UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES

ARAGÓN

"'IMPLEMENTACIÓN DE PROCESOS DE CALIDAD DEL SOFTWARE' COMO ESTRATEGIA INDISPENSABLE DENTRO DE LA ORGANIZACIÓN"

INFORME DE DESEMPEÑO LABORAL

QUE PARA OBTENER EL TÍTULO DE:
INGENIERA EN COMPUTACIÓN

P R E S E N T A:

NANCY BELÉN RAMÍREZ MARTÍNEZ

ASESOR:
MAT. LUIS RAMÍREZ FLORES

MÉXICO, 2014.

Agradecimientos

*A mi **Madre** por el amor y apoyo incondicional que siempre me diste.*

*A mi **Hermano**, por ser mi cómplice, mi confidente y en ocasiones mi inspiración.*

*A mi **Abuelo**, por ser el pilar de mi mundo, un ejemplo de fortaleza y respaldo a pesar de cualquier circunstancia.*

*A mi tío **Manuel**, por estar siempre a mi lado, enseñándome, apoyándome, cuidándome, pero sobre todo amándome como mi segundo padre.*

*A **Miguel Ángel Solís Durán**, por creer en mí, por tu amor, paciencia y apoyo incondicional en todas mis locuras.*

*A **Mis Amigos**, aquellos que me estuvieron empujando, presionando, regañando, etc., para que continuara con este informe.*

*A toda mi **Familia**, por su paciencia y apoyo.*

*A **Alejandra Andrade** por la amistad incondicional que me ha demostrado con sus acciones.*

*A **mis compañeros, staff y couch del Básico 59**, por formar parte de mi vida, confiar en mi y apoyarme en la generación de este gran objetivo.*

*Finalmente a la mayor inspiración para hacer este informe, el **"Bronson Team"** el equipo con el que aprendí que en medio del pantano, puede crecer una flor de loto, pues demostró en excelencia la frase que dice "El talento gana partidos, pero el trabajo en equipo y la inteligencia ganan campeonatos (Michael Jordan)".*

Muchas Gracias

Nancy Belén Ramírez Martínez

Contenido

Índice de Ilustraciones

Índice de Tablas

Introducción

Las empresas son el corazón de la economía mexicana, desde las "microempresas", hasta las "grandes empresas", todas ellas generan algún porcentaje del empleo en México. Estas empresas se han visto afectadas por la crisis económica que nos aqueja desde hace mucho tiempo. Deben luchar para sobrevivir y crecer en el mercado actual, deben convertirse en empresas competitivas, de lo contrario se estarían condenando al fracaso.

Durante la lucha para convertirse en una empresa competitiva, los empresarios se concentran en reducir sus costos, aumentar sus ganancias, incursionar en nuevos mercados, etc. Es decir, están concentrados en los "resultados", en algunos casos son ejemplo vivo de la frase "El fin justifica los medios", lo cierto es que "El fin, es tan importante como los medios".

Las empresas pasan por diferentes etapas de maduración antes de llegar a ser una empresa competitiva. Normalmente la empresa exitosa no siempre lo fue, si no que tuvo que afrontar miles de problemas y resolverlos para crecer y mejorar. Al igual que durante la obtención de cualquier competencia, las empresas tuvieron que pasar por procesos de ensayo-error antes de obtener sus mejores resultados.

Pero ¿Cómo se logra que una empresa sea competitiva? una empresa competitiva debe poseer:

- Eficiencia: Obtener los mejores resultados con los menores costos.
- Calidad: Ofrecer productos y servicios con calidad.
- Innovación: Ofrecer productos innovadores.
- Sustentabilidad: Son empresas socialmente responsables.

Una empresa que desee ser competitiva, debe ser capaz de identificar los problemas que evitan que logre estas cuatro características. Para entonces tomar las decisiones que resuelvan dichas situaciones, en otras palabras, la empresa debe encontrar la causa raíz de los problemas para entonces resolverlos y ofrecer mejores servicios y productos.

Gran parte de las empresas grandes se han preocupado por obtener certificaciones de calidad, con el objetivo de tener procesos y directrices que les permitan identificar las problemáticas y así resolverlas eficientemente. Es por medio de procesos de aseguramiento de la calidad que las organizaciones resuelven este tipo de problemas.

Sin embargo, durante la implementación de un proceso de aseguramiento de calidad, es necesario hacer reestructuraciones muy minuciosas de sus áreas, implica una inversión importante de tiempo y recursos. Y aún cuando logran dichas certificaciones, estas deben

renovarse continuamente, pues nada permanece estático por mucho tiempo; lo cual les obliga a evolucionar de un "aseguramiento de la calidad" hacia una "mejora continua", es decir, es un proceso que nunca termina.

Desgraciadamente es muy común que, cuando las empresas pasan por momentos de crisis, los procesos de calidad, se ven desplazados por otras necesidades urgentes para la empresa. Pues todos los integrantes están bajo cargas de trabajo extremas, lo cual no les deja tiempo suficiente para revisar continuamente sus procesos de calidad. Lo cual es una decisión de alto riesgo, pues las cargas de trabajo extremas pueden ser producto de deficiencias en el proceso.

Si estas empresas no logran designar el tiempo y esfuerzo necesario y continúan desplazando la importancia de los procesos de calidad, los problemas irán en aumento, pues la evolución de la empresa no se detiene. Como consecuencia comienzan a enfrentarse a problemas cada vez mayores, de pronto se encuentran aplicando más acciones correctivas, que acciones preventivas, pero sin lograr resolverlos, porque no logran identificar la raíz del problema.

Además podrían estar perdiendo enormes oportunidades de negocio, pues como decía Albert Einstein "La crisis es la mejor bendición que puede sucederle a personas y países, porque la crisis trae progresos".

Es justo a este tipo de problemas a los que me he enfrentado durante mi experiencia profesional, primero como consultor de sistemas empresariales, ofreciendo software de carácter empresarial a empresas pequeñas y medianas, que aún estaban en un proceso de madurez y buscaban a través de un software mejorar sus procesos y su productividad.

Después como Tester, me encontré en una empresa muy importante del sector financiero en México, que buscaba garantizar la calidad de sus productos, pues una falla a nivel productivo podría traer la pérdida de la confianza de sus clientes además de pérdidas millonarias.

Este informe de ejercicio profesional, pretende plasmar los retos a los que me enfrenté, la forma en que fueron atacados dichos retos y los resultados obtenidos. Del mismo modo se analizarán los resultados y se describirá de acuerdo a mi experiencia actual, el modo en que deberían ser atacados para resolver dichos problemas.

En el primer capítulo se describirá la experiencia implementando sistemas empresariales orientados a Pymes; en el segundo capítulo se describirá la experiencia aplicando diferentes pruebas a software bancario; y en el tercer capítulo se describirán las estrategias aplicadas como Tester Leader Backup, en el área de calidad del software.

CAPÍTULO 1: El reto del Software enfocado a Sistemas Empresariales *Open Source*

De 2008 a 2009 estuve trabajando como consultor, en una empresa dedicada a la implementación de sistemas a la cual llamaré "ESOFTWARE". Debido a que la sucursal en México era muy pequeña (3 personas incluyendo a mi jefe), mis responsabilidades, intervenían en la mayoría de las etapas de los proyectos de implementación. Por ejemplo: Identificar necesidades de prospectos, diseñar "demostraciones" durante el proceso de venta, analizar los procesos de los clientes, interactuar con el área de desarrollo durante la construcción de las adecuaciones, implementar el software, elaborar manuales "a la medida" y capacitar al cliente.

Los clientes potenciales de ESOFTWARE eran Pequeñas y Medianas Empresas (Pymes). Los principales productos que ofrecía la empresa eran:

- OpenbravoERP 2.5: Sistema de gestión de recursos empresariales Open Source[1]. La empresa era partner de Openbravo, que es una empresa española, por lo que se ofrecía el ERP como el productos "estrella", ya que el ERP era el que generaba más ganancias para la empresa.
- OpenbravoPOS: Punto de venta de Openbravo, que cuenta con la opción de integrarse con el OpenbravoERP.
- SugarCRM 5.2: Sistema de gestión de la relación con los clientes Open Source.
- Karomi BPM 3.0: Sistema de Administración de Procesos (Se trata de un software Hindú).

Ilustración I: Logotipos de ERPOpenbravo, SugarCRM y Karomi.

[1] *Se considera Open Source al software que además de garantizar el acceso a su código, cumple con los 8 criterios citados en el sitio de Open Source Initiative. Mismos que garantizan la libertad de distribución y uso del código fuente (Open Source Initiative).*

1.1.Necesidad de las Pymes

Para ofrecer un servicio de calidad al cliente, es indispensable conocer sus necesidades e incluso anticiparse a ellas. Las Pymes al ser empresas en crecimiento, tienen necesidades especiales.

Las Pymes se han convertido en una parte fundamental para la economía de México, este tipo de empresas se encuentran en constante lucha contra competidores grandes y enfrentan problemas particulares de las Pymes, junto con algunas problemáticas de empresas grandes. Para analizar los procesos de las Pymes, es necesario entender su importancia en México y las problemáticas a las que se enfrentan, pues además de las problemáticas "generales", cada empresa tiene su propio giro, lo cual cambia enormemente la implementación.

Ya he dicho antes que las Pymes han adquirido un valor fundamental para el país en los últimos años. Tanto que la Secretaría de Economía ha puesto a disposición de los emprendedores y empresarios, el Fondo Pyme, con el objetivo de "promover el desarrollo económico nacional", por medio de la creación y apoyo al crecimiento de las Pymes (Secretaría de Economía, 2013).
En la tabla "1.1 Estratificación de las micro, pequeñas y medianas empresas" se muestra las características que definen a una Pyme.

Tabla 1: Estratificación de las micro, pequeñas y medianas empresas (Secretaría de Economía, 2013).

Tamaño	**Sector**	**Rango de número de trabajadores**	**Rango de monto de ventas anuales (mdp)**	**Tope máximo combinado**
Micro	Todas	Hasta 10	Hasta $4	4.6
Pequeña	Comercio	Desde 11 hasta 30	Desde $4.01 hasta $100	93
	Industria y Servicios	Desde 11 hasta 50	Desde $4.01 hasta $100	95
Mediana	Comercio	Desde 31 hasta 100	Desde $100.01 hasta $150	235
	Servicios	Desde 51 hasta 100		
	Industria	Desde 51 hasta 250	Desde $100.01 hasta $150	250

El por qué de ese interés en las Pymes se respalda en lo que estas representan para la economía en México. La Secretaría de Economía (SE) las considera "columna vertebral de la economía nacional" (Secretaría de Economía). Debido a que generan el 52% del producto interno bruto (PIB) y el 72% del empleo en el país. Entre las ventajas que describe la SE acerca de las Pymes están.

Ventajas:

- Facilidad para cambiar los procesos técnicos necesarios.
- Tienen posibilidad de crecimiento y de llegar a convertirse en una empresa grande.
- Asimilan y adaptan nuevas tecnologías con relativa facilidad.
- Se establecen en diversas regiones del país.

Desventajas:

- "No se reinvierten las utilidades para mejorar el equipo y las técnicas de producción".
- "Es difícil contratar personal especializado y capacitado por no poder pagar salarios competitivos".
- "La calidad de la producción cuenta con algunas deficiencias porque los controles de calidad son mínimos o no existen".
- "Algunos otros problemas derivados de la falta de organización como: ventas insuficientes, debilidad competitiva, mal servicio, mala atención al público, precios altos o calidad mala, activos fijos excesivos, mala ubicación, descontrol de inventarios, problemas de impuestos y falta de financiamiento adecuado y oportuno".

Con lo anterior se puede entonces decir que las Pymes generalmente: cuentan con poco personal, tienen poco capital para invertir, no pueden pagar a expertos por lo que no logran resolver los problemas a los que se enfrentan, sumados a diversos problemas derivados de la falta de organización. No es raro entonces que los dueños de dichas empresas, estén demasiado ocupados atendiendo problemas a nivel "operación", en lugar de concentrarse en la toma de decisiones, perdiendo así, oportunidades de mejora y crecimiento.

Esas necesidades son precisamente para las que fueron diseñados los diversos software que ofrece ESOFTWARE, están diseñados para "Unificar la información y brindar herramientas de toma de decisiones". Si a eso sumamos la facilidad de las Pymes de cambiar los procesos técnicos necesarios y que ahora cuentan con el "Fondo Pyme" para financiar sus proyectos, entonces se puede entender, el por qué ESOFTWARE esta tan interesado en este mercado. De hecho no es la única empresa que ha orientado sus esfuerzos hacia las Pyme, se competía contra empresas con muchísima experiencia en empresas consideradas "empresa grande" en México, como SAP, Microsoft Dynamics, Intelisis, Oracle, entre otros. Mismas que bajaron sus precios a un nivel histórico para atacar este mercado, cuando anteriormente eran inaccesibles para las Pymes por sus altos costos, tanto de implementación como de licencias.

El ERP y el CRM se les ofrecía a las Pymes bajo la promesa de que al ser Open Source[2] los costos de implementación y mantenimiento eran mucho menores. Lo cual era parcialmente cierto. Pues aunque el costo de adquisición estimado era mucho menor y el costo de licencias era ausente, comparado con el software de la competencia; estos debían adecuarse a los procesos del cliente y en los casos más complicados, la empresa debía crear los procesos desde cero a la par de la implementación. El costo monetario podía ser menor, siempre y cuando todo saliera "de acuerdo a lo planeado".

Por lo anterior durante el diseño de la solución, debían adecuarse tanto el software como los procesos de la empresa. Lo cual implicaba un esfuerzo muy grande tanto para la Pyme como para ESOFTWARE.

Para cumplir esa "promesa" al cliente, era de vital importancia ofrecer un "Servicio de Consultoría a la Medida", es decir, demostrar al cliente que la solución que se le estaba ofreciendo, le resolvería "todos los problemas". Era necesario hacer una investigación previa de las necesidades de los clientes y mostrarles la forma en que la herramienta les ayudaría a cumplir sus objetivos. Justo ahí es donde comienza mi participación, realizar las demostraciones para asegurar la venta.

1.2. "Demostraciones para captar clientes"

Mi trabajo en esta etapa era apoyar al vendedor, durante el proceso de captación de clientes por medio de "demostraciones" del producto. Es decir mostrar el funcionamiento del software con un ejemplo del futuro cliente.
Durante el proceso de venta el vendedor recababa la información, identificando las necesidades del cliente (Aquellas que el cliente solicitaba explícitamente), así como el giro de la empresa. Esta información se me proporcionaba una semana antes de la siguiente sesión con el futuro cliente.

Con esta información, se debían elaborar demostraciones de no más de 15 minutos, normalmente consistían en mostrar los módulos principales del ERP, POS ó CRM, mas uno o dos ejemplos del caso particular del cliente.

Mi participación se centraba en demostraciones para OpenbravoPOS, OpenbravoERP y SugarCRM.

[2] *Actualmente OpenbravoERP y SugarCRM no cumplen completamente con todos los puntos de la definición de Open Source (Open Source Initiative), ya que manejan licencias con diferente alcance para sus diferentes versiones. Por ejemplo, una versión abierta y una versión comercial.*

Para **OpenbravoPOS** debía mostrarse:

- Ejemplo de una o más ventas.
- Imprimir nota de consumo.
- Reportes de inventarios.
- Inventarios afectados por compras y ventas.
- Registrar nuevos productos y su cantidad en inventario.
- (Si fuera el caso de un restaurante) Mostrar la funcionalidad con mesas.

Para **SugarCRM** debía incluir ejemplos con "descripciones" de acuerdo al giro del cliente (Es decir que los productos o servicios registrados en el CRM tuvieran los mismos nombres que los del cliente):

- Creación de un Prospecto.
- Creación de una oportunidad.
- Calendario – Crear una cita.
- Módulo "Actividades" (Mostrar actividades previamente cargadas)
- Módulo "Cuadro de Mando" que muestra gráficas de las ventas obtenidas y los prospectos reclutados por mes.

Para OpenbravoERP debía incluir ejemplos con "casos particulares del cliente" (Es decir nombre de productos, materiales, listas de producción con sus componentes, etc.):

- Mostrar los datos de clientes y proveedores.
- Una compra y una venta con su respectiva factura. Incluyendo equivalencias en diferentes tipos de moneda. Por ejemplo hacer una compra en dólares y una venta en pesos. Y que ambos movimientos se reflejen en la cuenta general.
- Validar la afectación en el módulo de Almacén.
- Mostrar en almacén datos como:
    - Fecha de caducidad.
    - Datos de compra a granel y su conversión a unidades de venta.

Con esto se lograban dos objetivos, por un lado, mostrarle al futuro cliente que la herramienta que le ofrecíamos era funcional y por otro lado, se identificaban aquellas necesidades que requerirían una modificación en el software, lo cual incrementaba el precio y el tiempo de la implementación.

Es importante señalar que al ofrecer software "Open Source" más de un cliente reclamaba un precio todavía menor, pues relacionaban "Open Source" con "Gratis". Así que parte del trabajo del vendedor era hacerle ver al cliente, que si bien podía descargar el software e instalarlo el mismo, no contaría con ninguna asesoría, ni tenía la experiencia para implementarlo, es decir, que no le vendíamos el software, sino la implementación del mismo.

1.3. La implementación de OpenbravoERP

Una vez que la venta quedaba cerrada, comenzaba entonces la implementación, la primera actividad en todos los proyectos era la junta de arranque o "Kick off". En esta reunión se veían temas como:

- Alcance del proyecto y proceso de control de cambios.
- Presentación del equipo por parte de ESOFTWARE.
- Responsabilidades de cada integrante del equipo.
- Presentación del equipo asignado por parte del cliente.
- Plan de trabajo.
- Fechas de entregables.

La etapa de Análisis.

La primera vez que participé en el análisis para implementación del OpenbravoERP, solamente apoyaba a la gerente de consultoría que venía de Guadalajara, quién utilizaba los cuestionarios y formatos previamente diseñados para dicho levantamiento, tenía un amplio conocimiento del sistema y varios años de experiencia en procesos "contables" generales para toda empresa.
Los puntos que debíamos cubrir, correspondían a los datos maestros y a los diferentes módulos del ERP.

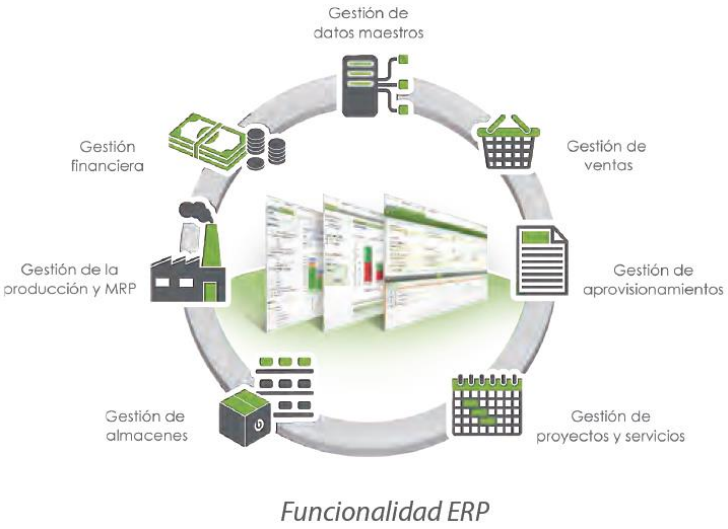


Ilustración II. Funcionalidad del ERP (Openbravo, 2014).

De acuerdo a los formatos establecidos por la empresa eran los siguientes:

- Gestión de Datos Maestros
    - Terceros
        - Información General
        - Por tipo de Terceros
            - Cliente

- Proveedor
- Empleado/Comercial

▪ Información de Detalle

- Cuenta Bancaria
- Direcciones
- Personas de Contacto
- Plantillas
- Descuentos
- Descuento por volumen de Compra

o Productos

▪ Información General

▪ Información de Detalle

- Lista de Materiales
- Sustituto
- Compras
- Coste (Pestaña de información que se llena con automáticamente)
- Precio
- Unidad de pedido
- Contabilidad
- Operaciones (Pestaña de solo lectura)

- Gestión de Compras
  - o Transacciones
    - ▪ Pedido de Compra
      - Modificación de precios (pestaña de solo lectura
      - Impuestos
      - Efectos
    - ▪ Albarán (Comprobante de entrega).
    - ▪ Factura
      - Modificación de precios
      - Dimensiones de Contabilidad
      - Impuestos
      - Efectos
    - ▪ Pedidos de Compra Cuadrados(Pestaña Solo lectura)
    - ▪ Facturas Cuadradas (Pestaña Solo lectura)
  - o Herramientas de Análisis
    - ▪ Análisis dimensional de pedidos

        - Análisis dimensional de albaranes
        - Análisis dimensional de facturas
        - Facturas
- Gestión de almacén
    - Transacciones
        - Inventario Físico
        - Movimientos entre almacenes
        - Nota de entrada
        - Operaciones de material (uso directo) Pestaña de solo lectura
    - Herramientas de análisis
        - Caducidades
        - Informe ofertas
        - Informe transportistas
        - Informe de trazabilidad
        - Movimientos almacén
        - Números de referencia
        - Stock
        - Valoración de Stock
        - Detalle movimientos artículo
            - Inventario
            - Detalle Inventario
            - Facturación consigna
- Gestión de Ventas
    - Transacciones
        - Pedido de Venta Simple
            - Impuestos
        - Pedido de Venta
            - Modificación de precios(pestaña de lectura)
            - Impuestos
            - Efectos
        - Nota
        - Factura
            - Modificación de precios(Pestaña solo lectura)
            - Impuestos
            - Efectos
        - Procesar comisión
            - Cuantía de la comisión

        - Herramientas de análisis
            - Análisis dimensional de pedido
            - Análisis dimensional de nota de entrada
            - Descuentos
            - Informe de devolución para pedido de venta
            - Pedidos
            - Pedidos facturados
            - Pedidos no facturados
            - Pedidos suministrados
            - Notas de Entrada
                - Detalle Facturas
                - Facturas
                - Análisis dimensional facturas
- Gestión Financiera
    - Gestión Cobros y Pagos
        - Transacciones
            - Extracto bancario
        - Diario de caja
        - Gestión estado de efectos
        - Liquidación manual
            - Efectos Contables
                - Conceptos
                - Reemplazar Conceptos
        - Liquidación
            - Efectos cancelados (Pestaña de solo lectura)
            - Efectos Generados (Pestaña de solo lectura)
    - Herramientas de análisis
        - Banco
        - Caja
        - Previsión de Tesorería
        - Efectos
        - Impuestos
        - Seguimiento de cobros y pagos
- Contabilidad
    - Transacciones
        - Asientos Manuales
        - Asiento

        - Apuntes
      - Presupuesto
    - Herramientas de análisis
      - Documentos no contabilizados
      - Cuadros plan general contable
      - Balance sumas y saldos
      - Libro mayor
      - Diario Asientos
      - Generación modelo
      - Informe de cash Flow
- Activos
  - Activos
    - Plan de amortización
    - Contabilidad
  - Categoría Activos
    - Contabilidad
  - Amortización
- MRP
  - Planificación de la producción
  - Planificación de compras
  - Necesidad de Material
  - Previsión de ventas
    - Línea de previsión de ventas
  - Información.

Cada uno de los puntos anteriormente listados hacen referencia a una funcionalidad y configuración del OpenbravoERP, ahí debían quedar registrados los campos actuales que permanecerían o cambiarían de nombre, además de los campos adicionales que solicitara el cliente. Era como un "Checklist de datos maestros". Por ejemplo un pedido de venta simple contendría los siguientes campos:

Tabla 2. Ejemplo de registro de levantamiento de información para un pedido de venta simple. (Elaboración propia, 2014).

Campos	Estándar	En uso	Tipo Dato	Observaciones
Nº documento	X	X	Numérico	
Fecha de pedido	X	X	Check	
Tercero	X	X	Pop-up	
Tarifa	X	X	Lista	
Moneda	X	X	Lista-lectura	
Almacén	X	X	Pop-up	

Campos	Estándar	En uso	Tipo Dato	Observaciones
Forma de pago	X	X	Lista	
Imp. total líneas	X	X	Numérico	
Importe total	X	X	Numérico	

Probablemente durante el levantamiento, el cliente solicitaría que en lugar de "Tercero" diga "Cliente"; que se quite el campo de Tarifa, por que maneja la misma tarifa para todos sus clientes; y que se agregue un campo que contenga el nombre del empleado que hizo la venta. Entonces en el documento de levantamiento quedaría así.

Tabla 3. Ejemplo de registro de levantamiento de información para un pedido de venta simple, quitando el campo "Tarifa" y agregando el campo "Vendedor". (Elaboración propia, 2014).

Campos	Estándar	En uso	Tipo Dato	Observaciones
Nº documento	X	X	Numérico	
Fecha de pedido	X	X	Check	
Tercero	X	X	Pop-up	Que cambie el nombre del campo a "Cliente"
Vendedor		X		Agregar
Tarifa	X		Lista	Quitar
Moneda	X	X	Lista-lectura	
Almacén	X	X	Pop-up	
Forma de pago	X	X	Lista	
Imp. total líneas	X	X	Numérico	
Importe total	X	X	Numérico	

Esto aplicaba para cada una de los formularios y pantallas del OpenbravoERP

Diseño y Construcción

Durante estas dos fases de implementación mi única responsabilidad era enviar los documentos de levantamiento "aprobados por el cliente" al equipo de desarrollo. Al mismo tiempo si ellos tenían alguna duda sobre la información que les enviaba, me solicitaban el apoyo. Yo debía aclararles la duda, incluso debía volver a llamar al cliente de ser necesario hasta que no les quedara ninguna duda sobre las necesidades y el alcance.
Con esa información ellos diseñaban y aplicaban las modificaciones al sistema, para después comenzar con la instalación.

Instalación

Solamente participé en una instalación del OpenbravoERP, para una empresa en Toluca que se dedicaba a distribuir materias primas para panaderías, es decir compraba las materias primas a

granel y luego las vendía al menudeo, esta empresa tenía varias sucursales, unas en el Distrito Federal y la matriz estaba en Toluca.

Esta empresa se negó a que la instalación se hiciera en los servidores de ESOFTWARE (debido al costo de hosting) y proporcionó un servidor con características mínimas para el funcionamiento del ERP. Es importante mencionar que el servidor que nos proporcionaron era una de las PC existentes en Toluca, era muy antigua y la conexión a internet con la que trabajaban era extremadamente lenta.

Por este motivo nos encontramos muchos problemas para instalar el OpenbravoERP. Ya no sólo debíamos instalar el ERP si no también poner en marcha un servidor en una PC, con recursos muy limitados.

Normalmente, la instalación era realizada por el equipo de desarrollo de Guadalajara, sin embargo ningún desarrollador estaba disponible en ese momento para trasladarse a Toluca a hacer dicha instalación. Así que me designaron a mí como responsable para dicha tarea. Como era la primera vez que instalaba el ERP, me proporcionaron la guía de instalación publicada por el sitio de openbravo, misma que seguí paso a paso.

A continuación describo los pasos que seguí groso modo:

1. Instalé CentOS 2.1[3] con ayuda de un CD que había preparado previamente, la instalación se efectuó con éxito. El problema fue que no existían *drivers* adecuados, debido a la antigüedad del equipo. Por este motivo tuve que buscar los drivers en línea desde el mismo servidor, en un proceso del tipo "ensayo y error", lo cual consumió más de una semana, debido a la velocidad de la conexión con la que contaba en las instalaciones del cliente.

2. Instalé y configuré las variables de entorno de los siguientes *software*, por medio de comandos en la consola de Linux:
   - Java 1.6.0
   - Apache Tomcat 6.0
   - Apache-Ant 1.7.1
   - PostgreSQL 8.2

3. Creé la base de datos para openbravo en PostgreSQL.

[3] *Sistema operativo Linux, creado a partir del código fuente de "Red Hat Enterprise Linux". La versión más actual hoy en día es la 6.5 (The CentOS Project, 2014).*

4. Importé la base de datos que había preparado previamente el equipo de desarrollo. Es decir, la base de datos que contenía parte de las modificaciones al ERP, además de los datos maestros del cliente (productos, proveedores, clientes, almacenes, cuentas contables, etc.).

5. Copié el archivo "openbravo.war" en la carpeta webapps de Tomcat y reinicié el servidor de Tomcat. Terminé el proceso exitosamente, sin recibir ningún error por parte de Tomcat, sin embargo, al momento de ingresar a openbravoERP, el sitio web simplemente no cargaba.

Los últimos cuatro pasos los terminé en un solo día, sin embargo, debido a que por alguna razón no funcionaba, me llevé dos días más intentando encontrar el problema en medio de otro proceso de ensayo-error. Por un lado parecía que el tomcat no lograba levantar el OpenbravoERP, pero no enviaba ningún error que indicara la forma de resolverlo. Durante los siguientes días revisé las variables de entorno, aumenté la memoria designada a "Apache-Tomcat 6.0", reinstalé el "Apache-Ant 1.7.1", recompilé el archivo openbravo.war. En resumen revisé y moví todo lo que se me ocurrió, apoyándome en los foros de la comunidad de openbravoERP, pero nada funcionó.

Ante esta situación y las quejas del cliente, ESOFTWARE envió a uno de los desarrolladores de Guadalajara (de hecho al más experto), para que solucionara el problema. Lo que sucedió entonces, es que tardó dos días más en hacer que el ERP cargara exitosamente. Pero al comenzar las pruebas funcionales básicas, el ERP volvió a fallar.

Después de un análisis minucioso, el experto llego a la conclusión de que el servidor que se había designado para instalar el ERP, no era el adecuado para dicha implementación, las características de la PC eran suficiente para tener instalado el ERP, pero se necesitaba más memoria para soportar la cantidad de datos que la empresa manejaba.

En medio de intentar resolver el problema, se investigaron otras soluciones:

- Aumentar la memoria de la PC: Debido al modelo antiguo, ya no podía expandirse más la memoria.
- Utilizar otra PC con mejores características: Encontramos que la mayoría de los empleados no tenían equipos, y además la red de la empresa tampoco ofrecía la infraestructura para que todos los empleados tuvieran acceso al ERP.

La única solución aparente era que el cliente comprara un servidor. Cuando se le informó al cliente la problemática, este se negó a adquirir un nuevo servidor, además de que no estaba dispuesto a pagar por "hosting", ni en los servidores de ESOFTWARE, ni en ningún otro. Se

acusó a la empresa de "incumplimiento del contrato", negaba rotundamente que se le hubiera advertido de este problema desde el principio. El tema estuvo a punto de llegar a problemas legales, tuvo que intervenir uno de los socios de ESOFTWARE y convenció al cliente de que sólo se cancelara el proyecto, se pudo entonces evitar un problema legal, pero la inversión en recursos y tiempo represento una pérdida económica muy grande.

Conclusiones

Los principales errores cometidos fueron:

- Las entrevistas y levantamiento de información se hicieron solamente en una de las sucursales del cliente, y solamente con los usuarios que el cliente nos asignó, por lo que no se identificaron los problemas de acceso a la red, ni la falta de infraestructura en las sucursales de Toluca.
- Se le informó al cliente desde el proceso de venta los requisitos mínimos para que funcionara el ERP, más el espacio necesario por la base de datos. Pero en el contrato "sólo quedaron plasmados los requisitos mínimos", esto fue una omisión por parte del equipo de ventas.
- No se solicitó con anticipación que se nos diera acceso al servidor "asignado" para la instalación del ERP. Por lo que nos dimos cuenta de la problemática cuando el proyecto ya estaba muy avanzado.
- En el tiempo de implementación no se contempló el tiempo de instalación del sistema operativo en una PC de esa generación, pues se **asumió** que el servidor contaría con los requisitos mínimos establecidos.

De acuerdo a mi experiencia actual puedo concluir que:

- El área de ventas debe estar en constante comunicación con el área de consultoría. Si se hubiera consultado a los "expertos", estos habrían identificado que los requisitos mínimos descritos en el contrato estaban incorrectos.
- En el proceso de análisis debió entrevistarse a más de un empleado. Las entrevistas se realizaron solamente con las personas que el cliente asigno, sin embargo, para hacer un buen análisis el consultor debe interactuar con la mayor parte de las personas involucradas en dichas áreas. Sólo de ese modo lograrán identificarse las problemáticas reales a las que podíamos enfrentarnos durante la instalación. Pues aunque se tuvo éxito entendiendo el proceso de la empresa y los datos maestros necesarios que deben cargarse en el ERP. No se identificaron los problemas con los que se enfrentaban los empleados todos los días. Como el acceso a la red o a una PC, por ejemplo.
- Debe solicitarse acceso a los equipos donde se instalarán los servidores con anticipación. Lo cual nos habría permitido, encontrar soluciones junto con el cliente. E incluso, si el cliente tomaba la decisión de cancelar, las pérdidas económicas, habrían sido mucho menores.

Comentarios adicionales

Este cliente esperaba que al implementar dicho software todos sus problemas se vieran solucionados "mágicamente", sin embargo, eso es una falacia. Un software ERP puede solucionar problemas de duplicidad de información, brindar herramientas que permiten tener el panorama completo de la empresa y así tomar decisiones. Pero ¿Cómo tener el panorama completo? Si las personas que alimentan con información al ERP no están comprometidas, o peor aún no tienen las herramientas necesarias para hacerlo, como una PC con conexión a internet.

Es decir, resolver los problemas de una empresa, no depende de un "Software ERP", si no de la capacidad de los líderes de la organización de resolver sus problemas internos; enfocar los esfuerzos en objetivos organizacionales claros; definir estrategias de comunicación; asegurar que todos los integrantes de dicha empresa tengan los recursos, herramientas e información necesaria, para que realicen su trabajo de manera eficaz.

Este cliente estaba muy enfocado en adquirir un ERP, pero no estaba enfocado en el crecimiento de su organización. Estaba enfocado en ahorrar recursos económicos, pero no ponía atención en sus recursos intelectuales. Una empresa que pone poca atención en su equipo de trabajo, podrá sobrevivir, pero no evolucionar. Podrá obtener ganancias, pero no logrará ser una empresa líder en el mercado, aunque instale el ERP más robusto y caro del mundo.

1.4. Capacitación SugarCRM

Mi participación en las implementaciones del CRM era escasa, existía otra compañera encargada de llevar todo el proceso de implementación del CRM con los clientes. Como los productos principales que tenía asignados eran el ERP y el BPM, yo sólo apoyaba en la parte de "capacitación al cliente".

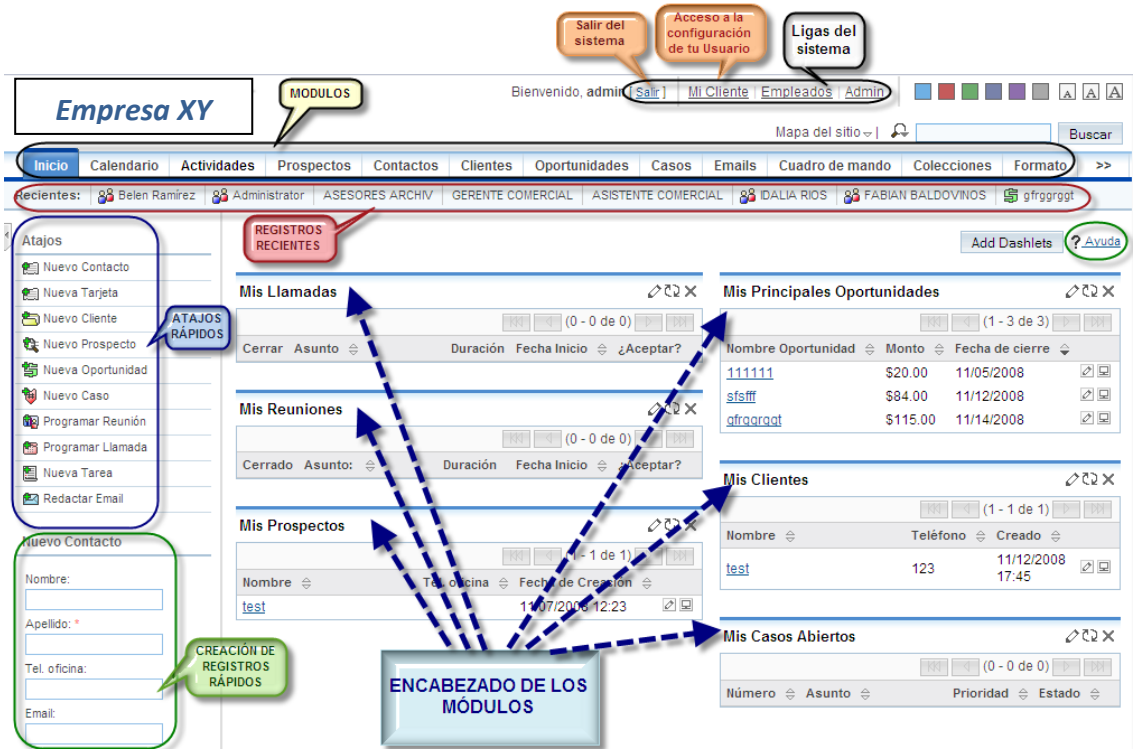


Ilustración III. Ejemplo de pantalla principal de CRMSugar, utilizados en la capacitación. (Elaboración propia ,2014).

La capacitación que se brindaba era siempre a dos niveles, a nivel usuario y a nivel administrador. Esta se impartía en las instalaciones del cliente.

Capacitación a nivel usuario

Consistía en capacitar al personal con rol de "vendedor o agente comercial" las funciones del CRM. Esta capacitación contenía los siguientes temas:

- Requisitos técnicos
- Acceso al CRM
- Preferencias de su perfil
- Navegación en la del interfaz de sistema CRM
    - Módulos del sistema
        - Opciones comunes de los módulos

        - Ver y manejar la información de los registros
        - Manejo de los Encabezados de módulos (Dashlets)
    - Buscar información en el sistema
        - Búsqueda global
        - Búsqueda básica
        - Búsqueda avanzada
        - Para guardar y manejar resultados de la búsqueda
- Uso del sistema CRM
    - Módulo de Inicio
        - Para agregar un dashlet:
        - Para invertir los ajustes de la página de Inicio
    - Módulo Mi Portal
        - Para agregar un sitio al portal:
        - Para administrar los sitios del portal:
    - Módulo Calendario
        - Crear citas
    - Módulo de Actividades
        - Programar llamadas y reuniones
        - Para agregar asistentes
        - Para manejar actividades
        - Crear tareas
        - Crear notas y archivos adjuntos
    - Módulo de Contactos
        - Para ver y manejar contactos
        - Crear Tarjetas de sitas
        - Para ver y manejar cuentas
    - Módulo de Prospectos
        - Crear prospectos
        - Administrar información de prospectos
    - Módulo de Oportunidades
        - Para crear una oportunidad
        - Para manejar oportunidades
    - Módulo de Casos
        - Para crear un caso
        - Para manejar casos
    - Módulo de Gestor de Incidencias
        - Para crear un reporte de un error
        - Para administrar reportes de errores

- Módulo de Documentos
  - Para crear un documento
  - Para manejar documentos
  - Para Actualizar un Documento
- Módulo de Correo Electrónico
  - Crear correos electrónicos
  - Para administrar entrada de correos electrónicos
  - Para administrar salida de correos electrónicos
  - Archivando Correos Electrónicos
  - Creando Plantillas de Correos Electrónicos
  - Para administrar plantillas de correo electrónico
- Módulo Campañas
  - Creando e Importando Objetos de Campaña
  - Para crear una lista objeto
  - Creando una Campaña
  - Tipos de campañas
  - Ejecutando una Campaña
  - Creando registros de marketing de email
  - Para Crear una URL de Seguimiento
  - Para Crear una Campaña usando el Asistente de Campaña
  - Corriendo Diagnósticos de Campaña
  - Administrando Campañas
  - Ver el Estado de la Campaña
  - Creando Formas Web para Candidatos
- Módulo de Proyectos
  - Creando un Proyecto
  - Crear una Tarea de Proyecto
  - Administrar un Proyecto
- Módulo RSS
  - Administrar sus Fuentes de Noticias RSS
- Módulo Cuadro de Mando
  - Modificar las graficas para requisitos particulares
  - Para manejar las graficas
- Módulo de Empleados
  - Crear un Empleado
  - Administrar Registros de Empleados
  - Convertir un Empleado a Usuario
- Importar y Exportar
  - Exportar Contactos de su Actual Administrador de Contactos

- Importar Datos de Cuenta
- Exportando Datos
- Exportar Datos de CRM

La única problemática con la que me enfrenté durante las capacitaciones es que aunque, se impartían en las oficinas del cliente, este no siempre contaba con las condiciones para impartir el curso. En más de una ocasión no contaban con proyectores, o bien como trabajaban desde computadoras de escritorio, era imposible impartir el curso con un ordenador.

La forma en que ataqué dichas situaciones fue:

- Actualizar los manuales de usuario, agregando más imágenes de ejemplo.
- Enviar los manuales en electrónico con anticipación y describirles los temas a revisar en la capacitación.
- Llevar manuales impresos para las personas que no tendrían acceso a su PC durante la capacitación.

Capacitación a nivel administrador

Consistía en capacitar al personal con rol de "administrador" las funciones del CRM para administrar el sistema, normalmente al personal que tendría este rol, se le impartían ambos cursos. Esta capacitación contenía los siguientes temas:

- Introducción
- Administración del Sistema
    - Sistema
    - Configuración del Sistema
    - Planificador
    - Herramientas de Diagnostico
    - Asistente de Actualizaciones
    - Configuración Regional
    - Copias de Seguridad
    - Reparación
    - Monedas
    - Cargador de Módulos
- Usuarios
    - Administración de Usuarios
    - Administrador de Roles
- Email
    - Configuraciones de Email
    - Administración de Cola de Email

    - Email Entrante
    - Configuración de Email de la campaña
- Herramientas de Desarrollo
    - Estudio
    - Editor de Listas Desplegables
    - Configurar Pestañas
    - Renombrar Pestañas
    - Configurar Grupos de Pestañas
    - Constructor de módulos
    - Portal
    - Migración de Campos Personalizados

Como esta capacitación normalmente iba dirigida a una menor cantidad de personas, (máximo a 3 personas) y además llevaba otro grado de dificultad. Se impartía a nivel personalizado directamente en el equipo de cada usuario.

1.5. Implementación del OpenbravoPOS

Análisis

Este software de punto de venta es muy amigable. Para este software debía recolectarse la siguiente información:

- Tipo de giro (tienda de artículos o restaurante).
- Productos.
    - Unidades.
    - Fotografías.
- Número y nombre de vendedores.
- Datos del administrador.
- Equipo disponible.
- Datos de almacén.
- En caso de ser un restaurante número de mesas y la distribución de las mismas.

Implementación

Toda esa información debía integrarse al punto de venta, no es necesario utilizar ninguna herramienta de desarrollo, al instalar la aplicación, sólo era cuestión de "personalizar":

- Personalizar los logotipos con los del cliente.
- Capturar los productos y sus características.
- Agregar la foto de cada producto.
- Configurar los datos del almacén.

- En caso de ser restaurante, configurar los pisos y mesas.

Ilustración IV. Ejemplo de la pantalla de OpenbravoPOS configurado para restaurante. (Elaboración propia, 2014).

Como toda la información del punto de venta se maneja por medio de la BD, después de cargar la información del cliente se exportaba la base de datos. Cabe señalar que siempre se instaló en laptops o PC con las que ya contaba el cliente. Los pasos a seguir durante la instalación son:

- Instalar el Ambiente:
  - Java 1.6.0
  - Postgres 8.2
  - OpenbravoPOS (cuenta con un asistente de instalación)
- Crear base de datos en Postgres 8.2.
- Cargar base de datos previamente exportada en la base que se creó anteriormente.
- Configurar la BD en OpenbravoPOS
- Reiniciar OpenbravoPOS.

Capacitación

Durante el proceso de capacitación se impartía lo siguiente:

- Antecedentes
- Introducción al Openbravo POS
  - Acceso al sistema
  - Funciones Generales
    - Menús
  - Navegación
- Sincronización de Productos y Clientes
- Operación del sistema (Ventas)
  - Alta de clientes
  - Nueva venta (ticket)

    - Devolución
    - Factura extemporánea
    - Descuentos
- Compras
    - Confirmación de traspasos
- Actividades Finales
    - Cierre de Caja
    - Sincronización de Ventas (Tickets)
- Reportes
    - Cierre de Caja
    - Reportes de Inventario
        - Almacenes
        - Existencias
        - Mínimo de Existencias
        - Diario de Existencias
        - Productos
        - Etiquetas del producto
        - Catálogo de Productos
        - Lista de Precios
        - Sugerencia de Compra
        - Movimientos de Almacén
- Reportes de Ventas
    - Caja por Vendedor
    - Ventas de Productos
    - Gráfico de Ventas
    - Diario de Tickets
    - Ventas de Producto (Gráfico)
- Reportes de Clientes
    - Clientes
    - Clientes Morosos

Este punto de venta permite identificar en un restaurante la ocupación y distribución de las mesas, además de asociar la orden de compra a cada mesa.
La única problemática encontrada en este software es que no cuenta con la función de formulas, para restaurantes. Por ejemplo, si se venden unas "enchiladas", la afectación a almacén no se refleja a nivel ingredientes, en un escenario "ideal" se esperaría que descuente 3 tortillas, 250 gr de pollo, 10 ml de crema, etc. Así que sólo quedaba registrada la solicitud de las mesas, pero no se actualizaba el inventario.

1.6. Implementación del BPM Karomi

Un software BPM tiene como objetivo automatizar los procesos de negocio de la empresa, de tal manera que estos se ejecuten con velocidad y sin desviaciones. ESOFTWARE ganó una licitación para implementar el BPM Karomi en una organización gubernamental, encargada de los servicios de agua potable, alcantarillado y saneamiento. Es importante aclarar, que este es el único software implementado por ESOFTWARE que no es OpenSource. Sin embargo la experiencia en la implementación fue, tan reveladora, que consideré de suma importancia mencionarlo.

El objetivo era implementar Karomi BPM en los procesos de bacheo, reparación de fugas y contratación de tomas de agua. Además de las oficinas centrales (donde se atendían las solicitudes de reparación de fugas y contratación de tomas de agua), tenían instalaciones en diferentes puntos del área que les correspondía, por lo que era necesario que todas las instalaciones, tuvieran acceso al sistema para llenar los formatos en línea y que toda esta información se concentrara en la base de datos.

Mi rol principal consistía en estar en contacto permanente con el cliente, fui la encargada de realizar el análisis, aplicar las pruebas funcionales, acompañar al cliente durante las pruebas de usuario, impartir la capacitación y fungir como intermediario entre el cliente y el equipo de desarrollo.

Análisis

Durante la etapa de análisis entrevisté a los empleados y/o jefes de cada una de las áreas que se asignaron para el proyecto por parte del cliente, no existían cuestionarios que me sirvieran de guía para realizar dichas entrevistas, el objetivo de mis entrevistas era "entender el proceso" para obtener:

- Diagrama de flujo del proceso.
- Participantes en el proceso.
- Interacción entre los procesos.
- Formularios utilizados.
- Documentos adicionales utilizados (Por ejemplo los requisitos para la contratación de una toma de agua).

Por cada proceso que terminaba, se creaba un documento de "análisis", mismo que debía enviarse a las áreas correspondientes y a los directivos. Si ellos detectaban alguna inconsistencia se nos notificaba y se aplicaban los cambios solicitados, hasta que estuvieran "aprobados" por los directivos.

Construcción

Cuando tenía los documentos de análisis aprobados, estos eran enviados al equipo de desarrollo. Igual que en el caso del ERP, si tenían alguna duda sobre los documentos, estas se aclaraban hasta que quedaran claros los procesos y el alcance.

Con esta información el equipo de desarrollo agregaba al software los procesos con sus respectivos formularios y documentos. Cabe señalar que la forma de agregar los procesos a BPM Karomi[4], se realiza por medio del mismo software, es decir, no se modifica el código.

Pruebas Funcionales

Las pruebas aplicadas, no estaban sujetas a ningún tipo de matriz de pruebas, o casos de uso, estas se realizaron, en base a los documentos de análisis.
Los defectos detectados se reportaban vía correo electrónico al equipo de desarrollo, quien a su vez los corregía y notificaba cuando ya estaban atendidos. Se encontraron defectos de criticidad baja, principalmente en las pantallas de formularios. Los defectos consistían en su mayoría "faltas de ortografía".

Con el proceso implementado en la herramienta, se reportaron los defectos uno por uno, con el detalle del defecto y la pantalla o formulario donde se encontró. Sin embargo con los procesos siguientes los defectos "ortográficos" no sólo se repitieron, si no que aumentaron.

Ante esta situación, al estar fungiendo el rol de tester, solicité que se "revisara y corrigiera" el tema de ortografía previo a las pruebas funcionales. La respuesta de desarrollo fue que "lo revisarían", pero los defectos ortográficos siguieron apareciendo constantemente.

Para cuando nos dimos cuenta se había terminado el tiempo de "pruebas" y había llegado la fecha de las pruebas de usuario. El problema se planteó al gerente de ESOFTWARE, quién estableció como estrategia que el equipo de desarrollo, corrigiera lo "más posible" y "como quedara" se llevaría a pruebas con el cliente.

Pruebas de Usuario

Durante las pruebas de usuario, se detectaron:

- Errores de funcionalidad en el proceso "Atención de solicitudes", pues cuando el encargado del área lo vio, se dio cuenta de que el proceso plasmado era incorrecto. La causa de esto es que cuando se realizó el levantamiento, la persona que asignaron para explicar el proceso, era de nuevo ingreso y aún estaba en "capacitación", pues el encargado del área "estaba de vacaciones".

[4] *Los asesores hindúes de KaromiBPM nos explicaron que de hecho, Karomi significa "hazlo tu mismo".*

Se tuvo que hacer nuevamente el levantamiento en dicha área y se encontró que en realidad eran dos procesos diferentes para la misma. Incluso uno de esos procesos no estaba dentro del alcance de implementación. Fue necesario entonces volver a crear el proceso desde cero en karomi. Lo cual implicó un "re-trabajo" por ambas partes.

- Defectos ortográficos, algunos fueron detectados en uno de los procesos del director del área, lo cual tuvo como consecuencia un correo electrónico por parte del director, solicitando "encarecidamente" que revisaran el tema ortográfico.
- Solicitud de cambios por parte del cliente. Como cambiar un texto por otro o incluso eliminar algún "paso" del proceso para simplificarlo.

Los hallazgos detectados en estas pruebas se notificaban vía correo electrónico al equipo de desarrollo. En el segundo ciclo de pruebas de usuario se detectaron nuevamente.

- Errores ortográficos y de redacción (Por ejemplo: "**El** órdenes de pago…", "**recivo**", falta de acentos, entre otros).
- Solicitud de cambios por parte del cliente.

Aunque el cliente ya había solicitado cambios previamente, debido a que el problema "ortográfico persistía", no se pudo negar la solicitud de cambios, sin embargo si se solicitó que fuera la "ultima" para evitar re-trabajos. Durante el tercer ciclo de pruebas ya no se detectaron defectos ni solicitudes de cambio por parte del cliente.

Capacitación e Instalación.

Una vez que se aprobaron las pruebas de usuario, se comenzó con los procesos de capacitación e instalación. Mientras se realizaba la instalación y configuración de los usuarios de Karomi en el servidor del cliente, se comenzó con la capacitación a nivel usuario, utilizando el servidor de prueba de ESOFTWARE.

Esta capacitación a nivel usuario se impartió a alrededor de 90 personas en sesiones organizadas de acuerdo a su proceso, pues los formularios y flujos aplicaban diferente para cada uno. No se me permitió llevar manuales para todos los usuarios, debido a una política de "paper-less", sin embargo se les envió a los directivos los manuales previamente, con la finalidad de que estos se los hicieran llegar en electrónico a su personal. La capacitación a nivel administrador se impartió solo a una persona y fue de manera personalizada.

A NIVEL USUARIO

- Antecedentes
    - Introducción al Karomi BPM
        - Acceso al sistema
        - Navegación
            - Government.

        - My Setup
        - Barra de Navegación Entre Carpetas:
    - Para Vista de Registros.
    - Tipo de Campos que se manejarán en los formularios
- Campos obligatorios:
- Para iniciar procedimientos:
    - Display Workflow
    - Manejo de Registros de Flujos de Trabajo.
- Revisión de Flujos
- Reportes

A NIVEL ADMINISTRADOR

- Antecedentes
- Introducción
- Acceso al sistema
    - Navegación
    - Funciones Generales
        - Opciones de Administración
    - Configuración General
        - Cambiar o establecer la imagen del el banner
        - Cambiar Colores del Portal
        - Agregar/Eliminar pestañas al portal
        - Asignar Pestañas a usuarios
        - Modificar las pestañas asignadas a usuarios
        - Ajustar el orden de las Pestañas
    - Configuración de usuarios
        - Agregar usuarios
        - Restablecer contraseña a usuarios
        - Administrar usuarios
        - Activar, Desactivar usuarios
        - Asignar políticas para la contraseña
        - Asignar privilegios de administrador
    - Administración de la aplicación
        - Administrar Vistas
        - Administración de Reportes
        - Asignar Usuarios a Actividades
        - Asignar usuarios para iniciar procesos
        - Crear formularios
        - Crear flujos de trabajo

Debido a la cantidad de personas, era imposible impartir una capacitación personalizada, así que se nos proporcionó una sala y un proyector para impartirla. Sin embargo me lleve una enorme sorpresa pues mientras que el personal que estaba asignado a las oficinas centrales tenía pocos o ningún problema para entender el funcionamiento del sistema. Aquel personal que se encontraba en las unidades externas a las oficinas centrales, tenían muchos problemas para entender no sólo el sistema, si no la forma de "acceder al sistema".

El problema más grave estaba en la unidad de bacheo, pues durante esa sesión pude darme cuenta de las carencias con que trabajaban en esa área. Durante la capacitación a este equipo, pude notar el enorme interés que tenía ese equipo de utilizar la herramienta, llegaron con cuaderno y pluma, respetaban el tiempo asignado de la agenda para preguntas y respuestas, escuchaban atentamente y procuraban no quedarse con dudas.

Sin embargo, cuando terminó la capacitación y ya no les quedaba ninguna duda del "sistema", me comenzaron a preguntar otras cosas, por ejemplo: "¿Y sabe cuando nos proporcionarán las computadoras y él internet?" incluyendo una pregunta que me dejó sin palabras: "Yo no sé ocupar la computadora, pero si vamos a usar este programa, entonces supongo que también nos va usted a capacitar para aprender a utilizarla ¿Verdad? Quería preguntarle si puede darnos hojas para ir estudiando antes del curso". Entonces les expliqué que el alcance solo incluía esa capacitación, pero que podríamos intentar resolverlo. Por un lado les pedí que solicitaran ellos a sus directivos la capacitación y equipamiento, mientras que yo como consultor prepararía una cotización para el paquete de capacitación y equipamiento, con el objetivo de ampliar el alcance e intentar resolver la situación.

De inmediato avise a mi jefe de la situación, el me dio la razón, sin embargo recalcó que el alcance sólo abarcaba la implementación y capacitación, además me informó que estaban afrontando problemas para instalar Karomi en los servidores del cliente, por lo que este no había depositado los pagos acordados. Finalmente se terminó con la instalación y configuración de los usuarios en el servidor del cliente y se me envió a realizar una validación rápida de pantallas y formularios. Detecté que había un error, pues las últimas modificaciones solicitadas por el cliente, no se veían reflejadas, se había instalado una versión anterior a la que aprobó el cliente.
Lo notifiqué de inmediato vía telefónica a mi jefe y al equipo de desarrollo. La respuesta fue clara "no informes nada aún, vamos a revisar el tema lo más pronto posible".

Hasta ahí llegó mi participación pues solicitaron mi apoyo en la capacitación de una implementación de SugarCRM, por lo que ya no estaría asignada a la implementación de Karomi. Posteriormente detecté que no se había implementado la corrección en los servidores del cliente y que de hecho, la herramienta no les estaba dando el resultado que esperaban.

Conclusiones

Los principales errores cometidos fueron:

- Se nos asignó a personal de nuevo ingreso durante el análisis.
- La dirección del cliente no estaba enterada de los procesos de sus diferentes áreas. Recordemos que se detectaron inconsistencias cuando ya estaban implementados los procesos en Karomi, pero que esta implementación se realizó con base a documentos "aprobados por la dirección". Lo cual obligó a invertir más tiempo y recursos.
- Los objetivos de la dirección del cliente eran incorrectos, implementar un sistema al que gran parte de sus empleados no podrán acceder por falta de equipo o conocimientos sólo les generará más problemas.
- Baja calidad en el proceso de desarrollo, las faltas de ortografía nos consumieron gran parte del tiempo de construcción y pruebas. Además de que al final se instaló una versión anterior a la correcta.
- Se detectaban defectos, pero no se manejaba un seguimiento por defecto, ni reportes estadísticos de los mismos.

La forma en que debían evitarse dichos errores de acuerdo a mi experiencia actual:

- Es necesario identificar y gestionar los riesgos durante la implementación de cualquier sistema, de tal suerte que todos los integrantes del equipo de implementación se manejen bajo una política de prevención de riesgos. Cabe mencionar que el equipo de implementación debe incluir al equipo de consultoría y al equipo de trabajo del cliente.
- El análisis de un área de la empresa debe hacerse por medio de una investigación que incluya todo o la mayoría del personal de esa área, pues aunque es importante del lado del cliente asignar a un responsable para el análisis, tomar la experiencia sólo de una persona del área, aumenta el riesgo de no detectar la raíz de los problemas de la misma.
- Deben establecerse procesos de control de cambio del alcance más estrictos, que promuevan acuerdos entre el cliente y el proveedor ; además deben incluir la gestión de riesgos. De manera que si se presenta la necesidad de cambios, queden establecidos los lineamientos a seguir para resolver las situaciones que se presenten durante la implementación.
- Las empresas deben poner especial cuidado en sus contrataciones, pues un directivo debe ser capaz de identificar los problemas del área, así como la causa raíz de los mismos para poder resolverlos. El poco conocimiento o interés durante el levantamiento de información del área de bacheo, repercutió en un costo de inversión muy alto, que no resolvió el problema real que era la falta de capacitación e infraestructura para su personal.

- El proceso de desarrollo debería complementarse con documentación como:

- Casos de uso.
- Matrices de prueba.
- Seguimiento de defectos.
- Proceso de Control de Calidad.
- Gestión de Riesgos.
- Control de Cambios.

1.7. Comentarios Finales

La mayor parte de los problemas con los que me topé durante mi estancia en esta empresa, fueron principalmente por un "Análisis e investigación escaso" de la situación de los clientes, es decir, sabíamos bien que datos necesitábamos recolectar para implementar el sistema, pero para ello era necesario también "adecuar" los procesos del cliente. Era necesario ofrecer un análisis que les ayudara a encontrar y resolver los problemas, lo cual nos permitiría implementar un sistema que cumpla con su función, mejorar la situación del cliente ganando así su confianza e incluso aumentar la posibilidad de un nuevo contrato.

Con el tiempo me enviaron cada vez a más entrevistas de análisis y descubrí que los formularios establecidos por ESOFTWARE no eran suficientes para obtener la información necesaria. Para lograr un levantamiento exitoso, era necesario un cambio de enfoque.

Decidí entonces que mi objetivo efectivamente era entender el proceso, pero no se limitaba a llenar los formatos y diagramar los procesos. Era determinante identificar las fallas, las inconsistencias, los problemas con que se enfrentaban los empleados de dichas áreas todos los días.

Los hallazgos fueron determinantes, encontré procesos descritos que no se llevaban a cabo, áreas que duplicaban información, personas que no conocían en qué consistían sus responsabilidades, personas que no hacían su trabajo, riesgos con proveedores, riesgos con clientes, herramientas sin utilizar, personas sin capacitación, etc. En más de una de las entrevistas se encontraron problemáticas que no se habían incluido en el alcance, o bien se descubría una solución diferente, a la establecida por los líderes de implementación por parte del cliente.

Con el tiempo logré identificar algunos problemas constantes en las empresas donde se realizaba el análisis. De estas el 100% eran Pymes:

- Falta de unidad en la información.
- Escasa o nula documentación de los procesos.
- Pocos procesos automatizados, es decir demasiados procesos manuales y manejo de información impresa.

- Poco personal.
- Exceso de acciones correctivas y ausencia de acciones preventivas.
- Inestabilidad.
- Problemas de comunicación interna.
- Alta rotación de personal.
- Omisiones a los procesos existentes.
- Ausencia de objetivos organizacionales.
- Objetivos inadecuados o incoherentes con las políticas de la empresa.

La realidad es entonces que, para aplicar la implementación de un software empresarial, particularmente una que afecte a más de un área de la empresa, es necesario hacer un trabajo de consultoría a fondo. De lo contrario se estaría ofreciendo solamente la "instalación y carga de datos", lo cual no ayuda a resolver ninguno de los problemas de la empresa, los cuales se agravarán con el tiempo a pesar de contar con el software más robusto y eficiente del mercado.

Esto rompe la promesa que se le hace al cliente durante el proceso de venta, bajando el nivel de calidad en el servicio al cliente, obteniendo como consecuencia la perdida de la confianza del cliente y con ello la oportunidad de crecimiento de la empresa consultora.

Capítulo 2. Cambio de Rumbo "Procesos de Calidad del Software"

Después de la experiencia que tuve con la implementación de sistemas empresariales, decidí cambiar el rumbo hacia procesos de "Aseguramiento de la calidad en software". Me dieron la oportunidad en una empresa muy importante del sector financiero en México, a la cual llamaré EBOLSA.

Ingresé al área de QA (Quality Assurance) de EBOLSA con la experiencia de un consultor, así que el primer reto era aprender a ser un "Tester". Encontré que los tester´s en esta empresa tenían un nivel muy alto de compromiso y experiencia; conocían perfectamente sus responsabilidades y trabajaban orientados a objetivos.

1.8. La capacitación

Cuando ingresé a EBOLSA, me dejaron a cargo de un Tester que tenía mucha experiencia en el área, para que me capacitara. Sin embargo, debido a la carga que esta persona tenía, sólo pudo enseñarme el manejo de algunas herramientas como:

- Herramienta de administración de pruebas "Quality Center HP".
    - Cargar script de pruebas.
    - Registrar el avance de las pruebas.
    - Levantar defectos.
- El repositorio de objetos "Serena".
    - Descargar las carpetas de código y documentación.
    - Subir los documentos de QA que correspondían a la evidencia.
- Repositorio para documentación de seguimiento de los requerimientos "Share Point".
    - Crear y actualizar la bitácora de avance del requerimiento. (Aquí quedaba registrado el detalle de avance, retrasos, acuerdos, etc. para que el líder pudiera ver el estatus si lo necesitaba).

También puso a mi disposición los procesos del área para que yo tuviera oportunidad de leerlos e ir asimilando las que serían mis tareas. En estos procesos estaba descrito que debían realizarse las siguientes pruebas:

- Pruebas de código y compilación.
- Pruebas funcionales y de regresión.
- Pruebas UAT en paralelo (Sólo si se habían definido previamente).
- Pruebas UAT.
- Revisión de Checklist.
- Entrega a Change Management.

Sin embargo me di cuenta de que me faltaba información, pues ya sabía que tenía que hacer, pero no había manuales suficientes que explicaran la metodología para realizar dichas actividades.
Pasadas un par de semanas, el tester al que yo estaba asignada tomó sus vacaciones, así que me quedé atendiendo sus últimos requerimientos, mismos que ya estaban prácticamente para entrega a Change Management, los había dejado muy avanzados y llegado el momento en que debían "entregarse" yo no sabía lo que tenía que hacer. Solicité ayuda a otros tester´s y me explicaron que para esa entrega restaba seguir los siguientes pasos:

- Adecuar la documentación pendiente.
- Llenar la solicitud de liberación en la herramienta "Service Desk".
- Revisión "Checker" por medio de otro tester.

Entre dos tester´s me ayudaron para hacer la entrega, mientras uno me iba explicando lo que tenía que llenar en los documentos, a la par el tester que realizaba el "Checker", nos avisaba de los errores que detectaba en la documentación. Así fue mi primera entrega al departamento de "Change Management".

Durante los siguientes 3 meses, continué pidiendo ayuda y preguntando las dudas que tenía, así que fui aprendiendo de la experiencia de varios tester´s. Cada tester tenía asignados varios sistemas, en los que ya había obtenido alguna experiencia, tanto en el sistema como en la interacción con el equipo de desarrollo correspondiente.

A los tester´s les asignaban dos tipos de actividades:

- **Revisiones SQA:** En base a un "Checklist de pruebas" se debía validar la documentación, dependiendo de en qué fase se encontrara el requerimiento. Se nos asignaban 2 o más al día.

- **Atención de Requerimientos:** La atención de los requerimientos consistía en aplicar las pruebas al software, dependiendo del alcance del requerimiento.

Revisiones SQA

El proceso completo de desarrollo de software incluía las siguientes fases:

- Planeación.
- Diseño.
- Construcción.
- Pruebas y Validación.
- SQA

Durante cada una de estas fases, se generaba determinada documentación, que quedaba como evidencia de la ejecución exitosa de dicha fase, esta evidencia consistía en documentos y aprobaciones por parte de las áreas involucradas. Es esa documentación junto con las aprobaciones la que se validaba en el proceso de "Revisión SQA". Las actividades que se realizaban durante este proceso eran:

- Asignar revisión SQA.
  El líder de SQA era el encargado de programar y asignar las revisiones, mismas que quedaban registradas en el calendario de revisiones. En este calendario quedaba definido:
  - ID Sistema.
  - ID Requerimiento.
  - Fase de revisión.
  - Fecha de revisión.
  - Tester asignado.

- Revisar documentación.
  El tester revisaba con base a un documento llamado "Checklist SQA" que la documentación necesaria estuviera en el repositorio de objetos, y que además tuviera los datos correctos.

- Registrar hallazgos.
  En el documento Checklist SQA se registraba que documentos se habían validado exitosamente y cuáles no. En el caso de los documentos que no se recibían o se recibían con errores, se registraba en ese mismo documento un NCI (No conformidad en los documentos), se anotaba:
  - Detalle del NCI.
  - Criticidad.

- Notificar término de revisión SQA.
  Se notificaba al responsable del NCI el término de la revisión SQA, agregando a la notificación:
  - NCI´s detectados.
  - Fecha límite de corrección de los mismos.
  - Se adjuntaba el Checklist SQA del requerimiento.
  - Si no se detectaban NCI´s o se corregían los NCI´s detectados, entonces se enviaba notificación de Término y "Fin de Fase".

- Reporte de NCI´s.

Este reporte se realizaba de manera semanal, por medio de un proceso semi-automático por los tester´s asignados a esta tarea, su responsabilidad consistía en cargar la información de todos los checklist en una macro de Excel. Existía un reporte por equipo de desarrollo, del que se obtenían las estadísticas por Fase de ejecución:

- NCI´s abiertos
- NCI´s vencidos
- NCI´s no atendidos

Esta información se enviaba por equipo a las áreas de Desarrollo, Arquitectura, Change Management y QA.

Es importante mencionar que los NCI´s detectados en esta revisión no representaban un límite para la liberación, es decir, un requerimiento podía liberarse aunque tuviera NCI´s abiertos, vencidos o no atendidos.

Pruebas de software

Las actividades que se realizaban durante este proceso eran:

1. Asignar el requerimiento al Tester:

- El "Tester Leader" asignaba el requerimiento al Tester de acuerdo a su planeación. Le notificaba al tester el alcance y las fechas programadas de entrega por medio de la herramienta Quality Center. El alcance estaba definido por:
    - **Tipo de modificación**. Podía ser un mantenimiento, donde sólo se modificaban algunos objetos, o incluso migraciones completas, donde se sustituía todo el código.
    - **Fecha programada de liberación**.
    - **Criticidad del requerimiento:**
        - Normal Schedule (NS). La prioridad era normal.
        - Business Critical (BC). Prioridad alta, para levantar un requerimiento de este tipo, el equipo de desarrollo debía convocar a una junta con los directivos de diferentes áreas, con el propósito de informar el motivo por el que necesitaba esa prioridad y la estrategia a seguir.
          Sólo se aprobaba si todos los participantes en esa junta estaban de acuerdo. Esta prioridad era muy alta, por lo que podía dejar de atenderse otros requerimientos para atender el BC. Durante dicha junta se definían las pruebas a aplicar y el lapso máximo de entrega.
        - Break Fix Emergency (BFE). Esto significaba que existía una emergencia a nivel productivo, es decir que algún sistema falló y debía corregirse urgentemente para evitar pérdidas económicas. Este tipo de requerimiento también necesitaba aprobaciones, entre los líderes de las diferentes áreas acordaban los cambios y las pruebas mínimas a aplicar.

Estas modificaciones se aplicaban directamente en producción, la responsabilidad de QA consistía en aplicar los cambios en el ambiente de QA junto con las pruebas acordadas e informar sobre los resultados de las mismas en máximo 24 horas.

- SCRUM[5]. Este tipo de requerimiento sólo era atendido por un selecto grupo de equipos de desarrollo y sólo los tester´s con más experiencia atendían requerimientos de este tipo, implicaba la aplicación de una metodología ágil de desarrollo, en la que se daba prioridad al desarrollo y las pruebas, para después en base a ello actualizar la documentación.

2. Captura de pruebas programadas en la herramienta Quality Center, esta tarea la llevaba a cabo el tester asignado.
   Las pruebas que se aplicarían al requerimiento debían registrarse en la herramienta Quality Center, pues por medio de esta herramienta se daba seguimiento al avance de las pruebas y los defectos registrados de la misma. Dependiendo del tipo de requerimiento y alcance, se registraban las pruebas a aplicar. Podían ser:
   - Pruebas de código y compilación. Existía una plantilla predeterminada en QC.
   - Pruebas de seguridad. Existía una plantilla predeterminada en QC.
   - Pruebas de funcionalidad y regresión. Se cargaban la matriz de pruebas definidas en la documentación del requerimiento.
   - Pruebas UAT. Se registraba un caso de prueba con el nombre "Pruebas UAT".
   - Pruebas en Paralelo. En requerimientos BC o urgentes se omitían las pruebas funcionales y de regresión, así que se aplicaban junto con el usuario.

3. Aplicar pruebas de código y compilación.
   a. Descargar el código del repositorio de objetos "Serena".
   b. Realizar el cuadre de objetos, es decir validar que los objetos modificados u objetos nuevos detectados en el código, correspondan al listado de objetos y versiones, definidos en la documentación del requerimiento.
   c. Validar que el código no contenga datos sensitivos, por ejemplo: Datos de clientes.
   d. Validar que el código no contenga datos de riesgo, por ejemplo: Datos de servidores productivos, contraseñas, correos electrónicos de producción, etc.
   e. Compilación de objetos.
      i. Objetos web y cliente-servidor: Era extremadamente importante identificar que al compilar los objetos actualizados, estos no

[5] *Este proceso pretendía ejecutar una metodología ágil de desarrollo de software, sin embargo no se apegaba completamente a la metodología SCRUM estándar, debido a que aún estaba en proceso de maduración.*

modificaran objetos no incluidos en la liberación. En caso de que dichos objetos se vieran afectados por el proceso de compilación, se validaba que el tamaño del objeto, fuera equivalente al tamaño de la versión anterior del objeto. Esta comparación se realizaba por medio de una herramienta de comparación de código, con la que se generaba un reporte, en el cual debían verse reflejadas las diferencias solamente en el código listado en la documentación del requerimiento. Dicho reporte debía anexarse a la documentación como evidencia de QA.

ii. Objetos de base de datos: Validar que la ejecución de los query´s en base de datos, no arrojara ningún error. Los query´s de ejecución exitosa o fallida, debían agregarse a la documentación como evidencia de QA.

Las validaciones de código eran de suma importancia, pues si se subían datos productivos o de clientes al servidor de pruebas de QA, se insertaban riesgos de "seguridad".

Al terminar se debía registrar el avance en Quality Center, así como los defectos detectados.

4. Pruebas de Base de datos.

   Se aplicaban pruebas unitarias a los componentes liberados vía base de datos. Dichas pruebas estaban definidas el Script de pruebas de la documentación del requerimiento, el resultado de estas pruebas debía registrarse directamente en Quality Center junto con los defectos detectados.

5. Aplicar pruebas de seguridad.

   Las pruebas de seguridad consistía en:

   - Intentar acceder al sistema con usuario y password correctos.
   - Intentar acceder al sistema con usuario correcto y password incorrecto:
     - Password incorrecto con caracteres alfanuméricos.
     - Password incorrecto con caracteres numéricos.
     - Password incorrecto con caracteres especiales.
   - Validar que no se pudiera ingresar con un mismo usuario en dos terminales diferentes.
   - Validar que se bloqueara el usuario después de 7 intentos con password incorrecto.
   - Validar registro de accesos incorrectos y bloqueos por exceso de intentos fallidos, en la bitácora del sistema de seguridad del servidor.

   El resultado de la ejecución de dichas pruebas se capturaba en imágenes y debían agregarse como evidencias a la documentación de QA.

Al terminar se debía registrar el avance en Quality Center, así como los defectos detectados.

6. Aplicar pruebas funcionales.
    a. Aplicar las pruebas previamente definidas en la matriz de pruebas del requerimiento.

   El resultado de la ejecución de las pruebas correspondientes a objetos tipo Web o cliente servidor, se capturaba en imágenes y debían agregarse como evidencias a la documentación de QA.
   Al terminar se debía registrar el avance en Quality Center, así como los defectos detectados.

7. Pruebas de rollback.
   Durante estas pruebas debían ejecutarse los pasos descritos en la documentación del requerimiento para "regresar a la versión anterior del sistema". Estos pasos se aplicaban en producción cuando se presentaban fallos y no se podían corregir a través de un BFE.

8. Pruebas de regresión.
    a. Aplicar pruebas para garantizar que los componentes que no se modificaron, no hayan sido afectados en su funcionamiento. Normalmente estos casos de prueba eran diseñados por los Tester´s en base a documentación anterior de los requerimientos y en la experiencia de los tester´s, pues cada tester se especializaba en los sistemas que tenía asignados.

   El resultado de la ejecución de dichas pruebas se capturaba en imágenes y debían agregarse como evidencias a la documentación de QA.
   Al terminar se debía registrar el avance en Quality Center, así como los defectos detectados.

9. Pruebas en paralelo.
   Este tipo de pruebas se aplican cuando un mantenimiento se aplica a dos sistemas diferentes, porque que interactúan entre sí. Por ejemplo el sistema de facturación y el sistema de pedidos. Por lo que es necesario aplicar las pruebas al mismo tiempo en dos sistemas diferentes. Normalmente son aplicadas por dos tester´s diferentes y las evidencias se registran en ambos requerimientos tanto en la documentación de QA como en el avance registrado en Quality Center.

10. Notificar al usuario la fecha de las pruebas UAT.
    Avisar al usuario que las pruebas por parte del área de QA habían terminado y podía comenzarse con las pruebas de usuario, en esa notificación se le proponía al usuario

una fecha. El usuario confirmaba el horario de pruebas o bien proponía una fecha diferente con un horario definido.
La respuesta de dicha notificación debía quedar registrada en los documentos de evidencia de QA.

11. Realizar pruebas UAT.
    El usuario se presentaba en el lugar del Tester y realizaba las pruebas funcionales, mas alguna prueba adicional que considerara necesaria.
    Dependiendo del alcance de las pruebas previamente definidas, estas sesiones podían durar desde 15 minutos, para un mantenimiento mínimo, hasta más de una semana para los mantenimientos grandes y migraciones.

12. Pruebas UAT en paralelo.
    Las pruebas funcionales son aplicadas junto con el usuario, es decir, no se aplican las pruebas previas a que el usuario pruebe. Este tipo de pruebas se aplican en requerimientos tipo BC.
    Las evidencias se registran en la documentación de QA y el avance en la herramienta Quality Center.
13. Aprobación UAT.
    Se le enviaba al usuario una solicitud de aprobación de pruebas UAT. Cuando el usuario respondía con su aprobación, entonces se podía comenzar con el proceso de cierre.
    El correo de aprobación del usuario, debía quedar registrado en los documentos de evidencia de QA.

14. Proceso de cierre.
    Durante el proceso de cierre debía validarse:
    - Evidencias de pruebas aplicadas en la documentación generada por QA.
    - Validar que los documentos necesarios para la entrega del requerimiento a "Change Management" estuvieran completos. Esto se realizaba con apoyo de un documento llamado "Checklist único de Pruebas". Mismo que debía ser aprobado por el "Tester Leader". Una vez aprobado, se podía solicitar el "Tipo de Liberación".
    - Revisión del paquete de código a entregar por el antivirus.

15. Tipo de liberación.
    Se solicitaba al PM (Proyect Manager) del equipo de desarrollo la fecha y hora de liberación del requerimiento por medio de una notificación. Cuando el PM nos respondía con la fecha y hora de liberación entonces se podía continuar con la entrega.

16. Entrega a Change Management.
    Se solicitaba por medio de la herramienta "Service Desk" la liberación del requerimiento, a la fecha y hora definida por el PM del equipo de desarrollo. A este proceso lo llamábamos "Copia de instancias".
    Si la solicitud de copia de instancias, no tenía los requisitos y los datos correctos solicitados por Change Management, esta era rechazada. Dependiendo del horario de entrega de la "corrección" y la carga de trabajo de Change Management, existía el riesgo de que no se liberara en la fecha programada. Por lo que era vital que dicha solicitud se entregara con una calidad alta.

Durante el proceso de capacitación con los diferentes tester´s, pude detectar que cada tester trabajaba de manera diferente, sin embargo esto me fue extremadamente útil, porque me permitió entender la importancia de cada documento y cada tarea ejecutada por los tester´s. Al punto de que algunos meses después me asignaron para realizar las revisiones "Checker" del equipo. También pude detectar que los procedimientos necesitaban una actualización urgente, pues en lugar de representar un apoyo para el equipo, se habían convertido en un bache que les complicaba la tarea.

1.9. La estrategia para reportar defectos

Es una regla natural, que exista entre el equipo de desarrollo y el equipo de testing una rivalidad, debido a los roles que tiene cada equipo. Por un lado el equipo de desarrollo adquiere un rol de "creador", al ver su esfuerzo reflejado en un software funcionando para lo que ellos lo diseñaron, los desarrolladores se sienten satisfechos y orgullosos.

Por otro lado los tester´s sabemos que ningún sistema esta exento de defectos, ni si quiera los sistemas de la NASA pueden decir que son "perfectos", por lo que nuestro trabajo es encontrar defectos en los sistemas, validamos que el software sea efectivo y si ya es efectivo que sea "eficiente".

Es decir, mientras el desarrollador se siente orgulloso de su software terminado, el tester es capaz de demostrarle que "aún tiene errores". Esto genera un ambiente de hostilidad, que corresponde a una reacción natural, sobre todo cuando los niveles de trabajo del área están muy altos. De ahí la importancia de reportar defectos con "tacto" y respaldo en evidencias.

Los defectos detectados en el proceso de pruebas del software, que se registraban en Quality Center si podían detener una liberación, la regla establecía que no podía liberarse ningún requerimiento si tenía defectos abiertos. Además en base a los defectos evaluaban el

desempeño del equipo de desarrollo. Por todo lo anterior los defectos debían estar perfectamente justificados, de lo contrario podrían causar malentendidos entre el tester y el desarrollador, junto con reclamos por parte de los PM's del equipo de desarrollo.

Parte de mi trabajo como Tester era hacer un "diagnóstico" previo del problema, pues en ocasiones los defectos podían darse debido a problemas en QA, por ejemplo: Pruebas mal aplicadas; problemas con los servidores de ambiente; falta de objetos instalados; objetos incorrectos instalados o instalados en el servidor incorrecto; etc.

Si se reportaba un defecto y la raíz de este, era responsabilidad de QA, la consecuencia era que se consumían tiempo y recursos adicionales, y atrasaban el proceso de pruebas. Normalmente cuando el equipo de desarrollo descubría la causa real del defecto, escalaba el tema hacia los líderes o jefes, exigiendo la cancelación del defecto y explicaciones del porque se daban estos problemas en QA.

Con toda la evidencia recabada a raíz del diagnóstico del defecto, se registraba el defecto en Quality Center, agregando el detalle y la evidencia del mismo. En un escenario ideal, en cuanto estaba registrado, la herramienta notificaba directamente a los desarrolladores. Sin embargo, en algunos casos los usuarios de dichos desarrolladores no se encontraban registrados, por lo que no les llegaban las notificaciones automáticas. Era hasta que su líder o PM podía re-enviarles la notificación que ellos recibían la información, atrasando el tiempo de atención de los mismos. Esto sucedía sobre todo en casos donde el desarrollador era externo (Lo que era muy común).

Debido a esto durante mi capacitación con el equipo, más de un tester me recomendó que notificara vía correo electrónico los defectos a los desarrolladores, para así garantizar que ellos lo recibieran a tiempo. En base a las necesidades establecidas, diseñé una plantilla para reportar defectos vía correo electrónico, la cual contenía:

- Reporte de defectos.
    - Número de revisión.
    - Pruebas aplicadas exitosamente.
    - Pruebas donde se presentaron defectos.
        - Evidencias
        - Diagnóstico.
        - Número de defecto asignado.
        - Criticidad.

Además de los problemas con Quality Center, otra de las prioridades era dejar evidencia de los ciclos de revisión que se ejecutaban. La justificación es que algunos equipos tenían reincidencia de defectos o incluso, en los casos más graves, la cantidad de defectos iba en aumento. Por

ejemplo había ciertos equipos que cuando corregían un defecto, surgían otros dos. Estos defectos reincidentes obligaban al tester a duplicar o triplicar sus revisiones, lo cual consumía tiempo y esfuerzo de todo el equipo, además de poner en peligro la fecha de entrega del requerimiento.

En las revisiones con este tipo de equipos, debido a que el crecimiento o reincidencia de los defectos, ponía en riesgo las liberaciones, era muy común que los PM´s de desarrollo solicitaran que se les copiara en los correos de notificación de defectos. O bien, se les enviara un reporte de avance al final del día. Por lo anterior mis reportes de avance contenían lo siguiente

- Reporte de avance.
    - Pruebas aplicadas y su estatus.
    - Ciclo de revisión.
    - Defectos pendientes de cierre.
    - Criticidad
    - Pruebas pendientes por aplicar.
    - Notas o comentarios adicionales.

Toda esta información podía extraerse de Quality Center, sin embargo debido a los problemas[6] descritos anteriormente, esta información no llegaba a todos los desarrolladores. Pero no todos los defectos eran responsabilidad del equipo de desarrollo, o del equipo de QA. Algunos defectos eran generados por las inconsistencias en la documentación y otros eran generados por los clientes.

[6] *Desafortunadamente por esa misma razón, los tester´s no registraban el avance conforme terminaban las pruebas. Se concentraban en registrar los defectos y la evidencia de ejecución de pruebas de base de datos. Registraban el resultado de las pruebas positivas hasta la etapa de "Proceso de cierre".*

1.10. La importancia de la documentación

Así como yo establecí mi estrategia para reportar los defectos y el estatus a los equipos problemáticos, cada tester había establecido su forma de trabajo.

De hecho pocos tester´s (o ninguno) cumplían el proceso al "pie de la letra", buscábamos cumplir los mismos objetivos, pero no trabajábamos de la misma manera. Gran parte del problema era que el procedimiento no estaba a la altura de las revisiones de los Tester´s, ni se hacía todo lo que estaba documentado en el proceso, ni estaba documentado en el proceso todo lo que se hacía.

Mientras la carga de trabajo se mantenía alta, la evolución del área continuaba, algunos cambios eran solicitados por los directivos para cumplir "políticas organizacionales", otras las establecían los líderes de QA, en un intento por "disminuir los problemas durante la atención de los requerimientos". Por ejemplo:

- **Debido a políticas organizacionales que debían cumplirse:**
    - **El repositorio de objetos Serena fue sustituido por TFS (Team Foundation Server[7]), debido a que la herramienta había sido declarada como obsoleta por EBOLSA.**

  La política se aplicó, pero no se establecieron los lineamientos para el uso de dicha herramienta. De hecho fue por colaboración de los líderes de QA que se establecieron algunos lineamientos, mismos que quedaron establecidos en una "Notificación vía correo electrónico", pero esto no se reflejó en los procesos involucrados en el proceso de testing. Que en este caso correspondía a los procesos de Desarrollo, QA y Change Management.

  Dentro de los lineamientos establecidos están:

  - La rama del sistema debe contener la siguiente estructura:
    - Integración
        - Objetos Web o Cliente Servidor.
        - Objetos BD.
    - Testing
        - Objetos Web o Cliente Servidor.
        - Objetos BD.
    - Liberación
        - Objetos Web o Cliente Servidor.

[7] *"Team Foundation es un conjunto de herramientas y tecnologías que permiten a un equipo colaborar y coordinar sus esfuerzos a la hora de crear un producto o llevar a cabo un proyecto" (Microsoft, 2014).*

- Objetos BD.
- Documentación
  - Análisis
  - Diseño
  - Construcción
  - Pruebas y Validación
  - SQA

- El código debe estar etiquetado sólo en la rama correspondiente. Por ejemplo, el equipo de desarrollo debe etiquetar su código sólo en la rama Integración.
- La documentación entregada por el equipo de desarrollo para la atención del requerimiento deberá estar dentro de la etiqueta.
- Sólo deben etiquetarse los objetos a liberar.
- La compilación deberá hacerse por medio de la misma herramienta TFS generando un build para cada rama. La responsabilidad de crear todos los build, recae en el equipo de desarrollo responsable de la construcción.

- **Se agregaron las pruebas automatizadas de código por medio de la herramienta de IBM Rational AppScan.**

Las problemáticas a las que nos enfrentamos cuando se estableció esta regla fueron:

- El área de desarrollo no contaba con esta herramienta.
- La herramienta detectaba riesgos de seguridad de acuerdo a mejores prácticas, pero estas no se aplicaban aún en EBOLSA, por lo que los desarrolladores no las habían previsto.
- Debíamos entregar reporte de ejecución de estas pruebas, sin embargo debido al punto anterior, los desarrolladores no las corregían. Así que esto se convirtió en un "mero requisito".

- **Estrategias establecidas por parte de los líderes:**
  - **QA debía participar en las juntas organizadas durante la fase de Diseño.**

En estas juntas participaba el equipo de Arquitectura, el equipo de desarrollo y el Tester responsable del sistema. El objetivo era identificar:

- Alcance de la modificación.
- Alcance de las pruebas.
- Requisitos previos para comenzar las pruebas:
  - Sistemas involucrados.
  - Ambientes especiales.
  - Carga de escenarios en base de datos, entre otros.

- Identificar riesgos en infraestructura previos a la planeación del requerimiento.
- Validar cumplimiento de compromisos.
- Acuerdos para la atención del requerimiento.

Cabe señalar que dichas juntas no estaban documentadas en el proceso de pruebas, sin embargo como las áreas involucradas, obtuvieron resultados positivos con esta estrategia, no generó problemas en su cumplimiento.

- **Se agregaron "Criterios de Aceptación.**

  Debido a que algunos equipos de desarrollo, entregaban los requerimientos incompletos, o con baja calidad, se establecieron los criterios de aceptación. El tester debía revisar:

  - Documentación completa, correcta y etiquetada en TFS.
  - Cuadre de objetos sin errores.
  - Documentación del alcance coherente con lo descrito en la junta de arquitectura y el documento de análisis.

  Si el requerimiento no cumplía con estos criterios, el requerimiento se "rechazaba". El equipo de desarrollo entonces debía corregir y volver a solicitar que se planearla la atención de su requerimiento.

  Esta estrategia si genero problemáticas, debido a que nunca antes se había establecido una política restrictiva de este tipo y esta no estaba establecida en el proceso, sino que había quedado como un comunicado a los PM´s de desarrollo. Algunos equipos incluso utilizaron esto como argumento para "exigir" la atención de sus requerimientos, junto con una exageración en la criticidad de los mismos. Lo que sucedía a continuación es que los jefes de QA nos enviaban la instrucción de continuar con la atención de esos requerimientos.

  A pesar de estos problemas esta estrategia nos ayudo a sacar datos estadísticos de cuáles eran los equipos que entregaban con baja calidad y cuales siempre cumplían con los criterios de aceptación. Poco a poco fue aumentando la cantidad de equipos que respetaban esta nueva política.

En medio de toda esta evolución se generaron muchos problemas, por un lado a QA se le solicitaba "exigir el cumplimiento de las nuevas directrices", pero los procedimientos no se actualizaban, todo se quedaba (en el mejor de los casos), en notificaciones hacia el equipo de desarrollo, Arquitectura y Change Management.

Por lo que en más de una ocasión se generaron discusiones con los PM´s de los equipos de desarrollo, exigiendo la cancelación de los defectos que surgían por estas nuevas disposiciones. Estos temas llegaron a escalarse a niveles muy altos, por lo que terminábamos cerrando los defectos sin que estos se corrigieran y en el peor de los casos, nos veíamos obligados a continuar atendiendo requerimientos que no cumplían con los criterios de aceptación. Generando un riesgo latente de atraso en la entrega, para ese requerimiento, los requerimientos atendidos en paralelo y los requerimientos planeados en un futuro.

En algunas ocasiones se cerraban los defectos, con un compromiso de corrección para siguientes liberaciones. Sobra decir que no siempre cumplían las promesas de corrección, debido a la carga de trabajo.

Es cierto que algunos equipos tomaban esos compromisos a la ligera, pero para la mayoría cumplir esos compromisos implicaba incumplir con tiempos de entrega ya establecidos con el cliente. Pues si bien había defectos que podrían corregir en un día, existían otros como el tema de adecuar el código para "compilación TFS", que les obligaba a modificar la estructura completa de su código, lo cual representaba un consumo de tiempo y recursos adicionales que no tenían previstos.

1.11. Las necesidades del cliente

Los objetivos definidos en la documentación del requerimiento, eran establecidos por desarrollo en base a las solicitudes de los clientes. Que eran usuarios de las diferentes áreas de negocio de EBOLSA.

Estos clientes definían sus necesidades y la prioridad de las mismas. De hecho los clientes tenían la responsabilidad de "aprobar" el alcance descrito en la documentación del requerimiento, misma que era llenada por el equipo de desarrollo. Por desgracia, la mayoría lo aprobaba sin revisarlo, de hecho los equipos de desarrollo sabían que los clientes daban su aprobación sin revisar el documento. Esta situación sumada a la carga de trabajo, tuvo como consecuencia que los equipos de desarrollo, construyeran la documentación de todas las fases del desarrollo y solicitaran las aprobaciones de este tipo de documentos uno o dos días antes de entregar a QA el requerimiento.

Estos descuidos por parte del cliente y el equipo de desarrollo, tuvieron como consecuencia:

- Solicitud de cambios por parte del cliente en etapas finales de la etapa de construcción.
- Solicitud de cambios por parte del cliente durante las pruebas UAT.
- Inconsistencias entre las especificaciones de los requerimientos contra las modificaciones aplicadas al código.

- Aumento de la carga de trabajo del equipo de desarrollo debido a re-trabajos.
- Aumento en la carga de trabajo del equipo de QA debido a re-test.
- Retrasos en los tiempos de entrega de desarrollo.
- Retrasos en los tiempos de atención de requerimientos por parte de QA.
- Retrasos en liberaciones.

Esto se debía a una mala aplicación del "Control de cambios", el cliente no era consciente del impacto de sus solicitudes y el análisis aplicado no lograba descubrir las necesidades puntuales del cliente.

Si a esta situación sumamos el problema de los procesos no actualizados, entonces entenderemos el por qué QA tenía una mala reputación ante los equipos de desarrollo, los clientes y por ende los directivos de nuestra área.

1.12. Conclusiones

Los problemas identificados:

- Ausencia de un programa de capacitación para personal de nuevo ingreso.
- Evolución del área no tomaba en cuenta la actualización de los procesos (Este problema aplicó a todas las áreas).
- La carga de trabajo no permitía a las diferentes áreas actualizar sus procesos.
- Los tester´s trabajaban sin apego al proceso y sin unicidad.
- El enfoque de los directivos era "exigir resultados", pero no escuchaban las peticiones o sugerencias de los tester´s o desarrolladores.
- Ausencia de un proceso de "control de cambios".
- Análisis escaso y clientes que no tomaban en cuenta el impacto de sus solicitudes.

La forma en que se podrían haber corregido:

- Actualizar los procesos del área para que todos trabajen bajo los mismos lineamientos.
- Establecer objetivos para toda el área de TI.
- Establecer estrategias de comunicación y generación de confianza entre las áreas incluyendo a los clientes.
- Establecer indicadores para evaluar a cada área y encontrar los problemas reales.
- Establecer un plan de trabajo que apoye a las diferentes áreas a actualizar los procesos.
- El enfoque de la dirección debería ser la confianza hacia su fuerza de trabajo.
- Establecer un proceso de control de cambios.
- Establecer un programa de capacitación

Un error constante en los directivos de esta área es que exigían resultados, establecían directrices y políticas sin consultar al equipo. Cuando el equipo expresaba su opinión, preocupación o reclamo por tomar dicha decisión, no los escuchaban y los tachaban de "rebeldes" y de no querer "hacer su trabajo".

También sucedió que por los problemas que se presentaban, juzgaban a algunos tester´s como de "bajo nivel", lo cual estaba muy alejado de la realidad, pues algunos de ellos eran los que más apoyaban al resto del equipo, los que trabajaban con más velocidad, tenían más experiencia o simplemente eran capaces de encontrar defectos "letales" diseñando su ejecución de pruebas de acuerdo a su experiencia.

Culpaban a los tester´s de incompetencia, pero el problema raíz eran los procesos no actualizados. Era "urgente" actualizar los procesos y reconstruir las estrategias de comunicación, en conjunto con las otras áreas.

La situación de la falta de confianza en el equipo de trabajo, no era exclusiva de QA, este escenario se replicaba en las otras áreas con las que interactuábamos, lo cual detonaba un enfoque incorrecto por parte de la dirección, que se esforzaba mucho en resolver consecuencias, pero no en analizar la raíz de los problemas.

Las consecuencias de no resolver el problema raíz y no manejar una política de confianza, orientada a mejorar el trabajo en equipo, tuvo como consecuencia que muchos tester´s, desarrolladores, arquitectos, líderes y demás personal involucrado, perdiera la motivación, se concentrara sólo en "cumplir con sus obligaciones" y en el peor de los casos, presentara su renuncia.

Capítulo 3. El proceso de evolución a "Tester Leader"

Después de mi segundo año como Tester Jr, ya había atendido requerimientos de más de 7 sistemas diferentes. Fue entonces cuando se me brindó la oportunidad de ser Tester Leader Backup.

Debido a la carga de trabajo fue necesaria una reestructuración del área, en esa reestructuración se dividió a los tester´s en 3 equipos, ya existían dos "Tester Leader" y la dirección de QA decidió brindarle la oportunidad a un tester que ya tenía muchos años de experiencia en el área, así que lo ascendieron como "Tester Leader". Fue en este equipo donde se me asignó como "Tester Leader Backup".

Nuestro equipo estaba conformado por 5 personas incluyendo al Tester Leader. Éramos el equipo más pequeño del área, enfrentábamos los mismos problemas que los otros equipos, nos habían asignado sistemas pertenecientes a equipos conflictivos y estábamos a cargo de un líder nuevo. Era necesario comenzar a ganar una "buena reputación como equipo".

Desde que comenzamos a trabajar en conjunto, el líder nos había solicitado el apoyo para lograr que el equipo obtuviera buenos resultados, además ofreció su apoyo incondicional y abrió la puerta para comentar cualquier duda, sugerencia, solicitud de ayuda o problema que nos aquejara.

1.13. Un equipo pequeño con buena reputación

Sobre la marcha fuimos aprendiendo a trabajar en equipo, seguíamos enfrentándonos a los problemas que aquejaban al área, así que comenzamos en trabajar en soluciones para los mismos. Las estrategias que definiré a continuación se establecieron durante las juntas internas que manteníamos, explicaré la forma en que se aplicaron y los resultados obtenidos a partir de ellas:

- **Comunicación interna basada en la confianza.**

Esta estrategia se asumió desde que comenzamos a trabajar juntos, de hecho esta política es la que provocó que estableciéramos soluciones en conjunto. Sin embargo esto no habría sucedido de no ser porque se cumplieron los siguientes factores:

  - **Apoyo incondicional por parte del líder y promoción del equipo.**

    Aún en situaciones donde cometíamos errores, el líder asumía la responsabilidad completa ante los jefes y las otras áreas. Después comentaba los errores con el o los tester´s responsables y nos proponía soluciones para prevenir dichos errores.

    Del mismo modo si algún miembro del equipo obtenía un logro significativo, el líder se encargaba de promover los logros del equipo hacia los directivos.

- **Retroalimentación.**
  Así como el líder nos brindaba retroalimentación cuando cometíamos algún error, también se mostraba abierto a observaciones, lo cual nos permitió lograr un nivel de responsabilidad mucho más alto.
- **Proactividad y compromiso entre los miembros del equipo.**
  La actitud de los miembros del equipo era proactiva, es decir, si alguno de los tester´s tenía "carga excesiva de trabajo", los tester´s que tenían poca carga o tiempos muertos ofrecían su apoyo para atacar el problema.

- **Definir objetivos por equipo.**
  Para obtener una buena reputación era necesario aumentar la calidad del servicio que ofrecíamos, así que se establecieron como objetivos internos:
  - **Unificar la forma de trabajo.**
    Con esto garantizábamos que los miembros del equipo trabajaban con el mismo nivel de calidad además de que obligábamos a que los equipos de desarrollo que atendíamos tuvieran menos elementos, para causar conflictos.

  - **Aumentar la calidad de nuestros entregables.**
    A raíz de las revisiones de SQA el área tenía varios NCI´s asignados, lo cual detonaba una baja calidad en los entregables de QA. Como consecuencia, obtuvimos una mala imagen, pues si tomamos en cuenta que QA es el área encargada de "asegurar la calidad", tener NCI´s asignados a nuestra área empeoraba nuestra reputación. Así que nuestro objetivo era disminuir a 0 los NCI´s de nuestros requerimientos.

  - **Disminuir a 0 la cantidad de rechazos de "Copia de instancias".**
    Lo que significaba aumentar la calidad de los entregables y evidencias de nuestros requerimientos. Al mismo tiempo acelerar el proceso de "copia de instancias".

  - **Disminuir a 0 los defectos en producción.**
    Parte de los defectos detectados en producción surgían debido al dimensionamiento incorrecto de las pruebas. Las pruebas funcionales se ejecutaban en base al Script de pruebas de la documentación del requerimiento, el problema era que algunos equipos de desarrollo, debido al exceso de carga de trabajo, sólo colocaban las pruebas "mínimas".
    Nuestra misión en este aspecto era aprovechar las juntas de arquitectura para obtener la mayor cantidad de información sobre el alcance del requerimiento y

solicitar las pruebas que consideráramos pertinentes a la documentación del requerimiento.

En base a estos objetivos nos evaluábamos constantemente, lo cual nos llevó a elevar considerablemente la calidad de nuestros entregables.
Nos evaluábamos tomando como indicadores:

- Reportes SQA.
- Respuesta de Change Management durante la copia de instancias.
- BFE´s presentados en producción.
- Pizarrón SCRUM.

Pizarrón SCRUM.

El pizarrón SCRUM lo adoptamos para que todos (especialmente el líder), pudieran identificar visualmente el estatus de determinados requerimientos. En él se registraban:

- ID Sistema.
- ID Requerimiento.
- Fecha de entrega.
- Etapas ejecutadas.
- Etapas terminadas.

A continuación coloco un ejemplo:

Tabla 4. Ejemplo de Pizarrón SCRUM (Elaboración propia, 2014).

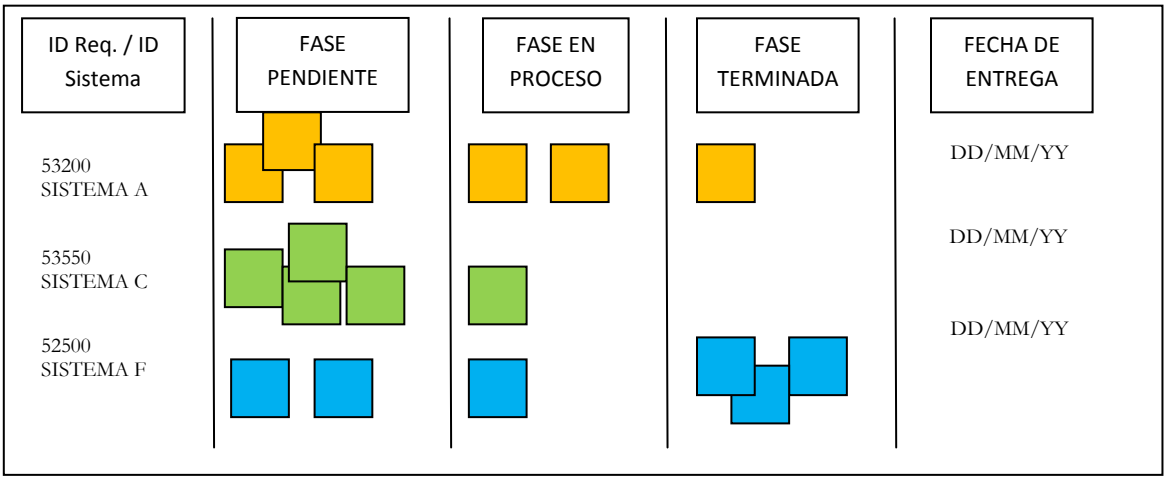
ID Requerimiento (ID Sistema)	Fase Pendiente	Fase en Proceso	Fase Terminada	Fecha de entrega programada.
53200 SISTEMA A	• Pruebas funcionales. • Pruebas de regresión. • Pruebas UAT. • Proceso de Cierre.	• Pruebas de código Defecto "A56"	• Criterios de aceptación.	10 de Marzo de 2014

Esta tabla se llevaba en el pizarrón de la oficina, las fases las movíamos de columna gracias a la utilización de "post-it". Nos pusimos la regla de que de cada requerimiento debía haber un avance de al menos una fase al día, o en su defecto debía anotarse el número de defectos registrados.

De ese modo, todos estábamos enterados del status de los defectos y podíamos informar si algún directivo nos solicitaba la información. Además de que si alguna fase permanecía estática

por mucho tiempo, el líder podía identificarla y averiguar junto con el tester, el problema y la forma de solucionarlo. A continuación se agrega un ejemplo gráfico de lo que es el pizarrón SCRUM.

Tabla 5. Ejemplo de pizarrón SCRUM con post-it (Elaboración propia, 2014).



El resultado fue convertirnos en el equipo con mejor reputación del área. Esto no sólo era visto por los directivos de QA, si no por las otras áreas como Desarrollo, Arquitectura y Change Management.

1.14. El cambio de mentalidad hacia Tester Leader

Paralelo a la evolución de nuestro equipo, el líder iba capacitándome sobre las actividades que se debían realizar como líder. Pueden englobarse en tres tipos de actividades:

Actividades administrativas

Corresponden a todas las actividades relacionadas con el manejo de personal, por ejemplo:

- Solicitar semanalmente horas consumidas durante la atención de los requerimientos, a los PM´s de desarrollo, por parte del equipo de QA, tanto tester´s como los administradores del Ambiente. Para realizar las solicitudes era necesario enviar a los PM´s:
    - ID Sistema.
    - ID Requerimiento.
    - Nombre y número de empleado del personal asignado (Tester o administrador de ambiente).
    - Descripción de la Actividad.
    - Tiempo consumido en dicha actividad (En horas).

  Los PM´s debían confirmar cuando las horas ya estaban registradas en la herramienta.

- Resolver problemas relacionados con el registro de horas en caso de que los PM´s no estuvieran de acuerdo con las horas solicitadas.

- Gestionar el tema de vacaciones de los miembros del equipo, para garantizar que las pudieran tomar sin afectar el nivel de atención de los requerimientos.

- Organizar los viernes flexibles del equipo, para garantizar que pudieran efectuarse sin afectar la operación. El viernes flexible consistía en que cada viernes, un miembro del equipo podía ingresar a las 7:00 a.m. y salir a las 3:00 p.m. siempre y cuando cumpliera con todas sus actividades.

- Atender las solicitudes de los directivos del área correspondientes a manejo de personal.

- Planeación de los requerimientos. Semanalmente se realizaba una junta en la que se revisaba el archivo de planeación, en este archivo los líderes de desarrollo y QA registraban el avance de los requerimientos, desde su nacimiento hasta su entrega. Este archivo contenía los siguientes datos:
    - Estatus Requerimiento
    - Id Sistema
    - Nombre de la solicitud
    - Id Requerimiento
    - PM
    - Tipo y número de objetos
    - Estatus de la entrega
    - Flujos dependientes
    - Menú de datos (Datos necesarios para las pruebas).
    - Comentarios Desarrollo.
    - Tipo de Requerimiento (NS, BC).
    - Fecha Planeada Entrega QA (Fecha planeada de entrega por parte de desarrollo).
    - Fecha Real Entrega QA (Fecha real de entrega por parte de desarrollo).
    - Reprogramación Entrega QA.
    - Holgura para atención de defectos. (Tiempo aproximado en días).
    - Pruebas UAT (Tiempo aproximado en días para pruebas UAT).
    - Fecha Planeada de Inicio QA.
    - Fecha Real Inicio QA.
    - Fecha Planeada de Fin QA.

- Fecha Real Fin QA.
- Fecha Planeada Liberación.
- Fecha Real de Liberación.

Comunicación efectiva y toma de decisiones

Era de suma importancia garantizar la integración del equipo, por ello, debían realizarse actividades para mantenerlo enfocado a resultados y unido. Entre las actividades que realizábamos estaban:

- Juntas de evaluación interna, donde se revisaban temas como:
  - Objetivos alcanzados.
  - Problemas presentados.
  - Estrategias de solución.
  - Retroalimentación del equipo.
  - Avisos por parte del líder.
- Actividades de integración:
  - Capacitación constante. Esta se manejaba a dos niveles, por un lado, el líder logró que EBOLSA tomara en cuenta a los miembros del equipo en los programas de capacitación y por otro lado, compartíamos constantemente conocimientos, de tal suerte que cuando un tester faltaba, el resto del equipo era capaz de cubrir sus actividades.
  - Además de las juntas, se procuraba que comiéramos juntos una vez a la semana, aprovechábamos este tipo de reuniones para divertirnos y saber cómo se sentían cada miembro del equipo.
- Toma de decisiones y negociación: Los equipos conflictivos seguían ocasionando problemas, lo cual retrasaba nuestros tiempos de entrega, era necesario extender las relaciones de "confianza", eso lo logró el Tester Leader por medio de una "Negociación efectiva". Esta básicamente consistía en conocer las necesidades de las otras áreas con las que interactuábamos y proponer acuerdos para llegar a soluciones en conjunto. Cabe señalar que estas negociaciones iban respaldadas por la reputación que el equipo había adquirido.

En un periodo menor a un año, el equipo obtuvo la calificación más alta en las revisiones SQA, cumplió con todos los lineamientos establecidos por la dirección, estableció la premisa de que era posible que todo el equipo trabajara de la misma manera, estableció relaciones de confianza con casi todos los equipos de desarrollo con los que interactuaba y fue reconocido por Change Management como el equipo que entregaba con mayor calidad durante el proceso de "copia de instancias".

Estos logros no habrían sido posibles sin la participación del equipo, pero la participación del equipo no se habría dado si no se hubiera establecido una comunicación basada en la confianza. El Tester Leader dio el primer paso, mostrándole al equipo que confiaba en su trabajo y ofreció su apoyo incondicional. Pero además reafirmó su compromiso, al promover y defender a su equipo, ante los directivos de todas las áreas.

1.15. Conclusiones

La empresa estaba atravesando por una crisis, en la que la carga de trabajo era muy alta, los procesos de calidad se habían desplazado por la urgencia de atender la operación del día a día. Y esto había causado que la operación evolucionara más rápido que los procesos del área.

En medio de un ambiente de crisis, nos enfrentábamos a personal sin motivación, directivos manejando una política de error-castigo, solicitudes de aplicación de acciones correctivas constantes y carga de trabajo excesiva. Pero aún con toda esa confusión, el equipo más pequeño demostró que si se podía aumentar la calidad de las entregas estableciendo políticas basadas en la confianza.

Establecimos y cumplimos nuestros propios objetivos como equipo, alineados a los objetivos de la empresa. Establecimos la premisa de que "si se pueden encontrar soluciones en medio del caos". Si esto logró un equipo de 5 personas en menos de un año, imaginen lo que se lograría si toda la organización basara sus políticas en relaciones de confianza. No sería fácil, pero creo que bien valdría la pena intentarlo.

Conclusiones

¿Qué es un ingeniero?, específicamente ¿Qué es un ingeniero en computación? De acuerdo al perfil de egreso que define nuestra escuela:

"El Ingeniero en Computación, es un profesional especializado en la aplicación de Tecnologías de la Información y capaz de generar y transformar sistemas relacionados con las telecomunicaciones y la computación, **proporcionando a su entorno soluciones innovadoras en beneficio de las personas e instituciones que lo requieran**. Su ejercicio profesional es con el **compromiso de desempeñarse con altos estándares de calidad y siempre acorde al código de ética que implica la profesión** (UNAM, 2013)**.** "

Además de que, de acuerdo al punto 12 de los "Principios Generales Relativos a la Docencia" del "Marco Institucional de Docencia" de la UNAM, dice que:

"Al mismo tiempo ha de advertirles del **compromiso** que asumirán, como egresados, de aplicar los conocimientos adquiridos en **bien del país**, contribuyendo a su transformación positiva y prevaleciendo el interés general sobre el individual (UNAM, 2003)"

Mi objetivo entonces es: En base a mis conocimientos y experiencia, "Buscar y aplicar soluciones para el bien del país". Lo que quiere decir que ya sea como consultor o como tester, mi misión siempre fue apoyar a las empresas para las que ofrecía mis servicios ya sea como empleada o como proveedor. Contribuyendo así a mejorar sus organizaciones y con ello apoyar el crecimiento del país.

Sin embargo, no es una tarea fácil, pues como todo en la vida, también se atraviesa por un proceso de "ensayo-error", mismo que contribuye a la adquisición de las diferentes competencias que el ingeniero aplica para cumplir esa misión.

En medio de la búsqueda por cumplir mi misión como ingeniera, pude comprobar que nada permanece "estático" por mucho tiempo. Ninguna empresa puede darse el "lujo" de descuidar sus procesos de aseguramiento de la calidad, sobre todo si los objetivos de la misma son llegar a ser, o mantenerse como una empresa competitiva.

Recapitulando:

- Empresa Pyme. La empresa que solicitó el OpenbravoERP, que sólo estaba dispuesta a invertir lo "mínimo" para la instalación de la herramienta a pesar de las advertencias de parte de ESOFTWARE; y que además no había considerado los equipos de cómputo para los empleados que utilizarían el ERP.

- Organización gubernamental. Aquella que había pagado por la implementación de Karomi BPM que les ayudaría a automatizar sus principales procesos de atención al cliente, pero no habían tomado en cuenta que las unidades de bacheo, no contaban con la infraestructura ni el conocimiento necesario para utilizar la herramienta.
- Empresa Pyme. ESOFTWARE donde omitimos el dato exacto de los requisitos mínimos del equipo; además de las omisiones que bien pudieron identificarse en la etapa de análisis.
- Empresa Grande. EBOLSA una empresa muy grande e importante, que atravesaba por una crisis debido a las cargas excesivas de trabajo, aunadas a la evolución inminente de sus diferentes áreas y el descuido de sus procesos de aseguramiento de la calidad.

Ningún tipo de empresa está exenta de problemas, porque todas las empresas están en constante evolución. Por eso es importante establecer estrategias que permitan a la organización aprovechar sus momentos de crisis, aprender de ellas y establecer procedimientos de mejora continua que les permita evolucionar junto con sus procesos.

Aún con problemáticas asfixiantes, en medio del caos, siempre existe la manera de comenzar un cambio. Por ejemplo, el cambio que logró aquel equipo de 5 personas, quienes al entender, que antes de "exigir" a las diferentes áreas, debían incrementar la calidad de su servicio. Lograron reconstruir los canales de comunicación con dichas áreas y encontraron un canal para satisfacer las necesidades de sus clientes a pesar de que los procesos seguían sin actualizarse.

Puedo concluir entonces que la organización debe poner especial atención a la administración de sus recursos intelectuales, estos representan la mejor inversión que la empresa puede hacer. Dentro de la administración de los recursos intelectuales, los temas vitales son:

- Liderazgo orientado al servicio.
- Capacitación.
- Estrategias de comunicación.
- Políticas basadas en la confianza.
- Objetivos claros.
- Indicadores de evaluación y recompensas para los empleados.

Me gustaría poder decir que gracias a los resultados que se obtuvieron con ese pequeño equipo, el líder recibió la promoción prometida (No tenía ni el puesto, ni el sueldo, pues estaba "a prueba"), que el equipo obtuvo mayor reconocimiento y que además se establecieron esas estrategias en los otros dos equipos, sin embargo no fue así.

Después de un año con buenos resultados, EBOLSA comenzó con un recorte de personal, en el cual despidieron a una de las líderes de QA. Esa líder tenía muchísimos años trabajando en EBOLSA, tenía conocimiento de todos los procesos (incluyendo los de desarrollo), la conocían todos los PM´s y equipos de desarrollo, conocía prácticamente todos los sistemas. La salida de esta líder fue un golpe muy fuerte para el área, porque el equipo al que lideraba era el que tenía a su cargo el testeo de los sistemas más críticos de EBOLSA. La justificación de su salida, fue que "no había logrado los resultados esperados", no se sabe en qué se basaron, pues acababan de liberar un proyecto que involucraba a todos los sistemas críticos y según cifras de la dirección "había sido un éxito".

´Durante la última "retroalimentación" con los jefes, se externó la preocupación tanto por los procesos no actualizados, como por la falta de indicadores que nos permitieran medir el desempeño real de los tester´s, el jefe del área "se comprometió" tomar en cuenta los comentarios y actualizar los indicadores. Con esa base, cuando despidieron a esta persona, yo le pedí que nos explicara en base a qué resultados había tomado dicha decisión. Además pregunté si los indicadores utilizados se habían actualizado "como lo había prometido", porque no quedaba claro, el ¿Por qué no se había hecho el esfuerzo por que se quedara un elemento tan crítico para el área? No hubo una respuesta real, sólo se dijo que "no se podía hacer nada".

Al mismo tiempo me designaron para comenzar con la actualización de los procesos del área, debido a que yo había mostrado gran interés en el tema. Sin embargo, me estaban pidiendo una reingeniería de los procesos de QA, en tres meses, pues se supone que solo consistiría en apegarse a los procesos globales de EBOLSA a nivel Latinoamérica.

A partir de ese día, se le solicitó al que era mi líder que llevara las actividades de los dos equipos, por lo que nuestra carga administrativa subió considerablemente. Yo tenía prohibido apoyar a mi líder pues mi prioridad era la actualización de los procesos, sin embargo, la carga era muy alta así que seguía apoyándolo en algunas actividades. Nuestro pequeño equipo continuaba con un buen ritmo y calidad de trabajo, pero un mes después, la dirección tomó otra decisión.

Debido a que mi líder había hecho un buen trabajo, lo asignaron al equipo que se había quedado sin líder. Al mismo tiempo ascendieron a una tester, como líder de nuestro pequeño equipo. Hubo un gran desconcierto por parte de los equipos de desarrollo que atendíamos, pues la nueva líder, no contaba con la experiencia para negociar, por lo que se enfrentó con la rivalidad natural entre desarrolladores y tester´s. Y es que durante una negociación, deben evitarse los "no rotundos", por que proyectan una falta de disposición para resolver un problema, si en cambio, se responde "veo complicaciones, pero déjame revisarlo", "lo tomo en cuenta, pero me gustaría que consideraras lo siguiente", "podríamos intentarlo, pero ten en

cuenta este riesgo", etc. Entonces se abre una puerta a la negociación, porque se demuestra la intención por resolver el problema, sin comprometerse a objetivos inalcanzables.

Sin embargo, la nueva en lugar de asumir responsabilidades en conjunto, comenzó a "acusar a cada tester" para dejar en evidencia los "errores u omisiones". Había comenzado a implementar una comunicación basada en la desconfianza.

En medio de todos esos cambios, mi líder decidió irse de la empresa, pues a pesar de los buenos resultados no le habían ofrecido ni el puesto, ni el sueldo prometidos. Además de que se supo que a otra persona que había sido contratada como Tester Sr unos meses atrás, se le ofreció un contrato permanente con un sueldo al nivel de la jefa del área, mientras que a mi líder, no le ofrecían ningún aumento por "falta de presupuesto".

Con la salida de mi líder, me llega la oferta de tomar el liderazgo de mi equipo, pero sin aumento en sueldo ni en puesto, tal como se lo habían ofrecido a mi líder anterior, estaría "a prueba", cabe señalar que la otra líder que fue ascendida, si obtuvo el puesto y el sueldo. Yo decidí aceptar, con el objetivo de apoyar a mi equipo "temporalmente". Unas semanas después, recibo una propuesta en otro proyecto y decido dejar esa empresa.

A pesar de haber dado buenos resultados, estos no valieron para que la dirección finalmente ofreciera el puesto y el sueldo acorde a las responsabilidades que nos habían sido asignadas. Las personas a las que se les aumentó el sueldo y se les ofreció el puesto, no eran precisamente las que tenían mayores capacidades, o las mejores calificaciones en los indicadores.

Esta situación puede suscitarse en cualquier área, en cualquier empresa o negocio. Y es deber de un ingeniero que este tipo de situaciones no "merme" la calidad de su trabajo, ni le obligue a seguir reglas sin fundamentos, sobre todo, si estas representan un problema a futuro para la organización, o nos colocan en medio de una decisión sin ética, porque fuimos preparados para resolver problemas con "ética" en base a nuestra experiencia y conocimientos, porque la calidad e integridad de nuestro trabajo, es la que permanecerá aún cuando el proyecto termine, quedará plasmada en los resultados logrados y sobre todo en los clientes; como la "evidencia del trabajo de un buen ingeniero".

Es importante mencionar que el ingeniero también debe reconocer su valía, debe tener dignidad, para poner un límite si no se ha reconocido su trabajo, confiar en sí mismo y arriesgarse a tomar nuevas oportunidades y buscar nuevos horizontes.

De aquel equipo de cinco personas, hoy en día sólo quedan dos en EBOLSA. Un detalle importante a mencionar, es que hoy en día los PM´s de los equipos de desarrollo, prefieren negociar con esos dos "Tester Jr", que con las líderes que manejan los equipos del área de QA.

En cuanto a la actualización de los procesos, actualmente se encuentran en medio de una migración a los procesos globales de EBOLSA a nivel Latinoamérica, misma que advertí al jefe del área que no sería tan sencilla, pues los procesos globales en los cuales debía basarse el nuevo proceso, describían ¿Qué hacer?, mas no el ¿Cómo?. Aún así lanzaron la migración por orden de los directivos, decisión que ha introducido riesgos muy altos en el proceso de desarrollo-testing, pues debido a que el proceso, pasó de ser un bache a ser un "stopper", los equipos de desarrollo han comenzado a saltarse el proceso de pruebas, liberando directamente en producción.

Hoy en día, si llego a encontrarme con los PM´s de desarrollo o con los desarrolladores de los equipos a los que atendíamos. Me saludan con respeto, me cuentan los problemas a los que se enfrentan actualmente con QA y reconocen que no han recibido la calidad de servicio que les ofrecíamos en aquel entonces. Lo cual me deja ver que quedo evidencia de la calidad de mi trabajo; que a pesar de que los directivos no la reconocieran; nuestros clientes si la reconocían, la reconocen, la extrañan y la reclaman a la actual administración.

Es importante que además de los conocimientos técnicos, los ingenieros egresados de la UNAM, adquieran conocimientos de administración de proyectos y habilidades directivas. Pues de acuerdo a mi experiencia los ingenieros egresados de la UNAM tienen enormes capacidades de crecimiento en el ambiente laboral, por lo que con esos conocimientos, podrían aumentar sus oportunidades de crecimiento y por lo tanto acceder a mejores puestos y sueldos.

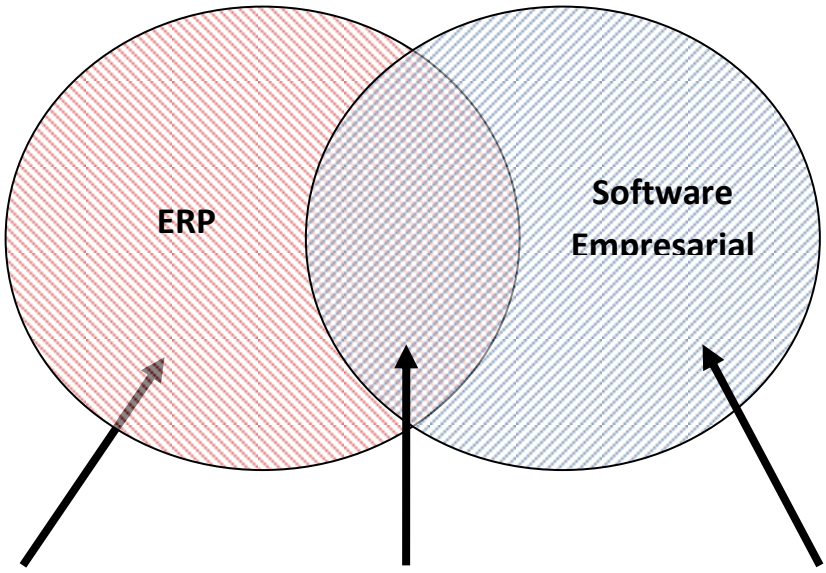
La frase que utilicé constantemente con mi equipo, en medio de todos esos cambios era: "Piensa que tu trabajo es tan bueno, que cuando te vayas, te van a extrañar".

Creo fervientemente, que esa debe ser la actitud de todo ingeniero: resolver problemas con ética y compromiso, manteniendo la calidad de su trabajo, evolucionando constantemente, aprovechando los momentos de crisis, porque es justo en medio de situaciones caóticas, donde se obtienen más conocimientos, son las que les brindarán las herramientas para afrontar los mayores retos y obtener esas satisfacciones que nos califican como "Ingenieros".

Apéndice A: Conceptos de sistemas empresariales

A continuación se complementa la definición de algunos conceptos utilizados en el capítulo 1.

- **ERP**: Enterprise Resource Planning o Planificación de Recursos Empresariales. El ERP debe soportar todas las áreas de la empresa, además de brindar herramientas para la planificación y toma de decisiones, de ahí que (Wallace & Kremzar, 2001) aseguren que el ERP no es un Software. Si bien es cierto que algunos sistemas empresariales contienen gran parte de las funciones del ERP, los procesos de negocio que el ERP contiene, no se encuentran al 100% en el software empresarial.



Procesos ERP que no forman parte de un típico Software Empresarial:

- Previsión de ventas
- Ventas y planeación de operaciones
- Sistemas avanzados de planificación
- Sistemas de calificación de proveedores
- Métricas de

Procesos ERP que están incluidos en un típico Software Empresarial:

- Programación de la producción
- Capacidad de planificación
- Planificación de requerimientos de materiales (MRP)
- Planeación de distribución de requerimientos
- Pedidos del cliente y apartados

Procesos de software empresarial, no incluidos en el ERP:

- Cuentas por pagar
- Cuentas por cobrar
- Libro mayor
- Gestión de efectivo
- Gestión de relaciones con el cliente (CRM por sus siglas en ingles "Customer Relation Ship"

Ilustración V. El ERP (Wallace & Kremzar, 2001).

Actualmente existen diversos software ERP en el mercado con diferentes características, sin embargo de acuerdo a mi experiencia, para que un software pueda considerarse como Software ERP es necesario que posea:

- Integración: La información de todos los módulos debe estar integrada en una única base de datos.
- Modularidad: Los módulos pueden integrarse o separarse sin que ello afecte el funcionamiento de los otros módulos.
- Flexibilidad: El sistema debe tener la capacidad para adecuarse a los cambios de la empresa.
- Mejores prácticas: Debe contener las mejores prácticas en los procesos de negocio.
- Debe cubrir la mayoría de los procesos de negocio de una empresa (Por ejemplo: Compras, ventas, contabilidad, inventarios).
- Debe manejar y devolver la información tanto en unidades como en datos financieros.

De acuerdo a lo anterior se puede definir el Software ERP como un **"Software integrado, modular y flexible, que soporta la mayor parte de las áreas funcionales de una empresa, para proveer la información de la situación de la empresa, a los tomadores de decisiones".**

- **BPM**: Business Process Management (BPM) o Sistema de administración de procesos, es la disciplina empresarial cuyo objetivo es mejorar la eficiencia de una organización por medio de la gestión sistemática de sus procesos de negocio. Su objetivo es mejorar el performance de cualquier organización en base al continuo mejoramiento de sus procesos de negocio.
  Por ejemplo, parte de los procesos de negocio de una empresa es la gestión del personal, así que si dicha empresa tuviera esos procesos en un sistema BPM, los empleados de la misma podrían solicitar sus vacaciones por medio del sistema, de tal suerte que la persona encargada de aprobar dichas vacaciones, sólo tendría que entrar al sistema para aprobar o rechazar la solicitud de vacaciones; una vez que se atendió la solicitud, el empleado recibiría en automático la respuesta a su solicitud de vacaciones.

- **CRM**: Customer Relationship Management, se entiende como la Gestión sobre la Relación con los Consumidores, es decir se refiere a una estrategia de negocios centrada en el cliente. Es decir, con este software, la empresa puede llevar el control de los vendedores y sus clientes.
  Los vendedores de dicha empresa la utilizarían como herramienta de seguimiento para cada cliente, desde que se establece el primer contacto, las citas planeadas, las llamadas,

los comentarios y solicitudes específicas del cliente, hasta la contratación. Ofreciendo así a los directivos un panorama completo de los resultados de sus vendedores, la popularidad de sus productos e incluso los clientes recurrentes.

- **POS**: Point of Sale, terminal de punto de venta.
  Permite llevar el control de ventas de cualquier punto de venta, como es las ventas, los productos más vendidos, inventario, formas de pago, registro de ingresos, corte de caja y empleados.

- **Open Source**: Se considera Open Source al software que además de garantizar el acceso a su código, cumple con los siguientes criterios(Open Source Initiative):
  1. Libre Redistribución.
  2. El programa debe incluir su código fuente, también al redistribuirse.
  3. Subproductos. La licencia debe permitir modificaciones y sub-productos bajo el mismo licenciamiento.
  4. Derechos de autor del código fuente.
  5. No discriminación de personas o grupos. Ni restricciones de uso.
  6. Licencia permanente y extensible.
  7. Software amigable con otros software´s.

  Software neutral, sin interfaces o estilos específicos o tecnológicos.

Apéndice B: Software utilizado en "Procesos de Calidad del Software"

A continuación se agrega la descripción de los software mencionados en el capítulo 2 y 3.

- **Share Point**: Las organizaciones usan SharePoint para crear sitios web. Puede ser utilizado como un lugar seguro para almacenar, organizar, compartir y acceder a información desde casi cualquier dispositivo (Microsoft, 2014).

- **IBM Rational AppScan**: Herramienta de prueba de seguridad de aplicaciones Web que automatiza las evaluaciones de vulnerabilidad y escanea y prueba todas las vulnerabilidades comunes de aplicaciones Web incluyendo inyección SQL, scripting entre sitios, desbordamiento de buffer y nuevos escaneos de exposición Web 2.0 y aplicaciones flash/Flex (IBM).

- **Quality Center HP**: Gestiona y automatiza la entrega de aplicaciones seguras, fiables y de alta calidad. Puede implementar su gestión de calidad completa y establecer procesos consistentes y repetibles para gestionar requisitos, pruebas y componentes empresariales con HP (Hewlett-Packard Development CompanyHP, 2014).

- **Team Foundation Server**: Visual Studio Team Foundation Server 2012 (TFS) es la plataforma de colaboración en el núcleo de la solución de administración del ciclo de vida de las aplicaciones (ALM) de Microsoft. TFS admite prácticas ágiles de desarrollo, varios IDE (Entorno de desarrollo integrado) y plataformas de manera local o en la nube y le proporciona las herramientas que necesita para administrar de manera eficaz los proyectos de desarrollo de software a lo largo del ciclo de TI (Microsoft, 2014).

Bibliografía

- Hewlett-Packard Development CompanyHP (2014). Quality Center Software. Consultada el 12 de Febrero de 2014, Disponible en: http://www8.hp.com/mx/es/software-solutions/software.html?compURI=1172141

- IBM. Prueba: IBM Rational AppScan. Consultada el 2 de Febrero de 2014, Disponible en: http://www.ibm.com/developerworks/ssa/downloads/r/appscan

- Microsoft (2014). Introducción a SharePoint. Consultada el 12 de Febrero de 2014, Disponible en: http://office.microsoft.com/es-mx/sharepoint-server-help/introduccion-a-sharepoint-HA102772778.aspx?CTT=5&origin=HA010378184

- Microsoft (2014). Team Foundation Server. Consultada en línea el 1 de Febrero de 2014, Disponible en: http://msdn.microsoft.com/es-es/vstudio/ff637362.aspx>.

- Microsoft Developer Network. Información general sobre Team Foundation. Consultada el 18 de Enero de 2014, Disponible en: http://msdn.microsoft.com/es-es/library/ms242904(v=vs.90).aspx

- Open Source Initiative. The Open Source Definition. Consultada el 11 de Enero de 2014, disponible en: http://opensource.org/docs/osd

- Openbravo (2007-2014). Licencia Comercial de Openbravo, Consultada el 05 de Enero de 2014, disponible en: http://www.openbravo.com/es/content/licencia-comercial-de-openbravo

- Openbravo (2014). Plataforma ERP de Openbravo. Consultada en línea el 01 de Abril de 2014, Disponible en: http://www.openbravo.com/sites/default/files/Openbravo_ERP_Platform_brochure_LOW_SP_March2014.pdf

- Openbravo (2007-2014). License / Business Model. Consultada el 4 de Enero de 2014, disponible en: http://www.openbravo.com/license-business-model

- Secretaría de economía (2013, Noviembre). Reglas de operación del fondo Pyme. Consultada el 15 de Enero de 2014, disponible en: http://www.dof.gob.mx/nota_detalle.php?codigo=5323150&fecha=25/11/2013

- Secretaría de Economía. Pymes, eslabón fundamental para el crecimiento en México. Consultada el 11 de Enero de 2014, disponible en: http://www.promexico.gob.mx/negocios-internacionales/pymes-eslabon-fundamental-para-el-crecimiento-en-mexico.html

- The CentOS Project (2014). CentOS Linux. Consultada el 12 de Abril de 2014, Disponible en: http://www.centos.org/about/

- Thomas F. Wallace & Michael H. Kremzar (2001). ERP: Making it Happen. Canadá, John Wiley & Sons, Inc.

- UNAM (2003, Octubre). Marco institucional de docencia. Consultada el 05 de Marzo de 2014, Disponible en: http://www.ingenieria.unam.mx/~centrodedocencia/induccionueingre/Documentos/Marco_institucional.pdf

- UNAM (2013, Junio). Descripción sintética del plan de estudios. Licenciatura de ingeniería en computación. Consultada el 05 de Marzo de 2014, Disponible en: https://www.dgae.unam.mx/planes/aragon/Ing-comp_aragon.pdf