Sobre la industrialización del código

Durante una de las clases estuvimos hablando acerca de las razones por las que las soluciones de software tienen una calidad relativamente baja si las comparamos con los productos que suelen producir otras ingenierías. Y me parece un tema interesantísimo.

Creo que hay muchas razones por las que esto ocurre. Principalmente estimo que se debe a una poca profesionalización dentro del mundo del desarrollo de software. La cual estimo es perjudicial. Si quieres entender la cultura de un grupo no hay mejor manera que ver su humor. Pásate por /r/programmerhumor.

Pienso que la programación está más cerca de un arte que de un proceso industrial. Resolver problemas es una actividad principalmente creativa, y teclear la solución no tiene un gran valor en comparación. La dificultad de programar es descubrir el diseño de la solución y para ello hay que explorar tecleando.

El trabajo de los desarrolladores no es escribir código, es solucionar problemas. Todas las métricas relacionadas con el código producido son absurdas. Lo "limpio" que sea el código (¿qué significa limpio?), el estilo con el que escribes las variables (camelCase, snake_case, …), cuántas líneas tiene una función, el número de argumentos de una función, longitud de las líneas, etc. todo esto son banalidades. Cuestiones que no creo merezcan libros.

El trabajo como ingenieros es elegir los *tradeoffs* adecuados en cada situación.

No silver bullet

En la programación no existe una solución que valga para todo. Cada sistema tiene unas limitaciones y unos requisitos distintos. Por eso es casi imposible desarrollar código que vaya a funcionar bien en todas las situaciones.

El desarrollo del software consiste en gestionar la complejidad. Desde cómo diseñar el sistema hasta que nombres ponerles a las variables es un trabajo de gestión de la complejidad.

La complejidad se puede dividir en dos categorías: esencial y accidental. La complejidad esencial es la intrínseca que requiere el problema que se va a resolver. La accidental es toda aquella que no entra en la categoría anterior. Las herramientas de desarrollo, los lenguajes de programación, las librerías, etc. contribuyen principalmente a las complejidad accidental. Y más que eliminarla o aumentarla lo que suele ocurrir es que se traslada la complejidad a otra parte del proyecto.

Malas herramientas

Las herramientas de las que se dispone a la hora de desarrollar software no están todo lo avanzadas que podrían. Las herramientas llevan 30 años estancadas sin tener mejoras sustanciales que ayuden a un desarrollo más rápido y eficiente.

Para empezar los lenguajes de programación llevan sin innovar considerablemente mucho tiempo. No han evolucionado significativamente en los últimos 50 años, no son más que pequeñas iteraciones de Algol 68. Diría más, los lenguajes han ido perdiendo poder expresivo y han crecido en complejidad. Los lenguajes de programación más populares ahora mismo se diseñaron para los ordenadores de hace 30 años en el mejor de

los casos. Para ordenadores que disponían de poca memoria y no más de un procesador. Se han tomado decisiones cuestionables en los lenguajes de programación que se han convertido en el estándar.

Por ejemplo, HTML era un lenguaje aceptable para su función original, CSS no está bien diseñado (Lo que a los estilos se refiere es aceptable pero, que un lenguaje que tiene que expresar posicionamiento no haya manera de decir “encima de”, “a la izquierda de”, …). JavaScript es un lenguaje que se diseñó para pequeños scripts en el cliente. Ninguna de estas 3 tecnologías son adecuadas para el uso que se le dá ahora a la web.

La prevalencia de los lenguajes interpretados y sin tipar, también se están usando para tareas que no fueron pensados inicialmente introduciendo muchos problemas. Desde el rendimiento a la memoria y a la creación de nuevas metodologías para lidiar con fallos por la falta de tipos y la naturaleza dinámica de los lenguajes (lo sé, inesperado).

Se ha seguido una tendencia hacia los lenguajes “con ruedines”. Se ve al programador como alguien a quien hay que limitar las posibilidades porque si no va a hacer algo perjudicial. La evolución de los lenguajes de programación ha ido restringiendo el conjunto de acciones posibles que puede ejecutar un programador con la premisa que de esta manera cometerá menos fallos. Sin embargo, aunque se cubren ciertos fallos no existe ninguna manera de evitar los fallos en la lógica del producto. En lugar de tener herramientas avanzadas para los programadores más experimentados que saben lo que hacen se les trata a todos por igual como principiantes. De esta manera no se saca todo el provecho que se podría de los desarrolladores más expertos.

Los entornos de desarrollo, herramientas de debugueo, las librerías, … tampoco son todo lo buenos que podrían ser. Faltan cosas conceptualmente sencillas como herramientas que

muestren los eventos de un sistema distribuido centralizados y ordenados en un único lugar, editores inteligentes que nos muestren los resultados del código según lo estamos escribiendo [9], lenguajes de programación que nos permitan metaprogramar.

*Pregunta, ¿Por qué programamos en texto todavía? ¿Y si programásemos en el AST directamente?

Esto hace que los tiempos de iteración sean mucho más lentos de lo que podrían ser dificultando la exploración de la solución.

*Preguntas, ¿Por qué asociamos lenguajes de programación a algunas tareas/ámbitos? ¿Por qué creemos que hay cosas que "solo" se pueden hacer en cierto lenguaje? ¿Por qué incluimos la librería estándar cuando hablamos de un lenguaje?

Industrialización

Cuando hablamos de industrializar el software nos referimos a crear piezas que se puedan usar y combinar de manera sencilla para facilitar el desarrollo de software futuro.

Primero tenemos que empezar por definir que es una *pieza* de software. Y ya se nos complica. ¿A qué nos referimos? ¿Dónde ponemos el límite? Una pieza es, ¿una función, un algoritmo, una librería o un sistema? Dependiendo del "tamaño" que definamos como pieza elegimos más rendimiento o menos. La filosofía Unix dice:

> *Haz que cada programa haga una cosa bien. Para hacer un nuevo trabajo, construye de nuevo en lugar de complicar los viejos programas añadiendo nuevas «características».*
>
> *Doug McIlroy*

Sin embargo, las limitaciones del mundo real, de la máquina en

la que se ejecutan los programas hace que esta filosofía no se pueda realizar [3].

Segundo, tenemos que hacer que las piezas se entiendan entre sí. Hay que estandarizar las interfaces, la comunicación entre las piezas. Esto supone no solo una pérdida de rendimiento sino que es una decisión que va a tener consecuencias durante muchos muchos años. O sea si se comete algún error ¿cómo se va a lidiar con la *retrocompatibilidad*? Empieza a crecer la complejidad.

Tercero, la gestión de distintos lenguajes de programación, librerías, "ecosistemas". La mayoría de lenguajes no exponen una manera sencilla de operar con otros lenguajes de programación. Pocos lenguajes tienen las funcionalidades necesarias para realizar estas comunicaciones de manera eficiente. La complejidad sigue creciendo.

Cuarto, las abstracciones y la composición de programas no escala bien. Las abstracciones como su nombre indica ocultan los detalles de las capas inferiores. Parece que esto reduce la complejidad del sistemas pero solo la traslada a otro lugar. Ahora en vez de tener un solo problema (un sistema que entender) hay dos [1]. Y sigue creciendo la complejidad.

Quinto, reutilizar código se parece más a un trasplante que a unir piezas de Lego [2]. El código siempre toma se adapta al problema y a la situación para la que se ideo. Cuando tomamos ese código y lo introducimos en otro proyecto (que no sea muy similar) es necesario introducir una capa de traducción o reescribir y adaptar casi todo el código copiado. Y seguimos complicando más la solución.

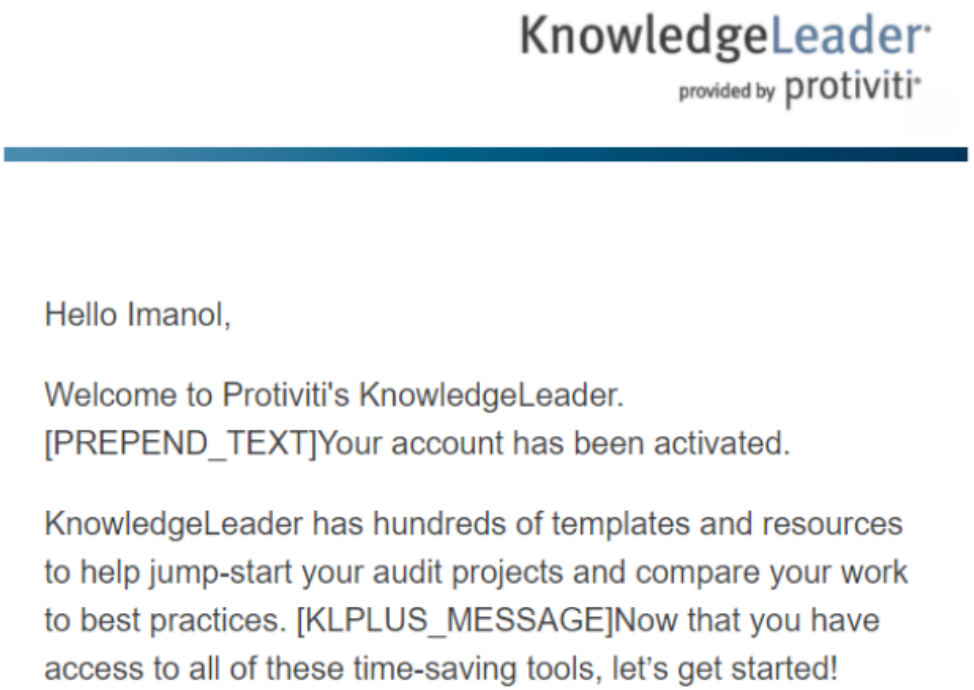
Sexto, ¿Se puede programar sin diseñar o diseñar sin programar? Si bien teclear es la parte sencilla de la programación, la atención al detalle que requiere esa acción ayuda a diseñar mejores soluciones. Diseñando en abstracto es muy complicado percatarse de todas las interacciones entre los

diferentes sistemas que componen la solución.

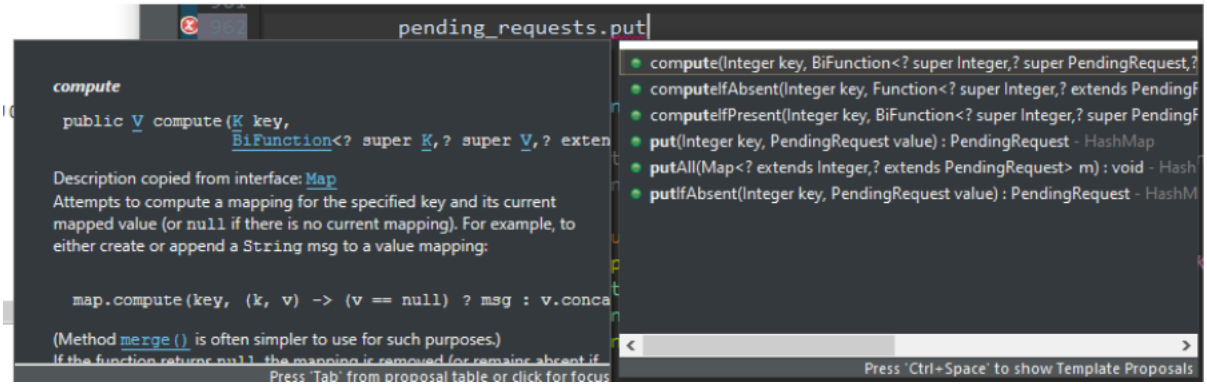
Malas metodologías

Hay muchas presiones externas a la programación que reducen la calidad de lo que se produce. La necesidad de ser los primeros en sacar un producto al mercado está en contraposición con hacer un producto de calidad. El resultado suele ser productos incompletos que se van a seguir desarrollando progresivamente. Pero una vez llegado al mercado es casi imposible realizar un cambio estructural grande, por lo que la solución se queda atrapada en el espacio que ha explorado.

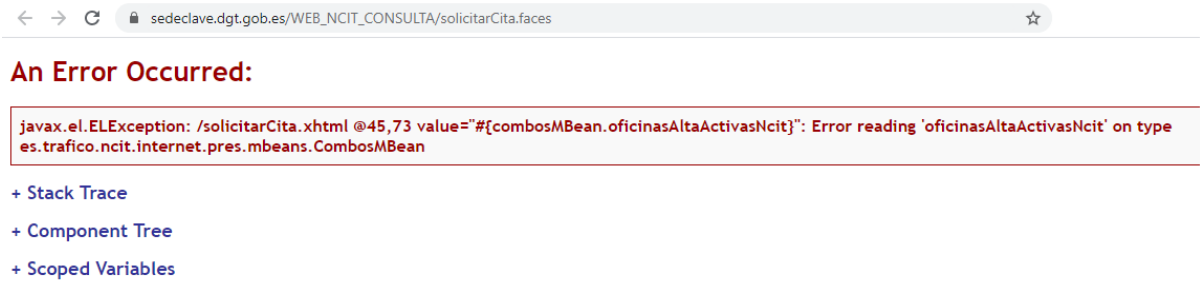
Aunque se ponga mucho énfasis en la calidad, la verdad es que no lo parece. Esto son cosas que he ido recogiendo durante la última semana.



Esta es la primera impresión que te llevas de KnowledgeLeader.



Un autocompletado que ordena mal las opciones.



La DGT que ante un error te muestra toda la información de desarrollador con el agujero de seguridad que es eso y la mala imagen que le transmite al usuario.

Referencias

[0] There is no Silver Bullet, https://en.wikipedia.org/wiki/No_Silver_Bullet

[1] The law of leaky abstractions, https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/

[2] LEGO blocks and organ transplants, https://www.johndcook.com/blog/2011/02/03/lego-blocks-and-organ-transplants/

[3] Where the Unix philosophy breaks down, https://www.johndcook.com/blog/2010/06/30/where-the-unix-philosophy-breaks-down/

[4] Jonathan Blow on Software Quality, part 1, https://www.youtube.com/watch?v=oJ4GcZs7y6g

[5] Jonathan Blow on Software Quality, full talk, https://www.bilibili.com/video/av21278220/

[6] Preventing the collapse of civilization, https://www.youtube.com/watch?v=ZSRHeXYDLko
[7] Reboot Develop 2017 – Jonathan Blow, Thekla Inc. / Making Game Programming Less Terrible, https://www.youtube.com/watch?v=De0Am_QcZiQ
[8] The thirty million line problem, https://www.youtube.com/watch?v=kZRE7HIO3vk
[9] Dion systems, https://media.handmade-seattle.com/dion-systems/
[10] Whitebox, https://azmr.itch.io/whitebox

Falta de interés

Quiero aprovechar este último post para acabar de contar algunas de las cosas que se han quedado pendientes por la falta de tiempo y otros motivos.

Cuando se nos preguntó sobre qué mejoraríamos de las clases dije “no asumas que tenemos interés”. Creo que quedó un poco brusco por la falta de tiempo y no puede expresar adecuadamente todo lo que quería decir. Aun así, pienso que sintetiza bastante bien un sentimiento compartido con mis compañeros por lo que me dijeron al acabar la clase.

La mayoría de mis quejas están dirigidas al sistema educativo, no creo que la universidad ni los profesores estén haciendo un mal trabajo dentro de las restricciones que se les imponen. En este aspecto envidio la universidad americana en la que no hay asignaturas ni están organizadas por año, sino que hay cursos y cada cual elige los que quiere cursar.

Volviendo al interés, ¿Cuál es el perfil del estudiante de informática? ¿Qué hace aquí? Me baso únicamente en mi

experiencia personal.

El estudiante universitario de ingeniería informática puede ser:

- Alguien con interés en la tecnología y/o los videojuegos. En este caso, lo normal es que haya elegido el grado porque quiere aprender a programar, habilidad que necesita para hacer juegos más sofisticados que lo poco que ha trasteado con las herramientas gratis y de programación visual o scripts básicos copiados de un tutorial.
- Alguien con interés en la inteligencia artificial. Aquí también es raro que haya trasteado mucho por su cuenta. En la mayoría de casos conocerá los algoritmos de moda como “deep learning” o “redes neuronales”.
- Alguien con interés en seguridad. O porque quiere ser “hacker” o porque le han dicho que es un perfil muy solicitado. Habrá “hackeado” a algún amigo siguiendo un tutorial en Youtube.
- Alguien que quiere hacer una página web y hacerse rico. No, este tampoco ha investigado mucho por su cuenta como se hacen estas cosas.
- Alguien que quería hacer una ingeniería pero no sabía cuál y ha acabado aquí.

Es raro encontrarse con estudiantes que tengan experiencia en los temas que se van a tratar – muy pocos saben siquiera programar algo. Creo que el perfil ha cambiado bastante, ya no existe el perfil del “hacker” no se puede trastear con los ordenadores actuales, tampoco sabe mucho de seguridad (porque hoy en día no es tan importante para el usuario como lo fue cuando los chats irc), son sistemas muy cerrados. Hablando con profesores, los nuevos estudiantes cada año necesitan más clases extras de Programación I. Sin embargo, muchos profesores esperan que sepamos de sus temas cuando no hemos sido expuestos a ellos nunca.

A menudo nos encontramos con que los profesores piensan que tenemos interés en la asignatura y hemos buscado información, aprendido cosas por nuestra cuenta o que tenemos experiencia en esos ámbitos.

Por otro lado está el problema de la universidad que no tiene claro si quiere formar a investigadores y expertos, o trabajadores. El grado de informática es una extraña mezcla entre asignaturas teóricas, rigurosas y matemáticas (pero no demasiado) y, asignaturas empresariales y de proyectos (pero no demasiado y centrado en empresas enormes). El viaje resulta muy superficial sobre "la informática" y no sientes ser experto en ninguna de las áreas al acabar. Las expectativas a menudo no se cumplen.

Volviendo otra vez al interés, (elijo esta asignatura por cercanía pero se puede aplicar a cualquier otra) ¿qué se supone que tenemos que aprender sobre sistemas de información empresarial? Afortunadamente nuestro profesor ha rellenado la asignatura con temas muy interesantes. Pero en lo que al núcleo de la asignatura se refiere, ¿qué tiene que saber un estudiante?

- ¿Tenía que aprender el significado de la infinidad de siglas, CRM, ERP, GIS, SCM, …?
  ¿Qué valor hay en eso?
- ¿Tenía que aprender que existen herramientas BI y las características de estas? ¿Y el año que viene cuando salgan nuevas?
  ¿Cuánto va a durar este conocimiento?
- ¿Qué tengo que aprender acerca de la planificación estratégica de una empresa? ¿Qué hay que tener sentido común y alinear los objetivos de los demás departamentos?
  ¿Cuánto se tarda en enseñar eso?

Muchas de estas cosas ya las hemos visto en otras asignaturas.

Es increíble cuánto se repiten entre sí las asignaturas.

Resumiendo, los estudiantes de informática solemos tener unos intereses muy concretos, muchas asignaturas son muy superficiales y no se llega a las partes interesantes (ni siquiera en las que estamos interesados). Los estudiantes a menudo no tenemos experiencia, ni laboral, ni en esa asignatura. Eso es lo que pretendía transmitir con lo del interés.

Propiedad intelectual, apuntes y retrospectiva

Durante esta serie de artículos hemos ido viendo el valor de las propiedades intelectuales, los riesgos y el valor del proceso de auditoría. En este artículo pretendo sintetizar el trabajo realizado en los anteriores. A la vez que complementar algunas cuestiones que creo que pude pasar por alto.

El viaje comenzó identificando que consideramos propiedad intelectual y cuales son las diferentes formas en las que se manifiesta. Hablamos de que estos activos en la era digital y de servicios en la que nos encontramos son los que más valor y diferenciación aportan a las empresas. Identificamos, patentes, modelos de utilidad, derechos de autor, propiedad industrial, *trademarks*, imagen comercial, secretos comerciales, … También vimos que la ley protege especialmente aquellas que se pueden vender, pero nosotros debemos considerar un espectro más amplio e incluir toda aquella información que sea relevante para la empresa.

En el siguiente artículo seguimos indagando las categorías que contemplaban las leyes, que sí y que no se puede proteger con

las herramientas legales del estado. Y mencionamos algunos marcos y estándares dirigidos a las propiedades intelectuales. Es interesante descubrir que no aparecen los típicos marcos que aparecen casi en cualquier otro tema de auditoría, sino que en este ámbito el referente es WIPO (*World Intellectual Property Organization*). Intuyo que esto se debe a que la gestión de las propiedades intelectuales está íntimamente relacionada con la seguridad en general. Así que supongo que podría añadir a aquel artículo la ISO 27000 a la lista de marcos.

Después pasamos a identificar los riesgos a los que se enfrenta cualquier organización en los que a propiedades intelectuales se refiere. En este apartado identificamos 3 grandes categorías en las que podíamos incluir un montón de riesgos: *olvidarse de las PI*, *perder o que nos roben PI* e *infringir las PI de otros*. Me aventuro a decir que las más comunes son la primera y última categoría. Entiendo que la segunda es una preocupación más dirigida a grandes empresas multinacionales que manejan cientos de propiedades intelectuales en equipos enormes y las diferencias en tecnología o procesos son importantísimas.

Y en el último post se esquematiza el proceso de una auditoría y los controles que se podrían implantar en una organización para securizar la PI. El proceso de la auditoría comienza por identificar todas las propiedades intelectuales de las que dispone la organización, haciendo un inventario, identificando a la gente con acceso a las PI, e investigando otras fuentes que nos digan que cosas hay a nombre de la empresa, como los dominios de internet que tenga comprados. Después hay que comprobar quienes tienen acceso a la información sensible y que no se este haciendo un uso indebido de alguna PI de terceros.

Como ya he dicho la mayoría de estos controles pertenecen a la seguridad. Desde el punto de vista TI los controles generales y de personas se extienden también al mundo digital. Por lo

que además debemos aplicar los controles de segregación de funciones, controles de acceso, registro de actividad, etc. a todos los sistemas de información que contengan información sensible, es decir, prácticamente todos. Particularmente en cuanto a las personas y terceros que puedan tener acceso los contratos de confidencialidad son indispensables.

En conclusión, la propiedad intelectual tiene un valor tremendo en cualquier organización, es lo que diferencia y más valor aporta. Afortunadamente existen herramientas legales que ayudan a proteger estos. (Des)afortunadamente, las empresas tienen que gestionar estos activos de manera adecuada y para ello necesitan el servicio que prestan los auditores y consultores. Para esto lo esencial es identificar todas aquellas propiedades intelectuales que sean de valor para la empresa y protegerlas o sacarles el mayor partido posible.

Enlaces

[1] <<Propiedad intelectual, introducción y contexto>>, PublicaTIC, https://blogs.deusto.es/master-informatica/propiedad-intelectual-introduccion-y-contexto/

[2] <<Propiedad intelectual, relevancia>>, PublicaTIC, https://blogs.deusto.es/master-informatica/propiedad-intelectual-relevancia/

[3] <<Propiedad intelectual, riesgos>>, PublicaTIC, https://blogs.deusto.es/master-informatica/propiedad-intelectual-riesgos/

[4] <<Propiedad intelectual, controles y auditoria>>, PublicaTIC, https://blogs.deusto.es/master-informatica/propiedad-intelectual-controles-y-auditoria/

Propiedad intelectual, controles y auditoría

En el artículo anterior comentamos los riesgos a los que se exponen las empresas en cuanto a la propiedad intelectual. Identificamos 3 categorías: No olvidarnos de nuestra PI, que no nos roben nuestra PI y no infringir la PI de otros.

Para cada uno de estos posibles riesgos existen distintos controles que podemos implementar para reducir la probabilidad de que ocurran. Asegurando de esta manera el correcto funcionamiento de la empresa y asegurando el uso y aprovechamiento de estos activos tan importantes.

Empecemos por lo más básico, tener controlado cuales son las PI de las que disponemos. Para evitar que se olvide cuales son las PI de las que dispone la organización hay algunos controles obvios que hay que implementar.

Como auditores nuestro trabajo comienza por *identificar las propiedades intelectuales*. Hay muchísimas maneras de descubrir toda la información que puede ser de importancia para la organización. Algunas de las técnicas en las que podemos pensar son:

- Realizar un inventario periódico de las PI de las que dispone la empresa.
- Identificación de la documentación relacionada con las PPII.
- Clasificación de las PI. Según tipo, importancia, …
- Descubrir e investigar licencias a nombre de la empresa.
- Validar que las licencias de PPII que hace uso la empresa son válidas.
- Investigar acuerdos con terceros sobre PI.

- Procedimientos investigación y flujos de trabajo bien documentados.
- Identificación de los principales generadores de PI. Seguramente el equipo de investigación o desarrollo.
- Análisis de repositorios de documentos.
- Descubrir nombres de dominios de internet relacionados con la organización.
- Si la organización opera internacionalmente, análisis del estado de las patentes, marcas, … en todos los países en los que opera.

Además queremos evitar perder estos datos por lo que necesitaremos establecer controles que eviten la pérdida de información. Aquí entrarían varias técnicas de *seguridad de la información*, como copias de seguridad o replicación de documentos.

Una vez tenemos identificado aquello que queremos proteger nos fijamos en los controles que nos ayudan a evitar que manos u ojos indeseados se hagan con nuestros activos. Esta parte se solapa mucho con la gestión de la *seguridad.*

- Controles de acceso.
- Restricciones de acceso.
- Segregación de responsabilidades.
- Identificación de usuarios.
- Registro de actividad.
- Ciberseguridad.
- Concienciación de los empleados acerca de la PI.
- Investigación de los contactos y pasado de los empleados.
- Contratos de confidencialidad.
- Marcar como confidencial la información o documentos para que no haya dudas.
- Expresar claramente en los contratos de los empleados la confidencialidad y las consecuencias en caso de incumplimiento.

Y por otro lado, debemos asegurar que las propiedades intelectuales están adecuadamente protegidas haciendo uso adecuado de las *herramientas legales* de las que se dispone.

- Patentes
- Modelos de utilidad
- *Copyright*
- Marcas
- Diseños industriales
- Derechos de autor
- Secretos comerciales
- Seguros
- …

También debemos asegurar que la estrategia en cuanto a la PI se alinea con la estrategia de la empresa.

El último problema al que nos podemos enfrentar es a estar infringiendo la propiedad intelectual de terceros. En estos casos nuestro trabajo como auditores es reportarlo a la dirección para que tomen las acciones necesarias para resolver el problema.

Referencias

[1] <<Intellectual Property Process Audit Report>>, KnowleadgeLeader, consultado el 20/11/2020, https://www.knowledgeleader.com/knowledgeleader/content.nsf/web+content/au/ditreportintellectualpropertyprocess

[2] <<Intellectual Property Management and Auditing>>, consultado el 20/11/2020, https://www.innovation-asset.com/the-audit-and-management-of-intellectual-property

[3] <<IP Audit Check-list>>, consultado el 21/11/2020, https://www.southeastasia-iprhelpdesk.eu/sites/default/files/publications/EN_Audit.pdf

[4] <<Law and Internal Auditing>>, consultado el 21/11/2020, https://na.theiia.org/training/Public%20Documents/Intellectual%20Property.pdf

[5] <<Fact Sheet IP audit: Uncovering the potential of your business>>, consultado el 22/11/2020, https://www.iprhelpdesk.eu/sites/default/files/newsdocuments/Fact-Sheet-IP-Audit-Uncovering-Potential-Your-Business-EN.pdf

---

Qué buscar en tu trabajo ideal

Una preocupación que tengo para cuando acabe de estudiar y me encuentre en el salvaje y desconocido mundo laboral, es si me gustará mi trabajo. Desde que entré en la carrera de informática he estado intentando averiguar cómo es el trabajo de los informáticos. Por un motivo muy sencillo, durante la carrera me he encontrado con muchas asignaturas que no me han interesado lo más mínimo.

Por suerte o por desgracia para mí, la programación era un hobby antes de empezar la carrera. Aprendí a programar muy temprano cuando me compraron mi primer ordenador al empezar la ESO. Cuándo llegué a la universidad ya sabía programar más que lo que me han exigido en cualquier asignatura. Soy una persona que le gusta entender muy bien cómo funcionan las herramientas que utiliza. Tengo una forma de mirar las cosas muy de "ingeniero", y no entiendo por qué la imagen del informático moderno está muy lejos de estas preocupaciones.

Me gusta reinventar la rueda*. Me gusta hacer las cosas a mano, hacerlas yo. Me gusta conocer las bases y los fundamentos, conocer al menos un nivel por debajo de lo que se

requiere. Saber de matemáticas, saber de hardware, entender de algoritmos y estructuras de datos. Valoro mucho los conocimientos fundamentales de cualquier área porque son aquellos que se van a poder aplicar siempre, son verdad.

No me gustan las modas, dan lugar a un conocimiento pasajero. ¿Hay conocimiento valioso en saber usar la última librería que está se ha puesto de moda? Para mí ese conocimiento es efímero. Cuándo se pase de moda o aparezca algo mejor, el tiempo y el esfuerzo invertido en eso no vale nada. Sin embargo, si aprendes como funciona internamente, cual es el problema que resuelve y cómo lo resuelve, entonces ese conocimiento lo podrás aplicar a la nueva librería que se ponga de moda, porque conoces los fundamentos.

Desafortunadamente mi experiencia es que incluso entre profesionales y profesores hay demasiado *hype*. Entusiasmarte demasiado por una herramienta nueva es un poco absurdo siendo ingeniero. O sea, un martillo nuevo puede estar guay, pero sigue siendo un martillo. Igual te permite hacer algo de manera más fácil, pero hará otra más difícil. No he encontrado mucho pensamiento crítico en una disciplina que se hace llamar ingeniería. Funciona por modas, por “me apetece probar X”, “¿usamos X?”, “he leído que nosequien ha usado X”. Rara vez veo que se valoren los pros y los contras de una tecnología antes de usarla, o que se valore el coste de las dependencias de los componentes.

Creo que mi caso es bastante poco frecuente, los temas que me interesan y la profundidad a la que me gusta entender las cosas es muy distinto a la materia que se da en la universidad – tal vez se deba a que el nivel tiene que ser mucho menor para que lo pueda seguir la mayoría de los alumnos – y la que al parecer se exige en las empresas.

Además se intentan abarcar tantas áreas que casi todas las asignaturas se pueden ver como “introducción a X”. Porque no hay tiempo para profundizar.

Por eso, más de una vez me he parado a pensar y he anotado algunas de las cosas que me incomodan o me gustan menos y las cosas que disfruto haciendo.

Para hacer este análisis es fundamental que te conozcas a tí mismo. Hay muchas maneras de conocerse, pero si no quieres pasar muchas horas reflexionando recomiendo que te bases en tus experiencias pasadas.

Durante años de psicoanálisis se han distinguido 5 rasgos de personalidad principales en las personas. Conocer cuales son estos rasgos y en qué rango de cada uno nos situamos nos proporciona algunos indicadores sobre los cuales enfocar el análisis y a comprender a otras personas.

Quiero matizar que estar en un extremo u otro de los rangos no es mejor que estar en el otro, ambos lados tienen sus pros y sus contras, y debemos ser conscientes de esto.

Apertura a las experiencias: Es la disposición natural de las personas hacia las ideas nuevas y la curiosidad, disfrutan del arte, son creativos y enfrentan cuestiones complejas. La gente menos abierta suele demostrar actitudes más tradicionales, prefieren lo conocido, simple y obvio. Internamente se puede dividir en intelecto y apertura o creatividad.

Responsabilidad/Conciencia: Este rasgo mide la capacidad de una persona de organizarse y trabajar. La gente que puntúa muy alto en este rasgo “no pueden estarse quietos” en un sentido literal, sienten que si no están haciendo algo están perdiendo el tiempo y no son capaces de soportarlo. Se puede dividir en la habilidad de ser organizado y ser trabajador.

Extraversión: Como de introvertida/extrovertida es una persona. La forma más sencilla de clasificarte en esta categoría es preguntarte si estar rodeado de gente o de fiesta te da energía o te drena. La gente introvertida tiende a fatigarse con las interacciones sociales y necesita “recargar”. Los extrovertidos sin embargo buscan interacciones

y reciben energía de estas. Se puede dividir en entusiasmo o excitación y confianza o asertividad.

Amabilidad (cooperativo/competitivo): Como los paréntesis indican la gente que puntúa alto en este rasgo son gente cooperativa que trabaja bien en equipo y que se preocupa por los demás y que no se suele meter en problemas (También se puede ser demasiado amable y eso acarrea sus propios problemas). En cambio la gente que puntúa al otro lado del espectro tiende a ser más egocéntrica, competitiva, más exigente. Se divide en compasión y en cortesía o cordialidad.

Neuroticismo: O estabilidad emocional según se mire, es el rasgo que define cuanto se ven afectadas las emociones de una persona. La gente que puntúa alto en esto suele tener niveles de ansiedad mayor y más emociones negativas. Por el contrario la gente que puntúa bajo tiende a no tener grandes cambios en sus emociones. Esta se puede dividir en la tendencia evitar la incertidumbre (no existe palabra en castellano para esto) y volatilidad.

En estas dos últimas categorías las mujeres suelen puntuar más alto que los hombres. Se cree que esto se debe a que la psique de las mujeres está adaptada a la mujer con hijos y por esto detecta más peligros y es más amable.

*Todas las parejas de subcategorías están fuertemente correlacionadas entre sí, o sea que es raro puntuar muy alto en una y muy bajo en la otra.

Desafortunadamente este tema como tantos otros tiene vocabulario que es mucho más compresible en inglés que en castellano.

Enthusiasm (spontaneous joy and engagement) and Assertiveness (social dominance, often verbal in nature) for Extraversion.

Withdrawal (the tendency to avoid in the face of uncertainty) and Volatility (the tendency to become irritable and upset

when things go wrong) for Neuroticism.

Compassion (the tendency to empathically experience the emotion of others) and Politeness (the proclivity to abide by interpersonal norms) for Agreeableness.

Industriousness (the ability to engage in sustained, goal-directed effort) and Orderliness (the tendency to schedule, organize and systematize) for Conscientiousness.

Openness (creativity and aesthetic sensitivity) and Intellect (interest in abstract concepts and ideas) for Openness to Experience.

Hay que remarcar que no se sabe si estos rasgos se pueden alterar, o sea que intentar ser alguien que no eres es muy complicado. Lo que sí se cree es que se puede estirar el rango o zona en el que nos sentimos a gusto en cada uno de los rasgos. Es decir, que una persona introvertida puede entrenar y mejorar su extraversión de manera que se sienta más a gusto rodeado de gente que al principio.

Conociendo cómo puntuamos en cada una de estas escalas nos puede ayudar a discernir qué trabajo se ajusta más a nuestra personalidad, sin tener que hacer una gran introspección (que es necesaria, pero un test online de estos cinco rasgos nos puede ayudar a enfocarlo de una manera más acertada).

A continuación, una lista de algunas de las cuestiones que he encontrado útil a la hora de decir que tipo de trabajo me gusta hacer.

1. ¿Construyes o mantienes?

¿Te gusta crear, hacer cosas nuevas, cambiar las cosas, programar o es algo que no te preocupa y lo que quieres es tener un sistema que funcione y dedicarte a su mantenimiento?

Se traduce en cuanta creatividad quieres en tu trabajo. Si quieres que tu trabajo sea creativo o no lo sea.

2. ¿Que uses tus habilidades actuales o tengas que aprender nuevas habilidades?

Casi siempre vas a tener que aprender cosas nuevas cuando empieces a trabajar pero deseas tener que seguir aprendiendo o una vez domines las habilidades seas eficiente y no te requiera el trabajo que aprendas más cosas, que sea decisión tuya.

La distinción radica en si usas mayoritariamente las habilidades que ya posees o si cada proyecto te va a exigir habilidades nuevas y que estés aprendiendo constantemente.

3. ¿Cuánta autonomía tienes?

¿Cuántas decisiones tienes la libertad de tomar? ¿Cuántas quieres tomar? Puede darse el caso de que estés realizando un trabajo creativo pero que no puedas tomar todas las decisiones que quisieras. O al contrario, que prefieras tener alguien que te guíe y te dé instrucciones claras.

4. Interacciones con otros
    1. ¿Cuánta interacción?
       Este punto trata sobre encontrar la cantidad de interacciones que mejor funcionan para tí. Si prefieres trabajar solo y tener poco contacto con los demás o te gusta trabajar en equipo y hacer muchas tareas juntos.
    2. ¿De qué tipo?
       También es importante cómo se da esta interacción, que tipo de relaciones son. Interactúas de cara al público con muchos desconocidos; interactúas con un selecto grupo de personas entusiasmadas con lo que hacen; como es la interacción con tus compañeros, con tus superiores; creas materia para otros o les enseñas directamente.

Hay muchísimos aspectos en esta pregunta. Cuánta gente, qué tipo de gente y, de qué manera se da la interacción.

5. Nivel de autoridad sobre otras personas

Se explica solo. Hay gente que le gusta tener más autoridad y gente que no se siente cómoda teniendo mucha autoridad en un grupo.

¿Quieres dirigir personas, equipos o proyectos? Igual no es para tí, igual estarías más cómodo trabajando dentro de un equipo que dirigiendolo. Hay gente para todo.

6. ¿Te gusta ser el centro de atención?

Cuánta atención quieres y a qué quieres que se deba esa atención. Igual quieres que se te preste mucha atención en aquellos temas en los que eres experto, o igual quieres tener siempre la última palabra. O quizá, cuanto menos atención haya sobre tí significa que has hecho mejor las cosas y no te sientes cómodo con mucha atención puesta en tí o lo que haces.

7. Conciliación entre la vida laboral y la personal

¿Cuánto tiempo libre te permite el trabajo? ¿Qué disponibilidad se te exige? Hay gente cuya vida gira entorno a su trabajo y gente que desearía trabajar lo menos posible y tener todo el tiempo libre posible para hacer lo que le gusta – Seguramente porque no les gusta mucho su trabajo, para eso están estos puntos!

¿Qué tipo de persona eres? ¿Qué buscas?

Incluso dentro del mismo trabajo casi siempre hay maneras de tirar más en la dirección en la que más a gusto te sientes. No es cuestión de encontrar lo que te gusta sino de hacer que te guste lo que haces. Además, hablando se entiende la gente. Si demuestras ser responsable y la gente confía en tí, puedes pedir como quieres hacer tu trabajo y rara vez te encontrarás oposición.

Enlaces de interés:

Modelo de los cinco grandes (Wikipedia):
https://es.wikipedia.org/wiki/Modelo_de_los_cinco_grandes

Hierarchical structure of the Big Five (Wikipedia):
https://en.wikipedia.org/wiki/Hierarchical_structure_of_the_Big_Five#Subcomponents_of_the_Big_Five

How to figure out what to do with your life:
https://www.youtube.com/watch?v=jVA-tr_euKU

---

*He movido esto aquí porque la “rant” se me estaba alargando mucho, creo sinceramente que lo más importante son los links que dejo al final.

Reinventar la rueda. Es un término muy utilizado en el mundo del desarrollo de software que aboga por reutilizar todo lo que se pueda. Entiendo lo que pretenden pero creo que se ha llevado demasiado lejos y además ni siquiera es una buena analogía. ¡Como si solo existiera un tipo de rueda! ¿Son iguales las ruedas de un coche y las de un patinete? ¿y una bici?

El software moderno intenta meter ruedas de camión en coches de Hot Wheels. De esta manera se piensa que se está ahorrando tiempo de desarrollo y que se está mejorando la fiabilidad de los productos que desarrollan. Sin embargo, lo que se consigue es que los desarrolladores no saben lo que está pasando “bajo el capó” de su aplicación. No entienden los fallos, ni el rendimiento, y aunque quizá tengan menos bugs, seguro que les lleva más tiempo arreglarlos que si hubieran hecho ese componente ellos mismos.

Da la impresión de que desarrollar software hoy en día consiste en coger componentes que ya están hechos y en

juntarlos para hacer algo nuevo. Y sinceramente, escribir el *glue code* es de las tareas más aburridas que hay para mí. Porque no estás pensando, lo único que hace el código es copiar datos de un lado a otro.

Otra de las cosas que me incomoda es que los componentes se tratan como cajas opacas y, si el proyecto es majo, tarde o temprano vas a querer saber lo que está pasando. Ya sea por temas de rendimiento o por bugs llegará el momento en el que tengas que mirar dentro de la caja y es algo en lo que no se suele pensar. Las dependencias tienen un coste enorme en los proyectos. Especialmente si se introducen librerías o componentes enormes de los que solamente se necesitan unas pocas funciones, que se podrían haber implementado a mano en un corto periodo de tiempo. Entonces, ¿la mayoría del software que se desarrolla es sencillo? No me veo haciendo software trivial en el que no haya que aprender nada interesante. Y tampoco me veo haciendo software en el que me voy a tener que pegar con las librerías por rendimiento o funcionalidad.

El rendimiento es otra de esas cosas que me flipan. Me cuesta creer que haya páginas web en las que hacer *scroll* ralentice el ordenador y por otro lado tenga un videojuego que dibuja millones de triángulos por segundo en pantalla. Algo se está haciendo mal. ¿Es que a la mayoría de desarrolladores no les importa el rendimiento de su aplicación? La mayoría de la gente que he conocido no es capaz de razonar sobre el rendimiento de una aplicación, de un algoritmo. No saben, ni quieren saber de las cosas que pasan a bajo nivel.

No entiendo el miedo que veo en muchos desarrolladores cuando se trata de temas de bajo nivel. Los punteros no muerden. A menudo se piensan que no se tienen que preocupar por estas cuestiones que sus lenguajes de alto nivel les abstraen de esos detalles. Sin embargo, esos detalles son igual de prevalentes en todos los niveles. La diferencia está en cuantos detalles están bajo tu control. Y la diferencia en los resultados es abismal.

La falta de preocupación de todos estos aspectos cristaliza en el desarrollo web. Que con lo que me han dado a probar en la carrera y con lo que he trabajado en ello he llegado a aborrecer. Hay una combinación desastrosa entre los tiempos de iteración por las malas herramientas y un afán por usar lo último de lo último que todavía no funciona del todo ni ha sido probado lo suficiente. Además es un área que cuando algo no funciona lo que hace es poner capas y capas encima, parches y parches en vez de corregir los problemas de base. Lo cual, de por sí, ya me parece una actitud fundamentalmente errónea.

Todo es más complejo de lo que debería ser. En el mundo del desarrollo se glorifican las herramientas no la solución; las metodologías y los paradigmas sobre la solución; se cree que añadiendo más se consigue algo mejor; no se entiende lo que está pasando; se valora la complejidad en vez de la simpleza; tiene poco de ingeniería. Y todo esto me frustra, siento que el propio sistema que quiero desarrollar me pone trabas, no me deja expresar lo que quiero, me limita y no me da el control que quiero. No me veo haciendo nada de esto por un periodo largo de tiempo.

Recomiendo mucho estos links. Si tienes tiempo, échales un vistazo:

Thirty Million Line Problem:
https://www.youtube.com/watch?v=kZRE7HIO3vk

When programming became a chore:
https://itnext.io/when-programming-became-a-chore-a0be2b9919c0

NPM & left-pad: Have we forgotten how to program:
https://www.davidhaney.io/npm-left-pad-have-we-forgotten-how-to-program/

The mess that is the web:
http://www.nilunder.com/blog/2013/04/15/the-mess-that-is-the-web/

Package manager mess:
http://www.nilunder.com/blog/2016/03/27/package-manager-mess/

“We don’t really care how long it takes”:
https://www.yosoygames.com.ar/wp/2014/10/we-dont-care-really-how-long-it-takes/

Twitter and Visual Studio Rant:
https://www.youtube.com/watch?v=GC-0tCy4P1U

Why we at $FAMOUS_COMPANY switched to $HYPED_TECHNOLOGY:
https://saagarjha.com/blog/2020/05/10/why-we-at-famous-company-switched-to-hyped-technology/

Casey on Software Quality:
https://www.youtube.com/watch?v=6azav9sXK_4

Propiedad intelectual, Riesgos

Riesgos, riesgos, riesgos, … De eso va la auditoría, no?

Hemos visto durante los primeros artículos la enorme importancia que tienen las propiedades intelectuales. Cómo son la base sobre la cual las empresas se apoyan, las distingue y las hace competitivas frente al resto. Es lo que marca la diferencia entre los mejores y el resto.

Como todo lo que es valioso, a las empresas les interesa protegerlo y cuidarlo. Las empresas u organizaciones al estar compuestas por personas los riesgos de los que se debe proteger a las propiedades intelectuales se pueden distribuir en dos categorías básicas: Que no la roben y que no se pierda.

Un riesgo muy habitual en organizaciones de cualquier tamaño es la infrautilización de alguna PI. Es común encontrarse que se está pagando un servicio al que no se le está dando uso; perder documentos al marcharse la gente del equipo; olvidar donde se encuentran ciertos documentos; olvidar que se tiene un dominio web porque no se le da uso; no hacer uso de las herramientas legales para proteger las PI; o, directamente, no saber que se tiene alguna PI.

Estos son riesgos internos de la organización. Son cuestiones que normalmente perjudican a la organización por no aprovechar bien sus recursos, pero el impacto suele ser pequeño en comparación con los riesgos externos.

Los riesgos externos de la PI es que sea robada. Hablamos de casos en los que la competencia logra información delicada o consigue cierta tecnología eliminando la ventaja competitiva con la que contaba la primera empresa. Cuando se trata de información delicada en vez de tecnología esta puede ser usada para adelantarse y perjudicar las estrategia de la primera empresa.

La mayoría de los robos no se suelen dar mediante planes muy sofisticados, basta con que haya algún empleado descontento con acceso a los documentos de interés.

Cuanto más puntera y más grande sea una empresa más cuidado tiene que tener una empresa. A estos niveles la competencia y los participantes en estas artimañas empiezan a tener escala internacional, países que para mejorar su competitividad usan recursos del país para robar tecnología y otros datos a grandes organizaciones para potenciar empresas del país.

Aunque parezca de película no es tan raro encontrarse noticias como la siguiente [1] en la que grupos de hackers organizados por un gobierno se dedican a robar información y tecnología a empresas para tener esa tecnología en el país sin tener que adherirse a las leyes internacionales y de esa manera poder

competir con potencias extranjeras.

Las cuestiones que dificultan el buen estado de las PI dentro de una organización son principalmente el tamaño de esta. Empresas grandes, internacionales, se encuentran con una complejidad enorme a la hora de gestionar sus PIs y las PIs de las que hacen uso. Cada país tiene leyes ligeramente distintas en cuanto a las PIs, y mantener el control y gestionar las interacciones de todas estas leyes se puede volver infernal.

Lo más habitual es que cuanto mayor es la organización, mayor sea el número de propiedades intelectuales que maneja. Además de tener más PIs, disponer de una plantilla mucho mayor aumenta la superficie de ataque considerablemente. Sinceramente la mayor fuente de riesgos en las organizaciones son las personas.

Por ejemplo, pensemos en una empresa que se dedica a la producción de software. El equipo de desarrollo tendrá seguramente todo el código y conocimientos en un repositorio privado al que solo unas pocas personas tengan acceso. De esta manera controlar quien puede y no puede acceder a esa información es sencillo.

Sin embargo, si la PI de una empresa son los procesos, contenido creativo, planos, o documentación que no está en un lugar controlado, sino que la información está a la vista o en los portátiles personales de los empleados, que no están gestionados por la empresa, cuanto más difuso sea qué constituye una PI más complicado es gestionar y protegerla.

A esto hay que añadir que cuanta más gente tenga acceso a la información más difícil es protegerla. La probabilidad de empleados descontentos o poco concienciados aumenta. Y las personas son fácilmente influenciables, basta con un poco de ingeniería social para robar información privilegiada, no hace falta ni que el ataque sea especialmente sofisticado. El sistema es tan seguro como el eslabón más débil.

La explosión combinatoria del número de empleados, gente con acceso a la PI y la dificultad para definir y limitar el acceso a las PI hace que la superficie de ataque y consecuentemente el riesgo que corren las PI sea enorme.

Referencias

[1] <<An Unfair Advantage: Confronting Organized Intellectual Property Theft>>, ASIS, consultado el 01/11/2020 https://www.asisonline.org/security-management-magazine/articles/2020/07/an-unfair-advantage-confronting-organized-intellectual-property-theft/

[2] <<Intellectual Property Protection High Tech's Crown Jewels>>, ISACA, vol 3 (2018): 39-44

[3] <<Intellectual Property Theft/Piracy>>, FBI, consultado el 02/11/2020 https://www.fbi.gov/investigate/white-collar-crime/piracy-ip-theft

[4] <<Five insights on cyberattacks and intellectual property>>, Deloitte, consultado el 02/11/2020 https://www2.deloitte.com/us/en/pages/advisory/articles/five-insights-on-cyberattacks-and-intellectual-property.html

---

Propiedad intelectual, relevancia

En el post anterior hemos hablado sobre qué es la propiedad intelectual y algunas de las maneras mediante las cuales podemos proteger derechos que tenemos sobre estas. Es

innegable el valor que aportan las PIs a las empresas. Es más, son los elementos que más valor aportan actualmente.

Es difícil cuantificar el valor de una PI, pero el valor de estas se vuelve muy claro cuando se realiza la venta de alguna de estas. Por ejemplo, fue muy sonado el desorbitado precio que pagó Disney por una de las sagas de películas más queridas de la historia. Star Wars se vendió por 4.050 millones de dólares. Un juego que seguro que también os suena se vendió por 2.500 millones de dólares. La marca Harry Potter, el mayor fenómeno literario de los últimos años, tiene un valor estimado de 25.000 millones de dólares.

En el mundo de la informática, empresas que se dediquen al desarrollo de software deben prestar mucha atención a las diferentes licencias que tienen las librerías, frameworks u otro software que utilicen. Algunas de estas licencias no son compatibles entre sí y es posible que haya grupos de herramientas que no se puedan usar conjuntamente en el mismo proyecto, o que en caso de hacerlo nos exijan licenciarlo de una manera específica.

Hay que tener mucho cuidado también con cómo son esas licencias. Es posible que partes de una tecnología tengan una licencia y otras partes tengan otra distinta. Por ejemplo, es bastante sonado el caso Google vs Oracle [4], en el que Google intenta defenderse de de una acusación de violación de copyrights por haber usado la librería estandar de Java para Android. En este caso Oracle, Sun en su día, promocionan el lenguaje como abierto o libre, sin embargo, la librería estándar de Java está protegida por copyright. Esto de licenciar de manera distinta partes de una tecnología hace muy confuso entender si se está haciendo un uso correcto del código o no. Lo mejor en estos casos es conocer bastante bien la legislación vigente. En resumidas cuentas se puede decir que la cuestión que va a determinar el tribunal superior de justicia de los Estados Unidos es si las APIs se pueden copiar o no.

Si bien su relevancia es cada vez mayor, no he encontrado muchos marcos, guías o estándares para gestionarlas. Tenemos algunas ISOs [3] y un montón de estándares por la WIPO (World Intellectual Property Organization) [2]. Casi todo gira entorno a identificar toda la propiedad intelectual que haya en una empresa, porque es bastante complicado saber qué se tiene, y sobre cómo sacarle valor a esta.

Una parte importante de la auditoría es conocer la legislación. Como en todos los casos, este es un tema complicado. Hay muchos organismos que se encargan de gestionar los derechos de las Propiedades Intelectuales. Varían entre países y en ocasiones pueden ser contradictorias. Para simplificar un poco el tema vamos a echarle un vistazo a la ley española a ver qué dice sobre la propiedad intelectual [1].

Los primero artículos definen qué es la propiedad intelectual y qué derechos se le atribuyen a su o sus creadores, quienes se les considera autores de una obra y, las diferentes clasificaciones básicas de tipos de propiedades intelectuales.

Es especialmente interesante para nosotros los informáticos y programadores el Título VII, del artículo 95 al 105 excluido, en los que se tratan los programas de ordenador en particular. Ahí se nos dice que tanto el software como toda la documentación técnica, manuales de usuario y otros documentos asociados al software están protegidos por la ley de propiedad intelectual. También debemos conocer en qué casos podemos realizar ingeniería inversa de un código de forma legítima. Podría resumirse en que solo se puede realizar si queremos crear otro programa que vaya a interoperar con el primero. Recomiendo que le echéis un vistazo al menos a esta sección que trata los programas de ordenador en particular. Una pregunta que me hago es, ¿cómo se deben interpretar estas leyes ahora que la mayoría del software está transicionando hacia un modelo de servicio?

Todo este embrollo legal es complicado, hay múltiples legislaciones que no tienen porque seguir las mismas directrices o clasificar las PI de la misma manera. Pero tan complicado como esto es identificar todas las propiedades intelectuales de las que dispone una empresa y el valor que estas aportan, identificar los riesgos y controlar las PI que maneja.

Referencias

[1] <<BOE>> núm. 97, de 22/04/1996.
Entrada en vigor:23/04/1996
Departamento: Ministerio de Cultura
Referencia: BOE-A-1996-8930
Permalink ELI: https://www.boe.es/eli/es/rdlg/1996/04/12/1/con

[2] <<List of WIPO Standards, Recommendations and Guidelines>>, consultado 20/10/2020
https://www.wipo.int/standards/en/part_03_standards.html

[3] <<03.140 PATENTS. INTELLECTUAL PROPERTY>>, consultado el 20/10/2020
https://www.iso.org/ics/03.140/x/

[4] << Oracle vs Google: en juego el derecho de autor y el desarrollo de software>>, consultado el 20/10/2020
https://www.brandsprotectionnews.com/oracle-vs-google-en-juego-el-derecho-de-autor-y-el-desarrollo-de-software/

Todos los proyectos se

retrasan

Me resulta fascinante cómo se nos dá tan mal a los seres humanos estimar el tiempo. Tengo la sensación de que en el mundo del software este efecto es mucho más pronunciado. Seguramente sea porque es el ámbito que conozco. Tal vez sea la naturaleza del software. Nunca sabes si vas a tener algún bug, y si lo tienes cuanto vas a tardar en arreglarlo. En esos casos, ¿qué estimación debes tomar? ¿El caso en el que va todo bien, en el que hay un fallo pero lo resuelves rápido o el peor de los casos en el que se te complica y tardas una semana en encontrar el fallo?

Basándose en su experiencia personal Erik Bernhardsson explica en su blog, con un análisis estadístico de datos que ha ido recogiendo, la razón de que esto sea así. No es necesario conocer la matemática necesaria para inferir la conclusión. El resultado es que las personas somos bastante buenas estimando cuánto tardaríamos la mayoría de las veces. Sin embargo, y aquí está lo interesante, es que la "mayoría de las veces" no tiene por qué corresponderse con la media de las cien veces. Las veces que tardas 3 veces más no se contrarrestan con las que tardas 3 veces menos. Si normalmente tardas 1, 3 + 0,333 = 3,333 si dividimos entre 2 la media nos da 1,666, bastante lejos de la "media" de 1 que habíamos calculado.

Además, algo que no se suele mencionar cuando se trata de proyectos, es el efecto que tiene el número de personas participantes en el proyecto. Independientemente de si tenemos una planificación más ágil o más tradicional es raro que se repare en el "delay" que introduce tener muchas personas en el proyecto.

Programar como su nombre nos indica tiene mucho de esto de planificar. Al fin y al cabo programar consiste en organizar sistemas, clases, funciones o instrucciones para que transformen una entrada en la salida deseada. Este paralelismo

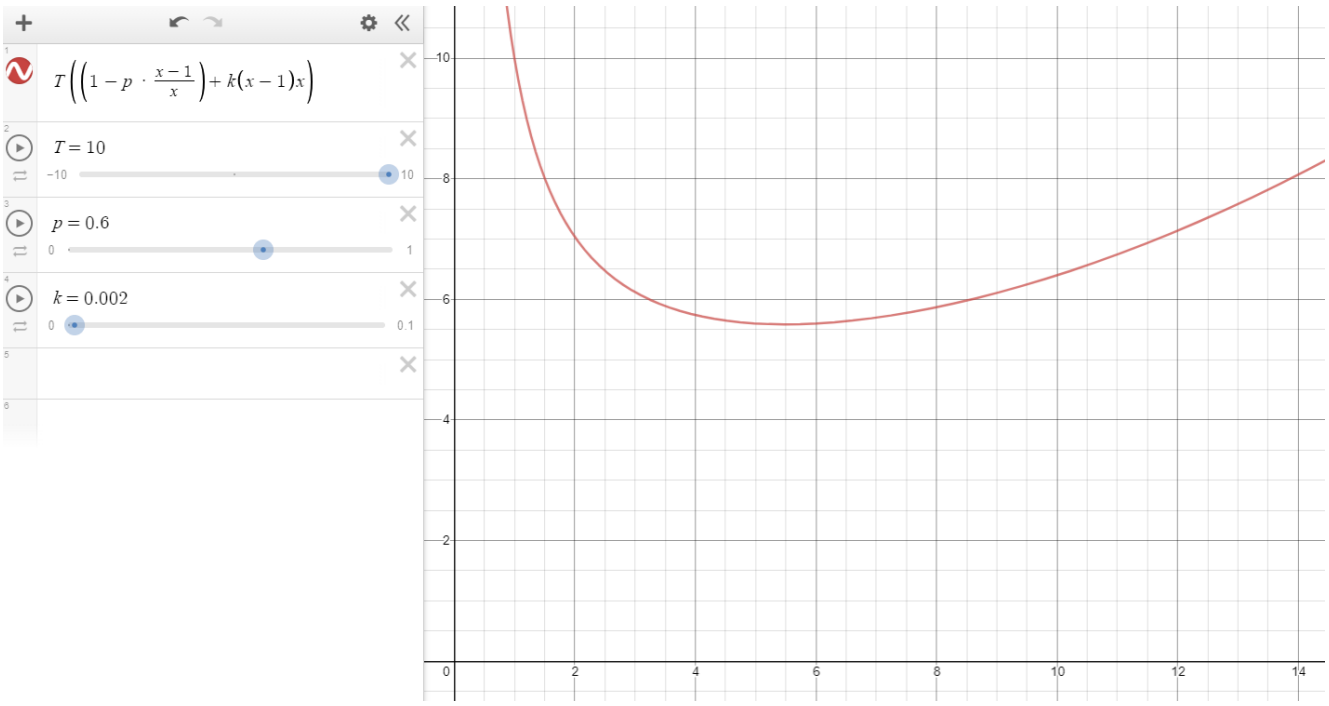
va más allá, los proyectos son una colección de tareas dependientes que tenemos que organizar. Algunas tareas son seriales y otras paralelas.

Si alguno tiene experiencia programando sistemas concurrentes sabe que las tareas no se vuelven infinitamente más rápidas cuantos más hilos de ejecución usen. Al principio sí, se cumple que la mejora es casi lineal, pero cuando introduces demasiados hilos estos se empiezan a molestar entre sí y el rendimiento decae rápidamente. Esto se traduce a que introducir más gente en un proyecto puede retrasarlo.

La explicación es bien sencilla (si has llegado hasta aquí entiendo que no te asustan las mates de la ESO). Supongamos que tenemos una tarea, proyecto, … que tarda T unidades de tiempo, si la tarea no es trivial (y porque si no no nos sirve para el ejemplo) vamos a asumir que tiene una parte paralelizable y otra serial. La parte serial no se puede hacer más rápido por mucha gente que metamos, la paralela sin embargo sí. Si a esta parte asignamos 2 personas en vez de una la tarea paralelizable durará la mitad, con 3 un tercio, etc.

Vamos a juntar la parte serial y la paralelizable. El nuevo tiempo de la tarea será el tiempo original menos lo que ahorremos de la parte paralelizable:
$T\ (1 - p\ (1 - 1/n))$.

Y a esto le tenemos que sumar el coste de comunicación multiplicado por el número de comunicaciones. Si todos hablan con todos la función de comunicación será (n-1)*n.



En la gráfica vemos una tarea de 10 unidades de tiempo, de la cual el 60% es paralelizable y el coste de comunicación es 500 veces más pequeño que el coste de la tarea. Aun así vemos cómo a partir de 6 personas el tiempo para completar la tarea aumenta.

Lo cierto es que, exceptuando el tiempo y el ratio paralelizable que son intrínsecos a la tarea, todo lo demás es decisión nuestra. Entonces para hacer que un proyecto vaya más rápido tenemos 3 alternativas. La primera, si no hemos saturado el sistema, podemos añadir más gente. La segunda, si tenemos demasiada gente en el proyecto, reducir el equipo de trabajo puede ayudar. Y la tercera, podemos organizar las comunicaciones de tal manera que se necesiten menos comunicaciones, dividiendo en equipos más pequeños, o reduciendo el coste de las comunicaciones.

Aquí tenéis la gráfica con la que podéis jugar, https://www.desmos.com/calculator/vgqyv2vb9c

Está en mi naturaleza ser analítico y matemático, no lo puedo evitar. Aun así, no creo que estos conceptos matemáticos puedan decirnos cómo tenemos que gestionar proyectos, la realidad y las personas son muchísimo más complejas.

Simplemente creo que modelos adecuados pueden hacernos entender o darnos cuenta de cuestiones que no siempre somos capaces de expresar.

Enlaces de interés

- Why software projects take longer than you think: a statistical model
  https://erikbern.com/2019/04/15/why-software-projects-take-longer-than-you-think-a-statistical-model.html
- Agility != Speed
  https://www.youtube.com/watch?v=kmFcNyZrUNM
- You weren’t meant to have a boss
  http://www.paulgraham.com/boss.html

Propiedad intelectual, introducción y contexto

La propiedad intelectual es uno de los mayores activos de las empresas actuales. La edad de la información se caracteriza principalmente por otorgar mucho valor a los datos y las ideas que surgen de las personas. Estos elementos intangibles incluyen: dibujos, formas, nombres, imágenes, obras literarias y artísticas, símbolos, modelos, fórmulas, etc. Manejar y controlar el uso y conocimiento de estas es muy complicado. ¿Cómo controlas algo que no se puede tocar?

Para proteger la explotación de estos recursos existen leyes que categorizan y otorgan ciertos derechos según el tipo de propiedad intelectual. Estas leyes varían según el organismo gubernamental. En general nos podemos encontrar con los siguientes [1]:

- Patentes: El gobierno otorga el derecho de explotación exclusiva de un producto, generando un monopolio artificial para la empresa o creador de la invención, a cambio de la divulgación del producto.
- Modelo de utilidad: Es un derecho que otorga el estado a una invención. Es muy similar a una patente, con la diferencia de que un modelo de utilidad no tiene que ser tan “novedoso” u “original” como una patente. Versiones, actualizaciones o extensiones de un producto que suponen una ventaja competitiva entran en esta categoría. A cambio la exclusividad sobre el producto es menor que en la patente.
- Derechos de autor: Son unos derechos que obtiene el autor de una obra artística, científica o didáctica, por el simple hecho de ser el autor. Aquí podemos

encontrarnos los derechos de copia (copyright) que especifican quién y cómo puede realizar copias de las obras. Expiran o pasan al dominio público después de muchos años. En la mayoría de los países ocurre pasados los 50 años, en muchas partes de Europa son 70 años y en México 100.

- Propiedad industrial: Una propiedad industrial puede ser una marca, símbolo, patente, dibujo o diseño industrial. Sobre estas el estado otorga el derecho decidir quién puede utilizar la invención, diseño o signo distintivo y a prohibir que un tercero lo haga.
- Marcas registradas o *trademarks*: Estas están compuestos por formas, dibujos, símbolos, iconos, logotipos, música, (olores en algunos países). Son elementos principalmente gráficos que un consumidor asocia a una marca. Y las instituciones gubernamentales otorgan a su titular, la posibilidad de autorizar o prohibir el uso de la misma a terceras personas.

También hay otros tipos de PI que son curiosas y/o destacables:

- Variedades vegetales: Como se puede intuir por el nombre, este tipo de propiedad otorga derechos de de uso comercial sobre una variedad de plantas.
- Imagen comercial: La imagen que proyecta una marca o empresa. La sensación, estilo, *feeling* que percibe un cliente también se puede considerar propiedad intelectual. Pero esta no está protegida por ninguna ley y, el cuidado de esta es puramente de la empresa. Tiene mucho valor, se puede dañar fácilmente y tiene mucho impacto en los clientes.
- Secreto comercial: Son fórmulas, prácticas, procesos, diseños, instrumentos, patrones o compilaciones de información que no se conoce o no se puede determinar de manera razonable, mediante la cual una empresa puede obtener una ventaja económica. Estos conocimientos se

podrían proteger mediante una patente pero en ese caso se deben publicar los detalles y en unos 20 años se perdería la exclusividad.

Cómo gestionar y controlar las ideas y la información es difícil, hay que tener localizada e identificada toda aquella información que sea relevante para la empresa. Lo que un trabajador aprende durante su jornada laboral, ¿le pertenece a él o a la empresa? Si este empleado se marcha a la competencia, ¿qué les puede contar? ¿Cómo de parecidos tienen que ser dos cosas para considerarse plagio? ¿Cuánto para que sea un producto distinto? ¿Cómo estimar cuánto vale una PI?

Si bien las leyes se centran especialmente en productos que se pueden vender las propiedades intelectuales también incluyen documentos internos de las empresas como las buenas prácticas, formularios u hojas de procesos entre otros. Todo el conocimiento implícito de una organización procesos, organización, experiencia y habilidades de los empleados es conocimiento y puede suponer un valor muy elevado.

Para que nos hagamos una idea del volumen que suponen las propiedades intelectuales durante el año 2018 en España se realizaron más de 80.000 solicitudes. Más de 50.000 de marcas, 18.000 diseños industriales, 12.000 nombres comerciales y 5.000 modelos de utilidad, patentes y expedientes [2].

[1] <<¿Qué es la propiedad intelectual?>>, OMPI, consultado el 30/09/2020, https://www.wipo.int/edocs/pubdocs/es/intproperty/450/wipo_pub_450.pdf

[2] <<La OEMP en cifras>>, OEMP, consultado el 02/10/2020, https://www.oepm.es/export/sites/oepm/comun/documentos_relacionados/Publicaciones/Folletos/La_OEPM_en_Cifras_2018.pdf