Desarrollo dirigido por Pruebas y Entrega Continua

DE MÉTODOS ARTESANALES HACIA UNA LÍNEA DE PRODUCCIÓN DE SOFTWARE PROFESIONAL

ING. JIMMY FIGUEROA

CLUB DE INVESTIGACIÓN TECNOLÓGICA

17 AGOSTO 2016

TEST DRIVEN DEVELOPMENT

La calidad es ciudadano

Clase A

DISEÑO

La arquitectura y los patrones de diseño cuentan en paleta !!

COLABORAR

Todo esta hecho para diseñar y desarrollar colaborativamente, hoy nada sucede en el aislamiento.

GITHUB ROCKS!

La elección de un gestor de versiones es prueba superada

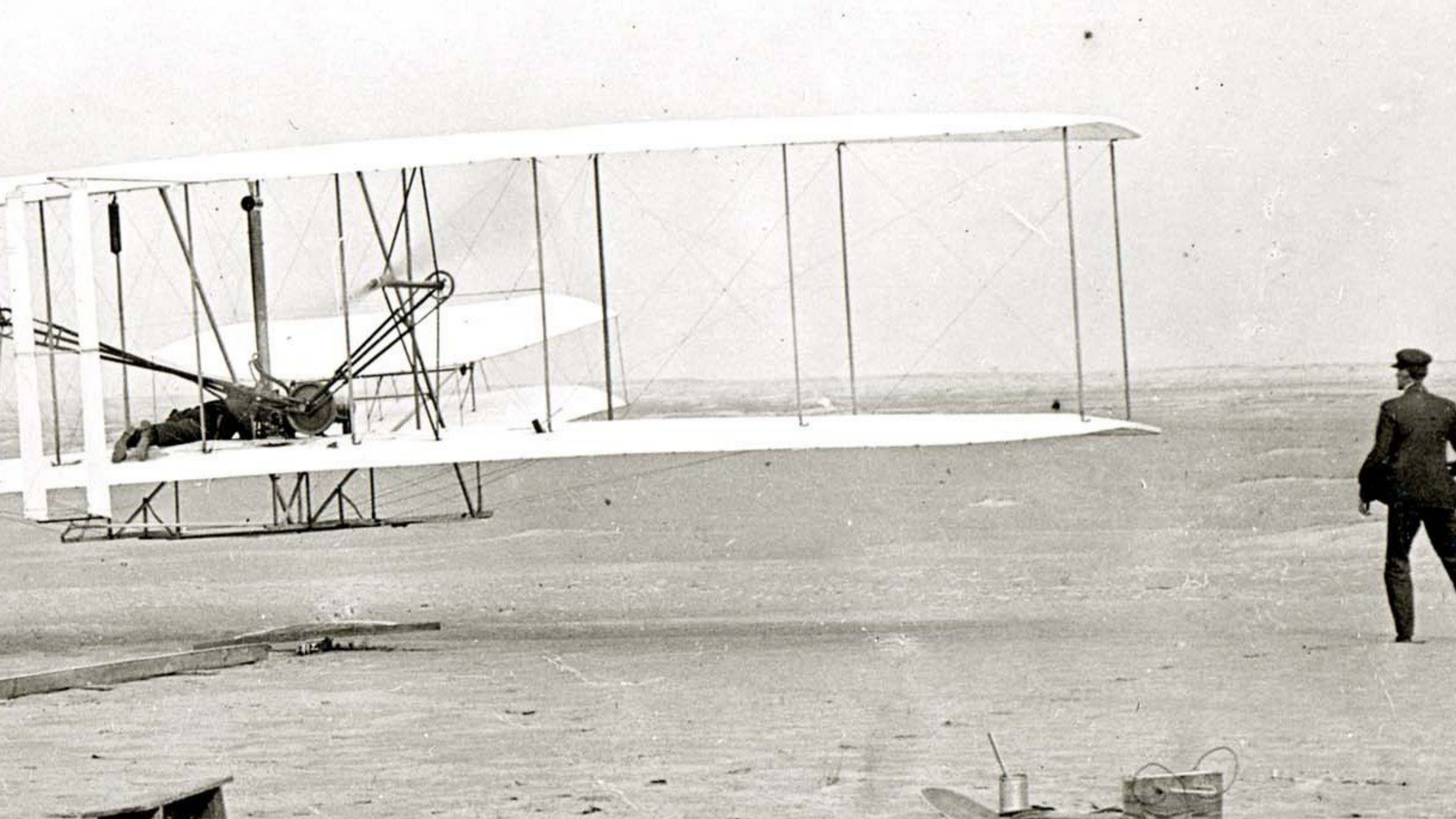
PROJECT MGMT NO ES SUFICIENTE

PM + SCRUM, ok, pero no basta, igual hay que saber un poco de D+D

CONTINUOUS INTEGRATION

La vaina es integrar todos los días

creative director
art director
engineering manager
software architect
principal graphic designer
interaction designer
information architect
usability engineer
media producer
graphic designer
content writer
content editor
marketing manager
front end web developer
interactive developer
back end developer
seo specialist
accesibility test engineer
test engineer
network engineer
server administrator
database administrator
client business owner
client domain expert
the REDMOND
NOW SHOWING
SOLD OUT!
A WEBSITE NAMED DESIRE
A LOOK AT WHAT IT TAKES TO GET A WEBSITE OFF THE GROUND
XHTML
CSS
HTTP://WWW.AWEBSITENAMEDDESIRE.COM/
WAR ROOM
NO LAST MINUTE FEATURE REQUESTS
Microsoft
STAGE 27
TODAY'S AGENDA
Finalize Concept!
POPCORN!
ELVIS LIVES!
W3C
Hh

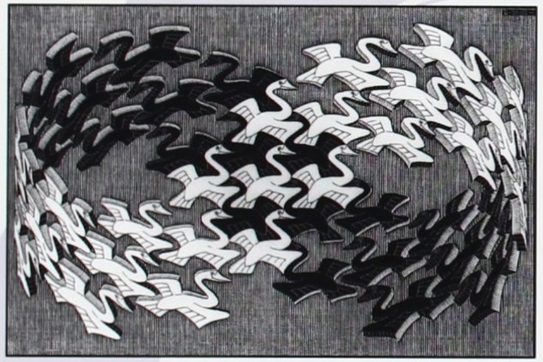


cassandra
PostgreSQL
Java
ORACLE
Microsoft
mongoDB
heroku
+ableau
SOFTWARE
MATLAB
Data-Driven Documents
redhat
ŷhat
APACHE HBASE
R
hadoop
RAILS
python
WEKA
The University of Waikato
HIVE
CODE CLIMATE
ubuntu
ThingWorx
A PTC Business
Spark

Design Patterns

Elements of Reusable Object-Oriented Software

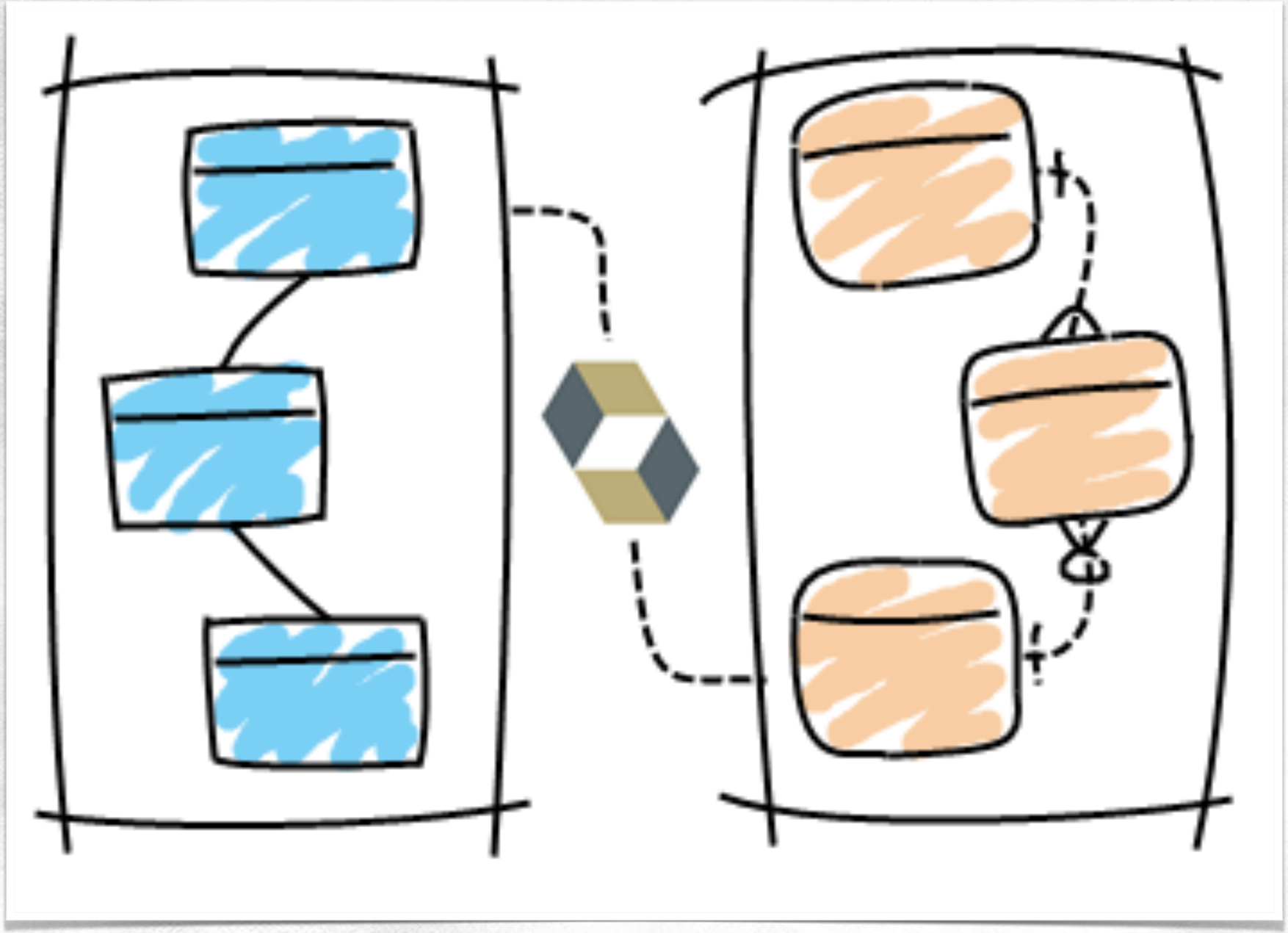
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

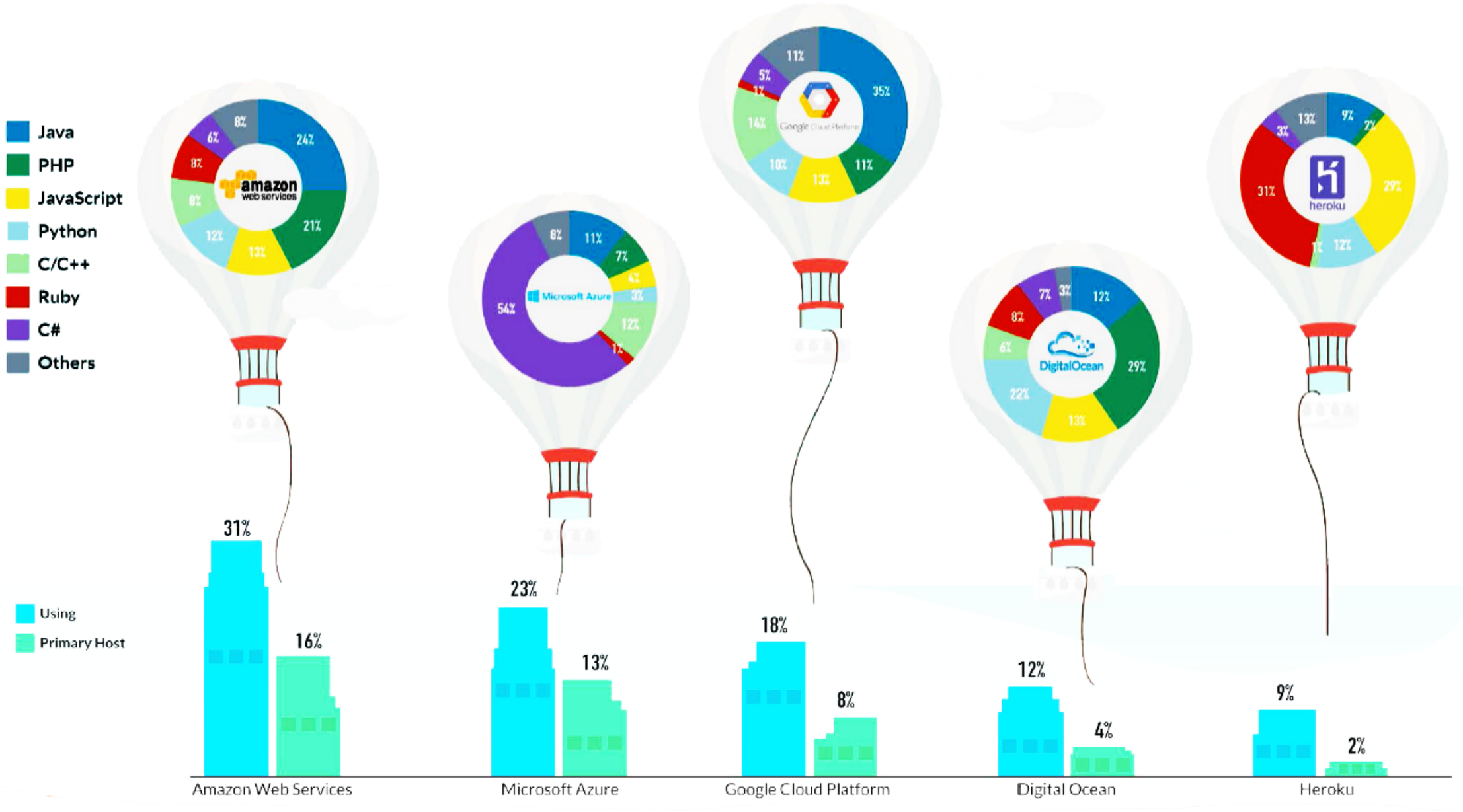


METEOR
npm
UNDERSCORE.JS
Sammy.js
browserify
Knockout.
BACKBONE.JS
Breeze
ANGULARJS
express
ember
HO
OCH3
DERBY
YO
Cappuccino
TypeScript
node
JS
JayData
mongoDB

ember
node
JS
Java

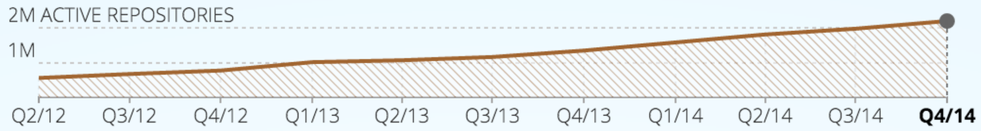
AZURE VS. HEROKU VS. AWS: IT'S ALL ABOUT THE LANGUAGE

54% of Azure developers prefer C# while 60% of Heroku developers prefer Ruby or JavaScript. AWS is the place for all

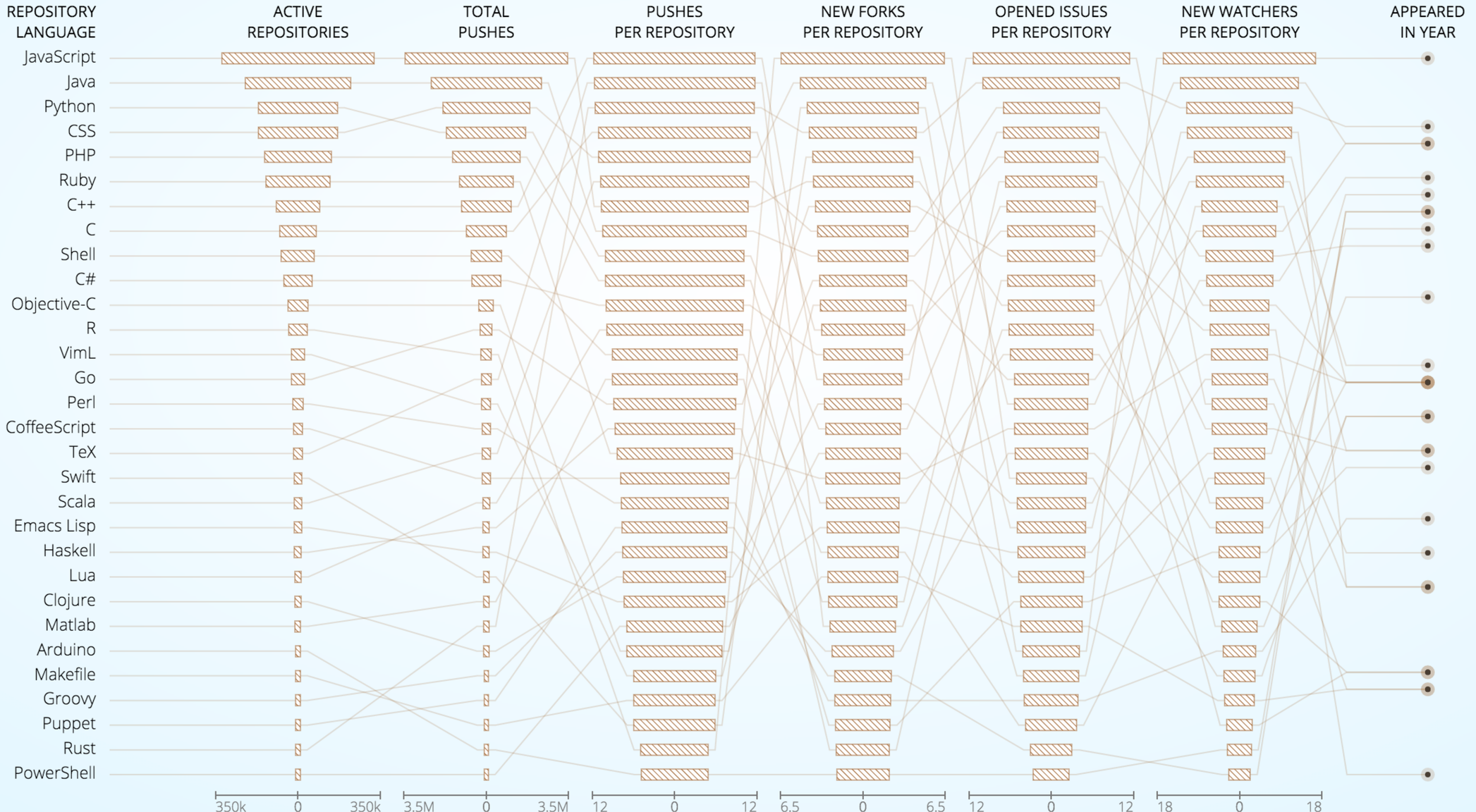


GitHut

A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB



< Q4/14 >



Application and Data

nginx

Web Servers

JavaScript

Languages

MySQL

Databases

Rails [9]

Frameworks (Full Stack)

Ruby

Languages

Redis

In-Memory Databases

Java

Languages

Amazon EC2 [14]

Cloud Hosting

Amazon S3 [15]

Cloud Storage

Sass

CSS Pre-processors / E...

React

Javascript UI Libraries

Amazon RDS [23]

SQL Database as a Ser...

Amazon ElastiCache [33]

Managed Memcache

Amazon EBS [35]

Cloud Storage

Hadoop

Databases

Presto [44]

Big Data Tools

Airflow

Workflow Manager

Airpal [49]

Database Tools

Druid [52]

Big Data Tools

Application and Data

GitHub, Inc. [US] https://github.com/geekjimbo

Jimmy Figueroa

geekjimbo

Add a bio

dvinciworks

san jose, costa rica

https://cr.linkedin.com/in/jim...

Joined on Oct 5, 2013

0	0	1
Followers	Starred	Following

Popular repositories

Customize your pinned repositories

Agile_Data_Code 0
Chapter-wise code for Agile Data the O'Reilly book

cognitiva 0

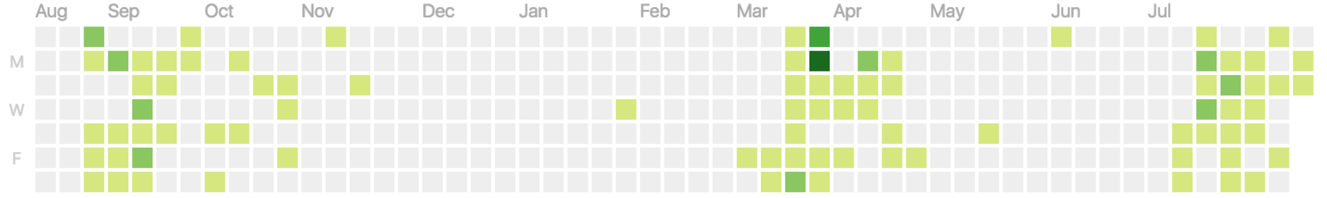
coopecg 0
coopeguanacaste

datascience 0
datascience

everydayrails 0

396 contributions in the last year

Contribution settings



Summary of pull requests, issues opened, and commits. Learn how we count contributions.

Less More

Contribution activity

Period: 1 week

8 REPUTATION

3

geekjimbo

aspirante a hacker y data scientist

Complete BIO (spanish version) here
http://www.dvinciworks.com/bio.html

BY DAY: Software developer freelancer (www.dvinciworks.com) Ruby, Rails, TDD, Python Developer don't like Java or C# so much but I code in that if needed to be BPMn2.0 and UML2.0 ninja Any SQL and Non-SQL DB Tableau Software fan and any D3js.org follower

BY NIGHT: Machine Learning (Data Science), Jetbrain's MSP and DSL practicioner

Create a Developer CV

1 answer | **4** questions | **~1k** people reached

- Costa Rica, Central America
- jimfig35
- geekjimbo
- dvinciworks.com
- Member for 1 year, 6 months
- 3 profile views
- Last seen 3 mins ago
- visited 332 days, 8 consecutive

Communities (1)

Stack Overflow 8

Edit list →

Top Network Posts

Like this site? Check out our other awesome communities:

Top Tags (15)

python	SCORE 0	POSTS 2	POSTS % 40
exception	SCORE 0	POSTS 2	
nameerror	SCORE 0	POSTS 2	
osx	SCORE 0	POSTS 2	
ruby	SCORE 0	POSTS 2	
pandas	SCORE 0	POSTS 2	

TEST DRIVEN DEVELOPMENT

o de cómo reducir dramáticamente el desperdicio

El autor Kent Beck (2002) nos ofrece una analogía para apreciar mejor el Desarrollo Dirigido por Pruebas:

> *Imaginemos la programación como girar una manivela para tirar un cubo de agua de un pozo. Cuando el cubo es pequeño, una manivela sin hilar está bien. Cuando el cubo es grande y está lleno de agua, te vas a cansar antes de que el cubo transcurra todo el camino hacia arriba. Es necesario un mecanismo de trinquete para que se pueda descansar entre episodios de arranque. Cuanto más pesado es el cubo, más cerca los dientes deben estar en el trinquete.*

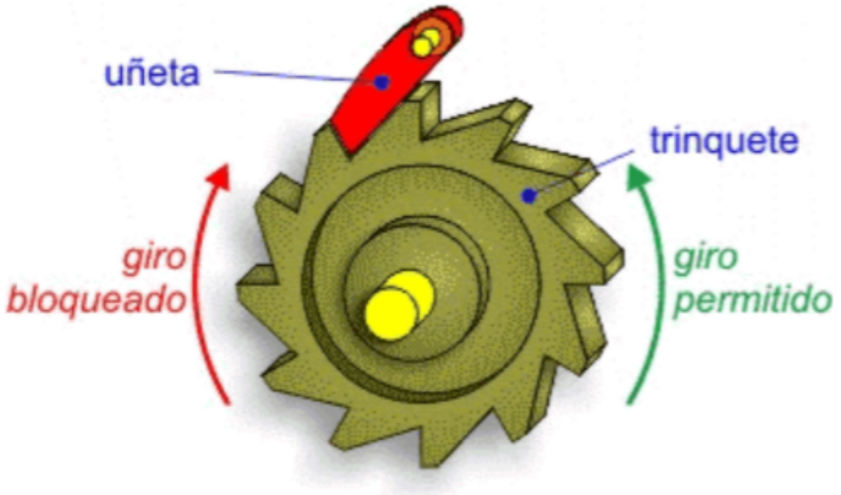


Figura 4: Mecanismo de trinquete (CNICE, 2006)

> *Las pruebas en el Desarrollo Dirigido por Pruebas (TDD) son los dientes del trinquete. Una vez que funciona la primera prueba, sabemos que funciona ahora y siempre. Estaremos un paso más cerca de tener todo el código funcionando de lo que estaríamos cuando la prueba estaba fallando. Así podríamos concentramos en hacer funcionar la siguiente prueba y luego la siguiente y luego la que sigue. Por analogía, entre más grande y complejo el problema, menor la superficie que debe cubrir la prueba.*

ANTES

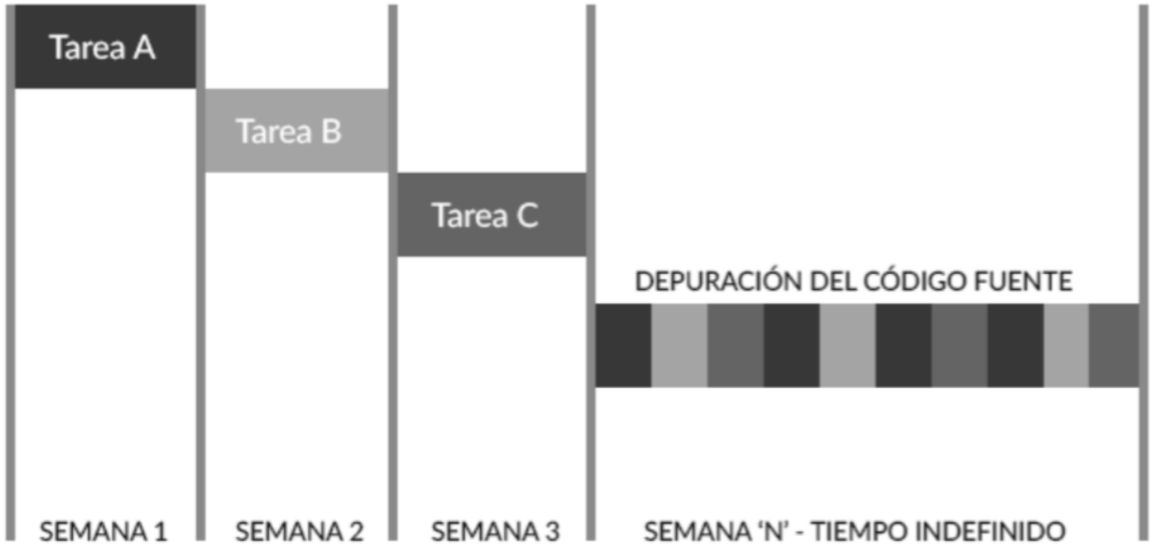


Figura 1. Proceso tradicional de diseño y desarrollo - Pruebas del código son posteriores

DESPUÉS

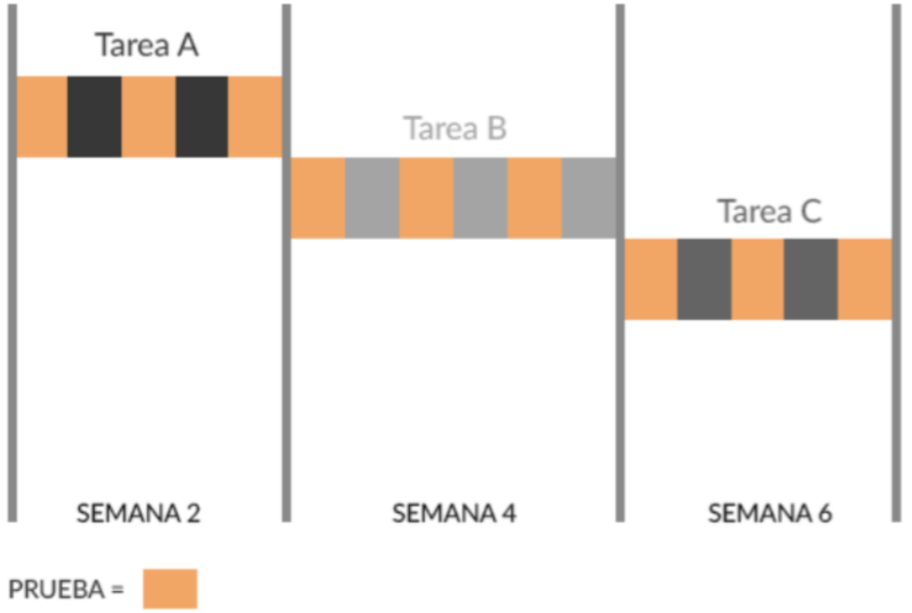


Figura 2. Diseño y desarrollo dirigido por pruebas
Las pruebas son parte del proceso de programación.

¿Y … PORQUE? TDD

Desarrollo dirigido por Pruebas – Test Driven Development

REFACTORING

Reducir el desperdicio mediante la normalización del código (refactoring)

REGRESSION TEST

Diseñar una base de código fuente más estable ante los cambios.

COMUNICACIÓN

Construir interfaces de alto nivel primero y desarrollar las interfaces de bajo nivel (APIs) después.

REDUCIR PULGAS

Prevenir la liberación de cosas "rotas"

INICIO
ESCRIBIR PRUEBA
ROJO
CÓDIGO LIMPIO
PRUEBA FALLA
DESARROLLO DIRIGIDO POR PRUEBAS
ELIMINAR DUPLICIDAD, ¿CÓDIGO LISTO?
FACTORI-ZAR
VERDE
ARREGLAR PRUEBA, PARA QUE PASE
CÓDIGO PASA LA PRUEBA

II

ENTREGA CONTINUA DEL SOFTWARE

automatizar la línea de producción del software

Cuadro 3: Anti-Patrones del Despliegue Tradicional de Software

Id.	Anti-Patrón
#1	Producción de documentación extensiva del paso a paso para desplegar el software
#2	Dependencia de pruebas manuales para confirmar el funcionamiento correcto
#3	Llamadas de urgencia al equipo de desarrollo durante la etapa del despliegue del software para explicar porqué no funcionan las cosas
#4	Cambios frecuentes al proceso durante el despliegue
#5	Entornos de producción que difieren de la configuración del entorno probado
#6	Liberaciones de software impredecibles que muchas veces requieren un retroceso a la última versión que funciona (*rollback*)
#7	Procesos de liberación y despliegue 'con los dientes apretados', no se sabe qué ocurrirá

El término con el que inició este patrón o proceso fue acuñado como "Integración continua", posteriormente fue referido como "Entrega continua" del Software y en el presente informe tratamos ambos términos intercambiablemente. Una posible definición sobre Entrega continua es la que ofrece Wikipedia, que parafraseamos así: *La integración continua consiste en hacer muy frecuentes integraciones automáticas de un proyecto, para así poder detectar fallos cuanto antes. Las integraciones comprenden la compilación y ejecución de pruebas de todo un proyecto. Los equipos producen software en ciclos cortos que aseguren que el producto puede ser liberado confiablemente en cualquier momento* (Wikipedia, 2016a, 2016b).

El proceso de "Entrega continua" del Software consta de 2 componentes principales:

1. La Tubería de implementación;
2. La Integración Continua del Software.

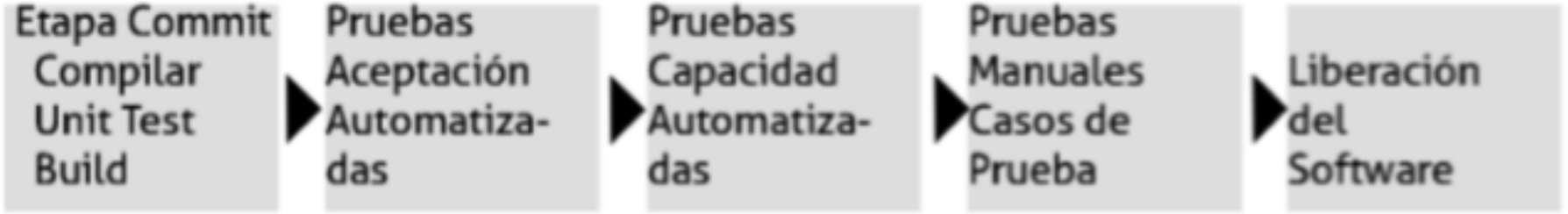
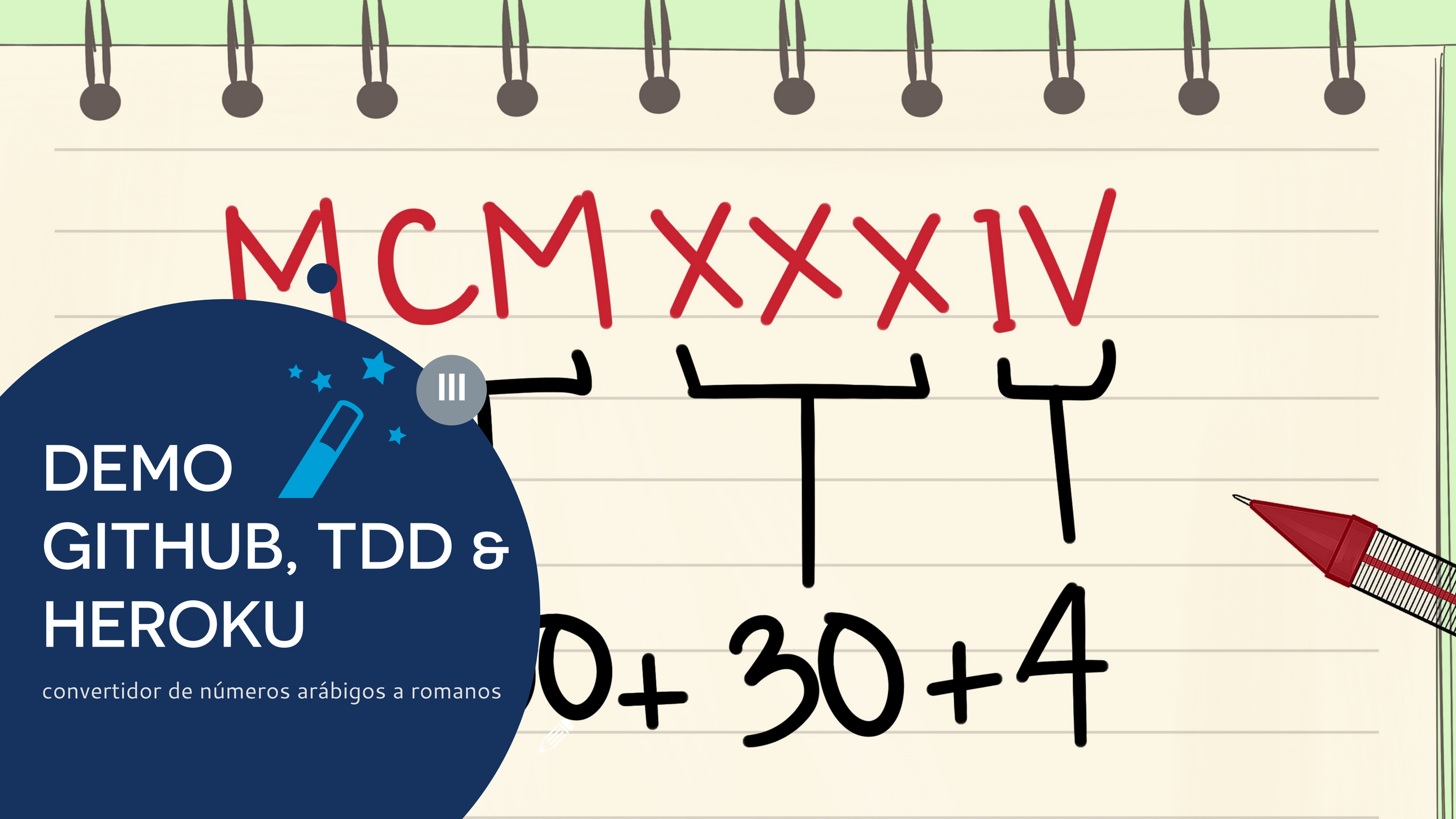


Figura 6. Tubería de Implementación para la Entrega Continua de Software

Cuadro 4: La Retroalimentación que ofrece la Tubería de implementación

Tipo	Alcance
Todo cambio dispara el proceso	En el código fuente, configuración, "scripts" de Base de datos, etc.
Retroalimentación en tiempo real	Pruebas rápidas, cubren al menos el 75% código fuente
Actuar sobre la retroalimentación	Desarrolladores, encargados pruebas, operaciones, Admin. de Base datos, especialistas en infraestructura, Administradores de proyecto

Figura 7. Servidor de integración para la Entrega continua del software



DEMO GITHUB, TDD & HEROKU

convertidor de números arábigos a romanos

MUCHAS GRACIAS

FOR SU PACIENCIA

nos vemos !