SG SOFTWARE GURU

NO.54 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

ESPECIAL
FINTECH
PAG. 18

ENTREGA CONTINUA

¿QUÉ ESPERAS PARA COMENZAR?

90% de lo que aprendes en un curso lo olvidas en sólo un mes

Descubre cómo las empresas orientadas al conocimiento están optimizando su presupuesto de capacitación con los Ambientes de Aprendizaje Ubicuos de Abiztar.

ABIZTAR
LEARNING TECHNOLOGIES

VIDEO Los cursos tradicionales son cosa del pasado.

Mira este revelador video en
abiztar.com.mx/video

ABIZTAR
LEARNING TECHNOLOGIES

Nuestro compromiso contigo no termina con un curso, termina con el éxito de tus proyectos.

(52) 55-5594 6411
cursos@abiztar.com.mx
www.abiztar.com.mx

ML ML BPMN CMMI MDA GRAILS Java SCRUM TOGAF SPARX SYSTEMS Authorised Reseller Partner MEMBER OMG OBJECT MANAGEMENT GROUP Registered Education Provider Project Management Institute

MÉXICO
CUTTER CONSORTIUM
Access to the Experts

AGILE MÉXICO CONFERENCE

NOVIEMBRE 8 Y 9 - 2017
CIUDAD DE MÉXICO

SAVE THE DATE

Sanjiv Augustine
Sheila Cox
Masa Maeda
Marco Navarro
Steve Bell
Karen Whitley

www.agileconference.cutter.mx

SGNEXT

Transformación Digital > DevOps > Arquitecturas de Nube

Ven a conocer el
nuevo paradigma de
desarrollo de software.

La 2da. edición de SG Next se enfocará en las estrategias, prácticas y plataformas que definirán la agenda de las organizaciones de TI durante los próximos años.

22 de junio 2017
Hotel Crowne Plaza
WTC, Ciudad de México

http://sg.com.mx/sgnext

SG VIRTUAL CONFERENCE

12va. edición

Participa en más de 20 conferencias virtuales durante todo un día y conéctate con miles de colegas alrededor del mundo.

Miércoles 24 de mayo 2017

Sede: Todo el mundo

Evento Gratuito.

Se parte del evento que acerca a la comunidad de software de habla hispana con las tendencias, herramientas y mejores prácticas, en cualquier parte del mundo, y sin costo alguno.

¡Participa como sede virtual!
Lleva el evento a tu empresa o institución.
Regístra tu sede: www.sg.com.mx/sgvirtual

Contacto: **eventos@sg.com.mx**

Softwareguru

@RevistaSG

Regístrate
sg.com.mx/sgvirtual

SG®
SOFTWARE GURU
NO.54 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

EN PORTADA

ENTREGA CONTINUA

024

El concepto de entrega continua está ganando tracción en las organizaciones; sin embargo, su adopción no es trivial. Compartimos varios artículos que te guiarán en este camino.

Entrega continua

En Software Guru nos ocupa mantenerte actualizado en tendencias,por esta razón, en las páginas de esta revista leerás sobre entrega continua, un modelo que consiste en hacer integraciones automáticas de un proyecto lo más frecuente para poder detectar fallos de manera temprana. Esta práctica se ha convertido en una parte casi esencial en el desarrollo de software; conocerás porqué es importante como desarrollador conocer esta tendencia y podrás decidir cómo usarla a tu favor.

También discutimos sobre Fintech, un dominio de actividad en el cual las empresas utilizan las tecnologías de la información para ofrecer servicios financieros de una manera más eficaz y obviamente con menor costo. Nos dimos a la tarea de reunir para ti opiniones de expertos sobre las Fintech para que puedas generar la propia.

Recuerda que el tema de la siguiente edición será sobre el automóvil como plataforma de software y casos de estudio sobre Salud, si te interesa colaborar, envía tu propuesta de contenido a editorial@sg.com.mx.

Para finalizar queremos agradecerte tu preferencia en los primeros eventos del año: SGTalento, Hackatour, CTO Forum y DataDay 2da Edición, ¡FUERON UN ÉXITO!, sin ti no hubieran sido posibles. Para los siguientes meses no te pierdas nuestros eventos, como es costumbre te sorprenderemos.

El equipo de Software Guru

SG es posible gracias a la colaboración de

Dirección Editorial **Pedro Galván** | Dirección de Operaciones **Mara Ruvalcaba** | Dirección Comercial **Claudia Perea**
Coordinación Editorial **Ana Loyo** | Arte y Diseño **Oscar Sámano** | Suscripciones **Mariana Torres**
Consejo Editorial: Luis Daniel Soto | Gunnar Wolf | Luis Vinicio León | Hanna Oktaba
Ariel Jatuff | Emilio Osorio | Gloria Quintanilla | Jorge Valdés

COLABORADORES EN ESTA EDICIÓN
Gilberto Sánchez, Edith Gómez, Alejandra Lagunes, Héctor Cárdenas, Adrián Sánchez,
Bjorn Cumps, Fermín Bueno, Misael León, Guilherme Siqueira, Iván Lozada, Herminio Heredia, Diego Maury.

EQUIPO SG
Coordinación de servicio **Yoloxochitl Juárez** | Developer Relations **Luis Sánchez e Hilda Ramírez**
SG Campus **Jhonnatan Castillo** | Servicios online **Ivett Sánchez** | Alianzas **Ángel Tello**
Contacto: info@sg.com.mx

SG Software Guru es una publicación trimestral editada por Brainworx, S.A. de C.V., San Francisco 238 Altos. Col. Del Valle. Los contenidos de esta publicación son propiedad intelectual de los autores y se hacen disponibles bajo licencia Creative Commons Attribution-NonCommercial 4.0 International. Todos los artículos son responsabilidad de sus propios autores y no necesariamente reflejan el punto de vista de la editorial. Reserva de Derechos al Uso Exclusivo: En trámite. ISSN: 1870-0888. Registro Postal: PP15-5106. Distribuido por Sepomex.

Noticias

HACKATOUR QUERÉTARO 1

Querétaro fue la primer parada oficial del Hackatour 2017 de Software Guru el viernes 24 y sábado 25 de marzo de 2017 en las instalaciones de la Facultad de Informática de la Universidad Autónoma de Querétaro. El hackathon formó parte de la celebración de los 30 años de la Facultad de Informática de la UAQ con el patrocinio de Red Hat. Los 130 participantes se organizaron en 28 equipos distintos que construyeron un prototipo para resolver algún problema relacionado con las siguientes temáticas: Ecología y sustentabilidad, inclusión de personas con capacidades distintas o lucha contra el crimen. Consulta las siguientes paradas del Hackatour en https://sg.com.mx/hackatour/

PRIMERA EDICIÓN DEL FESTIVAL DE LA FIRST® LEGO® LEAGUE JR. EN MÉXICO 2

El sábado 8 de abril se llevó a cabo la primera edición del FestivalFIRST LEGO League Jr. en nuestro país, en el Centro Cultural España, donde 150 niños divididos en equipos, exhibieron los proyectos resultantes de este programa, cada uno pensado en resolver un reto a través de distintas soluciones que mezclan tecnología y creatividad.

ASÍ SE VIVIÓ DATA DAY MÉXICO 2DA. EDICIÓN 3

El pasado martes 28 de Marzo tuvimos la oportunidad de compartir con cerca de 400 apasionados de los datos en la segunda edición del Data Day, el evento de datos más grande de México. Los contenidos se dividieron en 3 grandes tracks: Data Science MBA, Data Science + Machine Intelligence y Big Data Engineering, y hubo talleres prácticos a cargo de empresas como RStudio y Amazon Web Services. La próxima edición de Data Day será en marzo de 2018. Mantente al tanto en http://sg.com.mx/dataday

SG TALENTO, EL EVENTO DE RECLUTAMIENTO 4

El pasado 23 de Febrero, con más de 230 asistentes, 5 empresas virtuales, más de 70 candidatos , más de 10 expositores y más de 100 vacantes en CDMX, MTY, GDL y USA se vivió la primera edición del SG Talento. SG Talento es un evento de reclutamiento, networking y sesiones de desarrollo profesional, que reúne a las empresas más reconocidas en TI para ofrecer sus vacantes de trabajo en México y Estados Unidos, además de ofertas para capacitaciones como cursos, talleres, certificaciones, etc. En SG Talento también puedes disfrutar de charlas con ponentes expertos en TI que comparten sus conocimientos y experiencia a través de las sesiones de Desarrollo Profesional. Si quieres conocer más eventos como este puedes visitar https://sgtalento.com/eventos

DOCKERCON 2017 5

Más de 5,000 personas se dieron cita en la ciudad de Austin, TX del 17 al 20 de mayo para conocer las novedades alrededor la tecnología docker durante DockerCon. Entre los anuncios más importantes del evento podemos mencionar: la nueva capacidad de multi-stage builds para crear imágenes más ligeras; LinuxKit, un toolkit para construir distribuciones de Linux minimizadas e inmutables; y Moby Project, un proyecto open source para modularizar el desarrollo de la plataforma docker. Al ver el amplio interés de tantas personas y empresas de distintas verticales, nos queda claro que la tecnología de contenedores está alcanzando un punto importante en su madurez y está dejando de ser algo solamente para pioneros.

¿Por qué mi Equipo **Tiene Broncas?**

Por Hanna Oktaba

La Dra. Hanna Oktaba es profesora de la UNAM y su objetivo principal es generar conocimiento a través de la creación y promoción de estándares.
@hannaoktaba

Observando los equipos de mis alumnos, escuchando quejas de los que ya están trabajando y por experiencia propia, coordinando el equipo de Renovación de MoProSoft (si, dio efecto mi convocatoria de SG no.53) me he dado cuenta que convertir un grupo de personas en un equipo productivo y contento no son "enchiladas". Ya los agilistas en su manifiesto de 2001 nos advirtieron que hay que valorar más a los "individuos e interacciones sobre procesos y herramientas". Esta advertencia fue muy importante, surgieron varias propuestas de cómo empoderar a los miembros del equipo, con SCRUM como ejemplo más destacado.

Sin embargo, todavía nos falta entender cómo evolucionan los equipos. Para eso empezamos con mi alumna de doctorado Sandra Ramírez esculcar en Ciencias Sociales. La primera sorpresa fue que hay una diferencia entre un grupo de personas, un grupo de trabajo y un equipo!!! Según [1] y [2]:

- Un grupo: son 3 o más personas que interactúan entre sí para realizar un número de tareas y lograr un conjunto de objetivos comunes (para mi, son burócratas).

- Un grupo de trabajo: tiene miembros que quieren crear el entendimiento común de los objetivos y crear una estructura para lograrlos. (para mi, son empresarios).

- Un equipo: es un grupo de trabajo que tiene objetivos comunes y métodos efectivos para lograrlos (para mi, así deberían ser los equipos de desarrollo de software).

Es decir, no es suficiente juntar unas cuantas personas y decirles cuales son los objetivos del proyecto para que de inmediato se conviertan en un equipo productivo.

Buscando un poco más encontramos que ya en 1965 Tuckman [3] publicó un modelo de desarrollo, no de software sino de equipos. En este modelo se identifican cuatro fases por las que atraviesan los grupos en su desarrollo:

1. FORMING (FORMACIÓN)

En esta etapa el grupo apenas se integró, la gente empieza a conocerse, no tienen muy claro el objetivo del trabajo ni sus roles. Los individuos tratan de transmitir a otros sus habilidades, pero todavía no tienen claro su papel en el equipo y se sienten inseguros. Por lo tanto todos dependen mucho de las instrucciones del líder.

2. STORMING (ENFRENTAMIENTO)

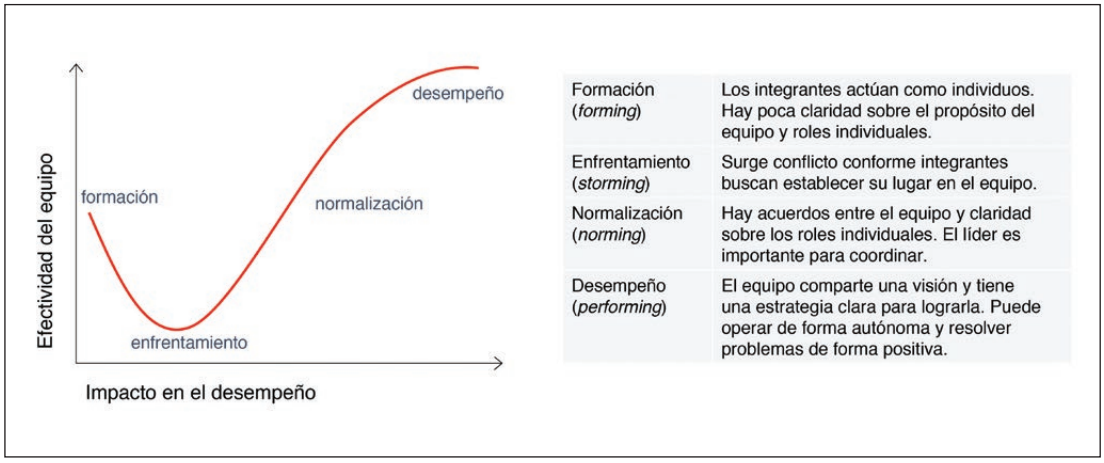
Los miembros del equipo luchan entre sí para posicionarse dentro del equipo. Tratan de establecer por sí mismos relaciones con otros miembros del equipo y con el líder. Se forman pandillas y agrupaciones y se pueden dar luchas de poder. El líder actúa como coach.

FASE 3: NORMING (NORMALIZACIÓN)

Los conflictos se reducen y los miembros empiezan a reconocer que pertenecen a un grupo. Se forman acuerdos y consensos dentro del equipo bajo la batuta del líder. Roles y responsabilidades son claros y aceptados. El equipo lleva a cabo reuniones para discutir y desarrollar sus procesos y su forma de trabajo. El líder es respetado por el equipo y parte del liderazgo es compartido por el equipo. El líder actúa como facilitador.

FASE 4: PERFORMING (DESEMPEÑO)

El equipo trabaja con un buen rendimiento y pocos conflictos, está preparado para tomar decisiones sin la necesidad de la participación del líder. El enfoque está en lograr resultados, el equipo tiene un alto grado



Formación (*forming*)	Los integrantes actúan como individuos. Hay poca claridad sobre el propósito del equipo y roles individuales.
Enfrentamiento (*storming*)	Surge conflicto conforme integrantes buscan establecer su lugar en el equipo.
Normalización (*norming*)	Hay acuerdos entre el equipo y claridad sobre los roles individuales. El líder es importante para coordinar.
Desempeño (*performing*)	El equipo comparte una visión y tiene una estrategia clara para lograrla. Puede operar de forma autónoma y resolver problemas de forma positiva.

Figura 1. Modelo de Tuckman

de autonomía. Cuando ocurren desacuerdos el equipo es capaz de resolverlos. También, el equipo realiza los cambios al proceso y a la estructura cuando sea necesario. El equipo no necesita ser instruido o asistido por el líder. El líder delega.

Algunos dicen que estas etapas de desarrollo de equipos se parecen a las etapas que pasamos los seres humanos en nuestras vidas: Forming –> Infancia, Storming –> Adolescencia, Norming –> Adultez y Performing –> Madurez. Esta similitud me encanta, primero, porque es muy fácil de acordarse y dos porque como ya estoy en la madurez, me gusta pensar que todavía tengo buen "performing" :).

Este modelo me ayudó a entender una de las razones por la que MoProSoft y otros modelos no siempre fueron adoptados con éxito. Sospecho que sus organizaciones estaban en la etapa de adolescencia y todavía no les caía el veinte que para ser productivos necesitaban pasar a la fase de normalización.

Otra cosa que este modelo me ayudó a entender es porqué los equipos no pueden ser muy productivos desde el primer día que se junten. Por más que seleccionemos buenos expertos técnicos, el equipo tendrá que pasar por las etapas de la infancia y la adolescencia antes de volverse adultos, léase productivos. La Fig.1 muestra, además, cómo en la etapa de adolescencia el equipo se vuelve menos efectivo como consecuencia de sus luchas internas; seguro que lo han observado en sus equipos favoritos de fútbol cuando se integran nuevos jugadores.

Y ustedes en sus equipos de trabajo, ¿con cuál etapa se identifican? Aparentemente, mientras más tiempo trabajan juntos deberían ya estar por lo menos en la etapa de adultez. Pero muchos me van a contestar: es que seguimos con las broncas de la adolescencia :(. Entonces analicen las habilidades de su líder. En las primeras dos etapas su papel es primordial.

Un buen líder tiene que:

- Conocer muy bien los objetivos del proyecto y saber transmitirlo al equipo.
- Fungir como una autoridad y capacitador en la parte técnica y en las formas de trabajar.
- Identificar habilidades de cada persona para poder coordinar el trabajo adecuadamente y generar la confianza mutua.
- Ser honesto y justo.
- Ser un buen psicólogo para manejar los estados de ánimo de las personas y saber cuándo "apapacharlas" y cuándo "jalarles las orejas", siempre con respeto.
- Asegurar las condiciones de trabajo adecuadas.

En pocas palabras, un líder tiene que ser un coach, un entrenador, que sobre todo en las primeras etapas de formación y conflicto, ayude a un grupo de personas convertirse en un equipo. Si no me creen, analicen las trayectorias de los entrenadores de fútbol.

Referencias

[1] J. Keyton, "Communicating in groups: Building relationships for group effectiveness", New York, McGraw-Hill, 2002.

[2] S. Wheelan & J. Hochberger, "Validation studies of the group development questionnaire", Small Group Research, vol. 27, no. 1, pp. 143-170, 1996.

[3] B. W. Tuckman, "Developmental sequence in small groups". Psychological Bulletin. 63 (6), 1965.

1 AZURE IOT SUITE

Ante la ola de la Industria 4.0, las empresas de manufactura comienzan a preguntarse qué necesitan hacer para comunicar y dotar de inteligencia a sus líneas de producción. Empresas proveedoras de equipo industrial como GE, Bosch y Siemens están levantando la mano para involucrarse en este espacio. Del lado de los proveedores tradicionales de software, Microsoft es de los más activos y recientemente actualizó las capacidades de su Azure IoT Suite, ofreciendo una solución preconfigurada llamada "Connected Factory" que facilita el arranque de soluciones de internet industrial o industria 4.0. Otra capacidad nueva es poder hacer analítica de datos en dispositivos en la orilla (edge) para poder analizar los datos en tu misma red sin tener que transmitirlos a la nube; esto da ventajas de velocidad, seguridad y consumo de banda ancha. Microsoft también dio a conocer una iniciativa SaaS para IoT denominada Azure IoT Central. Esta será una oferta SaaS para IoT completamente administrada que busca reducir significativamente la complejidad de las soluciones IoT. El servicio todavía no está abierto al público pero puedes conocer más al respecto en http://microsoftiotcentral.com

2 NEXT.JS 2

En el mundo del front-end, posiblemente la tecnología más candente en estos momentos sea React. Con esta popularidad, también han surgido frameworks para React, y uno de los que más nos gusta es Next.js. Este framework pequeño pero poderoso busca reducir significativamente la complejidad de construir aplicaciones React. La primer versión de Next.js apenas fue publicada en octubre del 2016, y en marzo de 2017 ya tenemos la segunda. Entre las capacidades incluidas en esta nueva versión está el soporte a componentes CSS que permite que los desarrolladores escriban CSS de forma tradicional abstrayéndose de las particularidades de CSS en React. Todo parece indicar que React tendrá fuerza por varios años, y Next se está posicionando como una buena opción de framework, así que si tienes oportunidad échate un clavado en https://github.com/zeit/next.js

3 JENKINS BLUE OCEAN

Jenkins, el servidor open source de automatización para entrega continua anunció la disponibilidad de Blue Ocean 1.0. Esta es una nueva interfaz diseñada para fácilmente crear, visualizar y diagnosticar pipelines de entrega continua. El objetivo es que Blue Ocean permita a equipos novatos con poco conocimiento de Jenkins arrancar rápidamente en su adopción hacia la entrega continua. Entre sus capacidades destacan:

- editor visual para pipelines, de manera que por medio de drag-and-drop puedas crear tu pipeline; visualización de pipelines;
- analizador de pipeline que provee un diagnóstico y localiza problemas sin necesidad de estar buscando en las bitácoras;
- dashboard personalizado para solo ver los pipelines que te interesan;
- integración profunda con github.

https://jenkins.io/projects/blueocean

4 CONTENEDORES PARA MODERNIZAR APLICACIONES

Durante DockerCon 2017, Docker dio a conocer una iniciativa denominada "Modernize Traditional Applications" que básicamente consiste en envolver aplicaciones legadas en contenedores para moverlas del hardware que utilizan actualmente hacia infraestructura moderna administrada como nube híbrida. La promesa es que sin necesidad de cambiar código fuente se obtengan mejoras en utilización de recursos, escalabilidad, uptime y seguridad, entre otros. Para sustentar esta oferta, Docker ofrece una versión especial de su tecnología llamada Docker Enterprise Edition. Adicionalmente, empresas como Microsoft, Cisco y HPE están involucradas en la iniciativa para asegurar que sus productos soporten adecuadamente a las empresas que deseen tomar este camino para modernizar sus aplicaciones.

http://www.docker.com/MTA

DevDay 4 <WOMEN>

ÚNETE A LAS MUJERES QUE DESARROLLAN
SOFTWARE GRANDIOSO

5TA. EDICIÓN

9 DE JUNIO 2017, UNIVERSIDAD AUTÓNOMA DE GUADALAJARA

Evento Gratuito, Regístrate

http://devday4w.com

Selenium WebDriver en un Ambiente de Pruebas Continuas

Por Gilberto Sánchez

Selenium automatiza los navegadores. ¡Eso es todo! Lo que hagas con ese poder depende de ti. Principalmente, es para la automatización de aplicaciones web con fines de pruebas, pero ciertamente no se limita a eso. Las tareas aburridas de administración basadas en web pueden (¡y deben!) ser automatizadas. Una definición muy concreta y directa, pero vamos a ampliar un poco más la definición.

El texto anterior es la introducción en la página principal de SeleniumHQ [1]. Selenium es un conjunto de herramientas de código abierto que nos ayuda a automatizar acciones que un usuario puede realizar sobre aplicaciones web. Cada herramienta dentro de este conjunto tiene un enfoque diferente para apoyar el proceso de automatización de pruebas.

Los cuatro componentes de Selenium son:

1. Selenium IDE: es un entorno de desarrollo integrado para scripts de Selenium. Se implementa como una extensión de Firefox y permite grabar, editar y depurar pruebas. Este IDE incluye todo el Selenium Core, que permite grabar y reproducir de forma fácil las pruebas en el entorno real en el cual se ejecutarán. Se recomienda su uso para prototipos de pruebas, debido a que no es capaz de generar ciclos ni condiciones.

2. Selenium RC (Remote Control): es una herramienta para automatizar pruebas de interfaz de usuario (UI) de aplicaciones web. Consta de dos componentes: a) un servidor que actúa como proxy para controlar e interactuar con un navegador web. b) bibliotecas para crear programas para el servidor usando una amplia gama de lenguajes de programación. Fue la herramienta original de Selenium para pruebas web pero a partir de la versión 2 fue integrado con WebDriver y se prefiere usar este último.

3. Selenium WebDriver: también es una herramienta para automatizar pruebas UI de aplicaciones web pero implementa un enfoque más moderno y estable que Selenium RC. WebDriver, a diferencia de RC no utiliza un middleware sino que controla el navegador comunicándose directamente con él.

4. Selenium Grid: se especializan en ejecutar múltiples pruebas a través de diferentes navegadores, sistemas operativo y máquinas. Puede conectarse con Selenium Remote especificando el navegador, la versión del navegador y el sistema operativo que desee. Hay dos elementos principales: hub y nodos.

En este artículo nos enfocaremos en WebDriver. Como ya comentamos, es un framework de automatización web que permite ejecutar casos de prueba sobre distintos navegadores. Debido a que es posible utilizar lenguajes de programación para la creación de scripts de pruebas, podemos tener estructuras de control como condiciones y bucles para controlar el comportamiento. Algunos de los lenguajes soportados son: Java, C#, Python, Ruby, PHP y JavaScript.

ARQUITECTURA

A primera vista puede parecer que Selenium está manejando el navegador directamente desde nuestro código, sin embargo este proceso es un poco más complejo de lo que pareciera. La arquitectura de WebDriver está dividida en tres partes principales: lenguaje de vinculación, WebDriver API y drivers.

Para ver cómo es que interactúan las partes entre sí, digamos que se ha escrito el script de prueba usando Java (lenguaje de vinculación) para comunicarse con la API de WebDriver. El código generado va a emitir comandos a través del WebDriver wire protocol, el cual es un servicio REST capaz de interpretar dichos comandos. El driver es un ejecutable que básicamente escucha en un puerto de la máquina local cuando se ejecutan las pruebas y espera que los comandos entren. Una vez que los comandos son captados por el driver, estos son interpretados y ejecutados sobre el navegador.

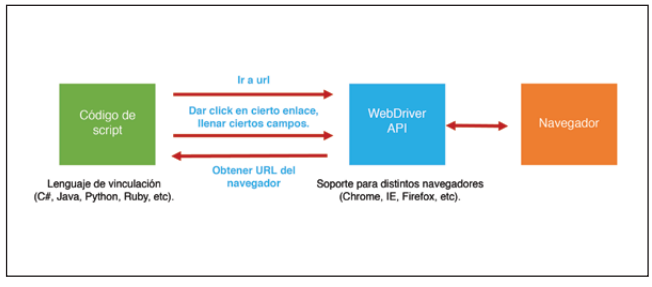


Figura 1. Arquitectura WebDriver.

Gilberto Sánchez Mares es Test Automation Engineer con experiencia de 5 años en diseño e implementación de proyectos de pruebas automatizadas utilizando varios frameworks y herramientas de automatización. gilberto.sanchez@upa.edu.mx

VENTAJAS Y DESVENTAJAS

La tabla 1 lista un comparativo de WebDriver contra otras herramientas que cumplen un propósito similar.

Característica	Selenium WebDriver	IBM Rational Functional Tester	HPE United Functional Testing
Lenguaje de scripting	Java, C#, Ruby, Python, Perl, JavaScript, PHP	Java y C#	VB Scripting
Tecnologías soportadas	Todas las tecnologías web	HTML, Java, AWT, SWT, Dojo	HTML, Java, .Net, WPF, SAP, Oracle
Soporte a navegadores	Internet Explorer, Chrome, Firefox, Safari, Opera	Internet Explorer, Chrome, Firefox	Internet Explorer, Chrome, Firefox
Soporte de ambientes	Windows, Linux, OS X	Windows y Linux	Sólo Windows
Soporte móvil	Sí	Sí	Sí
Manejo de repositorio de objetos	Bueno	Muy bueno	Excelente
¿Su uso tiene un costo?	No	Sí	Sí
Opción de Record & Play	Sí	Sí	Sí
Depuración	Depende del IDE	Muy bueno	Excelente
Soporte de frameworks	Data Driven, Keyword Driven, Modularity, Test Library Architecture e Híbrido	Data Driven, Keyword Driven, Modularity e Híbrido	Data Driven, Keyword Driven, Modularity, Test Library Architecture e Híbrido
Continuous integration	Posible usando Jenkins/ Hudson/Cruise Control/etc	Posible con Jenkins	Posible con ALM o Jenkins
Mecanismo de reporte	No tiene un mecanismo integrado	Mecanismo integrado	Mecanismo integrado
Soporte	Comunidad	Soporte de IBM	Soporte de HP

Tabla 1. Tabla comparativa

CONTINUOUS TESTING CON WEBDRIVER

Continuous Testing (CT) es en realidad una metáfora para un mecanismo de retroalimentación continua que impulsa la entrega de software funcional. La retroalimentación automatizada en cada punto de control es un disparador automático para el siguiente proceso en la cadena de entrega si la retroalimentación es para avanzar. Si la retroalimentación no avanza, el proceso se detiene inmediatamente y se toman medidas correctivas. Las organizaciones de TI tradicionales pueden acortar el camino para implementar CT reutilizando y realineando las capacidades de automatización de pruebas existentes.

Si se cuenta con un proceso de integración continua, la automatización de pruebas de UI es relativamente sencilla ya que se puede llevar a cabo con Selenium WebDriver y una herramienta de CI como Jenkins. El único requisito es utilizar un framework de pruebas unitarias como lo es TestNG, JUnit, NUnit, PHPUnit, etc. La figura 2 muestra un mapeo entre herramientas a través de distintas actividades y disciplinas del ciclo de desarrollo de software.

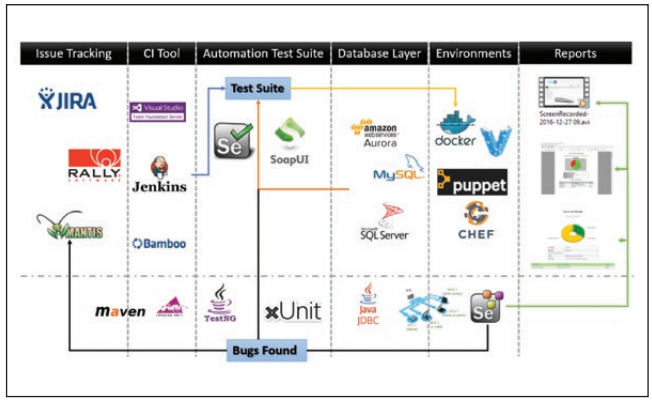


Figura 2. Selenium WebDriver dentro del proceso de integración continua.

EJEMPLO

A continuación, se presenta un ejemplo de cómo realizar pruebas dirigidas por datos (data driven testing) con el framework de pruebas unitarias TesNG. El ejercicio consistirá en probar el formulario de registro de usuario en la página Mercury Tours (http://newtours.demoaut.com). Todos los archivos y código usado para el ejercicio se encuentran disponibles en https://github.com/gsanchez-tiempodev/SoftwareGuruDemo

El caso de prueba que se ejecuta es el siguiente:

Descripción: Registrar usuario para la página Mercury Tours

Procedimiento:

1. Navegar a la página de Mercury Tours
2. Dar click en enlace REGISTER
3. Ingresar <First Name>
4. Ingresar <Last Name>
5. Ingresar <Phone>

6. Ingresar <Email>

7. Ingresar <Address>

8. Ingresar <City>

9. Ingresar <State>

10. Ingresar <Postal Code>

11. Seleccionar <Country>

12. Ingresar <User Name>

13. Ingresar <Password>

14. Ingresar <Confirm Password>

15. Dar Click en botón Submit

Resultado esperado:

Página con el mensaje: "Dear <First Name>, Thank you for registering. You may now sign-in using the user name and password you've just entered. Note: Your user name is <User Name>."

Cómo podemos ver, necesitamos llenar una cantidad considerable de campos en el formulario de registro, por lo que será útil alimentar los datos de prueba desde una hoja de cálculo. Así que preparamos un archivo FlightRegisterData.xls con una hoja llamada RegisterUser que contenga los datos mostrados en la figura 3, cuyas columnas corresponden a los campos que requerimos introducir en la forma de registro.

	A	B	C	D	E	F	G	H	I	J	K
1	FirstName	LastName	Phone	Email	Address	City	State	PostalCode	Country	UserName	Password
2	Gilberto	Sanchez	333-487-9876	gilberto@mail.com	Test Address 408	Guadalajara	Jalisco	20345	MEXICO	gsanchez	123
3	Lorena	Perez	477-567-9874	lorena@Yahoo.com	Lorenas Address 973	Campo Real	Madrid	21243	SPAIN	lperez	abc
4	Arturo	Castilo	932-934-7234	arturo@gmail.com	Downwoody Point 343	Atlanta	Georgia	34355	UNITED STATES	acastillo	castillo
5											

Figura 3. Datos de Prueba.

Nuestra clase principal llamada RegisterUsers.java, es donde se llamarán todos los métodos del negocio y algunos métodos comunes.

Para extraer los datos de nuestro archivo de Excel, creamos un arreglo bidimensional de objetos (Object[][]) que guardará los valores del Excel. Para iniciar el método que nos ayudará a la recolección de los datos, utilizamos la anotación @DataProvider y le damos un nombre, en este caso lo llamamos UserRegistration. Dicho método se apoya en una utilería para leer datos de nuestra hoja de Excel y vaciarlos en el arreglo de objetos que regresamos como resultado.

```
Object[][] testObjArray;
String testCaseWorkBook = System.getProperty("user.dir") + "\\testData\\FlightRegisterData.xlsx";

@DataProvider(name = "UserRegistration")
public Object[][] userRegister(Method m) throws Exception {
  return (testObjArray = ExcelUtils.getTableArray(testCaseWorkBook, "RegisterUser"));
}
```

Listado 1. Leer datos de prueba

Tenemos un método setUp con la anotación @BeforeTest que se encarga de abrir un navegador, ir al url de inicio y maximizar el navegador antes de que cualquiera de nuestras pruebas se ejecute. Al finalizar la ejecución queremos cerrar el navegador y terminar el driver que lo manipula, así que para ello creamos un método que llamamos tearDown y le ponemos la anotación @AfterTest. Antes de que se vuelva a llenar el formato de registro en cada iteración, deseamos que se le vuelva a dar click al enlace REGISTER; para realizar dicha acción creamos un método clickRegister con la anotación @BeforeMethod.

```
@BeforeTest
public void setUp(){
  navigateTo();
}

@AfterTest
public void tearDown(){
  getDriver().quit();
}

@BeforeMethod
public void clickRegister(){
  clickOnLink(LocatorType.LinkText, "REGISTER");
}
```

Listado 2. setUp, tearDown y clickRegister

El método contactInformation es el que se encarga de correr nuestra prueba y por ello lleva la anotación @Test. La firma de argumentos de este método es del tipo String ... dataProvider, la cual nos indica que acepta n número de argumentos (para este ejemplo serán 11). En este caso, vamos a estructurar los datos en 3 grupos: información de contacto (columnas 0 a 3), dirección de correo (columnas 4 a 8) e información de usuario (columnas

9 y 10). Separamos los datos y nos apoyamos en los métodos addContactInfo, addMailingInfo y submitUserInfo. Finalmente, hacemos un assert para validar que la página de resultado contiene el nombre del usuario.

```
@Test(dataProvider = "UserRegistration")
public void contactInformation(String ... dataProvider){
  String[] contactInfo = new String[4];
  String[] mailInfo = new String[5];

  //Store data for mail info and contact info
  for(int i = 0; i < 9; i++) {
    if(i>3) {
      mailInfo[i-4] = dataProvider[i];
    } else {
      contactInfo[i] = dataProvider[i];
    }
  }

  addContactInfo(contactInfo);
  addMailingInfo(mailInfo);
  submitUserInfo(dataProvider[9], dataProvider[10]);

  //Verify user name is displayed
  Assert.assertTrue(getElementText().contains(dataProvider[9]));
}
```

Listado 3. Método de prueba

La clase Common.java define métodos reutilizables de acciones que se pueden ejecutar sobre los objetos web tales como escribir en un text box, seleccionar un elemento de un combo box, navegar a la página web, etcétera. Estas acciones se logran por medio de métodos que implementa el driver de selenium. El listado 4 tiene un ejemplo. El contenido completo se puede ver en el repositorio.

```
// method to navigate into demo auto new tours
public void navigateTo(){
  getDriver();
  getDriver().get("http://newtours.demoaut.com/");
  getDriver().manage().window().maximize();
  getDriver().manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
}

// method to type into text box item
public void typeInTextBox(LocatorType locatorType, String locator, String textToType ){
  switch(locatorType.toString()){
  case "Name":
    getDriver().findElement(By.name(locator)).sendKeys(textToType);
    break;
  case "Id":
    getDriver().findElement(By.id(locator)).sendKeys(textToType);
    break;
  }
}
```

Listado 4. Algunos métodos en Common.java

Por su parte, Business.java contiene los métodos que siguen reglas de negocio, como lo puede ser un login, un alta de usuarios, etc. Un ejemplo es el método addMailingInfo, el cual utiliza los métodos definidos en Common.java para realizar el llenado de la información de correo del usuario que se va a registrar.

```
//Adding Mailing Information
public void addMailingInfo(String ... mailing){
  typeInTextBox(LocatorType.Name, "address1", mailing[0]);
  typeInTextBox(LocatorType.Name, "city", mailing[1]);
  typeInTextBox(LocatorType.Name, "state", mailing[2]);
  typeInTextBox(LocatorType.Name, "postalCode", mailing[3]);
  selectFromDropDown(LocatorType.Name, "country", mailing[4]);
}
```

Listado 5. addMailingInfo

Finalmente, en la figura 4 se muestran los resultados generados por TestNG al ejecutar nuestra prueba aplicando los distintos datos.

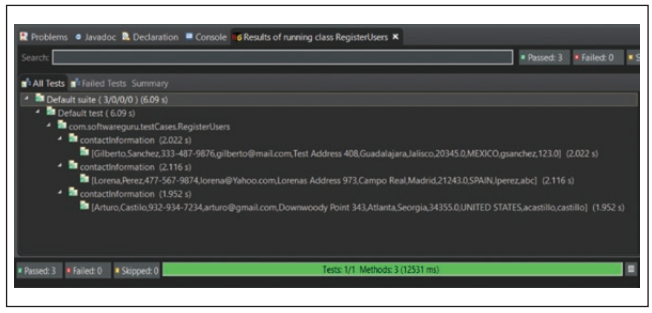


Figura 4. Resultados de ejecución generados por TestNG.

Referencias

[1] SeleniumHQ. http://www.seleniumhq.org

[2] "Introduction to Selenium". Guru99. http://swgu.ru/sk

[3] S. Honnamane & R. Bar. "Jumpstarting DevOps with Continuous Testing". Cognizant 20-20 Insights. http://swgu.ru/sl

¿Qué Ocurre con las Startups en Europa?

Por Edith Gómez

Muchos ya han dado el primer paso y han descubierto cómo comenzar una startup, e incluso hay quienes han participado en concursos para startups, pues han logrado conseguir dinero para su idea de negocio sin contactos. Y esto tiene sentido ya que el mundo de las startups, a nivel internacional —específicamente a nivel europeo— se mueve en un año que se vislumbra como prometedor.

Esto parece la continuación de un 2016 que, pese a ver reducidas sus rondas respecto a 2015, sí logró incrementar el volumen de inversión en más de un 7,8%, lo cual se traduce en 160.000 millones de euros, de acuerdo con las cifras arroja Tech.eu.

Europa es la segunda en fila, detrás del poderoso Estados Unidos, y también de una pequeña –pero activa- Israel, que parece centrar sus atenciones de grandes fondos desde este 2017.

Incluso, pensé al temido efecto del Brexit, aunque se tome con precaución la situación y el nivel de incertidumbre crezca cada día que pasa, el 21% de las pequeñas empresas con perfil tecnológico que reside en Reino Unido, tienen planificado abrir oficinas en algún país de la Unión Europea. Mientras se espera por saber cómo se moverán las decisiones políticas que apuntan al cambio, algunas cosas permanecen como están, y no se han producido migraciones masivas de emprendedores a zonas más tranquilas.

Pese a las circunstancias, se observa un panorama positivo, y el 2017 inicia bastante bien. A nivel nacional, se produjo la compra millonaria iniciando el año, por parte de Take-Two, de igual modo que Fever lograba hacerse con 5 millones de dólares, provenientes de dos fondos estadounidenses, y la catalana Syctl con 12 millones también de fondos extranjeros.

Europa y España están empezando a ser altamente atractivas para Estados Unidos: esto gracias a la menor competencia y a los precios más bajos de los que se podrían encontrar en Silicon Valley, lo cual incentiva a muchos inversores a encontrar el próximo unicornio made in Europe.

De esto se están dando cuenta los mismos inversores del continente. Pues, mientras la Unión Europea anunció un fondo de casi 10 millones de euros para impulsar empresas tecnológicas en Europa –cuyos fondos quizás no se destinen a Reino Unido-, y con dedicada atención a lo que sucede en un sur menos desarrollado, otros fondos se han destinado para fondear nuevas carteras.

El fondo creado por el fundador de Skype anunció un nuevo proyecto cuyos fondos eran de 765 millones de euros, centrados en el mercado europeo, y con el fin de encontrar a los futuros líderes emprendedores.

Aunque ya esto lo hicieron con BlaBlaCar, King, Zalando o Deliveroo, ellos siguen tras la pista, en sus fronteras europeas, para financiar a los posibles unicornios. Ya conocen los límites de la financiación en Europa y de la poca posibilidad que registran muchas empresas a la hora de querer irse al otro lado del Atlántico, lo cual no sucede a la inversa y es la razón por la cual ocupan los primeros puestos en un terreno que, aunque se perfila como fuerte, posee fuertes probabilidades de crecer.

Algo así le ha ocurrido a Project A, un fondo de venture capital con sede en Berlín, que siguió la huella del anuncio de Atómico y cerró su propia ronda de 140 millones de euros para invertir en lo mismo: grandes startups europeas centradas en servicios de atención al cliente, software y B2B.

La situación es que no serás los primeros ni los últimos en incursionar en el terreno de los grandes fondos a nivel europeo y se espera que en los meses que vienen se observen grandes anuncios por parte de los inversores, compradores y emprendedores a nivel europeo. Pues, el mejor posicionamiento de Europa está a la vuelta de la esquina.

Edith Gómez Benítez es Head of Online Marketing de http://gananci.com

Internet para **Cerrar Brechas**

Por Alejandra Lagunes

El 17 de mayo se celebra a nivel mundial el Día de Internet. Una fecha para recordar la poderosa herramienta que es Internet; un gran espacio de interacción que está transformando sociedades en todos los sectores: la producción, las relaciones, el acceso a la información y al conocimiento.

La digitalización impacta positivamente el crecimiento del Producto Interno Bruto (PIB), la innovación y la oferta de servicios públicos [1]. Por otra parte, Internet permite acceder a contenidos a los que hace pocos años no teníamos acceso. Tan sólo 90% de la información disponible hoy en día se generó los últimos 2 años.

Es cierto, entonces, que la red más grande del mundo ofrece múltiples oportunidades y beneficios; sin embargo, también es cierto que nuevos retos aparecen. Uno de los más importantes es lograr que Internet y las nuevas herramientas tecnológicas se conviertan en mecanismos igualadores, que permitan mejorar la calidad de vida de todas las personas. Alcanzar este objetivo se ha convertido en prioridad para los gobiernos y sociedades de todo el mundo.

En septiembre de 2015 la Asamblea General de la ONU adoptó la Agenda 2030 para el Desarrollo Sostenible. Se trata del documento que establece los objetivos de la humanidad para 2030 y que buscan erradicar la pobreza, fomentar la prosperidad y lograr mayor igualdad. No sólo eso, la Agenda 2030 reconoce a las TIC como un habilitador esencial para hacer frente a los grandes desafíos de la humanidad.

En México, el potencial de Internet para cerrar brechas es un tema que ha tomado cada vez más importancia y hoy, como nunca antes, está en la agenda de gobierno, sector privado y academia. En junio de 2013, se promulgó una Reforma en materia de Telecomunicaciones con la que el acceso a Internet se convirtió en un derecho constitucional y se definió el marco legal que establece mayor competitividad en la oferta de servicios de telecomunicaciones. El avance en el sector en los últimos tres años ha sido importante. Los precios de servicios de telecomunicaciones se han reducido más de 20%; mientras que los usuarios de Internet en nuestro país ya son cerca de 70 millones, lo que representa un incremento de cerca de 70% respecto a 2012.

Por otra parte, para fomentar la adopción y desarrollo de las TIC, bajo el marco de la Estrategia Digital Nacional, se han puesto en marcha diversos proyectos que fomentan que cualquier persona, sin importar condición social, edad o género, aproveche el potencial de Internet y las nuevas tecnologías. En primer lugar, se encuentra el Programa de Inclusión Digital, denominado @prende 2.0, que la Secretaría de Educación Pública puso en marcha. Este programa tiene 4 pilares: capacitación docente; una plataforma en línea con miles de materiales educativos; equipamiento y conectividad; y monitoreo y evaluación continúa. El objetivo final del @prende 2.0 es que la tecnología esté al servicio de profesores y estudiantes para que desarrollen habilidades digitales, sean creativos, colaborativos; que tengan las herramientas necesarias para ser competitivos en la sociedad del siglo XXI. Otro proyecto es CódigoX, cuyo propósito es impulsar que cada vez más mujeres y niñas se involucren en el área de las TIC, desde diversos ámbitos. Uno de sus componentes más relevantes es un programa de mentorías que acerca a mujeres líderes en áreas de ciencia y tecnología con jóvenes interesadas en acumular experiencia y aprendizaje en estas áreas. Además, por medio de CódigoX se han organizado talleres, conferencias y cursos, logrando que más de 800 niñas y mujeres sean parte de este proyecto. Por otra parte, desde 2015 se puso en marcha la Red Nacional de Puntos México Conectado. Esta Red se conforma por 32 Centros de Inclusión Digital, en los que se ofrecen talleres y cursos para fomentar la innovación, el emprendimiento tecnológico y el uso de las nuevas tecnologías.

Las acciones y programas mencionados son sólo algunos ejemplos de lo que que sucede cuando Internet se convierte en aliado del gobierno y de la sociedad para generar espacios de inclusión. Es una transformación que está empezando y que ya ha permitido que México mejore en diversos indicadores del mundo digital. De 2014 a 2016 avanzamos 16 lugares en el Índice de Servicios en Línea, al pasar de la posición 35 a la 19; avanzamos también 31 lugares en el Índice de Participación en Línea, pasando del lugar 45 al 14 de entre 193 países. En ambos casos, somos también líderes en América Latina.

Es claro que Internet permite construir mejores realidades; nuestro país vive un momento único en el que todos los sectores están sumando a la construcción de un México más moderno e incluyente. Es tarea de todos continuar con esa labor.

Referencias

[1] R. Katz. El ecosistema y la economía digital en América Latina. Fundación Telefónica, Editorial Ariel, CEPAL. Enero, 2015. http://swgu.ru/st

Alejandra Lagunes Soto Ruiz es Coordinadora de Estrategia Digital Nacional. Elabora, da seguimiento y evalúa periódicamente la Estrategia Digital Nacional; fomenta la adopción y el desarrollo de tecnologías de la información y comunicación; impulsa el gobierno digital; promueve la innovación, apertura, transparencia, colaboración y participación ciudadana para insertar a México a la sociedad del conocimiento.

SPINGERE

La empresa especializada en medición y estimación de Software

Alguna vez te has preguntado: ¿si lo que pagas por los proyectos de software es adecuado?, ¿si el tiempo/costo/esfuerzo que te dan como estimado para desarrollar un aplicativo es confiable?, ¿si hay alguna forma adecuada de determinar la productividad de un equipo de desarrollo?, ¿si es posible tener un control cuantitativo de tus proveedores y tus proyectos? ¿si estás contratando con las mejores condiciones para el Estado?

Existen varios estudios que presentan resultados no satisfactorios, que reflejan la falta de madurez de la disciplina de ingeniería de software, y en consecuencia los malos resultados en los proyectos desarrollados, por ejemplo, un estudio del Standish Group (2012) muestra que el desarrollo de proyectos exitosos es de menor al 40% , un estudio de McKinsey (2012) que indica que "en promedio, los grandes proyectos de TI se exceden en un 45% sobre el presupuesto y 7% sobre el tiempo, mientras que entregan 56% menos del valor esperado", otro ejemplo es el estudio de KPMG (2010) donde se menciona, entre otras cosas que "..un increíble 70% de las organizaciones han sufrido al menos un proyecto fallido en los 12 meses anteriores"

El The Global Information Technology Report (2016), proyecto especial del Foro Económico Mundial (www.weforum.org/gitr), establece que el mundo está entrando en la Cuarta Revolución Industrial, que es caracterizada por la convergencia de tecnologías digitales, físicas y biológicas, menciona claramente que esta Revolución no está definida por ningún conjunto de tecnologías emergentes, pero si por la transición de nuevos sistemas que serán construidos sobre la infraestructura de la revolución digital, siendo las tecnologías (TIC's) la columna vertebral de esta revolución.

Es claro, por tanto, los avances de la tecnología van a demandar la generación de una mayor cantidad de software, ¿se puede desarrollar software sin tener medidas de tamaño? La respuesta es sí, en la mayoría de los casos así se hace, sin embargo, esto fomenta que el desarrollo de software sea un arte, y no un proceso ingenieril, con resultados como los ya mencionados.

> "According to testimony by the Government Accountability Office last September, if were established more realistic baselines of requirements, cost, schedule and risk during project's planning phases, nearly half of canceled or over budget IT programs could be avoided. That would save $5.5 billion annually, according to a study made by Price Systems LLC, a software and consulting company in Mount Laurel, N.J., USA. The study consider 104 Government IT executives" (PMI, 2007)

Medir el software de manera formal mediante estándares, nos puede redituar en eficiencia del gasto y transparencia en la contratación, en tener mejores estimaciones de costos y gestionar mejor los proyectos tanto en el sector público como en el privado, además de blindar los procesos de compras, y dar cumplimiento a la normatividad para el caso de la Administración Pública Federal (APF).

SPINGERE presenta una oferta especializada en servicios de consultoría y de capacitación orientados a la medición, estimación y evaluación cuantitativa de proyectos de software, es la primera empresa especializada al respecto en México y América Latina, en idioma español, su oferta está basada en investigación aplicada desarrollada en México y presentada en distintas conferencias nacionales e internacionales.

Dentro de los beneficios que han obtenido nuestros clientes tanto del sector gobierno como de la iniciativa privada se encuentran:

- Certeza en la estimación de proyectos
- Eficiencia en uso del presupuesto y transparencia
- Gestión cuantitativa de las capacidades de desarrollo de software
- Mejor control de los proveedores
- Cumplimiento a normatividad (MAAGTICSI y LFMyN)
- Blindaje del gasto ante auditorías
- Mejoras en procesos de desarrollo de SW

La manera en que lo hacemos es a través de nuestra metodología de servicios denominada y registrada "Software Accountability Office - SAO ® ", que es una oficina integrada por un grupo de especialistas en gestión cuantitativa de proyectos, que te ayuda a dimensionar, estimar y evaluar proyectos de software, utilizando estándares internacionales como COSMIC (ISO/IEC 19761) o ISO 25000 apoyados de la investigación aplicada que se realiza.

Desde 2009, SPINGERE ha comercializado servicios relacionados con el método COSMIC, en abril 2014 fue designado como es el único "vendor" autorizado por el "Common Software Measurement International Consortium (COSMIC)" para proporcionar consultoría, capacitación e implementación de ISO/

Dimensionamiento y Estimación Profesional de Software

IEC 19761, así como realizar investigación al respecto en México, situación que se ratificó en febrero de 2017, ambas ocasiones mediante una carta emitida por el Consorcio COSMIC y apostillada por la Secretaría de Relaciones Exteriores.

¿Necesitas tener certeza de lo que pagas por el software? ¿Deseas disponer de elementos cuantitativos que te permitan negociar las propuestas presentadas por tus proveedores? ¿Te gustaría tener un mejor control de tus proveedores y tus proyectos? ¿Quieres saber si estás contratando con las mejores condiciones para el Estado? ¿deseas blindar tus contratos de fábrica de software respecto del gasto ejercido?, Contáctanos y conoce nuestra oferta de servicio y los beneficios que puede traer a tu organización en www.spingere.com.mx

En mayo de este año se realizará el 2do Congreso Nacional de Medición y Estimación de Software (CNMES17), durante el congreso los ponentes más representativos a nivel Nacional e Internacional, expondrán sus experiencias basadas en las últimas tendencias en medición y estimación de software con la finalidad de enriquecer el conocimiento de los asistentes y orientarlos para un eficiente desarrollo del software. Este esfuerzo es promovido por la Asociación Mexicana de Métricas de Software (AMMS), SPINGERE y el Consorcio COSMIC, quienes han concentrado un grupo de expertos en los temas de Medición y Estimación de Software, así como usuarios de estos métodos, con la intención de difundir la teoría y la práctica a través de la participación en conferencias.

Entre los ponentes confirmados se encuentran: Dr. Alain Abran (Chairman de COSMIC), Frank Vogelezang (presidente de COSMIC), Dr. Francisco Valdés Souto (Profr. Asociado C de la FC-UNAM y Fundador de la AMMS). La información completa del evento la encuentras en www.cnmes.mx

FINTECH

Integración de finanzas y tecnología

Por Héctor Cárdenas

Fintech es un concepto que surge a partir de la unión de las palabras finanzas y tecnología, y consiste en un fenómeno relativamente reciente de empresas que tratan de cambiar la industria financiera a través de la incursión de soluciones online para los procesos financieros.

La tecnología financiera presenta diversas ventajas frente al sistema financiero tradicional: bajos costos, comisiones menores y acceso fácil (siempre y cuando se cuente con Internet). Esto explica el explosivo crecimiento de numerosas startups, sin embargo son los millennials quienes han impulsado este crecimiento debido a su predilección por el uso de tecnologías en operaciones financieras, incluyendo las bancarias.

El sector financiero tradicional poco a poco ha ido reconociendo la necesidad de cambiar sus procesos a un ámbito más tecnológico reconociendo las ventajas que conlleva dicho cambio. En Europa, la creatividad y arrojo de las startups permite un desarrollo más acelerado, sin embargo, en América Latina y el Caribe se encuentra el terreno más fértil para estas empresas.

México tiene potencial para convertirse en un líder del sistema financiero global, siempre y cuando se determine a resolver los desafíos que conlleva esta transformación: colaboración del sector financiero tradicional y una regulación que brinde reglas a las empresas y proteja a los usuarios.

En la actualidad, sólo dos bancos utilizan los servicios de una fintech ya que la mayor parte de sus inversiones se canaliza a la banca móvil o en renovar sus instalaciones. La urgencia para llevar a cabo esta integración recae, principalmente en las startups y el gobierno que requiere ordenar este sector.

A través de la regulación se podrá dar un crecimiento y fortalecimiento a este sector para mantener de forma equilibrada todas las ramas del fintech. Uno de los principios básicos de la Ley Fintech radicará en regular cada sector de acuerdo con el servicio que presta y los riesgos inherentes a cada actividad. De acuerdo a la propuesta de ley sobre fintech, existen cuatro vertientes: financiamiento colectivo (crowdfunding), activos digitales, dinero electrónico y plataformas innovadoras.

No obstante, a pesar de la regulación gubernamental, el sector financiero necesita colaborar. Dicho ecosistema requiere refrescar y agilizar los modelos de negocio para que los usuarios se sumen más fácilmente a los procesos económicos del país. No hay que olvidar que en México, sólo el 68% de los adultos tiene un producto financiero (cifras de la Encuesta Nacional de Inclusión Financiera), no obstante, de esa cifra, el 92% prefiere pagar en efectivo cuando realiza alguna compra. Esto se debe, principalmente, a una percepción de desconfianza por la seguridad de su información y a sentir que es complicado su uso, especialmente en compras online.

En este contexto están surgiendo empresas mexicanas de tecnología enfocadas en soluciones de pagos digitales, producto de una fuerte necesidad en el sector financiero en este ámbito, y que están creando herramientas y sistemas para la recepción de pagos en línea con base en innovaciones tecnológicas de primer nivel.

A pesar de los esfuerzos de estas startups, el camino para la integración financiera en nuestro país aún es largo. El verdadero desafío para las políticas públicas y las empresas que han incursionado en el negocio electrónico se encuentra en crear oportunidades para la población que reflejen la apertura y accesibilidad de la red, y al mismo tiempo desarrollar las bases de una cultura financiera que beneficie a los consumidores y les de las herramientas necesarias para tomar decisiones de compra.

Por otro lado, mientras las instituciones financieras tradicionales y el gobierno esperan o tratan de avanzar, las startups fintech no se detienen y continúan innovando con la finalidad de sumar más y más mexicanos a la economía digital del presente.

Héctor Cárdenas es Co-Fundador y CEO de Conekta.

Riesgos y Retos que ENFRENTA FINTECH

Por Adrián Sánchez

Actualmente, la tecnología se ha adaptado para afrontar las necesidades de la sociedad en la que vivimos, facilitando, potenciando y economizando la satisfacción de las mismas. Poco habíamos visto de la innovación tecnológica en los procesos financieros, sin embargo, la demanda de los sectores no bancarizados y la posibilidad que la tecnología brinda a nuevos jugadores para tomar una rebanada del pastel financiero, inició una reacción en cadena de startups en este sector.

Estas startups han incursionado en el sector financiero logrando crear un nuevo sector denominado Fintech para cubrir necesidades como financiamientos, sistemas de pagos, remesas, asesoría financiera, comparadores de productos, incluso productos totalmente revolucionarios como bitcoin y blockchain. En este sentido, las Fintech están contribuyendo con la inclusión financiera, no sólo en México sino a nivel global ya que su mercado principal es el sector no bancarizado o sub-bancarizado, lo cual les permite alcanzar sectores de la población donde, por ejemplo, no se cuenta con una sucursal bancaria tradicional, y al ofrecer en su mayoría servicios disponibles en línea facilitan el acceso a los servicios financieros tradicionales y también a los mecanismos de financiamiento alternativos.

Si bien es cierto que los consumidores han logrado acceder a una mayor y más diversa cantidad de servicios financieros, también lo es el hecho de que en medio de esta diversificación de actividades y procesos existen riesgos que se deben evaluar, gestionar y reducir. En este sentido, los riesgos mayormente identificados en el sector Fintech son las oportunidades que el lavado de dinero y el fraude puede hallar en los procesos no regularizados de estas innovaciones financieras.

Un riesgo radica en que las startups en este espacio no cuentan con la experiencia necesaria en el manejo de riesgos como prevención de fraudes y prevención de lavado de dinero. Si bien cuentan con toda la experiencia en la parte tecnológica, no así en la correcta aplicación de políticas y procedimientos que les permitan hacer una debida diligencia de los clientes y así conocer a la perfección con quien hacen negocios o a quien otorgan crédito.

Un buen ejemplo de los riesgos intrínsecos en los procesos de las fintech es el caso del crowdfunding, donde se establecen mecanismos de financiamiento colectivo para determinado proyecto a cambio de un rendimiento. Si no se cuenta con controles estrictos de control de riesgos y debida diligencia, nos encontramos con la incertidumbre de no saber cómo se trabajaron estos recursos, quienes son los mediadores y quienes los beneficiarios finales. Esta incertidumbre de la que hablamos, permite que por un lado los fondeadores públicos que invierten su dinero o lo donan sean víctimas de fraude y, por otra parte, que a través de los procesos antes descritos los lavadores de dinero logren la integración del dinero sucio al sistema financiero legal.

Por lo anterior, el uso de la tecnología en este sector es clave y las soluciones que ayudan con la administración y filtrado de grandes volúmenes de datos son de suma importancia. Por ejemplo, por medio de analítica de big data es posible realizar análisis complejos de comportamiento de clientes o la identificación de los mismos. Además, para combatir posibles casos de robo de identidad y aprovechando los mecanismos que generalmente utilizan las fintech a través de medios online, existen soluciones que permiten desde un Smartphone, tableta o una computadora verificar con más de 50 pruebas a través de la toma de una imagen si una identificación es válida o no. En este sentido, las prestaciones de la innovación tecnológica se vuelven salvaguardas de sí mismas con el desarrollo de software de prevención de riesgos. Sin olvidar que este nuevo nicho financiero requiere de una regulación apropiada que permita su óptimo funcionamiento y servicio a los consumidores.

Adrián Sánchez es Director de Soluciones para Prevención de Lavado de Dinero y Cumplimiento de LexisNexis® Risk Solutions Latinoamérica.

Ecosistemas de SERVICIOS FINANCIEROS DIGITALES

Por Bjorn Cumps

La innovación tecnológica está transformando el sector financiero. Una cantidad importante de las innovaciones se están enfocando en la desintermediación, permitiendo a las personas e instituciones interactuar directamente sin necesidad de hacerlo a través de los operadores financiero tradicionales. Atestiguamos una nueva ola de democratización de los servicios financieros, brindando a los consumidores (o al menos a un segmento de ellos) fácil acceso a servicios que antes estaban fuera de su alcance.

PANORAMA FINTECH

La figura 1 muestra un panorama del qué y cómo de fintech; es decir, en qué se están enfocando las startups y cómo lo están haciendo [1]. El círculo externo muestra las funciones centrales de los servicios financieros que están siendo afectadas: pagos, seguros, depósitos y préstamos, fondeo, gestión de inversiones, aprovisionamiento de mercados.

Figura 1. Panorama fintech.

En cada una de estas funciones se puede detectar clusters de innovación, por ejemplo crowdfunding o préstamos alternativos (alternative lending).

El "cómo" se está innovando se captura a través de 6 temas que cortan a través de las funciones centrales e impactan distintos clusters de innovación:

1. Streamlined infrastructure involucra nuevas maneras de acceder, analizar y agregar datos. Esto incluye criptomonedas, tecnología blockchain, y análisis de corrientes de datos abiertos.

2. Automatización de actividades de alto valor se refiere a algoritmos y tecnologías para automatizar actividades manuales. Por ejemplo, automatización de transacciones bursátiles o automatización de la gestión del portafolio de inversión.

3. Reducción de intermediación consiste en reducir o eliminar a los intermediarios tradicionales, por ejemplo conectando directamente a prestatarios y prestamistas.

4. Rol estratégico de los datos se refiere al análisis de big data para entender mejor al negocio y los clientes. Un ejemplo es la utilización de datos sociales para el perfilamiento de clientes y análisis crediticio.

5. Productos especializados de nicho son típicamente resultado de la entrada de nuevos jugadores que se especializan en cierto nicho. Un ejemplo son los proveedores de gateways de pago.

6. Empoderamiento del cliente significa dar a los clientes acceso a activos e información que previamente estaban fuera de su alcance; por ejemplo, que el cliente pueda directamente realizar transacciones bursátiles vía internet.

Tomando en cuenta la taxonomía mostrada en la figura 1, vemos que la mayoría de los startups fintech se enfocan en una sola función y cluster de innovación, pero típicamente combinan varios de los temas identificados. Por ejemplo, la plataforma de préstamos Kabbage se enfoca en el segmento de PyMEs que buscan entre 2 mil y 100 mil dólares de capital de trabajo, y utiliza análisis de datos en tiempo real de fuentes como eBay, Amazon y PayPal para calificar el préstamo; así que recurre tanto a la especialización de nicho como al rol estratégico de datos. Las fintech startups ciertamente representan amenazas puntuales para los jugadores tradicionales del sistema bancario, pero el verdadero potencial está en la combinación de fuerzas entre distintos jugadores.

EL ROL DE LOS BANCOS

¿Será que el sector financiero en el futuro estará dominado por empresas de tecnología, relegando a los bancos tradicionales a ser meros operadores de servicios de infraestructura? La mayoría de los bancos está consciente de este riesgo y está buscando hacer algo al respecto, embarcándose en iniciativas de innovación digital; lo hacen aprovechando sus propias fortalezas y con una postura relativamente abierta hacia nuevas alianzas con empresas de tecnología. Estos bancos están creando incubadoras de startups fintech, así como colaborando e invirtiendo en programas de innovación y aceleradoras externas; también están construyendo mecanismos para sensar la escena fintech y entender qué es lo que los consumidores buscan en dichas startups.

Así que a diferencia de hace un par de años, el sentido de urgencia ya no es el problema en las instituciones bancarias tradicionales. Pero, de acuerdo con The Economist Intelligence Unit [2], los bancos están encontrando otros retos tecnológicos para mantener el paso a las fintech startups, tales como deuda técnica, estructuras rígidas de gobierno corporativo, insuficiente talento técnico interno, y falta de una cultura interna de innovación. A estos retos se agrega la carencia de una visión estratégica clara en el espacio digital, y el constante peligro de violaciones en la seguridad de información.

Los bancos tradicionales requieren balancear sus inversiones entre mantener sus sistemas legados centrales y construir nuevos productos digitales. Este es un gran reto, ya que se requiere operar con excelencia los sistemas legados al mismo tiempo que se innova y construye productos digitales de nueva generación; este fenómeno se conoce como una "organización ambidiestra" [3]. Es así que la principal amenaza para los bancos no es externa sino interna. Las fintech startups no son su principal amenaza; al contrario, ofrecen nuevas formas de trabajar que los bancos no habían considerado. El verdadero peligro de los bancos está en su dificultad adaptarse rápidamente al cambio, abrirse y determinar con quién colaborar, a quién adquirir, a quién ignorar y con quién competir. Esto determinará su modelo de negocio hacia los próximos años. La fintech puede ser vista por los bancos como un riesgo o como una oportunidad para revolucionar el sector financiero; el principal obstáculo es su propia organización interna.

APERTURA Y COLABORACIÓN

En este contexto, el éxito de los bancos dependerá de qué tan bien puedan integrar sus servicios en las actividades cotidianas de sus clientes, lo cual involucra cambiar de proveer productos bancarios rígidos hacia atender de manera fluida las necesidades financieras de sus clientes. Esto implica establecer alianzas no solo con otros proveedores de servicios financieros, sino también con otro tipo de organizaciones, como por ejemplo de salud, turismo, telecomunicaciones, comercialización y gobierno, entre otros.

Ilustremos esto con un caso del Commonwealth Bank of Australia (CommBank). Este banco construyó estableció alianzas con instituciones del sector inmobiliario para ofrecer a sus clientes que están buscando casa acceso rápido y sencillo a información tal como el precio de una propiedad, historia de transacciones, información demográfica del vecindario, la cual es enriquecida con información que el banco tiene del cliente (tasa de interés que el banco puede ofrecer, monto de los pagos). CommBank entiende que comprar una casa es una decisión emocional; o te enamoras de la casa o no. Así que cuando lo haces, el banco está ahí para ayudarte a cumplir tu sueño (similar a como las agencias de automóviles te ofrecen un préstamo ahí mismo cuando decides comprar un auto). CommBank busca adaptarse al "viaje del cliente"; reduce la búsqueda de préstamos para casas y bloquea a la competencia. Logra esto haciendo equipo con agencias de bienes inmobiliarios para brindar un servicio integrado a sus clientes.

De acuerdo con Peter Weill [4], quien dirige el Centro de Investigación de Sistemas de Información en la MIT Sloan School of Management, las empresas exitosas del futuro se enfocarán en dos aspectos principales:

1. ¿Qué tanto conoces a tu cliente?

2. ¿Qué tan bien funciona tu negocio en constelaciones de ecosistemas?

La figura 2 muestra un mapa de modelos de negocio con cuatro sectores derivados del comportamiento a través de los dos ejes mencionados: conocimiento del cliente y diseño del negocio.

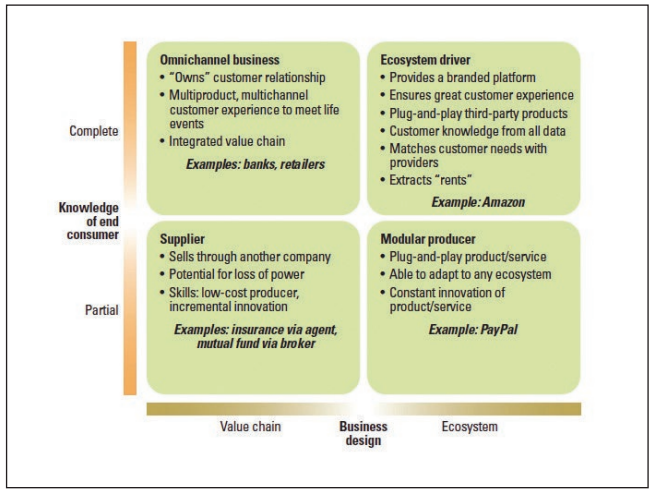


Figura 2. Modelos de negocio de la era digital

Como lo indica la figura, los bancos son un negocio omnicanal, ya que buscan adueñarse de la relación con el cliente y lo atienden por medio de distintos canales. Por otro lado, las fintech startups típicamente son productores modulares, insertándose con servicios "plug-and-play" en ecosistemas.

El reto de los bancos es lograr moverse de negocios omnicanal a líderes del ecosistema. El éxito dependerá de su capacidad para combinar sus activos digitales con los de terceros. Las alianzas digitales más exitosas están construidas sobre plataformas de ecosistemas digitales, así que los bancos requieren exponer sus activos digitales como servicios utilizables por terceros.

CONCLUSIÓN

El ecosistema de servicios financieros del futuro será abierto. Las organizaciones tradicionales de servicios financieros colaborarán con fintech startups y grandes empresas de tecnología, así como con empresas de otros sectores para construir servicios de valor agregado para los clientes; la competencia se dará entre ecosistemas, más que entre empresas individuales.

La versión original de este artículo apareció en el Cutter Business Technology Journal de noviembre 2016 (http://swgu.ru/ss) bajo el título "Toward Digital Financial Services Ecosystems". El texto que aparece aquí fue traducido y editado por Software Guru con el permiso de Cutter Consortium México.

Referencias

[1] "The Future of Financial Services: How Disruptive Innovations Are Reshaping the Way Financial Services Are Structured, Provisioned and Consumed." World Economic Forum, 2015. http://swgu.ru/sn

[2] "The Disruption of Banking." The Economist Intelligence Unit, 2015. http://swgu.ru/so

[3] C.A. O'Reilly, et al. "The Ambidextrous Organization." Harvard Business Review, abril 2004. http://swgu.ru/sq

[4] P. Weill, S. Woerner. "Thriving in an Increasingly Digital Ecosystem." MIT Sloan Management Review, verano 2015. http://swgu.ru/sr

El Dr. Bjorn Cumps es profesor de Innovación de Servicios Financieros en la Vlerick Business School, en Bélgica.

FINTECH FOR INCLUSION
en América Latina

Por Fermín Bueno

Desde Finnovista, organización dedicada a impulsar el Fintech en América Latina y España, estamos siendo testigos de cómo este sector se ha convertido en una de las mayores oportunidades para los emprendedores tecnológicos e inversores afines a las tesis de disrupción tecnológica en el mundo.

Estamos convencidos que 2017 será el año en que el Fintech se consolidará como la más potente y más veloz palanca para erradicar la exclusión financiera en América Latina, dejando atrás a otras iniciativas clásicas desplegadas por microfinancieras y entidades financieras tradicionales. A través de nuestros radares de innovación hemos visto cómo más de 1,000 emprendimientos Fintech en América Latina están innovando en todos los ámbitos de las finanzas, desde pagos, préstamos, crowdfunding o herramientas de gestión hasta seguros o gestión de patrimonio. Pero hay una tendencia muy destacable, y es que el 40% de estos están enfocados en servir los clientes invisibles: a los no-bancarizados y sub-bancarizados, tanto consumidores como micro, pequeñas y medianas empresas. Estas startups Fintech, que hemos llamado 'Fintech for Inclusion', ya están captando cientos de millones de dólares de financiación y está acelerando la innovación de soluciones inclusivas a ritmos nunca vistos hasta ahora en la industria financiera tradicional. Todo apunta a que en los próximos años veremos cómo esta cifra de financiación, que llega para servir a los no-bancarizados y sub-bancarizados de América Latina, se verá multiplicada por un factor cercano a 10, convirtiendo al Fintech for Inclusion en una de las más dinámicas oportunidades de inversión a nivel internacional.

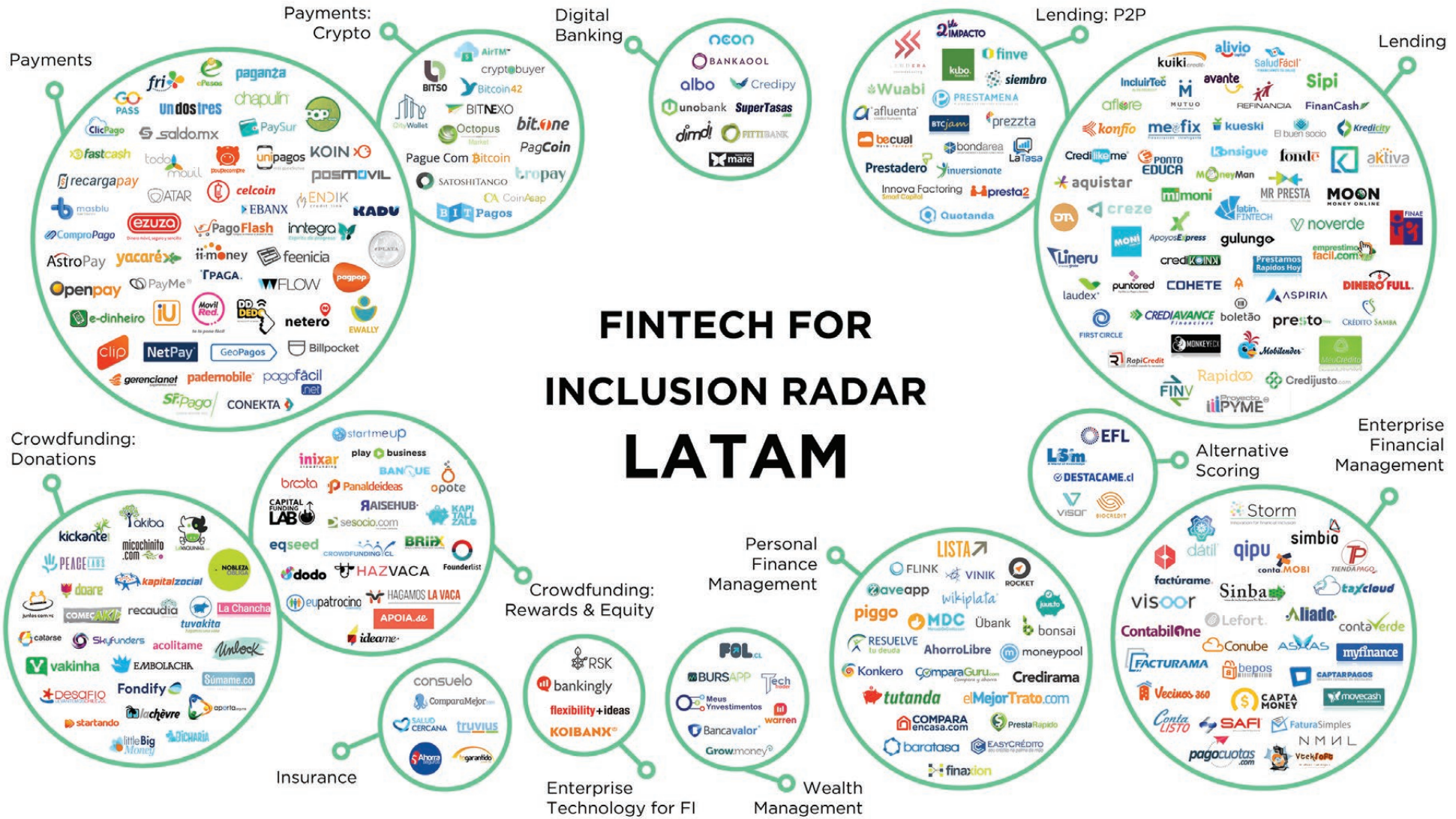
Los menores costos para la distribución digital de servicios financieros, así como la mejora en el desarrollo de evaluaciones de riesgo han facilitado el surgimiento de servicios orientados a la base de la pirámide de manera más accesible. No es fortuito, por lo tanto, que diversos gobiernos en la región estén considerando el desarrollo de las Fintech como uno de los principales pilares para la disminución de la exclusión financiera.

El diagrama en la siguiente página muestra el Fintech for Inclusion Radar de América Latina, donde hemos identificado 271 startups con propuestas de valor orientadas hacia consumidores o pymes con pobre o nulo acceso a servicios financieros. Estas empresas se distribuyen en 10 grandes segmentos, sobresaliendo en primer lugar el de Préstamos (Lending) con 27% de las startups identificadas. El acceso al crédito, y en general a mecanismos de financiación alternativa, adquiere mayor relevancia al ser una necesidad latente de los sectores relegados por las entidades financieras tradicionales. En este contexto, vale la pena señalar que el tercer mayor segmento de este radar está conformado por las plataformas de Crowdfunding con 18%, las cuales mediante donaciones y otorgamiento de recompensas o acciones, facilitan el acceso a recursos financieros que de otra forma no estarían disponibles para consumidores y pymes. Así, de manera conjunta estos dos segmentos contabilizan 45% del total de las startups destacadas en este radar.

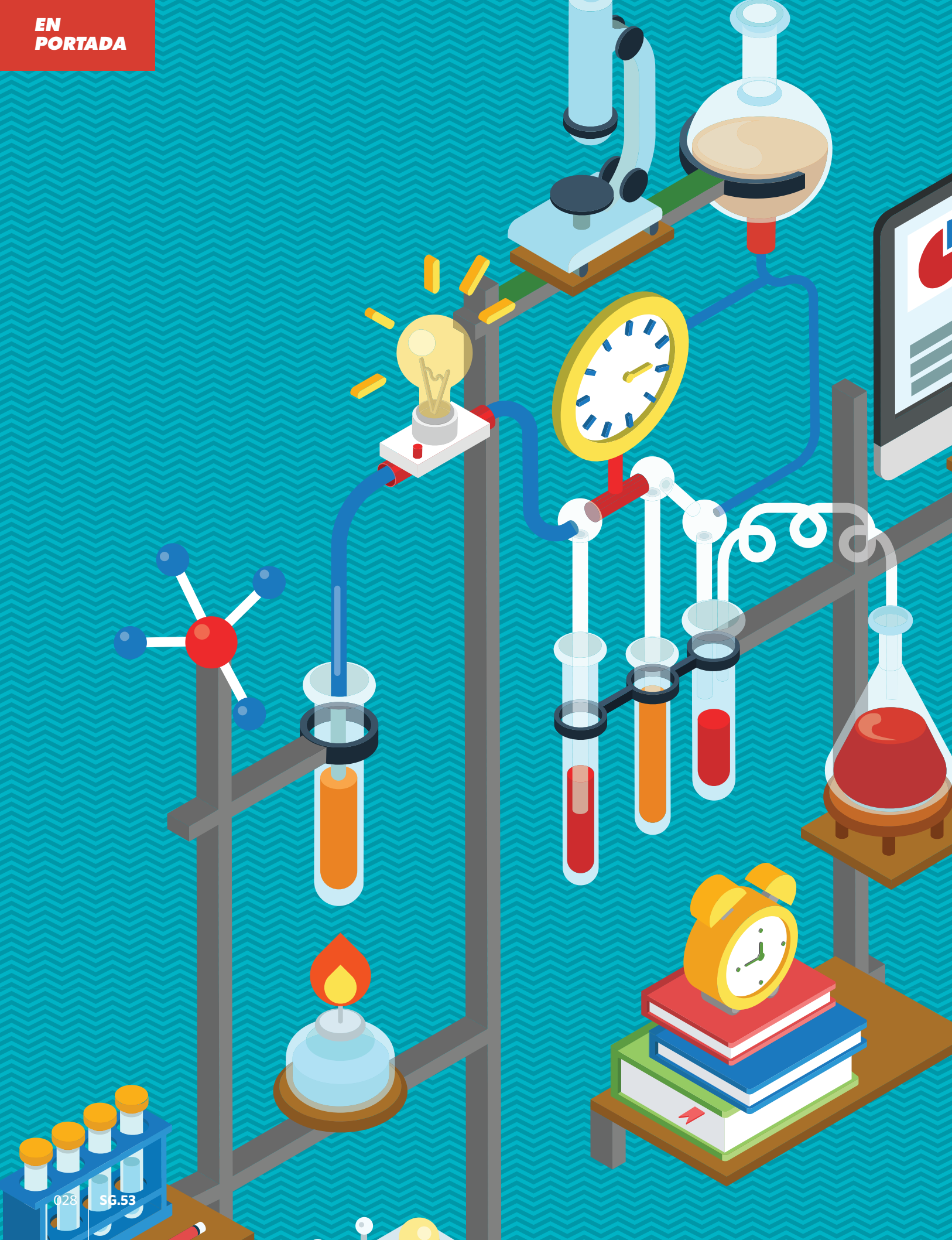
Hemos llegado a la tormenta perfecta en América Latina: ecosistemas de emprendimiento tecnológico con masa crítica y en pleno funcionamiento, talento joven y experto en servicios financieros y tecnologías digitales, inversores de capital riesgo nacionales e internacionales apostando por la región, tecnologías digitales baratas y ubicuas en manos de la amplia mayoría de la población, gobiernos ávidos de que la innovación llegue a los no-servidos, una creciente y joven clase media y soluciones innovadoras que no se encuentran limitadas por la infraestructura legacy. La tendencia es clara: América Latina está a las puertas de un "digital leapfrogging" que permitirá poner en las manos de amplios segmentos de la población soluciones financieras digitales a bajo coste y de manera ubicua, que llevará a la región a erradicar la exclusión financiera.

Fintech va camino de convertirse en el mayor dolor de cabeza de las entidades financieras tradicionales de la región. En 2016, consultoras como PwC y McKinsey profetizaron que hasta el 30% de los ingresos de la industria tradicional está en riesgo debido a las Fintech. En 2017 esperamos empezar a recibir los primeros mensajes por parte de analistas financieros haciendo referencia a las Fintech como la razón de pérdida de beneficios en el sector bancario. Las entidades financieras y las microfinancieras de la región que se hayan preparado cultural y tecnológicamente durante estos últimos años para la disrupción digital se verán beneficiadas por la colaboración con las Fintech, y los que no hayan hecho sus deberes quedarán en posiciones de mercado débiles o irrelevantes.

Fermín es cofundador y Managing Partner de Finnovista, donde ha liderado la creación de una innovadora plataforma Fintech formada por una red colaborativa a través de competiciones de startups, programas de aceleración, eventos Fintech y programas de Open Innovation patrocinados por diversas organizaciones en Europa, Estados Unidos y Latinoamérica.



Fintech for Inclusion Radar LATAM 21/12/2016
http://www.finnovista.com/fintechforinclusionradarlatam



INTEGRACIÓN CONTINUA

EL PRIMER PASO

Por Pedro Galván

El desarrollo de software está lleno de mejores prácticas de las que frecuentemente hablamos, pero rara vez hacemos. Uno de estos casos es el de tener un proceso automatizado para ensamblar y probar versiones ejecutables de nuestro software, de manera que el equipo de desarrollo pueda construir y probar varias veces al día el software en que están trabajando. Este concepto es conocido como "integración continua" y es una de las prácticas de Extreme Programming, sin embargo no es necesario estar siguiendo Extreme Programming para aplicarla.

**Nota: a lo largo de este artículo menciono frecuentemente el concepto de "integrar" software. Esto se refiere al proceso específico de generar una versión ejecutable de un software, que puede involucrar compilar el código fuente, empaquetarlo, incluir archivos de configuración, etcétera; es lo que en inglés se expresa como "to build software". Adicionalmente, mencionaremos el producto resultante de este proceso, que en inglés se conoce como "software build", y en español lo estaré llamando "versión ejecutable".*

¿QUÉ ES?

Martin Fowler es uno de los autores más reconocidos en el ámbito de prácticas ágiles y define la integración continua de la siguiente manera:

"La integración continua es una práctica en la que miembros de un equipo integran su trabajo frecuentemente, típicamente cada persona integra al menos una vez al día, generando varias versiones por día. Cada versión ejecutable es verificada por un sistema automático de integración y pruebas para detectar errores de integración lo más rápido posible."

Implementar esta práctica exitosamente involucra ciertos requisitos:

- Tener un repositorio maestro donde esté disponible todo el código fuente y del que cualquier integrante del equipo pueda obtenerlo.
- Automatizar el proceso de integración para que cualquier persona pueda generar una versión ejecutable del sistema a partir del código fuente.
- Automatizar las pruebas para que sea posible ejecutar la matriz (suite) de pruebas en cualquier momento con un solo comando.
- Asegurar que cualquiera puede obtener el ejecutable más reciente y tener la confianza de que es la mejor versión hasta el momento.

Todo esto requiere bastante disciplina y requiere un esfuerzo significativo para introducirlo a un proyecto, pero una vez habilitado es sencillo mantenerlo y brinda grandes beneficios.

BENEFICIOS

Estos son algunos beneficios puntuales de la integración continua:

- Reducir problemas de integración.
- Mejorar la visibilidad del estatus del producto de software.
- Acelerar la detección de fallas.
- Disminuir el tiempo dedicado a depurar errores.
- Evitar la espera para averiguar si un código funciona.

En resumen, los beneficios de la integración continua consisten en "resolver problemas rápidamente". Dado que el software es integrado frecuentemente, cuando se encuentra un error típicamente no es necesario retroceder mucho para descubrir donde se introdujo la falla. En comparación, cuando un equipo no sigue una estrategia de integración continua los periodos entre integración son largos y la base de código es muy distinta entre cada integración por lo que cuando se encuentran errores hay que revisar mucho más código, lo cual requiere un mayor tiempo y esfuerzo.

En palabras de Fowler, "la integración continua no elimina los defectos, pero si hace que sea mucho más fácil encontrarlos y corregirlos".

¿CÓMO SE HACE?

Aunque implantar una estrategia de integración continua no es algo trivial, sí es algo que está al alcance de cualquier organización que desarrolla software sin importar su tamaño, presupuesto o tecnología utilizada. Más que nada es cuestión de que el equipo entienda las prácticas y sea disciplinado.

Comencemos por las prácticas en las que se basa la integración continua [2]:

- Mantener un repositorio único del código fuente.
- Automatizar el proceso de integración (build).
- Hacer que el producto de software se pueda probar a sí mismo.
- Cada que se generan nuevas versiones en el repositorio ("hacer commit") se debe generar una nueva integración en una máquina designada para ello (integration machine).
- Mantener rápido el proceso de integración.
- El ambiente de prueba debe ser un clon del ambiente de producción.
- Los integrantes del equipo deben poder obtener fácilmente la versión ejecutable más reciente.
- Todos deben poder conocer el estatus en cualquier momento.
- Automatizar el despliegue.

Vale la pena aclarar que aunque la primera práctica indica mantener un repositorio único de código fuente, esto no significa que se tenga que usar un sistema de control de versiones centralizado. Hay que recordar que en un sistema de control de versiones distribuido (DVCS) también hay un repositorio único, es simplemente que tiene varias copias distribuidas.

Una vez que hemos entendido las prácticas y principios podemos entrar a actividades específicas. A continuación describo un posible flujo de actividades para implementar dichas prácticas:

1. Cada desarrollador obtiene una copia del código en su espacio de trabajo privado.

2. Cuando un desarrollador termina su trabajo, envía sus cambios al repositorio.

3. El servidor de integración continua monitorea el repositorio y detecta los cambios automáticamente.

4. El servidor (de integración continua) integra el sistema y corre las pruebas unitarias y de integración.

5. El servidor publica los artefactos ejecutables.

6. El servidor asigna una etiqueta a la nueva versión y su ejecutable.

7. El servidor informa al equipo el resultado de la integración.

8. Si la integración o pruebas fallan, el servidor alerta al equipo y el equipo lo resuelve lo antes posible.

9. Se continúa integrando frecuentemente a lo largo del proyecto.

Para que todo esto funcione se pueden establecer políticas que los integrantes del equipo deban cumplir, tales como:

- Registrar (commit) su código al menos una vez al día.
- No registrar código con errores.
- No registrar código sin probar.
- No generar nuevas versiones con código que no funciona.
- No pueden terminar sus actividades del día hasta haber registrado su código y que el sistema se integre exitosamente.

Algunos equipos generan rituales alrededor de estas políticas para ellos mismos controlar que se cumplan sin necesidad de supervisión de algún coordinador o gerente.

ENTREGA Y DESPLIEGUE CONTINUO

¿Qué sucede si extendemos el concepto de la integración continua hasta llevarlo al despliegue a producción? Es decir, que cada que se cambia el código de la aplicación, automáticamente se genere un build nuevo, y si pasa exitosamente todas las pruebas automáticamente se libere a producción. Esto es lo que se conoce con el nombre de despliegue continuo (continuous deployment) y aunque parezca utópico, es algo que sí existe y hay organizaciones que lo hacen [2].

¿Qué sucede si queremos quedarnos un pasito antes y simplemente dejar el código "listo para producción", y solamente si estamos seguros que queremos hacerlo, entonces ya "presionemos el botón" para poner en producción? Justamente eso es lo que se conoce como entrega continua (continuous delivery) y es una práctica recomendada, especialmente en el contexto DevOps [3].

Aunque la entrega continua parezca simplemente agregar un paso más a la integración continua y no tener mayores consecuencias, en realidad no es solo eso ya que por lo pronto involucra tener automatizadas todas las pruebas y scripts para migración de datos así como para rollback en caso de fallas. Adicionalmente, tiene un impacto importante en el modelo de ciclo de vida de software ya que embebe prácticas ágiles en el mismo proceso de planeación y construcción de software. Tan es así, que hay quienes dicen que para realmente estar haciendo "Agile" necesitas tener entrega continua, sino lo que estás haciendo no puede ser llamado Agile [4]. A fin de cuentas, la meta de la entrega continua es poner al cliente al mando de lo que se libera y cuándo se libera.

Pero bueno, es un hecho que antes de correr hay que caminar, así que antes de pensar en entrega continua necesitamos dominar la integración continua.

Referencias

[1] M. Fowler. "Continuous Integration". http://swgu.ru/su
[2] T. Fitz. "Continuous deployment at IMVU: Doing the impossible 50 times a day". http://swgu.ru/sw
[3] J. Humble. "Continuous Delivery vs. Continuous Deployment". http://swgu.ru/sx
[4] I. Buchanan. "Why Agile Development needs Continuous Delivery". http://swgu.ru/sy

Pedro Galván Kondo es director editorial de Software Guru.

ENTREGA CONTINUA

¿CÓMO LLEGAMOS AQUÍ?

En prácticamente cualquier industria, el éxito de una organización depende cada vez más de la capacidad de su software. La web, el cómputo móvil y las aplicaciones embebidas definen la forma en que los clientes perciben a una marca, la forma en que sus empleados colaboran, y la competitividad de la empresa. Los usuarios cada vez toleran menos la inconveniencia y buscan gratificación inmediata. El no lograr cumplir dichas expectativas puede ser fatal para la empresa. Para mantenerse competitivas, empresas como Amazon, Facebook y Etsy han adoptado una disciplina conocida como entrega continua.

La entrega continua es una disciplina que lleva la práctica del desarrollo ágil a su conclusión lógica, creando software que siempre está listo para ser liberado. Para lograrlo incorpora prácticas y herramientas de Agile, integración continua y DevOps para transformar la manera en que se entrega software. Al automatizar tareas rutinarias y tardadas como la compilación pruebas e instalación -y ejecutarlas de forma temprana y frecuente- la entrega continua hace que el proceso de software sea más predecible y repetible, mejorando significativamente la calidad y frecuencia de las entregas.

Pero ... ¿cómo llegamos aquí? Aunque pudiera parecer que la entrega continua es una simple moda o algo que solamente aplica para startups, en realidad no es así. La entrega continua es, como ya lo indicamos, una continuación del desarrollo ágil hasta su conclusión. Explico esto a continuación ...

PRIMERO, FUIMOS ÁGILES

El manifiesto ágil define 12 principios, y el primero de estos dice "... satisfacer al cliente por medio de la entrega temprana y continua de software valioso".

Es decir, las metodologías ágiles nos enseñaron a agregar valor a los productos de software una historia a la vez, de manera que aceleremos los ciclos de retroalimentación y continuamente estemos alineando los productos que construimos con las necesidades del mercado. Estas metodologías promueven la integración continua, y esto provocó un flujo de liberaciones más pequeñas y frecuentes, que llegaron hacia los equipos de calidad y operación de TI.

LUEGO ADOPTAMOS DEVOPS

DevOps está surgiendo como un modelo de operación que une a todos los involucrados en la cadena de producción de software para colaborativamente perseguir metas de negocio conjuntas, tales como entregar un nuevo producto de alta calidad que opere de manera estable.

Al estandarizar los procesos, configuración de versiones y aumentar la colaboración, DevOps habilita a los equipos para trabajar juntos de manera más eficaz y eficiente, mejorando la velocidad y predictibilidad.

Es difícil definir qué es DevOps. Pero es sencillo ver lo que DevOps hace. Habilita la confianza y coordinación transparente entre equipos alineados con un solo propósito: poner en producción software grandioso. En esencia, DevOps no se trata de desarrollo ni de operaciones, se trata del negocio.

AHORA ENTREGAMOS, DE MANERA CONTINUA

La entrega continua lleva Agile a su conclusión lógica con una forma de trabajo que asegura que el software siempre está listo para liberarse. Para lograrlo, se apoya en prácticas de Agile, integración continua y DevOps.

La entrega continua es un proceso de punta a punta que abarca el ciclo entero de construcción, prueba, despliegue y liberación. Para que la integración continua sea una realidad, debemos automatizar estas actividades. Esto habilita a las organizaciones para innovar, reducir costos y diferenciarse en mercados competitivos, por medio de un proceso eficiente de entrega de software.

¿CÓMO DETERMINAR SI HACES ENTREGA CONTINUA?

Sabes que estás haciendo integración continua cuando:

- Tu software se puede liberar a lo largo de su ciclo de vida.
- Tu equipo le da mayor prioridad a mantener el software listo para liberar que a incorporar nuevas capacidades.
- El equipo puede obtener retroalimentación rápida y automatizada sobre el estatus del producto.
- Es posible mover cualquier versión del software a cualquier ambiente (staging, producción) con un solo comando.

La prueba clave de si has logrado o no la entrega continua es que el usuario de negocio pueda solicitar en cualquier momento que la versión actual de desarrollo pueda ser liberada a producción, y el equipo no sienta temor por esto.

Este texto es una versión traducida y editada del artículo "Continuous Delivery: What is it? How did we get here?" publicado en el sitio web de Electric Cloud.
http://electric-cloud.com/resources/continuous-delivery-101/what-is-continuous-delivery

EL CASO DE NEGOCIO DE LA ENTREGA CONTINUA

¿Qué sientes cuando escuchas la frase "liberación a producción"? ¿Tranquilidad? ¿Emoción? Si tu equipo todavía depende de pruebas manuales e instala en producción manualmente, lo más seguro es que sientas cosas cercanas al miedo y preocupación. La figura 1 ilustra el ciclo emocional típico de hacer liberación manual de software. Se comienza por un periodo de impasibilidad, que al llegar la fecha original de entrega se convierte en urgencia, después de lograr la entrega se convierte en alivio, luego en celebración y finalmente en complacencia.

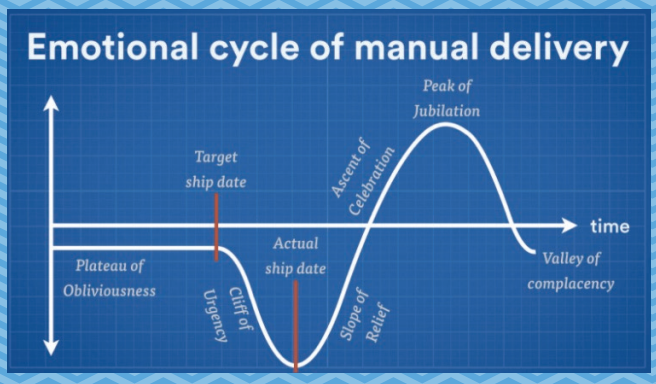


Figura 1. Ciclo emocional de la entrega manual

Es por ello que el desarrollo de software se está moviendo hacia la continuidad. El énfasis reciente en la integración continua, pruebas automatizadas, monitoreo constante y analítica todas apuntan a una clara tendencia: aumentar la capacidad para reaccionar. Conforme las organizaciones exploran lo que estos cambios implican para ellas, invariablemente descubren la entrega continua.

Que quede claro: la entrega continua no es algo que solo aplica para empresas "unicornio". Cualquier equipo puede y debería practicar la entrega continua. Este artículo revisa el caso de negocio alrededor de hacer este cambio. Aborda el trabajo requerido y beneficios que traerá.

BENEFICIOS PRINCIPALES DE LA ENTREGA CONTINUA

Tiempos de reacción más cortos. El beneficio más claro de la entrega continua es que permite reaccionar rápidamente a estímulos; ya sea que el mercado tenga un cambio repentino o el negocio cambie la estrategia. ¿A qué empresa no le gustaría tener esa capacidad de ajustar sus productos digitales rápidamente?

Reducción de riesgo. En la mayoría de las organizaciones, liberar un sistema a producción es todo un logro digno de presumirse. ¿Y por qué no? Se hacen disponibles nuevas capacidades a los usuarios y se corrigen defectos. Todo mundo está feliz, ¿cierto? El problema es que típicamente liberar una nueva versión involucra un gran esfuerzo de parte de los equipos de QA y operaciones. En comparación, bajo un modelo de entrega continua el software se libera constantemente (no necesariamente al usuario final). Así que la ceremonia y riesgo de una nueva versión se reduce. Si confías en tu procedimiento de liberación a diario, encontrarás y corregirás las deficiencias mucho más rápido que si solo lo haces pocas veces al año.

Descubrir ineficiencias y costos. Cualquier organización de software incurre en costos para liberar software, y en la mayoría de los casos, estos costos no están claros. La entrega continua hace este proceso mucho más transparente, no solo al equipo de desarrollo sino también hacia la dirección. Al construir un pipeline, quedará claro donde se involucran actividades manuales, cuáles son los cuellos de botella, y qué actividades se pueden automatizar fácilmente. Una vez implementado, el pipeline crea un ciclo virtuoso; en lugar del procedimiento largo y obscuro se establecen incentivos claros para una dinámica sana de entrega de software.

Flexibilidad para liberar. Moverse a un modelo de entrega continua requiere establecer infraestructura, tanto en un contexto operativo como de arquitectura de software. Pero una vez que se cuenta con ella, provee al negocio más flexibilidad sobre cómo libera nuevas capacidades y correcciones al software. Por ejemplo, es posible liberar funcionalidad específica solamente a un subconjunto de usuarios para asegurar que se comportan de la manera esperada antes de liberarlas a todos los usuarios. La funcionalidad se puede construir y probar, pero dejar escondida para habilitarla en el momento deseado. Por ejemplo, si el equipo de marketing hace un anuncio importante que va acompañado de nueva funcionalidad en el sistema de software, no tienes que liberar una nueva versión en ese momento.

NO SOLO ES PARA UNICORNIOS

Ante estos beneficios, parecería que la entrega continua es algo mágico, solo al alcance de pocos. No lo es. Tan solo es un ajuste en la forma en la que pensamos sobre el diseño, construcción y entrega de software, soportado por una inversión enfocada en las iniciativas requeridas para implementarla. La mayoría de las implementaciones fallidas de entrega continua se deben a esta falta de este compromiso.

Dado que la entrega continua requiere cambiar o agregar procesos, herramientas y roles, puede sentirse como un cambio demasiado grande. Esto es agudizado por el hecho de que hay que cambiar áreas que previamente habían sido ignoradas o a las que se les había dado baja prioridad. El lado positivo es que los humanos somos muy adaptables, especialmente si estamos motivados por la expectativa de tener un proceso de liberación que no sea un infierno.

La inversión inicial más grande es lograr un acuerdo de que la transición a la entrega continua es una meta de negocio digna de perseguirse, que solo se puede lograr por medio de la participación y compromiso de la organización de software y la dirección de negocio.

"Durante los próximos meses no vamos a agregar nueva funcionalidad al sistema y mejor nos dedicaremos a trabajar en automatizar pruebas y construir infraestructura para integrar y liberar versiones", dijo ningún gerente de producto nunca. La forma de conseguir que los gerentes y usuarios de negocio apoyen este esfuerzo es mostrarles qué tan rápido podrán liberar nuevas capacidades una vez que los esfuerzos de la entrega continua comiencen a rendir frutos. Imagina un mundo en el que un gerente de producto tiene una nueva idea para incorporar en un software y en cuestión de días ésta es liberada a un subconjunto de usuarios, instrumentada con analítica de eventos para evaluar la reacción de los usuarios; esta es la promesa de la entrega continua.

En realidad, no es necesario detener por completo la construcción de nuevas características hasta que se implante entrega continua. Simplemente es cuestión de que todas las áreas del negocio estén convencidos de los beneficios a largo plazo. Esto no suena tan difícil, pero se requiere bastante disciplina para mantener este compromiso durante un periodo largo de tiempo, de manera que la iniciativa de implementación de entrega continua no quede en segundo plano o se termine abandonando.

ÁREAS QUE REQUIEREN INVERSIÓN

Adoptar una práctica de entrega continua va mucho más allá de adquirir herramientas de software. Hay diversas áreas en las que se debe invertir para obtener los beneficios de la entrega continua.

Pruebas automatizadas. Todas las capacidades del sistema deben poder ser probadas de forma automatizada. Esto incluye no sólo pruebas unitarias, sino también de integración y sistema. Las pruebas deben ser coordinadas de forma automatizada, usando una herramienta para ello. Adoptar una práctica de pruebas automatizadas desde cero, puede tomarle a una organización de software de 6 a 18 meses.

Cambios en arquitectura aplicativa. Es necesario evolucionar la arquitectura de las aplicaciones hacia servicios. En el caso de software que no es SaaS, es necesario establecer una arquitectura de componentes y control de configuración que permita que cada componente se pueda liberar de manera independiente. De esta manera podemos habilitar capacidades como feature flagging y versiones canario. Aunque este cambio puede parecer demasiado complicado o costoso, brinda beneficios más allá de la entrega continua ya que al desacoplar sistemas complejos en componentes o servicios bien definidos y con bajo acoplamiento es más fácil tanto desarrollarlos como reutilizarlos de forma independiente. A la mayoría de los equipos les toma entre 4 y 8 iteraciones cambiar la arquitectura de una aplicación hacia servicios.

Granja de integración y prueba. Cada que se genera una nueva versión del código del navegador Firefox se disparan automáticamente miles de pruebas que requieren más de 200 horas de CPU. Poder soportar esto requiere una inversión significativa en hardware, ya sea interno o en la nube. Tu granja de integración y prueba requiere suficiente capacidad para soportar entrega continua, lo cual acelerará tu ciclo de retroalimentación de desarrollo y facilitará probar contra distintos sistemas operativos, navegadores, etcétera. Ajustar la capacidad de tu granja es un proceso continuo, por lo que típicamente lo más conveniente es utilizar servicios de cómputo en la nube para este propósito.

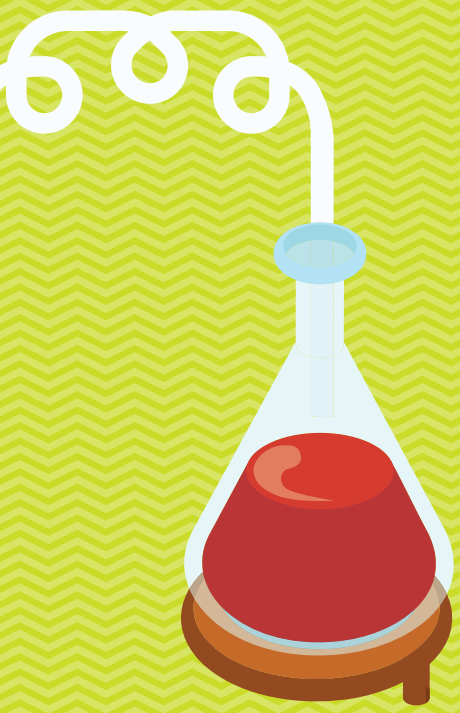
Cambio cultural. La entrega continua requiere un cambio cultural, ya que en esencia se están reemplazando controles humanos por un flujo continuo y automatizado. Inevitablemente, esto provocará que se eliminen silos y habrá personas que se sientan amenazadas ante el prospecto de perder el control. Aunque esto es un proceso continuo, el periodo más intenso durará entre 2 y 6 meses.

Reclutamiento enfocado. Para avanzar de manera sostenida en la adopción de entrega continua puede ser necesario contratar personas que se dediquen solamente a trabajar en la infraestructura requerida para la entrega continua. La duración de estos contratos típicamente será de 2 a 6 meses.

Este texto es una versión traducida y editada del artículo "The business case for continuous delivery" publicado por la empresa Atlassian en https://www.atlassian.com/continuous-delivery/business-case-for-continuous-delivery

EVALÚA LA CAPACIDAD DE TU ORGANIZACIÓN PARA ENTREGA CONTINUA

Por Pedro Galván



El concepto de entrega continua está ganando tracción en las organizaciones; sin embargo, su adopción no es trivial. El cambio de entregas poco frecuentes a un flujo continuo puede intimidar a cualquiera. Adicionalmente, las organizaciones grandes y/o con varias décadas de operación típicamente tienen una gran variedad de herramientas independientes para soportar el desarrollo y gestión de software, que no se integran entre sí. Así que antes de comenzar un esfuerzo de adopción de entrega continua, bien vale la pena evaluar nuestro estatus actual.

En este artículo presento un modelo de evaluación de capacidad para entrega continua desarrollado por la empresa Electric Cloud y documentado a mayor detalle en su whitepaper "The Journey to Enterprise Continuous Delivery" [1]. El objetivo es que las organizaciones puedan evaluar sus capacidades actuales relacionadas con entrega continua y establecer un mapa de ruta para su adopción.

El modelo considera 3 dimensiones:
1. Tecnología
2. Personas
3. Procesos

En cada dimensión se identifican cinco niveles y para cada uno se indican rasgos comunes.

Aunque cada dimensión es independiente, las organizaciones típicamente maduran de forma concurrente a través de ellas. Es decir, difícilmente encontraremos una organización que mejore significativamente sus procesos sin mejorar en sus personas y tecnología.

PERSONAS

Una organización requiere la cultura adecuada para soportar e incentivar la entrega continua. Desafortunadamente, demasiadas organizaciones perciben que el trabajo necesario para establecer la colaboración y comunicación necesaria es un obstáculo demasiado grande que bloquea la productividad. Es solo cuando estos equipos maduran que se dan cuenta de la importancia de su cooperación.

Nivel	Rasgos comunes	Resultados
0	Integrantes con personalidad de "divas". No hay compromiso con objetivos del equipo y el negocio.	Hay resistencia a usar estándares o procedimientos dado que se cree que limita creatividad. La calidad no es consistente. Se dan problemas en la integración ya que cada persona defiende su forma de hacer las cosas.
1	Las personas están aisladas; la comunicación se hace por correo electrónico y típicamente es reactiva; el liderazgo es "a la antigua" que se reúne 1 o 2 veces por semana para ver avance; los integrantes con menor experiencia no tienen mentoría.	La comunicación típicamente sólo se da de forma vertical (entre jefes y sus coordinados); hay poca colaboración y la retroalimentación entre pares no se recibe bien; se opera en modo reactivo. Las personas se dedican a solamente hacer lo que tiene asignado en el plan de trabajo y se alarman cuando este cambia.
2	Algo de colaboración entre equipos; los programadores se enfocan en construir componentes individuales que "pasan" a que otros revisen y liberen; las iniciativas de mentoreo se dan de forma silvestre.	Hay algo de colaboración pero no es muy efectiva; las personas están algo conscientes de las metas de negocio pero su enfoque está en proteger agendas individuales.
3	Amplia conciencia de las metas de negocio; hay interés en colaborar para conseguir las metas; se busca y aprecia la retroalimentación; se da algo de atención a la mentoría.	Alta adaptabilidad al cambio, las personas entienden los objetivos y respetan los planes pero entienden que los planes pueden cambiar; hay interés en la satisfacción del cliente, reflejada en jidoka (cualquiera puede detener la línea de producción) y kaizen (mejora continua).
4	La colaboración y cooperación son clave para cualquier actividad; hay amplio interés en la mentoría.	Los equipos son dirigidos por la mejora continua y continuamente se ajustan a las necesidades del negocio.

PROCESOS

La entrega continua implica un flujo continuo a través del cual todos los involucrados colaboran para entregar mayor valor a los usuarios de forma consistente. Esto implica una planeación y estilo de liderazgo distinto al tradicional, pero cualquier organización puede lograr establecerlo con la dedicación y compromiso adecuados.

Nivel	Rasgos comunes	Resultados
0	No hay planes ni procesos, solo procedimientos ad-hoc.	Construir software es considerado una actividad creativa, no de ingeniería; la mayoría del tiempo se dedica a generar y discutir formas de hacer las cosas.
1	Existen planes de proyecto pero faltan procesos bien definidos para gestionar peticiones (cambio, diseño, escalamiento); no hay pruebas continuas.	La falta de procedimientos para atender peticiones resulta en "bomberazos" frecuentes. Esto impacta incluso al código fuente, que está regado en distintas ramas y repositorios.
2	Hay procesos definidos (ej. Integración continua, pruebas, despliegue) pero la ejecución no es consistente o integrada de punta a punta; es común que se utilice Scrum.	Los procesos son vistos como engorrosos, así que las personas toman atajos.
3	Existen procesos bien definidos, integrados y automatizados; la última versión del código está en una sola rama del repositorio; es común que se utilice Kanban.	Dado que los procesos están cuidadosamente planeados e instrumentados, tienden a hacerse invisibles, permitiendo que las personas se enfoquen en sus responsabilidades principales.
4	Existen procesos de mejora continua para satisfacer requerimientos de negocio dinámicos apoyándose en herramientas de monitoreo del proceso.	Los procesos se mejoran de manera continua utilizando datos en tiempo real como parte de su ciclo de retroalimentación.

TECNOLOGÍA

Existen una gran oferta de herramientas para instrumentar la entrega continua. Sin embargo, siempre hay que tener en cuenta que las herramientas por sí solas no resuelven el problema. Conforme las organizaciones maduran en las otras 2 dimensiones, hacen que las herramientas se ajusten a sus necesidades, en lugar de estas ajustarse a las capacidades de las herramientas.

Nivel	Rasgos comunes	Resultados
0	Procedimientos manuales; herramientas open source básicas.	Fuera de algo de scripting básico, hay poca automatización.
1	Cada equipo utiliza distintas herramientas para actividades específicas (ej. compilar, probar) no integradas entre sí, y administradas por personas distintas sin un proceso común.	Se depende de personas para integración manual; las personas se convierten en cuellos de botella; la salida de una persona del equipo tiene un gran impacto.
2	Se crea infraestructura común pero no se considera el proceso de punta a punta ni hay una buena integración.	Los programadores dedican más tiempo a construir y mantener herramientas que a mejorar el producto. No hay estandarización a través de áreas o departamentos, así que transferir un producto digital a otra área/departamento es caótico.
3	Automatización a través de todo el ciclo de integración/prueba/despliegue; la configuración y aprovisionamiento de la infraestructura también está automatizada; se utiliza infraestructura centralizada y elástica disponible bajo autoservicio.	La tecnología se encarga de la orquestación; las herramientas están tan integradas que pasan desapercibidas; se genera ejecutables de manera rápida y confiable, cuando hay errores se reportan en tiempo real.
4	Por medio de dashboards cualquiera puede estar enterado del estatus en tiempo real; el negocio determina qué se libera y cuándo.	Cuando se necesitan ajustes, se puede modificar las herramientas y procesos en tiempo real con impacto mínimo a la operación.

Adoptar una disciplina de entrega continua involucra tener la cultura, procesos y herramientas adecuadas. Espero que este sencillo modelo te ayude a determinar donde está tu organización.

Referencias

[1] A Journey to Enterprise Continuous Delivery. Electric Cloud. http://swgu.ru/sz

Cómo Hacer Investigación de Usuarios en Equipos Ágiles

Por Misael León

La clave para prosperar en el entorno competitivo de los productos digitales, sin duda, es centrarse en la Experiencia del Usuario (UX). La gente simplemente no va a pagar dinero por un producto que no se adapte a sus necesidades.

Por su lado, Agile se ha convertido en la estrategia más común en la industria del software. Según una encuesta del 2015 [1] sólo el 2% de las empresas todavía operan bajo el modelo tradicional de desarrollo en cascada. Simplemente, Agile funciona.

Los equipos ágiles pueden adaptarse rápidamente al cambio en las preferencias del usuario y la Investigación de Usuarios es la herramienta para detectar ese pulso de cambio. Sin embargo, existen todavía varios conceptos erróneos que se interponen en el camino de investigar usuarios en equipos ágiles. En este artículo encontrarás algunas recomendaciones prácticas para hacer que funcionen.

NO HAY UX SIN EL USUARIO

En el centro de UX se encuentra la investigación de usuarios, la cual consiste en un conjunto de técnicas para comprender los comportamientos, necesidades y motivaciones de los usuarios a través de la observación y la medición.

Existen actividades específicas de investigación de usuarios que pueden ayudarte en cada etapa del proceso de desarrollo del producto: entrevistas [2]; pruebas de usabilidad [3], investigación contextual; ejercicios generativos; encuestas; analytics; pruebas A/B. El desafío es identificar cuál es el adecuado para el problema que deseas resolver y asignar tiempo suficiente para ejecutarlo y obtener valor de él. La clave reside en probar poco y a menudo.

MITOS DE INVESTIGACIÓN DE USUARIOS EN EQUIPOS ÁGILES

Cuando trates de introducir la investigación de usuarios en un equipo ágil, escucharás objeciones como: *"toma demasiado tiempo"*, *"es el trabajo de otra persona"*, *"oh no, eso es muy caro"*, *"es muy difícil"*, *"ya no tenemos tiempo en este sprint"*. Estos mitos son el resultado de una mentalidad enfocada a shipping features. El trabajo se recompensa en función de cuántas características se liberan; y no de cuánto estamos cambiando el comportamiento del usuario hacia nuestros objetivos de negocio.

Por lo general, bajo *Agile* no existe un paso dedicado a la investigación que esté bien diferenciado. La investigación y los requerimientos se combinan a menudo en un solo paso. El desarrollo se guía por un prototipo o *wireframe*, que puede o no reflejar lo que los usuarios realmente quieren. Normalmente, este trabajo de investigación no tiene un responsable. No hay tiempo dedicado para explorar las motivaciones de los usuarios, sus expectativas y cómo utilizan el producto. En el mejor de los casos, los propietarios de productos presentan especificaciones de diseño justo antes que comience el desarrollo.

ROMPIENDO LOS MITOS

Para incorporar las actividades de la investigación de usuarios en tu proceso ágil, todo el equipo debe cambiar a una visión del producto centrada en el usuario. El cambio comienza cuando el aprendizaje de los usuarios está en el centro de tu trabajo. Sí, seguirás liberando *features*, pero éstas se alinearán a cambiar el comportamiento del usuario y mejorar el producto.

1. Prueba pequeñas hipótesis

Evalúa pequeñas partes de tu producto mediante la formulación de una hipótesis sobre cómo resolver un problema con un nuevo diseño o función. Por ejemplo, podrías decir: "Creemos que si implementamos una funcionalidad de alerta de conflictos podremos reducir los *tickets* de soporte en un 30%." De repente, el trabajo cambia. Ahora el objetivo es reducir los tickets de soporte, no liberar una funcionalidad determinada. Esto significa que el equipo explorará opciones como la creación de un rol de moderador. No hay necesidad de investigar el producto completo, solamente esa pequeña idea.

2. Asignar el trabajo de investigación

Para que las actividades de investigación de usuarios sean tomadas en serio, tienen que ser visibles y ser priorizadas, así que deben ser incluidas en el backlog; de esta manera ya no podrán ser ignoradas. Este es otro paso para pasar de una cultura de shipping a una cultura de continuo aprendizaje.

3. Maximiza el uso del tiempo

El rápido ritmo de los sprints podría no permitir profundizar en la investigación de usuarios. Debes ser creativo en la aplicación de los métodos que utilizas. Por ejemplo, puedes realizar una prueba de usabilidad de una porción pequeña del producto y al mismo tiempo realizar una entrevista regular con preguntas más subjetivas. De esta manera se maximiza el tiempo de la sesión. También puedes reclutar usuarios ya familiarizados con el producto para que su curva de aprendizaje durante la prueba no sea muy prolongada. Otro tip útil es programar sesiones de investigación regulares con los usuarios. Esto hace que convierta en una rutina semanal y todos en el equipo saben que determinado día es "el día de investigar usuarios". Al final lo que importa es tener claro lo que se quiere validar y encontrar la técnica correcta para encontrar las respuestas a tus incógnitas.

Misael León (@misaello) es un UX/Product Designer que trabaja en Nearsoft investigando usuarios, desarrollando ideas de productos y diseñando prototipos. Su misión es la de crear herramientas intuitivas para que otros puedan realizar su trabajo. Le apasiona difundir las mejores prácticas de UX en las comunidades de desarrollo.

4. Involucra a todo el equipo

Para construir un entendimiento compartido de los problemas del usuario no hay sustituto para invitar a todo el mundo del equipo a las sesiones. Ser testigo de cómo a los usuarios se les dificulta completar una tarea, motivará inmediatamente a los desarrolladores a querer arreglarlo de inmediato.

Idealmente, podrías instruir al resto del equipo a ejecutar las sesiones por sí mismos. Inicialmente, podrían tomar notas, hablar con los clientes y hacer preguntas sencillas. Con el tiempo se verán tan acostumbrados al proceso que serán capaces de hacerlo por su cuenta.

CONSEJOS PARA EQUIPOS LOCALES

Atlassian, la compañía que creó JIRA y Confluence, hizo un gran trabajo construyendo su propio laboratorio de investigación de usuarios, ellos lo llaman el Atlab [4]. No es de ninguna manera un laboratorio costoso o de alta tecnología. Es un espacio de trabajo exclusivamente dedicado a hacer este tipo de investigaciones; siempre disponible para todos. Más que eso, lograron crear la mentalidad correcta del equipo alrededor de solucionar problemas reales de gente real.

CONSEJOS PARA EQUIPOS DISTRIBUIDOS Y USUARIOS REMOTOS

La investigación de usuarios también se puede realizar de forma remota. A continuación comparto algunos consejos:

- Las pruebas de usabilidad se pueden realizar a través de videollamadas. El ideal es usar alguna herramienta que muestre lo que sucede en la computadora del usuario (desktop sharing) y también muestre su webcam, de esta manera puedes observar tanto las acciones que realiza en pantalla como sus expresiones faciales conforme trata de usar tu producto.
- Para crear diagramas de afinidad que ayuden a interpretar los resultados de tu investigación puedes utilizar post-its virtuales en Google Drawings o algo similar. Si bien no es lo ideal, es mejor que no realizar ninguna investigación UX en absoluto.
- Para reclutar usuarios que prueben y evalúen tu producto puedes utilizar herramientas como ethn.io o usertesting.com
- Haz *Tree Tests*, *Card Sorting* o *Surveys* con optimalworkshop.com
- Prioriza tu *backlog* con la encuesta de satisfacción de la Metodología Kano en FeatUR.me

INCORPORACIÓN EN CICLO ÁGIL

La figura 1 muestra cómo se incorporan actividades de investigación de usuarios en un ciclo de vida ágil. Antes de entrar al ciclo de desarrollo (marcado en amarillo) debemos incluir un ciclo de investigación y validación de ideas (marcado en rojo). Este ciclo de "descubrimiento" se realiza un par de semanas antes de iniciar el sprint, de tal manera que al final tendrás un prototipo ya validado. El cual a su vez se convertirá en el documento de requerimientos que hará posible la planeación del *sprint* y el desarrollo de lo planeado. En ambos ciclos hay actividades de investigación específicas que puedes realizar. Recuerda, probar poco y a menudo es la clave.

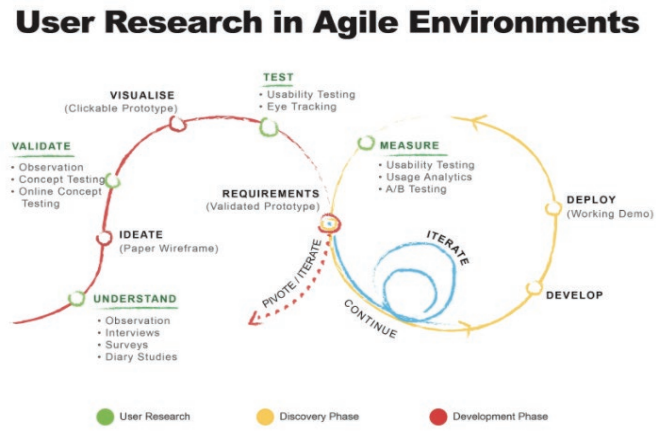


Figura 1. Investigación de usuarios en un ciclo ágil

CONCLUSIÓN

Establecer una práctica de investigación de usuarios llevará tiempo, pero poco a poco ayudarás a tu equipo a integrar la voz del cliente en su trabajo. El tiempo que dediques a este cambio de mentalidad valdrá cada segundo que inviertas.

Todos los integrantes de tu equipo se beneficiarán:

- Los product managers ahorrarán dinero. Ellos pueden asegurarse de que están construyendo lo correcto, y que ahorrarán tiempo para mantener la preparación del producto aún más.
- Los diseñadores estarán seguros sobre sus decisiones de diseño. Ellos tendrán una mejor comprensión del problema y validarán sus suposiciones con usuarios reales.
- Los desarrolladores pueden ver su trabajo siendo utilizado en tiempo real. Podrán idear soluciones alternativas al problema que están observando. Desarrollarán empatía para las personas que utilizan lo que ellos construyen.

No es posible crear soluciones significativas sin incluir la perspectiva del cliente. La gente simplemente no va a pagar dinero por productos que no se adapten a sus necesidades y su estilo de vida.

Referencias

[1] John Jeremiah. "Survey: Is agile the new norm?". TechBeacon. http://swgu.ru/sq
[2] "Moderated User Research: Unveiling insights with 1-on-1 interviews". UX Team by Nearsoft. http://swgu.ru/sh
[3] "Elito Method: Users reactions vs. assumptions." UX Team by Nearsoft. http://swgu.ru/si
[4] Becky White. "Atlab: Set up your own user research lab". http://swgu.ru/sj

La Importancia de la **Ingeniería de Requerimientos**

Por Guilherme Siqueira

La ingeniería de requerimientos es una de las disciplinas fundamentales de la ingeniería de software y proporciona información para la mayoría de las demás disciplinas. Este artículo presenta resultados de investigaciones que fundamentan de manera cuantitativa esta cuestión. El propósito es demostrar las consecuencias del descuido de la disciplina de requerimientos: retrasos en el cronograma y costo adicional, nivel alto de defectos en el software y principalmente la entrega de un software que no satisface las necesidades del cliente.

FALLOS FAMOSOS

Para ilustrar, a continuación se muestran casos famosos de fallos en proyectos de software relacionados con algún tipo de discapacidad en el ejercicio de la ingeniería de requerimientos.

La sonda espacial Mars Climate Orbiter (MCO)

La MCO fue una sonda espacial cuyo objetivo principal era estudiar el clima de Marte. Fue lanzada en diciembre de 1998, alcanzando Marte nueve meses y medio después. Al entrar en la órbita de Marte, la sonda fue destruida en la atmósfera debido a un error de cálculo en la maniobra. El error fue causado porque el software responsable del cálculo de ruta de entrada esperaba datos usando el sistema de medidas imperial (libras fuerza) y recibió datos en sistema métrico universal (newtons). No se sabe si la NASA proporcionó la especificación equivocada al proveedor que desarrolló el software o si hubo una falla del proveedor en el levantamiento de requerimientos.

Misil antibalístico Patriot

Durante la Guerra del Golfo, los Estados Unidos usaron un sistema de defensa con misiles antibalísticos llamado Patriot. El 25 de febrero de 1991 este sistema falló y no interceptó el misil Scud lanzado por Irak que mató a 28 militares y lesionó otros 98.

La raíz de la falla fue que el software original del sistema utilizaba datos de las señales del radar para calcular la ruta del Patriot y trabajaba con una precisión de fracciones por segundo. Para hacer frente a misiles de más alta velocidad, se creó una subrutina para manejar el tiempo con mejor precisión (más decimales). Sin embargo, esta subrutina no fue aplicada a todas las partes necesarias del software lo que generó una acumulación de fallas de precisión; y después de cierto tiempo, el sistema se quedaba ineficaz. No fue sólo un fallo de programación, fue también un fallo de evaluación en el impacto del cambio.

Cohete Ariane 501

El 1996, el cohete Ariane 501 fue lanzado y unos 37 segundos después del despegue, se desvió del curso esperado para después explotar. Fue el primer lanzamiento de la serie Ariane 5. La pérdida del cohete y su carga ascendieron a una pérdida de más de 370 millones de dólares. La comisión de investigación señaló que la falla había estado en el sistema de referencia inercial, el cual había sido reutilizado de la serie Ariane 4. El problema fue que el Ariane 5 estaba diseñado para transportar más carga, lo que implica otros estándares de trayectoria y velocidad diferentes, y esto no se tomó en cuenta al realizar el diseño y pruebas pertinentes.

Archivo Virtual FBI

A principios del 2000, el FBI comenzó el desarrollo de un software llamado Virtual Case File (VCF) que permitiría el intercambio de información de casos entre los agentes. Originalmente se estimó que el desarrollo tomaría 3 años. Después de la tragedia del 11 de septiembre de 2001 el FBI recibió fuertes críticas por no anticipar el ataque; las evidencias estaban expuestas, el error fue no enlazarlas entre sí. Ante esto, se decidió aumentar la prioridad del VCF y extender su alcance. Cinco años y 170 millones de dólares después, el sistema no había podido ser terminado y se canceló su desarrollo. La investigación a cargo determinó que las principales causas de fracaso fueron:

- Cambios frecuentes en los requerimientos. El proveedor alegó que el FBI adoptó la filosofía de trabajo de: "Yo sé lo que quiero después de ver el producto."
- Alta rotación de gestión (también contribuyó a los cambios frecuentes).
- Aumento descontrolado del alcance, con requerimientos agregados incluso cuando el proyecto ya estaba retrasado.

EL PRINCIPAL MOTIVO DE FRACASO

De acuerdo con cifras del Project Management Institute (PMI) [1], el 47% del fracaso de los proyectos es causado por una deficiencia en el ejercicio de la ingeniería de requerimientos. Algunos casos comunes:

- El producto se entrega sin cumplir con los requerimientos necesarios, sin identificar varios de ellos.
- La entrega final es un producto que no satisface al cliente, aunque a tiempo y dentro del presupuesto.
- El proyecto incorpora requerimientos que no deben estar en el alcance.
- La estimación de costo/esfuerzo se hace en base a un alcance equivocado ya que no considera algunas áreas funcionales y procesos de negocio.
- Fallas de comunicación sobre requisitos, lo que resulta en la entrega de un producto defectuoso.
- Cambios innecesarios debido a la falta de atención por comprender correctamente las necesidades del cliente al principio.

UNA DE LAS PRINCIPALES CAUSAS DE DEFECTOS

Los defectos pueden surgir en cualquier etapa del ciclo de vida del proyecto, siendo el más común aquellos insertados durante la construcción, donde el producto construido tiene un comportamiento diferente al especificado.

Asimismo, un número significativo de defectos se origina a partir de los

requerimientos. Pero, ¿La calidad no es cumplir con los requerimientos? ¿Cómo sería posible ello? Simple, basta que la especificación no represente correctamente las necesidades del usuario.

Según, Capers Jones [2], 1 de cada 5 posibles defectos se origina en los requerimientos, y estos pueden ser hasta el 60% del total de errores en el proyecto según explica Pohl [4]. La problemática es que estos sólo se detectan en etapas avanzadas del proyecto, a menudo cuando se aprueba el producto. Por otro lado, Leffingwell [5] acota que en proyectos complejos la fuente más común de errores son los requerimientos. Esto último se refleja en los resultados mostrados por Jones y Bonsignour [3] que se capturan en la tabla 1 donde se muestra la tasa potencial de defectos por punto de función segregados por la fuente del defecto y estratificado de acuerdo al tamaño del sistema.

Origen del Defecto	Tamaño del sistema (en puntos de función)		
	100	1,000	10,000
	Defectos potenciales (bugs/PF)		
Requerimiento	0.75	1.00	1.25
Arquitectura	0.10	0.25	0.50
Diseño	1.00	1.25	1.50
Código fuente	1.70	1.75	2.00
Material de pruebas	1.50	1.85	2.00
Documentación	0.65	0.70	0.75
Base de datos	2.00	2.75	3.00
Website	1.50	1.75	2.00
Total	9.20	11.30	13.00

Tabla 1. Defectos en potencial por tamaño de sistema y origen.

Los defectos originados en requerimientos son más difíciles de eliminar por métodos tradicionales: pruebas y análisis estático. Esto se debe a que cuando se aprueba la especificación con un requerimiento defectuoso, los casos de prueba son elaborados a partir del comportamiento equivocado.

Adicionalmente, los cambios en requerimientos tienden a una densidad de defectos mayor, debido a que típicamente son tratados a toda prisa. Y son más difíciles de eliminar porque "saltan" el control de calidad del proyecto. Por ejemplo, un crecimiento del 10% en el alcance del proyecto implica un aumento del 12% en el número de defectos.

EL COSTO DE REPARACIÓN DE LOS DEFECTOS

La industria del software tiene un gran potencial para explotar las ganancias de eficiencia. Según Boehm [6] del 40 al 50% del esfuerzo en los proyectos de software se gasta en la corrección de defectos. Uno de los factores que contribuyen para esta ineficiencia es que al corregir un defecto existe un 20-50% de posibilidades de introducir otro error al software [7].

Pohl [4] sostiene que la mayoría de los errores originados en los requerimientos se detecta en las etapas avanzadas del proyecto. Él afirma que una parte significativa de estos errores se identifica durante la validación por el cliente. El esfuerzo de encontrar y corregir un defecto después de la entrega, suele costar a menudo 100 veces más que corregirlo cuando todavía se está desarrollando los requerimientos. Como se observó en los casos citados, los errores detectados con el software en operación pueden causar daños en varios órdenes de magnitud mayor al costo de corrección del defecto.

El objetivo es desarrollar el producto correcto en el primer intento. De esta manera, se reducen los errores en las primeras etapas del proyecto en donde se puede mitigar con mayor intensidad la reanudación. El método de ensayo y error, además de ser más caro y demorado, crea insatisfacción en el cliente. Por lo que se busca es evolucionar en la calidad del trabajo con requerimientos para lograr un impacto positivo en el costo, el tiempo y la satisfacción del proyecto.

Referencias y notas

[1] PMI - Project Management Institute. PMI's Pulse of the Profession: Requirements Management - A Core Competency for Project and Program Success, 2014.
[2] C. Jones. Software Defects Origins and Removal Methods, 2012.
[3] C. Jones, O. Bonsignour. The Economics of Software Quality. Addison-Wesley, 2012.
[4] K. Pohl, C. Rupp. Requirements Engineering Fundamentals. Rocky Nook Computing, 2011.
[5] D. Leffingwell. "Calculating the Return on Investment from More Effective Requirements Management". American Programmer 10(4); 13-16; 1997.
[6] B. Boehm & V. Basili, V. "Software Defect Reduction Top 10 List". Computer 34(1). IEEE, 2001.
[7] F. Brooks. The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1995.

Guilherme Siqueira Simões es socio de FATTO Consultoría y Sistemas, donde actúa en consultoría y entrenamiento en medición, estimación y requisitos de software. Es graduado en Ciencias de la Computación por la UFES, posgraduado en Gestión Empresarial por el IEL/UFES, certificado como experto en Puntos de Función por IFPUG y COSMIC, Director de Proyectos (PMP) por PMI e Ingeniero de Requisitos por IREB. guilherme.simoes@fattocs.com

Tips para Enfrentar la Problemática Actual en el **Desarrollo de Software**

Por Iván Lozada

El desarrollo de software es una de las áreas de especialización con mayor auge en las últimas décadas. Cada día la tecnología evoluciona, y al mismo tiempo, los campos de aplicación de los sistemas informáticos se van ampliando, pasando desde la realización de un portal de compras, hasta software para la automatización robótica para operaciones quirúrgicas, proyectos de explotación de millones de registros de información y automatización de procesos de todo tipo; dicho crecimiento se da tanto en la iniciativa privada como en el sector educativo y gubernamental.

Diversos organismos y comunidades han realizado un gran trabajo en definir modelos de procesos, metodologías, disciplinas, estándares buscando que las empresas desarrollen productos que cubran la satisfacción del cliente a través de proyectos exitosos, tales como: CMMI, ISO9001:2000, SPICE, Unified Process, ITmark, MoProSoft, CompetiSoft, COBIT, TSP/PSP, Scrum, Extreme programming, entre otros.

Adicionalmente se cuenta con herramientas ALM (Application Lifecycle Management) para apoyar en la gestión del ciclo de vida de desarrollo. Existen opciones comerciales de marcas como IBM, HPE, Atlassian y Thoughtworks, entre otros, así como algunas open source como Tuleap y Endeavour Agile ALM. A pesar de esto, el porcentaje de empresas en el mundo que utilizan tanto un modelo de proceso efectivo como herramientas para su gestión sigue siendo bastante pequeño.

Aun cuando podemos hacer uso de esta infraestructura que nos facilita la ejecución de un proyecto de software, existen diversos problemas a los cuales nos enfrentamos en los proyectos. Algunos de los más comunes son:

- Proyectos de software sin un alcance claro.
- Productos de software de baja calidad.
- Falta de métricas para estimar proyectos.
- Proyectos que rebasan significativamente las estimaciones iniciales de tiempo y/o costo.
- Desconocer habilidades y capacidades de las personas previo a la asignación de roles en el proyecto.
- Procesos para el desarrollo de software ineficientes.
- Falta de capacidades técnicas en los integrantes del equipo de trabajo.
- Gestión del proyecto inadecuada.

De acuerdo a mi experiencia existen algunas acciones clave que podemos realizar para mejorar el porcentaje de proyectos exitosos (aquellos que terminen en el tiempo, esfuerzo y alcance pactado), entre las cuales destacan las siguientes:

IMPLEMENTACIÓN DE UN MODELO DE PROCESOS ADECUADO

El mejor modelo de procesos a implementar es el que se ajuste a la forma de operar de la empresa. Cuando en un proyecto el equipo de trabajo enfoca más esfuerzo al proceso que al producto, es una señal de que se debe invertir esfuerzo en analizar y ajustar los procesos de desarrollo. Es por esto que el tomar la decisión de implementar un proceso debemos considerar que éste permita ser tan ligero o tan robusto como lo necesite el proyecto.

ENFOCAR A LOS EQUIPOS EN GENERAR PRODUCTOS DE ALTA CALIDAD

Existen diferentes estrategias para verificar la calidad de los productos de software, revisiones personales, revisiones entre pares, caminatas e inspecciones de código entre otras. Dentro de los procesos institucionales es recomendable definir y promover el uso de checklist de revisión para efectuar dichas estrategias, de esta manera el costo de la calidad se reducirá de manera significativa. Debemos cambiar la mentalidad de ver al área de QA como aquellas personas que encuentran mis errores, y verlos como aquellos que garantizan que el producto cumpla satisfactoriamente con lo que el cliente espera y pago por ello.

UTILIZAR ESTÁNDARES PARA GENERAR LOS PRODUCTOS DE SOFTWARE

El uso de estándares nos permite homologar la manera en la que construimos los productos requeridos en un proyecto de software, sin ellos, cada uno de los integrantes del equipo genera "a su manera" las actividades asignadas y puede ocasionar un gran descontrol; lo cual se transforma en re-trabajo, defectos e incremento de costos para todos los involucrados.

ANÁLISIS DE MÉTRICAS DE EJECUCIÓN DEL PROYECTO

Al finalizar cada proyecto, es muy recomendable realizar una sesión de trabajo para analizar el desempeño del equipo en el proyecto y obtener métricas de trabajo enfocados a la mejora continua de los involucrados, tales como desviación en la estimación

de esfuerzo, tiempo, datos enfocados a calidad del producto , cantidad de productos añadidos posteriores al arranque del proyecto, entre otros. De esta manera en proyectos posteriores se reducirá gradualmente el porcentaje de desviación e incrementaremos la satisfacción del cliente

CONTAR CON UN PLAN DE CARRERA Y CAPACITACIÓN

Los procesos necesitan personas para ser ejecutados, por lo que el capital humano es un factor crítico de éxito para los proyectos. En muchas ocasiones se piensa que al tener un modelo de proceso implementado es la solución, sin embargo, las habilidades y capacidades técnicas y administrativas de cada uno de los participantes en este proceso son primordiales. Recomiendo considerar los siguientes puntos:

- Plan de carrera: Tener planes de carrera individualizados permite establecer trazabilidad del crecimiento profesional de cada persona, y lo más importante, ellos lo saben, lo cual ocasiona certidumbre y confiabilidad en la institución; reduciendo la rotación de personal. En este punto es donde podemos incluir a los recién egresados como servicio social y practicantes, pensando en ellos a mediano plazo y planeando una estadía mínima de 5 años en la institución.

- Capacitación de acuerdo al rol que desempeñan: Como ya se mencionó anteriormente, los procesos necesitan personas y las personas conocimiento y habilidades, en caso de carecer o tener debilidad en algunas de ellas, dentro del plan de carrera se debería de considerar la capacitación y certificación del personal, generando un valor agregado a la institución y a las personas. La empresa no es la única responsable de capacitar al personal, deben incentivar a la capacitación a través de cursos en línea o ser autodidactas.

ESTABLECER VÍNCULOS DE CONFIANZA ENTRE INVOLUCRADOS

La confianza es la base de toda relación entre las personas. Al momento en que un cliente contrata a una empresa para realizar un proyecto y la empresa integra un equipo de trabajo para ejecutarlo, se están creando vínculos de confianza que debe ser mantenida durante todo el proyecto. Si los involucrados en el proyecto como cliente, usuarios, equipo de desarrollo y gerencia tienen comunicación efectiva, compromiso y caminan hacia un mismo objetivo, la confianza se mantendrá y fortalecerá. Es muy fácil perder la confianza, y recuperarla no garantiza que todo sea igual que antes; es como romper un jarrón y pegarlo, sigue siendo el mismo objeto pero quedan marcas de la ruptura.

ADECUADA GESTIÓN DEL PROYECTO

La gestión del proyecto es un tema crítico en el éxito o fracaso, podemos contar con personal con mucha experiencia y con excelentes conocimientos técnicos y un modelo de proceso institucionalizado y ejecutado. Sin embargo, si la gestión del proyecto, desde la fase de planeación e identificación del alcance, hasta el cierre, es deficiente, los problemas seguirán existiendo y caminaremos rumbo al fracaso y frustración. Al día de hoy contamos buenas prácticas enfocadas a esta área de conocimiento pero no siempre se lleva a cabo de la mejor manera; por lo que es importante que el administrador del proyecto y el líder técnico del equipo tenga en claro sus responsabilidades y exista un adecuado canal de comunicación.

Otro punto adicional es reducir el compromiso de fechas de entrega sin consultar el equipo de trabajo ya que generalmente esto ocasiona presión excesiva y que estos compromisos no se cumplan y por ende, la satisfacción del cliente no sea la deseada.

La operación día a día provocan que no realicemos un análisis de las cosas que realizamos bien y cuáles debemos mejorar en cada proyecto, lo cual ocasiona que sigamos cometiendo los mismos errores y muchas veces no los cuantificamos; es importante hacer un alto y reflexionar para poder realizar los ajustes donde sea necesario con la finalidad de finalizar proyectos exitosos y con alta calidad.

Referencias y notas

[1] G. Cuevas, et al. Gestión del Proceso Software. Editorial Centro de Estudios Ramón Areces, 2005.

[2] W. S. Humphrey, et al. Team Software Process (TSP) Body of Knowledge. http://swgu.ru/sm

Iván Lozada ha desempeñado diversas funciones tanto en el sector público como en la iniciativa privada, actualmente su función es TSP Coach en INFOTEC. ilozada@gmail.com

Special Purpose Languages

PARTE 6. MÁQUINAS ABSTRACTAS Y LENGUAJES

Por Luis Vinicio León Carrillo

Luis Vinicio León Carrillo es Director General y co-fundador de e-Quallity. Fue profesor-investigador en la universidad ITESO. Realizó estudios de posgrado en Alemania, durante los cuales abordó temas relacionados con la prueba de software y los métodos y lenguajes formales.

Comenzamos esta serie abordando los lenguajes formales de una manera más bien no-procedural, en términos algebraicos y de conjuntos; después lo hicimos de una forma más procedural, utilizando gramáticas. En este número los abordaremos con un enfoque muy procedural, mediante Máquinas de (Conjuntos de) Estados Finitos (o Finite States Machines), MEFs.

Pero antes de continuar: dado que en el número anterior no aparecieron referencias bibliográficas, las anexo al final de esta columna. Si quisieran profundizar en lo que abordaremos en este número, pueden consultar algún texto sobre Análisis y Diseño de Algoritmos (en particular la sección de NP-Completeness). Y (como creo haber dicho) si quisieran ahondar en los otros temas que hemos abordado pueden consultar textos sobre Autómatas y Lenguajes Formales y Compiladores.

Sin pretensión de tratar de ser exhaustivos aquí, y limitándonos solo a la formalidad suficiente para poder ser claros (aunque sacrificando precisión), vayamos a nuestro tema.

MÁQUINAS DE (CONJUNTOS DE) ESTADOS FINITOS (MEFS)

Las Máquinas de Turing son un tipo de MEFs con las que se abordó la cuestión de la "computabilidad" (otro enfoque fueron las funciones recursivas), tratando de descubrir los alcances y las limitaciones de las máquinas de cómputo que estaban construyéndose o por construirse (Turing publicó su trabajo más importante en 1936).

Las MEFs que presentaremos aquí tienen en común que:

- están constituidas por un conjunto finito de estados en los que la MEF puede encontrarse cuando procesa la cadena de entrada, hay un estado inicial en que la MEF comienza su procesamiento y uno o más estados en que puede terminar;
- los símbolos que conforman la cadena de entrada son elementos de un conjunto finito (el "alfabeto");
- las operaciones de la MEF se definen mediante una relación de transición ρ, que toma el estado en el que se encuentra la MEF en ese momento y el símbolo actual de la cadena de entrada y regresa el estado al que la MEF debe moverse; si la relación de transición ρ es una función, entonces se le denota con δ y se dice que la MEF es determinística, en caso contrario se denota con ρ y se dice que es no-determinística.

Aquí usaremos dos convenciones: a) considerar que la MEF termina su procesamiento indicando que no acepta la cadena de entrada cuando alcanza una transición en la relación/función de transición que no haya sido explicitada; y b) mostraremos la MEF mediante un grafo, en el cual los nodos serán los estados y en las aristas colocaremos los elementos que terminen de definir las transiciones de la relación/función de transición.

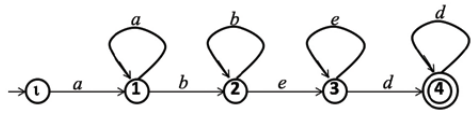
Veamos ahora las MEFs particulares.

AUTÓMATAS FINITOS (AFS)

Los AFs (*finite automata*) se definen como quíntuples de la forma $AF = \langle S, \iota, F, A, \rho \rangle$ en los cuales

S es el conjunto de e**S**tados,
ι ($\in S$) es el estado **I**nicial,
F ($\subset S$) es el conjunto de estados **F**inales,
A es el **A**lfabeto del lenguaje de la cadena de entrada,
ρ (ó δ) es la función de transición $\rho : (s_1, a) \rightarrow s_2$
donde $s_1, s_2 \in S, \quad a \in A$.

Como podrá verificarse, el siguiente AF es determinístico y procesa el lenguaje $a^m b^n e^i d^k$, para $m, n, i, k \geq 1$ e independientes entre sí.



El AF inicia el procesamiento en el estado ι ; lee una a de la cadena de entrada y pasa al Estado 1, en el cual lee cero o más a (la transición es un ciclo); después procesa una o más b, luego una o más e, y finalmente una o más d. Siguiendo nuestra convención, si nos encontráramos en el estado 2 y en la cadena de entrada no tuviéramos una b o una e el autómata no aceptaría la cadena y terminaría su procesamiento.

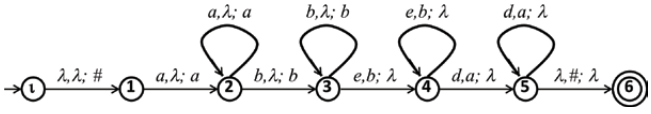
Se ha demostrado que los AFs procesan solamente los Lenguajes Regulares (pero todos ellos). En particular, no pueden procesar los Lenguajes Libres de Contexto porque no tienen memoria que les permita empatar caracteres por pares, como lo requieren este tipo de lenguajes. Se ha demostrado también que en el caso de los AFs el no-determinismo no implica mayor poder de procesamiento. Igualmente, se ha demostrado que para cada expresión regular existe un AF que procesa el mismo lenguaje y viceversa (de hecho, lo usual es que las herramientas que procesan textos mediante expresiones regulares primero las transformen en AFs –como las usadas para definir analizadores léxicos lex y flex).

AUTÓMATAS DE PILA (APS)

Los APs (ó Push Down Automata) se definen como séxtuples de la forma $AP = \langle S, \iota, F, A, \Sigma, \rho \rangle$ en los cuales

S es el conjunto de e**S**tados,
ι ($\in$ S) es el estado **I**nicial,
F ($\subset$ S) es el conjunto de estados **F**inales,
A es el **A**lfabeto del lenguaje de la cadena de entrada,
Σ es el alfabeto de la pila (o stack), que es el tipo de memoria que tienen los APs ($\# \in \Sigma$, $A \subset \Sigma$),
ρ (ó δ) es la función de transición $\rho : (s_1, a, \sigma_1) \rightarrow (\sigma_{2,} s_2)$
donde $s_1, s_2 \in S$, $a \in A$, $\sigma_1, \sigma_2 \in \Sigma$.

Como podrá verificarse, el siguiente AP es determinístico y procesa el lenguaje $a^i b^n e^n d^i$, para $n, i \geq 1$.



Iniciamos en ι y pasamos al estado 1 sin leer símbolos la cadena de entrada (eso significa la primera λ de la primera transición), no sacamos elemento alguno de la pila (eso significa la segunda λ) y metemos "#" a la pila. Del estado 1 pasamos al estado 2 leyendo una a de la cadena de entrada sin sacar elemento alguno de la pila y metiendo una a a la pila. En el estado 2 repetimos cero o más veces lo siguiente: leer una a de la cadena de entrada, no sacar elemento alguno de la pila y meter una a a la pila. En las siguientes transiciones (hasta la del estado 3) ejecutamos un proceso semejante al anterior pero ahora con las b: por cada b que leemos de la cadena metemos una b a la pila. Las 2 transiciones subsiguientes (hasta la del estado 4) se ocupan de sacar una b de la pila por cada e que se lee de la cadena de entrada. Las 2 transiciones subsiguientes (hasta la del estado 5) sacan una a de la pila por cada d que lean de la cadena de entrada. La última transición no lee elemento alguno de la cadena de entrada y saca el # de la pila sin meter elemento alguno en ella.

Se ha demostrado que los APs procesan solamente los Lenguajes Libres de Contexto (pero todos ellos). En particular, no pueden procesar los Lenguajes Sensibles al Contexto porque la pila sólo les permita empatar caracteres por pares anidados y no en tercias o por pares no anidados, como lo requieren algunos lenguajes de este tipo.

También se ha demostrado que para los AP el no-determinismo sí implica mayor poder de procesamiento, por lo que en los Lenguajes Libres de Contexto (LLC) tenemos los LLC no-deterministicos englobando a los LLC deterministicos. Un caso concreto es el lenguaje $a^i b^i \cup a^i b^{2i}$, para $m, i \geq 0$ que es un LLC para el que no existe un AP determinístico que lo procese.

Igualmente, se ha demostrado que para cada Gramática Libre de Contexto existe un AP que procesa el mismo lenguaje, y viceversa (de hecho, lo usual es que las herramientas que procesan ese tipo de gramáticas "determinísticas" primero las transformen en APs determinísticos –como las usadas para definir analizadores sintácticos yacc y bison).

MÁQUINAS DE TURING (MTS)

Las MT se definen como séxtuples de la forma
$MT = \langle S, \iota, \eta, A, \Pi, \rho \rangle$ en los cuales

S es el conjunto de e**S**tados,
ι ($\in$ S) es el estado **I**nicial,
η es el estado Final,
A es el **A**lfabeto del lenguaje de la cadena de entrada,
Π es el alfabeto de la cinta de Procesamiento, el tipo de memoria que tienen las MTs $(\#, \$ \in \Sigma ; A \subset \Pi)$,
ρ (ó δ) es la función de transición $\rho : (s_1, \pi_1) \rightarrow (\pi_2, \mu, s_2)$, para $s_1, s_2 \in S$, $\pi_1, \pi_2 \in \Pi$, $\mu = L$ (mover la cabeza a la celda de la izquierda) ó μ = R (moverla a la de la derecha).

En el caso de las MT tomaremos la convención de que la cadena de entrada se coloca en la cinta de la siguiente manera: primero se coloca un "#", luego la cadena de entrada, y un "$" después de ella.

Se ha demostrado que las MTs procesan solamente (pero todos) los Lenguajes Estructurados por Frases (o "Recursivamente Enumerables"). En particular, no pueden procesar los Lenguajes Generales (así como no son susceptibles de ser definidos con reglas, tampoco lo son de ser procesados por máquinas).

En el caso de las MTs el no-determinismo no implica mayor poder de procesamiento.

Los trabajos con las MTs han dado lugar a lo que se conoce como la Tesis de Turing:

> *Las MTs modelan exactamente lo que pueden realizar las computadoras.*

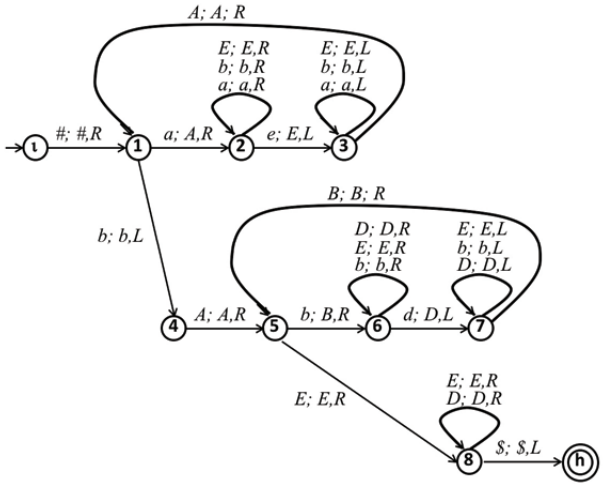
Esta tesis es muy relevante y útil: aún cuando (se asume que) las MTs tienen el mismo poder de procesamiento que las computadoras, estas máquinas abstractas son muy simples en su funcionamiento, lo cual facilita realizar demostraciones sobre las capacidades y limitaciones de las computadoras.

En las MT tenemos varias clases muy importantes, que procedemos a revisar.

Autómatas Delimitados Linealmente (ADLs)

Los ADLs (ó Linear Bounded Automata) son MTs que para procesar una cadena solo requieren una cantidad de celdas en la cinta de procesamiento prácticamente igual a la de la cadena de entrada. Los ADLs pueden procesar los Lenguajes Sensibles al Contexto.

Como podrán verificar, el siguiente ADL es determinístico que procesa el lenguaje $a^i b^n e^i d^n$, para $n, i \geq 1$.



La MT inicia en ι y pasa al Estado 1 leyendo el "#" que antecede a la cadena de entrada, escribiendo el mismo "#" y moviéndose a la celda de la derecha.

Del estado 1 pasa al estado 2 leyendo una a de la cadena de entrada, escribiendo una A y moviéndose a la derecha. En el estado 2 podemos repetir cero o más veces algunas de las siguientes operaciones (en realidad deberíamos dibujar 3 ciclos en el autómata, uno para cada transición, pero por motivos de espacio dibujamos solo uno y apilamos las transiciones):

1. leer una a de la cadena de entrada, escribir una a y moverse a la derecha (o sea: nos saltamos las a);
2. leer una b de la cadena de entrada, escribir una b y moverse a la derecha (nos saltamos las b);
3. leer una E de la cadena de entrada, escribir una E y moverse a la derecha (nos saltamos las E).

Después pasamos al estado 3 leyendo la primera e, transformándola en E y moviéndonos a la izquierda; luego nos movemos hacia la izquierda saltándonos todas las a, b, y E hasta encontrar una A, nos movemos a la derecha y llegamos nuevamente al estado 1.

Enseguida podemos repetir cero o más veces las operaciones de las transiciones de los estados 1, 2 y 3, las cuales se pueden resumir como "cambiar una e por E por cada a que se haya cambiado antes por A (con ello verificamos si hay igual cantidad de a que de e). Dejamos de hacer eso cuando encontramos una b; entonces escribimos b, nos movemos a la izquierda y vamos al estado 4. Leemos una A, escribimos una A y vamos al estado 5.

En las transiciones de los estados 5, 6 y 7 realizamos con y las b y d algo semejante a lo que hicimos con las a y e en los estados 1, 2 y 3: "cambiar una d por D por cada b que se haya cambiado antes por B" para verificar si hay igual cantidad de b que de d. Dejamos de hacer eso cuando encontramos una E; entonces escribimos una E, nos movemos a la izquierda y vamos al estado 8, donde nos saltamos todas las D y E moviéndonos a la derecha hasta encontrar el # que debe venir después del final de la cadena de entrada.

MÁQUINAS DE TURING QUE TERMINAN (MTTERM) – ALGORITMOS

Las MT_{term} son máquinas de Turing que siempre terminan cuando procesan cualquier cadena —aunque eventualmente después de mucho tiempo. Las MT_{term} pueden procesar los Lenguajes Decidibles (o "Recursivos").

La penúltima frase puede parecer obvia, especialmente con tanta ciencia ficción y con tan altas expectativas puestas sobre la nouvelle Artificial Intelligence, pero ocurre que hay problemas que una MT simplemente no puede resolver (y, de acuerdo a la Tesis de Turing, tampoco las computadoras). Un ejemplo es la demostración de teoremas en el Cálculo de Predicados de Primer Orden, que describiremos aquí rápida e informalmente: desde hace décadas tenemos sistemas de inferencia (finalmente, MTs) que pueden generan todos los teoremas en ese Cálculo; inicialmente se pudo pensar que, para saber si una fórmula era un teorema, simplemente bastaría con comparar esa fórmula con cada uno de los teoremas que fuera generando el sistema; y efectivamente, si la fórmula es un teorema, en algún momento (quizás en un día, quizás en 1,000,000 de años) el sistema anunciará que encontró un teorema que es equivalente a nuestra fórmula; sin embargo, si la fórmula no es un teorema, el sistema (la MT) nunca terminará porque no generará un teorema equivalente a nuestra fórmula.

A partir de algo como lo anterior, es que se suele definir Algoritmo como una MT que siempre termina (MT_{term}).

PROBLEMAS, ALGORITMOS Y HEURÍSTICAS

Ahora bien: hay problemas para los cuales no se han encontrado algoritmos eficientes que los resuelvan (y se presume que no existen). En esos casos se suele optar por abstenerse de buscar la solución óptima mediante la revisión de todas las alternativas en el grafo de búsqueda, y en su lugar recorrer solo parte de él para encontrar buena solución. A estos métodos solemos llamarlos Heurísticas.

Para ejemplificar lo anterior tomemos el Problema del Vendedor Viajero (PVV): Un Vendedor debe visitar n-1 ciudades partiendo de una inicial, considerando que viajar de una ciudad a otra tiene un costo asociado.

El PVV se puede modelar mediante un grafo dirigido con pesos en las aristas.

n =	1	10	100	1,000	10,000	100,000	1,000,000	1.E+07	1.E+08	1.E+09	1.E+10	1.E+11	1.E+12
$log_2 n$	0.0	3.3	6.6	10.0	13.3	16.6	19.9	23.3	26.6	29.9	33.2	36.5	39.9
n	1.0	10.0	100.0	1,000.0	10,000.0	100,000.0	1.0E+06	1.0E+07	1.0E+08	1.0E+09	1.0E+10	1.0E+11	1.0E+12
n^2	1.0	100.0	10,000.0	1.0E+06	1.0E+08	1.0E+10	1.0E+12	1.0E+14	1.0E+16	1.0E+18	1.0E+20	1.0E+22	1.0E+24
2^n	2.0	1.0E+03	1.3E+30	1 E+(3E+2)	1 E+(3E+3)	1 E+(3E+4)	1 E+(3E+5)	1 E+(3E+6)	1 E+(3E+7)	1 E+(3E+8)	1 E+(3E+9)	1 E+(3E+10)	1 E+(3E+11)

Tabla 1. Valores para curvas exponenciales

Sabemos que, dado un grafo dirigido con n nodos, la cantidad máxima de aristas (dirigidas) es n (n+1), pero: cuántas rutas hay en ese grafo? Eso nos permitiría saber cuántas rutas tenemos que comparar para escoger la de costo mínimo.

Para simplificar nuestro análisis, pensemos que el costo para ir a de una ciudad a otra es el mismo de ida que de vuelta. Tenemos entonces un grafo no dirigido con n nodos, por lo cual la cantidad máxima de aristas es n (n+1) / 2 . Ahora bien, cuando el grafo:

- tiene 3 nodos, partiendo del nodo inicial solo tenemos 2 rutas para visitar los otros 2 nodos.
- tiene 4 nodos, partiendo del inicial podemos iniciar 3 rutas yendo a cada uno de los otros 3 nodos, y entonces nos quedan 2 nodos por visitar en cada caso que como vimos en el punto anterior, genera 2 rutas cada uno; por tanto tendríamos 6 rutas.
- tiene 5 nodos, partiendo del inicial podemos iniciar 4 rutas yendo a cada uno de los otros 4 nodos, y entonces nos quedan 3 nodos por visitar en cada caso que como vimos en el punto anterior, genera 2 rutas cada uno; tendríamos entonces 24 rutas.

Supongamos que en general, si el grafo tiene n nodos, habrá (n-1)! (= 1*2*3*...*n-1) rutas.

En la tabla 1 se muestran los valores para las curvas $log_2 n$ (la búsqueda en un árbol binario con n datos requiere una cantidad de operaciones múltiplo de $log_2 n$), n, n^2 (ordenar n datos con el método de "burbuja" requiere una cantidad de operaciones múltiplo de n^2), y 2^n ($n! > 2n$ a partir de n=4).

Los numeros en azul son aproximaciones mías (suficientemente acertadas para nuestro propósito); el resto los produjo la computadora (el primero de ellos se leería "uno por diez elevado a la tres por diez elevado a la $2" = 1*10^{300}$).

Hoy en día hablamos de Internet of Things y big data. Supongamos que nos topamos con un problema equivalente al PVV con una cantidad de ciudades igual a 10^9 (si cada ciudad se llevara un Byte, entonces estamos hablando de un gigabyte, nada extraordinario).

Para hacer nuestro análisis sin problemas de desbordamiento, tomemos valores no de $n!$ sino de n^2 (para $n=100$, $n!$ vale casi 10^{158}). Haciéndolo así, con un gigabyte de ciudades, el algoritmo que revisara todas las rutas para encontrar la óptima tendría que realizar $10^{300,000,000}$ operaciones (ver la tabla). ¿Eso implica mucho, o poco tiempo?

Supongamos que podemos paralelizar mil millones de supercomputadoras que pueden realizar cada una 10^{21} operaciones por segundo (un millón de PetaFlops). En un año todas ellas podrían realizar $10^9*365*24*60*60*10^{21} = 3.1536 * 10^{37}$ operaciones, así es que (asumiendo que una operación bastara para procesar una ruta completa) para obtener la ruta óptima para un grafo con 10^9 nodos, el algoritmo del PVV se tardaría solamente $10^{300,000,000-37}$ años. Esperar a obtener el resultado es, al menos, impráctico. Por supuesto, la estrategia a seguir sería utilizar una Heurística.

Con los elementos del análisis anterior estamos en posición de responder la pregunta que nos había quedado pendiente en un número anterior, acerca de la eficiencia con la que podemos procesar los distintos tipos de lenguajes que hemos venido revisando. La ofrecemos en la siguiente tabla, considerando una cadena de entrada de longitud n (como dijimos al principio, debido a que nos limitamos solo a la formalidad suficiente, es necesario sacrificar precisión):

Lenguajes	**Cantidad de operaciones proporcional a..**
Regulares	$log\ n$ ("logarítmica") para algunos de esos lenguajes
Libres de contexto determinísticos	n ("lineal") para casi todos esos lenguajes
Libres de contexto no-determinísticos	n^c ("polinomial") en el caso general, $c=3$
Sensibles al contexto	k^n ("exponencial") en algunos casos, $k=2$
Recursivos	k^n ("sobre-exponencial") en muchos casos, $k>2$
Recursivamente enumerables	∞ ("infinito") en el caso general (la cadena no forma parte del lenguaje)

Tabla 2. Eficiencia para probar distintos lenguajes.

Recuerda que si quieres discutir sobre este artículo, puedes hacerlo por medio de comentarios en la versión online del sitio de Software Guru. Si hay fé de erratas, también las compartiremos ahí.

Referencias

[1] Luis Vinicio León Carrillo: Traducción de Frases del Español al Francés en un Dominio restringido; Tesis de Licenciatura, ITESO, 1990

[2] Pierre Janton: Einfürhrung in die Esperantologie; Georg Olms, 1993.

[3] Roland Hausser: Foundations of Computational Linguistics; Springer, 1999.

Construcciones Reproducibles

Por Gunnar Wolf

Gunnar Wolf es administrador de sistemas para el Instituto de Investigaciones Económicas de la UNAM y desarrollador del proyecto Debian GNU/Linux.
http://gwolf.org

El premio Turing [1] de 1983 fue otorgado a Ken Thompson y Dennis Ritchie por «su desarrollo de la teoría genérica de los sistemas operativos, y específicamente, por la implementación del sistema operativo Unix». Su discurso de aceptación del premio, «Reflections on Trusting Trust» [2] (pensamientos acerca de confiar en la confianza) ha sido uno de los pilares de la práctica de la seguridad informática.

En dicho trabajo, Thompson sostiene que mientras haya gente como él, capaces de escribir un compilador a pelo, no podremos confiar auditoría alguna que hagamos a nuestro código —su artículo demuestra cómo se puede troyanizar un compilador para ocultar en él comportamiento maligno que sólo se dispara al compilar un programa determinado— o al compilar a una nueva copia del compilador mismo.

Si bien ya ha llovido en los más de treinta años que han transcurrido desde este artículo [3] el principal argumento de Ken Thompson se sostiene: en el supuesto de que hagamos una auditoría al código de determinado proyecto, ¿qué garantiza que ese sea el código que realmente ejecuta el despliegue productivo?

El planteamiento que hago no tiene nada de hipotético, y seguramente muchos de ustedes podrán encontrar ejemplos de código que no concuerda a detalle con el que pasó un proceso de certificación; parte del papel de quien audite y certifique un sistema debe ser una forma de asegurar que el software estudiado sea efectivamente el empleado.

EL ECOSISTEMA DE LAS DISTRIBUCIONES DE SOFTWARE LIBRE

Conviene mencionar, pues, el punto de partida del proyecto del cual voy a hablar en esta columna. El proyecto de *Construcciones Reproducibles* [4] nace de la inquietud de un grupo de desarrolladores de software libre, en un principio mayormente desarrolladores de la distribución de Linux *Debian*.

Debian es uno de los proyectos de software libre más grandes y longevos que hay que sigue llevándose a cabo como hace veinte años: como un esfuerzo comunitario, colaborativo; como la suma de miles de pequeñas contribuciones personales. No existe una compañía detrás de Debian, y todos quienes participamos en su desarrollo lo hacemos como voluntarios y a título personal.

Esto puede verse tanto como una ventaja (el soporte para cada una de las ideas que forman parte del proyecto depende del interés de un individuo, no hay una decisión corporativa que pueda "matar" ideas que no generen ingresos) como una desventaja (al depositar la confianza de nuestra infraestructura en un proyecto tan profundamente descentralizado, estamos confiando en *cada uno de los individuos* involucrados). Y las *Construcciones Reproducibles* nacen como respuesta a esta desventaja.

La principal garantía de funcionamiento *no-malicioso* que ofrecen los proyectos de software libre a sus usuarios es que, dado que el código fuente está disponible, cualquier usuario preocupado de que su sistema pueda estar de alguna manera troyanizado o capturando y reportando datos sin su autorización puede inspeccionar el código fuente de los programas que emplea y compilarlos. Como dicen, *¡pare de sufrir!* Sin embargo... esta solución no escala.

Hay distribuciones de Linux, como Gentoo o Arch, que basan su modelo de operación únicamente en la distribución de *fuentes*, y cada uno de los usuarios compila los paquetes al instalarlos. Estas distribuciones gozan del favor de los usuarios con inclinaciones más técnicas, pero si bien ganan en flexibilidad, resultan imprácticas para su uso en producción para la mayor parte de los operadores. La mayor parte de las distribuciones opta por la distribución de *binarios*, paquetes precompilados, con información para una resolución automática de dependencias.

Todas las distribuciones de Linux hoy en día proporcionan repositorios de software con verificaciones criptográficas fuertes, asegurando que éste fue construido en el sistema que sus políticas de desarrollo depositen su confianza. Para muchos, y por muchos años, eso ha sido suficiente.

¡NO ES PARA QUE CONFÍEN EN MÍ!

Sin embargo, en junio de 2013, Mike Perry (desarrollador de la red anonimizadora Tor [5] pasó dos meses afinando el navegador derivado de Firefox que emplea Tor para que, si se compila dos veces, el resultado sea idéntico bit por bit. Al explicar su motivación para hacer esto, envió un mensaje [6] del cual traduzco y reproduzco porciones a continuación:

No pasé seis semanas logrando una construcción reproducible para demostrar que soy honesto o confiable. Lo hice porque no creo que los modelos de desarrollo de software basados en la confianza a una única persona puedan estar seguros contra adversarios serios.

Los últimos años hemos visto un sostenido crecimiento en el (...) uso de explotación de software por múltiples gobiernos. (...)

Esto significa que el desarrollo de software debe evolucionar más allá del "confía en el repositorio firmado por GPG de mi máquina confiable". (...) Aquí es donde las construcciones reproducibles entran en juego: cualquier individuo puede usar nuestra red de anonimato, bajar nuestro código fuente, verificar ante una serie de repositorios públicos, firmados, auditados y replicados, y reproducir nuestra construcción exactamente, asegurándose no ser víctima de tales ataques dirigidos.

Hay antecedentes previos, pero este es el primer esfuerzo serio y dedicado para lograrlo; quien esté familiarizado con la red Tor comprenderá la importancia que históricamente han dado a una fuerte protección al anonimato mediante herramientas técnicas apropiadas. Perry puso el tema sobre la mesa, y pocos meses más tarde, Jérémy Bobbio comenzó a explorar lo que haría falta para aplicar las modificaciones y principios que Perry fue descubriendo y documentando al repositorio Debian —formado por cerca de 25,000 paquetes independientes.

El trabajo iniciado por Jérémy demostró resonar en la mente de diversos desarrolladores interesados en la seguridad. El proyecto aceleró velozmente, y dos años más tarde habían ya logrado una amplia concientización respecto al problema y lograr la construcción de la mayoría del archivo; participantes de otros proyectos de software libre comenzaron a verlo con interés, y a fines de 2015 celebraron un primer congreso en Atenas.

Si bien Debian es el proyecto con mayor avance, al día de hoy están trabajando en conjunto con Coreboot, OpenWRT, NetBSD, FreeBSD, Archlinux y Fedora. En consonancia con el tema de este número de *Software Gurú*, pueden ver en la herramienta de integración continua [7] el impresionante grado de avance que han logrado: Al momento de escribir este artículo, más del 93% de los paquetes de la próxima versión estable de Debian logra una construcción reproducible; apenas hace un año y medio la meta era llegar a un 80%.

¿Y DÓNDE RADICA EL PROBLEMA?

Muchos de ustedes recordarán sus años formativos, y puede que cueste un poco ver el problema. Estamos hablando de la compilación de software. Si yo tengo el mismo código fuente y lo compilo dos veces, ¿no debería obtener exactamente el mismo resultado? ¿Por qué este punto resulta problemático?

Hagan la prueba. Compilen su proyecto favorito. Guarden el resultado en un *tar.gz* (o en un *zip*, dependiendo de su sistema favorito). Salgan al patio, tómense un café, y vuelvan a hacerlo. Hagan un *checksum* sencillo de los archivos. Les garantizo que será diferente. Imaginen ahora hacerlo desde la computadora de un compañero de trabajo, o de arbitrariamente cualquier otra computadora en el mundo (que tenga el mismo compilador instalado).

Algunas de las fuentes de variación que el proyecto de *construcciones reproducibles* ha tenido que abordar son:

- La hora y fecha de compilación
- Algunas de las variables de entorno (TZ, LANG, USER, etc.)
- El nombre del directorio desde donde se compila
- El ordenamiento de los archivos que presenta el sistema de archivos subyacente
- Nombre de la computadora
- Versión del kernel
- ID del usuario y grupo, permisos de sesión (p.ej. umask)

Claro, parte importante del trabajo ha radicado en identificar y aislar factores que llevan a que una construcción no sea reproducible; si el tema les interesa, los invito a probar la sorprendente herramienta *diffoscope* [8], que encuentra y presenta las diferencias entre dos archivos de entre un tipo impresionante de formatos.

EN SUMA

El problema planteado por Thompson puede ampliarse, dada la realidad del mundo interconectado en el que vivimos, a que a un agente suficientemente poderoso le basta hacerse del control en *una* computadora (la de un desarrollador de software que trabaje dentro de una distribución de Linux, en nuestro ejemplo) para, desde ella, hacerse de privilegios elevados en millones de computadoras en todo el mundo. Las construcciones reproducibles harán evidente cuando un ataque de este tipo se dirija al producto binario.

Y no es que examinar a detalle el código fuente buscando código malicioso sea sencillo, pero... por lo menos, es posible hacerlo. Este proyecto plantea un fuerte cambio en el modelo de seguridad que ofrecen los proyectos de software libre, con el que todos los usuarios ganarán mucho mayor confianza en que el software que corren está efectivamente libre de código malicioso.

Referencias y notas

[1] Al que comúnmente se hace referencia como el premio Nóbel de la computación
[2] https://doi.org/10.1145/358198.358210
[3] Para los interesados en el argumento académico, David A. Wheeler hizo su tesis doctoral en 2009 acerca de cómo «contrarrestar la confianza en la confianza mediante la doble compilación diversa», https://www.dwheeler.com/trusting-trust/
[4] https://reproducible-builds.org/
[5] https://torproject.org/
[6] https://mailman.stanford.edu/pipermail/liberationtech/2013-June/009257.html
[7] https://tests.reproducible-builds.org/
[8] https://diffoscope.org/

Inyección de Dependencias

Por Herminio Heredia

Los frameworks de desarrollo se han vuelto populares por la facilidad con la cual puedes crear prototipos y aplicaciones con componentes poco acoplados. Para lograr ese bajo acoplamiento es muy común que los framework utilicen una estrategia que se conoce como inyección de dependencias.

QUÉ ES LA INYECCIÓN DE DEPENDENCIAS

La inyección de dependencias es un principio que no pertenece a un lenguaje o framework en particular, así que es común encontrar implementaciones de este principio en frameworks de distintos lenguajes. La idea detrás de la inyección de dependencias es que las clases o componentes de un sistema no deben de configurar sus dependencias de forma estática sino que deben de ser configuradas desde el exterior.

Para profundizar en el tema me valdré de un ejemplo muy sencillo pero que nos permitirá comprender el problema que resuelve la inyección de dependencias y de esa forma crear una definición.

Vamos a suponer que en nuestro sistema tenemos una clase llamada GenericClass que requiere escribir en una bitácora y para ello utiliza una clase diseñada para este propósito llamada Logger. La forma más convencional de expresar este sencillo modelo es mediante una asociación como lo muestra la figura 1.

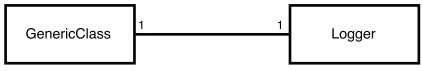


Figura 1. Asociación

El problema con este diseño es que es poco flexible ya que si hacemos cambios en Logger tal vez también tengamos que hacer cambios en GenericClass. A este efecto se le llama alto acoplamiento.

Algunos de los problemas que puedes tener con este diseño son los siguientes.

- Cambiar o actualizar Logger implica cambios en GenericClass.
- La dependencia con Logger debe de estar disponible en tiempo de compilación.
- GenericClass tiene la responsabilidad de crear una instancia de Logger.
- No se puede probar de forma aislada GenericClass.

La forma en la que se pueden resolver estos problemas, implica que GenericClass, en lugar de depender directamente de las implementaciones de Logger, solo dependa de una abstracción que puede implementar Logger. Este cambio se muestra en la figura 2.

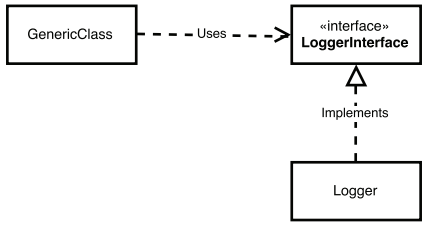


Figura 2. Dependencia a interface.

Ahora GenericClass solo depende de una interfaz, esto nos permite reducir el acoplamiento entre clases y darle flexibilidad a nuestro diseño. De esta forma podemos cambiar el comportamiento de Logger sin necesidad de cambiar completamente GenericClass.

En la práctica este diseño requiere que GenericClass ya no tenga el control para crear una instancia de Logger sino que se indica en la declaración de la clase, lo que obliga a que la creación de Logger esté fuera de GenericClass. Esto es lo que se conoce como inyección de dependencias ya que se requiere de otro elemento de software que proporcione una instancia de Logger que pueda utilizar GenericClass.

Con lo anterior podemos crear una definición simple y sencilla de comprender.

"La inyección de dependencias es cuando los componentes de un sistema reciben sus dependencias mediante su constructor, métodos o sus propiedades y dichos componentes no obtienen sus dependencias por ellos mismos."

La inyección de dependencias reduce el acoplamiento entre clases, de manera que los cambios en una clase no afecten a la otra. Asimismo, las clases pueden ser probadas de forma aislada.

TIPOS DE INYECCIÓN DE DEPENDENCIAS

De acuerdo a la definición se puede deducir que existen tres tipos de inyección, que podemos apreciar en los listados 1, 2 y 3. En realidad, la inyección vía propiedades no es muy recomendable dado que expone detalles del cliente, pero la incluimos simplemente como ejemplo.

```php
class MyClass {
  private $logger;
  public function __construct(LoggerInterface $logger) {
    $this->logger = $logger;
  }
  public function send($message) {
    return $this->logger->info($message);
  }
}
```

Listado 1. Inyección vía constructor

```php
class MyClass {
  private $logger;
  public function setLogger(LoggerInterface $logger)  {
    $this->logger = $logger;
  }
  public function send($message) {
    return $this->logger->info($message);
  }
}
```

Listado 2. Inyección vía métodos

```php
class MyClass {
    public $logger;
     public function send($message) {
        return $this->logger->info($message);
    }
}
```

Listado 3. Inyección vía propiedades

La inyección de dependencias nos permite reducir el acoplamiento de los componentes de un sistema permitiendo que estos sean más reutilizables. Es aconsejable su uso en sistemas grandes pero en sistemas complejos puede ser problemático ya que las dependencias se crean en otro punto dentro de la aplicación.

Herminio Heredia (@HerminioHeredia) es Ingeniero en Sistemas Computacionales, desarrollador freelance y coordinador en laraveles.com

WIZELINE

somos
HECHO EN MÉXICO

En Wizeline queremos construir un nuevo Silicon Valley justo aquí, en México, y necesitamos tu ayuda.

Buscamos a los mejores ingenieros de software, diseñadores de UX y científicos de datos.

Súmate a nuestra misión de construir el software más inteligente del mundo en nuestras oficinas en México, San Francisco y Vietnam.

¿Aceptas el reto? ¡Únete a nuestro equipo!
www.wizeline.com/tacos

COMMUNITY (AND TACOS)

```
$ git merge silicon-valley mexico
```

1

UPRIGHT GO

"Ai mijito(a), enderézate". Sí, todos hemos aquí hemos oído esas palabras, especialmente cuando estamos en la computadora o leyendo algo en el teléfono. Upright GO viene a reemplazar esa voz de forma discreta y conveniente. Es un pequeño wearable que monitorea la postura de tu columna vertebral y cuando lo tienes en modalidad de entrenamiento te avisa (por medio de una ligera vibración) cuando tienes mala postura para que la corrijas. El objetivo es que desarrolles memoria muscular, para poco a poco mejorar tu postura. Tiene una app móvil compañera a través de la cual puedes monitorear tu postura a través del tiempo. El dispositivo tiene un tamaño de alrededor de 5 centímetros y un peso de 12 gramos, y se adhiere a tu piel. Es resistente al agua y usa una batería recargable que dura aproximadamente 10 días. Actualmente está en Kickstarter a un precio de 59 dólares con planes de entrega para agosto.

http://swgu.ru/s-

2

VCONDUIT

VConduit es un regulador de energía eléctrica compacto y programable que puedes utilizar para dar energía a todo tipo de microcontroladores y motores eléctricos. Simplemente conecta el dispositivo a tu computadora y desde una app controla el voltaje de salida. VConduit tiene un rango de voltaje de 1.2 a 25 volts y se puede usar fácilmente en protoboards (en un extremo tiene 3 pines listos para conexión a protoboard) o conectar por medio de soldadura. Actualmente está en Kickstarter a un precio de 25 dólares con planes de entrega para junio.

http://swgu.ru/t0

3

AMPLIFI HD WI-FI ROUTER

Como sabes, los ruteadores WiFi caseros que incluyen la mayoría de los proveedores de internet son bastante limitados en rango, velocidad y cantidad de nodos soportados. Ante ello, algunos optan por adquirir equipos de nivel empresarial con mejor desempeño o utilizan tecnología mesh para agregar más antenas; el problema es que estos equipos típicamente son feos, caros y/o complicados de configurar. Afortunadamente están saliendo al mercado nuevas alternativas sencillas de configurar y bonitas. Una de ellas que nos gusta particularmente es el sistema AmpliFi HD.

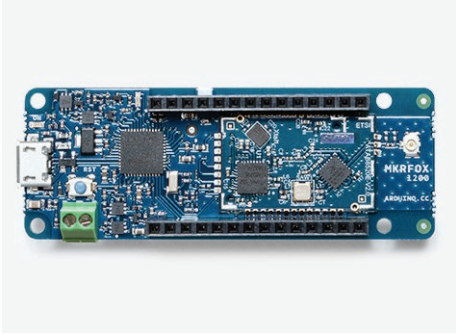
El sistema AmpliFi HD consta de un ruteador WiFi bastante atractivo al ojo, que tiene un arreglo de 3x3 antenas MIMO 802.11AC. El sistema es extendible usando "mesh points" que son antenas que simplemente conectas a un enchufe eléctrico en la zona donde tengas mala recepción, y automáticamente se conectan y coordinan con el ruteador central. El sistema es configurable desde una app móvil o desde una computadora por medio del navegador.

https://amplifi.com

4

RAZER BLADE

Durante varios años hubo un fuerte éxodo de desarrolladores que cambiaron de usar laptop PCs hacia MacBooks. Pero últimamente, entre que Apple no ha impresionado precisamente con sus equipos más recientes, adicionado a que tanto Windows como Linux en el desktop han mejorado considerablemente, parece ser que el segmento de laptop PCs está resurgiendo. Esto es reforzado por nuevas ofertas de laptops con gran desempeño a precios "relativamente" accesibles. Una de las opciones que más nos atrae por el momento es la Razer Blade, una laptop para gaming con gran balance de poder y portabilidad. La edición 2017 viene con procesador Intel i7 (Kaby Lake), pantalla Full HD mate (disponible también en 4K multi-touch), gráficos NVIDIA GeForce GTX 1060, 16 GB de memoria DDR4 a 2,400 MHz, y cuenta con puertos USB-C, USB 3 y HDMI 2. Todo este poder envuelto en un elegante chasis de aluminio dando un peso menor a 2 kg es una opción que vale la pena considerar si planeas renovar tu laptop próximamente. La versión básica con SSD de 256 GB ronda los 1,900 USD y la versión con pantalla touch 4K y 1 SSD de 1 TB queda en cerca de 2,800 USD. Por el momento, esta edición no está disponible en Latinoamérica así que mientras eso cambia, considera ordenarla online.



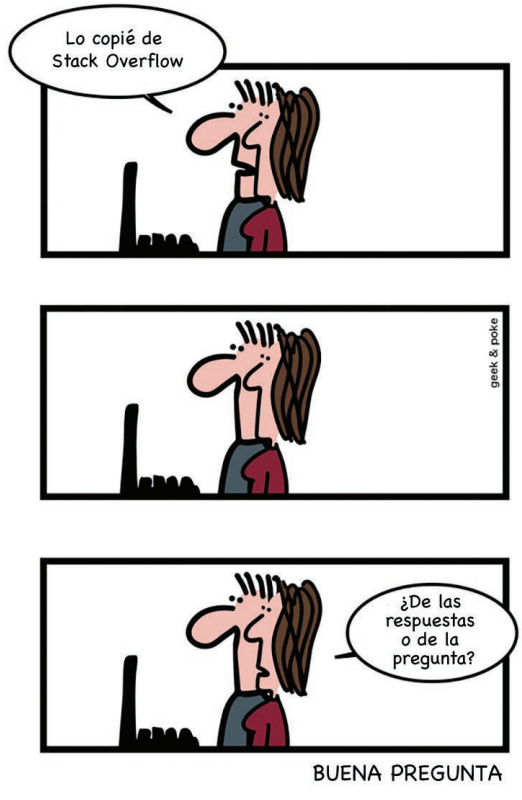
5

MKRFOX1200

La MKRFOX1200 es una tarjeta que combina la funcionalidad de la Arduino MKRZero con conectividad Sigfox, brindando una solución ideal para makers interesados en diseñar proyectos de Internet de las cosas usando esta tecnología de red. Si no conoces Sigfox, es una tecnología para redes de área ancha (WAN) con bajo consumo de energía, por lo que es considerada una buena opción para IoT (otra alternativa es LoRA). La MKRFOX1200 está basada en el microchip SAMD21 con un módulo Sigfox ATA8520. Destaca por su tiempo de vida de batería, con una solución típica dando un rendimiento de más de 6 meses con dos baterías AA de 1.5 volts.

http://swgu.ru/t1

Humor



Original en http://geek-and-poke.com/geekandpoke/2016/11/27/good-questions

Ir a la Guerra con **Fusil**

La importancia de un portafolio y de prácticas profesionales

Por Diego Maury

Debido al ritmo acelerado de estudiantes graduados en el mundo y el hecho de que la cantidad de empleos no aumente igual de rápido, salir a buscar trabajo se ha convertido en una zona de guerra para muchos profesionales incluidos los profesionales en tecnologías de información.

Cuando alguien entra a una zona de guerra sin las armas adecuadas podemos asumir que esta persona podría acabar muerto de forma prematura. Así que para los candidatos es esencial llevar las mejores armas al campo de batalla, y estas son su currículum (CV) y un portafolio profesional.

Es muy probable que sepamos bien que es un CV, pero ¿qué es un portafolio? Básicamente es un muestrario o recolección de trabajos, proyectos o colaboraciones que has acumulado durante tu vida estudiantil y profesional.

Hoy en día, el portafolio se ha convertido en un factor diferenciador para un candidato a una vacante como programador. Un portafolio bien elaborado se ha convertido en una extensión de tu tarjeta de presentación, de tal forma que la mejor manera de comprobar tu experiencia es a través de esta recolección de trabajos pasados. En numerosas ocasiones, el departamento de recursos humanos ni siquiera lee el CV del candidato si este no incluye un portafolio.

¿POR QUÉ ES IMPORTANTE?

En el mundo de las T.I. es bien sabido que no siempre el que se gradúa de las mejores universidades o el que lleva más años de experiencia es el mejor para el puesto; tal es el caso de un *UX/UI* o un *Front-end* que aunque tengan años de experiencia su trabajo no siempre refleja ese periodo de tiempo, dando un falso mensaje de calidad sobre el candidato, causando un conflicto para el empleador.

Un portafolio bien recolectado y organizado, podría marcar la diferencia crucial entre tú y otro candidato al momento de elegir al indicado para una vacante, ya que este puede decir mucho más de ti de lo que crees, como mostrar tu nivel de experiencia y enfoque laboral.

Tener un portafolio organizado y actualizado puede mostrar la calidad de trabajo de lo que eres capaz y que puede esperar tu empleador de ti, hablando tanto de calidad como de giro o desarrollo. En el caso de un portafolio con Apps móviles ya publicadas habla muy bien de la calidad del trabajo ya que ha sido lo suficientemente bueno como para ser publicidad en alguna tienda online de apps.

¿A PARTIR DE CUÁNDO DEBO CREARLO?

Lo ideal es comenzar desde la universidad, con tus primeras muestras de código, diseños de páginas, aplicaciones desarrolladas, sitios web o la cantidad de proyectos realizados en clase. Entre mayor experiencia, progreso y desarrollo muestre tu portafolio profesional, mayor impacto tendrá al momento de aplicar a alguna vacante. Obviamente la mejor recomendación es colocar solamente lo mejor de lo mejor dentro, para que puedas demostrar tus increíbles talentos y habilidades.

Si conoces diferentes lenguajes de programación, recuerda demostrar por lo menos un gran ejemplo con cada uno.

Debido a la naturaleza de la profesión de T.I., los portafolios son creados de manera digital o en la nube, existen diversas páginas web dedicadas a apoyar a los programadores a desarrollar su portafolio, tal como lo son *GitHub*, *Bitbucket*, *Programming Blog*, *Stack Overflow*, etc.

Tu portafolio será decisivo al momento de una entrevista, es donde se verá reflejado y concretado todo tu esfuerzo y trabajo. Para un empleador es una vista rápida de lo que puede esperar de ti al momento de contratarte, más que tus calificaciones o más que tu carrera profesional, solamente quieren ver qué tan grande es tu experiencia y qué tanto vas a poder aportar a la empresa.

Salir al campo de batalla con tu CV en una mano y tu portafolio profesional en la otra, te abrirá camino entre las tropas enemigas para lograr la añorada victoria. Recuerda que depende de uno mismo perfeccionar estas armas y que una es tan importante como la otra, pero si logras obtener una ventaja al fortalecer tu portafolio, tus posibilidades de obtener el trabajo que buscas aumentarán bastante. Ve hacía la victoria, prepárate y entrena para lograrlo. Adelante soldado.

TIPS PARA CREAR UN PORTAFOLIO:

- Hazlo en línea e incluye el link en tu CV.
- Recuerda incluir tus mejores trabajos, tu portafolio es tan fuerte como tu proyecto más débil.
- Agrega proyectos personales y profesionales.
- Nunca dejes de trabajar en él.
- Demuestra por lo menos un ejemplo de cada lenguaje de programación que sabes.
- En caso de existir, muestra los links de tus aplicaciones y trabajos publicados.

Diego Maury es Director de Vinculación y Reclutamiento de CodersLink. Emprendedor, curioso y dedicado. Vicepresidente de COPARMEX Jóvenes de Mérida. Apasionado a las relaciones públicas y organización de eventos masivos.