UNA NUEVA FORMA DE REALIZAR LA INGENIERÍA DE SOFTWARE

Autor: **profesor Augusto Vega Pinedo**

Universidad Ricardo Palma

Escuela Profesional de Ingeniería Informática

Correo electrónico: alvpurp@gmail.com

Recibido: 24/04/2018
Aceptado: 19/07/2018

Resumen

Es muy difícil conseguir que el cliente y los usuarios finales de un sistema informático queden satisfechos con su producto de software, si no han podido experimentar con el sistema lo suficiente como para darse cuenta de lo que realmente necesitan y cómo lo requieren. A su vez, el cliente tendrá que adaptar su empresa para poder seguir usando el software en el transcurso del tiempo, salvo que contrate continuamente a los desarrolladores para que transformen el sistema a fin de que vaya de acuerdo con los nuevos requerimientos.

Es imperativo que el software que se entrega a los clientes tenga la suficiente flexibilidad como para dejar que ellos mismos lo adapten, lo cual es el resultado de ciertas características tales como la experimentación. Si los clientes pueden adaptar el software, entonces también lo pueden hacer crecer y adaptarse aún más, al ritmo de las transformaciones que sufre la empresa a través del tiempo.

Esta idea presenta un reto, porque habría que darle al cliente un conjunto de herramientas para que él mismo desarrolle su sistema, en vez de darle un sistema construido ad-hoc.

Abstract

It is very difficult to get the client and the end users of a computer system to be satisfied with their software product, if they have not been able to experiment with the system enough to realize what they really need and how they need it. In turn, the client will have to adapt his company to be able to continue using the software over time, unless he continually hires the developers to transform the system so that it goes according to the new requirements.

It is imperative that the software delivered to the clients have enough flexibility to allow themselves to adapt it, which is the result of certain characteristics such as experimentation. If customers can adapt the software, then they can also make it grow and adapt even more, at the pace of the transformations of the company undergoes over time.

This idea presents a challenge, because it would be necessary to give the client a set of tools for him to develop his system, instead of giving him a system built ad-hoc.

1. Introducción

Este ensayo trata sobre una nueva forma de realizar la Ingeniería de Software, la cual le permite al ingeniero de esta especialidad establecer, muy rápidamente, sistemas informáticos que calcen perfectamente con las necesidades de las empresas y de sus usuarios finales. Esta nueva forma de ingeniería logra productos de software que crecen al ritmo de crecimiento de las empresas, con una guía mínima por parte de los desarrolladores de software.

Este enunciado, aunque simple para cualquier persona, para el ojo del conocedor demanda que los sistemas de software que cumplen con estas exactitudes y flexibilidades, sean tan complejos

que llegando a obtenerse podrían transformar la forma en que actualmente se realiza la ingeniería de software, alterando de pasada el estado del arte.

Para implementar sistemas informáticos que cumplan con el enunciado de arriba, debemos primero obtener una forma de realizar la ingeniería que, por el contrario, sea práctica y sencilla de aplicar; de otra forma estaría destinada al fracaso. Para obtener tal grado de exactitud y de flexibilidad, muchas veces tenemos que lidiar primero con complejidades extremas, las cuales resultan en productos que son de los más sencillos de utilizar.

Para lograr sistemas tan exactos para sus usuarios y tan flexibles como para poder transformarse sin la alteración del código fuente, y sobre todo que sean fáciles de aplicar, se ha tenido que realizar un trabajo muy intenso de investigación, en donde se ha lidiado con las complejidades extremas para obtener un producto final cuyas partes son sencillas de utilizar. Por lo tanto en este ensayo, antes de entrar en las complejidades, primero debemos entender y convencernos de ciertos hechos que nos dicen qué debemos hacer y para qué, para luego emplear libremente nuestra energía para inmiscuirnos en los detalles de la forma en que hacemos realidad estos hechos. Por ello, entendamos primero el problema para convencernos de la necesidad de su resolución.

2. Desarrollo

Al igual que cuando adquirimos cualquier producto, cuanto más sofisticado sea éste, mayores deberían ser las pruebas que le hagamos como futuros usuarios de tal artículo, lo cual en muchos casos no bastaría con la simple inspección y el uso de algunas de sus bondades mientras nos encontramos en la tienda. En tales casos de prueba mínima, en poco tiempo después de haber adquirido el producto sofisticado, nos llevaríamos una gran decepción al no satisfacer completa y correctamente nuestras necesidades.

Peor aún, si el producto sofisticado es nuevo en cuanto a sus características, desconocido será para nosotros su funcionamiento en esas nuevas facetas. Adquiriríamos el producto con la esperanza de que cómodamente encajen con nuestras necesidades. Lo más probable es que después de un tiempo lo abandonemos o regresemos a la misma tienda o a cualquier otra para adquirir uno nuevo que sea más ad-hoc a nosotros.

Refiriéndonos al software, se dice que mediante un programa de computadora se puede resolver cualquier problema, aunque la dificultad sea el "cómo". No encontramos fácilmente productos mundanos que puedan transformarse a como cambien nuestras necesidades; por ejemplo una lavadora que crezca cuando nuestra familia aumenta. Transformaciones de esa índole sí se pueden realizar con los productos de software, ya que, como se ha mencionado, mediante programas de computadora se puede resolver lo que sea. A pesar de que el "cómo" de todas maneras está presente, vale la pena el esfuerzo para lograr sistemas de software que permitan su transformación y su crecimiento. Piense en los dueños de las empresas: todos ellos quieren que sus negocios crezcan continuamente y mejoren, aunque para ello tengan que amoldarse a las nuevas necesidades.

Con tal propósito, aquí, en la Universidad Ricardo Palma, se está probando un sistema ERP (**E**nterprise **R**esource **P**lanning) pre construido, el cual es amoldable porque permite su total adaptación a cualquier empresa; es dinámico porque permite su establecimiento en tiempo de ejecución, lo cual es realizado principalmente por los usuarios finales; es extensible porque puede crecer y transformarse al ritmo que crece y se transforma la empresa u organización y, por último, es experimentable porque se logra un exacto establecimiento del sistema en cuanto a las necesidades, al permitirle al usuario final probar cada opción y cambiarla a como crea conveniente hasta que se llegue a su total aceptación.

Estas nuevas características imponen un nuevo estilo para realizar la ingeniería de la

información. Refirámonos a este estilo mediante el siguiente eslogan:

Dejar que el usuario final "diseñe todo".

donde ***"diseñe todo"*** se encuentra entre comillas porque al menos para el establecimiento del sistema tendrá como guía al ingeniero de software.

Entendamos entonces en qué consiste y a qué está orientado el sistema ERP que actualmente se está probando.

Todo comenzó con una gran incógnita muy natural que todo desarrollador de software se hace al principio:

¿Cómo hacer para superar a aquellas empresas que durante décadas han desarrollado software que resuelve problemas que nosotros también queremos resolver?

Es decir, cómo hacer mejores sistemas para los dueños de las empresas y sus usuarios finales, lo cual comprende, entre otros: contabilidad, procesos gerenciales, procesos industriales, inventarios, reportes, auditoría, comunicaciones, nómina, personal, . . .

La primera respuesta que se nos viene a la mente es eso que siempre queremos obtener como producto informático, es decir, ***simplemente desarrollar un sistema informático que calce exactamente con los requerimientos del cliente***. Esta respuesta es cierta, por supuesto, pero tiene un gran problema que no sale a la luz y se queda en la mente de los clientes quienes han adquirido el software. Para poder entender este problema, primero tenemos que ver, aunque a grandes rasgos, los pasos para desarrollar un sistema informático:

En su forma más simple, el proceso de desarrollo de software consiste en los siguientes pasos:

1) Análisis de requerimientos.
2) Especificación.
3) Diseño y arquitectura.
4) Programación.
5) Prueba.
6) Documentación.
7) Mantenimiento.

Por diversas razones, tales como refinamientos o errores, estos pasos pueden convertirse en una secuencia de acciones que han de realizarse en unos cuantos ciclos.

El centro del problema de este estilo es, sin lugar a dudas, que el sistema informático empieza a construirse ni bien el cliente acepta la especificación completa o parte de ésta, pero sin que el cliente haya experimentado plenamente con el software. Como usted se ha dado cuenta, aquí hay una contradicción: ¿Cómo es que el cliente va a probar algo que todavía no está hecho? Desgraciadamente la interacción importante con el cliente termina con el término de la segunda etapa: la especificación, momento en el cual el cliente firma el contrato para la construcción de su sistema de software.

Desde cierto punto de vista, la complejidad en el desarrollo de un sistema informático resulta en dos frentes por batallar:

1) La determinación de lo que realmente quieren y necesitan los clientes y los usuarios finales.
2) La construcción del software en sí, lo cual es una tarea grande y compleja.

El primer punto anterior puede llegar a ser el más difícil: muchas veces el usuario no se da cuenta de lo que realmente quiere en el momento de la firma del contrato; se da cuenta

después de meses de uso del sistema que le han proporcionado. Para ese momento, si el usuario quiere una rectificación, tendría que contratar otra vez a los constructores del software para que adecúen el sistema, lo cual podría ser nada fácil y podría también generar muchos gastos adicionales.

Habiendo identificado el problema de la determinación exacta de las necesidades de los usuarios finales, se puede citar la primera importancia de la presente investigación:

"Debido a la complejidad de los problemas que han de resolverse mediante un sistema informático, es necesario una forma más práctica para determinar el software del cliente, lo cual va más allá de la simple obtención de los requisitos en pro de la exactitud de las necesidades."

A manera de ilustración, podemos confrontar a la informática con cualquier otra especialidad. Pongámonos en el caso de la contabilidad. Entonces tenemos los siguientes axiomas:

1) Aunque generalmente un informático es especialista solamente en informática, puede hacer cualquier sistema, tal como por ejemplo un módulo de contabilidad.
2) Pero si un informático también es contador, entonces su módulo de contabilidad será superior.
3) Es poco probable que todos los módulos de contabilidad construidos por buenos informáticos sean superiores.

Aquí, por lo tanto, surge la siguiente pregunta:

¿Entonces, cómo un informático podría hacer un buen sistema, pero de los superiores, permitiendo así que el cliente quede por siempre totalmente conforme con su producto?

La respuesta es la clave que se dio al principio de este artículo:

Dejar que el usuario final "diseñe todo".

donde se vuelve a aclarar que **"*diseñe todo*"** se encuentra entre comillas porque al menos para el establecimiento del sistema tendrá como guía al ingeniero de software. El usuario final diseña todo mediante la interacción directa y la experimentación con el mismo sistema.

Ahora pensemos en una empresa de cualquier tamaño la cual requiere de un sistema informático para funcionar eficazmente. Lo cierto es que sus integrantes van a querer que su empresa crezca. Que la empresa crezca implica, entre otros, el crecimiento en sí, la modificación, el cambio y la creación de áreas de trabajo, de procesos, de roles, . . . Lo ideal es que el sistema informático vaya creciendo al ritmo de la empresa. Este crecimiento puede realizarse contratando a los que desarrollaron el software de la empresa o a otros profesionales, para que en el tiempo vayan transformando el sistema de tal manera que se pueda acomodar a las nuevas necesidades. Esta forma de hacer crecer el software tiene el gran inconveniente de que demanda mucho tiempo, esfuerzo y dinero.

Habiendo entendido el problema de la necesidad de la constante transformación de los sistemas informáticos debido a las constantes transformaciones de las empresas, se puede citar la segunda importancia de la presente investigación:

"Debido a la continua e inevitable transformación de las entidades a quienes se les va a construir sistemas informáticos, es necesario una forma más práctica de implantar el software del cliente, lo cual obliga a que el sistema permita su continua y fácil transformación para ajustarse a los nuevos requerimientos."

Es decir, aunque al inicio un sistema informático puede llegar a encajar con las necesidades de la empresa, debe brindar la posibilidad de ser transformado fácilmente.

En este punto, entonces, surge esta segunda pregunta:

Después de todo, ¿cómo un sistema informático puede tener tal grado de flexibilidad como para que pueda ser transformado fácilmente, de tal manera que siempre esté a la par con las nuevas necesidades de la empresa?

Otra vez la respuesta es la clave que se dio al principio de este artículo:

Dejar que el usuario final "diseñe todo".

Recuerde que **"*diseñe todo*"** se encuentra entre comillas porque al menos para el establecimiento del sistema tendrá como guía al ingeniero de software. El usuario final diseña todo mediante la interacción directa y la experimentación con el mismo sistema.

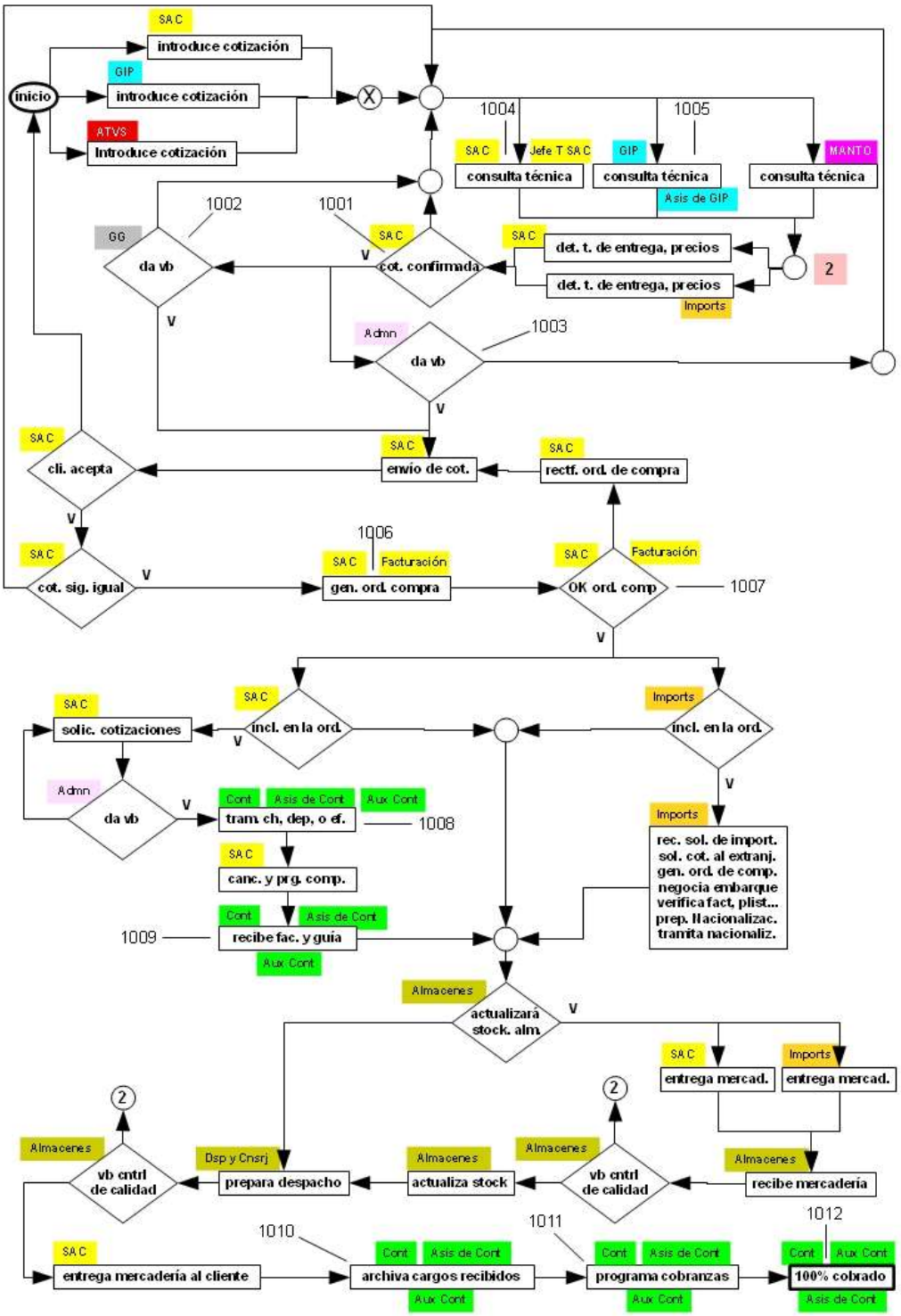
Del ERP, veamos a continuación sólo sus módulos de procesos, de contabilidad y de reportes. Considere que estos módulos, así como todos los demás, se construyeron y funcionan sobre una plataforma propietaria de quien desarrolló el ERP. Y así, todos los módulos conjuntamente con la plataforma, de manera dinámica permiten el establecimiento del sistema de la empresa; es decir, en tiempo de ejecución. Tal es el caso del establecimiento de la estructura de la empresa: empleados, áreas de trabajo, las ligaduras de los dos anteriores; módulos y sub-módulos activos para la empresa, privilegios; reportes; administración del sistema: respaldo, auditoría; contabilidad; inventario; procesos, y así sucesivamente.

Sobre los procesos

Para que el sistema cumpla con su estilo de dejar al usuario final "diseñar todo", es necesario que los procesos puedan ser implantados en tiempo de ejecución del sistema, es decir, en forma dinámica. Cada usuario final que tenga asignado el privilegio de crear procesos y generar sus instancias, podrá realizar estas acciones cuando lo crea conveniente, como parte del uso normal del sistema informático.

Un proceso es simplemente una secuencia de acciones las cuales pueden ser plasmadas en un diagrama de flujo. Este diagrama de flujo contiene puntos que muestran acciones que han de cumplirse en esos lugares de la secuencia, además de condicionales y bucles. En su forma más flexible, aquí, un diagrama de flujo puede tener más de un punto de inicio y más de un punto de salida. Un diagrama de flujo está relacionado con una entidad. Una entidad siempre tiene componentes. Cada punto en el diagrama de flujo puede estar ligado a cero o a más componentes de la entidad. Para una explicación sencilla en este artículo, consideremos como entidad a una empresa comercial. Siendo una empresa nuestra entidad de ejemplo, debemos considerar como sus componentes a sus diferentes áreas de trabajo: gerencia general, gerencia de personal, gerencia de contabilidad, gerencia de administración; mantenimiento, almacenes, auxiliar de contabilidad, servicio de atención al cliente, facturación, seguridad, sistemas, investigación y desarrollo y todas las demás que conocemos.

El siguiente es un diagrama de flujo de un proceso de nuestra empresa de ejemplo, el cual trata sobre cotizaciones para los clientes:



Debe fijarse que cada punto puede estar ligado a cero o más áreas de la empresa. Además, es importante notar que se puede realizar en forma paralela partes de un proceso.

Entonces, lo que a continuación debe hacer el usuario final es establecer el proceso en el sistema informático -por ejemplo, en el ERP en cuestión-. Una vez que se encuentre implantado, el mismo usuario u otros usuarios finales determinados podrán generar todas las instancias de este proceso que se requieran; en este caso de ejemplo, una instancia sería una cotización solicitada por un cliente.

Aquí hay dos preguntas que podemos hacernos: la primera es ¿cómo implantar el proceso, sobre todo si se hace dinámicamente?, la segunda es ¿después de implantar el proceso, ¿cómo

generar y ejecutar sus instancias? Estas dos preguntas no son fáciles de responder por lo complejo que es seguir una secuencia que pasa por todas las áreas de la empresa, que en cada punto del proceso puedan actuar cero o más áreas de la empresa, que en ciertas partes del proceso se pueda bifurcar en paralelismos y todo establecido en tiempo de ejecución.

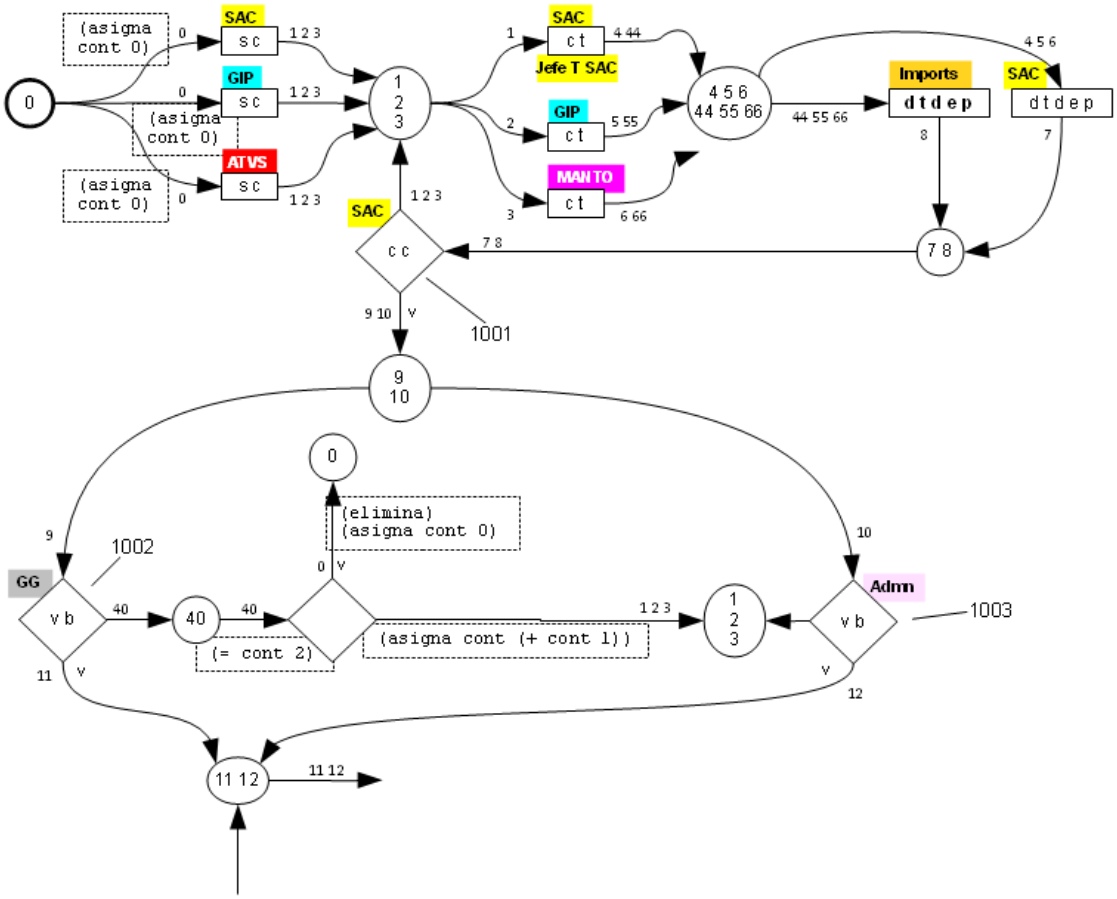
Para realizar estas dos acciones debemos seguir un procedimiento, el cual a grandes rasgos es el siguiente:

1) Una vez que contamos con el diagrama de flujo del proceso, lo convertimos en un grafo dirigido específico y, a través de éste, obtenemos un sistema de producción especial.
2) El sistema de producción especial, el cual debe estar contenido en un archivo de texto, lo introducimos de alguna manera en el sistema informático. Una vez realizado este paso, el proceso ya se encuentra implantado en el sistema.
3) A continuación ya se pueden generar las instancias que se requieran del proceso, las cuales serán ejecutadas por un algoritmo específico.

Hay que notar que aparte de ser todo realizado en forma dinámica, el sistema informático no genera código de ningún tipo, ni para la implantación de los procesos ni para la ejecución de las instancias de estos procesos.

Como parte de las acciones que han de realizarse en cada punto, pueden existir invocaciones a procedimientos que existen como librerías en el sistema informático. Como cualquier procedimiento de un programa, cada uno de estos puede tener una tarea tan simple como la evaluación de una expresión o el envío de un correo electrónico; o una tarea tan compleja como la realización de un gran sistema experto o la generación misma de una instancia de otro proceso.

Asimismo, como parte del proceso, en ciertos puntos puede haber expresiones cuyos resultados ayudan a guiar mejor el camino dentro del diagrama de flujo del proceso. Por ejemplo, en la siguiente figura la cual puede ser sólo un tramo del grafo dirigido, se puede observar que se usa un contador para establecer que el Gerente General (GG) puede desaprobar hasta dos veces una cotización (ver etiqueta "1002"); asimismo, en cada uno de estos sucesos el proceso va hacia el colector que contiene los números "1, 2, 3"; cuando una cotización es desaprobada por tercera vez, el proceso va al colector inicial, es decir, al que contiene el número cero, "0".



El procedimiento para la implantación dinámica de procesos y sus instancias, también puede ser aplicado a procesos totalmente industriales o híbridos, tal como para la fabricación automatizada. Un centro de fabricación automatizada consta principalmente de lo siguiente: (1) estaciones de trabajo programables para realizar una labor determinada, (2) una red de dispositivos PLC, en donde cada uno está ligado a una estación de trabajo y, por último, (3) un servidor para controlar y monitorear el sistema de producción a través de la red de los dispositivos PLC.

El procedimiento permite implantar dinámicamente procesos que dirigen toda la producción automatizada industrial, logrando un producto con cada instancia del proceso que se genere. El procedimiento también facilita en gran medida el establecimiento de toda la fábrica para la producción de artículos diferentes.

Sobre la contabilidad

Para cumplir con el estilo que dice que se debe dejar al usuario "diseñar todo", debemos darle al contador todas las facilidades para que verdaderamente diseñe todo su módulo de contabilidad en el sistema informático. Para apreciar lo que el ERP en cuestión le ofrece al contador para que establezca su contabilidad, veamos primero qué es un módulo de contabilidad de un sistema informático y qué debe contener.

Un módulo de contabilidad es aquel que registra todas las transacciones de contabilidad manteniendo consistente la ecuación contable y que, además, muestra reportes; todo basado en el catálogo de cuentas.

Los registros se basan en los libros contables, en donde los principales son los que sus nombres tienen como prefijo a "*libro diario de*" o simplemente a "*diario de*". Todos los demás libros se generan a partir de los datos que tienen estos libros de diario. Algunos libros de diario son obligatorios; otros existen por conveniencia de la contabilidad de la entidad.

La contabilidad se lleva por periodos, en donde cada uno por lo general dura un año calendario. Al final del periodo se ejecutan transacciones las cuales dejan el estado de toda la contabilidad como para poder iniciar el siguiente periodo contable, con los saldos de las cuentas llamadas "temporales" en cero. Entre periodos contables pueden variar las cuentas del catálogo de cuentas y algunas otras características como los libros que son obligatorios llevar en el siguiente ejercicio contable, dependiendo de las leyes contables del gobierno las cuales son susceptibles de cambio entre los periodos.

Entonces, otra vez, para cumplir con la propuesta que dice que hay que dejarle al usuario "diseñar todo", es necesario que el mismo contador, en forma dinámica, es decir, en tiempo de ejecución, pueda realizar lo siguiente: crear y actualizar el catálogo de cuentas; crear, editar y eliminar libros de diario, cerrar correctamente el ciclo contable y crear los reportes. Con ello podemos realizar la contabilidad y observar todo lo que hace ésta, además de mostrar cualquier reporte que sea necesario, tales como los estados financieros o cualquier otro que se necesite como por ejemplo para el análisis financiero. El ERP que se menciona en este artículo, el cual se está probando, permite realizar la contabilidad de esta manera.

Lo más crítico de esta forma de aplicar la contabilidad es satisfacer la necesidad del contador para generar cualquier reporte que necesite en cualquier momento; es decir, como se vuelve a repetir: ***cualquier reporte que necesite en cualquier momento***. Nunca le vamos a dar al contador una ventana para que ejecute SQL: nunca le vamos a mostrar, a ninguna persona, la lógica de la base de datos. Entonces cómo es que con el ERP en cuestión se logra que el contador genere sus reportes, los que quiera, en cualquier momento; la respuesta es mediante un lenguaje de cuarta generación, de los conocidos como 4GL o "4 **G**eneration **L**anguage". Estos lenguajes son de muy alto nivel, y sirven, entre otros usos, para que personas que no son programadoras puedan realizar una programación aplicada a un tema muy específico, en donde los contadores y la contabilidad están incluidos. Recuerde que para este caso prescindimos totalmente de los constructores de software.

Y así para tal propósito se ha inventado e implantado en el ERP un lenguaje de programación el cual es muy fácil y rápido de aprender, el mismo que usa el contador para establecer sus reportes y generar sus instancias en los momentos más oportunos. Por ejemplo, si uno de los reportes que el contador ha establecido es el estado financiero llamado "Balance", entonces puede generar una instancia de éste por la mañana y al anochecer generar otra instancia del mismo, con lo cual puede comparar y tomar decisiones conjuntamente con otros gerentes. Este lenguaje de programación se llama **Overlapping®©**, por lo que da la impresión de tratase de transparencias que se colocan por encima de los reportes requeridos, con el propósito de indicar lo que debe mostrarse en el reporte y cómo se debe mostrar. A continuación vea el siguiente programa en Overlapping, el cual es un pequeño ejemplo de un minúsculo reporte de "balance". Observe que invoca a otro estado financiero: el llamado "Estado de Variaciones del Capital Contable":

```
( {BALANCE}
  {ACTIVO}
  (101 + 21312 + 30111)
  {."Total del activo".}
  !
  {PASIVO}
  [(40114) "Total del pasivo"] +
  {"CAPITAL CONTABLE"}
  [(@ # "Estado de Variaciones
         del Capital Contable")
   CAPITAL]
  {."Total del pasivo y capital contable".}
)
```

Ahora vea una instancia de este reporte mediante una vista de todas las posibles que se pueden emplear para mostrar los datos resultantes. Tenga en cuenta que las vistas de un reporte pueden variar, más no así los datos que contienen.

BALANCE			
ACTIVO			
	(101)	Caja EFECTIVO Y EQUIVALENTES DE EFECTIVO	10100.00
	(21312)	Valor razonable De origen animal Productos agropecuarios y piscícolas terminados PRODUCTOS TERMINADOS	280.00
	(30111)	Valores emitidos o garantizados por el Estado Instrumentos financieros representativos de deuda Inversiones a ser mantenidas hasta el vencimiento INVERSIONES MOBILIARIAS	0.00
Total del activo 10380.00			
PASIVO			
	(40114)	IGV – Régimen de retenciones Impuesto general a las ventas Gobierno central TRIBUTOS, CONTRAPRESTACIONES Y APORTES AL SISTEMA DE PENSIONES Y DE SALUD POR PAGAR	280.00
Total del pasivo 280.00			
CAPITAL CONTABLE			
CAPITAL 10100.00			
Total del pasivo y capital contable 10380.00			

Hay otra facilidad más que el ERP les da a los contadores, la cual trata sobre transacciones que ellos mismos deben realizar para mantener la contabilidad; tal es el caso de los cierres de periodo, en donde, como se ha mencionado, se debe dejar en cero las cuentas temporales, como las de gastos, pero quedando registrado todo el encadenamiento que se realiza con las cuentas involucradas.

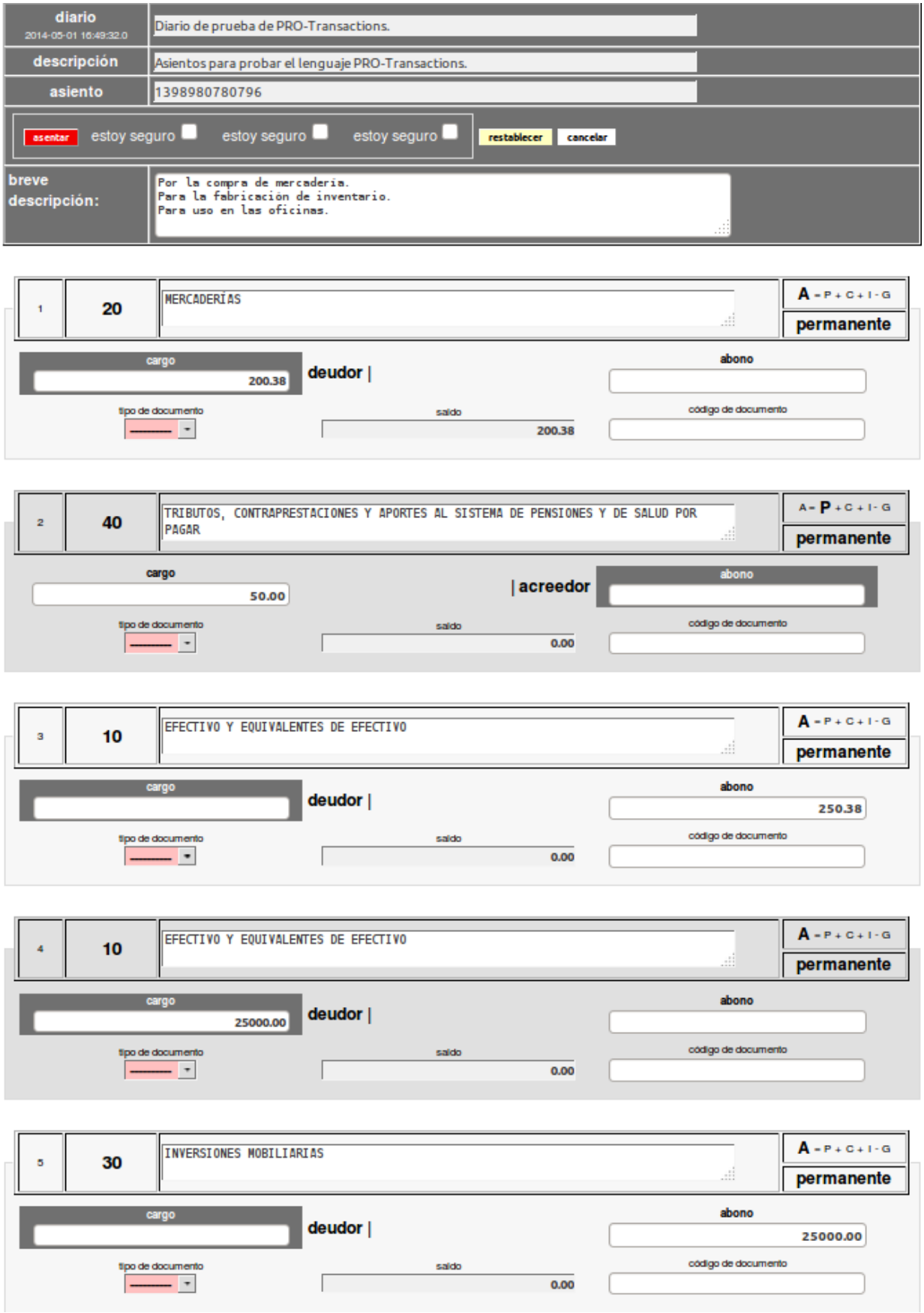
Las transacciones contables pueden realizarse directamente por intermedio de formularios o indirectamente como efecto de las ventas o pagos específicos que realiza la empresa, en cuyo caso serían transparentes para el usuario. Pero al contador mismo podríamos darle el estatus de súper usuario de la contabilidad de la empresa, por lo que él y la contabilidad de la empresa equivalen al administrador de la base de datos de la empresa y la misma base de datos. El administrador de la base de datos puede usar las interfaces gráficas (formularios) para interactuar directamente con la base de datos, sin interferir con todo lo demás que los usuarios

finales usan para realizar sus labores asignadas con el sistema informático. Las vistas (es decir, las interfaces gráficas) que usa el administrador de la base de datos sirven para interactuar plenamente con la base de datos; esto es, entre otras operaciones, para adquirir, insertar, eliminar y actualizar datos. Una de las vistas puede contener una ventana para que este administrador ejecute sentencias SQL, con lo cual tendrá pleno dominio sobre la base de datos para hacer con ésta todo lo que sea conveniente.

Por el lado del súper usuario de la contabilidad de la empresa, el ERP que nos referimos aquí, también presenta una ventana para que el contador incremente su dominio sobre la contabilidad. En este caso, el contador podrá ejecutar programas que escriba en el lenguaje de programación **Pro-Transactions®©**, el cual se ha inventado e implantado en el ERP. Este lenguaje de programación también es un lenguaje 4GL especial para contadores. Un programa en este lenguaje es un conjunto de especificaciones de transacciones de contabilidad basadas en un libro de diario, códigos de cuenta, montos de cuenta, documentos y códigos de documento, invocaciones a programas escritos en el lenguaje Overlapping y breves explicaciones de cada asiento, los cuales se pueden anidar. Note que un programa en Overlapping genera un valor numérico, por lo tanto una invocación a este lenguaje desde Pro-Transactions puede considerarse como una expresión.

Lo siguiente es un pequeño programa en Pro-Transactions; la figura que le sigue (©) es generada por el sistema, en donde el usuario contador puede dar la última chequeada a lo que va a suceder antes de presionar el botón de "asentar":

```
"Diario de prueba de PRO-Transactions."
{   "Por la compra de mercadería."
    {   "Para la fabricación de inventario."
        [(20 '200.38 40 '50) (10 '250.38)]
    }
    { "Para uso en las oficinas."
      [(10 '25000) (30 '25000)]
    }
}
```



Después de todo lo que se ha comentado aquí sobre la contabilidad, podemos deducir fácilmente que en el ERP en cuestión, el cual sigue el estilo mencionado, no tiene contabilidad mientras el mismo contador no la establezca, a conveniencia de la entidad donde labora. La contabilidad en el ERP es establecida por el mismo usuario final al cual nos referimos como "*el contador*".

Sobre los reportes

Aparte de los importantísimos reportes de contabilidad, los reportes en general son una característica infaltable de todo sistema informático que usan las diferentes entidades. Por ejemplo, los reportes de empleados, de inventario, de ventas, de clientes, de privilegios de empleados sobre los módulos del sistema, de auditoría, de administración del mismo sistema informático, de áreas ad-hoc, y así sucesivamente: reportes puede haber para todo. Es más, la cantidad de reportes puede hasta duplicarse por el hecho de que algunos pueden existir en dos versiones: uno para únicamente lectura y el otro para escritura, en donde éste último permite la transformación de los datos de la base de datos según las especificaciones del usuario final.

En este caso, para cumplir con el estilo que dice que se debe dejar al usuario "diseñar todo", debemos darle a este usuario la oportunidad de expresar lo que quiere leer y lo que quiere escribir (modificar en la base de datos). A diferencia de los procesos y de la contabilidad, los reportes los debe implementar el desarrollador del software, dado que él tiene acceso al nivel lógico de la base de datos y es el único quien lo debe tener.

Pero, qué hay del estilo que estamos persiguiendo: el desarrollador debe ser capaz de implementar y modificar en forma inmediata los reportes del usuario final, de tal manera que este usuario pueda determinar los reportes, experimentar con éstos y refinarlos poco a poco hasta obtener aquella versión final, infalible, la cual sin lugar a dudas es la que, sin saberlo, requería.

Siendo así, el usuario final es quien verdaderamente implementa y realiza el refinamiento de sus reportes; el desarrollador del software sólo hace el papel de asistente y, para obtener las siguientes versiones de los reportes corriendo en el sistema informático, procede en forma inmediata con lo que tiene que hacer de acuerdo a las indicaciones del usuario final.

Para tal efecto, el desarrollador debe contar con herramientas que permitan cumplir con los mandatos del usuario final, implantando y modificando reportes, en tiempo record; es decir, a más tardar para el día siguiente. Esta es una necesidad dado que mientras que no se haya terminado, el usuario final y sus reportes se encuentran en la etapa de refinamiento, la cual es realizada poco a poco. Visto de otra forma, el usuario final llega a la versión definitiva de sus reportes a través de la ejecución de un ciclo al cual le podemos llamar "refinamiento por pasos", el mismo que parte del diseño inicial y la implantación del reporte inicial en el sistema.

Habrán excepciones a la regla: algunos reportes no son tan simples como para estar implantados o modificados al día siguiente, tales como los complejos reportes que permiten buscar en el inventario; pero quizás estos reportes difíciles podrían estar listos para el segundo día o para uno muy cercano a éste.

En el sistema ERP en cuestión, el que se está probando en la URP, todos los reportes son obtenidos de la manera que se ha detallado, en donde se le permite al usuario final experimentar plenamente hasta que quede totalmente satisfecho. Es más, los reportes que los usuarios obtienen constan de características predefinidas para buscar en el resultado o para depurarlo, además de la opción para escoger las vistas que más se acomoden a sus propósitos específicos.

3. Conclusión

El ERP que se está probando en la URP se ha construido siguiendo el estilo que promueve dejar al usuario final diseñar todo. Siempre estará presente el desarrollador o el profesional de la informática quien es el que implanta el sistema. Este profesional podrá asesorar en todo momento a los usuarios finales, quienes en caso extremo pueden no haber utilizado alguna vez un software similar.

Se puede aseverar que es nuevo el estilo que aquí se ha tratado para realizar la ingeniería de software. Analizando los demás estilos (o enfoques) existentes, se nota que coinciden con las siguientes características las cuales no son características del estilo que aquí se expone:

1) En cuanto se establece la aceptación del cliente se procede con la construcción del software o parte de éste, orientando de alguna manera todos los esfuerzos para que el producto o módulo termine siendo lo que exactamente el cliente ha aceptado. Estos enfoques provocan la permanencia del gran riesgo de que a la larga los sistemas resultantes no coincidan verdaderamente con las necesidades del cliente o de los usuarios finales.
2) En todo el camino hacia la versión final del sistema, se requiere la reprogramación de los prototipos y rediseños.
3) El sistema informático resultante no tiene la posibilidad de crecer al ritmo del negocio, sin la participación de los constructores del software.

Entonces, ¿por qué deberíamos preferir un sistema informático como el ERP referido en este artículo, en vez de uno convencional? Esta pregunta se contesta mediante la siguiente relación de sus características y ventajas:

1) Los desarrolladores de software **ya no diseñan ni implantan módulos; ni tampoco procesos**.
2) **Aspectos específicos**, como por ejemplo la implantación de los reportes de contabilidad, **no forman parte del trabajo de los constructores de software**.
3) La aplicación se obtiene **en base al desarrollo**; no a la construcción. Entre otros, el desarrollo se basa en:
   a) **El establecimiento de la empresa.**
   b) **El diseño y la implantación de procesos.**
   c) **La programación con lenguajes 4GL.**
   d) **El refinamiento gradual de los reportes.**
   e) **La especificación del inventario.**
   f) **El establecimiento de la contabilidad.**

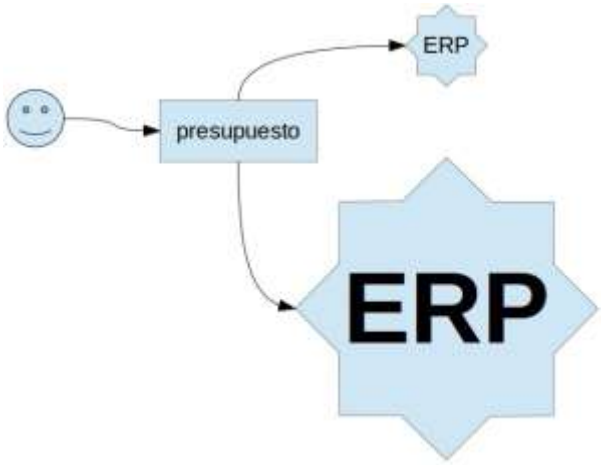
   Éstas acciones son realizadas por los usuarios finales de la empresa.

   No es necesario que el sistema esté completado al 100% para poder utilizarlo.
4) Los desarrolladores de software, **a lo más, se encargan de la realización de códigos específicos determinados por los diseñadores** de los procesos: al estilo **POJO**.
5) Herramientas muy simples para los usuarios finales que diseñan e implantan procesos: **lenguajes de programación 4GL** (**F**ourth **G**eneration **L**anguages), **procedimiento para la implantación dinámica de procesos y sus instancias**.
6) **Los nuevos módulos se obtienen mucho más rápido.**
7) **Flexibilidad extrema para la administración de los procesos: nuevos procesos, nuevas versiones, eliminaciones**.
8) **Flexibilidad extrema para la administración del resultado de la ejecución de los programas escritos con lenguajes 4GL: nuevos resultados, nuevas versiones, eliminaciones**.
9) **Posibilidad de experimentar** con los nuevos diseños.

10) **Con un mismo presupuesto**, los clientes tienen la posibilidad de tener un ERP que satisface sus necesidades básicas, **y también un ERP mucho más grande el cual va de acuerdo al resultado de sus propios planes de expansión y crecimiento**.

Todo depende del cliente y de los usuarios finales.

El último punto anterior hay que resaltarlo. Vea la siguiente figura, en donde se representa a un cliente el cual inicialmente tenía su ERP funcionando y satisfaciendo todas sus necesidades básicas. Con el tiempo, el mismo usuario pudo hacer crecer su ERP y está satisfecho por lo que el sistema creció y se amoldó a su negocio, sin alterar el presupuesto o emplear otro.



Antes de terminar debemos estar convencidos de lo siguiente:

1) Lo diferente puede ser mejor que lo que estamos tan acostumbrados.
2) El cambio del rol de nuestros clientes y de los usuarios finales puede ser dramático al inicio; después puede convertirse en un estándar imprescindible.
3) Nosotros también somos capaces de hacer cosas buenas, a nivel mundial.

Establezcamos entonces este estilo poniéndole como nombre a una palabra que puede en el futuro considerarse universal, derivado de una frase en inglés; esto es:

LUDA ("Let the User Design All")

Por sus características, el ERP sometido a prueba pertenece al área de Ciencias de la Computación, aplicado a la Ingeniería de Software.

El autor del presente sistema ERP es el suscrito, quien también es el responsable del proyecto que lo prueba. El equipo que prueba directamente este software está conformado por el profesor Ingeniero Luis Alarcón Loayza y sus colaboradores: Pamela Pimentel Huerta y Fabiana Fiori Orbegoso, ambas alumnas pertenecientes al tercio superior y se encuentran cursando los últimos semestres académicos de la carrera de Ingeniería Informática de la universidad Ricardo Palma.

Le agradezco a usted por el interés mostrado en este ensayo y lo invito a que lea los artículos posteriores, en donde se profundizará algunos temas tratados aquí.

4. Referencias

Patentes, derechos de autor y marcas de fábrica pertenecientes al autor:

1) El ERP sometido a prueba©: Partida registral Nº 0259-2018/DDA, INDECOPI
2) Patente en proceso, "Procedimiento para la implantación dinámica de procesos y sus

instancias"®:

3) INDECOPI, expediente 002581-2015/DIN del 2015-12-09
   Solicitud Internacional PCT/PE2014/000003 del 2014-03-17
4) El Lenguaje de programación Overlapping©: Partida registral Nº 00205-2014, INDECOPI
5) El lenguaje de programación Pro-Transactions©: Partida registral Nº 01503-2014, INDECOPI
6) Overlapping®: Resolución Nº 010582-2017/DSD, INDECOPI
7) Pro-Transactions®: Resolución Nº 010846-2017/DSD, INDECOPI