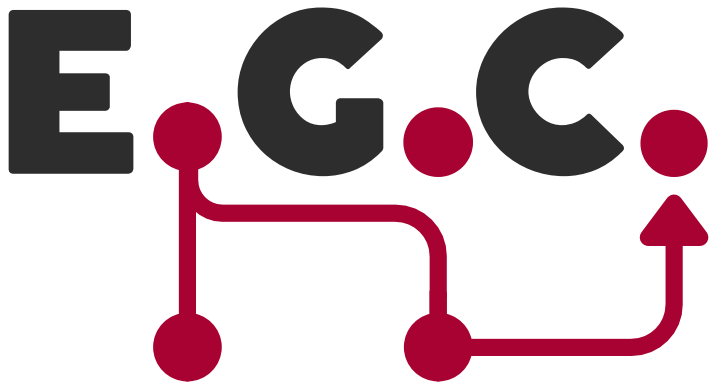


Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración

Universidad de Sevilla 1505

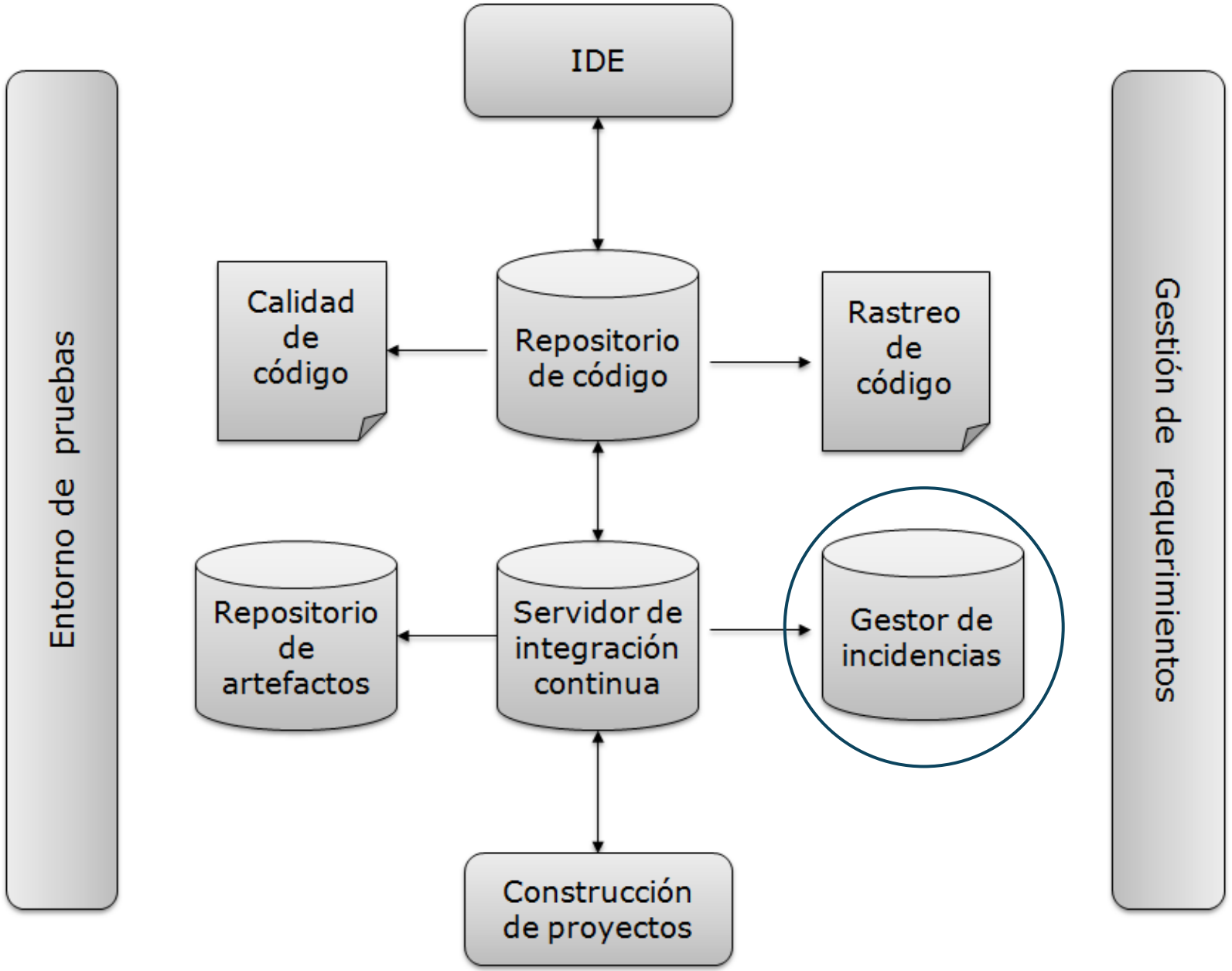
Escuela Técnica Superior de Ingeniería Informática

Gestión de tareas e incidencias en el ciclo de integración continua

Objetivo principal:
Saber cómo organizar las tareas relacionadas con los *work-items* y las incidencias

1. **Conceptos básicos**
2. Tipos de cambio
3. Depuración
4. Resumen
5. Bibliografía

1. Conceptos básicos | ecosistemas de desarrollo



1. Conceptos básicos | proceso de petición de cambios

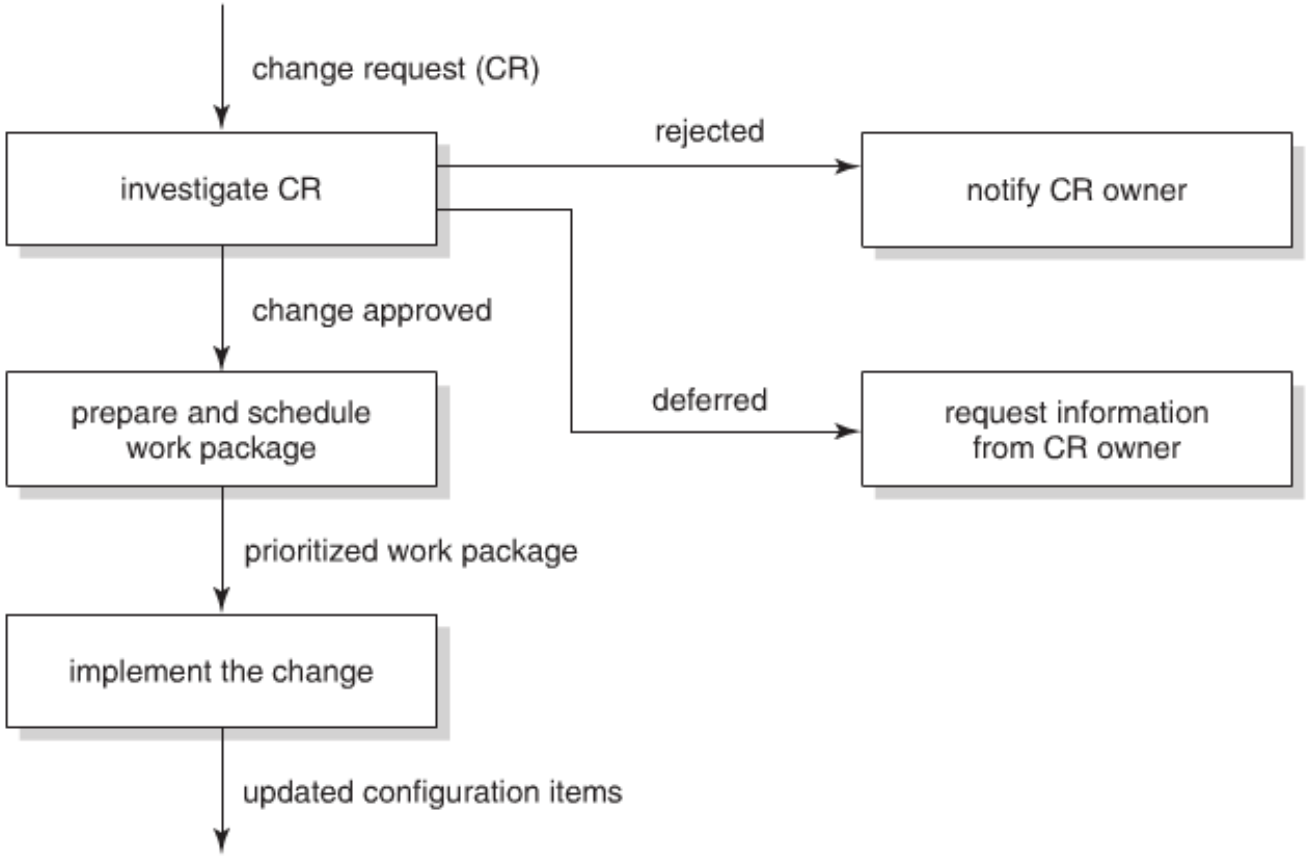
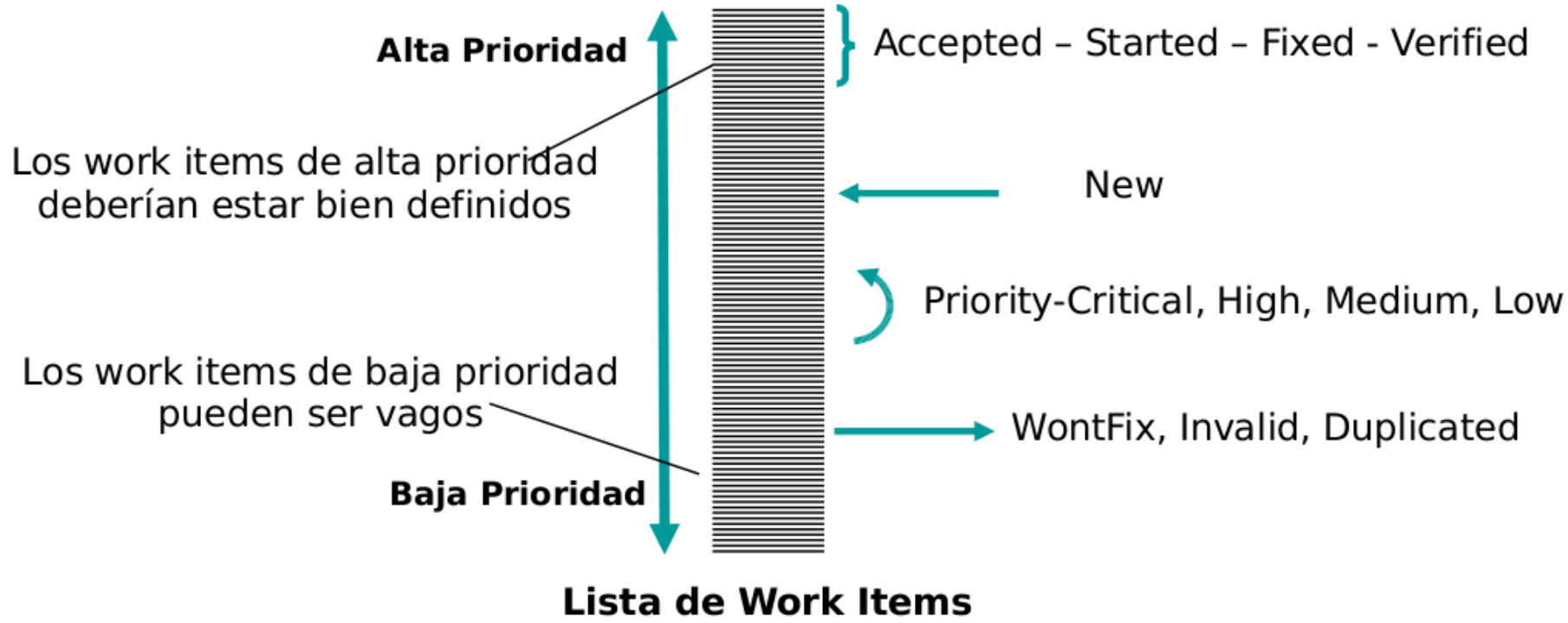
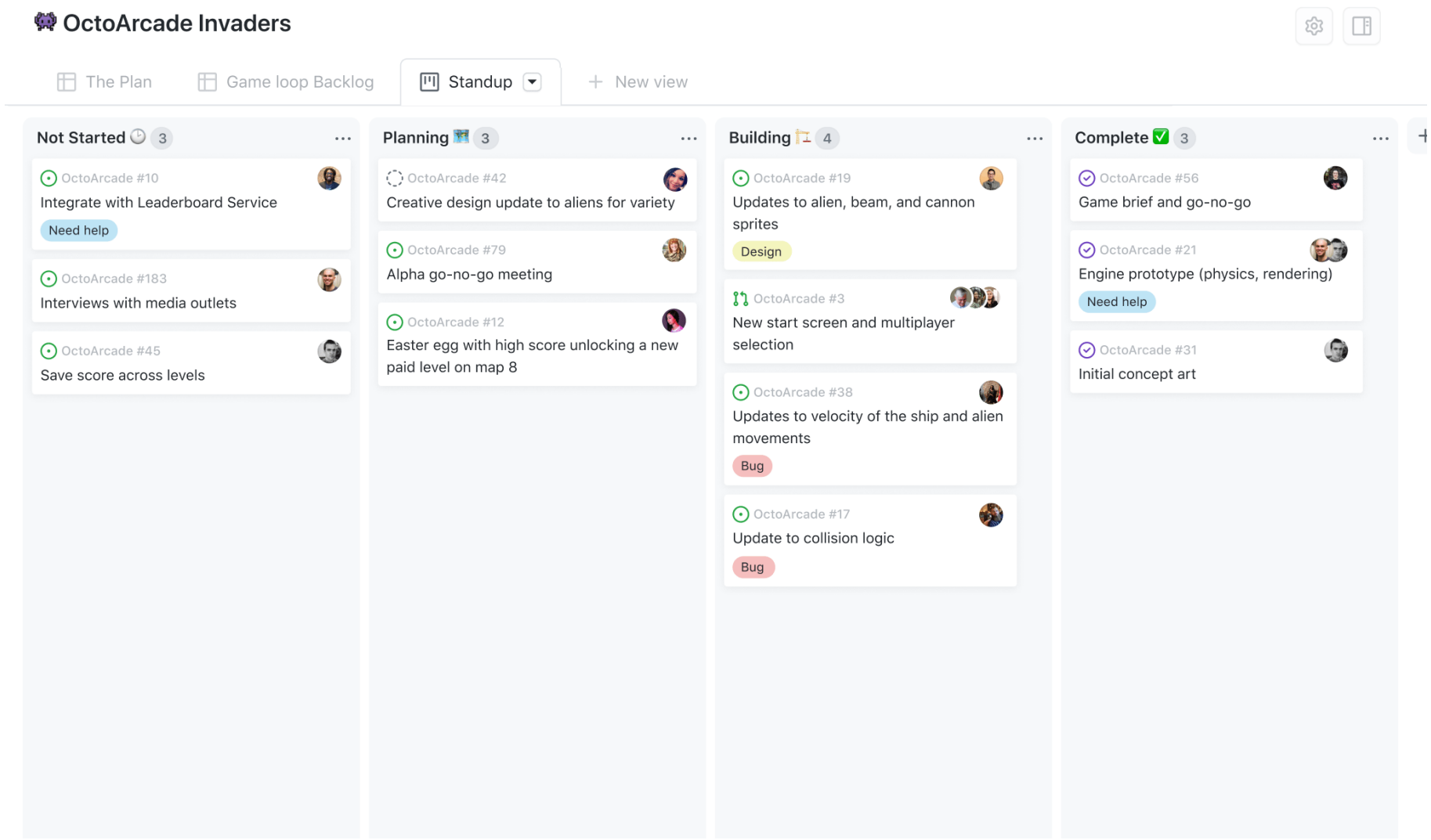


Figure 4.1 Workflow of a change request

1. Conceptos básicos | proceso de petición de cambios



1. Conceptos básicos | ejemplo de tablero



1. Conceptos básicos | lo básico

Todo proceso de gestión del cambio debe tener al menos tener:

- Prioridad del cambio
- Estado en la que se encuentra
- Tipo de cambio
- Roles en la gestión de dicho cambio

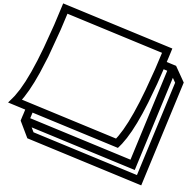
1. Conceptos básicos
2. **Tipos de cambio**
3. Depuración
4. Resumen
5. Bibliografía

2. Tipos de cambio | posibles taxonomías

- Por quien los origina
- Por el impacto
- Por la intención
- Por el tiempo

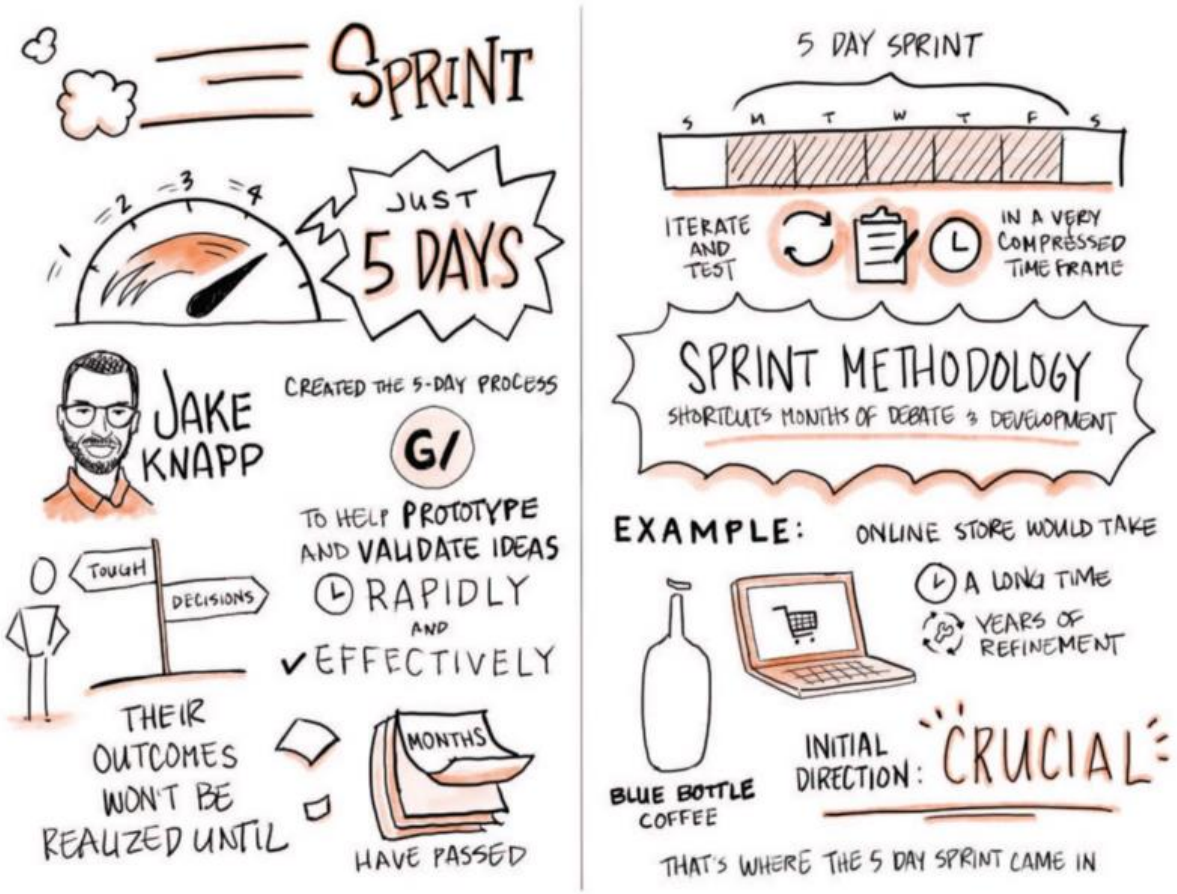
2. Tipos de cambio | nuestra taxonomía

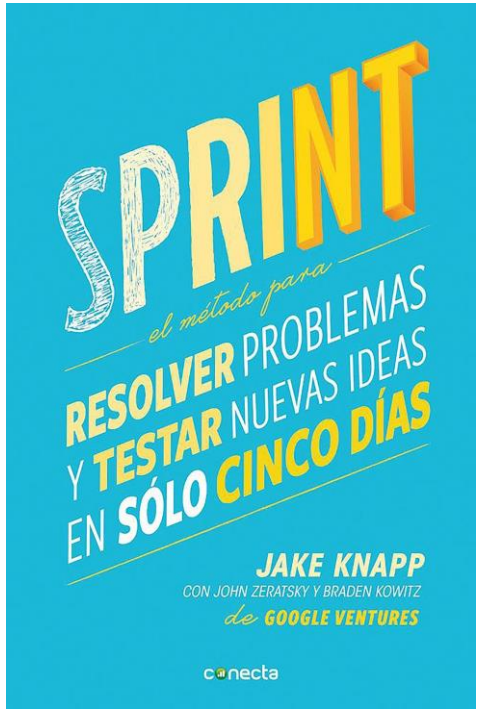
Work items y sus derivados



Incidencias

2. Tipos de cambio | Work ítems (WIs)





2. Tipos de cambio | Design sprint

day 1

understand

- who are the users
- what are their needs
- what is the context
- competitor review
- formulate strategy

2

diverge

- envision
- develop lots of solutions
- ideate

3

decide

- choose the best idea
- storyboard the idea

4

prototype

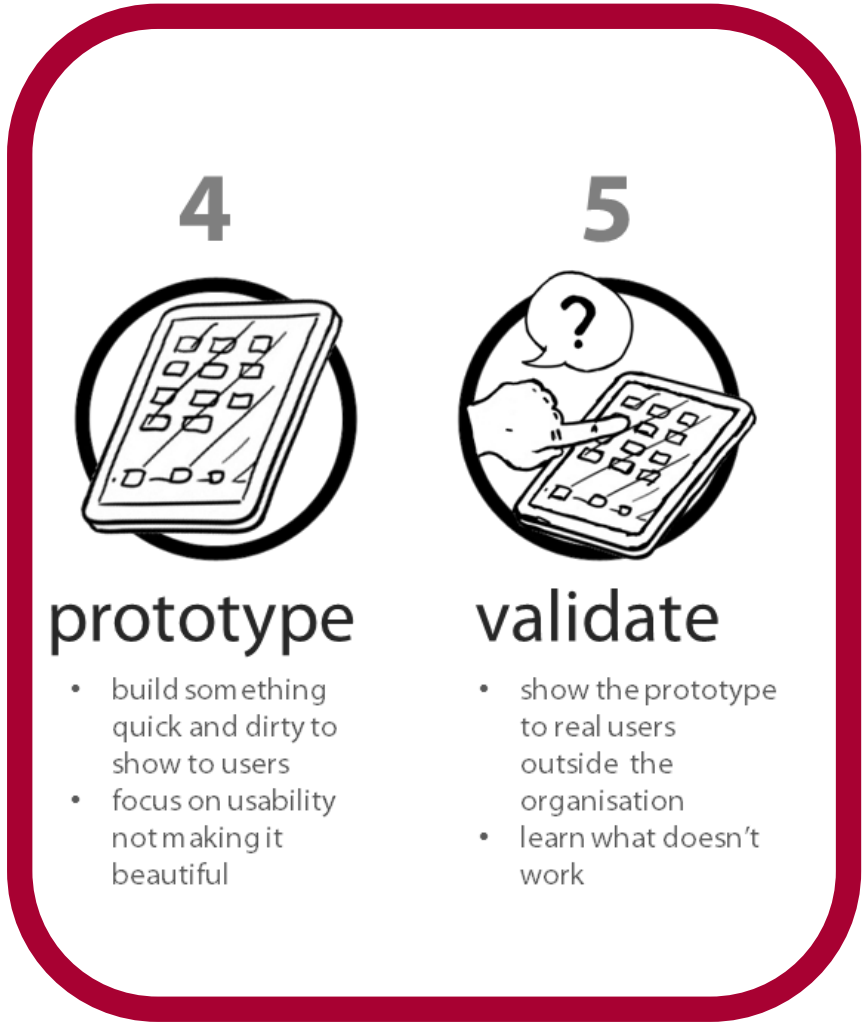
- build something quick and dirty to show to users
- focus on usability not making it beautiful

5

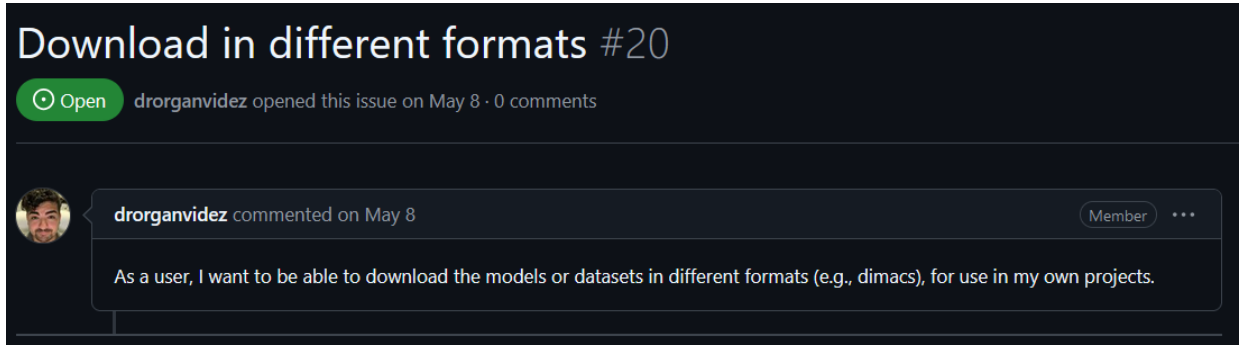
validate

- show the prototype to real users outside the organisation
- learn what doesn't work

2. Tipos de cambio | Prototipado realista

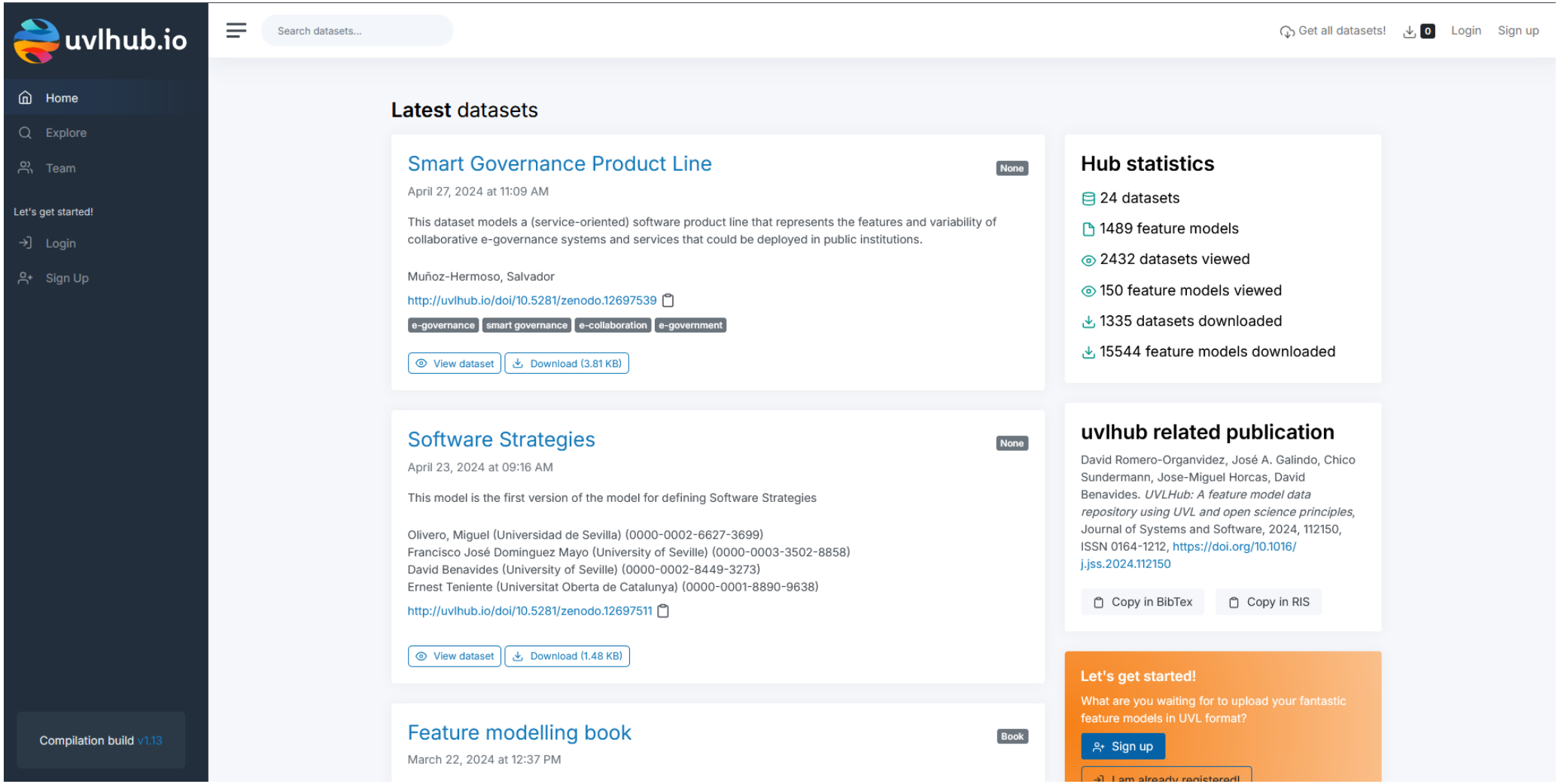


Download in different formats

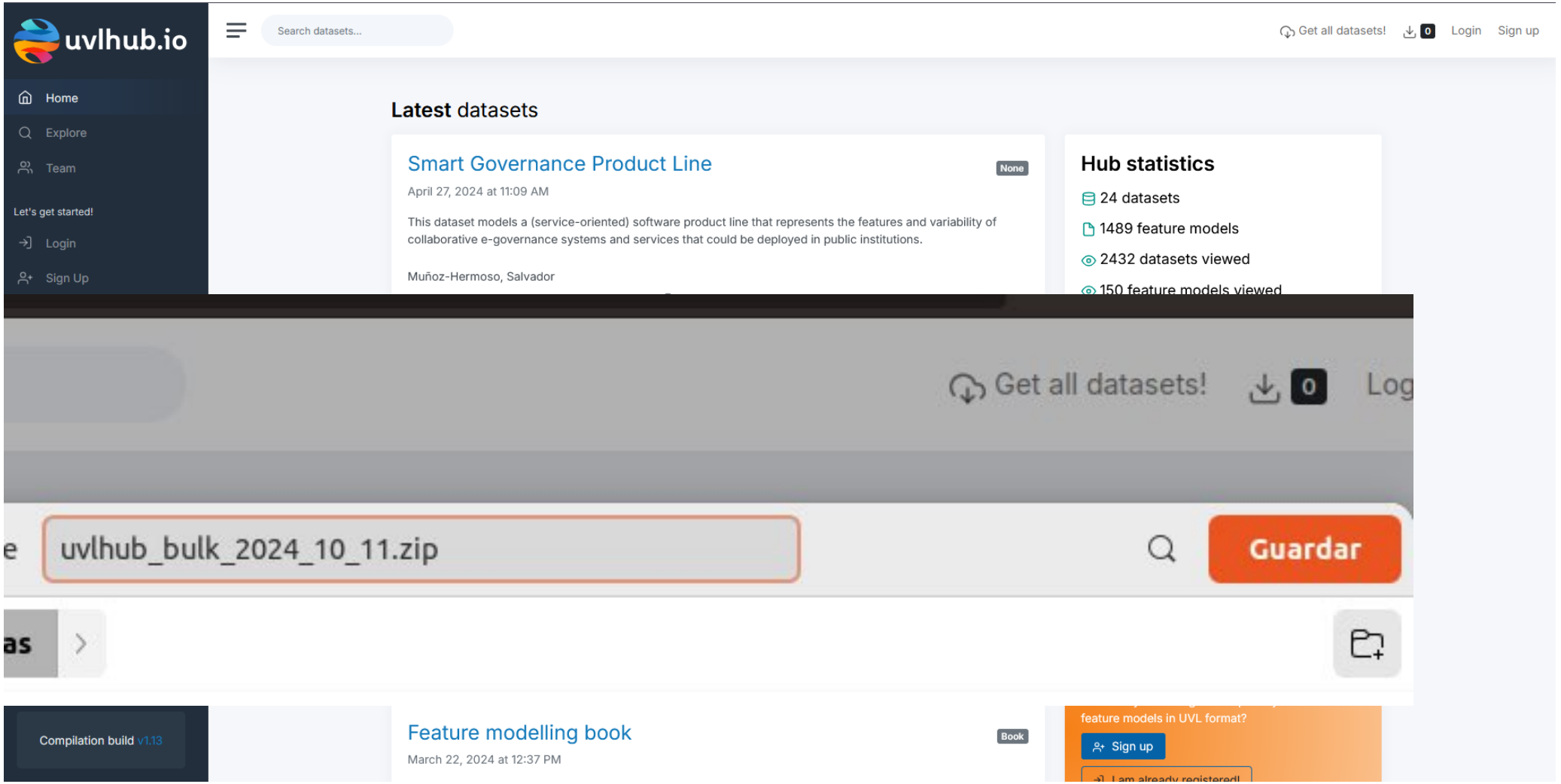


M1

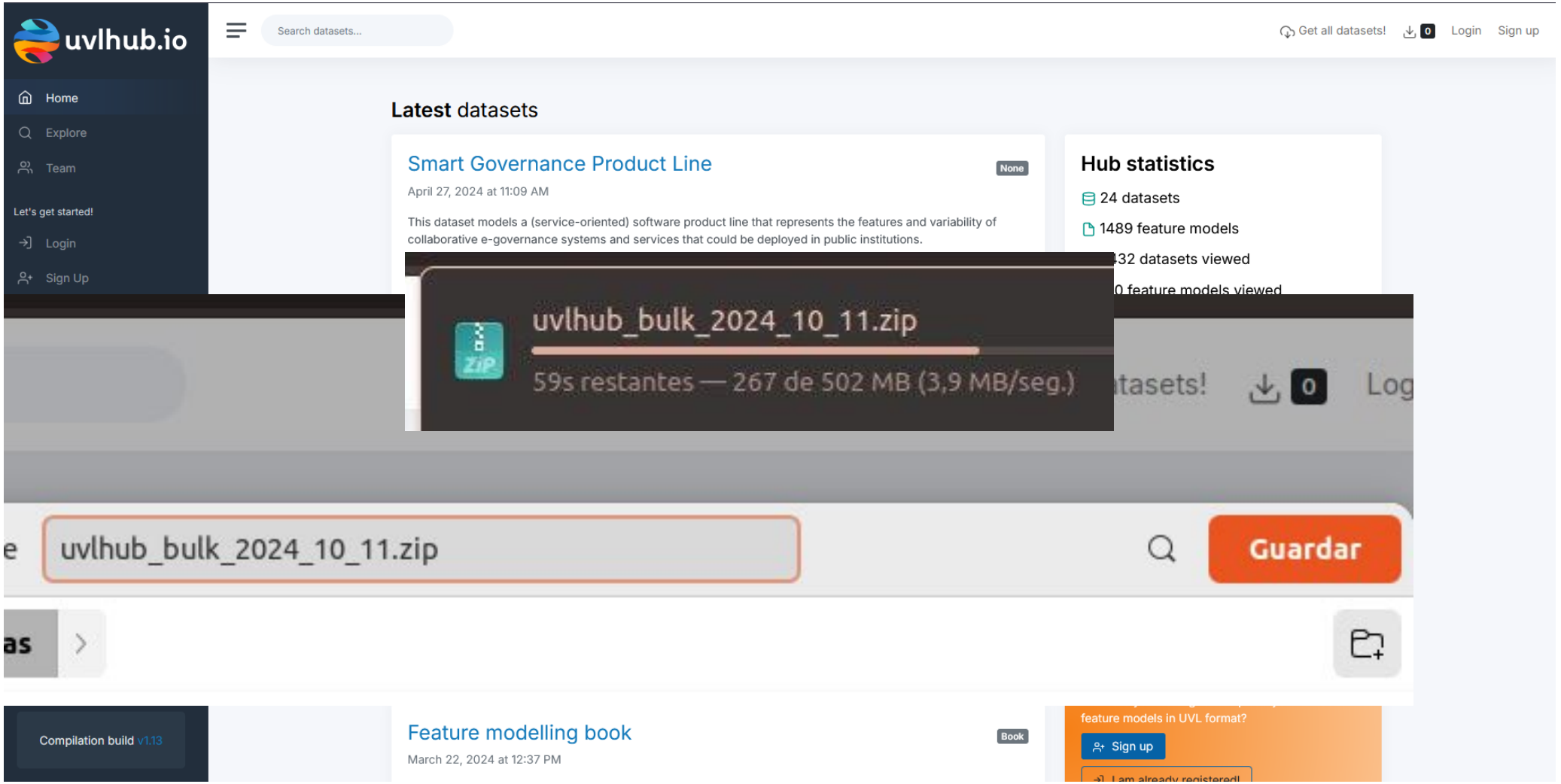
2. Tipos de cambio | ejemplo



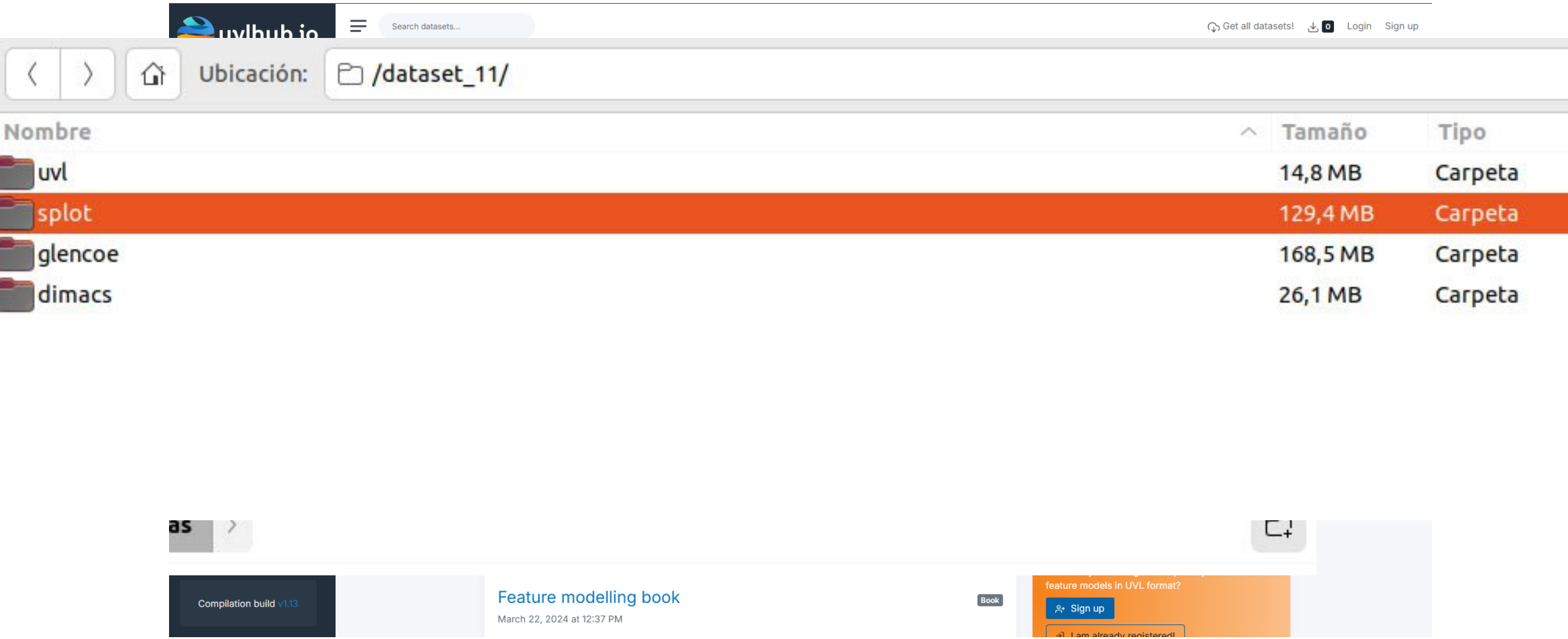
2. Tipos de cambio | ejemplo



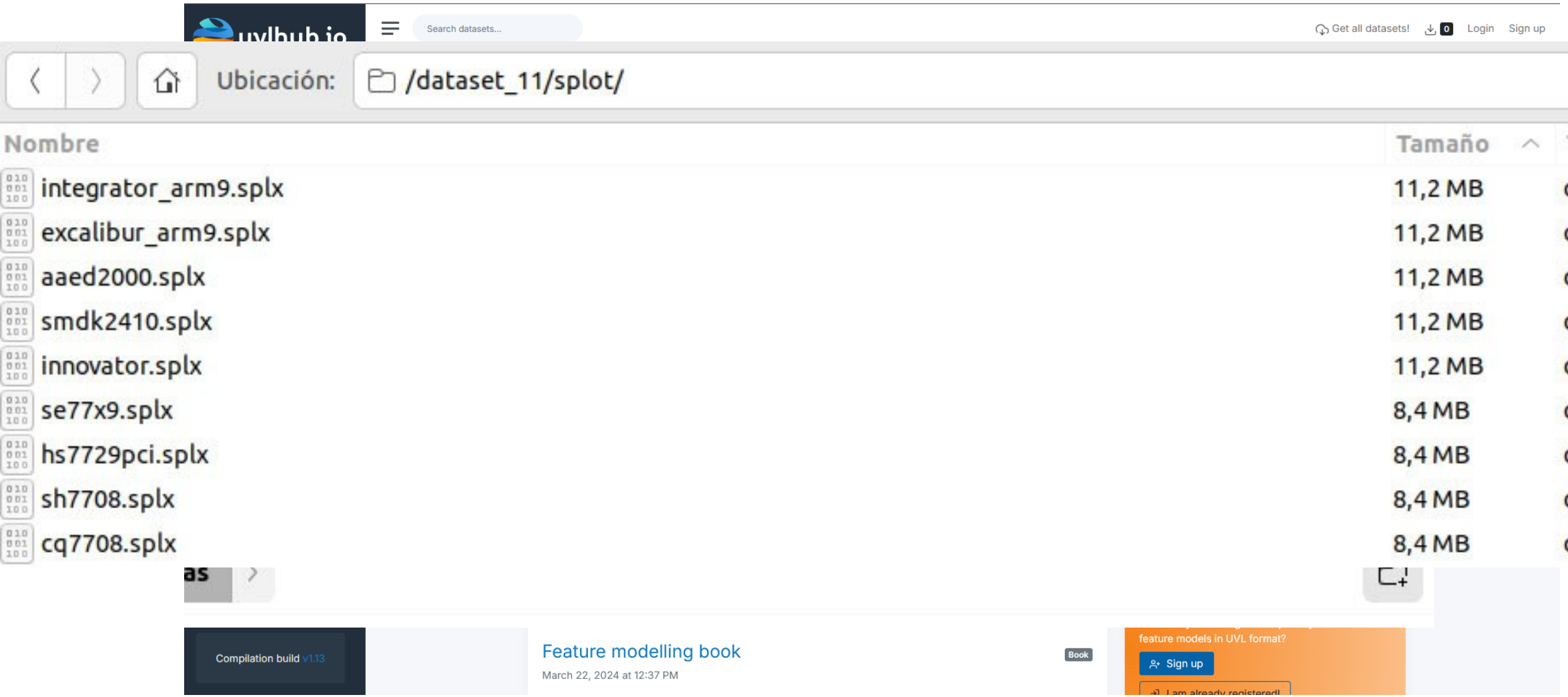
2. Tipos de cambio | ejemplo



2. Tipos de cambio | ejemplo



2. Tipos de cambio | ejemplo



2. Tipos de cambio | Incidencias

1. Conceptos básicos
2. Tipos de cambio
3. **Depuración**
4. Resumen
5. Bibliografía

3. Depuración| Proceso general en caso de error

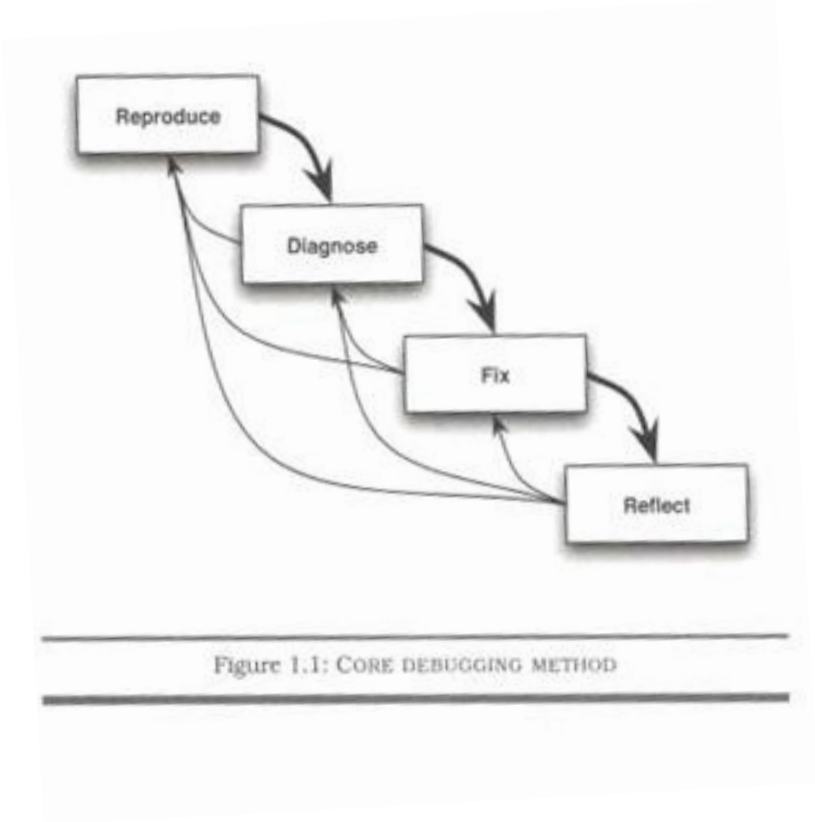


Figure 1.1: CORE DEBUGGING METHOD

- Reproducir: buscar una manera de reproducir el problema de una manera sistemática
- Diagnosticar: Construir hipótesis y validarlas haciendo experimentos
- Reparar: Diseñar e implementar cambios para solucionar el problema
- Analizar: aprender de las lecciones del error

3. Depuración| Distintos pasos

Depuración

- **Informar de la incidencia**
- Reproducir
- Diagnosticar
- Arreglar
- Analizar

Ley de Linux:

"dado un número suficientemente elevado de ojos, todos los errores se convierten en obvios“

3. Depuración| Informar de la incidencia

- Antes de empezar a intentar reproducir debemos estar seguros de entender qué es lo que está pasando:
  - Necesitamos un *bug-tracking system*:

3. Depuración| *bug tracking system*

- Nos permite:
  - No olvidar la incidencia
  - Forma estándar de informar sobre incidencias
  - Para auditar *bugs* (e.g en las entregas)
  - Para priorizar y construir la "*work item list*"
  - Para comunicación en el equipo
  - .....

Todo proyecto de software debe contar con un sistema de gestión de incidencias

3. Depuración| *bug tracking system*

Pero un sistema de gestión de incidencias es el medio, no el fin. Debe haber buenas prácticas al usarlo.

3. Depuración| ¿Cómo informar de una incidencia?

- Cuántos más detalles mejor
- Intentar que sea específico, sin ambigüedades, detallado.
- Debe ser único (si ya se reportó otra vez, ignorarlo o mezclarlo)

Así no	Mejor así
Salta una excepción cuándo pulso el botón de descargar un data set	Al pulsar el botón de descargar un dataset en un dataset con más de un modelo, con el navegador Firefox versión XX, salta la excepción “Overflow Exception” (compio mensaje de error)
Cuando entro en la sección de visualizar un modelo, uno de los elementos a visualizar me sale con símbolos raros	Al intentar visualizar el model M del dataset XXX, sale con un número de caracteres así: C%%%&&&

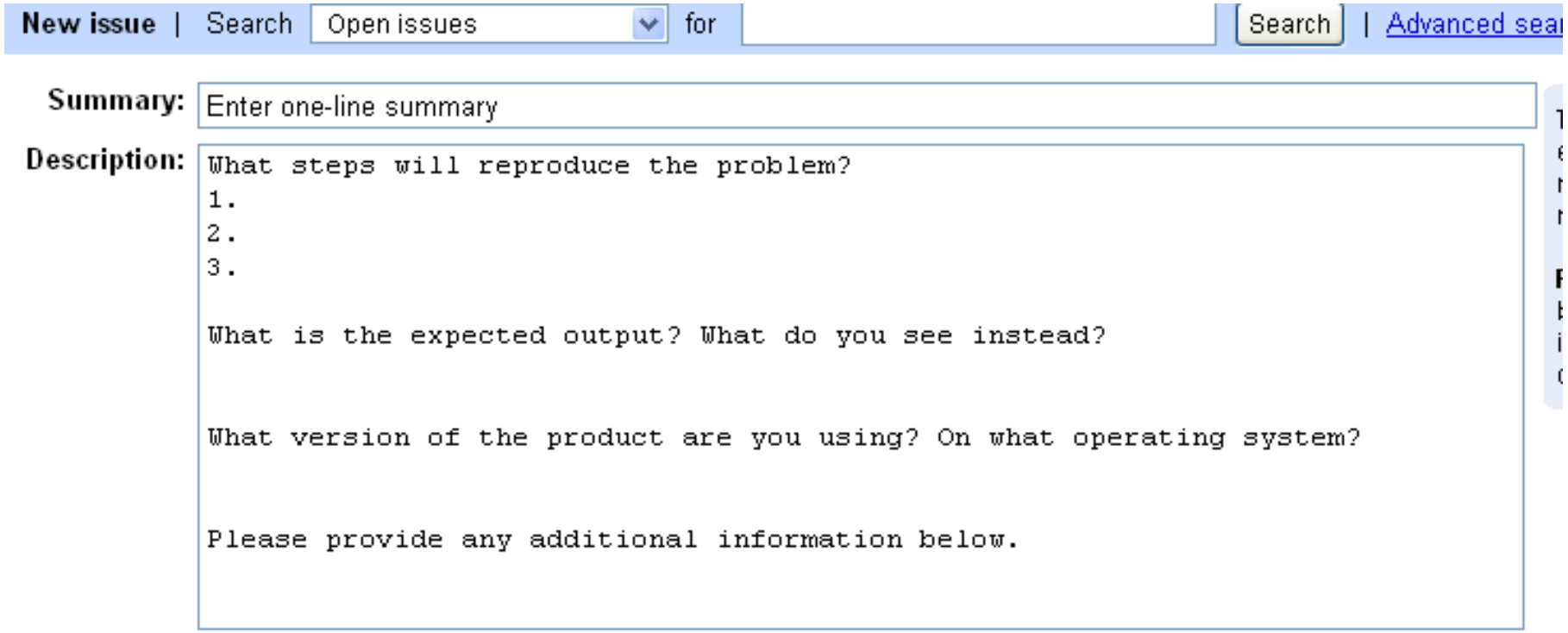
3. Depuración| ¿Cómo informar de una incidencia?

Información de los usuarios

- Usar una plantilla
- Automatizar al máximo

Asumir que de cada error que reporte un usuario, habrá entre 10 y 100 que lo hayan experimentado y no lo reporten [butcher]

3. Depuración| ¿Cómo informar de una incidencia?



3. Depuración| ¿Cómo informar de una incidencia?

What Makes a Good Bug Report?

Thomas Zimmermann, *Member*, *IEEE*, Rahul Premraj, Nicolas Bettenburg, *Member*, *IEEE*, Sascha Just, *Member*, *IEEE*, Adrian Schröter, *Member*, *IEEE*, and Cathrin Weiss

Abstract—In software development, bug reports provide crucial information to developers. However, these reports widely differ in their quality. We conducted a survey among developers and users of APACHE, ECLIPSE, and MOZILLA to find out what makes a good bug report. The analysis of the 466 responses revealed an information mismatch between what developers need and what users supply. Most developers consider steps to reproduce, stack traces, and test cases as helpful, which are, at the same time, most difficult to provide for users. Such insight is helpful for designing new bug tracking tools that guide users at collecting and providing more helpful information. Our CUEZILLA prototype is such a tool and measures the quality of new bug reports; it also recommends which elements should be added to improve the quality. We trained CUEZILLA on a sample of 289 bug reports, rated by developers as part of the survey. The participants of our survey also provided 175 comments on hurdles in reporting and resolving bugs. Based on these comments, we discuss several recommendations for better bug tracking systems, which should focus on engaging bug reporters, better tool support, and improved handling of bug duplicates.

Index Terms—Testing and debugging, distribution, maintenance, and enhancement, human factors, management, measurement.

3. Depuración| Distintos pasos

Depuración

- Informar de la incidencia
- **Reproducir**
- Diagnosticar
- Arreglar
- Analizar

3. Depuración| Reproducir

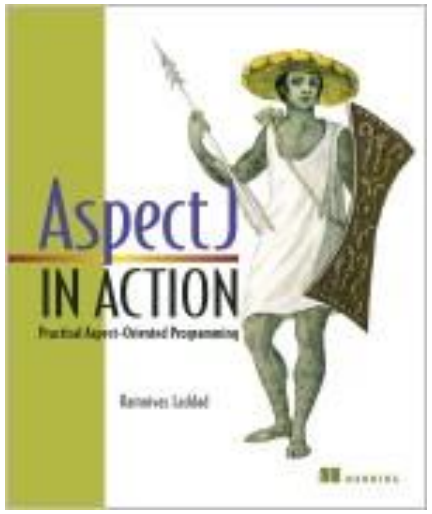
- Dos escenarios: tengo toda la información, no la tengo
- Algunas recomendaciones si no la tengo:
  - Intentar reproducir sin preguntar
  - Empezar por lo simple
  - Intentar reproducirlo en el mismo entorno (máquinas virtuales)
  - Al menos tres escenarios: desarrollo, preproducción y producción.
  - Usa técnicas de pruebas
  - Si el fallo ha sido detectado usando casos de prueba, reproducir dichos casos de prueba
  - Hacer lo que aparentemente es no determinista, determinista.
  - Usar información de *log* (*log* sí *vs log* no)

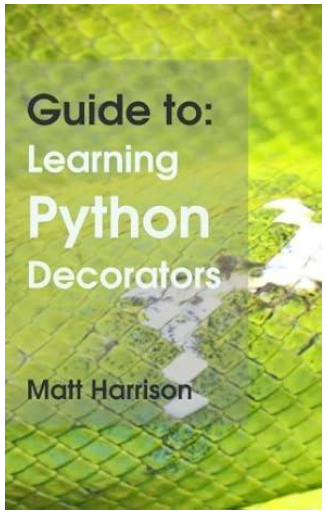
¿*Logging* sí o no?

3. Depuración| ¿*logging* sí o no?

- ¿Argumentos a favor del logging?
- ¿Argumentos en contra?
  - Ensucia el código impidiendo diferenciar “las hojas de las ramas”
  - Puede tener los mismos problemas que los comentarios: tienen que tener un sincronización con los cambios en el código que según la experiencia, no siempre es así [butcher]
  - Conjetura: “No importa la cantidad de información de *log* que añadas, nunca será la que necesitas”

- Posible solución:





3. Depuración| Distintos pasos

Depuración

- Informar de la incidencia
- Reproducir
- **Diagnosticar**
- Arreglar
- Analizar

3. Depuración| Diagnosticar

- Para diagnosticar software hace falta método y agilidad mental.
- Un experimento debería poder probar algo.
- Un experimento debe introducir un solo cambio.
- Anotar la traza de experimentos
- Pedir una mano al compañero/a
- Ejemplos:

Hipótesis: El error se debe a que nuestro software no soporta la versión del driver de selenium
Experimento: cambiar el driver selenium

Hipótesis: Hay una variable que no se inicializa o se inicializa a null.
Experimento: inicializar esa variable con un valor distinto de null.

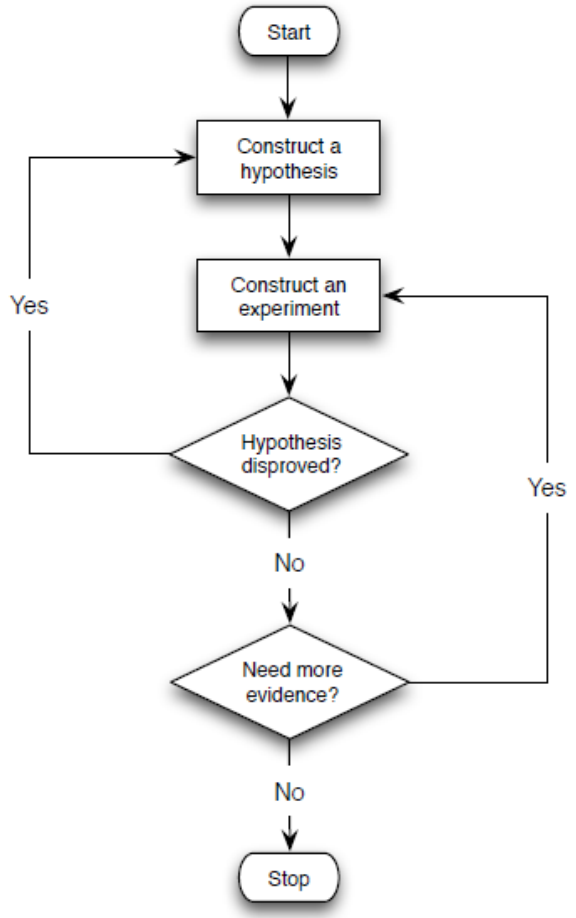
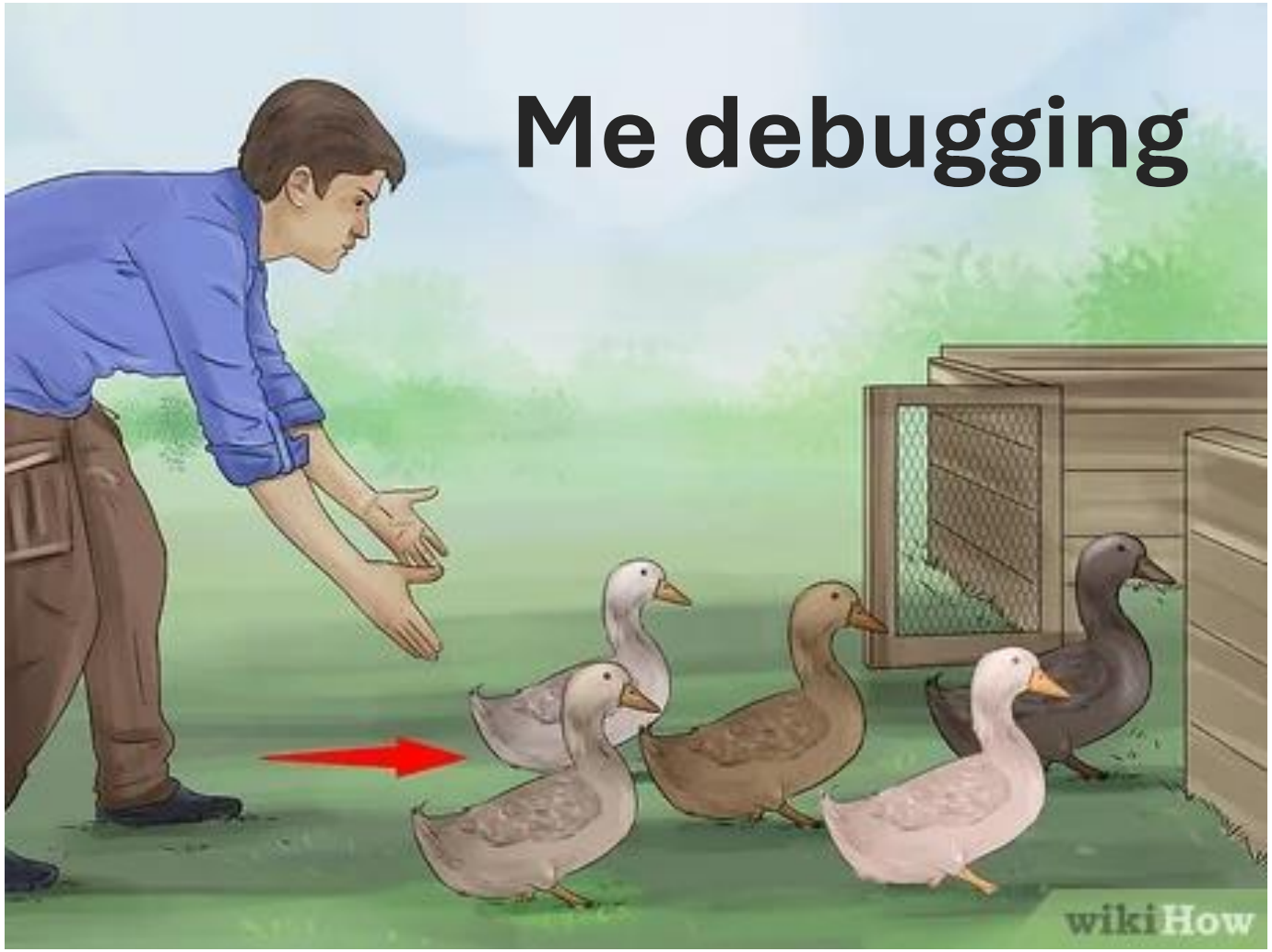


Figure 3.1: A Debugging Method

3. Depuración| Un método para construir hipótesis


Me debugging
wikiHow

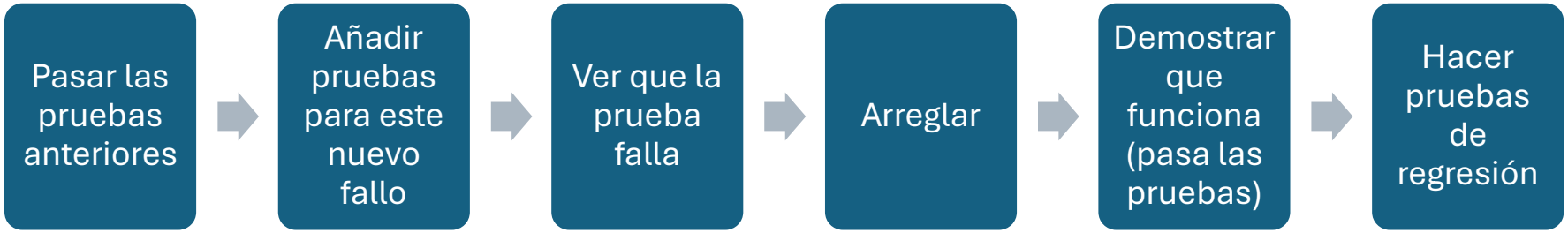
3. Depuración| Distintos pasos

Depuración

- Informar de la incidencia
- Reproducir
- Diagnosticar
- **Arreglar**
- Analizar

3. Depuración| Arreglar

- Arreglar no significa dejar el sistema en el último estado del último experimento
- Primero hay que volver a empezar en el estado inicial.
- Usemos la función *diff y cherrypick*
- Pasos para arreglar:



- Subir el cambio a “preproducción” o enviar como un “*patch*” y documentar y gestionar en el “*work item list*”

3. Depuración| Arreglar

IMPORTANTE

- Arreglar las causas no los síntomas: si hay un valor de una variable que parece ser incorrecto, no cambiar el valor de la variable, pensar por qué ha llegado ese valor incorrectamente.
- No intentar arreglar y refactorizar al mismo tiempo

3. Depuración| Distintos pasos

Depuración

- Informar de la incidencia
- Reproducir
- Diagnosticar
- Arreglar
- **Analizar**

3. Depuración| Analizar

- Intentar aprender de lo corregido.

> *"Tu mejor maestro es tu último error."*
> Ralph Nader (1934-?) Activista y abogado de USA

- Las etapas de corrección de errores:
  1. No puede ser
  2. No sucede en mi máquina
  3. No debería pasar
  4. ¿Por qué está pasando?
  5. Ah, ¡ya!
  6. **¿Cómo podía estar funcionando?**

"The six stages of debugging"

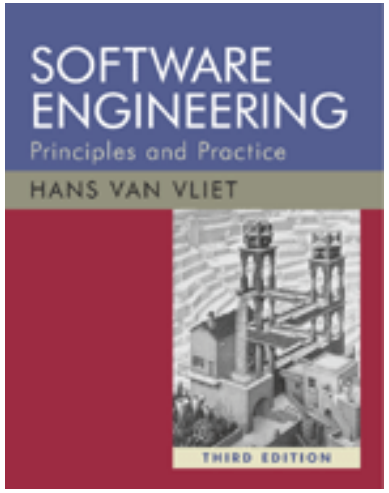
1. Conceptos básicos
2. Tipos de cambio
3. Depuración
4. **Resumen**
5. Bibliografía

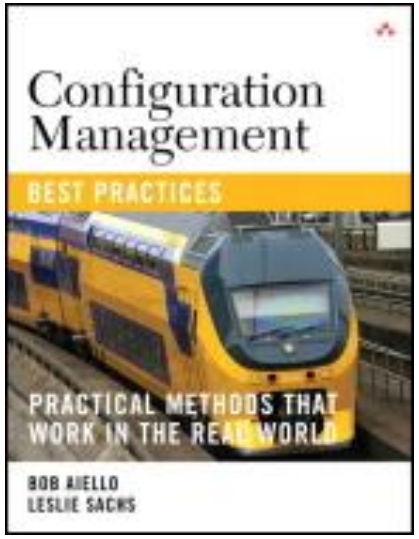
4. Resumen

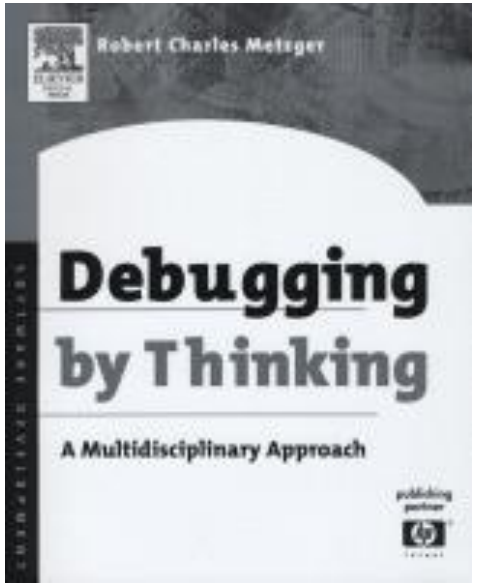
- ¿Qué hemos aprendido?
  - El cambio es inevitable
  - Si no se gestiona bien puede haber muchos problemas
  - Hay que gestionar los cambios de manera ordenada con estados predeterminados, con tipos de incidencias, con prioridades y con roles
  - Informar de un error es una tarea crucial
  - Para depurar hace falta método y estrategias

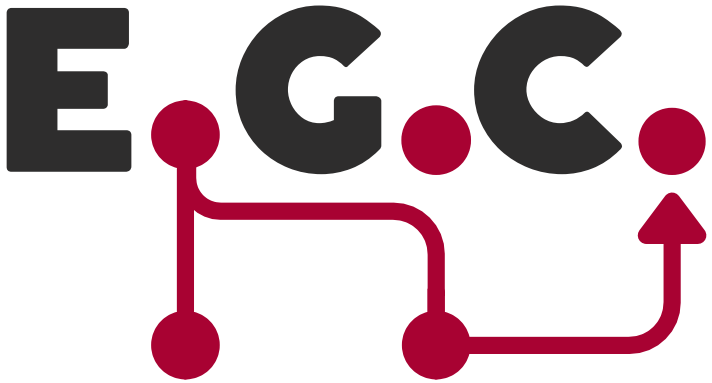
1. Conceptos básicos
2. Tipos de cambio
3. Depuración
4. Resumen
5. Bibliografía

5. Bibliografía









Grado en Ingeniería Informática - Ingeniería del Software

Evolución y Gestión de la Configuración

Universidad de Sevilla 1505

Escuela Técnica Superior de Ingeniería Informática

¡Gracias!