición HTTP Bibliotecario Registrado	211	6	6	9	2	33	100,00%	62,1/sec	95,8
TOTAL	422	5	6	8	2	33	50,00%	123,1/sec	169,9

Gráfico de Resultados

Nombre: Gráfico de Resultados

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

Gráficos a Mostrar: Datos | Media | Mediana | Desviación | Rendimiento

Sobre los datos registrados por JMeter

El tiempo se calcula en milisegundos

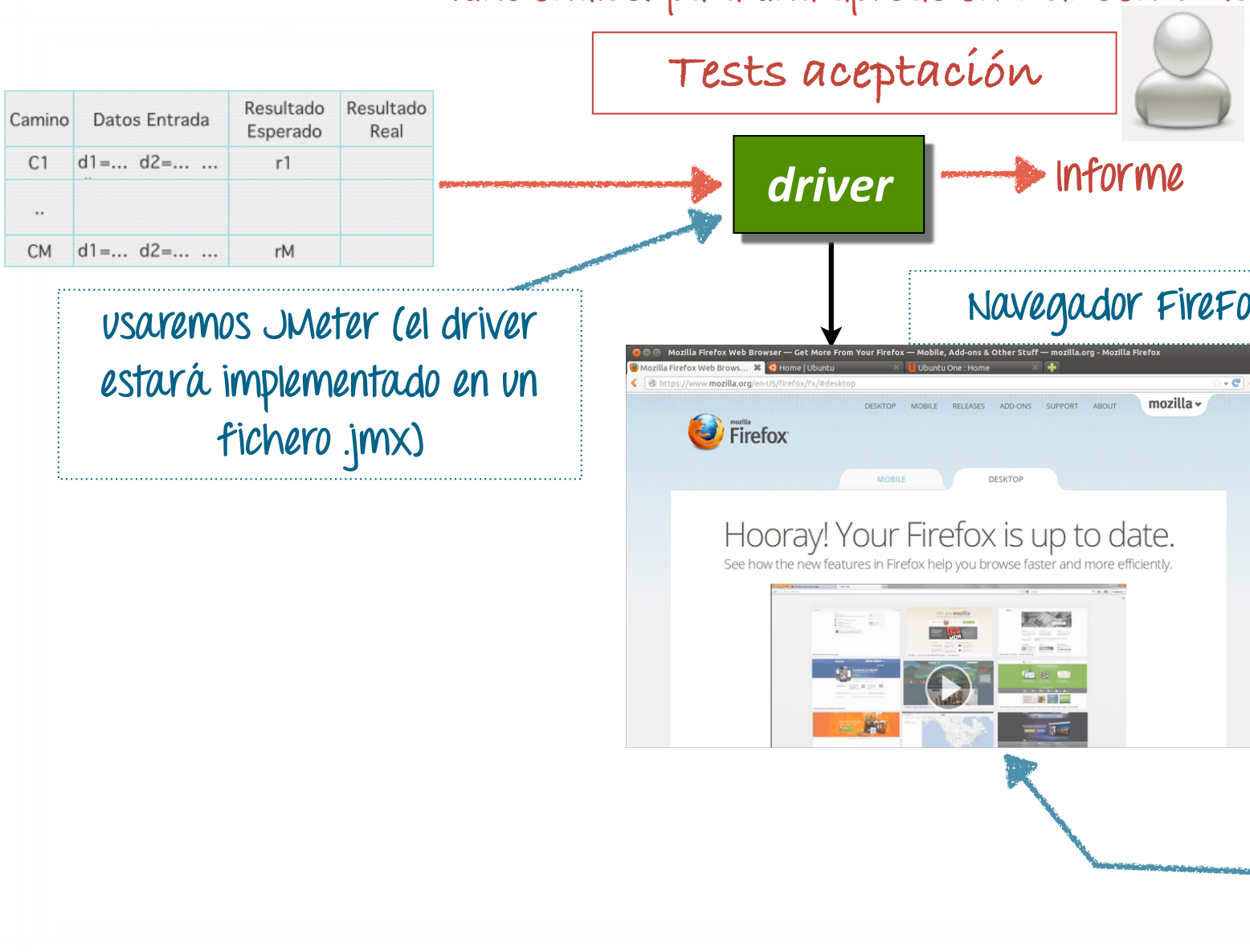
- Para cada muestra (sampler), JMeter calcula:
  - # Muestras - Número de muestras con la misma etiqueta
  - Media - Tiempo medio de respuesta (en milisegundos)
  - Mediana - The median is the time in the middle of a set of results. 50% of the samples took no more than this time; the remainder took at least as long.
  - Línea de 90% (percentil):90% of the samples took no more than this time. The remaining samples took at least as long as this
  - Min - Tiempo mínimo de respuesta para las muestras con la misma etiqueta
  - Max - Tiempo máximo de respuesta para las muestras con la misma etiqueta
  - % Error - Porcentaje de peticiones con errores
  - Rendimiento (Throughput) - número de peticiones por segundo/minuto/ hora. La unidad de tiempo se elige en función de que el valor visualizado sea como mínimo 1.
  - Kb/sec - rendimiento expresado en Kilobytes por segundo

Consejos JMeter

- Utiliza escenarios de prueba significativos, y construye planes de prueba que prueben situaciones representativas del mundo real
  - Los casos de uso ofrecen un punto de partida ideal
- Asegúrate de ejecutar JMeter en una máquina distinta a la del sistema a probar.
  - Esto previene que JMeter afecte sobre los resultados de las pruebas
- El proceso de pruebas es un proceso científico. Todas las pruebas se deben realizar bajo condiciones completamente controladas
  - Si estas trabajando con un servidor compartido, primero comprueba que nadie más esta realizando pruebas de carga contra la misma aplicación web.
- Asegúrate de que dispones de ancho de banda en la estación que ejecuta JMeter
  - La idea es probar el rendimiento de la aplicación y el servidor, y no la conexión de la red.
- Utiliza diferentes instancias de JMeter ejecutándose en diferentes máquinas para añadir carga adicional al servidor
  - Esta configuración suele ser necesaria para realizar pruebas de stress. JMeter puede controlar las instancias JMeter de las otras máquinas y coordinar la prueba
- Deja una prueba JMeter ejecutarse durante largos periodos de tiempo, posiblemente varios días o semanas
  - Estarás probando la disponibilidad del sistema y resaltando las posibles degradaciones en el rendimiento del servidor debido a una mala gestión de los recursos

Y ahora vamos al laboratorio…

Vamos a implementar tests de aceptación (para validar propiedades emergentes NO funcionales) para una aplicación web con JMeter



Referencias bibliográficas

- Software testing and quality assurance. Kshirasagar Naik & Priyadarshi Tripathy. Wiley. 2008
  - Capítulos 8 y 15
- Página oficial JMeter:
  - http://jmeter.apache.org/usermanual/index.html