Buenas prácticas en un desarrollo

Sugerencias y beneficios

|  |  |
| --- | --- |
| Autor | Samuel Jesus de Paul Barrera Bastidas |
| Fecha | 14 de junio del 2017 |
| Propósito | Taller de habilidades de comunicación oral y escrita |
| Profesor | María Francisca Carrasco Retamal |
| Sección | EH102IECIRE/11 |

Contenido

[Introducción 3](#_Toc485763078)

[Presentación 4](#_Toc485763079)

[Objetivos 4](#_Toc485763080)

[Metodología 4](#_Toc485763081)

[Para el Lector 4](#_Toc485763082)

[El Asesino que se encuentra tras tus pasos 5](#_Toc485763083)

[Divide et impera 7](#_Toc485763084)

[¿Y cómo evitamos la extinción de nuestros proyectos? 9](#_Toc485763085)

[I.- Los individuos 9](#_Toc485763086)

[II.- El equipo 9](#_Toc485763087)

[III.- La organización 10](#_Toc485763088)

[IV.- El producto 11](#_Toc485763089)

[Juramento del Programador 12](#_Toc485763090)

[Pastelero a tus Pasteles 14](#_Toc485763091)

[Conclusión 17](#_Toc485763092)

[Bibliografía 18](#_Toc485763093)

[Glosario 19](#_Toc485763094)

[ANEXOS 22](#_Toc485763095)

[San Saru, La leyenda de los tres monos sabios 23](#_Toc485763096)

[Lectura recomendada 25](#_Toc485763097)

[Principios del Manifiesto Ágil 26](#_Toc485763098)

Introducción

El cómo afrontar un nuevo desarrollo informático siempre tiene múltiples aristas, suelen recorrerse diferentes caminos para lograr concretar un proyecto, en esto influye la experiencia del equipo de trabajo y las herramientas que usan. La forma en que las acciones se vayan realizando afectaran no solo al desarrollo del proyecto, sino a su resultado integral, es en esto donde debemos reparar en el cómo deseamos que estas acciones se lleven a cabo para tener un resultado óptimo en el tiempo.

Antes de iniciar nuestro proyecto debemos preguntarnos:

* ¿Cómo equipo estamos siendo eficaces para alcanzar nuestro resultado?
* ¿Estamos haciendo uso de métodos, técnicas o actividades promuevan la eficiencia?
* ¿Conozco los principios o guías que se encuentran consensuados en la comunidad?, ¿Los aplico?

Es muy posible que usted este logrando hacer la labor que tiene propuesta, el tema es si ha considerado si puede mejorar...

El siguiente documento comenta algunos lineamientos sencillos pero prácticos respecto al que considerar en un nuevo desafío informático.

Presentación

El presente informe tiene el propósito de ofrecer al lector un grupo de prácticas, experiencias y herramientas, que en su conjunto y siendo apropiadamente ocupadas tienen efectos positivos en un proceso de continuidad de un desarrollo informático, pero para aclarar, esta es solo una guía, son sugerencias que el mismo lector deberá valorar y ver cuáles son realmente apropiadas para su problemática; Se debe entender que nada se encuentra en piedra y lo que puede ser un consenso, o considerado bueno, solo el día a día de la práctica dictará lo apropiado, lo más relevante es que se acomode a su labor.

Objetivos

El principal objetivo es la eficiencia en el resultado de la labor que el lector emprenda, muchas de estas prácticas son de consenso en la comunidad, pero lo más relevante, es que por medio de trabajo cómodo se consiga un trabajo que posea continuidad y calidad.

Metodología

Para la confección de este informe se dispuso de investigaciones y experiencias hechas en el marco de proyectos agiles y buenas prácticas en la programación, muchas de ellas basadas en el manifiesto de la artesanía de software, pero también cabe aclarar que es un informe interpretativo; Lo que debe rescatar es la experiencia de quien lo lea y cómo puede llegar a usar estas consideraciones.

Este documento se encuentra enfocado en el mundo de la artesanía de software y quienes se encuentren en la cruzada de buscar una mejora constante a su labor.

Para el Lector

Confío en que las impresiones que se dejaran en este documento le resulten de utilidad, pero más que un texto inerte genere conversación entre sus pares y contra su fuero interno, somos criaturas perfectibles y nuestra labor no es ajena de esta condición. Me resultaría muy gratificante si temporalmente pudiese retroalimentar este documento con sus consideraciones o experiencias.

El Asesino que se encuentra tras tus pasos

Un motivante en las buenas prácticas es entender este principio que ha pasado generación tras generación:

Programa siempre como si el tipo que acabe manteniendo tu código fuera un psicópata violento que sabe dónde vives.

**Martin Golding**

Esta jocosa frase, hace referencia a que nuestra codificación debe resultar clara para quien asuma la labor de mantenerla en un futuro. Esto aplica en todo lo que hacemos, desde la definición de los espacios de nombre a las variables que ocupamos.

Es usual que nuestro trabajo no sea para nosotros, así que lo ideal es antes de comenzar conocer cuáles son las convenciones en la nomenclatura de los objetos que generamos, dependiendo de la madurez de la institución en la cual prestemos nuestro servicio es sí dispondrán o no esta especificación, en la situación de que no dispongan de ella, lo ideal es evitar la creatividad natural que nos puede invadir, lo mejor es acudir al consenso de la comunidad o de alguna institución representativa, por ejemplo:

* [Convenciones de nomenclatura Microsoft](https://msdn.microsoft.com/es-es/library/cc467490.aspx)
* [Convenciones de nomenclatura Wikipedia](https://es.wikipedia.org/wiki/Convenci%C3%B3n_de_nombres_(programaci%C3%B3n))
* [Convenciones de nomenclatura Javascript](http://snowdream.github.io/javascript-style-guide/javascript-style-guide/es/naming-conventions.html)
* [Convenciones de nomenclatura IBM](https://www.ibm.com/support/knowledgecenter/es/SSLVQG_7.0.1/datacollection_survey_reporter_ddita/datacollection/reporter/xml/naming_conventions.html)

Lo importante de estas convenciones es evitar situaciones como la siguiente:

function a (b) {

var c = b + 3;

alert(c);

};

Qué es lo no apropiado en el código que se puede apreciar, pueden responder las siguientes preguntas:

* ¿Cuál es el propósito de la función?
* ¿Qué tipo y propósito tiene el parámetro de la función?
* ¿Cuál es la intención de la variable llamada c y que implica la adición del número 3?

Para alguien con alguna experiencia puede tender a suponer qué es y qué hace cada elemento de esta función, pero esta acción es interpretativa, por ende, factible de un error interpretativo. La idea es que nuestro código sea universalmente claro, hay quienes recomiendan que la codificación se haga en un lenguaje más común, como por ejemplo el inglés, así obtenemos un código algo más cosmopolita, pero esto debe ser un acuerdo con nuestro cliente y su cultura en los desarrollos.

Una frase para cerrar este punto:

Cualquier tonto puede escribir código que un ordenador entiende. Los buenos programadores escriben código que los humanos pueden entender.

**Martin Fowler**

Divide et impera

Esta frase se le asocia a Julio Cesar, pero no la verdad es que se desconoce el origen real, hoy en día en el mundo de programación este refrán toma fuerza en las lógicas de atender la resolución de un algoritmo. La experiencia nos dice que la estrategia de seccionar un problema en pequeñas fracciones lo vuelve más manipulable o incluso con una resolución obvia, pero dudo que esto sea ajeno a usted, el tema aquí es:

* ¿Debemos atender cada sección de este problema?
* ¿En qué orden atendemos estas secciones?
* ¿es realmente necesario cambiar la mecánica de estas secciones?

En este tema tenemos muchas aristas que atender, cuando iniciamos un desarrollo tenemos que optar por una determinada arquitectura y herramientas, incluso alguna tecnología específica, en muchos casos uno tiene la tendencia a optar por lo más moderno y vigente, pero nuevamente debemos detenernos y pensar en nuestro cliente, adecuarnos a su realidad y cómo afronta este desarrollo.

Si conoces al enemigo y a ti mismo, no debes temer el resultado a un ciento de batallas.

**Sun Tsu**

Es fundamental conocer nuestro cliente, sus ambientes, nuestros usuarios y las habilidades de nuestro equipo, con esto en mente, debemos optar en consenso por un Patrón de diseño y las herramientas que usaremos.

Dentro de estas consideraciones me gustaría citar los principios del manifiesto de la artesanía de software:

* No sólo software que funciona, sino también software bien diseñado
* No sólo responder al cambio, sino también agregar valor constantemente
* No sólo individuos e interacciones, sino también una comunidad de profesionales
* No sólo colaboración de clientes, sino también asociaciones productivas

**Manifiesto Software Craftsmanship**

De estos puntos destacó los dos primeros para este tema, un diseño inteligente proporciona flexibilidad al cambio de nuestro desarrollo, esto ofrece la opción de ser ágiles antes los cambios, y también el detalle de que nuestras entregas no solo sean la simple resolución de un problema, sino que sean un valor para nuestro cliente, aquí está la respuesta a la pregunta ¿En qué orden atendemos estas secciones?, al conocer a nuestro cliente, debemos ser capaces de entender cómo aportar con nuestro desarrollo y sumar a la solución.

¿Y cómo evitamos la extinción de nuestros proyectos?

Al enfrentar un desarrollo, debemos considerar los riesgos en los cuales nos embarcamos, uno de ellos es que no logremos llegar a puerto, dudo que existe la fórmula mágica más que mucha experiencia y una fuerte capacidad al cambio, pero enumero cuatro puntos que pueden ser una buena base a considerar durante un desarrollo.

I.- Los individuos

Es necesario una permanente autocrítica y una evaluación del proyecto al fin de cada entrega, un continuo seguimiento de las tareas pendientes. Esto implica que cada individuo del proyecto debería saber claramente que mantener, que arreglar y que probar, a esto le podemos sumar la necesidad de entender cuáles de esos aspectos que se mencionaron provocaron frustración, enojo o alegría...

Este análisis debe ser algo sobre lo cual trabajar y velar constantemente, el detalle de este punto, es contar con un equipo maduro, donde se haya formado una confianza mutua y un ambiente de cordialidad, donde los integrantes comprendan que se critica las actitudes específicas de una persona y no a la persona. Esto claramente no es sencillo, pero no es imposible, lo relevante es enfocarse en trabajar este objetivo de manera individual y grupal.

II.- El equipo

Un equipo eficiente es el fruto de una formación, de una fina decantación del tiempo que produce una extraña gema, pensar en alterar este ecosistema de alguna manera debe ser algo altamente meditado, de hecho, debería ser una de las máximas de cada área llegar a un equipo de alto rendimiento que opere feliz y en condiciones, ya que es el motor de todo desarrollo que implementemos.

Si bien es normal que en estos equipos se manifiesten roles o responsabilidades, es fundamental que en la mecánica diaria este equipo actué de manera horizontal, me refiero con esto a que todos sean iguales, todos deben entender que son vitales, que todos pueden fallar y posiblemente lo hagan; Y que deben estar dispuestos a mejorar.

Otra cosa que ayuda mucho es tener equipos multidisciplinarios, donde se tiene especialistas o responsables de sub-temas específicos, es ideal contar con un agente "Facilitador” de esta interacción, una buena práctica es armar equipos de dos o más personas para trabajar simultáneamente en un tema, si bien puede parecer un gasto o desperdicio a la larga se consigue mejores resultados.

Los temas económicos es mejor atenderlos con una visión de equipo más que a nivel personal, más en esta figura y no como un incentivo específico, sino como una condición constante, también se recomienda que más que simplemente motivar a los individuos, se les debe ofrecer desafíos a sus habilidades, y aunque pueda leerse extraño, controlar el afán o necesidad de sobresalir de los individuos, procurar el pensamiento comunitario, promover la generosidad entre y por el equipo.

III.- La organización

Para que el equipo pueda llegar bien a puerto, no basta que sea el mejor equipo del mundo, requiere que la organización en la que se encuentre inmersa esté comprometida de la misma forma, en caso de no ser así, ver la forma de fidelizar a las contrapartes, ser evangelizadores de un buen ambiente.

Es posible que cada equipo piense en usar alguna metodología de trabajo, lo importante es ser fiel al pie de la letra de ella, no "Nacionalizar" esa metodología y hay que ser muy puristas en esto, la adaptación por más cómoda que se pueda visualizar es un simple ¡Riesgo!, de esta forma es necesario evaluar si realmente es apropiado optar por tal o cual metodología...

Independiente de la metodología por la cual se opte, se debe generar una confianza mutua entre las partes, pese a una posible ausencia de formalidad, para ello es vital tener métricas de todo, esto nos permitirá disminuir incertezas y así generar confianza y seguridad en el proceso.

Se recomienda dividir el proyecto en secciones que aseguren el "Producto mínimo viable", esto pensando en constantes mejoras para así ofrecer una real oportunidad frente a posibles cambios, para ello es obligatorio disponer en el equipo de alguien que maneje completamente el proceso y que cada individuo del equipo conozca lo que esté haciendo cabalidad.

IV.- El producto

En muchos sentidos somos artesanos, verdaderos artistas que generamos obras de arte colectivas, pero es necesario que cada producto en el cual estemos involucrados, genere un valor, sea de utilidad tangencial y esto es algo que puede leerse obvio, pero la verdad no lo es, alguien debe velar por estas mínimas...

Es necesario no caer en la tentación del encantamiento tecnológico, siempre es mejor velar por el camino seguro, si bien la creatividad es bien recibida, es el producto a lo cual debemos acomodarnos, es decir a lo necesario y solicitado.

Junto con el producto, lo normal es documentar cómo y qué se hizo, para esto existen mil formas, pero lo mínimo que se debería considerar en esta documentación son las justificaciones de los lineamientos que se tomaron, justificar por qué se optó por tal camino y no por otro, las dependencia y los artefactos usados y lo más importante, no documentar lo que no estará actualizado, eso es solo una pérdida de tiempo.

Y algo fundamental pero que es parte del inicio del proceso, acordar de manera clara con el cliente cómo se evaluará la calidad del producto, eso les quitara un asteroide de encima.

Juramento del Programador

Esto es una referencia al aporte de Tío Bob (Robert C. Martin), creo que es algo que uno debe tener en consideración a la hora de afrontar la vida en este rubro:

* No produciré código dañino.
* El código que produzca será siempre el mejor posible. No entregaré a sabiendas código defectuoso en su comportamiento o en su estructura.
* Realizaré en cada entrega una comprobación segura y repetible de que cada elemento del código hace lo que debería.
* Haré entregas pequeñas y frecuentes, para no obstaculizar el trabajo de otras personas.
* Mejoraré el código sin temor y sin descanso cada vez que sea posible. Nunca empeoraré el código.
* Haré todo lo posible para mantener mi productividad y la de otras personas tan altas como sea posible. No haré nada que reduzca esa productividad.
* Me aseguraré de forma continua que otros pueden cubrir mi trabajo y que yo puedo cubrir el suyo.
* Produciré estimaciones que son honestas tanto en magnitud como en precisión. No haré promesas sin certidumbre.
* Nunca dejaré de aprender ni de mejorar mis habilidades.

**El Blog código limpio", por Robert C. Martin (Uncle Bob)**

Estas directrices están pensadas en defender y preservar el honor de la profesión de los programadores informáticos.

Los programas deben ser escritos para que la gente los lea y sólo incidentalmente, para que las máquinas los ejecuten.

**Abelson / Sussman**

Esta frase rescata en buena medida los lineamientos expuestos, nuestra labor debe ser clara, honesta e inteligentemente sencilla, más pensada en nuestros pares que en la maquinaria que los ejecutara, pensada en nuestro cliente y en proceso de continuidad.

Pastelero a tus Pasteles

En la artesanía del software y dependiendo de donde desempeñamos nuestra labor, es posible que los roles en la institución esté o no, bien definidos, incluso en una labor particular, estos roles u obligaciones deberían estarlo.

Lo usual en que en una institución del rubro veamos una figura como la siguiente:

Esta es una figura algo idílica en muchos casos, ya que en algunas situaciones nos veremos simultáneamente llevando responsabilidades de diferentes roles, pero independiente a esto, lo ideal por un tema de continuidad, es que cada acción se encuentre apropiadamente dividida.

El **Cliente**, ya sea interno o externo deberíamos diferenciarlos a lo menos en dos tipos de roles, los usuarios que usaran la solución, denominados **usuarios finales**, y el usuario que define el qué, cómo y cuándo de nuestra solución, a él lo denominamos **Usuario líder**; Es usual que los desarrollos suelen olvidarse casi por completo de los usuarios finales hasta las pruebas y mantención (que usualmente se fijan al final del proceso) y solo se tiende a tratar con el usuario líder, esta situación es algo errada, para un apropiado desarrollo se debe trabajar a par con el usuario final, específicamente es saber que requiere y en entender cómo dar un valor a nuestra entrega para él.

Respecto al desarrollo en sí, lo ideal sería que hubiese un área encargada de velar por la prioridad de los proyectos o soluciones a implementar, esta área suele llamarse **Gestión de la demanda**, luego viene el área dura de desarrollo, usualmente liderada por un jefe de proyecto, un equipo de desarrolladores, analistas e ingenieros y claro, los diseñadores, sumamos a esto la gente que vela por los datos y su gestión, más le gente vela por la calidad de lo hecho. En este proceso, debemos pensar en cómo lo **integramos**, como **automatizamos los procesos**, los pasos a producción, la **ejecución de pruebas** y cómo se logra **transparentar el estado del proyecto**.

Aunque no siempre es el caso, las soluciones que construimos por un tema de seguridad o políticas del cliente son administradas por un tercero o un área especializada, ellos velan por dónde y cómo vivirán nuestras soluciones, llamaremos a este grupo de personas **Infraestructura**, de la misma forma, en algunas empresas disponen de una validación de la solución entregada y por motivos de aseguramiento se opta por una **Auditoria** externa (a lo menos un área diferente de desarrollo).

Hoy en día se habla de los siguientes tipos de “Programadores” en función de su alcance:

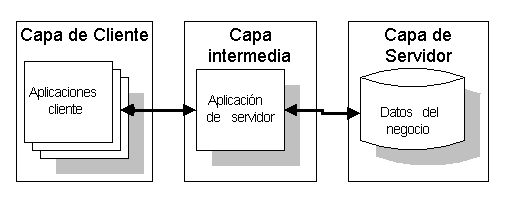
* La confección de los elementos visuales y su presentación (Diseñadores)
* La confección de la interfaz de usuario (Frontend Developers)
* La confección de la lógica de negocio (Backend Developers)

Pero debido a un tema de alcance, surgen dos tipos más:

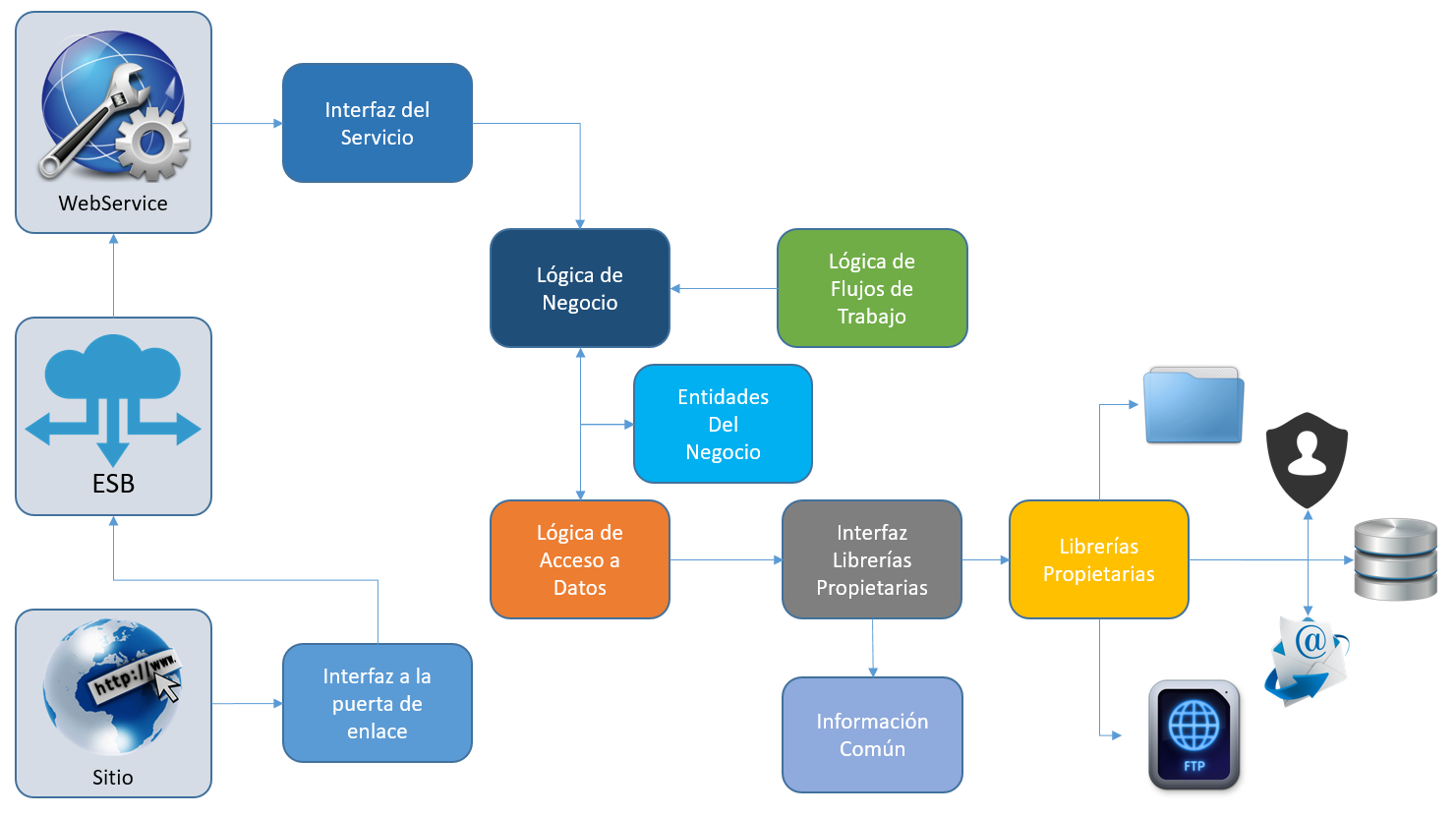
* Full Stack Developers, quienes reúnen las habilidades del Frontend y Backend
* Devops, quienes reúnen las habilidades del Backend, QA e Infraestructura.

Lo relevante de estas tipificaciones, es en el cómo llevamos a cabo nuestra labor, si bien es posible que con un determinado “Patrón de diseño” hayan resuelto el cómo afrontar el desarrollo, la programación modular del proyecto es vital, con esto me refiero a la abstracción en cada capa de desarrollo y a los responsables de ellas.

Lo usual es ver los desarrollos en las típicas tres capas:



Pero pensando en los procesos de transformación digital que hoy vive el mercado, la tendencia es a volver más atómicas estas secciones, un ejemplo lo podemos apreciar en la visión que proporciona Microsoft de este modelo de [Tres Capas](https://msdn.microsoft.com/en-us/library/ff648105.aspx), esta es una interpretación del modelo propuesto:



Conclusión

Las buenas prácticas, como lo dice su nombre, es una convención, pero es la convención que se acomode a nuestra realidad, es nuestro deber procurar mantenernos vigentes respecto a lo que la comunidad propone y tener la certeza de lo que nuestros clientes requieren, y es en función de esto, más el bagaje que tengamos lo que nos dirá qué metodología y qué herramientas deberemos usar.

Es necesario tener claro que debemos ser flexibles al cambio, las buenas prácticas o las prácticas establecidas aseguran un nivel de calidad y entendimiento entre las partes, con esto podemos establecer que la continuidad y la integración se podrán efectuar sin mayores tropiezos. Pero nuestra labor debe ser más que solo eso, debe tener nuestra firma personal impresa en ella, para esto esta la única e individual forma artística en la que nosotros anexamos un valor adicional a nuestras soluciones.

Este documento no es algo imperativo o rígido, es simplemente una exposición de experiencias y técnicas que se nutren de la implementación y adaptación, es la retroalimentación, las opiniones y criticas lo que se busca, plantar la duda de cómo mejorar, de que puedo rescatar de lo que hago y hacia donde deseo llegar.

**Bibliografía**

* Robert C. Martin. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Canada: Prentice Hall.
* Michael Feathers . (2004). Working Effectively with Legacy Code. Canada: Prentice Hall.
* Robert C. Martin . (2015). The Programmer's Oath. 2015, de El Blog código limpio Sitio web: <http://blog.cleancoder.com/uncle-bob/2015/11/18/TheProgrammersOath.html>
* Robert C. Martin. (2001). Manifiesto por el Desarrollo Ágil de Software. 2001, de Manifiesto por el Desarrollo Ágil de Software Sitio web: <http://agilemanifesto.org/iso/es/manifesto.html>
* Comunidad ágil. (2009). Manifiesto de la artesanía de software. 2009, de softwarecraftsmanship.org Sitio web: <http://manifesto.softwarecraftsmanship.org/>
* Samuel Barrera. (2017). [varios]. 2017, de Agilidad y Buenas practicas Sitio web: <https://dinosauriomoderno.blogspot.cl/>

Glosario

**Algoritmo:** En matemáticas, lógica, ciencias de la computación y disciplinas relacionadas, un algoritmo (del griego y latín, dixit algorithmus y este a su vez del matemático persa Al-Juarismi) es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la algoritmia.

**Espacio de nombre:** Técnica y correctamente definido como namespace, en su acepción más simple, es un conjunto de nombres en el cual todos los nombres son únicos. Un espacio de nombres es un contenedor abstracto en el que un grupo de uno o más identificadores únicos pueden existir. Un identificador definido en un espacio de nombres está asociado con ese espacio de nombres. El mismo identificador puede independientemente ser definido en múltiples espacios de nombres, eso es, el sentido asociado con un identificador definido en un espacio de nombres es independiente del mismo identificador declarado en otro espacio de nombres. Los lenguajes que manejan espacio de nombres especifican las reglas que determinan a qué espacio de nombres pertenece una instancia de un identificador.

**Función:** En informática, una subrutina o subprograma (también llamada procedimiento, función, rutina o método), como idea general, se presenta como un subalgoritmo que forma parte del algoritmo principal, el cual permite resolver una tarea específica. Algunos lenguajes de programación, como Visual Basic .NET o Fortran, utilizan el nombre función para referirse a subrutinas que devuelven un valor.

**Parámetro:** Un parámetro es una constante o una variable que aparece en una expresión matemática y cuyos distintos valores dan lugar a distintos casos en un problema.

**Patrón de diseño:** Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

**Software:** Se conoce como software al equipo lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos' necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. Los componentes lógicos incluyen, entre muchos otros, las aplicaciones informáticas, tales como el procesador de texto, que permite al usuario realizar todas las tareas concernientes a la edición de textos; el llamado software de sistema, tal como el sistema operativo, que básicamente permite al resto de los programas funcionar adecuadamente, facilitando también la interacción entre los componentes físicos y el resto de las aplicaciones, y proporcionando una interfaz con el usuario.

**Variable:** En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad de información conocida o desconocida, es decir un valor. El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que representa. El identificador, en el código fuente de la computadora puede estar ligado a un valor durante el tiempo de ejecución y el valor de la variable puede por lo tanto cambiar durante el curso de la ejecución del programa. El concepto de variables en computación puede no corresponder directamente al concepto de variables en matemática. El valor de una variable en computación no es necesariamente parte de una ecuación o fórmula como en matemáticas. En computación una variable puede ser utilizada en un proceso repetitivo: puede asignársele un valor en un sitio, ser luego utilizada en otro, más adelante reasignársele un nuevo valor para más tarde utilizarla de la misma manera. Procedimientos de este tipo son conocidos con el nombre de iteración. En programación de computadoras, a las variables, frecuentemente se le asignan nombres largos para hacerlos relativamente descriptivas para su uso, mientras que las variables en matemáticas a menudo tienen nombres escuetos, formados por uno o dos caracteres para hacer breve en su transcripción y manipulación.

ANEXOS

**San Saru, La leyenda de los tres monos sabios**

¿Alguna vez han visto esta imagen?, ¿Saben de qué se trata?

Proviene de la leyenda de los tres monos sabios, Kikazaru, Mizaru y Iwazaru… **“San Saru”**.

Cuenta la leyenda que tres monos fueron enviados por los dioses para delatar y castigar las malas acciones de los humanos:

* **Kikazaru, el mono sordo**, era el encargado de utilizar el sentido de la vista para observar a quienes realizaban malas acciones y comunicárselo a Mizaru, mediante la voz.
* **Mizaru, el mono ciego**, no necesitaba su sentido de la vista, puesto que tan sólo se encargaba de transmitir al tercer mono, Iwazaru, los mensajes que le pasaba Kikazaru.
* **Iwazaru, el mono mudo**, escuchaba los mensajes transmitidos por Mizaru para decidir la pena de los dioses que le caería al desafortunado humano que lo mereciese y observar que se cumpliese.

Colocados los tres monos según sus habilidades y limitaciones, obtenemos un mono que ve, otro que escucha y otro que habla. Los monos, juntos y organizados, pueden alcanzar metas que no lograrían por separado. Si bien existen varias posibilidades de colocar a los tres monos, todas ellas son situaciones de comunicación fallida o de colaboración imposible, excepto una:

**Kikazaru (sordo) > Mizaru (ciego) > Iwazaru (mudo).**

Esta idea da pie a una metodología de tomas de decisión por integración, afinidad y complemento, es una forma de contemplar desafíos o decisiones con un prisma grupal.

Lectura recomendada

1. The Phoenix Project - Gene Kim, Kevin Behr, George Stafford
2. Clean Code: A Handbook of Agile Software Craftsmanship - Robert C. Martin
3. Agile Software Development, Principles, Patterns, and Practices - Robert C. Martin
4. Por un Scrum Popular - Tobias Mayer y Alan Cyment
5. Refactoring: Improving the Design - Existing Code - Martin Fowler
6. Lean Software Development: An Agile Toolkit - Mary Poppendieck y Tom Poppendieck
7. La Meta - Eliyahu Goldratt
8. Agile Retrospectives: Making Good Teams Great (Pragmatic Programmers) - Esther Derby, Diana Larsen y Ken Schwaber
9. Agile in a Flash - Jeff Langr y Tim Ottinger
10. Coaching Agile Teams - Lyssa Adkins
11. Scrum and XP from the Trenches - Henrik Kniberg
12. Kanban and Scrum – Making the Most of Both - Henrik Kniberg y Mattias Skarin, 2010.
13. User Stories Applied: For Agile Software Development - Mike Cohn.
14. Extreme Programming Explained: Embrace Change - Kent Beck
15. Succeeding with Agile: Software Development Using Scrum - Mike Cohn
16. Crystal Clear: A Human-Powered Methodology for Small Teams Alistair Cockburn
17. Alistair Cockburn – Diana Larsen y Ainsley Nies
18. Gestión práctica de proyectos con Scrum: Desarrollo de software ágil para el Scrum Master - Antonio Martel
19. The Agile Samurai - Jonathan Rasmusson
20. Scrum, a pocket guide - Gunther Verheyen
21. The Heart of Change - John P. Kotter
22. Kanban from the Inside - Mike Burrows
23. The 5 Dysfunctions of a team - Patrick M. Lencioni
24. The Pragmatic Programmer, From Journeyman To Master - Andrew Hunt y David Thomas

Principios del Manifiesto Ágil

Estos son los principios del Manifiesto Ágil:

* Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
* Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
* Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
* Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
* Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
* El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
* El software funcionando es la medida principal de progreso.
* Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
* La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
* La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
* Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
* A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.