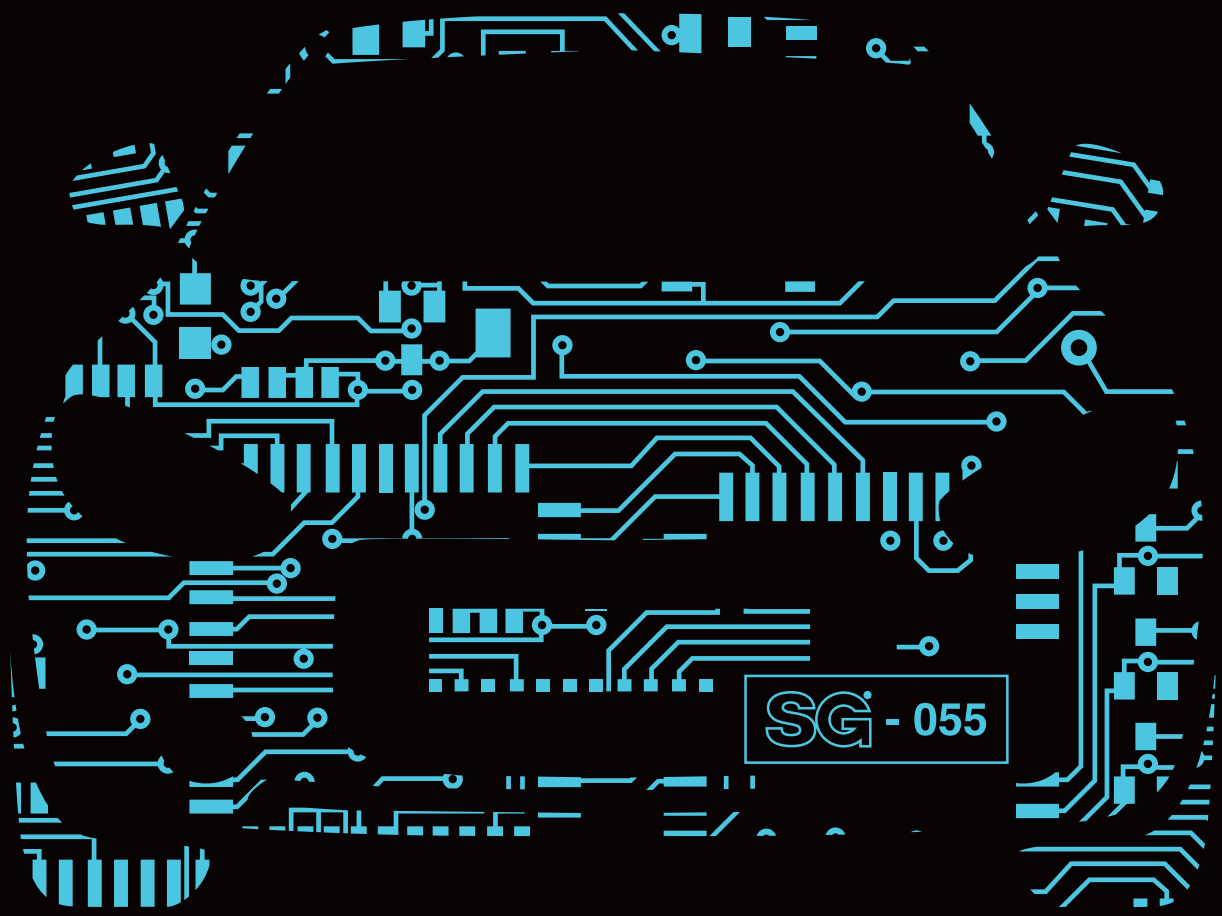
SG®
SOFTWARE GURU
NO.55 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

GUÍA DE ARRANQUE
VISIÓN POR COMPUTADORA
PAG. 08



SOFTWARE EN EL AUTOMÓVIL

LA SIGUIENTE GRAN PLATAFORMA

COMPROMISO
CON LA LIBERTAD Y LA EXCELENCIA

Miriam Ramírez
EXALUMNA DE LA LICENCIATURA EN MATEMÁTICAS APLICADAS

ITAM Posgrados

- MBA-Maestría en Administración
- Maestría en Administración de Riesgos
- Maestría en Ciencia de Datos
- Maestría en Ciencias en Computación
- Maestría en Contaduría
- Maestría en Derechos Humanos y Garantías
- Maestría en Dirección Internacional
- Maestría en Economía Aplicada
- Executive MBA-Maestría en Dirección de Empresas
- Maestría en Finanzas
- Maestría en Mercadotecnia
- Maestría en Políticas Públicas
- Maestría en Tecnologías de Información y Administración
- Maestría en Teoría Económica
- Doctorado en Economía

PREGUNTA POR NUESTRAS SESIONES INFORMATIVAS SEGÚN EL PROGRAMA DE TU INTERÉS.

Av. Camino a Santa Teresa No. 930 Col. Héroes de Padierna, C.P. 10700, Ciudad de México, México.
Tel: (55) 5628 4000 ext. 2612, 01 800 000 ITAM,
posgrados@itam.mx, www.posgrados.itam.mx
Síguenos en: Posgrados ITAM PosgradosITAM

MBA-Maestría en Administración, R.V.O.E. 2001, 01/10/01; Dirección Internacional, R.V.O.E. 944005 13/06/94; Economía Aplicada, clave 2015; Finanzas R.V.O.E. 944195 14/11/94; Tecnologías de Información y Administración, R.V.O.E. 964028 15/04/96; Doctorado en Economía, R.V.O.E. 86820 09/03/01. Reconocimiento de validez oficial mediante Decreto Presidencial publicado en el Diario Oficial de la Federación el 19 de enero de 1963: Administración de Riesgos, clave 2002; Contaduría, clave 2012; Executive MBA-Maestría en Dirección de Empresas, clave 2008; Mercadotecnia, clave 2011; Maestría en Políticas Públicas, clave 2014; Teoría Económica, clave 2001. Maestría en Ciencia de Datos, Maestría en Ciencias en Computación y Maestría en Derechos Humanos y Garantías: clave 2012.

TE INTERESA:

- Realizar mejores estimaciones de tiempo/alcance/costo en proyectos de software.
- Determinar la productividad de tu equipo de trabajo o fábrica de software.
- Realizar un adecuado control de contratos.
- Controlar el "scope creep"
- Tener un mayor control de tus proveedores de software
- Incrementar tu eficiencia en gastos en software

spingere

Dimensionamiento y Estimación Profesional de Software

Seguro te podemos ayudar! Visita www.spingere.com.mx y entérate de los servicios profesionales en medición y estimación de software que ofrecemos.

SPINGERE desde 2014 es el único vendor autorizado por el Common Software Measurement International Consortium (COSMIC) para proporcionar consultoría, capacitación e implementación de ISO/IEC 19761, así como realizar investigación al respecto en México.
Esto se certifica en una carta que ha sido apostillada por la Secretaría de Relaciones Exteriores. La primera carta fue emitida en Montreal, Canadá el 16 de abril de 2014, ratificándose el 22 de febrero de 2017 por el Cónsul de méxico en Montreal, donde está constituido legalmente el Consorcio.

MOBILE DAY
México 2017

Mobile Day es un día de conferencias y sesiones prácticas dirigidas a profesionistas de TI involucrados en el desarrollo de aplicaciones móviles en contextos empresariales.

26 de septiembre 2017

Crowne Plaza Hotel de México, Ciudad de México

sg.com.mx/mobileday

organizado por:

SG® SOFTWARE GURU

DevDay 4 <WOMEN>

ÚNETE A LAS MUJERES QUE DESARROLLAN
SOFTWARE GRANDIOSO

PRÓXIMA EDICIÓN

OCTUBRE, 2017, CIUDAD DE MÉXICO

http://devday4w.com

SG®
SOFTWARE GURU

NO.55 | CONOCIMIENTO EN PRÁCTICA
www.sg.com.mx

EN PORTADA

EL SOFTWARE SE HA APODERADO DEL AUTOMÓVIL

012

Compartimos un panorama sobre las oportunidades, tecnologías y retos que presenta esta revolución.

T HERRAMIENTAS Y TECNOLOGÍAS

R REPORTAJE

I INDUSTRIA

P PRÁCTICAS

C COLUMNAS

V VOCES

O EN CADA NÚMERO

F FUNDAMENTOS

Qué decir cuando todos son expertos

#NosotrosConfesamos que hacer esta edición de SG fue complicado. Frecuentemente, el principal reto que nos encontramos para hacer SG es lograr tomar temas "obscuros" o abstractos, y aterrizarlos. Pero en esta ocasión estamos hablando de automóviles, algo con lo que todos tenemos contacto frecuente y sobre lo que continuamente somos bombardeados con información. Presentar conocimiento valioso sobre un tema en el que todos se consideran expertos, no es sencillo. Nuestra intención ha sido presentar información que ayude a entender qué cambios tecnológicos se están dando en el automóvil y cómo estos generan oportunidades para emprendedores y profesionistas de software. Parte importante del mensaje que buscamos transmitir es que aunque la autonomía es un componente importante de lo que está sucediendo, la conectividad es fundamental y es justamente la que detonará oportunidades para desarrolladores.

Como parte de este número también presentamos un pequeño caso de estudio que refleja algo de lo que se está haciendo para modernizar la infraestructura tecnológica en el Metro de la Ciudad de México. Independientemente de lo que pensemos de la calidad del servicio del Metro, hay que reconocer que modernizar infraestructura tecnológica de hace 40 años mientras se mantiene la operación para atender a millones de usuarios es un reto muy interesante.

El resto de este año en SG estaremos muy activos con congresos (Mobile Day), hackathons (consulta si Hackatour estará cerca de tí) y nuevos retos. Como siempre, será un gusto verte ahí. Y si no se puede, nos seguimos leyendo.

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
Ash Maurya, Gunther Barajas, Gary Silberg, Liliana Toledo, Misael León, Hugo de la Mora, Jorge Heras,
Alessandro Porro, Olivia Salas, Raúl Guerrero, Hugo Hernández, Sergio E. Moreno.

EQUIPO SG
Coordinación de servicio **Yoloxochitl Juárez** | Developer Relations **Luis Sánchez**
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.

Reserva de Derechos al Uso Exclusivo: En trámite. ISSN: 1870-0888. Registro Postal: PP15-5106. Distribuido por Sepomex.

Noticias

MONGODB WORLD 2017 1

La ciudad de Chicago fue sede de MongoDB World 2017, el congreso internacional organizado por MongoDB, la empresa detrás de la base de datos NoSQL con el mismo nombre. En estos días en que existe una gran variedad de almacenes de datos NoSQL similares a mongoDB, es interesante conocer los planes de la empresa para mantener una posición de liderazgo y diferenciarse del resto. Por lo que pudimos ver, hay dos estrategias clave: cloud y analytics. En el caso de cloud, se hizo el lanzamiento de Stitch, un Backend as a Service (BaaS) que contribuye a agilizar el desarrollo de aplicaciones; adicionalmente se dio a conocer que Atlas, el servicio de base de datos en la nube (DBaaS) de MongoDB amplía su disponibilidad a Google Cloud Platform y Azure, además de AWS. En el ámbito de analytics, la empresa comentó que está buscando atender las necesidades de sus clientes que usan mongoDB como base de datos transaccional y desean poder hacer análisis OLAP de forma ágil y económica; con miras a esto, mongoDB estará liberando durante los próximos meses nuevas herramientas y servicios para analizar, extraer y graficar datos.

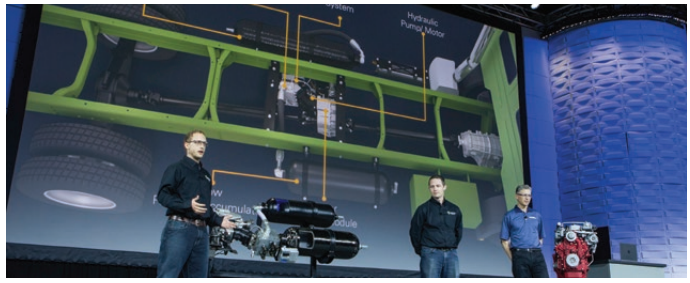
DEV DAY 4 WOMEN GUADALAJARA 2

Más de 200 mujeres se reunieron en la 5ta edición de Dev Day 4 Women, organizado por Software Guru en la Universidad Autónoma de Guadalajara. Como es costumbre, el evento contó con la participación de mujeres en el ámbito de tecnologías de información que compartieron conocimiento, experiencias e inspiración para desarrollarse profesionalmente en este sector.

XAMARIN CHAMPIONSHIP MÉXICO 3

Más de 1,200 desarrolladores participaron en la iniciativa Xamarin Championship organizada por Microsoft México, que después de varias semanas de entrenamiento y retos culminó con un hackathon en las oficinas de Microsoft en Ciudad de México donde los finalistas compitieron para determinar al campeón nacional.

El ganador fue Jonathan Bravo, estudiante del Tecnológico de Estudios Superiores de Ecatepec, quien entre otros premios se llevó el codiciado cinturón con motivos de la Lucha Libre Mexicana.



NIWEEK 2017 4

A finales de mayo se realizó en Austin la edición 2017 de NIWeek, la conferencia global para usuarios de National Instruments. A lo largo de 4 días tuvimos oportunidad de conocer la visión de la empresa y el tipo de productos que sus clientes y aliados están construyendo con su tecnología.

Una de las industrias que tuvo mayor participación fue la automotriz, ya que prácticamente todo el ecosistema de esta industria se encuentra rediseñando sus productos tomando en cuenta el nuevo contexto de vehículos conectados y autónomos. Otro tema de gran importancia fue el de la tecnología inalámbrica 5G, donde National Instruments planea jugar un papel clave.

1 WEBASSEMBLY

Crecientemente, JavaScript ha estado jugando dos roles distintos: uno como lenguaje de programación, y otro como destino de transpilación —es decir, si tienes código en el lenguaje X y quieres ejecutarlo en el navegador, usas un transpilador que "traduce" tu código hacia JavaScript.

Como podrás imaginarte, esto no es una solución óptima. Es por ello que se creó WebAssembly, que consiste en un nuevo formato de código binario que funciona como destino de compilación para otros lenguajes y que puede ser ejecutado por el motor de JavaScript del navegador. La ventaja de tener un formato de código expresamente diseñado con este propósito es que permite optimizarlo extensivamente para que su ejecución sea muy rápida. Así que podrás tener código JavaScript que pueda interactuar de forma transparente con componentes programados en otros lenguajes (por ejemplo C++ o Rust) que tengan muy alto desempeño y aprovechen capacidades de hardware a bajo nivel. Actualmente WebAssembly es soportado en Chrome y Firefox, y se espera que sea soportado en Safari y Edge antes de que termine el año.

https://webassembly.org

2 ARKIT

En el WWDC 2017, Apple dio a conocer ARKit, un framework para construir apps de realidad aumentada (AR) para iOS. La realidad aumentada no es nada nuevo, y es un hecho que Apple viene tarde a la fiesta. Pero en realidad la fiesta todavía no ha comenzado, y todo parece indicar que lo que realmente iniciará la fiesta de AR será ARKit, que es parte de iOS 11 y llegará este otoño. Mientras que empresas como Google y Microsoft siguen tratando de crear visores sofisticados y/o buscando "killer applications" para AR, Apple está tomando un camino distinto: habilitar a su extensa base de desarrolladores con un framework que les permita fácilmente construir aplicaciones de AR para cientos de millones de dispositivos que ya están en el mercado. ARKit brinda soporte para capacidades tales como detección de movimiento, detección de planos horizontales, estimación de luz y tamaño.

https://developer.apple.com/arkit

3 LABVIEW NXG

LabView es un ambiente de programación visual hecha por National Instruments para construir sistemas electrónicos. Tiene cerca de 20 años en el mercado y a lo largo de este tiempo ha sido una herramienta de nicho, utilizada principalmente por ingenieros eléctricos para captar y registrar mediciones de instrumentos con alta precisión.

La novedad es que National Instruments está lanzando una nueva generación de esta herramienta, llamada LabView NXG. El objetivo es hacerla mucho más amigable de usar, y con ello poder atraer a personas que no tengan una educación formal en diseño de sistemas electrónicos.

La estrategia tiene bastante sentido, considerando que la necesidad por este tipo de herramientas aumentará conforme las empresas se involucren en proyectos de IoT e internet industrial.

https://ni.com/labviewnxg

4 NET CORE 2 Y .NET STANDARD 2

Microsoft liberó oficialmente la versión 2 del .NET Core SDK, y de manera conjunta publicó la versión 2.0 de .NET Standard. Si no te queda claro qué significa esto, te explicamos: En 2016, con el afán de permitir que los desarrolladores pudieran usar el .NET Framework para construir aplicaciones para distintas plataformas, Microsoft decidió soportar distintas implementaciones de .NET, entre ellas el .NET Framework (Windows), .NET Core (cross-platform y open source), y Xamarin (enfocado a móvil). Para facilitar la portabilidad, se definió una capa abstracta llamada .NET Standard que básicamente especifica las APIs que deben ser soportadas a través de las distintas implementaciones de .NET. En un inicio, .NET Standard definía menos de 10 mil APIs (extraídos del .NET Framework en Windows) que debían ser soportados, y esto ha ido creciendo hasta que la versión 2.0 del .NET Standard especifica alrededor de 32 mil APIs. Después de ese breve contexto, sigamos con los detalles de .NET Core 2. Esta versión agrega soporte de ejecución en nuevas versiones de sistemas operativos tales como Ubuntu 17.04, macOS High Sierra, Fedora 26 y Linux Mint 18. También agrega Visual Basic como lenguaje fuente (limitado a componentes no visuales). Así que si por alguna extraña razón algunas vez has querido programar una biblioteca en Visual Basic para Linux, ya puedes hacerlo.

https://www.microsoft.com/net/core

SGNEXT

El 22 de junio Software Guru organizó la edición 2017 de SG Next. El objetivo de este congreso es difundir la nueva generación de estrategias, prácticas y plataformas para desarrollo de software.

En esta ocasión, la temática giró alrededor de los temas de transformación digital, entrega continua y arquitecturas de aplicaciones nativas a la nube. Como comentó Pedro Galván durante la sesión de cierre: la entrega continua y la arquitecturas nativas a la nube son habilitadores de la transformación digital: la primera agiliza la forma de trabajar y la segunda agiliza la infraestructura de cómputo. Pretender conseguir una transformación digital sin estos dos pilares es inverosímil.

Estos son algunos de los hallazgos que resaltaron durante SG Next:

- La entrega continua no es una leyenda urbana. Hay empresas locales que la practican, así como guías para facilitar su adopción y herramientas para sustentarla.
- Los contenedores brindan grandes ventajas para agilizar el despliegue de aplicaciones en la nube. Todavía hay aspectos por pulir, pero los beneficios sin duda superan las complicaciones.
- El stack tecnológico para construir aplicaciones modernas está sufriendo grandes cambios. Las arquitecturas centradas en application servers para una sola plataforma y herramientas de un solo proveedor están quedando en el pasado. Los arquitectos y desarrolladores interesados en construir aplicaciones de nueva generación deben considerar una variedad de plataformas, lenguajes y herramientas.
- Más allá del "buzz word", la transformación digital sí está impactando a las organizaciones en nuestra región y moldeando la forma en que trabajan. Incluso en un sector tan regulado como el de la banca ya están haciendo cosas como desplegar aplicaciones en nube pública y ofrecer APIs para que terceros puedan ofrecer servicios de valor agregado a sus clientes. Otro impacto de la transformación digital en corporativos es la tendencia hacia el "insourcing" de actividades de innovación tecnológica, lo que aumenta la demanda de talento tecnológico en corporativos.

Los participantes de SG Next también aprovecharon la oportunidad para participar en el taller impartido por PayPal y acreditarse como PayPal certified developer.

Otro aspecto positivo de SG Next fue ver a empresas locales entre los patrocinadores, como Tacit Knowledge que nos hizo ver como desarrolladores locales participan en proyectos globales para clientes del más alto nivel. Otro patrocinador local fue Tinkerware, una plataforma de desarrollo bajo una filosofía DevOps.

Visita https://sg.com.mx/sgnext para más información, presentaciones y fotografías.

Visión Computacional

¿qué es y por dónde comenzar?

—

Por Ana Loyo

La visión computacional trata de emular en las computadoras la capacidad que tienen nuestros ojos. Es decir, trata de interpretar las imágenes recibidas por dispositivos como cámaras y reconocer los objetos, ambiente y posición en el espacio.

Lograr tal interpretación al mismo nivel que el ser humano es un problema complejo. Sin embargo, ha habido avances considerables a lo largo de los años.

Del procesamiento de imágenes a la visión computacional

Durante la década de los 1970s se dieron los primeros avances significativos en el campo de análisis de imágenes por computadora. Resaltan los siguientes trabajos::

• Azriel Rosenfeld es uno de los pioneros en este campo, su trabajo se enfocó principalmente en el desarrollo de técnicas empíricas basadas en criterios matemáticos para representar objetos físicos por medio de un conjunto de discreto de datos.

• Otro enfoque fue el propuesto por Waltz y Mackworth en 1975, reduciendo el alcance del problema a un mundo de bloques blancos mate iluminados sobre fondo negro. Los bloques podían tener cualquier forma, siempre que todas sus superficies fueran planas y todos sus bordes rectos. Aunque este enfoque fue útil para experimentar, dadas sus restricciones no se pudo llevar a mundos complejos.

• Por su parte, el trabajo de Berthold Horn se enfocó en establecer modelos de cálculo que expresan la formación de imágenes a través de ecuaciones diferenciales que relacionan los valores de intensidad de la imagen con la geometría, la reflectancia de la superficie y el punto de vista del observador. La idea era que debía existir un nivel adicional de comprensión en el que el carácter de las tareas de procesamiento de la información llevadas a cabo durante la percepción se analicen y comprendan de modo independiente a los mecanismos y escrituras particulares que los implementa en nuestros cerebros.

Ahora bien, debemos considerar que la visión computacional y el procesamiento de imágenes son cuestiones diferentes. El procesamiento de imagen trata sobre cómo mejorar una imagen para su interpretación por una persona mientras que la visión computacional trata de interpretar las imágenes por la computadora.

La visión computacional se puede dividir en 3 grandes etapas:

1. Procesamiento de nivel bajo. Se extraen propiedades como orillas, gradiente, profundidad, textura, color, etcétera.

2. Procesamiento de nivel intermedio. Agrupa elementos de nivel bajo para obtener contornos y regiones con el propósito de segmentar.

2. Procesamiento de alto nivel. Consiste en la interpretación utilizando modelos y/o conocimiento del dominio del problema.

EXPERIMENTANDO CON OPENCV

Actualmente, una de las tecnologías más usadas para visión computacional es OpenCV. Esta es una librería publicada bajo licencia BSD que está disponible para una gran variedad de sistemas operativos y tiene más de 500 funciones que abarcan una amplia gama de áreas en el proceso de visión y reconocimiento de objetos, calibración de cámaras, visión estérea y visión robótica. Se puede utilizar con los lenguajes de programación Python, C++ y Java, entre otros.

A continuación explicaré como instalar un ambiente de desarrollo para visión computacional con OpenCV y Python.

En este caso, Python es una excelente opción de lenguaje de programación por ser un lenguaje de código abierto, multiplataforma, con una sintaxis intuitiva, una curva de aprendizaje corta y un amplio ecosistema de bibliotecas. En caso de que no tengas Python instalado, puedes obtenerlo en https://www.python.org/downloads. Procura usar una versión 2.7.x de Python (al momento de este artículo la versión más reciente de esta rama es la 2.7.13) ya que si usas una versión 3.6.x tendrás problemas de compatibilidad cuando trates de usarlo con las otras herramientas que recomiendo en este artículo.

Si quieres utilizar un editor de código, personalmente me gusta LiClipse, pero en gustos se rompen géneros así que utiliza tu preferido.

Ana Luz Loyo Páez es Ingeniero en Sistemas Computacionales (UNIVO) con Maestría en Comercio Electrónico (ITESM). Actualmente es Coordinadora de contenidos en Software Guru, y catedrático en la Universidad Veracruzana. Además es desarrolladora, instructora y asesora de TI y Negocios Electrónicos freelance. TW @ana_loyo

El siguiente paso a seguir es la instalación de NumPy (en su última versión), un paquete fundamental de Python que contiene sofisticadas funciones y extiende el soporte para vectores, matrices y funciones matemáticas de alto nivel. La forma más sencilla de instalar numpy es por medio de la utilería pip. Si trabajas en Windows, como yo, recuerda agregar las variables del sistema correspondientes a la ruta donde tienes tu archivo Python.exe y a la carpeta Scripts dentro de Python. Hecho lo anterior solamente tendrás que ejecutar desde la línea de comandos:

```
pip install numpy
```

Si todo sale bien obtendrás un mensaje confirmando que numpy se instaló exitosamente.

Ahora vamos a instalar OpenCV. Si usas Windows, para instalar OpenCV debes descargarlo de http://opencv.org/releases.html donde encontrarás un ejecutable de Windows. Este archivo es un paquete que descomprime OpenCV en la carpeta que elijas. Una vez hecho esto, entra a la carpeta que se creó y localiza el archivo cv2.pyd bajo la ruta build/python/2.7/x64 (el x64 ruta es suponiendo que tu sistema operativo sea de 64 bits, que es lo más probable). Ahora copia el archivo cv2.pyd a la carpeta lib/site-packages de tu instalación de Python.

Si usas Linux, puedes instalar OpenCV del repositorio de paquetes de tu sistema operativo. En el caso de OSX, lo más sencillo es instalar OpenCV por medio de Homebrew (tienes que habilitar el repositorio homebrew/science).

Para verificar que nuestra instalación de OpenCV funciona correctamente hagamos un pequeño programa. Crea en tu computadora una carpeta donde pongas un archivo de imagen, y crea un programa python con el código mostrado en el listado 1.

```
import numpy
import cv2
imagen=cv2.imread("carro.jpg")
cv2.imshow("ventana", imagen)
cv2.waitKey()
```

Listado 1. Prueba de opencv

Reemplaza "carro.jpg" por el nombre de archivo de tu imagen (recuerda incluir la ruta en caso que tu imagen esté en una imagen distinta a tu programa).

Si todo está bien, al ejecutar tu programa se desplegará una ventana con tu imagen, tal como se muestra en la figura 1.

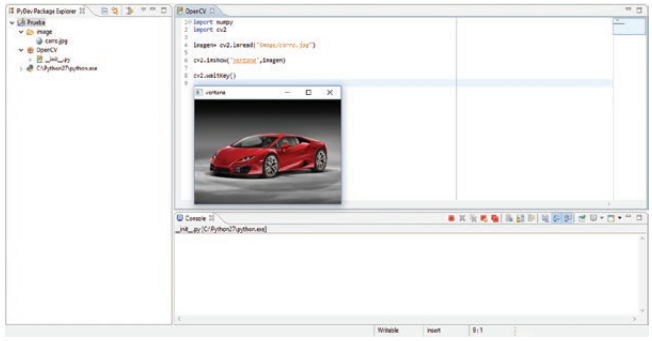


Figura 1. Ejecución de programa con OpenCV

Teniendo todo esto instalado puedes comenzar a experimentar con el reconocimiento de imágenes. Obviamente necesitarás más que esto pero el objetivo del artículo es que puedas realizar una configuración mínima necesaria para comenzar, suerte con tus proyectos.

Te comparto algunas recomendaciones para continuar en tu exploración:

- Puedes tener problemas al querer usar PIP, verifica que estén bien las variables de entorno en tu sistema operativo.

- Cuando descomprimas OpenCV asegúrate de estar copiando el archivo correcto en la ruta correcta porque si no lo haces bien, no reconocerá el import de cv2.

- En http://opencv.org/links.html puedes encontrar varios tutoriales e información de referencia sobre OpenCV.

- Cuando trabajas con dispositivos como la RaspBerry tu código será un poco distinto. No pretendas probar sin tener el hardware, inicia tu código y pruebas hasta que lo tengas y te evitarás re-trabajo.

Es Momento de Despedir al Plan de Negocio

Por Ash Maurya

¿Alguna vez has hecho un plan de negocio (*business plan*)? ¿Lo disfrutaste? He hecho esta pregunta a miles de emprendedores y el resultado que he obtenido es que solo el 30% de ellos ha hecho business plan, y menos del 2% lo disfrutó.

Por otro lado, he preguntado a inversionistas si acostumbran leer el plan de negocio completo, y menos del 2% aceptan hacerlo, prefiriendo algo más corto como un resumen ejecutivo de una página, una presentación de 10 láminas o un pitch de elevador.

¿Por qué entonces seguimos forzando a las personas a dedicar semanas escribiendo un documento de 40 páginas que nadie lee?

En este artículo explicaré por qué el plan de negocio es una reliquia, y cómo se puede hacer de forma efectiva en dado caso que estés obligado a hacer uno.

EL ROL DEL PLAN DE NEGOCIO

El plan de negocio ha sido usado históricamente como instrumento para levantar inversión. Punto. Aunque algunos puedan mencionar metas adicionales, todavía no conozco a ningún equipo que mantenga su plan de negocio actualizado después de haber levantado inversión.

Arrancar un negocio acostumbraba requerir de bastante capital. Así que la única opción para los emprendedores era crear un plan de negocio que sirva para presentar un caso convincente de la inversión. Pero el mundo ha cambiado.

Los servidores, licencias de software y espacio de oficina que acostumbraban costar decenas de miles de dólares hoy son casi gratuitos. Con costos bajos y aplicando metodologías como Lean Startup, las empresas ya no necesitan cantidades tan grandes de capital. Podemos comenzar a eliminar los riesgos de nuestras ideas a partir del primer día.

La mejor evidencia de un negocio que funcionará no es pichar una historia ficticia de lo que podría suceder, sino una historia real de qué sucedió cuando tu idea se encontró con clientes — esto es lo que se conoce como tracción.

Figura 1. Gráfica de tracción

La tracción es el ritmo al que se captura valor monetizable de los clientes.

Si muestras a cualquier inversionista una gráfica de tracción en las primeras etapas de un palo de hockey (ver figura 1), no necesitas más para obtener su interés; inmediatamente querrán entender tu idea.

En pocas palabras, los inversionistas y emprendedores deben enfocarse en tracción obtenida, no en un plan para obtenerla.

PLANES DE NEGOCIO VS. PLANEACIÓN DE NEGOCIO

Desgraciadamente muchos emprendedores se han ido al otro extremo y optan por una estrategia de no planear y hacer todo al vuelo. No creo que esto sea bueno tampoco.

Intentar construir un negocio sin planeación es como construir una casa sin planos. De hecho, una casa es menos compleja y tiene mucho menor incertidumbre que un negocio.

Es por ello que hago la distinción entre plan de negocio y planeación de negocio, ya que recomiendo la segunda fuertemente.

El principal problema que tengo con el plan de negocio no es la intención sino el formato. Requerir un entregable que no es leído es un desperdicio. Más allá de esto, al innovar enfrentamos una gran cantidad de elementos desconocidos; y estos no se pueden documentar a priori.

Necesitamos una nueva generación de herramientas ligeras para planeación del negocio diseñadas para trabajar de forma ágil y que resuelvan las necesidades tanto de emprendedores como inversionistas.

LEAN CANVAS

El Lean Canvas es una herramienta que he diseñado para cumplir los requerimientos descritos. Es un modelo de negocio capturado en una página, basado en el Business Model Canvas de Alex Osterwalder. Al igual que este, el Lean Canvas usa un conjunto de bloques que ayudan a dar un panorama de la historia de tu modelo de negocio (ver figura 2).

Aunque el Lean Canvas comparte varias secciones con el Business Model Canvas, el Lean Canvas tiene un enfoque distinto. A continuación describo las principales diferencias:

Enfocado a emprendedores. El Lean Canvas está diseñado en base a un paradigma de Cliente-Problema-Solución. Los puristas de los modelos de negocios pueden criticarlo por estar demasiado orientado a producto, pero justamente esa es su ventaja. Al disfrazar una herramienta de planeación de negocio como si fuera una herramienta de captura de ideas aumentamos el involucramiento.

Enfrenta la parcialidad del innovador (*innovator's bias*). A pesar de que el lean canvas tiene una caja para describir la solución, intencionalmente es una de las cajas más pequeñas. No porque construir una gran solución no sea importante, sino

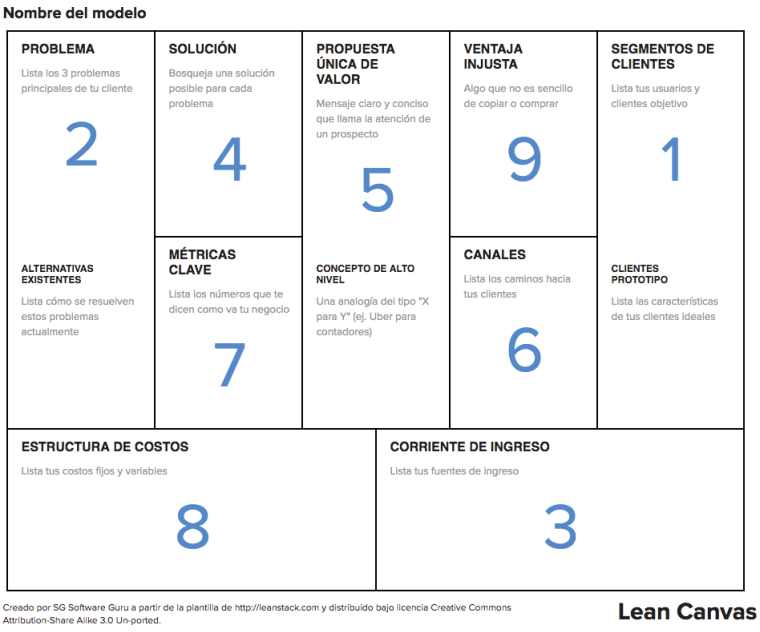


Figura 2. Lean Canvas

porque construir una solución es tan solo una parte del verdadero producto de un emprendimiento. El verdadero producto es el modelo de negocio, no la solución. Conforme los emprendedores llenan el canvas se dan cuenta que esa idea clara que pensaban tener clara en la cabeza en realidad no está tan clara. Recuerda que la principal razón de falla de los startups no es que no logren construir el producto que ellos querían, sino que no construyen el producto que los clientes en realidad querían. Entender profundamente a los clientes y sus problemas es el prerrequisito para construir lo que los clientes quieren.

Sencillo de entender. Otra meta de diseño del lean canvas fue mantener las etiquetas de cada sección lo más simples e intuitivas como fuera posible. Sabía que pedir a los emprendedores que usaran una herramienta que requiriera un manual para saber llenarla, no iba a funcionar.

Facilita comparación. En las primeras etapas de un producto todo parece posible y es difícil capturar todas las posibilidades en un plan de negocio. La solución es crear varios modelos de negocio y compararlos para evaluar su desempeño. En mi libro, Running Lean [1] describo este proceso a detalle.

Facilita conversación. Cuando pones un mapa como este en frente de una persona, no pueden evitar leerlo y tampoco pueden evitar tener una opinión. El Lean Canvas facilita obtener retroalimentación rápida de colegas, asesores e inversionistas, guiándolos a tu mismo patrón de pensamiento.

Permite iterar. La naturaleza ligera del Lean Canvas implica que puedes mantenerlo actualizado sin gran esfuerzo de manera que pueda ser un artefacto viviente para capturar la evolución de tu modelo de negocio. Mientras que el plan de negocio tradicional promueve una estrategia de "planea y ejecuta", el Lean Canvas promueve una estrategia de "descubre y ejecuta".

OBJECIONES FRECUENTES

El cambio siempre enfrenta resistencia, y la adopción del Lean Canvas no es distinta. Estas son algunas de las objeciones más comunes que he recibido.

¿Lo aceptan inversionistas? A pesar de ser una herramienta joven, siendo creada apenas en el año 2009, el Lean Canvas ya es usado en cientos de startups, aceleradoras y universidades. Los inversionistas quieren hacer dinero, no leer planes de negocio. Lo que más les interesa es la tracción, y ésta no se obtiene con planes de negocio, se obtiene validando un modelo de negocio.

Estoy obligado a hacer un plan de negocio. Si a pesar de todo requieres escribir un plan de negocio formal, te recomiendo que antes de hacerlo crees varios canvas y los compartas con inversionistas candidatos, ejecutando algunos experimentos durante un par de semanas para validar tus supuestos de mayor riesgo. De esta manera, cuando hagas el plan de negocio podrás basarte en hechos y no en ficción, y por lo tanto tendrás mayor posibilidad de obtener inversión.

Necesito más detalle . Claro que no es posible capturar todos los aspectos de un negocio en una página, y precisamente esa es la meta. El Lean Canvas es principalmente una herramienta de conversación.

¿Puedo enviar mi Lean Canvas a otros? No recomiendo simplemente enviar el canvas sin contexto. Si no te es posible explicarlo en vivo, recomiendo que incluyas un video de 5 minutos.

No considera planeación financiera. Es cierto, un modelo de negocio no es suficiente para determinar si esta es una buena opción de inversión. En mi libro Scaling Lean [2] explico como usar la técnica de Fermi para generar estimaciones rápidas para evaluar un modelo de negocio.

¿Estás listo para deshacerte de tu plan de negocio?

Este artículo fue traducido y editado por Software Guru con el permiso del autor a partir de la versión original disponible en https://blog.leanstack.com/its-time-to-fire-the-business-plan-for-good-df165fcc78f6

Referencias

[1] A. Maurya. Running Lean. O'Reilly Media, 2nd edition, 2012. https://www.goodreads.com/book/show/13078769-running-lean

[2] A. Maurya. Scaling Lean. Portfolio, 2016. https://www.goodreads.com/book/show/26895164-scaling-lean

Ash Maurya es asesor de startups y autor de los libros "Running Lean" y "Scaling Lean", así como creador de la herramienta de modelado de negocio "Lean Canvas".


/Autonomous
/Sensing
/Communication
/Battery
/Navigation
/Mirrorless
/Ecology
100m
48
mph
/Autonomous
/Sensing
/Communication
/Battery
/Navigation
/Mirrorless
/Ecology

EL SOFTWARE SE HA APODERADO DEL AUTOMÓVIL

A TRAVÉS DE MÁS DE 100 AÑOS DE HISTORIA, LOS AUTOMÓVILES HAN IDO EVOLUCIONANDO DE FORMA GRADUAL Y CONSTANTE. TÍPICAMENTE EN CADA DÉCADA LA INDUSTRIA HA ADOPTADO UNA O DOS INNOVACIONES NOTABLES, DESDE LA DIRECCIÓN HIDRÁULICA A LAS BOLSAS DE AIRE Y LOS FRENOS ANTI-BLOQUEO.

PERO CONFORME EL SOFTWARE SE ESTÁ APODERANDO DE LOS AUTOMÓVILES, EL RITMO DE INNOVACIÓN SE ESTÁ ACELERANDO DE FORMA DRAMÁTICA. HACE APENAS UNOS AÑOS, PENSAR EN AUTOMÓVILES QUE SE CONDUCIERAN POR SÍ SOLOS SONABA A ALGO DE CIENCIA FICCIÓN; HOY EN DÍA ES UNA REALIDAD.

HABIENDO PASADO DE LAS COMPUTADORAS DE ESCRITORIO A LOS TELÉFONOS MÓVILES, LA SIGUIENTE GRAN PLATAFORMA DE EJECUCIÓN DE SOFTWARE ES EL AUTOMÓVIL. ES UN CLIENTE CON MILES DE MILLONES DE INSTALACIONES EN TODO EL MUNDO, CON EL QUE SUS USUARIOS CONVIVEN VARIAS HORAS AL DÍA. LAS OPORTUNIDADES PARA DESARROLLADORES DE SOFTWARE SON INFINITAS.

EN LAS SIGUIENTES PÁGINAS COMPARTIMOS UN PANORAMA SOBRE LAS OPORTUNIDADES, TECNOLOGÍAS Y RETOS QUE PRESENTA ESTA REVOLUCIÓN.

EL CAMINO DE LA EXPERIENCIA HACIA LA INNOVACIÓN AUTOMOTRIZ

Por Gunther Barajas

A medida que entramos en la industria 4.0, una nueva era de sistemas ciber-físicos, Internet de las cosas y procesos conectados ¿cuál es el futuro que vamos a crear para la industria automovilística durante los próximos 10 o 15 años?

Conforme visualizamos la movilidad del futuro ¿Podremos crear y producir nuestro propio coche personalizado? ¿Podemos ir más allá de un vehículo de cero emisiones a un modo de transporte que apoye activamente la salud y el equilibrio medioambientales? ¿Qué pasa con los vehículos que son tan inteligentes que literalmente no pueden chocar e incluso pueden contribuir a la seguridad personal y urbana?

Una cosa es cierta, con el inmenso impacto social y emocional de la movilidad y con la creciente tasa de cambio tecnológico, la industria automotriz evoluciona a un ritmo nunca antes visto.

USO DE DATOS PARA CENTRARSE EN LA EXPERIENCIA DEL CLIENTE

Con los vehículos autónomos previstos para estar disponibles al público antes del año 2020, el foco de los fabricantes de autos se centrará en la creación de experiencias de movilidad en torno a cómo los consumidores utilizan e interactúan con sus automóviles. Los pasajeros pueden ahora mirar una película, cenar o trabajar con otras personas, entre otras muchas opciones, esto requiere un entendimiento integral de las capacidades y necesidades de los consumidores más allá de la conducción, inspirando a los proveedores a forjar nuevas alianzas tecnológicas y de negocios mientras desarrollan nuevos estándares de industria y tecnología.

Un típico "nuevo modelo" de auto viene con 100 millones de líneas de código y de 50 a 80 unidades de control electrónico que responden a más de 30 mil requerimientos funcionales. La electrónica comprende entre el 20 y el 40 por ciento del costo actual del desarrollo de vehículos, con un número que se prevé que crezca hasta el 50% para 2030, modificando rápidamente la complejidad del vehículo e influyendo en el proceso de desarrollo. Se espera que el 80% de las innovaciones de vehículos en el futuro sean en sistemas integrados, gran parte de ellos enfocados en seguridad activa o pasiva, entretenimiento y rendimiento.

Será cada vez más crucial validar los subsistemas y el desempeño de un vehículo en una etapa temprana del ciclo de desarrollo del producto. Las nuevas metodologías de la ingeniería de sistemas, que enlazan dominios de procesos dispares , serán necesarias para hacer pruebas dinámicas del vehículo previo a su producción, para garantizar que se cumplan estándares de desempeño y seguridad.

Para reducir el riesgo de innovación en este nuevo mundo de la experiencia de movilidad, los clientes estarán fuertemente involucrados en el proceso de creación a través de experiencias virtuales realistas en 3D. La capacidad de ejecutar pruebas dinámicas e inmersivas en un entorno virtual durante la etapa de diseño, ofrecerá nuevas formas de entender y validar los deseos del consumidor en torno a la movilidad. Además, el aprovechamiento de cantidades masivas de información no estructurada (big data) catalizará la creación de experiencias personalizadas de movilidad. Los sensores dentro de los vehículos proporcionarán nuevos niveles de retroalimentación en tiempo real para que técnicos y empresarios puedan interpretar el comportamiento del cliente. Los subsistemas de los automóviles son cada vez más inteligentes y dependientes de la conexión con los datos de otros sistemas. De los faros a los sistemas de escape, cuando antes un "ajuste y acabado" eran la única integración necesaria, entender ahora cómo software, hardware y electrónica trabajan juntos es crucial. Toda esta información, mezclada con datos externos de las redes sociales en línea, se utilizará para mejorar continuamente el rendimiento de la conducción y proporcionar así la experiencia deseada.

Para comprender el impacto completo de la innovación, los fabricantes irán más allá de las características físicas y tecnológicas del vehículo hasta simular aspectos funcionales y de comportamiento. Por si fuera poco, el automóvil, el cual es ahora visto como un sistema inteligente, será simulado dentro de sistemas tan amplios como el tráfico, la energía o incluso las redes comunitarias y domésticas.

La propuesta Link & Go de AKKA Technologies es un ejemplo de cómo los innovadores en la cadena de sumini tro automotriz pueden combinar conocimientos avanzados de ingeniería con innovación social para ofrecer nuevas perspectivas para la movilidad autodirigida en las carreteras urbanas [1].

SER ÁGIL ADMINISTRANDO LA CRECIENTE COMPLEJIDAD

Esto puede sonar contradictorio, pero en la era de Automotive 4.0 los fabricantes de automóviles serán capaces de dominar la inmensa complejidad de los sistemas detrás de los procesos de desarrollo y fabricación, simplificando al mismo tiempo la experiencia del usuario en toda la empresa. El primer método para lograr esto es la integración dentro de los actuales entornos de las tecnologías de la información, en los que muchas aplicaciones aisladas han crecido a lo largo de décadas, y donde las oportunidades son numerosas, entre ellas:

- integración de diseño e ingeniería para preservar la idea original de una creación,
- integración de ingeniería y simulación para compensaciones costo-rendimiento y cumplimiento de productos,
- integración mecánica, de software y desarrollo punto a punto para obtener una arquitectura de desarrollo optimizada,
- integración de pruebas y validación con mercadotecnia y ventas para conocer los requerimientos.

Cada parte de la organización se beneficiará de una intensa integración, conectando personas, información y procesos a través de un entorno único, intuitivo y colaborativo.

La siguiente clave para la gestión de la complejidad es la modularización, desde los requerimientos sobre el diseño de sistemas, la ingeniería detallada y las pruebas hasta la fabricación. La integración comprenderá el mundo virtual y físico: pruebas integradas de hardware y software, planificación y operación integrada en la fabricación, etcétera, mediante la cual objetos inteligentes y modulares establecerán el comportamiento de sistemas complejos.

La tecnología base y los conceptos están disponibles hoy en día. El siguiente paso —la implementación de sistemas inteligentes y autoadaptables en la fabricación y la logística— sólo estará limitado por el alcance de las estrategias de integración y modularización. Si las impresoras 3D son capaces de manejar una parte de la producción de piezas de repuesto ¿cuál será el impacto en el diseño del producto, su empaquetado y los aspectos comerciales?

Últimamente, la gestión de la complejidad tiene otra dimensión. Al igual que los usuarios de transporte tendrán la posibilidad de personalizar su experiencia, los creadores de soluciones de movilidad también tendrán que usar un banco de trabajo más flexible, intuitivo y personalizable para las necesidades individuales. Fiat Chrysler Automobile, ha comenzado a implementar la solución de experiencia por industria "Drive Emotion" en sus estudios de diseño global. Esto permite a sus expertos colaborar en tiempo real en las experiencias y elementos holísticos del usuario como el estilo artístico o calidad de la superficie del vehículo.

ASUMIR UNA MAYOR RESPONSABILIDAD

Las empresas están acostumbradas a actuar como ciudadanos corporativos responsables, pero estas obligaciones crecerán con la capacidad de las tecnologías del mañana. Los aspectos ambientales han sido durante mucho tiempo parte de las preocupaciones de la industria automotriz, los conductores de hoy en día esperan la entrega de vehículos eco-amigables y eficientes en cuanto al combustible sin sacrificar el rendimiento. Con tanto software en los vehículos, los principales competidores de la industria del automóvil han trabajado juntos para desarrollar el estándar Automotive Open System Architecture (AUTOSAR). Estos fabricantes tienen ahora la tarea de cumplir con el nuevo estándar de seguridad funcional eléctrica/electrónica ISO26262, que requiere la documentación del proceso completo, análisis y verificación de todos los sistemas integrados de los productos que fabrican.

Con diferentes lineamientos que cumplir en todo el mundo, los proveedores de esta industria utilizarán una plataforma que gestione la complejidad de las normas y permita la simulación de modelos y comportamientos, que muestren el balance energético y el potencial impacto del diseño de sus vehículos en el medio ambiente.

El aumento de la seguridad de los pasajeros y los peatones seguirá siendo un elemento que empuje la innovación futura y pronto veremos como este enfoque crece rápidamente. Esta es la razón por la cual los enfoques Target-Zero-Defect son (y seguirán siendo) tan importantes en la agenda de esta industria. Estos dirigen la operación sin fallos y sin incidentes alrededor de los diversos procesos y datos corporativos.

Y ¿qué pasa con la retención de clientes y el uso de sus datos? Con la gama de sensores en el vehículo e información externa disponible, se puede esperar que la industria contribuya realmente al bienestar social y la salud personal. Estos aspectos podrían comenzar por el registro del estado de un pasajero y - según la situación – proveer una sensación calmante dentro del vehículo, iniciar un masaje que evite el dolor de espalda o coágulos de sangre, o conectarse inmediatamente a un médico especialista en caso de emergencia.

Generar y procesar este tipo de datos personales, al menos en los países donde la información es sensible, requerirá que los pasajeros sientan una gran familiaridad con las compañías. Las promesas de las marcas y de los fabricantes, así como la dimensión de confianza, tendrán que reflejarse en modelos de negocios que llevarán a variables como "el placer de conducir" o "ventaja tecnológica" a un nuevo nivel.

VIENDO HACIA ADELANTE

No somos clarividentes para predecir el futuro de la industria automotriz, sin embargo, estamos seguros de que la integración y el impacto de la innovación social serán elementos fundamentales para controlar la creciente complejidad de la innovación en la industria de la movilidad, lo que contribuirá a aumentar el tiempo de comercialización y su productividad.

Una mayor conectividad dentro de los vehículos con sus entornos, así como la demanda continua de movilidad individualizada, abrirán nuevas oportunidades para que los innovadores automovilísticos creen experiencias diferenciadas para sus clientes. Los ambientes virtuales de trabajo que son realistas, colaborativos, "inteligentes" e intuitivos serán los laboratorios de innovación para crear estas experiencias. Las tecnologías de hoy día, aunque en constante evolución, ya construyen un puente para proporcionar el nivel de colaboración necesario para operar mañana el Automotive 4.0.

Referencias

[1] http://blogs.3ds.com/perspectives/autonomous-cars-in-the-age-of-experience/

Gunther Barajas (@GBV_001) es Vice Presidente de Dassault Systèmes para México. https://www.3ds.com/es

TECNOLOGÍAS DETRÁS DEL AUTOMÓVIL CONECTADO

Por Pedro Galván

Está muy bien que platiquemos sobre las posibilidades del automóvil conectado, los escenarios de uso que abre y el impacto que tendrá en distintas industrias. Pero como desarrolladores de software, una de las preguntas que inevitablemente nos hacemos es: ¿cómo podemos desarrollar aplicaciones para este segmento?, ¿qué tecnologías se utilizan y dónde puedo aprender al respecto?, ¿qué oportunidades están abiertas para desarrolladores externos?

Para contestar dichas preguntas, echemos un vistazo a las principales áreas en las que se puede construir soluciones para el automóvil conectado, así como las empresas que están detrás.

CAPAS APLICATIVAS

Los automóviles modernos utilizan software para resolver distintos aspectos. Aunque no hay un estándar de los distintos grupos aplicativos, en general podemos reconocer los de telemática, infotainment, instrumentación, ADAS y conducción autónoma. A continuación explico cada uno.

INFOTAINMENT

Como su nombre lo indica, esta capa ofrece servicios de información y entretenimiento, apoyándose en las capacidades de conectividad, audio y despliegue de información en pantallas. Esto se puede utilizar tanto para escenarios de productividad (ej. hacer llamadas telefónicas, escuchar mensajes de correo) como de entretenimiento (servicios de música, juegos o video para pasajeros).

TELEMÁTICA

La capa de telemática brinda información sobre la actividad, historial y ubicación de un automóvil. El uso más básico de estos servicios es para construir aplicaciones de navegación por GPS, asistencia en el camino o localización de autos robados. Sin embargo, al contar con la información del estado de un automóvil, tanto histórico como en tiempo real, y combinarlas con servicios de analítica predictiva las posibilidades aplicativas son infinitas. Por ejemplo:

- Ofrecer pólizas de seguro personalizadas a los hábitos de manejo de un conductor.
- Alertar al conductor cuando se esté acercando a un cruce donde hay una alta incidencia de accidentes.
- Alertar al conductor cuando se esté estacionando en un lugar prohibido, o de alto riesgo.

Si adicionalmente combinamos esta información con la capacidad de identificar al conductor (por medio de biométricos o con mecanismos más rudimentarios como contraseñas), podemos habilitar escenarios tales como restringir a que nuestros hijos solo puedan manejar dentro de cierta área geográfica o llegar a cierto límite de velocidad.

CLUSTER DE INSTRUMENTOS

El cluster de instrumentos es el espacio que despliega información crítica para el conductor, tal como velocidad, temperatura y alertas. En los automóviles modernos se está comenzando a utilizar pantallas digitales en lugar de los tradicionales medidores análogos. En principio podríamos pensar que esto se podría delegar al sistema de infotainment, sin embargo acostumbra manejarse como una capa distinta ya que se requiere desempeño en tiempo real con alta confiabilidad. En otras palabras, no hay mucho problema si el sistema de entretenimiento no despliega correctamente los metadatos de una canción, pero sí hay problema si no sabemos que el auto está sobrecalentado. Es por ello que el cluster de instrumentos requiere operar a bajo nivel sobre un OS para tiempo real conectado al bus interno del automóvil, mientras que el sistema de infotainment opera más arriba sobre frameworks orientados a la información y conectividad externa.

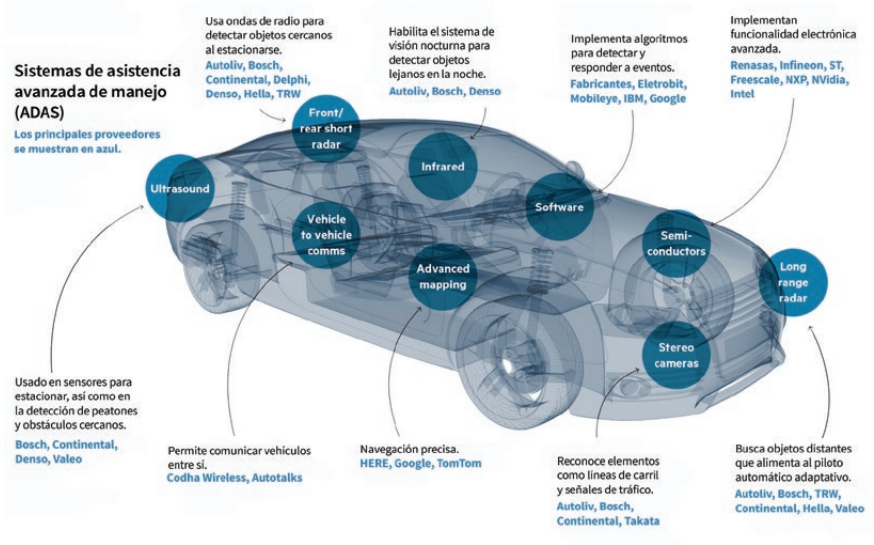


Figura 1. Componentes y proveedores ADAS

ASISTENCIA DE MANEJO

Esta capa, conocida en inglés como ADAS (advanced driver assistance systems) contiene servicios y aplicaciones para ayudar al conductor a manejar el automóvil, por ejemplo alertar al conductor cuando se detectan objetos cercanos o se está saliendo del carril. Los servicios no solo son de alerta, sino que incluso también pueden controlar cierto aspecto del automóvil, como por ejemplo el frenado de emergencia o el control de velocidad adaptativo (adaptive cruise control). La figura 1 muestra los principales componentes involucrados en estos sistemas y las principales empresas proveedoras para cada uno.

CONDUCCIÓN AUTÓNOMA

Se refiere a la plataforma que habilita que un automóvil pueda conducirse de manera autónoma. Involucra la operación de los sensores, motor de analítica/decisión, y control del automóvil.

Como es de esperarse, las capas que están más cercanas al control del automóvil (instrumentos, ADAS, conducción autónoma) típicamente son cerradas y como desarrolladores externos no hay mucha posibilidad de interactuar con ellas. En contraste, la capa de infotainment es donde hay mayor apertura y dinamismo. En la siguiente sección revisamos a los principales jugadores y plataformas.

PLATAFORMAS Y PROVEEDORES

A continuación describo las que considero las plataformas de software más relevantes para automóvil conectado.

QNX

QNX Software Systems es parte de BlackBerry (fue comprada en 2010), y es líder de mercado en el segmento automotriz. El corazón de su tecnología es el Neutrino RTOS, un sistema operativo de tiempo real basado en Unix que destaca por su arquitectura de microkernel.

QNX tiene una alta penetración en las áreas de telemática e infotainment. Automóviles de marcas como Audi, BMW, Ford, Honda y General Motors, entre otros, utilizan esta plataforma. Típicamente, QNX opera como plataforma de bajo nivel, y las marcas construyen sus propias aplicaciones sobre esta.

El desarrollo de aplicaciones para QNX se hace en C++ utilizando el framework Qt. Específicamente en la capa de infotainment (QNX Car Platform for Infotainment) también se puede utilizar tecnologías web como HTML5 y Javascript. [1]

LINUX

Linux ha sido una opción común para las marcas que construyen su propio OS para automóviles. De hecho, el software que opera los automóviles Tesla está basado en Linux, sin embargo es cerrado.

En 2009 se creó la GENIVI Alliance con el objetivo de impulsar la adopción de software open source en sistemas de infotainment. Entre los miembros de alto nivel están empresas como BMW, Honda, Nissan, Daimler y Volvo. El principal resultado de esta alianza es la Genivi Development Platform (GDP),

una plataforma abierta basada en Linux para construir aplicaciones de infotainment. GDP puede funcionar sobre distintas arquitecturas de hardware tales como Intel, NVidia, Qualcomm e incluso tabletas de bajo costo como Raspberry Pi. [2]

Otra iniciativa que vale la pena notar es Automotive Grade Linux (AGL), que es operada por la Linux Foundation. Actualmente, la plataforma de AGL solo cubre infotainment, lo cual la hace parecer redundante con GENIVI, pero a diferencia de esta última AGL busca cubrir todo el espectro de software para un automóvil incluyendo telemática, clúster de instrumentos, ADAS y conducción autónoma. Aunque ambas organizaciones son abiertas, y hay muchas empresas que forman parte de ambas, hay empresas con mayor peso detrás de cada una. Por ejemplo, en el caso de GENIVI BMW es uno de los miembros con mayor protagonismo mientras que en el caso de AGL es Toyota. De hecho, el primer despliegue de AGL en un automóvil comercial será el sistema de infotainment del Camry 2018. [3]

APPLE CARPLAY

CarPlay es básicamente un mecanismo que permite ver y controlar las apps de un iPhone desde la consola de infotainment de un automóvil. Varios automóviles ya lo soportan y funciona con dispositivos iPhone 5 o superior.

ANDROID AUTOMOTIVE

A lo largo de los años Android ha tenido distintos acercamientos en el área de infotainment en automóviles. Posiblemente lo más conocido es Android Auto, que es prácticamente lo mismo que CarPlay (mostrar y controlar las apps de tu smartphone desde el sistema de infotainment), pero para Android.

Otra posibilidad es instalar una capa de Android directamente como sistema de infotainment (sin necesidad de un smartphone). Algunas marcas ya han intentado esto, tomando el código fuente de Android y modificándolo ellos mismos. Al no ser un esfuerzo oficialmente soportado por Google, los resultados no han sido óptimos ya que modificar el sistema

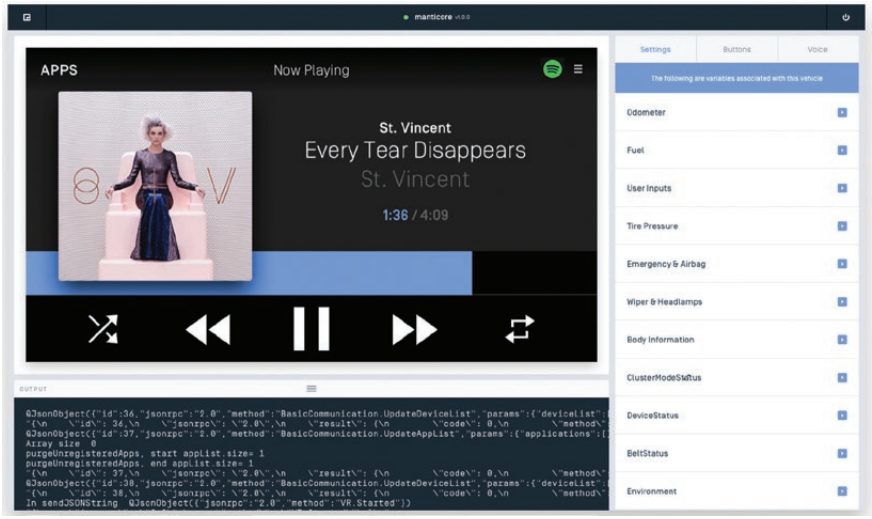


Figura 2. Manticore permite probar apps en SDL desde tu navegador

operativo no es trivial y requiere bastante esfuerzo por lo que los automóviles terminan usando versiones de Android anticuadas; por ejemplo, el Honda Accord 2017 usa Android 4.2, un sistema operativo del 2012. La buena noticia en este sentido es que Google anunció en mayo del 2017 la iniciativa Android Automotive, que básicamente será una versión de Android optimizada para el automóvil y oficialmente soportada por Google. Volvo y Audi están contemplando incluirlo en algunos modelos de 2019. [4]

SMARTDEVICELINK

SmartDeviceLink (SDL) es, al igual que CarPlay y Android Auto, un framework para que los smartphones puedan interactuar con los sistemas de infotainment de los automóviles. La diferencia es que SDL es abierto e independiente de Apple y Google. SDL fue originalmente creado por Ford, pero fue donado como open source y actualmente está siendo adoptado por varias automotoras, entre ellas Toyota (parece ser que el Camry 2018 que usará Automotive Grade Linux también soportará SDL para interactuar con smartphones).

De las distintas tecnologías para interactuar desde el smartphone con el automóvil, SDL es la que me parece más amigable para desarrolladores terceros. En su sitio web [5] puedes encontrar guías de arranque, y los SDK están disponibles en GitHub.

Otra capacidad atractiva para desarrolladores de SDL es Manticore. Esta es una instancia del core de SDL disponible en línea para que puedas probar cómo se comporta tu app en un automóvil con SDL. Lo único que haces es entrar desde un navegador web a Manticore [6], apuntar tu aplicación hacia el URL y puerto que te indique, y se cargará en el navegador tu app corriendo para que puedas probar las interacciones y depurarla (ver figura 2). Su uso es gratuito, simplemente regístrate como desarrollador SDL.

MICROSOFT CONNECTED VEHICLE

Microsoft fue uno de los pioneros en el área de sistemas operativos para infotainment, lanzando Microsoft Auto en 1998 que después de varios cambios de nombre terminó como Windows Embedded Automotive. Las primeras dos generaciones del sistema Sync de Ford estaban basadas en este sistema operativo, sin embargo en 2014 Ford movió Sync hacia QNX. En 2016 Microsoft hizo un cambio drástico de estrategia, abandonando la intención de ser el sistema operativo del automóvil y enfocándose en ofrecer servicios basados en la nube que habiliten aplicaciones en el automóvil. Esta iniciativa actualmente lleva el nombre de Microsoft Connected Vehicle y entre otras cosas abarca servicios de infotainment (llamadas por Skype, servicios de Cortana), analítica predictiva basada en telemática, e información en tiempo real sobre las condiciones del camino. [7]

SISTEMAS DE CÓMPUTO

Una batalla no menos interesante es la que se está dando en el lado del hardware sobre el que se ejecuta el sistema

operativo del automóvil. Hasta hace poco, los automóviles usaban sistemas en chip (SoC) de gama relativamente baja. Sin embargo, conforme los automóviles se hacen más inteligentes demandan soluciones expresamente diseñadas para este segmento y con mayor poder. Estos son los jugadores y lo que están haciendo ...

Renesas Electronics, una empresa de origen japonés especializada en semiconductores y microcontroladores para el sector automotriz es uno de los jugadores originales de este segmento. Aunque tradicionalmente ha estado enfocada en el área de instrumentación y ADAS, recientemente mostró un prototipo de vehículo autónomo.

NVIDIA se ha posicionado rápidamente como líder en cómputo para inteligencia artificial para conducción autónoma, siendo la elección de marcas como Tesla y Toyota. Su oferta para automóviles considera: Drive PX, una tarjeta de cómputo basada en GPUs optimizada para procesar la información captada por las cámaras y sensores del automóvil; Driveworks, un SDK con herramientas y bibliotecas para construir capacidades de conducción autónoma y NVIDIA DGX que son sistemas para el centro de datos optimizados para aprendizaje profundo (deep learning), estos sistemas analizan los datos obtenidos por los automóviles, los analizan y en base a ello entrenan a la computadora del automóvil. Es por ello que Tesla indicó que el Modelo S ya incluye todo el hardware necesario para conducción autónoma pero esta capacidad será activada hasta que se hayan analizado suficientes horas de conducción, lo que está sucediendo es que estos automóviles vienen equipados con una computadora Drive PX que estará enviando información al centro de datos para ser analizada y en base a ello entrenar a la computadora a bordo para tomar decisiones instantáneas. [8]

Intel recientemente lanzó la línea de SoC Atom A3900 orientada específicamente al sector automotriz. Este sistema está diseñado para habilitar clusters de instrumentos y sistemas de infotainment. Sin embargo, la meta importante para Intel está en alcanzar a NVIDIA en el ámbito de cómputo para conducción autónoma. Su estrategia es similar, contemplando una computadora SoC a bordo del automóvil que se comunique con sistemas en un centro de datos donde se haga el aprendizaje profundo; sin embargo, la oferta de Intel también considera la plataforma para comunicación inalámbrica utilizando tecnología 5G. [9]

Qualcomm ha sido un poco lento en entrar al espacio automotriz, pero ya está corrigiendo el rumbo. Por el momento está empujando el Snapdragon 820a como SoC para habilitar capacidades de infotainment y telemática. Su propuesta hace fuerte énfasis en la conectividad, de hecho uno de los puntos a destacar del Qualcomm Connected Car Reference Platform que estará disponible en la segunda mitad del 2017 es que se basa en el modem X16 LTE que puede alcanzar velocidades de Gigabit. Más allá de esto, la reciente compra de NXP Semiconductors habilitará a Qualcomm para ofrecer sistemas automotrices más avanzados y de conducción autónoma. [10]

BOSCH Y SU OMNIPRESENCIA

Conforme me acerco al final de este artículo considero que me hace falta mencionar a Bosch, pero confieso que no encuentro donde acomodarlos, ya que están involucrados en distintos niveles. Bosch tiene una amplia tradición como proveedor de partes para el automóvil tradicional, pero se ha modernizado junto con el automóvil. También provee sensores y componentes para manejo asistido (ADAS), así como componentes clave para vehículos eléctricos.

Entre las tecnologías que ha creado Bosch para el automóvil moderno está CAN (Controller Area Network) [11] que es el bus de comunicación que conecta todos los sensores y componentes electrónicos. Es un protocolo de comunicación serial que soporta control en tiempo real con alta seguridad, y que prácticamente todos los automóviles modernos utilizan.

Pero más allá de los componentes individuales para automóviles, lo que destaca a Bosch es su visión y capacidad para soluciones integrales en el contexto de ciudades inteligentes. Un ejemplo de esto sería un sistema de estacionamiento inteligente donde un automóvil que está buscando lugar de estacionamiento pueda ser informado por la red de automóviles en la cercanía sobre dónde hay lugar disponible. Este tipo de soluciones involucra una gran cantidad de subsistemas tales como la inteligencia artificial para detectar lugares, comunicación entre vehículos y coordinación con infraestructura de la ciudad. Bosch es una de las pocas empresas que está pensando a ese nivel, y que tiene amplias capacidades tanto a nivel de ingeniería de bajo nivel como de integrar "sistemas de sistemas". Por cierto, vale la pena mencionar que Bosch recientemente abrió un centro de desarrollo de software y servicios de ingeniería en Guadalajara, así que seremos testigos de cerca de lo que hagan en este espacio. [12]

CONCLUSIÓN

El automóvil conectado involucra una gran variedad de tecnologías y empresas que lo habilitan. En este artículo echamos un vistazo a algunos de los jugadores clave y sus estrategias. Algunas de estas tecnologías son cerradas o fuera del alcance de los desarrolladores independientes, pero hay otras que son abiertas y con las que cualquiera puede comenzar a experimentar. Así que si te interesa, te invito que consultes los enlaces que comparto en la sección de referencias para conocer más.

Referencias

[1] http://www.qnx.com/content/qnx/en/products/qnxcar
[2] https://www.genivi.org
[3] https://www.automotivelinux.org
[4] https://source.android.com/devices/automotive
[5] https://www.smartdevicelink.com
[6] https://smartdevicelink.com/resources/manticore/
[7] https://www.microsoft.com/en-us/internet-of-things/connected-vehicles
[8] http://www.nvidia.com/object/drive-px.html
[9] https://www.intel.com/content/www/us/en/automotive/automotive-overview.html
[10] https://www.qualcomm.com/solutions/automotive
[11] https://www.canbus.us
[12] http://www.bosch-mobility-solutions.com

DEEP LEARNING Y
VEHÍCULOS AUTÓNOMOS

Por Gary Silberg, et al.

Piensa en todas las tareas que tu cerebro ha hecho mientras conduciste tu automóvil al trabajo esta mañana. Viste 235 letreros, 12 bicicletas, 20 peatones y 1,376 automóviles. Aceleraste, frenaste, diste vuelta y cambiaste de carriles.

Ahora imagina escribir un programa que pudiera conducir a un automóvil en ese mismo viaje. Probablemente requeriría miles de millones de líneas de código.

Parece imposible, pero hacia allá vamos. El aprendizaje profundo (deep learning), una forma de inteligencia artificial, está llegando al punto de poder conducir automóviles de forma autónoma y en cualquier tipo de condición.

¿QUÉ ES EL APRENDIZAJE PROFUNDO?

Aunque no lo sepas, seguramente ya has estado expuesto a aplicaciones de aprendizaje profundo. Es lo que hace que Siri pueda responder a tus preguntas, Google lo usa para encontrar los resultados que buscas, y Amazon para recomendarte un producto. En el ámbito automotriz, el aprendizaje profundo permite que los ingenieros no tengan que programar cada uno de los eventos y acciones posibles que se darían en un trayecto. En lugar de esto, la computadora recopila experiencias y a partir de estos datos genera deducciones para la toma de decisiones.

El sistema de aprendizaje de un automóvil autónomo trabaja aprendiendo a detectar objetos y patrones de forma automática. Conforme se le alimentan más datos va aumentando su precisión. El sistema es entrenado por medio de repetición, continuamente recibiendo datos de entrada y optimizando su habilidad para realizar predicciones sobre qué significan los datos y qué hacer con ellos. Al desplegar un sistema de este tipo a través de una flotilla de vehículos, la flotilla entera se beneficia del conocimiento colectivo generado por cada uno de los vehículos. Cuando un automóvil encuentra algo nuevo —por ejemplo un camino no mapeado o una acción no contemplada por parte de otro automóvil— envía la experiencia al sistema de aprendizaje, éste la analiza y envía el aprendizaje a toda la flotilla.

De esta manera, el aprendizaje profundo tiene la capacidad de resolver uno de los principales retos de la conducción autónoma: lidiar con lo impredecible. Sería prácticamente imposible que ingenieros humanos programaran manualmente algoritmos para expresar de forma adecuada todos los escenarios posibles al manejar.

El aprendizaje profundo ya es utilizado para habilitar capacidades de conducción autónoma en automóviles comerciales. Por ejemplo, el sistema de cámaras del Audi A7 utiliza aprendizaje profundo para reconocer señales de tráfico. El sistema Autopilot de Tesla también lo utiliza, y aunque el Modelo S ya tiene todo el equipo necesario para conducción autónoma, por el momento se encuentra recopilando experiencias de los automóviles en las calles y aprendiendo, para que cuando se determine que ha aprendido lo suficiente se pueda activar la conducción autónoma de niveles superiores.

Hacia el futuro, la mayoría de las empresas automotrices han anunciado que planear tener vehículos autónomos en el mercado alrededor del año 2020. En un principio estos vehículos estarán restringidos a ambientes sencillos y bien mapeados, pero eventualmente serán capaces de llevarnos a cualquier lugar sin importar las condiciones.

TÉCNICA 1: ABSTRACCIÓN SEMÁNTICA

Existen dos grandes técnicas para aplicar el aprendizaje profundo a la conducción autónoma: abstracción semántica, y aprendizaje de punta a punta.

La abstracción semántica se enfoca en dividir el problema en distintos componentes. Por ejemplo, un componente se dedica a detectar vehículos, otro componente detecta carriles, otro más detecta qué hay más allá del carril (otros automóviles, banqueta, barrera). Utilizando conocimiento previo, los ingenieros de software entrenan cada componente individual etiquetando o anotando imágenes para identificar su significado en el mundo real.

Una vez que cada uno de los componentes ha dominado su tarea, los distintos componentes comparten datos para colaborar y decidir cómo controlar el vehículo. Es decir, una vez que el componente de detección de vehículos sabe dónde están los vehículos, y el componente de detección de carriles sabe dónde están los carriles, y el componente de detección de fronteras sabe dónde está la banqueta, combinan este conocimiento con muchos otros datos aportados por otros componentes. La computadora maestra entonces decide qué trayectoria tomar y cuál es la velocidad adecuada para realizar la maniobra de manera segura.

Ventajas:

- Menor tolerancia a fallas. Antes de que la red sea descargada a un vehículo autónomo activo, los algoritmos responsables de cada componente son entrenados de forma independiente hasta que cada uno tenga un alto nivel de precisión en su tarea.
- Facilita detectar errores. En caso de que haya una falla —el automóvil toma una decisión incorrecta— los ingenieros pueden detectar qué componente provocó el error y corregirlo.
- Mayor capacidad para lidiar con lo impredecible: El sistema ha sido alimentado previamente con datos etiquetados que lo ayudan a adaptarse a situaciones inusuales. Por ejemplo, si de pronto un niño cruza la calle persiguiendo una pelota, el sistema sabe que el niño es una persona, cómo se comportará y cómo evitar golpearlo.

Desventajas:

- Se requiere mucho trabajo previo. Antes de que el sistema esté disponible el equipo de ingenieros debe

Gary Silberg es socio director para sector automotriz en KPMG Estados Unidos.

diseñar una red de conocimiento y codificar algoritmos para cada componente. Posteriormente tiene que entrenar cada componente por medio de etiquetar meticulosamente grandes cantidades de datos, lo cual requiere mucho tiempo y esfuerzo.

• Típicamente se etiqueta más información de la necesaria. Dado que se busca cubrir de forma anticipada todos los escenarios posibles es necesario etiquetar una gran cantidad de datos y al final muchos de ellos serán redundantes, por lo que hay un desperdicio en el esfuerzo.

TÉCNICA 2: APRENDIZAJE DE PUNTA A PUNTA

El aprendizaje de punta a punta (end-to-end learning) es una técnica más disruptiva en la que el aprendizaje se realiza de forma automática por medio de captar y analizar experiencias reales de manejo. En vez de dividir la conducción en distintos aspectos y entrenar componentes por separado, el sistema opera de forma holística a través del conjunto de datos completo, sin enfocarse en detectar eventos en componentes individuales.

Ventajas

• Al ser alimentado con big data, el sistema puede clasificarlos sin necesidad de que los datos estén previamente etiquetados. El sistema no necesita que previamente se le diga qué es un vehículo o cómo se ve un carril, sino que toma las entradas de datos y automáticamente clasifica los objetos y situaciones conforme los va experimentando. Esto significa que los ingenieros no necesitan predefinir todos los componentes que se conjuntan para conducir.

• Conforme hay más datos sobre distintas situaciones, el sistema se hace más robusto. Las empresas automotrices pueden aprovechar el poder de su flotilla para rápidamente captar millones de horas de experiencia.

• Se optimiza considerando el proceso completo. Los algoritmos internos son optimizados teniendo en consideración el desempeño del sistema completo, no de componentes individuales (como por ejemplo detectar carriles). Esto puede llevar a sistemas con mejor desempeño, que resuelvan las tareas usando el mínimo de pasos necesarios.

Desventajas

• Se requiere capturar cantidades de datos enormes. Como ya comentamos, en el caso de los automóviles esto se puede mitigar acumulando la experiencia de una flotilla grande pero en otros casos de uso esto podría no ser posible.

• Es difícil entrenar para escenarios fuera de lo normal. Es posible que aún después de acumular millones de horas no se cuente con experiencia de casos extremos. En estos casos, los sistemas deben ser aumentados con asistencia humana para ayudar al sistema a tomar una decisión acertada.

•Técnicamente complicado. El diseño de estas redes de aprendizaje es complicado, requiere conocimientos avanzados y tiempo para optimizar los algoritmos.

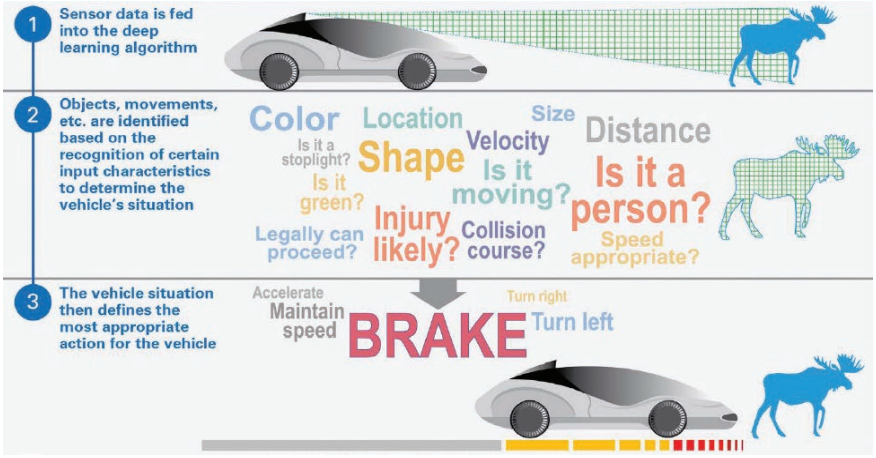


Figura 1. El proceso de pensamiento de un vehículo autónomo

VER, PENSAR, CONDUCIR Y APRENDER

Para lograr vehículos autónomos necesitamos que dichos vehículos tengan la capacidad de ver, pensar, conducir y aprender para mejorar. A continuación explicamos cómo funciona cada uno de estos pasos.

La capacidad de "ver" se refiere a cómo es que los vehículos detectan el ambiente a su alrededor. Para poder ver, los vehículos hacen uso de tres grandes fuentes de datos:

• Sensores: Existen distintos tipos de sensores que obtienen datos en tiempo real sobre el entorno del automóvil. Los sensores más usados son: ópticos (cámaras que capturan color y contraste), LIDAR (pulsos de luz), radar (radio ondas), ultrasónico (alta frecuencia). Cada uno tiene un mejor desempeño en distintas condiciones, por lo que típicamente se usa una combinación de varios sensores para asegurar que el automóvil pueda detectar con precisión su entorno en todo tipo de condiciones.

• Datos en la nube: Para reconocer su entorno los vehículos también hacen uso de datos en la nube tales como mapas y rutas, señales de tráfico, condiciones del camino.

• Datos de otros dispositivos: Al corto plazo, el escenario más común es el de un smartphone que pueda proveer información al vehículo, pero eventualmente los automóviles tendrán comunicación entre sí (vehículo a vehículo), así como la infraestructura en ciertos lugares (ej. un estacionamiento inteligente que informa al automóvil donde hay espacios disponibles).

"Pensar" se refiere a cómo el automóvil integra la información obtenida y determina la acción correspondiente. La figura 1 ilustra este procedimiento a grandes rasgos. La ejecución de este procedimiento (integrar datos, analizarlos y decidir una acción) debe tomar milisegundos a lo mucho. Adicionalmente, a partir de lo que está sucediendo en el momento, el automóvil debe ir extrapolando para intuir lo que sucederá en el futuro inmediato. El aprendizaje profundo ayuda a realizar dicha extrapolación.

Una vez que el sistema toma una decisión, requiere ejecutarla de manera confiable y segura, a esto nos referimos con "conducir". Por ejemplo, debe determinar el ángulo de giro adecuado, así como la intensidad de aceleración o frenado. A su vez, los distintos componentes proveen retroalimentación sobre el resultado de la acción en ejecución (ej. que sí se esté logrando disminuir la velocidad de acuerdo a lo planeado, que se esté manteniendo la tracción necesaria).

Cada que un automóvil se encuentra con una nueva experiencia o caso extremo, el sistema lo analiza y "aprende" de lo sucedido. El aprendizaje se despliega a toda la

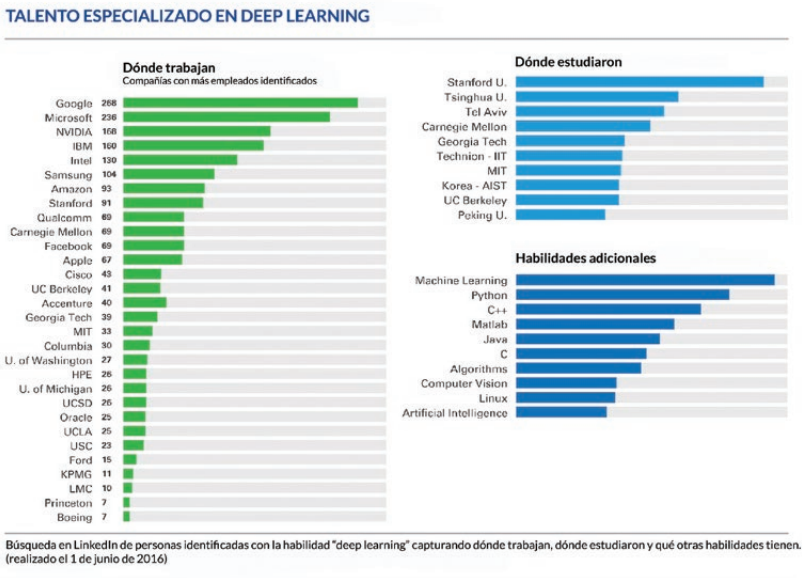


Figura 2. Talento especializado en deep learning

flotilla de vehículos para que estén preparados la próxima vez que se encuentren con dicho escenario.

IMPACTO EN LA INDUSTRIA

Conforme nos dirigimos a la conducción autónoma, las empresas del ecosistema automotriz deben reconocer las implicaciones de este cambio. En principio, surgirá un nuevo mercado para servicios de movilidad basados en conducción autónoma y conectividad. Los vehículos podrían estar en uso continuo, por lo que habrá el modelo de propiedad de vehículos cambiará, y la cantidad de kilómetros que acumula cada carro aumentará significativamente. KPMG estima que el mercado de servicios basados en conducción autónoma tendrá un valor anual de más de un millón de millones de dólares para el año 2030.

La conducción autónoma también traerá una reducción significativa de accidentes viales, ya que se estima que el 94% de los accidentes son provocados por errores humanos. Uno de los sectores que más será afectado por este cambio es el de las aseguradoras. Otro sector que será afectado de forma indirecta es el de las autopartes y talleres de reparación.

Las empresas automotrices están desarrollando nuevas capacidades para los automóviles que aprovechen este contexto, desde tecnologías para mejorar "el sistema nervioso" del automóvil hasta servicios de entretenimiento y productividad para los pasajeros (que estarán liberados de tener que conducir el automóvil y ahora podrán hacer otras cosas a bordo). Empresas de software como Google, Apple y Microsoft ya están buscando penetrar el segmento de software y servicios a bordo para pasajeros, pero las empresas automotrices no cederán fácilmente esta rebanada del pastel y ya están desarrollando tecnología y canales para ofrecer estos servicios bajo sus propios términos.

CONSIDERACIONES PARA EL NUEVO PARADIGMA

Las empresas automotrices que no reaccionen de forma sabia y rápida a los efectos que provocará este nuevo paradigma en su industria, desaparecerán en pocos años. Estas son algunas consideraciones que deben tener en cuenta:

Las empresas automotrices ya no se dedicarán solamente a hacer automóviles. La mayoría evolucionará a ser proveedores de servicios de movilidad. La estrategia que cada una elija afectará cómo generará ingresos y si logra mantenerse en el mercado.

Los kilómetros son oro. El aprendizaje profundo detonará una carrera para acumular kilómetros manejados. De especial valor serán las situaciones extraordinarias que extiendan el aprendizaje de los vehículos.

El talento es clave. Las personas con la capacidad de construir y operar sistemas de aprendizaje profundo serán muy valiosas. Actualmente la mayoría de las personas con esta habilidad trabaja en empresas de software (ver figura 1) pero las empresas automotrices ya se las están peleando.

El sistema nervioso será el enfoque de diseño del automóvil. Entramos a una nueva era de diseño automotriz centrada en el "sistema nervioso" del automóvil, que incluye el cerebro del automóvil, sensores, controladores y telecomunicación. Este es un cambio radical para los fabricantes de automóviles.

El poder de la flotilla eclipsará la importancia del automóvil individual. La interacción con una flotilla y el ecosistema será esencial. La conectividad asegurará una interacción constante entre los integrantes de la red.

La forma de innovar cambiará. Conforme la industria automotriz adopta y desarrolla capacidades de aprendizaje profundo, aprenderá a innovar de una forma distinta que afectará a otras industrias.

Las plataformas de software automotriz serán clave. Las plataformas de software surgirán como un mecanismo para operar y actualizar el automóvil. Contar con una plataforma sofisticada, segura, capaz de desplegarse en distintos modelos de automóvil y que pueda ser actualizada fácilmente, será un elemento crucial de competitividad para las empresas automotrices.

CONCLUSIÓN

El aprendizaje profundo está acelerando las capacidades de autonomía y antes de que nos demos cuenta estará en la mayoría de los vehículos. Esto traerá cambios masivos. Aunque nadie sabe con certeza lo que sucederá, es un hecho que se generarán nuevas oportunidades para empresas de distintas industrias: automotriz, software, servicios, entretenimiento. Específicamente, las empresas automotrices deben revisar en qué espacio competirán, con qué producto y servicios, cómo diseñarán los automóviles, cómo gestionarán su flotilla, y cuál es su enfoque hacia la innovación, el talento y la tecnología.

Este artículo es una versión traducida y editada por Software Guru del reporte original "I see, I think, I drive (I learn)" publicado por KPMG.

Referencia

[1] G. Silberg, et al. "I see, I think, I drive (I learn)". KPMG.
http://swgu.ru/t4

LOS 6 NIVELES DE AUTONOMÍA

Por Pedro Galván

"Esta empresa ya tiene vehículos con nivel 3 autonomía, y se espera que antes del año 2020 lance vehículos nivel 4."

¿Alguna vez has leído o escuchado una frase similar, donde se hable de niveles de autonomía en los vehículos? Si no te ha sucedido, pronto sucederá. Así que es buena idea de una vez entender a qué se refiere eso de los niveles de autonomía.

A continuación comparto una guía que explica qué significa cada nivel, desde el nivel 0 donde el humano hace todo hasta el nivel 5 que es completamente autónomo.

La escala de niveles de autonomía fue publicada originalmente por la Society of Automotive Engineers (SAE) en el año 2014 como parte de su reporte "Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems". El reporte describe un mapa de ruta en el que los fabricantes de vehículos se están basando para desarrollar las capacidades de sus vehículos autónomos. En Estados Unidos, la National Highway Traffic Safety Administration (NHTSA) había ya creado su propia escala en 2013 sin embargo en septiembre de 2016 decidió descartarla y adoptar la escala de SAE como estándar. A continuación describo los distintos niveles de la escala SAE.

NIVEL 0: SIN ASISTENCIA

Este es el nivel inicial, en el que no hay asistencia y por lo tanto el conductor realiza todas las tareas (acelerar, frenar, girar) en todo momento. Es posible que el automóvil tenga sistemas de notificación al detectar eventos u objetos, pero aún así el conductor tiene el control completo. La mayoría de los automóviles en las calles actualmente se encuentran en este nivel.

NIVEL 1: CONDUCCIÓN ASISTIDA

Estos automóviles pueden tener uno o más sistemas que controlen la velocidad o la dirección, pero no ambos al mismo tiempo. Por ejemplo, algunos automóviles modernos cuentan con piloto automático adaptativo (que acelera o frena de forma automática dependiendo del tráfico), que es un ejemplo de una capacidad de nivel 1. Otro ejemplo son los sistemas con control de carril, que automáticamente evita que un automóvil se salga de su carril. En resumen, el conductor sigue controlando el carro y debe mantener las manos en el volante, pero recibe asistencia para escenarios específicos.

NIVEL 2: AUTONOMÍA PARCIAL

Un vehículo nivel 2 tiene capacidad de tomar el control tanto de la dirección como de la aceleración y frenado, pero solo bajo ciertas condiciones y por periodos cortos de tiempo (típicamente menos de un minuto). Aunque el automóvil se está conduciendo solo, el conductor debe estar detrás del volante y atento para tomar el control en cualquier momento. Tesla Autopilot ha estado en este nivel desde hace un par de años, aunque ya otras marcas tienen en el mercado automóviles con capacidades similares.

NIVEL 3: AUTONOMÍA CONDICIONAL

En el nivel 3 el automóvil ya se puede conducir solo en la mayoría de las situaciones y por periodos extendidos de tiempo, pero al igual que en el nivel 2 se requiere que un conductor humano esté detrás del volante por si el automóvil no sabe qué hacer en alguna situación y necesita ceder control al conductor. El problema con este nivel es desde la perspectiva legal, ya que en caso de algún accidente no está claro qué tanta responsabilidad tendría el conductor; es por ello que algunas empresas planean saltarse este nivel.

NIVEL 4: ALTA AUTONOMÍA

Un automóvil con autonomía nivel 4 se conduce por sí solo y sin necesidad de que una persona esté al pendiente. En caso de que el automóvil se encuentre ante una situación en la que no pueda conducirse —por ejemplo durante una tormenta donde se reduzca significativamente la visibilidad—, busca un lugar seguro donde detenerse y espera a que se resuelva la situación o el pasajero le provea información adicional. Un automóvil nivel 4 no necesariamente cuenta con controles para conducción manual. Algunas empresas ya tienen prototipos con autonomía nivel 4 —tal es el caso de Waymo (Google)—, y en el caso del Modelo S de Tesla ya cuenta con todo el hardware necesario, pero las computadoras a bordo todavía están recopilando horas de manejo para entrenar los algoritmos de conducción autónoma; una vez que se cuente con suficiente entrenamiento se activará en los automóviles este nivel de autonomía. Otras marcas como Ford y Volvo han anunciado que planean tener disponibles autos nivel 4 a más tardar en el año 2021.

NIVEL 5: AUTONOMÍA COMPLETA

En este nivel el automóvil se conduce por sí solo en todos los escenarios, incluso los más extremos. Aunque el paso entre nivel 4 y 5 parece pequeño, en realidad involucra superar una gran cantidad de retos técnicos para poder satisfacer todos los escenarios. Es por ello que ninguna empresa ha hecho compromisos para tener automóviles nivel 5 en menos de siete años.

Referencia

[1] https://www.sae.org/misc/pdfs/automated_driving.pdf

Pedro Galván es cofundador y director editorial de Software Guru.

LA NECESIDAD DE 5G

Por Pedro Galván

Desde el punto de vista de comunicación por medio de un smartphone, la tecnología de comunicación inalámbrica 4G nos provee prácticamente todo lo que necesitamos. Con LTE podemos hacer llamadas, chatear, navegar sitios web, hacer streaming de música e incluso de video. Bajo este criterio, podríamos pensar que el mundo no necesita una nueva generación de conectividad inalámbrica; después de todo, no es como si nos serviría de mucho poder hacer streaming en resolución 4K de 10 películas al mismo tiempo.

Pero si pensamos que un automóvil autónomo, para poder operar requiere estar recibiendo cientos de Megabytes de datos por segundo con una latencia de unos cuantos milisegundos, con una muy alta disponibilidad y que en un kilómetro cuadrado podemos llegar a tener miles de automóviles que demandan la misma conectividad, entonces nos quedará claro que necesitamos una nueva generación de tecnología inalámbrica. Es por ello que la industria ya está volteando hacia la nueva generación de conectividad inalámbrica, las redes 5G.

CARACTERÍSTICAS DE 5G

Aunque todavía no hay un estándar con las características específicas que deben cumplir las redes 5G, sí hay metas de diseño. Algunas de ellas son:

- Comunicación "ultra confiable" para soportar comunicación de misión crítica.
- Latencias de unos cuantos milisegundos.
- Distintas escalas de transferencia de datos dependiendo de la cantidad de conexiones, desde Gigabits por segundo para cientos de conexiones hasta decenas de Megabits por segundo para decenas de miles de conexiones.
- Mejora significativa de cobertura respecto a 4G.
- Alta eficiencia en el uso del espectro.

Mientras que las redes celulares 3G y 4G fueron diseñadas para comunicación entre personas, las redes 5G serán diseñadas para comunicación máquina-a-máquina. Es por ello que hay un énfasis importante en la latencia y confiabilidad, ya que de esta conectividad dependerá la operación exitosa de automóviles, maquinaria e infraestructura inteligente.

Estas características hacen de la tecnología 5G un candidato ideal para las comunicaciones denominadas "V2X". Este acrónimo se refiere a comunicación entre un vehículo y el exterior, incluyendo no solo vehículo a vehículo (V2V) sino también vehículo a peatón, vehículo a nube, o vehículo a infraestructura (V2I) —por ejemplo, un semáforo.

DSRC, UN ESTÁNDAR COMPETIDOR

Actualmente ya existe un estándar propuesto para sistemas de transporte inteligente conocido como DSRC (Dedicated Short-Range Communications) que habilita escenarios V2V. De hecho, el Cadillac STS 2017 ya implementa esta tecnología, y otras empresas como Mercedes-Benz han anunciado modelos próximos con DSRC. Incluso la Comisión Federal de Comunicación (FCC) en Estados Unidos está considerando la posibilidad de obligar a todas las marcas a equipar sus automóviles nuevos con DSRC.

A pesar de que DSRC es una tecnología que ya está implementada y parece ser suficiente para resolver escenarios V2V, expertos indican que esto solo será una solución temporal, y que una vez que se despliegue 5G esta reemplazará a DSRC ya que podrá hacer todo lo que DSRC y más. Adicionalmente, dado que 5G se utilizará para comunicar una gran variedad de dispositivos, logrará mayores economías de escala y por lo tanto menores costo.

ALGUNAS POSIBILIDADES

Se estima que la comunicación V2X traerá grandes beneficios. Aquí algunos escenarios que se contemplan:

- Capacidad de las autopistas. Un estudio encontró que al coordinar el uso del piloto automático adaptativo en varios automóviles se puede aumentar la densidad de automóviles en las calles en un orden de 50 a 80% [3].
- Disminución de tráfico. Con automóviles intercomunicados se disminuirían los embotellamientos ya que se coordinarían automáticamente para hacer cambios de carril y cruces en calles perpendiculares con la máxima eficiencia posible.
- Estacionamiento automático. Si los automóviles saben dónde están los lugares de estacionamiento, esto acelera el flujo y disminuye la congestión.
- Adicionalmente, la tecnología 5G será clave para proveer información y entretenimiento a los pasajeros a bordo.

CONCLUSIÓN

La tecnología 4G apenas está terminando de desplegarse en algunas regiones y parece ser suficiente para escenarios de comunicación entre personas. Sin embargo, los escenarios de conducción autónoma y ciudades inteligentes requieren una nueva generación de comunicación inalámbrica. En Asia se realizarán los primeros despliegues de redes 5G durante el 2018, así que en Latinoamérica podríamos estar viendo esta tecnología alrededor del 2020. Parece distante, pero antes de que nos demos cuenta las redes 5G serán una realidad que habilitarán escenarios inimaginables hasta hace poco.

Referencias

[1] "A look ahead at 5G's impact on the automotive industry". Qualcomm. http://swgu.ru/t5

[2] S. Shladover, D. Su. & X. Lu. "Impacts of Cooperative Adaptive Cruise Control on Freeway Traffic Flow," Proceedings of the 91st Annual Meeting of the Transportation Research Board.

Pedro Galván es cofundador y director editorial de Software Guru.

THE FUTURE OF TRANSPORTATION STACK

COMET LABS

Fuente: "263 Self-Driving Car Startups to Watch". https://blog.cometlabs.io/263-self-driving-car-startups-to-watch-8a9976dc62b0

Adquisición y Análisis de Datos en el STC Metro

Por Liliana Toledo

La adquisición y análisis de datos cobra especial interés en un mundo que se dirige a la digitalización. Para lograr esto, las señales análogas deben ser capturadas, purificadas de ruido y acondicionadas rápidamente para ser de utilidad. El Sistema de Transporte Colectivo Metro es una de tantas organizaciones que enfrenta este reto de digitalización. En este artículo compartimos el caso de un proyecto realizado recientemente por esta organización.

CONTEXTO

El STC se construyó hace más de cuarenta años con tecnología de punta de esos momentos. Cuatro décadas después, darle mantenimiento a este sistema para que pueda atender a más de 5 millones de usuarios diarios, tiene diversas complicaciones. Una de ellas surge de que la mayoría de los componentes electrónicos en los vagones utilizan tecnología vieja que ya no es soportada.

Tal es el caso de los equipos de tracción que utilizan más de 17 mil tarjetas electrónicas para controlar el movimiento de los más de 150 trenes. Aunque los vagones siguen funcionando bien, los escáneres de lógica de tracción se diseñaron con microcontroladores antiguos que después de tantos años presentan fallas de cableado y dejan de funcionar. Al solicitar nuevos equipos al proveedor comentó que ya no soportaba esa tecnología y la única opción era comprar vagones nuevos.

SOLUCIÓN

A través de una colaboración con el Centro de Ingeniería y Desarrollo Industrial (CIDESI) con sede en Querétaro, se estableció el Laboratorio de Electrónica Digital Avanzada (LEDA), que homologa las comunicaciones de este medio de transporte en un formato de electrónica digital. CIDESI se encarga principalmente de implementar tecnología para adquisición y análisis de datos, mientras que el personal de STC aporta su profundo conocimiento de los trenes y operación.

Uno de los proyectos derivados de esta iniciativa fue la actualización del sistema de diagnóstico de las tarjetas de potencia. Anteriormente, este era un procedimiento lento y complicado en el que las señales analógicas de voltaje y/o corriente eran sometidas a diversos sistemas separados de prueba que incluían osciloscopio, fuente, generador, etcétera. Cada uno entregaba diferentes gráficas y en base a ellas los técnicos diagnosticaban las reparaciones requeridas.

El sistema de pruebas ahora se basa en una plataforma de software a la que se conectan tarjetas de adquisición de datos y las tarjetas de tracción. Lo más interesante es que, a través de la plataforma, se pueden simular las condiciones de uso real de los trenes y con ello extraer los datos correctos para hacer el diagnóstico. El sistema de pruebas consta de dos etapas. La primer etapa consiste en obtener una señal analógica, limpiarla y digitalizarla. En la segunda etapa, la plataforma trabaja con esta señal digital para inyectar señales que simulan el comportamiento de un tren en movimiento. Ya con los resultados de la simulación, el técnico determina que área de la tarjeta de tracción es la que presenta problemas.

BENEFICIOS

Sergio Guevara, laboratorista del STC, nos comenta que el nuevo sistema ha cuadruplicado su capacidad para diagnosticar tarjetas. Adicionalmente, el nuevo sistema disminuye significativamente la curva de aprendizaje de los técnicos y operarios, ya que es mucho más amigable.

Otra ventaja consiste en que el conocimiento para diagnosticar las tarjetas ya no solo está en la mente de los técnicos que trabajan en STC por muchos años y que al jubilarse o cambiar de empleo se pierde, sino que ahora también reside en una plataforma que permite tener esa información siempre disponible.

Después de este proyecto, se han hecho otras evoluciones tecnológicas dentro de STC, la mayoría basada en software, como por ejemplo, las pruebas de los pilotos automáticos de los trenes.

El Sistema de Transporte Colectivo se incorpora de esta forma a la era moderna de la adquisición y análisis de datos, teniendo como misión, dentro de su recién inaugurado Laboratorio de Electrónica Digital Avanzada, seguir evolucionando conforme el metro lo requiera.

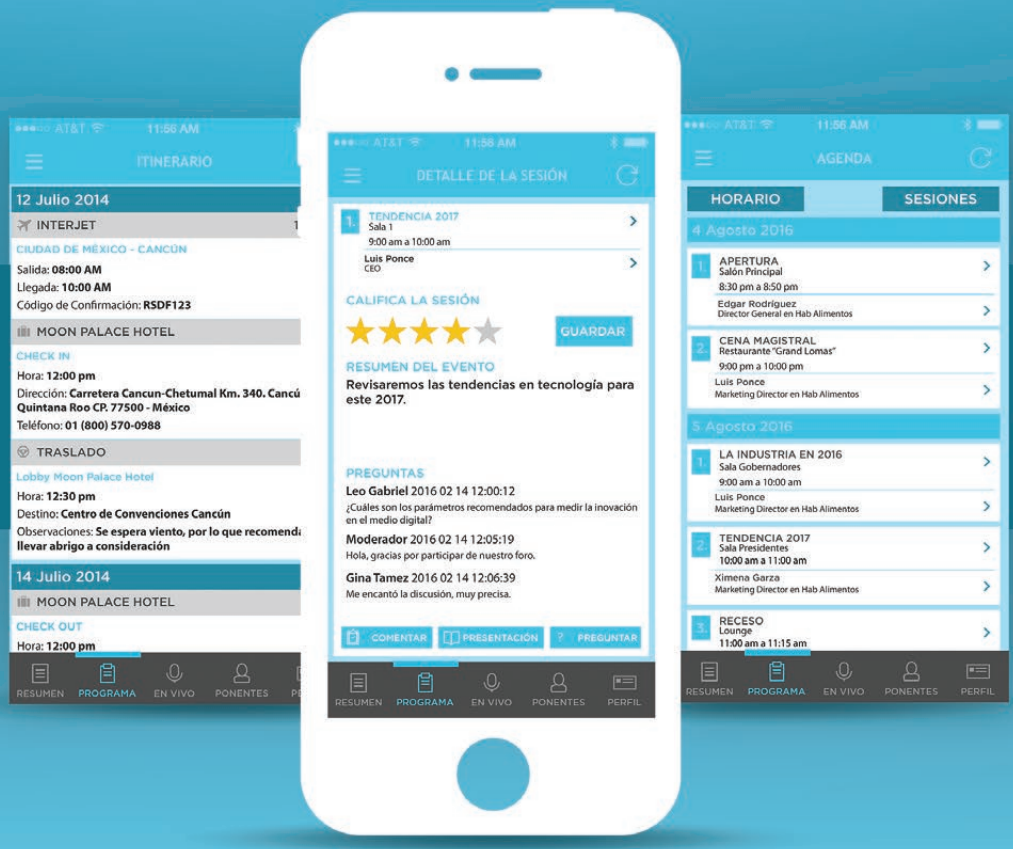
Liliana Toledo Rivera (@lilitoledo_), egresada de la Ingeniería de Comunicaciones y Electrónica por la Universidad de Guadalajara; dentro de su multifacética ruta profesional es periodista de ciencia y tecnología en diversos medios.

Eventto

Apps para todos tus **eventos**

Imagina la experiencia de vivir tu evento *totalmente digital*

Itinerario de vuelos, hoteles y transporte *por invitado*



Encuestas y votaciones en *tiempo real*

Solicita tu **Demo**

iOS

Más de **150** eventos en todo el **mundo**

www.eventto.com.mx

Personaliza **Eventto** con tu marca como nuestros clientes

ARCACONTINENTAL · Coca-Cola · BCD travel · HAVAS MEDIA · GRUPO BIMBO · HEINEKEN · Chemours · gsk · Hard Rock Hotel · ALL INCLUSIVE COLLECTION · Lilly · Roche · SANOFI · SAT MEXICO CORPORATION · SG EVENTOS · CLAUT · csoftmty · CVT · MHM MMASM · Incubadora de Iniciativas Emprendedoras Tecnológico de Monterrey Campus Monterrey · usaria · VIVRI · Festival Internacional de Sta. Lucía Monterrey, NL, Mx.

Special Purpose Languages

PARTE 7. LENGUAJES DE COMPUTACIÓN

—

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.

Ahora que ya cubrimos algunos fundamentos teóricos, podemos abordar aspectos más prácticos que nos ayuden en nuestro objetivo original de obtener elementos para desarrollar lenguajes propietarios de propósito particular que nos ayuden a incrementar nuestra productividad en el desarrollo de software. Doy la bienvenida a Aarón Moreno, con quien estaré escribiendo en los siguientes números.

Todos estamos familiarizados con el concepto de Lenguajes de Programación (LPs) y existe bastate literatura sobre el tema (pueden profundizar en mucho de lo que veremos aquí en cualquier libro sobre "Lenguajes de Programación"). Sin embargo, si buscamos incrementar nuestra productividad deberíamos abrir nuestra panorámica y considerar no solo los LPs, sino lo que llamaremos aquí "Lenguajes de Computación" (LCs), concepto que incluye los LPs, pero también los Lenguajes de Documentación (como LaTeX y HTML), de Arquitectura (como ABACUS y CBabel), de Especificación (v.gr. de sanners/parsers como lex/YACC), de Documentación de Procesos (como XPDL y JPDL), y para la Prueba (como LoadRunner y Selenium), entre otros; desarrollar lenguajes propietarios para estas áreas también puede incrementar nuestra productividad. (El tema de este número es "El auto como plataforma de software"; el pasado marzo estuvimos en el CeBIT –exposición tecnológica en Alemania, tal vez la más grande del mundo–, y en ella se mostraron avances en el sector automotriz que por supuesto implican LCs de propósito particular para ese sector.)

Los LCs (y los LPs en particular) se desarrollan a lo largo de un proceso de sistematización–formalización–automatización durante el cual se va conociendo mejor el área para la cual se diseña el lenguaje (el "Application Domain").

Procedamos a abordar nuestro tema apoyándonos en el desarrollo histórico de los LPs.

LENGUAJES DE PROGRAMACIÓN (LPS)

Cuando los LPs son de propósito general (general purpose languages) tienen constructos que son comunes a muchos otros LPs, tales como mecanismos de secuenciación, alternación y repetición de instrucciones, definidos en el marco de lo que llamamos un Sistema de Control (*control system*); y mecanismos de composición de tipos, definidos en el marco de un Sistema de Tipos (*type system*). Cuando los LPs son de propósito particular (special purpose languages) tienen además una parte diseñada especialmente para abordar más efectivamente problemas del Dominio de Aplicación, como tipos de datos y estructuras de control ad hoc (v.gr. si el dominio fuera la Matemática, el LP podría tener construcciones y operaciones sobre matrices, entre otras cosas).

Hoy tenemos LPs incluso para programar videojuegos. Sin embargo, la inquietud que nos llevó a tener LPs no era generar una industria de entretenimiento, ni siquiera una industria de software. Los trabajos sobre los que finalmente se basa esta tecnología fueron principalmente de lógicos-matemáticos-filósofos que querían construir máquinas que pensaran... o "calcularan" (en el sentido lógico-matemático) como decía Leibniz (que por cierto, es lo que aún se busca, varios siglos después).

EL SISTEMA DE CONTROL

La estructura del sistema de control está fuertemente influenciada por el paradigma con el que fue desarrollado el LP (ver sección "Paradigmas" más adelante), pero podemos hablar en general de constructos (o "abstracciones") como los descritos en la tabla 1.

Abstracciones de control		
Primitivas	**Compuestas**	**De Unidad**
Instrucciones simples como: goto, return, asignaciones, llamadas a subrutinas o macros	Instrucciones definidas en términos de otras instrucciones, en particular para construir: Alternación (if, switch), Repetición (while, for), Secuenciación (begin-end), Subrutinas (procedimientos, funciones	Mecanismos para organizar código en distintos archivos (include, uses, require), bibliotecas, abstract data types.

Tabla 1. Abstracciones de control

Abstracciones de datos		
Primitivas	**Compuestas**	**De Unidad**
Datos básicos (y predefinidos) como: ordinales (char, int, boolean), contínuos (real), apuntadores, tipo nulo, rangos, archivos.	Datos definidos en términos de otros tipos: registros, conjuntos, secuencias, tipos no lineales (árboles, grafos), clases.	Mecanismos para almacenar segmentos de código con declaraciones de datos y tipos de dato.

Tabla 2. Abstracciones de datos

EL SISTEMA DE TIPOS

La estructura del sistema de tipos también está influenciada por el paradigma del LP, pero podemos igualmente hablar en general de construcciones (o "abstracciones") como las descritas en la tabla 2. El diseño del sistema de tipos es una actividad en sí misma, dada su complejidad y relevancia ya que define elementos vitales de la semántica del lenguaje.

La definición del sistema de tipos incluye cuestiones muy importantes como las siguientes:

a) Chequeo de tipos: Distinguimos entre strongly-typed y weakly-typed languages, dependiendo de la menor o mayor tolerancia a que a las variables de un programa no se les haya asociado explícitamente un tipo de dato. Si el chequeo de esta asociación se realiza en tiempo de compilación hablamos de statically-checked languages; si ocurre en tiempo de ejecución hablamos de dynamically-checked languages.

b) Equivalencia de tipos: Se definen criterios para determinar cuándo se considera que dos variables tienen el mismo tipo, aunque eventualmente hayan sido declaradas de manera distinta. Algunas alternativas son:
- Declaration equivalence: dos variables tienen tipos equivalentes cuando remiten a la misma declaración de tipo.
- Name equivalence: dos variables tienen tipos equivalentes si fueron declaradas usando el mismo nombre de tipo en el mismo ámbito (scope).
- Structural equivalence: dos variables tienen tipos equivalentes si fueron definidas partiendo de los mismos tipos básicos y utilizando los mismos constructores de tipos.

c) Conversión de tipos: Para combinar variables de varios tipos se aplican mecanismos como los siguientes:
- Conversión implícita (promotion): como cuando una variable entera es convertida automáticamente a real por aparecer en una expresión real.
- Conversión explícita: la que realiza el programador mediante la utilización de funciones predefinidas, o facilidades del lenguaje para "asignar tipos" (casting).
- Generalización de un tipo de una variable a un tipo genérico, como el void del lenguaje C.

d) Inferencia de Tipos: Permite deducir el tipo adecuado de una subexpresión dentro de una expresión. Es particularmente importante en los weakly-typed languages.

CLASIFICACIÓN DE LENGUAJES DE PROGRAMACIÓN

Para clasificar los lenguajes podríamos basarnos en distintos criterios:

1. Su aspecto ante el programador, por ejemplo lenguajes visuales y textuales.
2. Su "composición estructural" (y la complejidad de procesar cada lenguaje), lo cual es el origen de jerarquías como la Jerarquía de Lenguajes de Chomsky.
3. El "marco conceptual" detrás de los programas que se escriben en un lenguaje particular, lo que facilita la clasificación en Paradigmas de LPs.
4. El "origen cronológico" en el que fueron creados, lo que da pie al concepto de Generaciones de LPs.

Vamos a poner en perspectiva algunos LPs relevantes considerando los 2 últimos criterios..

PARADIGMAS

La primera división usual en términos de paradigmas es la distinción entre lenguajes procedurales y no-procedurales. En los primeros, el programador debe especificar con precisión cómo debe ejecutarse la solución; en los segundos se busca que más bien se especifique qué es lo que debe solucionarse.

Dentro de los **lenguajes procedurales** tenemos los lenguajes no-estructurados, estructurados, y orientados a objetos:

- En los lenguajes no-estructurados el programador tenía amplia libertad para definir datos y manipularlos como le pareciera mejor; incluso podía saltar del interior de un bloque hacia otro punto en el interior de otro bloque (los famosos *goto*). Esta falta de estructura (que generaba lo que suele llamarse "código spaghetti") causaba grandes problemas para entender y mantener el software.

- Los lenguajes estructurados ofrecen un sistema de tipos con tipos básicos y constructores para especificar tipos compuestos a partir de los básicos. Ofrecen también mecanismos para agrupar bloques de instrucciones que pueden ejecutarse solo de una manera: comenzando en su único punto de inicio y terminando en su único punto de cierre (*single-entry, single-exit*).

- En los lenguajes orientados a objetos ya no se especifica solamente la definición de datos estructurados, sino que se vinculan esos datos con las operaciones válidas aplicables a los mismos. Ya no se puede entonces aplicar una operación

cualquiera sobre los datos, sino que debe ser una de las explícitamente vinculadas a éstos.

Los lenguajes no-procedurales, por su parte, suelen clasificarse en declarativos y funcionales. Suelen ser altamente expresivos, weakly-typed y dynamically-checked, y en ellos tanto los datos como los programas suelen tener una representación uniforme común, lo cual los hace aptos para el desarrollo de prototipos:

- En los lenguajes declarativos (o lógicos), se especifica qué debe hacer la computadora no através de instrucciones sino a través de sentencias lógicas que deben cumplirse tales como conjunciones, disyunciones, negaciones.

- En los lenguajes funcionales (o aplicativos), la especificación de la solución se hace en términos de funciones (asemejando las utilizadas en las matemáticas). Una representación común para datos y programas son las listas (como en LISP, lenguaje prototipotípico de este paradigma), y usualmente tienen la recursión como un mecanismo de repetición.

GENERACIONES

La primer generación se refiere al lenguaje máquina y ensamblador. Los primeros programas de software, que fueron escritos en lenguaje máquina. La inserción de una nueva instrucción podía requerir tener que revisar manualmente el programa completo, lo cual motivó el desarrollo de lenguajes "ensamblador", que hacían actualización automática de referencias y permitían utilizar nemónicos en lugar de códigos de operación de instrucciones.

La 2ª Generación la conforman los primeros lenguajes de alto nivel de propósito general (aunque prácticamente solo se usaban en la ciencia). Para desarrollarlos se habían detectado patrones de construcciones comunes, como la evaluación de expresiones aritmético-algebraicas, el llamado a subrutinas, y las instrucciones de alternación (como el "if") y de repetición (como el "while").

La 3ª Generación incluye lenguajes de propósito general de alto nivel estructurados, en los que solo se permiten instrucciones estructuradas (Single-Entry, Single-Exit) y no instrucciones goto. Para desarrollarlos se habían detectado también patrones en el uso de datos, que condujeron al desarrollo de Sistema de Tipos que incluían tipos básicos y constructores de tipos. Vino la introducción de módulos, que permitió "aislar" subrutinas y tipos de datos para impedir su modificación, y proveer solamente interfaces para el uso de los mismos a (el resto de) los programadores.

Con la 4ª Generación se asocian lenguajes de propósito particular (special purpose Languages), diseñados para realizar con facilidad tareas en un área de aplicación específica (como lenguajes para Manejadores de Bases de Datos (SQL) y para desarrollar Compiladores (lex/YACC)). Suelen tener constructos no-procedurales.

Con la 5ª Generación se asocian lenguajes de muy alto nivel (very high-level Languages), que fueron utilizados principalmente en la Inteligencia Artificial "clásica" (no en la "nouvelle AI" actual) para hacer por ejemplo Procesamiento de Lenguaje Natural o Razonamiento Automático, y para desarrollar Sistemas Expertos. Se trata de LPs escencialmente no-procedurales. En esta generación renació un interés por continuar los trabajos procedentes del Paradigma Funcional.

DESARROLLO HISTÓRICO

A continuación listamos algunos LPS que sentaron las bases para LPs modernos.

Plankalkül ("Cálculo de Planes"), es un lenguaje muy poco conocido pero lo mencionamos porque fue el primer LP de alto nivel. Fue desarrollado (1944) por el alemán Konrad Zuse para la Z3, primera computadora programable moderna, desarrollada también por él en 1941.

FORTRAN (FORmula TRANslation) fue desarrollado en IBM por un grupo encabezado por John Backus (1957). Uno de los criterios más importantes para su implantación fue la eficiencia del código generado por el compilador. FORTRAN introdujo, entre otras cosas, los arreglos, la instrucción de alternación "if", y los ciclos controlados por una variable indexada. Los programas debían escribirse con un formato léxico fijo (*fixed-format*), en el cual ciertas cosas tenían que escribirse en columnas preestablecidas.

LISP (LISt Processor) fue diseñado en el Massachussets Institute of Technology (MIT) por John McCarthy (1958). Se le considera el LP funcional por excelencia. Su diseño estuvo influenciado por los trabajos en el Cálculo Lambda de Alonzo Church. Introdujo conceptos como árboles, dynamic typing, funciones de órden superior, recursión, linked lists, recolección de basura, así como las S-Expressions (*symbolic expressions*), que son constructos para expresar tanto datos como programas, lo cual permite agregar nuevos elementos sintácticos al lenguaje.

COBOL (COmmon Business Oriented Language) fue desarrollado en el Departamento de Defensa de los EEUU (1960). Es quizás aún el lenguaje de programación más utilizado. "Introdujo" la idea de separar los datos del programa colocándolos en secciones diferentes, y el formateo de la impresión en pantalla usando pictures.

ALGOL-60 (ALGOrithmic Language) fue desarrollado por un comité (1960). Tuvo una enorme influencia en el desarrollo

posterior de los lenguajes de programación. Introdujo instrucciones estructuradas, declaraciones de tipos, paso de parámetros por valor, y fue el primer lenguaje en el que se utilizó BNF (Backus-Naur Form) para definir la sintaxis.

BASIC (Beginners All-purpose Symbolic Instruction Code) fue desarrollado por John G. Kemeny y Thomas E. Kurtz (1964), para hacer accesible la programación a estudiantes.

Simula (1960s) introdujo el concepto de clase y se le considera el primer lenguaje de programación orientado a objetos.

Pascal fue diseñado por el suizo Niklaus Wirth (1969), destilando ideas de ALGOL-60 en un lenguaje pequeño, simple y estructurado.

C fue desarrollado por Dennis Ritchie en los Bell Labs (1972), con el objetivo de crear un lenguaje sencillo, reduciendo la complejidad del sistema de tipos y el ambiente de tiempo de ejecución. Es muy portable y ofrece facilidades para acceder el hardware a bajo nivel.

PROLOG (PROgramming in LOGic) fue desarrollado por un grupo en Marsella dirigido por A. Colmerauer (comenzando en 1972). Se trata fundamentalmente de un demostrador de teoremas. Se ha utilizado para hacer procesamiento de lenguaje natural y razonamiento automático, y fue escogido por los japoneses para su Proyecto de 5ª Generación de Computadoras.

Scheme es un dialecto de LISP desarrollado en el MIT (1978).

Smalltalk fue desarrollado en Xerox Corporationc(1972-1980) para aplicar el enfoque orientado a objetos en una forma consistente. Se le considera el ejemplo más puro de un lenguaje orientado a objetos.

CRITERIOS DE DISEÑO

Cuando diseñamos un LC debemos considerar los objetivos que se persiguen con él. A continuación describimos algunos criterios comunes al diseñar LPs.

Simplicidad. La simplicidad facilita el uso del lenguaje; sin embargo, hay que considerar que "Everything should be made as simple as possible, but not simpler", pues la sobresimplificación le resta expresividad, y lo deja sujeto a muchas restricciones. Las características de generalidad, ortogonalidad y uniformidad conllevan frecuentemente a una disminución de la simplicidad. Ejemplos:

- BASIC: La falta de constructos fundamentales, como declaraciones y bloques, hace más difícil la programación de aplicaciones grandes.
- LISP, PROLOG: pocos constructos básicos facilitan las cosas, pero se depende de un sistema de ejecución complejo.

Legibilidad. Facilidad con la que alguien no-experto puede leer programas escritos en ese lenguaje y entenderlos.

Mantenibilidad. Facilidad con la que puede modificarse un programa, por ejemplo detectar y corregir errores, extender funcionalidad.

Expresividad. Facilidad con la cual, escribiendo poco código, se pueden expresar procesos y estructuras complejos. La expresividad puede entrar en conflicto con la simplicidad, por ejemplo PROLOG es muy expresivo, pero no simple. C también es muy expresivo pero no es de lo más sencillo.

Generalidad. Facilidad para combinar constructos estrechamente relacionados en uno solo más general. Por ejemplo, Pascal tiene declaración de procedimientos y paso de procedimientos como parámetros, pero no tiene variables que tomen procedimientos como valores.

Ortogonalidad. Permite que los constructos del lenguaje puedan ser combinados de cualquier manera que haga sentido, pero al mismo tiempo impide que la interacción entre diferentes constructos, o el contexto en que se están usando, causen restricciones o comportamientos inesperados o inadecuados. Por ejemplo, en C una función puede retornar valores de cualquier tipo de datos excepto arreglos.

Uniformidad. Consistencia en la apariencia y comportamiento de los constructos del lenguaje. Evitar que cosas no similares se vean o se comporten de manera similar, así como evitar que cosas similares se vean o comporten de manera distinta. Por ejemplo, en Pascal el constructo repeat-until abre y cierra un bloque de instrucciones, mientras que las instrucciones 'while' e 'if' requieren de pares 'begin' - 'end'.

Extensibilidad. Existencia de un mecanismo general que permita al programador añadir nuevas características al lenguaje; añadir nuevas palabras clave y constructos (como matrices, números complejos, etcétera). No se consideran extensiones la definición de nuevos tipos de datos, ni la definición de funciones nuevas en una librería.

¡Continuamos en la siguiente entrega de esta columna!

Aarón Moreno Monroy es Director de Operaciones y co-fundador de e-Quallity. Fue profesor de asignatura de la Universidad ITESO. Estudió su maestría en el CINVESTAV; su trabajo de tesis tuvo que ver con verificación formal, la cual aborda la prueba de software utilizando modelos matemáticos.

Cómo Ejecutar un Proyecto de Diseño de Experiencia

Por Misael León

Iniciar un proyecto de diseño de un producto digital es retador y muy emocionante. Sin embargo, también puede resultar abrumador cuando no sabes por dónde iniciar.

Esto me sucedía todo el tiempo al comienzo de mi carrera como UX Designer. Con el tiempo fui asentando una serie de pasos que, hasta la fecha, me ayudan a vencer esa ansiedad inicial producto de ver una hoja en blanco.

La figura 1 muestra una plantilla que utilizo para definir proyectos de UX. Esta no es una receta que asegura el éxito del proyecto, pero sí te facilitará la definición de los pasos iniciales. A continuación explico cómo llenarla. Esta plantilla está disponible para descarga en http://bit.ly/ux-blueprint

PROBLEMA SUPUESTO

Para diseñar una solución es obvio que primero hay que entender el problema. La única manera de lograrlo es realizando Investigación de Usuarios; recuerda que no hay UX si no involucras a los usuarios de alguna manera. En el número 54 de SG escribí sobre cómo hacer investigación de usuarios.

Para efectos del formato que exploramos en este artículo, comienza por definir tus supuestos sobre lo que sabes del problema. Enlista los hechos conocidos, los datos duros y las preguntas sobresalientes que quieres resolver.

A continuación muestro un ejemplo de cómo podríamos llenar el formato para el caso de una aplicación para llevar control de inventarios. Pondré uno o dos valores en cada campo, pero ya en la práctica se pueden poner más.

Problema Supuesto: Los usuarios no pueden completar el proceso de crear un reporte, debido al flujo complicado y la inconsistencia de patrones de diseño.

Datos duros: Se reciben 60 tickets mensuales de soporte solicitando ayuda para generar reportes.

Preguntas sobresalientes: ¿En qué parte específica del proceso de generar reportes los usuarios están teniendo dificultades?

ESCENARIO IDEAL

Aquí describimos cómo sería la aplicación y/o el producto idealmente y sin tomar en cuenta limitaciones. Como diseñador, a menudo conocemos la solución a estos problemas de antemano, tal vez no su ejecución a detalle, pero sí sabemos nombrar la medicina que curará el mal conocido.

Siguiendo con nuestra aplicación de ejemplo, podemos determinar lo siguiente.

Escenario Ideal: Un usuario es capaz de identificar los movimientos en su inventario y fácilmente generar un reporte mensual o anual. Puede descargar el reporte y/o enviarlo a su supervisor o colegas.

Elementos necesarios para que se cumpla este escenario ideal:

- Patrones de diseño consistentes (estilos, íconos, componentes, etc.)
- Flujo de tarea consistente y con un reducido número de pasos
- Conocimiento de problemas específicos de usabilidad al generar reportes

A grandes rasgos la solución al problema planteado son esos tres conceptos. Dado que no podemos arreglar todo al mismo tiempo, debemos enfocar nuestra solución a algo medible. Lo que haremos es formular una hipótesis que te ayudará a medir el éxito de tu proyecto.

Hipótesis: Creemos que si consolidamos el flujo de tarea de generación de reportes lograremos reducir los tickets mensuales de soporte en un 75%

ACTIVIDADES DE UX

Aquí ponemos las actividades concretas que llevaremos a cabo para lograr el escenario ideal. Estas actividades serán el centro de nuestro proyecto de UX.

Ejemplo:

- Pruebas de Usabilidad con 5 usuarios.
- Entrevistas contextuales a 5 usuarios para entender hábitos de uso.
- Creación de Wireframes para representar el flujo ideal.

Estos conceptos son las técnicas de UX en sí mismas. Existe abundante literatura que describe cómo realizar cada una de ellas. Al final de cuentas, su realización depende del tiempo disponible que tengas para realizarlas.

OBJETIVOS

Es importante que definas el objetivo que deseas lograr con cada una de las actividades de UX que te propusiste ejecutar. Debes entender perfectamente lo que quieres lograr para que puedas identificar cuando ya lo has logrado. Los objetivos deben ser lo más claros posible en términos de lo que obtendrás y cómo resuelven el problema.

Ejemplo:

- Identificar el punto crítico donde la usabilidad se rompe al momento de generar un reporte.
- Determinar cómo el contexto de uso influye en la ejecución de la tarea y detectar si existe un feature gap en la aplicación.
- Consolidar el flujo de creación de reportes con una librería de patrones que permita la ejecución de la tarea en 3 pasos.

Redactar los objetivos de esta manera permite comenzar a visualizar los entregables y documentación necesaria que deberemos crear en los siguientes pasos.

TAREAS

Aquí listamos cada una de las tareas operativas que debemos realizar para completar

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.

las actividades de UX y alcanzar los objetivos que nos planteamos. Este es nuestro plan de acción para los siguientes días. Te aconsejo que lo rompas en hitos fáciles de identificar y nombrar; esto para que sepas en cual etapa del proyecto te encuentras en determinado momento.

Ejemplo:

1. P*lanear Pruebas de Usabilidad*
   - *a. Definir las tareas a probar*
   - *b. Definir perfil de usuarios*
   - *c. Reclutar usuarios*
   - *d. Poner citas con usuarios*
   - *e. Interpretar resultados*
   - *f. Realizar reporte de usabilidad*
2. *Consolidación de flujos*
   - *a. Mapear el flujo existente*
   - *b. Bocetar flujos ideales*
   - *c. Creación de mapa de arquitectura de información ideal*
3. *Creación de librería de patrones*
   Diseñar flujos mejorados, etc.

Estas tareas pueden vaciarse en un calendario de actividades o un listado con checkboxes que podrás tachar, dependiendo de tu preferencia.

Considera que en la práctica varios factores te sacarán de tu proyección ideal de tiempo, especialmente cuando entra en juego la disponibilidad de los usuarios para las entrevistas y pruebas de usabilidad. Te aconsejo que agendes estas actividades tan pronto como tengas un perfil definido. No importa que no tengas las tareas definidas todavía, esas las tendrás listas a tiempo. Lo que quieres evitar es retrasarte cuando los usuarios no te contesten o no tengan disponibilidad.

ENTREGABLES

Aquí debes definir el tipo de entregables que presentarás. Es vital tener claro cómo se verán los resultados de tu trabajo para poder comunicarlos.

Cada audiencia es distinta y debes crear lo mínimo necesario para cada una. A continuación explico ejemplos de entregables adecuados para cada audiencia:

• **Equipo de desarrollo:** Un prototipo con las especificaciones de diseño, en estos días el puente entre desarrollo y diseño es muy estrecho. Esto es debido a herramientas como InVision y Zeplin que proporcionan a los desarrolladores medidas exactas, clases de estilo ya definidas, librería de assets, etc. Procura describir a detalle cada paso de los nuevos flujos de tareas con todos los escenarios posibles.

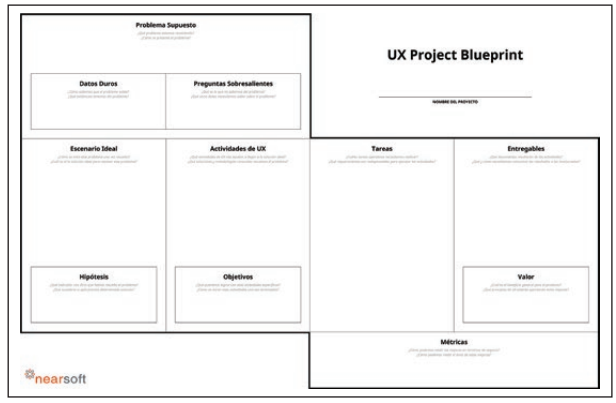


Figura 1. Plantilla para proyectos UX

• **Ejecutivos:** Un reporte de usabilidad, un reporte breve sobre tus hallazgos de investigación, un prototipo que se pueda navegar con clicks, de tal manera que se los acerque visualmente a la solución que estás proponiendo.

• **Product managers:** Un reporte más completo sobre tus hallazgos, un documento que muestre paso a paso los flujos propuestos, evidencias de la investigación, como fragmentos de videos de las pruebas o un compilado de frases dichas por los usuarios. Lo que necesitas es convencerlos de los problemas que encontraste y los fundamentos de las soluciones que estás proponiendo.

Suponiendo que trabajas bajo filosofía ágil, te recomiendo que seas breve con tu documentación pero que tengas a la mano evidencias de tu proceso de trabajo. Esto más que nada es para que en el caso de ser necesario se pueda entender la justificación detrás de las decisiones.

Un diseñador no debe, bajo ninguna circunstancia, titubear sobre sus propuestas. Es para ello que juntamos evidencia y hablamos con usuarios directamente. Para solucionar un problema primero debemos entenderlo.

VALOR

En este apartado describe brevemente cómo es que este proyecto de UX añade valor al producto. Piensa de manera abstracta, esto te ayudará a no desviarte del camino. Describir el valor esperado te permitirá entender exactamente lo que se logrará con los cambios propuestos.

Ejemplo:

- *Consolidar flujo de tareas para la creación de reportes.*

- *Consistencia de patrones de diseño.*

MÉTRICAS

Como en todo proyecto, es importante medir de manera concreta el alcance y los beneficios que obtendrás al final de su ejecución. En el terreno de UX es crucial demostrar el valor y el retorno de inversión (ROI). Recuerda que al hablar de UX estamos hablando del negocio desde la perspectiva del usuario. La gente simplemente no pagará por un producto que no se apegue a su estilo de vida y hábitos de uso.

Ejemplo:

- *Disminuir en un 75% los tickets mensuales de soporte relativos a la generación de reportes.*

CONCLUSIÓN

Con este formato tendrás una visualización completa de tu proyecto. Desde la identificación del problema, la formulación de un comprobable solución hipotética, detalles de su ejecución y de los entregables (también en términos del valor que brinda al negocio), así como de la medición de su efectividad.

A pesar que puede ser un documento exhaustivo, una vez que lo dominas no te llevará más de 30 minutos completarlo. En mi experiencia me funciona perfecto cuando lo lleno a mano. Es mi momento de poner mis ideas en orden y pensar sobre lo que haré en las próximas semanas. Espero te sea de utilidad.

El Diseño de la Experiencia de Usuario es un área que requiere mucha tenacidad de tu parte. Debes lograr el balance ideal entre tecnología, el negocio y el aspecto humano. No es un trabajo sencillo pero sin duda es muy gratificante. Estás logrando que la tecnología sea más amigable. Enhorabuena por ti y tus proyectos.

Exportando Sistemas Relacionales a **Sistemas Columnares Distribuidos**

Por Hugo de la Mora

El manejo de información masiva se ha hecho cotidiano, y una estrategia común para analizar grandes cantidades de datos es moverla de las bases de datos relacionales tradicionales (RDBMS) hacia bases de datos columnares distribuidas.

En el presente artículo muestro los pasos necesarios para migrar información relacional a una infraestructura de Hive en Hadoop. La idea básica es importar grandes datos de un origen relacional como Microsoft SQL Server a un almacenamiento en Hadoop como Hive o HDFS, teniendo la posibilidad de analizar esa información a través de Hue.

PANORAMA

Para esta labor utilizaremos Apache Sqoop [1], una herramienta precisamente creada para mover datos entre bases de datos estructuradas y Hadoop. Sqoop trabaja con tablas individuales, cada fila de una tabla se trata como un registro en HDFS. Todos los registros se almacenan como datos de texto en archivos de texto, en Hive o HBase (ver figura 1).

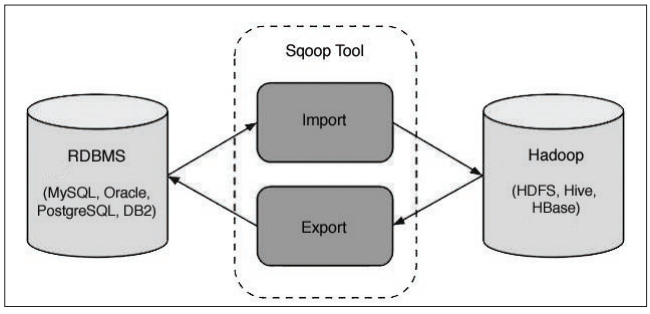


Figura 1. Flujo de Sqoop

PRERREQUISITOS

Para el propósito de este tutorial voy a suponer que ya se tiene instalado y funcionando tanto un ambiente de SQL Server como otro de Hadoop. Si no tienes el de Hadoop, para propósito de aprendizaje puedes utilizar una máquina virtual de Cloudera Quickstart [2], que está disponible como máquina virtual para distintos ambientes.

En tu ambiente de Hadoop necesitas tener instalado Sqoop. Checa si ya tienes, invocando el comando "sqoop" desde la línea de comandos. Si no lo tienes, lo más conveniente es instalarlo por medio del manejador de paquetes de tu sistema operativo (en el caso de OSX está disponible para Homebrew).

PREPARANDO LA CONEXIÓN

Nos conectaremos a SQL Server por medio de JDBC. Para que esto funcione primero necesitamos habilitar TCP/IP en nuestra instancia. Esto se hace desde el SQL Server Configuration Manager.

Posteriormente vamos a descargar el driver de JDBC de SQL Server en nuestro ambiente Hadoop. Este se obtiene del sitio web de Microsoft y al momento de escribir este artículo la versión más reciente se obtiene en http://swgu.ru/t3 . Dentro del paquete encontraremos una carpeta jre7 y otra jre8 y dentro de cada una hay un archivo .jar que corresponde al driver. Así que seleccionaremos el archivo que corresponda a la versión de java que estemos usando y lo copiaremos en /var/lib/sqoop . Si esta carpeta no existe, la creamos y le asignamos los permisos necesarios como se ve a continuación.

```
$ mkdir -p /var/lib/sqoop$ chown sqoop:sqoop /var/lib/sqoop
$ chmod 755 /var/lib/sqoop
$ sudo cp sqljdbc_6.0/enu/jre8/sqljdbc42.jar /var/lib/sqoop/
```

IMPORTANDO DATOS

Para verificar que podemos conectarnos de nuestro ambiente Hadoop a SQL Server vamos a comenzar listando las bases de datos. Para ello utilizamos sqoop list-databases con los argumentos de conexión requeridos (en este ejemplo nuestro SQL Server está disponible en la ip 192.168.56.1 y exponiendo el puerto 1444, y tiene un usuario hadoopUser con contraseña hadoop...).

```
$ sqoop list-databases --connect jdbc:sqlserver://192.168.56.1:1444 --username hadoopUser --password hadoop...
```

Con lo anterior obtendremos una lista de las bases de datos. A continuación importamos la base de datos y tablas que queremos (en este caso la tabla employee de la base de datos sqoopDB).

```
$ sqoop import --connect "jdbc:sqlserver://192.168.56.1:1444;username=hadoopUser; password=hadoop...; database=sqoopDB" --table employee --hive-import
```

Por cierto, si quisieramos importar todas las tablas de la BD sin necesidad de especificarlas una por una utilizaríamos
sqoop import-all-tables –connect "jdbc://sqlserver...." --hive-import

Si quieres revisar lo que se importó a Hadoop, puedes hacerlo por medio Hue, una interfaz gráfica web para navegar Hadoop (ver figura 2).

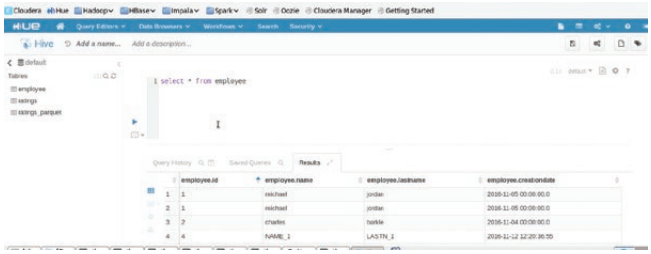


Figura 2. Consultar los datos importados en Hue

HAGÁMOSLO REPETIBLE

Si quisieramos estar realizando esta tarea frecuentemente, lo más conveniente es crear un "job" en sqoop que simplemente podemos invocar. Para ello usamos sqoop job con el argumento --create y el resto de la información para conectarse a SQL Server.

```
$ sqoop job --create jobImpEmpl -- import --connect "jdbc:sqlserver://192.168.56.1:1444;username=hadoopUser;password=hadoop...;database=sqoopDB" --table employee --hive-import
```

Lo más probable es que no querramos importar todos los datos cada vez que ejecutamos la tarea, sino que solo querramos importar los datos nuevos o modificados. Para lograr esto usamos la opción --incremental e indicamos la columna con el timestamp a verificar.

```
$ sqoop job --create jobImpEmplInc -- import --connect "jdbc:sqlserver://192.168.56.1:1444;username=hadoopUser;password=hadoop...;database=sqoopDB" --table employee --incremental lastmodified -check-column creationdate --hive-import
```

Ya creamos los jobs, pero todavía no los ejecutamos. Para ejecutar un job utilizamos `sqoop job --execute <nombre-del-job>`. Para ver los jobs que tenemos definidos podemos usar `sqoop job --list`. Y si queremos consultar los detalles de algún job usamos `sqoop job --show <nombre-del-job>`.

EJECUCIÓN PERIÓDICA AUTOMATIZADA

Si deseamos estar importando datos de manera periódica y automatizada, lo más conveniente es programar la tarea por medio de Oozie, que es una herramienta del ecosistema Hadoop que permite gestionar workflows desde una interfaz web.

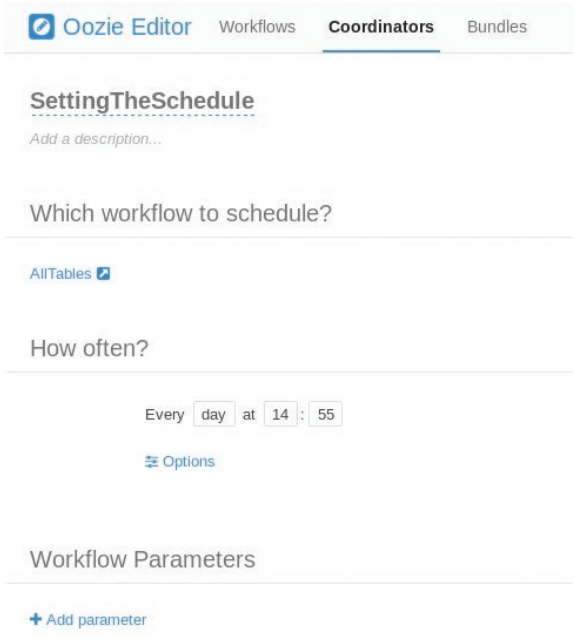


Figura 3. Coordinar workflows en Oozie

Oozie tiene un editor en el cual defines el workflow. Si ya tienes el comando de sqoop que deseas ejecutar simplemente puedes "pegarlo" en un campo de texto al crear tu workflow. Después de eso podemos crear "coordinadores" para calendarizar el envío de información al sistema HDFS (ver figura 2).

CONCLUSIÓN

Se ha mostrado de una forma sencilla la extracción de datos desde un popular ambiente de bases de datos relacionales hasta un sistema columnar como Hive.

Referencias

[1] http://sqoop.apache.org
[2] https://www.cloudera.com/developers/get-started-with-hadoop-tutorial.html

El MSC Hugo Enrique De La Mora Carrillo (@hugodelamora) es Chief Information Security Officer en CONTPAQi. Anteriormente ha trabajado con sistemas distribuidos, big data, desarrollo para cloud y móvil.

Hacer Ágil vs **ser Ágil**

Por Jorge Heras

A pesar de que este tema pareciera ser un poco repetido, sigo encontrándome con empresas y personas con serias complicaciones en su proceso hacia la agilidad. Dichas empresas tienen en curso esfuerzos e inversiones importantes en capacitación e implementación de metodologías y prácticas Ágiles como reuniones diarias, iteraciones de dos semanas, reuniones de retrospectiva, etcétera. Sin embargo, muchas de estas se quedan a la mitad del camino y otras terminan regresando a las formas de trabajo tradicionales, sin alcanzar a ver los beneficios que se esperan; en otros casos escucho que "X" empresa es ágil ya que siguen algunas metodologías como Scrum o Kanban pero tienen empleados y clientes descontentos, y en muy pocas ocasiones me encuentro con iniciativas que apoyen y faciliten la transición hacia una forma de trabajo diferente y sus beneficios correspondientes.

LAS BASES

Me gusta mucho iniciar talleres o conferencias con la siguiente pregunta: ¿Qué es Ágil? Y la respuesta más común es: "es una metodología". Si regresamos a las bases, el manifiesto de desarrollo de software ágil está conformado por cuatro principios y doce valores, no contiene procesos ni métodos que lo constituyan como una metodología. Yo prefiero interpretarlo como una filosofía de trabajo, incluso me atrevo a describirlo como una forma de vida que nos permite enfrentar la incertidumbre y complejidad que nos rodean hoy en día. Entonces, si ágil es interpretado como una metodología, lo que vamos a tratar de implementar son prácticas, procesos y métodos (hacer ágil); en cambio, si lo pensamos como una filosofía vamos a intentar vivir sus principios y valores, lo que requiere de un "mindset", una forma particular de pensar que se deriva en comportamientos (ser ágil).

HACER ÁGIL

Cuando implementamos métodos y prácticas ágiles estamos "haciendo" algo para lograr la agilidad. Esto nos ayuda a tener los eventos artefactos y procesos adecuados para entregar valor de forma continua con un buen nivel de calidad, gracias a esto podemos tener algunos beneficios como: adaptación a los cambios, mejorar la visibilidad, incrementar la productividad, mejorar la calidad y reducir los riesgos. Un buen programa de entrenamiento nos puede ayudar a alcanzar este estado.

SER ÁGIL

Cuando además de implementar prácticas adoptamos una forma diferente de pensar estamos "siendo" alguien distinto, alguien que vive la agilidad y cuyo comportamiento está alineado a los valores y principios ágiles, esto nos da beneficios adicionales como: deleite del cliente, placer por el trabajo, compromiso, innovación, creatividad y aprendizaje continuo. Cuando hablamos de ser ágil a nivel organizacional nos referimos a empresas que adoptan una cultura que permite obtener estos beneficios mediante un cambio de mentalidad en todos los niveles de liderazgo, no solo de los equipos de trabajo.

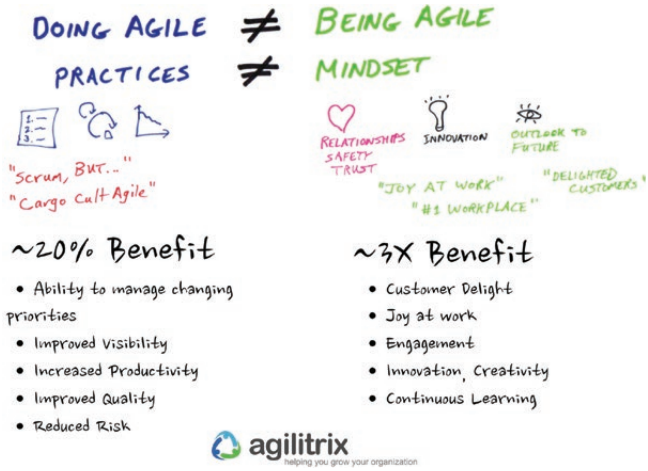


Figura 1. Doing Agile vs Being Agile [1]

La figura 1, obtenida del artículo "Doing Agile vs. Being Agile" de Michael Sahota ejemplifica la diferencia entre hacer ágil y ser ágil.

En mi experiencia las mejores transiciones hacia la agilidad se logran cuando además de la adopción de prácticas ágiles también se adopta el mindset Ágil, esto definitivamente es mucho más complejo, ya que un entrenamiento no será suficiente, se necesita de un acompañamiento adecuado, de preferencia de un coach experimentado que pueda ayudar a lograr una transición cultural.

CONCLUSIÓN

Hacer ágil es definitivamente distinto a ser ágil, y aunque ninguno de los dos es fácil no debemos perder de vista los objetivos que queremos alcanzar, si queremos el mayor de los beneficios debemos desafiar nuestras actuales formas de pensar y promover ese cambio alrededor de nosotros. Mi recomendación es solicitar apoyo de un experto, ya sea interno o externo, no solo en metodologías Ágiles sino también en el proceso de transformación hacia nuevos paradigmas.

Referencias

[1] 1. Sahota, M. Doing Agile vs. Being Agile. http://agilitrix.com/2016/04/doing-agile-vs-being-agile

Jorge Heras (@jherasq) es Ingeniero en Sistemas Computacionales con especialidad en Ingeniería de Software por el Instituto Tecnológico de Hermosillo y certificado como International Business Coach por la Academia Interamericana de Coaching. Actualmente es Director of Software Engineering en Tiempo Development, y Agile Coach de forma independiente.

La Cadena de **Destrucción Cibernética**

Por Alessandro Porro

El mundo de la seguridad informática utiliza una gran cantidad de conceptos que provienen del ámbito militar, no porque suenen bien sino porque hay muchas analogías interesantes que se pueden encontrar.

Uno de ellos es el concepto acuñado por Lockheed Martin de la cadena de destrucción cibernética: un modelo de inteligencia para la temprana detección, identificación y prevención de ataques. Esta cadena es uno de los métodos que se pueden utilizar para entender las intrusiones en la red.

INDICADORES

Antes de especificar lo que es exactamente la cadena de destrucción, tenemos que entender uno de los elementos fundamentales de la inteligencia: los indicadores. Hay tres tipos:

- Atómico (por ejemplo, direcciones IP o de correo electrónico)
- Computarizado (por ejemplo, hash de archivos)
- Conductual (colecciones de indicadores computarizados y atómicos, que a menudo describen los diferentes pasos en una parte de la intrusión).

Los indicadores son los que se utilizan para detectar las diferentes fases de la cadena de destrucción.

LA CADENA DE DESTRUCCIÓN CIBERNÉTICA

La idea central de la cadena de destrucción es que un atacante debe reunir suficiente material para romper un entorno, mantener su punto de apoyo, y luego pasar a su objetivo final.

La cadena consiste en siete fases:

1. Reconocimiento. Investigación, identificación y selección de los blancos. Mucho de esto se puede hacer a través de fuentes públicas.

2. Militarización. Después de identificar una posible vulnerabilidad, el atacante construye (o adquiere) un malware bien elegido que puede explotar la vulnerabilidad

3. Entrega. Enviar el software malicioso a la víctima (por ejemplo, como un adjunto de correo electrónico).

4. Explotación. Ejecutar el código malicioso que fue enviado a la víctima.

5. Instalación. La instalación del código malicioso en el sistema de la víctima para que el atacante puede retener el acceso.

6. Comando y Control (C2): Cuando se instala el código malicioso, le tiene que informar al atacante que fue un éxito y esperar instrucciones.

7. Acciones sobre objetivos. Este es el objetivo final que el atacante quería lograr, por ejemplo, el robo de información.

MEDIDAS DE DEFENSA

Entonces, ¿Qué mecanismo de defensa se puede tomar para evitar a estos atacantes? En este modelo, el punto crucial es que la ruptura de cualquier paso rompe toda la cadena de destrucción, lo que significa que los atacantes tienen que pasar por todo el modelo nuevamente para tener éxito.

Para el reconocimiento, se pueden utilizar analíticas web y/o ingresar en una página de forense para la detección. Limitar la cantidad de información que se expone al público puede ayudar. No publicar el esquema de red interna es obvio, pero también se debe limitar la cantidad de información sobre el personal y procedimientos de trabajo. Poner reglas de firewall y controles de acceso es una obligación.

No hay mucho que se pueda hacer acerca de la militarización, ya que se produce en el mismo recinto del atacante, pero se pueden utilizar diferentes líneas de defensa para la entrega, la explotación, la instalación y el uso de fases C2.

La sensibilización de los usuarios (de vigilancia) y el filtrado de proxy puede impedir la entrega.

Las otras fases se pueden detener mediante el uso de (host-based) de detección de intrusos, sistemas antivirus, sistemas de aislamiento y filtrado de salida adecuado. Además, no se debe descuidar el uso de datos de inteligencia de amenazas disponibles para actualizar los dispositivos de filtrado e inspección. Y hacer una administración de registros adecuada debe ser parte de los procesos de TI.

EL CASO DEL MALWARE

Una de las críticas al modelo de destrucción cibernética es que está demasiado centrada en el malware. Este código maligno es sólo un posible medio de ataque, pero las amenazas actuales ahora también pueden implicar una amenaza interna, en la ingeniería social, o intrusiones basadas en el acceso remoto (por ejemplo, a través de un proveedor o credenciales capturadas).

Algunas de las fases se seguirán aplicando y pueden ayudar a prevenir o detectar incidentes. Pero a medida que los atacantes van cambiando sus métodos, todos debemos hacer lo mismo.

Alessandro Porro es Vicepresidente de Ventas de Ipswitch para Latinoamérica.

CDO: Se buscan líderes digitales

Por Olivia Salas

Una de las preguntas clave que la alta dirección se hace al emprender una iniciativa de transformación digital es: ¿quién debe liderar el viaje hacia la innovación? En muchas organizaciones, la respuesta se ha materializado en un nuevo rol corporativo: el director de Digitalización, o Chief Digital Officer (CDO), a quien se le encomienda concretar la evolución innovadora del negocio.

Por la sustancia digital del cargo, que exige una sólida competencia tecnológica en temas como IoT, Big Data, movilidad, entre otras, se asume que el CIO es un candidato ideal para dirigir una iniciativa de transformación digital. Sin embargo, de acuerdo con encuestas entre corporaciones globales, sólo el 35% de los CIO dirigirá la transformación digital de sus organizaciones. Estos especialistas también deben afrontar una pregunta, al parecer, sin respuesta sencilla: "¿Y por qué yo no?".

CDO: UNA EVOLUCIÓN POSIBLE PARA EL EXPERTO TI

En la elección de un líder para dirigir la transformación digital, según los expertos, la mayoría de los CIO pierde la oportunidad por la misma razón: no logran demostrar de forma contundente que su habilidad tecnológica está firmemente alineada a los valores de negocio. Por ejemplo, los CIO pueden proyectar un gran conocimiento sobre IoT, pero pierden confianza ante cuestionamientos que un CDO debería responder: ¿qué significará IoT para la experiencia del cliente?, ¿esta innovación generará nuevas oportunidades o líneas de negocio?, ¿tendrá un efecto negativo en una oferta?, ¿cómo influirá IoT en la estrategia de ventas y mercadotecnia?

En ese sentido, en el marco de una transformación digital, el CDO entiende qué necesita cambiar, por qué debe hacerlo y cuál será la repercusión de negocio, mientras que el CIO se concentra en cómo realizar la modificación.

No obstante, un líder del área de Tecnología o Sistemas puede tomar las riendas de una iniciativa digital si cambia su enfoque y desarrolla nuevas capacidades. Entre otros aspectos, debería poner atención a:

• Mejorar la comunicación con los líderes de negocio. En sus diálogos con otros gerentes y directivos es necesario reducir el uso de palabras como "algoritmo", "protocolo" o "zettabytes", y privilegiar conceptos como "trayecto del cliente", "ventas multicanal" y "experiencia del usuario", entre otros. Esto implica una nueva actitud y enfoque: no sólo pensar en el costo, la implementación y la gestión de la innovación, sino en cómo utilizarla para alcanzar metas de negocio que generan valor a la empresa.

• Definir nuevos parámetros de colaboración. Dado su foco y experiencia en la operación, los CIO tienden a hablar y pensar en escala de "proyectos". Hoy, la prioridad es interactuar con los distintos departamentos empresariales, entender sus objetivos y plantear "resultados de negocios". El CDO es un habilitador de capacidades, no un gestor de tareas. Los mejores CDO detectan las necesidades de las áreas corporativas, incluso antes de que éstas las reconozcan.

• Nueva escala de valores. En la dinámica de su equipo de trabajo original, el departamento de Sistemas o Tecnología, el CIO debe inculcar una cultura de la velocidad, la agilidad y el foco en el cliente. Esto creará una mentalidad "de CDO" entre los especialistas técnicos, lo que enriquecerá su visión de la empresa, la toma de decisión y el liderazgo.

• Colaboración multidisciplinaria. En su camino a la posición de CDO, un líder tecnológico hará bien en acostumbrarse a trabajar con especialistas en ventas, mercadotecnia, logística, administración, producción, redes sociales, atención al cliente, etc. Sumar la perspectiva de estos colaboradores, en las metas de TI, fortalece la capacidad de análisis y ejecución de un CDO.

Más allá de estos aspectos, para asumir el timón de una transformación digital, los CIO deben superar un gran desafío. En muchas ocasiones, abrumados por las tareas operativas de TI que deben atender todos los días, los CIO no tienen tiempo para desarrollar la visión, la creatividad y las capacidades que exige el rol de CDO.

En este aspecto,soluciones de software para infraestructura, automatización, desarrollo y gestión de nube entre mucha más, facilitan la administración y operación de la infraestructura de TI existente, y habilitan el camino a tecnologías emergentes. Ambas capacidades, permiten que el líder de TI se concentre en la transformación innovadora del negocio, y tome mucho menos tiempo en tareas operativas.

Un CIO que se compromete con una nueva visión y el desarrollo de nuevas habilidades, sin lugar a dudas, puede ponerse al frente de una transformación digital, la transformación de la organización hacia el futuro de un negocio que evoluciona.

Olivia Salas es Directora de Marketing de Red Hat México.

El Desarrollo de Software y la Dinámica del Miedo

Por Raúl Guerrero

El síntoma más cotidiano en los equipos de desarrollo de software con problemas de desempeño es el caos. Este caos frecuentemente tiene una causa raíz en común, para ilustrarlo imaginemos la siguiente historia ...

Todo inicia en la reunión de dos compañías, el cliente "Somos Grandes Inc" y un proveedor de TI "Monkeys Devs". En la reunión un gerente de línea de negocio quiere resolver un problema por medio de una solución tecnológica pero no sabe realmente cómo hacerlo, así que el director comercial de Monkeys Devs toma nota de la información que puede para entonces crear una propuesta comercial.

El director comercial está consciente de que la única forma de crear una propuesta adecuada es primero haciendo un análisis detallado, o proponiendo un modelo iterativo que permita ir descubriendo la solución por fases. Pero sabe que su principal competidor está dispuesto a dar una propuesta comercial sin dicho análisis, aun cuando el riesgo de desviarse sea amplio. Así que entrega una propuesta basada en supuestos.

Durante las juntas de revisión de la propuesta inicia el estire y afloje debido a que la propuesta rebasa el presupuesto del cliente, y al final, después de mucha magia y decisiones que desafían las leyes del tiempo y la materia, se aprueba el proyecto. Así que se da inicio al proyecto, y el equipo de trabajo no tiene acceso al contrato acordado (no vaya a ser que los desarrolladores se enteren cuánto se cobra por ellos), y por lo tanto no se tiene todo el detalle del compromiso adquirido.

Este sería un buen momento para hacer un alto en el camino y cuestionar todas las malas decisiones que se han ido tomando, tal como los cambios sin sentido en el plan de trabajo, o cotizar una tarifa fija sin tener un análisis, pero cortaríamos el suspenso de esta historia.

Una vez iniciado el proyecto, se inicia el análisis real. Se crea una nueva lista de requerimientos distinta a la del contrato, pero no se cuestiona porque los analistas no tienen acceso a éste. Se llenan documentos y minutas sin ton ni son y empieza la vorágine de los compromisos adquiridos durante las juntas, en donde el miedo a decir "eso no estaba en el alcance" hace su aparición. Por lo tanto, al final del análisis se cuenta con un nuevo compromiso.

Lo triste es que no importa cuánto te hayas tardado en el análisis, una vez terminado empiezan los cambios. Y por supuesto nadie quiere documentarlos; como si negar su existencia nos fuera a permitir ignorarlos. Así que empiezan los malos entendidos y juntas interminables en las que se habla de los cambios y todos se quedan callados; nadie quiere exponerse y entonces en muchas ocasiones se asume que el cambio se realizará.

La historia podría continuar y continuar, y si has llegado hasta esta parte de la historia es probable que recuerdes la sensación que tuviste alguna vez que te pasó lo mismo en un proyecto. Pero, ¿sabes algo? No estás solo, está es la vida de miles de personas que se dedican al desarrollo de aplicaciones. Y si nunca te ha pasado es probable que todavía no te haya tocado estar en un proyecto lo suficientemente complejo.

Ahora, si analizamos esta historia veremos que la mayoría de estas pesadillas se resuelven atendiendo la enfermedad.

¿Cuál es esa enfermedad? El miedo. Miedo a perder el proyecto por hacer el análisis, miedo a compartir el contrato con los desarrolladores, miedo a ser honestos y entregar el plan de trabajo con la estimación de los desarrolladores, miedo a administrar los cambios, miedo a reconocer que las cosas se tienen que hacer diferentes.

Es por ello que veo con esperanza cuando las compañías implementan prácticas ágiles y son capaces de sobreponerse al miedo, reconociendo que: el cliente quiere obtener valor continuo aún cuando no sabe de manera detallada lo que quiere, que es imposible planear de forma detallada lo que hará un desarrollador en 6 meses, ni siquiera se tiene dominio de las tecnologías requeridas para crear la solución. Es por ello que la entrega continua de compromisos de 3 a 4 semanas me parece tan adecuada y facilita las conversaciones cuando los cambios se solicitan, porque simplemente cada ciclo empezamos de nuevo.

La forma más tangible de encontrar la solución al problema es enfrentándonos nosotros mismos a romper el miedo de perder nuestra área de confort. Y en el camino encontraremos nuevos miedos y retos a enfrentar.

Así que estimado lector, ¿cuál es su primer miedo a romper?

Raúl Guerrero (@jrwarrior) es evangelizador técnico senior en Microsoft México. Cuenta con más de 20 años de experiencia profesional en el desarrollo de software y es reconocido por Software Guru como SG Rockstar.

Independencia en el Ciberespacio

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

Es de sobra conocido que la comunicación sobre redes de datos TCP/IP es fácil de espiar — El diseño de Internet desde sus inicios está enfocado a la confiabilidad, no a la privacidad, hecho que se hace obvio en sus protocolos a todos niveles. Esta es una realidad a la cual los usuarios de Internet nos hemos acostumbrado desde siempre.

La respuesta clásica se centró en el cifrado de las comunicaciones (usar secure shell en lugar de telnet, emplear https en vez de http, etc.) Esto, sí, lleva a que el contenido de las comunicaciones cuyos protocolos pueden ser cifrados se vuelvan inintelegibles para un adversario. Esta respuesta, tristemente, ha demostrado ser insuficiente por la profundidad y seriedad de cómo se ha configurado el modelo de adversario.

La sospecha de que las redes de telecomunicaciones (sean analógicas o digitales) sean utilizadas para el espionaje no es nada nuevo. A inicios de la década de los noventa, era frecuente que los jóvenes que enviábamos mensajes a través de las redes de *BBSes (Bulletin Board Systems*, sistemas de boletín electrónico, y de cierto modo antecesores de Internet para las comunicaciones personales) agregáramos al final de todos nuestros mensajes, como parte de nuestras firmas, líneas autogeneradas con palabras clave alineadas para la idiosincrasia contraterrorista del momento, por el estilo de «Bomb Lybia Arafat Plane Washington Kill President». Hacíamos esto, argumentábamos, porque el gobierno estadounidense analizaba todas nuestras conversaciones, primero efectuando un filtrado *grueso* y posteriormente haciendo trabajo a detalle; si incluíamos estos mensajes en todas nuestras comunicaciones, obligaríamos al filtrado grueso a reportar muchos falsos positivos, sobrecargando sus (mucho más valiosos) sistemas de filtrado fino. Hubo varios *nombres clave* que tuvo este espionaje en los años que a mí me tocó ser parte de la vanguardia que luchaba por proteger a nuestro amado ciberespacio, ese entorno que apasionadamente defendió John Perry Barlow (músico, poeta, ensayista, y uno de los primeros pensadores sociales que se dedicaron a analizar el significado de la red para la sociedad) en su *Declaración de Independencia del Ciberespacio* [1] en 1996. Cito el inicio de esta declaración, en traducción propia:

> *Gobiernos del Mundo Industrial, viejos gigantes de carne y acero, vengo del Ciberespacio, el nuevo hogar de la Mente. A nombre del futuro, les pido que nos dejen en paz. No son bienvenidos entre nosotros. No gozan de soberanía donde nos reunimos.*
>
> *No tenemos un gobierno electo, y es poco probable que lleguemos a tenerlo, así que me dirijo a ustedes sin más autoridad que aquella con que siempre habla la libertad misma. Declaro que el espacio social global que estamos construyendo es naturalmente independiente de las tiranías que ustedes buscan imponernos. No tienen derecho moral de gobernarnos, ni poseen método alguno de coerción que tengamos alguna razón para temer.*
>
> *Los gobiernos obtienen sus justos poderes del consentimiento de los gobernados. Ustedes no han ni solicitado ni recibido el nuestro. No los invitamos. No nos conocen, ni conocen nuestro mundo. El ciberespacio no yace dentro de sus fronteras. No piensen que pueden construirlo, cual si fuera un proyecto de construcción pública. No pueden. Es un acto de la naturaleza, y se crece a sí mismo mediante nuestras acciones colectivas.*

Hace veinte años, este escrito resonó muy fuertemente con todos nosotros, quienes llevábamos cinco, diez, veinte años creando nuevos medios de comunicación con la tecnología y los recursos que teníamos a nuestra disposición; la variedad tecnológica en esa época era impresionante, y había espacio para la participación de todo tipo de perfiles... Siempre alineados a lo que con el tiempo dio en conocerse como la *ética hacker.*

Pero me estoy *yendo por las rama*s hacia un tema del cual se puede hablar mucho más, y del cual disfruto mucho rememorar y escribir. A los interesados en esta temática, puedo recomendarles el libro publicado el año pasado por la Universidad del Claustro

de Sor Juana, «Ética hacker, seguridad y vigilancia» [2], en el que tuve el honor de participar con un capítulo.

Aterricemos: Hace veinte años, lo que sabíamos al respecto eran meras suposiciones. Claro está, las telecomunicaciones internacionales estaban a un nivel incomparable de donde están ahora; las llamadas telefónicas internacionales eran carísimas, y las comunicaciones digitales estaban al alcance de un porcentaje risible de la población. Si en esa época dedicáramos nuestro tiempo real de trabajo, o espacio en una revista como Software Gurú, a hablar de la vigilancia generalizada... Se nos tacharía indudablemente de "conspiranóicos".

Este tema no es del todo nuevo en nuestra revista: En esta misma columna, escribí ya al respecto en los números 32 (mayo de 2011) y 42 (noviembre de 2013).

Desde las primeras revelaciones de *Wikileaks*, parte muy importante de las filtraciones de alto nivel que se han presentado son relativas a la profundidad del espionaje que se realiza de forma indiscriminada. Vemos por un lado acuerdos entre agencias de seguridad de diferentes gobiernos y conjuntos de puertas traseras en software de uso generalizado que se mantienen por años ocultamente escondidos por las mismas, y por el otro la proliferación de técnicas analíticas a profundidad, sustentadas en el indefinible Big Data, en manos de particulares.

Cabe mencionar, los marcos regulatorios que norman el funcionamiento de las citadas agencias de seguridad nacional se mantienen firme y convenientemente anclados en su anticuado lugar, hace cuando menos veinte años. Por ejemplo, ante la restricción de que el gobierno estadounidense no debe espiar la comunicación entre sus ciudadanos, analizar conexiones de red que se mantengan dentro de sus fronteras se vuelve complicado. Bueno, recientemente salió a la luz que la NSA efectúa ataques de ruteo BGP (*Border Gateway Protocol*, empleado para el ruteo a gran escala, entre los llamados Sistemas Autónomos) — "Empuja" las rutas de algunos paquetes para pasar por conexiones fuera de los Estados Unidos, facilitando el marco regulatorio dentro del cual pueden operar. Esto, volviendo al punto mencionado al inicio de esta columna, es posible porque BGP está diseñado para confiar; cualquier *sistema autónomo* puede indicar que es mejor o peor candidato que sus vecinos para determinadas rutas.

Hasta cierto punto, la respuesta de la sociedad ha cambiado. ¿Para bien? No me atrevo a afirmarlo ni rechazarlo categóricamente: Ya no se burlan de quienes hablamos de un espionaje a gran escala etiquetándonos como conspiranóicos con sombreros de papel aluminio... Pero, bajo el credo de que "la privacidad ha muerto, ya supérenlo", se le resta importancia y se relega a segundo término. Después de todo, ¿por qué habríamos de escondernos de empresas que mediante la analítica de Big Data nos entregan la información que nos importa, así sea publicidad enfocada o noticias que encajan con nuestra ideología, buscando mantenernos como fieles seguidores de sus sitios? Y... ¿los gobiernos? Bueno, si no tengo nada que esconder, ¿qué tengo que temer?

Afortunadamente, existen muchas herramientas cuyo uso puede llevarnos a dejar un rastro mucho menos seguible en la red; la más popular de ellas tal vez sea la Red Tor, una serie de nodos provistos por voluntarios en todo el mundo que proveen una malla de anonimización mediante ocultamiento criptográfico. El funcionamiento de Tor es verdaderamente sencillo, y espero abordarlo en una posterior entrega, y parte importante del esfuerzo del proyecto que la sustenta se ha enfocado en llevar a la Red Tor a quien la necesite, desarrollando herramientas para su usabilidad sin requerir de avanzados conocimientos técnicos.

Sé que dejo esta columna aparentemente inconclusa. Estoy iniciando con un proyecto que me llevará a trabajar el tema a mayor profundidad; retomaré esta temática en la próxima edición, profundizando en Tor y en cómo los usuarios de distintos niveles técnicos pueden aprovecharlo.

No quiero cerrar esta entrega sin mencionar el caso de Dmitry Bogatov, un jóven maestro universitario de matemáticas, activista y desarrollador de software libre, participante al igual que yo del proyecto Debian. Dmitry fue acusado de publicar material subversivo extremista en Internet, aunque está demostrado que no pudo haberlo hecho; es muy creíble que su acusación esté más bien relacionada con que operaba un nodo de salida de Tor. Dmitry fue encarcelado el pasado diez de abril, y hasta el momento de entrega de mi columna, no ha habido avance real en su proceso. Desde esta modesta trinchera, y con la poca influencia que una publicación tecnológica en México pueda tener, los invito a visitar el sitio Web https://freebogatov.org/en/ para enterarse de más detalles del caso y, en caso de que lo juzguen correcto, apoyar al caso.

Referencias y notas

[1] https://www.eff.org/cyberspace-independence
[2] http://elclaustro.edu.mx/pdf/EticaHackerSeguridadVigilancia.pdf

Tips para Conseguir tu **Primera Chamba** como Programador

Por Hugo Hernández "Don Chambitas"

Comienzo haciéndoles saber que el presente escrito es una colaboración de ideas entre la banda de programadores con quienes más convivo recientemente y su compa el Chambitas.

SI ERES RECIÉN EGRESADO

Si recién egresaste este verano y aún no cuentas con experiencia dentro de algún proyecto real o empresa, ¡ponte a hacer algo ya! Y consigue dominar al menos las nociones básicas de tus tecnologías favoritas. El ser autodidacta es la cualidad más apreciada por las compañías de giro tecnológico.

Punto a tu favor si hablas inglés. Sé de empresas que cuando buscan aprendices prefieren entrenar a los candidatos en X tecnología, que enseñarles inglés.

SI VAS A HACER TUS PRÁCTICAS PROFESIONALES

Si estás por pasar a ese semestre donde ya al fin no tomarás clases, sino que debes presentar tus prácticas profesionales dentro de alguna organización; mi más sincera recomendación es que busques el lugar donde el tipo de #chamba que realices sea el que más se acerque al que te gustaría hacer al egresar. No te conformes con la promesa de que tu servicio de prácticas profesionales será aprobado a cambio de hacer actividades que no vayan con tu camino. Si quieres ser programador o programadora, ¡comienza ya!

SI TODAVÍA FALTA PARA GRADUARTE

Si aún estás a la mitad de tu ingeniería, o apenas iniciándola, estás a buen tiempo de comenzar a aplicar estos tips que algunos de los devs más chidos en México y un servidor les compartimos:

• Mismo consejo que en las primeras dos situaciones: ¡comienza a chambear en algo relacionado con programación ya! Y por "chambear" muchas veces pensamos en alguna actividad que nos dé dinero, si nos lo da, bienvenido. Pero si te invitan a colaborar en un proyecto cuyo objetivo sea aprender en equipo, lo cual también es chamba, siéntete una persona afortunada. Yo pensaría que si me invitan es porque confían en que me la rifo. En las primeras experiencias de chamba es precisamente la experiencia la mejor paga o retribución.

• Crea de una vez tu perfil en LinkedIn. Sí, para mí y sé que para varios reclutadores de diferentes países, LinkedIn es el nuevo currículum. Hasta nos la pusieron fácil, ¿para qué quebrarnos la cabeza pensando en el diseño que le daré a mi CV cuando desde años existe una plataforma que nos da un formato? Solo hay que capturar lo que hemos hecho y listo. Son unos 20 minutos lo que te tardas en hacer tu perfil en LinkedIn, pero neta que bien invertidos. Actualízalo de vez en cuando, ponle una foto amigable, ve agregando cada proyecto en el que participes así como las tecnologías que utilices, coméntale a tus profes que ya usas LinkedIn y que te gustaría agregarles para futuras recomendaciones/colaboraciones; conéctate con personas que tú consideres líderes en la industria pero hazlo inteligentemente: por ejemplo, pon una breve nota al agregar a alguien, hazle saber por qué lo quieres tener en tu red. Porfa que esa nota No sea un "busco chamba", mejor háblale de lo que sabes hacer. Aprende a venderte. Si al egresar ya tienes tu perfil en LinkedIn bien armado y con contactos de tu industria, podrías tener la chamba que quieras. Va de nuez: más si hablas inglés.

• Pero aguanta, el conocimiento real de lo que hace un programador se encuentra en Github. No necesitas ser un experto en la programada para estar en Github, al contrario, ahí le entrarás a aprender de código y mejores prácticas por parte de otros programadores con más tiempo de experiencia que tú. Así que arma de una vez tu perfil en esta plataforma y colabora en los proyectos que encuentres interesantes. Crea tus propios repositorios. ¿No sabes lo que es Git? Es un sistema de control de versiones, como mercurial, subversion, etc. pero que a diferencia de los demás modela los datos como un conjunto de instancias de un mini sistema de archivos. Algunos líderes de la industria que conozco opinan que si conocen a un programador que no sabe cómo utilizar Git les hace pensar que a esta persona no le nace estar actualizada o no tiene iniciativa. Y entonces, ¿qué hago estudiando una ingeniería relacionada con tecnología si no estoy informado de las

actualizaciones de mis futuras herramientas de chamba?

• Y hablando de tecnologías de las chidas, prueba Linux. El involucrarse en temas open source te dará una amplia perspectiva de lo que es la industria del desarrollo de software, pues según estadísticas, el 86% de los servidores en el mundo manejan Linux. Y lo más importante, Linux te abre la mente a una manera más libre de trabajar con sistemas operativos.

• Haz una lista de las compañías TI en las que te gustaría chambear e investiga qué tecnologías utilizan en sus proyectos (así como cuáles developers chambean ahí). Si no sabes nada sobre las herramientas que se utilizan, no te agüites, si realmente quieres ser parte de ese equipo es mejor que comiences a ponerlas en práctica. Al terminar esta lista contarás al menos con un temario por estudiar. Recuerda estar investigando sobre las tecnologías de tus empresas favoritas al menos una vez cada seis meses, los lenguajes y frameworks que se utilizan al día de hoy en los proyectos de desarrollo muy probablemente hayan cambiado o se hayan actualizado en el siguiente año.

• Consíguete un mentor. ¿A poco no estaría bien chido contar con alguien en tu día a día que haya recorrido un camino similar al que te gustaría recorrer? Un mentor es una guía, una persona quien te compartirá su historia para que de la misma tomes lo que consideres positivo para la tuya. Yo lo veo como una persona que enfocará tu camino hacia rumbos que, con base en su experiencia, considere los más apropiados para tu desarrollo profesional. Por lo regular un mentor es una persona que te inspira. ¿Y dónde podría encontrar personas de las cuales inspirarme? Chécate el siguiente y último tip.

• Mi consejo más valioso es: únete a la comunidad de programadores de tu localidad. Si todavía no existe alguna, ¿qué esperas para comenzarla? Para arrancar una comunidad, creo yo, se requieren de estas dos cosas: ganas de compartir tu conocimiento con #LaBanda, y disposición a invertir tu tiempo. Si por tus tiempos de la escuela y la chamba no puedes formar parte activa y voluntariamente de una comunidad, al menos asiste a los eventos mensuales que las personas voluntarias de tu comunidad local organizan. La mayoría de estos eventos son hospedados por las mismas empresas de TI de tu región y son promovidos mediante la plataforma de meetup.com, ¡únete ahora! El asistir a estos meetups te llenará de conocimiento por parte de los actores que participan como conferencistas, mismos que muchas veces residen en la ciudad; y pondrá a prueba la oportunidad de desarrollar tus habilidades sociales en un ambiente informal y relajado. Es como exponer tu currículum pero en vivo y sin filtros.

En conclusión, me atrevo a decir que las llamadas "social skills" (habilidades sociales) son algo en que se fijan mucho las empresas. Desde integrarte a un proyecto de desarrollo entre tus compas, o unirte a la comunidad de devs de tu localidad, hasta crear tu perfil en LinkedIn o Github, todas estas chambas tienen una base social; la diferencia es que una es en vivo y la otra en línea.

Es de hecho una de las características más chidas de nosotros los seres vivos: nuestra capacidad de relacionarnos y en el mejor de los casos, entendernos. Las empresas buscan seres humanos no androides. Si algo me he dado cuenta en este tiempo que llevo reclutando ingenieros de software es que mi cliente (la empresa) siempre se va a decidir por ese candidato que, aunque le falte conocimiento o práctica sobre alguna tecnología de desarrollo, sabe cómo comunicar alguna recomendación, advertencia, queja o comentario positivo que tenga sobre alguna situación de la chamba. Creo que vivimos en los tiempos en que las tecnologías libres nos empujan a no únicamente ser bueno en lo que hagamos sino cómo lo vendemos ante los demás.

Hugo Hernández es Reclutador de la industria TI, mejor conocido como Don Chambitas (@donchambitas).

Software para Aprender a Programar

Desde la perspectiva enseñanza-aprendizaje

Por Sergio E. Moreno

Este artículo está dirigido a todas aquellas personas que desean aprender o enseñar a programar. Ante la gran variedad de herramientas y lenguajes de programación, elegir la opción adecuada para aprender a programar tiende a ser una decisión intimidante. Además de considerar el tipo de personas a las cuales está dirigido cierto lenguaje o herramienta, otro factor a considerar es la metodología de enseñar y aprender, lo que muchas personas no toman en cuenta.

La metodología aquí expuesta fue probada con estudiantes del nivel medio superior sin conocimientos previos en el contexto de la programación y que al final obtuvieron resultados satisfactorios, lo cual expondremos paso a paso indicando el software como herramienta de enseñanza-aprendizaje, así como el lenguaje de programación.

ACERCAMIENTO AL CONOCIMIENTO

En esta primera etapa se debe reforzar el concepto de algoritmo (secuencia de pasos) y lógica (toma de decisiones) por lo que se recomienda el uso de Karel, este software permite crear programas haciendo uso de instrucciones 100% en español. Se debe considerar que hasta este momento no se hace uso de algún lenguaje de programación, lo que permite comprender las instrucciones básicas de la programación procedural o estructurada. La ventaja de hacer uso de nuestro propio lenguaje para programar propicia en primera instancia a tener un acercamiento del conocimiento previo con uno nuevo. En el sitio web de la comunidad Karelotitlán [1] puedes encontrar información en español sobre Karel, así como algunos ejercicios para practicar.

Otro software que se puede utilizar en esta etapa de aprendizaje es Scratch [2], que diferencia de Karel cuenta con una interfaz gráfica llamativa, y hace uso de elementos gráficos que permiten identificar visualmente los procesos de ejecución. El inconveniente para algunas personas es el idioma, ya que está en inglés. A pesar de esto la recomendación es empezar con Karel, resolviendo los escenarios propuestos en su guía y después seguir con Scratch.

COMPRENDIENDO Y ANALIZANDO EL LENGUAJE

Después de tener este acercamiento con la programación de algoritmos y procesos, llega el momento de elegir el lenguaje de programación. Puede ser Java o C++, no solo por que figuran en los primeros lugares de popularidad, sino porque hacen uso de una sintaxis definida acorde al paradigma de Programación Orientada a Objetos (POO); y por buenas prácticas de programación, es importante aplicar correctamente los tipos de datos, operadores, métodos y clases. Esta es la fase del aprendizaje en la que los estudiantes tienen más problemas debido a la conceptualización de las propiedades del paradigma de POO (encapsulación, herencia, sobrecarga y polimorfismo), por lo que se debe tener mucha paciencia y demostrar a través de escenarios su aplicación. Sea cual sea el lenguaje elegido, el ambiente de desarrollo (IDE) es un elemento que facilitará el trabajo de codificación al estudiante. Idealmente, el IDE debe tener capacidades que permitan identificar elementos del lenguaje y muestren errores a través de colores y algunas propiedades del texto como negritas o cursivas. Para quienes usan Java, una buena opción es Netbeans [3] y en el caso de C++ puede ser Visual Studio Code [4].

Aunque también existe la posibilidad de utilizar editores de código más ligeros como Sublime Text, Notepad++, Vim o emacs, considero que no son una buena opción para desarrollador novatos ya que requieren configuraciones adicionales para poder compilar y ejecutar programas; es importante recordar que se trata de facilitar el conocimiento para no terminar en la frustración.

APLICANDO Y UTILIZANDO EL CONOCIMIENTO

Una vez que ya se tiene la suficiente conceptualización y uso del lenguaje, podemos aprender a apoyarnos en bibliotecas que nos permitan crear software con mayores capacidades. De esta manera el estudiante podrá corroborar sus conocimientos y adaptarlos a otras aplicaciones con ciertos estándares y reglas. En el caso de nuestro estudio, como se trabajó con adolescentes y con el lenguaje Java, decidimos realizar un videojuego. Para ello elegimos Greenfoot [5], una aplicación que permite desarrollar videojuegos en 2D; también se pudo haber elegido Alice [6] para el desarrollo de videojuegos en 3D. Si hubiese sido el caso del lenguaje C++, lo ideal sería trabajar con la librería SDL (Simple DirectMedia Layer) y STL (Standard Template Library) para el desarrollo de videojuegos.

REGULANDO EL CONOCIMIENTO

Es un hecho que uno no termina de aprender solo con unas horas o un curso, por lo que hace falta practicar o repasar algunos temas. Afortunadamente existen diversos sitios web como Coding Bat [7] y Coding Game [8] que contienen ejercicios de práctica de programación, incluyendo soluciones.

CONCLUSIÓN

Con esta metodología, si eres profesor tus estudiantes tendrán las bases para usar cualquier otro lenguaje de programación, framework u otra tecnología para desarrollar un sistema más complejo y resolver problemáticas. Si eres autodidacta y no quieres caer en la frustración trata de seguir esta secuencia, no intentes correr sin aprender a caminar.

Aprender o enseñar a programar es un gran reto debido a que existe una diversidad de lenguajes y no es lo mismo que hace algunos años; estamos en un ambiente cambiante en donde hay nuevos entornos de desarrollo, necesidades y tendencias. Debido a esto el proceso de aprendizaje debe ser relativamente rápido, por lo cual es necesario brindar al estudiante una serie actividades y herramientas de software que le favorezcan en el desarrollo de su conocimiento.

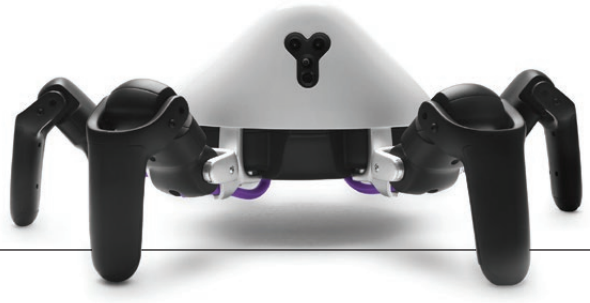
Referencias

[1] http://www.cmirg.com/karelotitlan
[2] https://scratch.mit.edu
[3] https://netbeans.org
[4] Visual Studio Code
[5] http://www.greenfoot.org
[6] http://www.alice.org
[7] http://codingbat.com
[8] https://www.codingame.com
[9] http://www.olimpiadadeinformatica.org.mx

El Maestro Sergio Ernesto Moreno Soto es Profesor de la carrera Técnico en Programación y Técnico en Desarrollo de Software en el Instituto Politécnico Nacional. ernestomoor@hotmail.com

1

HEXA



Habrá quienes consideren que los robots para uso cotidiano se están desarrollando (y adoptando) a un ritmo rápido. Sin embargo, si consideramos que en 1999 Sony lanzó al mercado el perro robot "Aibo" (vendiendo 150 mil unidades), es posible que cuestionemos cómo es que este campo no ha avanzado más en estos casi 20 años. Vincross, una empresa de origen chino dedicada a la robótica, argumenta que se debe a que los desarrolladores tienen un acceso muy limitado a robots ya que básicamente o son muy caros o son muy simples. Ante esto Vincross presenta HEXA, un robot que más que un producto final, busca ser una plataforma de desarrollo para aplicaciones robóticas. HEXA está diseñado para ser altamente maniobrable, por ello cuenta con 6 patas con 3 grados de libertad lo cual le permite subir escaleras y acceder a lugares que serían difíciles para otro tipo de robots. Cuenta con una cámara 720p, sensor de distancia, transmisor infrarrojo, acelerómetro y conectividad WiFi. Parte fundamental de HEXA es MIND, un sistema operativo basado en el kernel de Linux optimizado para robótica que incluye un SDK con funciones de alto nivel que facilitan la programación del comportamiento del robot. Se espera que HEXA esté disponible en primavera del 2018 y tendrá un costo de alrededor de 600 dólares.

2

WATER GARDEN

Water garden es una pecera de circuito cerrado que se limpia por sí misma. Utilizando la ciencia de la acuaponia, los desechos del pez son recolectados y utilizados como fertilizante para plantas que a su vez limpian el agua para el pez. El tanque tiene un capacidad de 3 galones, y es una gran adición para cualquier oficina, o salón de clases. El kit incluye el tanque, comida para el pez, grava, semillas, fertilizante y bomba de agua. Lo único que necesitas agregar es agua, el pez y listo.

3

CUBIIO

Una máquina de grabado láser es algo que típicamente pertenece en un taller y cuesta varios miles de dólares, es por ello que la mayoría de nosotros —tristemente— no tiene acceso a una. Cubiio viene a cambiar esto, es una máquina de grabado láser portátil y económica que puedes utilizar para todo tipo de proyectos personales. A pesar de caber en la palma de tu mano y pesar tan solo 150 gramos, proyecta un rayo láser con una potencia máxima de 800 mW (el grado de intensidad es ajustable) capaz de grabar distintos materiales como madera, papel, cartón, tela, piel y pan (sí, pan). Cubiio se controla desde una app móvil donde cargas el archivo con el patrón que deseas grabar (soporta imágenes y archivos G-code). Antes de grabar, puedes optar por el modo de previsualización con el cual se hace una simulación del grabado usando una luz de baja intensidad, y si te gusta entonces ya usas la intensidad para grabar. Cubiio actualmente está disponible en kickstarter con un precio de 299 dólares, y se planea comenzar a entregar pedidos en noviembre 2017.

4

MOPHIE POWERSTATION USB-C XXL

mophie lanzó una nueva batería portátil que tiene la capacidad de cargar laptops. El powerstation USB-C XXL tiene una batería de 19,500 mAh y está equipado con un puerto de salida USB-C para cargar laptops y otro USB-A para cargar smartphones y otros dispositivos electrónicos. El puerto USB-C cuenta con tecnología USB-PD que le permite cargar laptops rápidamente entregando hasta 30 watts de potencia. Esta tecnología también acelera la velocidad de carga del powerstation, permitiendo cargarla por completo en tan solo 3 horas. A pesar de su gran poder, su tamaño es relativamente pequeño (15 x 8.3. x 2.3 cm). Aunque el mophie powerstation USB-C XXL es anunciado como un cargador para Macbooks, en realidad puede utilizarse con otras laptops que se carguen por USB-C.

5

RELOJ DESPERTADOR MARATHON

Entre tantos gadgets con funcionalidad sofisticada, es bueno también encontrar cosas sencillas, elegantes y prácticas; tal es el caso del reloj despertador Marathon. No toca música, no se conecta al WiFi, no consulta el pronóstico del tiempo ni te dice tu horóscopo del día. Simplemente da la hora y suena una alarma. Lo que lo hace muy conveniente es el par de puertos USB de carga rápida que tiene al frente.

Humor

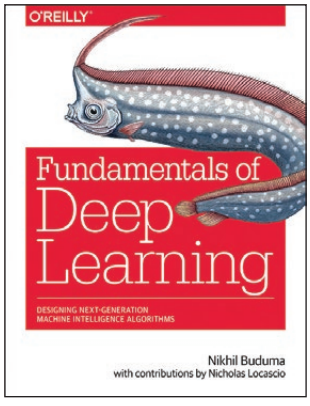
1 BLOCKCHAIN: LA REVOLUCIÓN INDUSTRIAL DE INTERNET

A. Preukschat, et al. Gestión 2000, 2017.

Blockchain esto, blockchain lo otro; sí, lo sabemos, empieza a cansar. Pero haremos la excepción con este libro por varias razones: 1) que está escrito originalmente en español; 2) reúne la colaboración de 30 autores distintos, y 3) explora su aplicación en una gran variedad de industrias, y aunque el contexto de referencia es España, hay bastante compatibilidad con lo que sucede en nuestra región.

El libro está estructurado en 3 partes, aunque en realidad podríamos considerar que es una parte principal complementada por anexos de referencia. La primera parte, que es la principal y forma aproximadamente ¾ del libro, analiza casos de aplicación de blockchain en distintos sectores: banca, seguros, telecomunicaciones, energía, salud, entre otros; también aborda el caso de los contratos inteligentes y temas de inversión. La segunda parte del libro tiene un tinte más filosófico, enfocado en "la descentralización como modelo de vida". Para las personas ya entradas en temas de hacktivismo y apasionados del ciberpunk, no habrá mucha novedad en esta sección, pero aún así es valioso como anexo para introducir a la audiencia de negocio sobre las implicaciones individuales y sociales del blockchain. Por último, la tercera parte explica los fundamentos técnicos de la tecnología blockchain, especialmente su implementación para bitcoin: cómo se estructuran los bloques, cómo se aplica criptografía, etcétera. Aunque no llega a mucha profundidad, sirve bastante bien para entender los aspectos técnicos a alto nivel. El libro cierra con un pequeño relato futurista que brinda una perspectiva de cómo podría ser la vida de cualquiera de nosotros en un futuro probablemente no tan lejano en el que varias de las ideas que se cuentan en el libro ya sean realidad.

Otro aspecto notable de esta obra es que está acompañada del sitio web http://libroblockchain.com en el que los distintos autores continúan publicando casos y opiniones sobre la aplicación del blockchain en distintas industrias.



2 FUNDAMENTALS OF DEEP LEARNING

Nikhil Buduma. O'Reilly, 2017.

Como su nombre lo indica, Fundamentals of Deep Learning es un libro dedicado a explicar los fundamentos del aprendizaje profundo. Dado que este es un tema complejo, el autor parte de la noción de que el lector ya cuenta con conocimientos de cálculo y álgebra lineal, así que posiblemente sea necesario dar un repaso previo a estos conceptos antes de abordar este libro.

Fundamentals of Deep Learning comienza explicando qué es una red neuronal y cómo se entrena, y muestra cómo implementar una red con TensorFlow. Posteriormente aborda temas como redes neuronales convolucionales, representación del aprendizaje y análisis de secuencias. El libro tiene un enfoque bastante práctico y contiene ejercicios con código fuente en Python para realizar actividades de aprendizaje profundo usando herramientas como TensorFlow.

En general, Fundamentals of Deep Learning es una gran introducción al tema, y sus ejemplos con código serán de gran ayuda para las personas interesadas en adentrarse en esta área. El único inconveniente que tiene es que los ejemplos (disponibles como repositorio en Github) están un poco desactualizados ya que se basan una versión pre-1.0 de TensorFlow, y esto provoca que algunos ejemplos no funcionen directamente con la versión más reciente de TensorFlow (1.2). Las fallas no son graves y pueden corregirse sin mucho problema, pero aún así pueden alentar el aprendizaje. El libro apenas se publicó en junio 2017, así que ojalá el autor aproveche para dar una actualizada a los ejemplos.