Universidad Autónoma de Querétaro
Facultad de Contaduría y Administración
Maestría en Administración

FACTORES E INDICADORES RELACIONADOS CON LA PRODUCTIVIDAD DE UNA EMPRESA DEDICADA AL DESARROLLO DE SOFTWARE

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Administración

Presenta:
Edgar Hernández Meraz

Dirigido por:
Dra. Ma. Luisa Leal García

SINODALES

Dra. Ma. Luisa Leal García Presidente	Firma
M. en A. José Antonio Robles Hernández Secretario	Firma
M. en C.Lorena Alcocer Gamba Vocal	Firma
Dra. Amalia Rico Hernández Suplente	Firma
Dr. Fernando Barragán Naranjo Suplente	Firma

M. en I. Héctor Fernando Valencia Pérez
Director de la Facultad de Contaduría y Administración

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Junio, 2009
México

RESUMEN

El siguiente estudio fue realizado para buscar los factores o indicadores que ayuden a medir la productividad de una empresa dedicada al desarrollo de software. Para lograr esto se analizaron las investigaciones de diversos autores y compañías que se desenvuelven en este medio. Los temas de estas investigaciones están relacionados con los factores, herramientas y métricas involucrados con la productividad en el desarrollo de software. Las fechas de los artículos consultados datan desde 1972 hasta el año 2007. En cuanto al diseño empleado en el presente estudio es no experimental, ya que no se posee control directo de las variables independientes, debido a que sus manifestaciones ya han ocurrido. El tipo de instrumento empleado en este estudio son formatos basados en la jerarquización de elementos. Estos formatos fueron llenados con una serie de factores y métricas relacionadas con la productividad del software así como también con algunos propuestos por integrantes de la empresa en estudio que de acuerdo a su experiencia consideraron serian de utilidad medir. Se hizo una primera selección de factores y métricas por un comité de la empresa encargado de determinar las métricas, metas y objetivos de la misma, posteriormente los factores y métricas que se consideraron más apegados a lo que la empresa necesitaba medir se sometieron a un análisis más exhaustivo para determinar cuales se tomarían para ser medidos en una primera etapa. Las primeras factores y métricas a ser medidos fueron: Problemas de usuario por mes, Calidad del defecto reparado y Métrica de seguimiento de valor de costo. Diversos autores determinaron categorías que se deben medir en lo que al software se refiere, estas son: Productos de software, Procesos de producción y estructuras de software y Configuraciones de producción de software. Dentro de estas categorías existen muchos factores y métricas, sin embargo, muchos de ellos solo pueden ser más informativos que útiles para la toma de decisiones importantes. Muchas veces estar midiendo tantos factores o indicadores puede causar sobre carga de trabajo, por lo que la productividad se puede ver afectada y entonces el objetivo de medir la productividad para ser más eficientes ya no se cumple. Por lo que es importante elegir de todos los factores e indicadores aquellos que puedan ser más útiles a las necesidades de las empresas.

(**Palabras clave:** software, productividad, métricas, empresa)

SUMMARY

The following research was made in order to look for the factors or indicators that help to measure the productivity of a software development company. To achieve this goal several researches of several authors and companies related with software development were analyzed. The topics of these researches are related with factors, tools and metrics involved with software development productivity. Dates of consulted articles are among 1972 to year 2007.The type of the design for this research is no experimental, because there is no direct control with the independent variables, due to they have already occurred. The type of instrument used is a form, based on item ranking. These forms were filled out with several factors and metrics related with software productivity as well as some other metrics proposed by some experienced people of the company under research which were considered useful to be measured. A first selection of metrics and factors was made by a company committee in charge of select metrics, goals and objectives for the company. Then after selecting those metrics considered more useful a second round was performed to analyze them in detail and finally choose those that were considered to be measured in a first stage. The first factors and metrics to be measured were: Monthly user problems, fix quality and cost/earned value tracking. Several authors have determined categories to be measured within software, these are: software product, production process and software structures, and software production configuration. Within these categories there are several factors and metrics; however some of them are more informative than useful for important decision making. Sometimes measuring so many factors or indicators can be cumbersome and cause work overload affecting productivity, and the goal of being more effective can not be achieved. Therefore is important to choose only factors and indicators that are more aligned with the company's needs.

(**Keywords:** software, productivity, metrics, company)

DEDICATORIAS

Con todo cariño para mis padres que siempre me han apoyado en todo lo que he emprendido a lo largo de mi vida y que su dedicación, buenos ejemplos y consejos me han ayudado a superarme cada día para lograr mis metas.

AGRADECIMIENTOS

Quiero agradecer a la Dra. María Luisa Leal García por el tiempo dedicado a asesorarme en el desarrollo de esta tesis aún fuera del periodo de asesorías. También quiero a agradecer a los supervisores de la empresa en estudio por su colaboración en la propuesta, selección y evaluación de factores y métricas relacionadas con la productividad del desarrollo de software.

ÍNDICE

	Página
RESUMEN	i
SUMMARY	ii
DEDICATORIAS	iii
AGRADECIMIENTOS	iv
ÍNDICE	v
ÍNDICE DE FIGURAS	vii
ÍNDICE DE TABLAS	viii
I. INTRODUCCIÓN	1
II. MARCO TEÓRICO	4
2.1 Conceptos	4
2.2 Antecedentes	9
2.3. Características de la empresa	10
2.3.1 Descripción	10
2.3.2 Historia	10
2.3.3 Organización	12
2.3.4 Misión	13
2.3.5 Visión	13
2.3.6 Principales enfoques	15
2.3.7 Productos y funciones principales	17
2.4 Investigaciones relacionadas	19
III. CARACTERÍSTICAS DE LA INVESTIGACIÓN	38
3.1 Justificación de la investigación	38
3.2. Planteamiento del problema	39
3.2.1 Preguntas de investigación	39
3.3 Definición de los objetivos	39
3.3.1 Objetivo general	39
3.3.2 Objetivos particulares	39
3.4 Definición del universo	40
3.5 Tamaño y tipo de la muestra	40
3.6 Identificación de variables	40
3.6.1 Definición operacional de las variables	40
3.7 Hipótesis	41

IV. METODOLOGIA 42
4.1 Diseño del estudio 42
4.2 Tipo de estudio 42
4.3 Instrumento 42
4.4 Procedimiento 42
4.5 Procesamiento 43

V. RESULTADOS, DISCUSION DEL TEMA Y PROPUESTAS 44
5.1 Características de la población 44
5.2 Resultados 44
5.3 Discusión del tema 48

CONCLUSIONES 52

BIBLIOGRAFIA 53

APÉNDICE 54

ÍNDICE DE FIGURAS

Figura — Página

1. Ciclo de vida típico del desarrollo de software.
   Fuente: Freeman, Peter et al, 1982, p. 46 — 6

2. Software Process Building Block.
   Fuente: Diseño de la empresa en investigación. — 8

3. Organigrama de la empresa.
   Fuente: Diseño propio con datos de la empresa en investigación. — 12

4. Ciclo de mejora continua.
   Fuente: Diseño de la empresa en investigación. — 17

5. Productos del departamento de diseño de componentes.
   Fuente: Diseño de la empresa en investigación. — 18

6. Productos del departamento de software.
   Fuente: Diseño de la empresa en investigación. — 18

7. Productos del área de infotainment.
   Fuente: Diseño de la empresa en investigación. — 19

8. Gráfica de resultados.
   Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores. — 45

9. Gráfica de frecuencias.
   Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores. — 47

ÍNDICE DE TABLAS

Tabla Página

1. Definiciones de métricas del sistema de software.
   Fuente: Smith Duncan Anne 1988, p .44 30

2. Definiciones de métrica del proceso de desarrollo de software.
   Fuente: Smith Duncan Anne 1988, p. 44 31

3. Resultados de 9 supervisores de distintas áreas.
   Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores. 44

4. Frecuencia de métricas para cada prioridad.
   Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores. 46

5. Descripción de métricas en términos de su importancia.
   Fuente: Diseño propio usando información del marco teórico. 49

I. INTRODUCCIÓN

La empresa de cual se esta llevando esta investigación es una empresa que trata de mejorar día con día para ser una empresa competitiva, ha estado trabajando duro para obtener las certificaciones de ISO-9001, TS-16749 para ser considerada como proveedor de la industria automotriz. La filosofía de mejora continúa esta siempre presente y es por eso que ahora se quiere seguir mejorando en todas las áreas de la empresa, una de las más importantes es el área de software.

Existen varios competidores a nivel mundial en esta área en países como India, China, Polonia, Alemania entre los principales. Entonces para ser la mejor opción para nuestros clientes es importante buscar superarse y detectar las áreas de oportunidad. Pero para mejorar, primero es necesario medir para saber en donde nos encontramos, en que estamos bien, en que estamos mal.

Dentro de la organización actualmente no existen métricas para medir la productividad en el área de desarrollo de software. Hasta hace poco se han hecho esfuerzos por tratar de buscar las variables que podrían ayudar a medir la productividad. Sin embargo, medir la productividad de software no es algo sencillo, pues no es como en el área de manufactura por ejemplo en donde se pueden contabilizar el número de piezas elaboradas ya sea por hora, por día, por semana, etc. Sin embargo para medir la productividad del área de software no es solo contabilizar el número de líneas producidas en un cierto periodo de tiempo, hay que considerar muchos factores que influyen entre ellos la experiencia, el número de errores introducidos (re trabajo), experiencia, complejidad del trabajo que se esta desarrollando, incluso tipo de equipo de trabajo utilizado (características de equipo de computo utilizado), etc.

Pueden ser muchos los factores pero lo importante o el principal enfoque de esta investigación es identificar cuales son los factores o indicadores más importantes que ayuden a medir la productividad del área de software en la empresa. Este es el objetivo

general de esta investigación. Sin embargo, otros temas de investigación que se pueden abrir con ayuda de esta son: medir la productividad en el área de pruebas de software y como mejorar la productividad partiendo de las primeras métricas tomadas.

Esta investigación es importante para la empresa y se justifica realizarla ya que al descubrir los factores o indicadores que ayuden a medir la productividad de manera efectiva serviría entre otras cosas para:

- Incrementar el volumen de trabajo con el mismo número de ingenieros
- Hacer el mismo trabajo con menor número de ingenieros
- Desarrollar productos más complejos o de mayor valor de mercado con el personal actual.
- Evitar contratar personal adicional para incrementar la carga de trabajo
- Reducir el número de errores en los productos que se entregan al cliente.
- Reducir el esfuerzo para rectificar errores
- Identificar defectos del producto en etapas tempranas del desarrollo de software.
- Identificar cuellos de botella
- Recursos no utilizados adecuadamente,
- Incrementar la calidad del software
- Desarrollar mejores productos a menor precio

Para proponer los factores o indicadores que ayuden a medir la productividad dentro de la empresa se tomaron las investigaciones de varios autores y empresas un extracto de estas investigaciones se puede ver en el marco teórico. Entonces se realizó una primera ronda de selección de los factores e indicadores propuestos por varios autores y empresas mencionados en el marco teórico. A los factores e indicadores se les dio el nombre de métricas. La selección fue realizada por el líder de equipo del comité de métricas de la empresa. De la primera selección se filtraron algunas métricas y se eligieron otras que podían ser más factibles de ser implementadas en la empresa de acuerdo a una clasificación de calidad, servicio y costo. Esta nueva selección se paso a

los supervisores de grupo para que eligieran de acuerdo a su criterio cuales de las métricas propuestas consideraban más factibles de ser implementadas en la empresa. Las métricas fueron explicadas y discutidas en una junta con el comité y los supervisores de área (también llamado grupo focal). Los resultados de la jerarquización fueron mostrados a los supervisores. De ahí se discutieron los resultados y se eligieron las métricas más útiles o factibles de ser adoptadas por la empresa.

Tres métricas fueron elegidas para una primera etapa, las cuales se van a estar monitoreando constantemente para ir haciendo ajustes y tomar acciones preventivas y correctivas para etapas subsecuentes del desarrollo, posteriormente se irán agregando más métricas para ser medidas y que ayuden a mejorar la productividad.

II. MARCO TEÓRICO

2.1 Conceptos

Actualmente muchas compañías dedicadas al desarrollo de software a nivel mundial están buscando como mejorar para ser más competitivos convertirse en mejores proveedores para sus clientes y ganarle a la competencia. Sin embargo para poder mejorar se tiene que tener una base contra la cual comparar. Para obtener esa base se tiene que hacer mediciones, cálculos e investigaciones. Uno de los puntos importantes para mejorar es medir la productividad del desarrollo de software.

Pero para entender más este tema primero tenemos que entender algunos conceptos básicos como lo es el de software, desarrollo de software y productividad.

Definiciones de software se pueden encontrar varias en la red o en libros algunos definiciones son: "software es la parte intangible de un sistema de computación (no es físico)" (Barrachina S., Mayo Rafael (n.d.)), o software son "Las instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el comportamiento deseado" (Barra P. Carlos (n.d.)). La Real Academia Española lo define como: "Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora". Esta última definición me parece más apropiada, pues el software al final de cuentas ejecuta instrucciones en código máquina para dar los resultados que el usuario desea.

Existen diferentes tipos de aplicaciones de software entre algunas están (Barrachina Sergio, Mayo Rafael n.d.):

- Software de sistemas. Sirve a otros programas, interactúan con el hardware de la computadora (compiladores, gestores, etc.)
- Software de tiempo real mide/analiza/controla sucesos del mundo real conforme ocurren.
- Software de gestión Procesa información comercial.

- Software de ingeniería o científico “Manejo de números”.
- Software empotrado Controla productos y sistemas de los mercados industriales y de consumo (hornos industriales, lavadoras, etc.)
- Software de PC Procesamiento de textos, hojas de cálculo, bases de datos, gráficos, entretenimiento, etc.
- Software de inteligencia artificial Sistemas expertos o basados en el conocimiento. Reconocimiento de patrones, habla, etc.

El desarrollo de software es básicamente como cualquier otro tipo de desarrollo que consta de varias etapas incluso como el humano con etapas como la concepción, nacimiento, crecimiento, maduración y término. Que es en sí un ciclo de vida. Algo parecido sucede con el software también consta de varias etapas. Según (Freeman, Peter, et al 1982), un ciclo de vida de software provee las bases para la categorización y control de las actividades del proyecto. Esas actividades están generalmente definidas en términos de:

- Productos resultantes
- Recursos requeridos
- Productos aceptados requeridos de distintas fases
- Metodologías de desarrollo usadas

Esos productos incluyen documentos tales como:

- Estándares a ser usados en el proyecto
- Especificaciones (requisitos, diseño)
- Documentación interna (ejemplo: diccionario de datos)
- Resultados experimentales de simuladores, prototipos, etc.
- Datos de pruebas y/o reportes
- Verificación y validación de datos/resultados
- Criterio de aceptación, resultados de pruebas, etc.

- Documentos externos (ejemplo, manuales de usuario, manuales de referencia, etc.
- Listado de programa/modulo

Las fases más típicas de un ciclo de vida de software mostradas en la figura X, son precedidas por una fase de planeación inicial, durante la cual las bases técnicas y económicas para el proyecto son establecidas a través de estudios comprensivos, desarrollos experimentales y formulación formal del concepto.

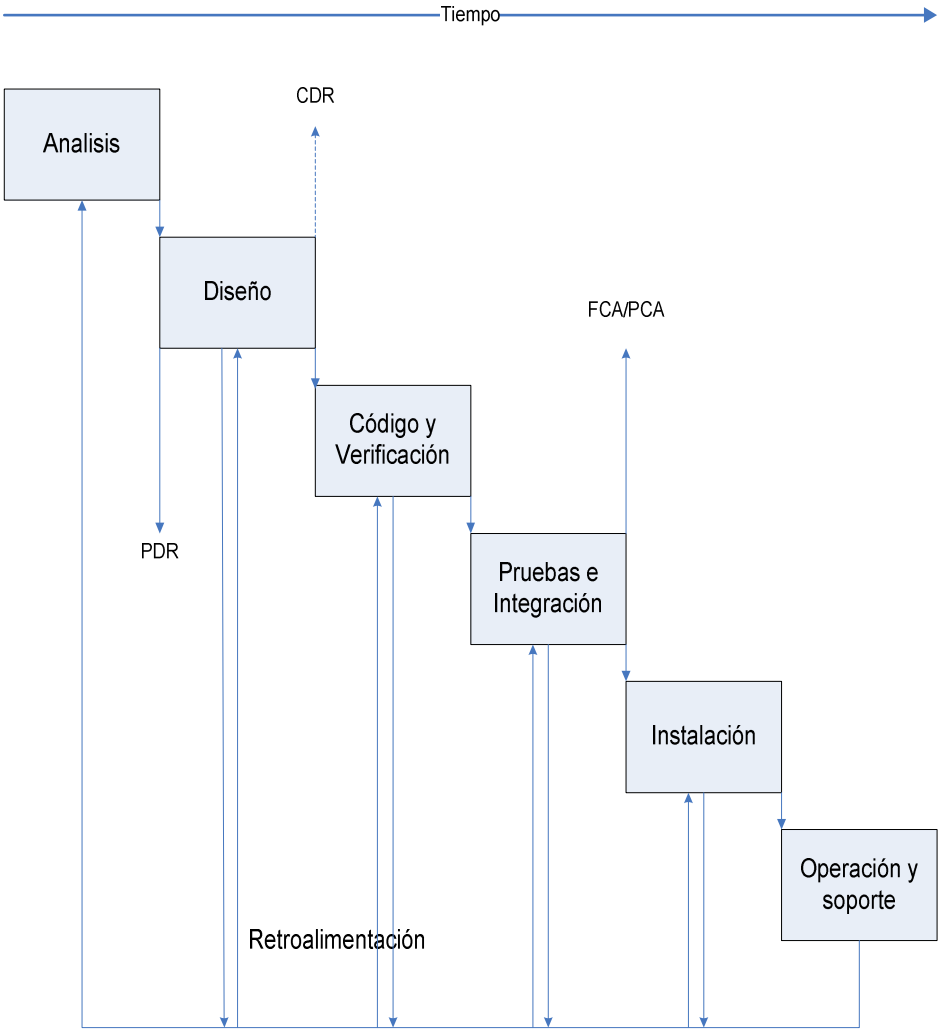


Figura 1
Ciclo de vida típico del desarrollo de software.
Fuente: Freeman, Peter et al, 1982, p. 46

Las etapas de desarrollo de software de la empresa de la cual se esta haciendo el estudio son las siguientes:

Planeación
Análisis de requisitos
Diseño
Codificación
Integración y Pruebas
Liberación del producto
Cierre del proyecto

Cada una de estas fases puede ser dividida de diferentes maneras, esta división se la llama Software Process Building Block (SPBB), las divisiones pueden ser como: un solo SPBB combinado, subdividido como una colección de SPBB separados (planeación, requisitos, diseño, etc.), una mezcla de fases combinadas y fases separadas (requisitos y diseño una fase, planeación otra fase, etc.), SPBB subdivididos como sea necesario (planeación1, planeación2, requisitos1, requisitos2, etc.)

Cada SPBB consiste de las partes mostradas en la figura siguiente:

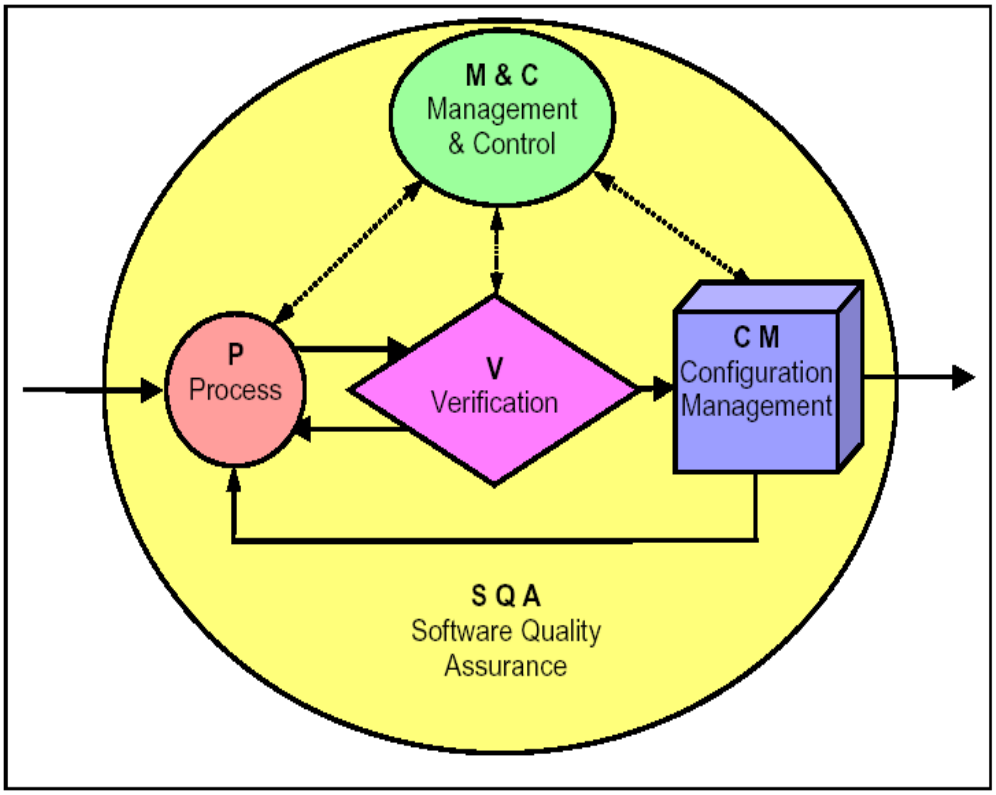


Figura 2
Software Process Building Block.
Fuente: Diseño de la empresa en investigación.

Proceso (P): El elemento de proceso describe la planeación o actividades de ingeniería para desarrollar o modificar los productos de trabajo del software el Proceso (P) debe especificar:

- Las tareas a ser ejecutadas
- Quien es responsable de ejecutar las tareas
- Entradas requeridas y
- Las salidas a ser producidas

Verificación (V): La verificación que se lleva a cabo en los ítems de configuración (revisiones, checklists de verificación, etc).

Administración de la Configuración (CM): Involucra el almacenamiento físico de los productos de trabajo, ítems de configuración y su control de versiones.

Control de la Configuración (CM): El proceso, la verificación, y los componentes de la administración de la configuración, deben recibir dirección y status para y desde la gerencia.

Aseguramiento de la calidad (SQA): El proceso, la verificación, la administración de la configuración y los componentes de la administración de la configuración del SPBB están sujetos a la supervisión del grupo de aseguramiento de la calidad para asegurar la integridad del proceso, el cual revisa las actividades de desarrollo y audita las salidas producidas de acuerdo al plan de SQA.

La productividad se puede considerar como: una proporción entre entradas y salidas (Institute for software research (2005)).

Las métricas de software son la medida que se da entre un producto de software y su correspondiente proceso de desarrollo. Son muy útiles para mejorar la calidad y productividad del software, midiendo cuantitativamente los modelos de un proceso de desarrollo de software (Adkins Gerald, Liu Yi, Yao Jenq-Foung, Williams Gita (2007)).

2.2 Antecedentes

Algunas investigaciones se han llevado a cabo para determinar si algún atributo de desarrollo de software, herramienta técnica, o combinación de las anteriores tiene algún impacto significativo en la producción de software. Estas investigaciones se han llevado a cabo en diferentes compañías tales como IBM, TRW, NASA, USC System Factory, ITT entre otras.

2.3. Características de la empresa

2.3.1 Descripción

Los principales actividades de la empresa son el diseño de sistemas de conexión y componentes, sistemas de distribución eléctricos / electrónicos, y el desarrollo y pruebas de software. Trabajando principalmente para la industria automotriz, aunque también se ha trabajado con compañías de diferentes ramos como lo es el de la industria telefónica.

2.3.2 Historia

Se empezó con lo que se llama un0 "join venture" entre una empresa extranjera y una mexicana formando el grupo de desarrollo de diseño en 1993, el cual se encargaría del diseño de sistemas de conexión y componentes.

Posteriormente en 1996 se hizo un contrato con Teléfonos de México para el diseño de diferentes componentes, tales como MUFAS, identificadores de llamadas, etc.

En 1997 empezaron los diseños de arneses en 2 y 3 dimensiones. Después de haber hecho el "join venture" y de haber iniciado con el área de diseño de sistemas de conexión y componentes y de haber visto que la respuesta de los ingenieros era la adecuada en este mismo año empezaron las pláticas para ver la posibilidad de abrir una nueva área la cual desarrollaría software para sistemas incrustados o embebidos.

Una vez que la compañía extranjera se convenció de la capacidad de los ingenieros mexicanos decidió invertir en un nuevo departamento el cual es el departamento de software que se consideraba como un pequeño grupo el cual empezó operaciones en Agosto de 1998. Este nuevo grupo constaba del área de RSE (Restrain Systems Electronics) la cual se encarga principalmente de programar las diferentes funciones para sensores electrónicos, despliegue de bolsas de aire, habilitación e inhabilitación automática de bolsas de aire.

En enero del año 2000 se abrió una nueva área, el área de Audio la cual se encargaría de programar las funciones necesarias para los radios de los vehículos de las diferentes plataformas de las compañías de automóviles como GM, FORD, CHRYSLER, entre otros.

Actualmente a esta área se le cambio el nombre por la de infotainment, que es un nuevo concepto entre diversión e información en el vehículo.

En Abril del 2001 se decidió abrir una nueva área la de Sistemas e IT&V (Independent Test and Verification). El área de sistemas se encarga de coordinar las diferentes áreas involucradas en el desarrollo de un producto como los el área mecánica, eléctrica, electrónica, software, pruebas, etc. El área de IT&V se encarga de realizar las pruebas necesarias a los productos de software desarrollados en la empresa.

En Agosto del 2001 se realizó un cambio de supervisor técnico, debido a que esta posición se cambia entre algunos de sus gerentes cada dos o tres años.

En Enero del 2002 el área de IT&V empezó operaciones desarrollando los primeros procedimientos de prueba o FTP's (Functional Test Procedures) para verificar varios de los proyectos.

En Noviembre de 2002 se abrieron varias nuevas áreas las cuales eran: HVAC (Heating Ventilation and Air Conditioning), la cual se encarga de programar las funciones de los aires acondicionados de los vehículos y la de BSE (Body and Security Electronics), la cual se encarga de programar las funciones de seguridad de los vehículos como lo son las de monitoreo de la presión de las llantas de los vehículos (TPMS), los avisos de uso de cinturón de seguridad, habilitar/deshabilitar bolsa de aire, puerta cerrada o abierta, sistemas antirrobo, etc.

En Febrero de 2003 se abrió el área de Diseño Eléctrico la cual se encarga de hacer o mejorar los diseños de las tarjetas usadas en los diferentes departamentos de desarrollo de software de la empresa.

2.3.3 Organización

El siguiente organigrama muestra los diferentes departamentos por las cuales esta compuesta la organización dentro. Los principales son: el de sistemas de conexión y componentes, sistemas de distribución eléctrico/electrónicos, software, pruebas y diseño eléctrico como se menciono anteriormente. Los demás departamentos son auxiliares para el funcionamiento de la organización como lo son el departamento de tecnologías de información, recursos humanos e ingeniería industrial. Existe también un nuevo departamento muy importante que es el de calidad, para asegurar que los productos que reciban los clientes cumplan o excedan sus expectativas.

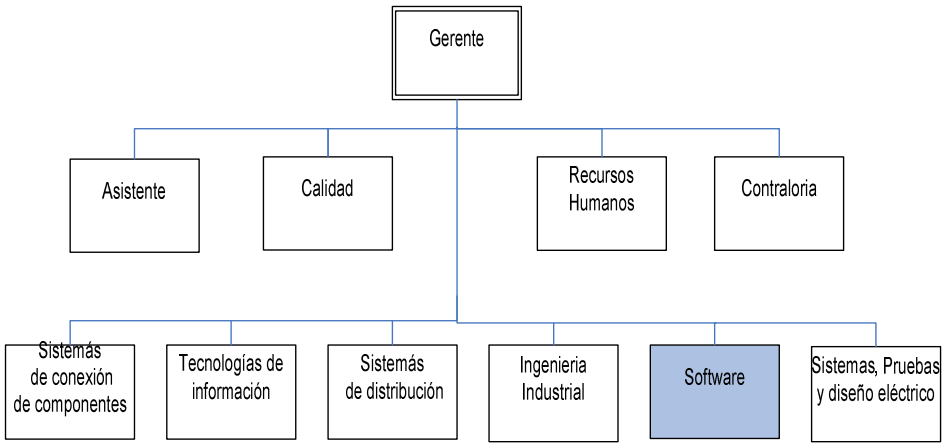


Figura 3
Organigrama de la empresa.
Fuente: Diseño propio con datos de la empresa en investigación.

Por el momento estas son las áreas de la empresa, sin embargo es posible que en un futuro no muy lejano se abran nuevas departamentos y áreas de trabajo como el departamento de hardware con todas las áreas de las cuales se compone como son las de diseño y pruebas.

2.3.4 Misión

Contribuir al logro de los objetivos de rentabilidad, crecimiento y consolidación de nuestros clientes y accionistas a través del diseño de sistemas mecánicos, eléctricos y electrónicos.

2.3.5 Visión

1. Ser la mejor opción en calidad, servicio, costo y tecnología para nuestros clientes y accionistas

2. Ser rentables (utilidad de operación y/o relación costo / beneficio)

3. Ser los promotores de la innovación tecnológica para la mejora de la rentabilidad

4. Ser altamente competitivos en:

- Desarrollo de sistemas eléctricos/electrónicos e iluminación en el mercado automotriz global
- Desarrollo de software para diseño y otras aplicaciones
- Desarrollo de pruebas de software y sistemas
- Diseño de componentes para nuevos productos y aplicaciones automotrices
- Desarrollo de sistemas de manufactura y logística
- Desarrollo de nuevos productos
- Desarrollo de sistemas de calidad, garantías, ambientales, seguridad e higiene

A parte de la misión y la visión se fijan diferentes políticas. En la empresa existen políticas de calidad, ambiental salud y de seguridad, las cuales son las siguientes:

Conscientes de nuestra responsabilidad ante los clientes, accionistas, trabajadores y la sociedad, todos los que integramos la empresa nos comprometemos a:

1. Satisfacer a nuestros clientes y accionistas proporcionando productos y servicios que cumplan con los requisitos de calidad, costo, entregas, tecnología y medio ambiente de nuestros clientes, buscando hacer crecer el negocio.

2. Trabajar en equipo bajo un sistema disciplinado, constante y de mejoramiento continuo para cumplir nuestros objetivos y metas.

3. Prevenir los riesgos de trabajo, asegurando que las condiciones de seguridad e higiene en el trabajo cumplan con la normatividad legal vigente.

4. Prevenir la contaminación a la atmósfera, agua, suelo y el uso o afectación de recursos naturales.

5. Evaluar el impacto ambiental de nuestras actividades y reducirlo estableciendo medios de control apropiados.

6. Cumplir con la normatividad legal, los requisitos de las dependencias gubernamentales y otros aplicables a nuestros productos, procesos y condiciones de trabajo.

Estas políticas así como la visión y misión son publicadas y difundidas al personal por medio del departamento de recursos humanos del cual se apoya la dirección para esta labor y muchas otras. Por lo que desde mi punto de vista este departamento seria como parte del staff. Ya que también ayuda a elaborar el plan de capacitación, descripciones de puestos, organigramas, etc.

Aunque también existe un comité de operaciones formado por los gerentes de las diferentes áreas de la empresa los ayudan al gerente general a la elaboración de los principales documentos del negocio como lo son: el manual de negocio, las políticas de calidad, ambiental y de seguridad e higiene mencionadas anteriormente, objetivos y

metas ambientales, etc. Este comité quizá también ayude en la elaboración de las estrategias, ver los proyectos estratégicos, etc.

No existe un departamento de finanzas, sin embargo el departamento de contabilidad o contraloría quizá ayude en la elaboración de los presupuestos (de costos, ventas, etc.), a la revisión de los estados financieros, elaborar las requisiciones de compra, etc. Este departamento considero que sería parte del staff también.

2.3.6 Principales enfoques

La empresa hace mucho énfasis en tres aspectos para mantener el negocio: Enfoque al cliente, Enfoque a los procesos y la mejora contínua.

Enfoque al cliente: El cliente es una organización o persona que recibe un producto y puede ser interno o externo. Los siguientes aspectos son importantes para la empresa con respecto a los clientes:

- Dependemos de nuestros clientes
- La Dirección debe asegurarse de que los requisitos del Cliente se Determinan y se Cumplen con el propósito de aumentar la satisfacción del Cliente

  Es vital:

  Determinar y entender sus necesidades y expectativas

  Cumplir sus necesidades y expectativas

  Demostrar al Cliente que continuamente estamos mejorando

El enfoque en los procesos también es importante, los procesos son conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman entradas en salidas

Las entradas de un proceso son generalmente salidas de otros procesos. Los procesos de la organización son generalmente planificados y puestos en práctica bajo condiciones controladas que aportan valor.

Un aspecto que es muy importante dentro de la empresa y en el que se hace énfasis es la Mejora Continua, en donde se identifican áreas de mejora: Producto, Procesos y / o sistemas de gestión de calidad y ambiental. Así como también documentar y utilizar las lecciones aprendidas de proyectos anteriores y aplicarlos a nuevos proyectos con el propósito de evitar cometer los mismos errores. Entonces la mejora continua afecta no solo a las áreas sino incluso también a las políticas y sistemas implementados en la empresa y esto se ha visto reflejado incluso en la misión, visión y en las políticas de calidad, ambiental, seguridad e higiene que han cambiado en los últimos años.

La mejora continua dentro de la empresa es como un ciclo en donde se involucran las acciones de Planear, Hacer, Verificar y Actuar.

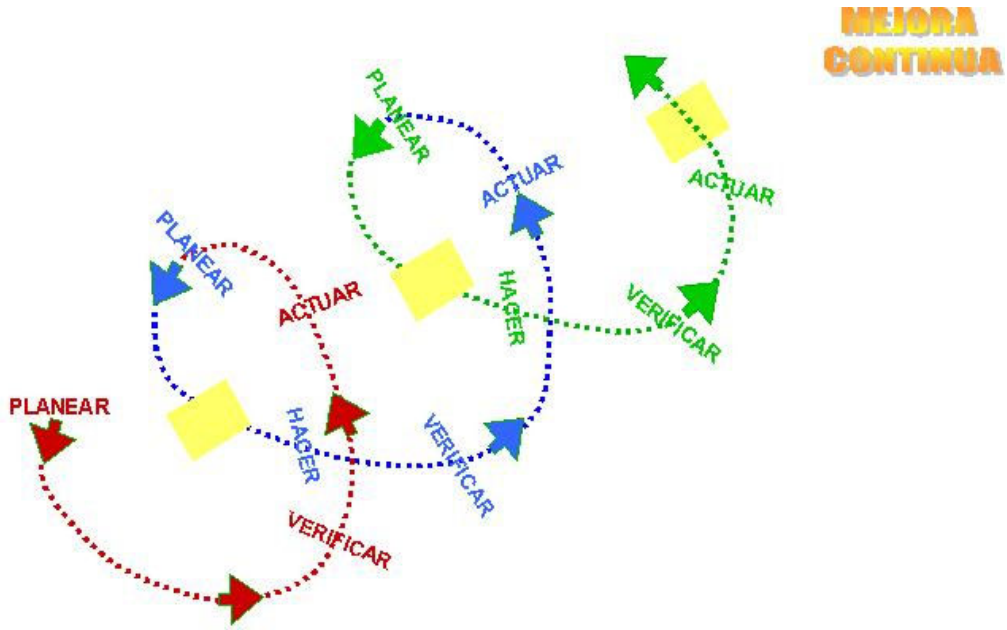


Figura 4.
Ciclo de mejora continua.
Fuente: Diseño de la empresa en investigación.

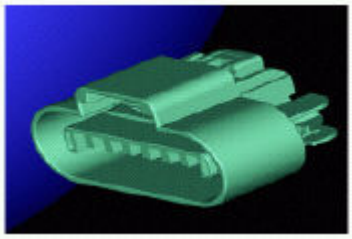
Otro de los aspectos no menos importantes y que también se consideran dentro de la empresa son los Insumos (recurso humano, materiales, energía, información, etc.) los cuales son las entradas a los procesos, y los productos que son el resultado de los procesos que son el software y los componentes que se diseñan.

2.3.7 Productos y funciones principales

Los principales productos que se diseñan en los departamentos de sistemas de conexión y componentes y sistemas de distribución eléctrico/electrónicos se encuentran los Arneses eléctricos que son un conjunto de cables que se utilizan para proveer de energía a los diferentes dispositivos que van a conectarse al vehículo, tales como, radios, luces, seguros, etc.

También en estos departamentos se diseñan los conectores para estos arneses, clips, abrazaderas y para otros usos dentro de la telefonía como mufas, identificadores de llamada, pagers, etc.

Arneses Eléctricos



Conector

Clip

Figura 5
Productos del departamento de diseño de componentes.
Fuente: Diseño de la empresa en investigación.

Dentro del departamento de software la principal función es la programación de microcontroladores (conocido como embedded systems o sistemas incrustados) para controlar las funciones de los diferentes dispositivos de seguridad que van dentro del vehículo, tales como: bolsas de aire, sensores, sistemas de frenos, control de combustible, entre otros como lo muestra la figura de abajo. Así como la comunicación entre estos y otros componentes del vehículo.

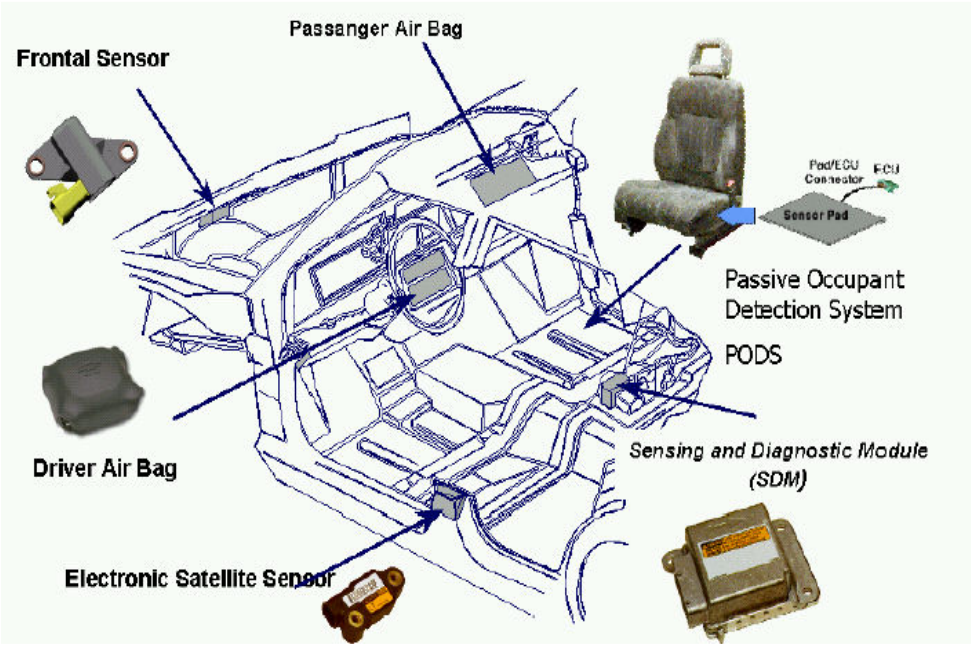


Figura 5
Productos del departamento de software.
Fuente: Diseño de la empresa en investigación.

También otra serie de productos muy importantes los cuales utilizan estos microcontroladores programados son los DVD's, radios, aire acondicionado, etc.

DVD Radio Aire Acondicionado

Figura 7
Productos del área de infotainment.
Fuente: Diseño de la empresa en investigación.

2.4 Investigaciones relacionadas

Diversas empresas, universidades e investigadores han llevado a cabo estudios acerca de la productividad en el desarrollo de software. (Scacchi W. 1995) establece lo siguiente:

"Medir la productividad del software presupone una habilidad para construir un programa de mediciones comparable a aquellos empleados en diseños experimentales para estudios de conducta. Es necesario asegurarse que las métricas empleadas son confiables, validas, exactas y repetibles".

El propone tres tipos de elementos para el diseño de la investigación: casos de estudio cualitativos, encuestas cuantitativas y estudios de triangulación. Los casos de estudio cualitativos pueden proveer conocimiento descriptivo más profundo acerca de cómo ocurre el trabajo de producción de software, que tipos de problema surgen y cuando. Tales estudios son generalmente eventos de una sola vez que son de bajo costo para iniciar, emplean estrategias de colección de datos antropológicas abiertas-cerradas, generalmente requieren análisis externo y producen hallazgos importantes aunque no generalizables.

Los estudios de encuestas cuantitativas emplean alguna forma de instrumentación, tal como un cuestionario para obtener datos en una manera adecuada para análisis estadístico. La encuesta de ejemplo debe ser cuidadosamente definida para asegurar resultados válidos y confiables. La siguiente secuencia de actividades puede ser llevada a cabo para un estudio cuantitativo:

1. Desarrollar un instrumento de colección de datos de productividad (forma o cuestionario)
2. Hacer una prueba piloto y revisar instrumentos iníciales para asegurar que los datos deseados pueden ser colectados desde asignaturas con modesto esfuerzo.
3. Implementar la actividad de colección de datos y calendario con planes a seguir en varias rondas en caso de tener muchas personas que no respondan.
4. Validar y "limpiar" los datos coleccionados (para remover y clarificar preguntas ambiguas)
5. Codificar los datos en variables analíticas, escalar y normalizar
6. Aplicar las variables seleccionadas análisis univariado para determinar estadísticas descriptivas de primer orden, variables de segundo orden y factores para análisis de varianza.
7. Seleccionar las variables significativas de primer y segundo orden desde análisis univariado hasta análisis multivariado, correlaciones parciales y análisis de regresión.
8. Formular un modelo analítico de aparente relación cuantitativa entre factores
9. Formular un modelo descriptivo del fenómeno estadístico analizado para consolidar los hallazgos y recomendaciones.

Los estudios de triangulación intentan extraer las fuerzas relativas de los estudios cualitativos y cuantitativos para maximizar la profundidad analítica, la generalización y la robustez.

(Scacchi W. 1995) menciona que las opciones para medir son muchas y complejas. Sin embargo, que es claro que solo enfocarse en atributos de programa como líneas de código, expresiones fuente o puntos de función no llevara a una comprensión significativa en la contribución o confusión de los efectos del proceso de producción de software o las características de configuración de la producción de la productividad del software ni viceversa.

De acuerdo a varios estudios hechos por varias empresas entre las cuales están IBM, TRW, ITT, entre otras, concluyo que diferentes características individuales y colectivas de los productos, procesos y configuración de producción afectan a la productividad del software. Sin embargo dentro de los estudios analizados hay según Scacchi algunas inconsistencias en determinar cuales características afectan lo que incrementa o decrementa la productividad del software. Por lo cual una estrategia integrada de medida o mejora de productividad debe tomarse en cuenta para las características de los productos, procesos, y configuraciones para delinear el potencial de las interrelaciones. Esto es necesario puesto que no se puede predecir con anticipación cuales variables revelaran el mayor significado o varianza en diferentes proyectos o medios ambientes computacionales. Similarmente, se debe esperar que las características del producto de software, los procesos y configuraciones necesiten ser medidas usando una combinación de métricas nominales, ordinales, de intervalos y de proporción. Se pueden considerar los siguientes elementos de productos de software, procesos de producción y configuraciones de producción.

Productos de software:

- Entregables (nuevo contra modificado) instrucciones fuente para etapas de desarrollo del ciclo de vida, incluyendo aquellas automáticamente transformadas o expandidas por herramientas de software, tal como generadores de aplicación,
- Análisis del desarrollo de software (conocimiento acerca de como un tema particular fue producido): análisis de requisitos, especificaciones, diseños arquitecturales y detallados y planes de prueba,

- Conocimiento de aplicación de dominio: conocimiento generado y hecho explicito acerca del dominio del problema (e.g., como electrónicamente cambiar a un gran volumen de tráfico de mensajes telefónicos bajo control computacional),
- Documentos y artefactos: reportes internos y externos, diagramas de sistema, y despliegues de terminal producidos para puntos importantes del desarrollo de calendarios, y guías de mantenimiento de sistema y
- Habilidades de desarrollo de software mejoradas, nuevas oportunidades ocupacionales o oportunidades de carrera para el personal del proyecto, y nuevas formas de cooperación con objeto de desarrollar otros productos de software.

Si se quiere estimar, medir y entender las variables que afectan la producción de software durante todo su ciclo de vida, se necesita delinear las actividades que constituyen el proceso de producción. Se puede entonces buscar aislar cuales tareas dentro de las actividades pueden dramáticamente impactar toda la productividad. Las actividades del ciclo de vida de software incluyen:

- Análisis de requisitos del sistema: frecuencia y distribución de cambios en requisitos del sistema operacionales durante toda la duración del proyecto,
- Especificaciones de requisitos de software (posiblemente incluyendo prototipos rápidos): número e interrelaciones de objetos computacionales, atributos, relaciones y operaciones centrales a los componentes críticos del sistema (e.g., aquellos en el kernel),
- Diseño de arquitectura de software: complejidad de la arquitectura de software medida por el número, interconexiones, y cohesión funcional de los módulos de software, junto con métricas comparables de organización de equipos de proyecto. También, medida por la frecuencia y distribución de cambios en la configuración de la arquitectura de software y la estructura de trabajo en equipo.

- Diseño de unidad de software detallado: tiempo gastado para diseñar un modulo dando el número de participantes del proyecto, y la cantidad de componentes existentes o reusados incorporados dentro del sistema que esta siendo desarrollado,
- Codificación de unidad de software (o generación de aplicación): tiempo para codificar módulos diseñados, y la densidad de inconsistencias descubiertas (bugs) encontradas entre un diseño detallado del módulo y su código fuente,
- Pruebas de unidad, integración y pruebas de sistema: proporción de tiempo y esfuerzo asignados para (contra actualmente gastado en) pruebas, esfuerzo gastado para reparar errores detectados, densidad de tipos de errores, y la cantidad de mecanismos automatizados para generar y evaluar los resultados de los casos de pruebas.

Variables similares para consideración pueden también ser articuladas para otras actividades de desarrollo y evolución de sistemas incluyendo aseguramiento de la calidad y administración de la configuración, pruebas preliminares del cliente, producción de la documentación del cliente, cambios en las entregas, operación sostenida y evolución del sistema.

Un número de estudios realizados por algunos investigadores indica que la codificación generalmente consume 10-20% del total del esfuerzo de desarrollo. Por otro lado, las pruebas de software e integración consumen la mayoría del total, representando generalmente el 50% del esfuerzo total. Actividades tempranas de desarrollo consumen del 10-15%. Claramente, el código fuente de software entregado es un producto valioso.

¿Hace la configuración del software alguna diferencia en la productividad?, Se espera que la producción de software en el sistema de defensa en TRW sea diferente al centro de programas de tecnología de software o al laboratorio de inteligencia artificial del MIT?. ¿Se espera que la producción de software sea diferente en varios departamentos de la misma corporación?.

La respuesta a todo esto es si. La configuración de la producción de software difiere en diferentes maneras incluyendo:

- Lenguaje de programación en uso (Ensamblador, Fortran, Cobol, C, C++, Java, etc.)
- Aplicaciones de computadora (interruptor de telecomunicaciones, comando y control, aplicaciones de Inteligencia Artificial, administración de bases de datos, análisis estructural, procesamiento de señales, control de procesos de refinería, etc.)
- Computadoras y sistemas operativos.
- Diferencias entre huésped (desarrollo) y objetivo (usuario final) medio ambiente computacional y configuración, así como también entre servidores de computadora y sistemas cliente.
- Herramientas de desarrollo de software (compiladores, editores, generadores de aplicación, generadores de reporte, sistemas expertos, etc.) y prácticas en uso (técnicas estructuradas, diseño modular, especificación formal, administración de la configuración, etc.)
- Base de habilidades del personal (personal con grado de técnico, licenciatura, maestría, etc.) y experiencia en área de aplicación.
- Dependencia en unidades organizacionales externas (vendedores de sistemas, Mercadotecnia, planeación y análisis de negocios, directorios de laboratorio, comités de control de cambios, clientes, etc.)
- Magnitud de la participación del cliente, su experiencia con aplicaciones de sistemas similares y su expectativa para soporte de sistemas ininterrumpido
- Frecuencia e historia de innovaciones de proyectos medios en el medio ambiente de producción de computación (que tan frecuente una nueva liberación sistema operativo, actualización de procesador de memoria, nuevos periféricos de computadora, etc. ocurren durante previo desarrollo de proyectos)

- Frecuencia e historia de anomalías y errores que surgen durante previo desarrollo de proyectos (e.g., sobrepasar fechas de entrega del calendario, pasarse del presupuesto, alta rotación inesperada del personal, nuevo software liberado y poco confiable, etc.)

Otras instituciones como el (Institute for software research de la universidad de california 2005), establece también que hay tres categorías que se deben medir en lo que al software se refiere, estas son:

- Productos de software
- Procesos de producción y estructuras de software
- Configuraciones de producción de software

Dentro de los productos de software consideran a los siguientes elementos:

- Instrucciones de código entregadas, puntos de función y componentes reutilizados o externos.
- Análisis de desarrollo de software
- Documentos y artefactos
- Conocimiento de aplicación de dominio
- Habilidades de desarrollo de software adquiridas en los productos o líneas de producto

Para los Procesos de producción y estructuras de software se consideran a los siguientes elementos:

- Análisis de requisitos: frecuencia y distribución de cambios de requisitos y otras métricas volátiles.
- Especificación: número de interconexiones de objetos computacionales, atributos, relaciones y operaciones en los sistemas y su volatilidad.
- Diseño arquitectural: complejidad del diseño, la volatilidad la configuración de la arquitectura, espacio de versión y composición de equipo de diseño; tasa componentes nuevos y reutilizados.

- Diseño de unidad: esfuerzo de diseño, número de potenciales defectos de diseño detectados y removidos antes de codificar.
- Codificación: esfuerzo para codificar los módulos diseñados; tasa de inconsistencias encontradas entre el diseño del modulo y la implementación de los codificadores.
- Pruebas: tasa de esfuerzo destinado a un módulo, subsistema o pruebas de sistema; densidad de tipos de errores conocidos; magnitud de mecanismos automatizados empleados para generar casos de prueba y evaluar los resultados.

En cuanto a las configuraciones de producción de software se consideran a los siguientes elementos:

- Lenguajes de programación
- Tipos de aplicación
- Plataformas de computación
- Disparidad entre las plataformas huésped y objetivo
- Desarrollo del medio ambiente de software
- Base de habilidades del personal
- Dependencia en organizaciones externas
- Magnitud de la participación del cliente o usuario final
- Frecuencia e historia de actualizaciones de plataformas de proyectos medios
- Frecuencia e historia de anomalías y errores en proyectos anteriores.

(Vosburg, J., B. et al 1984) examinaron datos de producción de software de 44 proyectos de 17 subsidiarias de ITT en 9 diferentes países.

Ellos identificaron dos tipos de factores relacionados con la productividad del software: factores relacionados con el producto que no son generalmente controlables por un administrador de proyectos y factores que son controlables por administradores y así proveer las oportunidades para mejorar la productividad.

Los factores relacionados con el producto que ellos identificaron incluyen:

- Restricciones de recursos de computadora: la productividad disminuye cuando el software que esta siendo desarrollado tiene restricciones de tiempo, utilización de memoria y ocupación de CPU.
- Complejidad del programa: productividad disminuye cuando el software son aplicaciones de sistemas operativos, comandos en tiempo real y control, y tolerante a fallas que requieren detección de errores extensivos, funciones de deshacer y rutinas de recuperación.
- Participación del cliente: la productividad incrementa con la participación del cliente, con experiencia de aplicación del cliente y la participación en requisitos y articulación de la especificación.
- Tamaño del producto del programa: la productividad disminuye conforme vaya aumentando el número de líneas de código.

Los factores que ellos identificaron relacionados con el proceso incluyen:

- Desarrollo de hardware-software concurrente: la productividad disminuye con desarrollo de hardware concurrente.
- Desarrollo de tamaño de la computadora: la productividad aumenta como lo haga el tamaño de la computadora (velocidad del procesador, capacidad de almacenamiento primario y secundario, etc).
- Estabilidad de las especificaciones y requisitos: la productividad aumenta con especificaciones estables y precisas.
- Uso de prácticas de programación modernas: la productividad aumenta con el uso extensivo de diseño top-down, diseño modular, revisiones de diseño, inspecciones de código y programas de aseguramiento de la calidad.
- Experiencia del personal: la productividad aumenta con más personal experimentado en el desarrollo de software.

(Smith D. A. 1988) menciona que en las organizaciones de desarrollo de software, el software y la productividad de programación esta frecuentemente definida como una métrica del tiempo y/o costo requeridos para entregar y mantener sistemas de software. Pero el sistema de software en si mismo no es el objetivo del desarrollo de software. Para satisfacer las necesidades de clientes y usuarios el sistema debe ser útil. Así que para entender la productividad del software, se debe también considerar el sistema de software resultante.

Hay dos dimensiones mayores de la productividad de la ingeniería de software:

- El cambio en la cantidad de software producido para un periodo de tiempo dado a un costo dado.
- La calidad del sistema de software resultante

El producto y los procesos necesitan ser considerados uno en el contexto del otro. Entonces la productividad puede ser medida y los resultados usados por equipos y administradores del proyecto para entender los efectos de cambios en el proceso de desarrollo.

Para medir el sistema de software y el proceso de desarrollo de software, se definen atributos por cada dimensión, y las técnicas de medidas asociadas a cada atributo. Estas métricas pueden ser usadas para describir el producto de software y el proceso de desarrollo. Entonces varios productos y proyectos pueden ser comparados unos con otros.

Al estar buscando si el uso de herramientas y el incremento de componentes comunes de reutilización habían mejorado la productividad de la organización, se identificaron atributos y definieron métricas para el sistema de software y las dimensiones del proceso de desarrollo.

Métricas de sistema de software

Los sistemas de software pueden ser medidos por métricas apropiadas al software y al producto. Los atributos del sistema de software incluyen el tamaño, la complejidad, usabilidad, mantenibilidad, corrección y ejecución. Los equipos de proyecto de software usan métricas como retroalimentación acerca de la calidad y la cantidad de su trabajo. Para este ensayo los atributos de defecto y tamaño son considerados.

Se considera importante el tamaño porque estudios han demostrado una relación directa entre el tamaño de un sistema de software y el costo de construir el software.

El atributo defecto es importante por dos razones:

- El producto de software necesita ser útil y si tiene una alta tasa de defectos, la utilidad es disminuida, si no es que eliminada.
- El costo de corregir los defectos después que el sistema de software ha sido liberado al cliente es alto. No se seria capaz de aplicar recursos a nuevos productos si el costo de defecto-corrección no esta bajo control.

La métrica compuesta índice de defectos es otra métrica del sistema de software. Varios datos han sido publicados indicando que el índice de defectos para la industria de software de América es 10 defectos por mil líneas de código y que el índice de defectos varía de 5 a 30 defectos por mil líneas de código.

Las definiciones de esas métricas se muestran en la siguiente tabla.

DEFINICIONES DE MÉTRICAS DE SISTEMA DE SOFTWARE

ATRIBUTO MÉTRICA	DEFINICIÓN
TAMAÑO	
T	$T = \frac{Te + Tc}{1000}$ *Te* = Número de líneas de código incluyendo declaraciones de datos *Tc* = Número de líneas de comentarios
CALIDAD	
C	*C* = Cb + Cd + Cr *Cb* = Número de reportes del cliente detectados como defecto o corregidos *Cd* = Número de reportes del cliente detectados como error de documentación *Cr* = Número de reportes del cliente detectados como restricción en el uso del software
TASA DE DEFECTO	
T D	$TD = \frac{C}{T}$

Tabla 1
Definiciones de métricas del sistema de software.
Fuente: Smith D. A. 1988, p .44

Métricas del proceso de desarrollo de software:

Los atributos que describen el proceso de desarrollo del sistema de software incluyen el costo de desarrollo (in recursos humanos, recursos de hardware y tiempo de calendario), el costo de mantenimiento, la predictibilidad de la planeación y de la capacidad del software entregado, y el número y tipo de revisiones de proyecto. Para este ensayo, el costo y las métricas de productividad de ingeniera son considerados.

Se define el costo en términos del total de meses de desarrollador desde el inicio del diseño (fase 1) hasta completar las pruebas externas y la liberación del producto a los clientes (fase 3). Usando tiempo en lugar de dinero se pueden comparar proyectos a través de los años sin que influyan los factores de la inflación.

Métricas complejas tal como la métrica de productividad de ingeniería son también métricas de proceso. La métrica de productividad de ingeniería es útil para entender si la cantidad de código producida por el equipo del proyecto esta cambiando, y los datos están normalizados así que proyectos de diferentes tamaños puedan ser comparados. Las definiciones de esas métricas se muestran en la siguiente tabla:

DEFINICIONES DE MÉTRICAS DEL PROCESO DEL DESARROLLO DE SOFTWARE

ATRIBUTO MÉTRICA	DEFINICIÓN
COSTO	
C	*C* = Cp1+ Cp2 + Cp3
	Cp1 = Número de ingenieros – meses para la fase 1
	Cp2 = Número de ingenieros – meses para la fase 2
	Cp3 = Número de ingenieros – meses para la fase 3
PRODUCTIVIDAD DE INGENIERIA	
P E	$P\,E = \frac{T}{C}$

Tabla 2
Definiciones de métrica del proceso de desarrollo de software.
Fuente: Smith D. A. 1988, p. 44

Hay limitaciones para un estudio como este. Esas métricas no reflejan el producto de software completo. Por ejemplo el tamaño o métricas de costo no cuentan para el usuario y documentación del proyecto, materiales de entrenamiento, archivos de ayuda, archivos de mensaje de producto específico y pruebas y procedimientos de comando.

El dato de defecto considera solo los defectos encontrados y reportados por clientes, los defectos encontrados por el equipo de proyecto u otros usuarios digitales no son incluidos. Adicionalmente, la severidad de el defecto no es considerado. Puesto que la cantidad de uso que un producto recibe también influye en el número de defectos encontrados y reportados, el uso del producto (el cual puede ser reflejado por el número de licencias vendidas, número de usuarios, o uso total mensual) seria de ayuda. Esos datos pueden ser usados para normalizar datos para comparaciones.

Por su parte (H. Kan S. 2004) categoriza las métricas de software de la siguiente manera:

- ✓ Métricas de calidad del producto
- ✓ Métricas de calidad en el proceso
- ✓ Métricas para el mantenimiento de software

Dentro de las métricas de calidad del producto describe las siguientes:

- Tiempo medio a falla. Es usado en sistemas críticos de seguridad tal como: sistemas de control de tráfico aéreo, aplicación de la electrónica a la aviación (avionics) y control de armamento.
- Densidad de defecto. Esta dada por la relación del número de defectos entre el número de líneas de código. Dentro de esta métrica también entran los puntos de función que direccionan algunos de los problemas asociados con la cuenta de líneas de código en tamaño y métricas de productividad, especialmente la diferencias en número de líneas de código que resultan debido a diferentes niveles de lenguajes de programación. Es un peso total de cinco componentes principales que abarca una aplicación:
  - Número de entradas externas
  - Número de salidas externas
  - Número de archivos internos lógicos
  - Número de archivos de interface externos
  - Número de encuestas externas
- Problemas del Cliente. Se toman en cuenta todos los problemas que el cliente encuentra al estar usando el producto, contando incluso problemas en la documentación o errores del usuario. Esta métrica esta expresado de la siguiente forma:
  - Problemas de usuario al mes = Número total de problemas que los usuarios reportaron (defectos reales o problemas no orientados a defectos) en un periodo de tiempo dado + Número total de licencias mensuales en el periodo.

- Satisfacción del Cliente. Mide el porcentaje de satisfacción del cliente:
  - Completamente satisfecho
  - Satisfecho
  - Neutral
  - Insatisfecho
  - Completamente insatisfecho

Dentro de las métricas de calidad en el proceso describe las siguientes:

- Densidad de defecto durante pruebas de máquina. Son las pruebas que se realizan después de que el código fue integrado en la biblioteca del sistema.
- Patrón de llegada de defectos durante pruebas de máquina. Se checa el patrón que tienen los defectos en las pruebas de máquina.
- Patrón de eliminación de defectos basados en fase. Da seguimiento de los defectos en todas las fases del ciclo de desarrollo, incluyendo a las revisiones de diseño, inspecciones de código y verificaciones formales antes de las pruebas
- Efectividad de eliminación de defectos. Mide la eficiencia para remover los defectos, se puede expresar de la siguiente forma:
  - Defectos removidos durante una fase de desarrollo / Defectos latentes en el producto * 100 %

Dentro de las métricas para el mantenimiento de software describe las siguientes:

- Carga de trabajo fija e índice de administración de carga de trabajo. Es una métrica relacionada a la tasa de llegada de los defectos y la tasa de disponibilidad para arreglar los defectos. El índice de carga de trabajo se refiere a los problemas abiertos o no resueltos, esta dado por:
  - No. Problemas cerrados durante el mes / No, de problemas recibidos durante el mes

- Tiempo de respuesta para arreglo de defectos y grado de de reacción para arreglo de defectos. El grado de reacción esta relacionado con las expectativas del cliente, con la habilidad de cumplir con lo que este pidió. El tiempo de respuesta es generalmente calculado como sigue para todos los problemas así como también por nivel de severidad:
  - Tiempo medio de los problemas desde que se abren hasta que se cierran
- Porcentaje de arreglos de fallas delincuentes. Esta expresado por:
  - No. De reparaciones que excedieron el criterio de tiempo de respuesta por nivel de severidad / No. De reparaciones entregadas en un tiempo especifico * 100)
- Calidad del arreglo de falla. Se refiere al número de defectos introducidos después de haber reparado un defecto.

Algunas métricas establecidas por algunas empresas para medir la calidad y productividad que (H. Kan Stephen 2004) lista en su libro son las siguientes:

Motorola:

- Fase de ciclo de vida y métrica de seguimiento a programa o itinerario: Dar seguimiento al programa basado en fases del ciclo de vida comparar lo actual con lo planeado.
- Métrica de seguimiento de valor de costo: Comparar el costo actual del proyecto contra el costo presupuestado.
- Métrica de seguimiento a requisitos: Dar seguimiento al número de cambios en requisitos a nivel de proyecto.
- Métrica de seguimiento a diseño: Dar seguimiento al número de requisitos implementados en el diseño contra el número de requisitos escritos.
- Métrica de seguimiento a tipos de falla.

Hewlett Packard

- Promedio de defectos reparados por día

- Promedio de horas de ingeniería / número de defectos reparados
- Promedio de defectos reportados por día
- Defectos entre tiempo de pruebas
- Ramificaciones cubiertas entre número total de ramificaciones: Cuando se ejecuta un programa indica el número de puntos de decisión que fueron ejecutados.
- Porcentaje de tiempo extra: Promedio de tiempo extra / No. De horas de trabajo a la semana.

IBM

- Llamadas del cliente debido a fallas por mes
- Time de respuesta de arreglo de fallas
- Número de reparaciones defectuosas
- Efectividad por fase (para cada fase de inspección y pruebas)
- Cobertura de inspección y esfuerzo
- Fallas de compilación y defectos de construcción/integración
- Severidad de defectos
- Causa de defecto y problema de análisis de componente
- Confiabilidad: Tiempo medio para carga inicial del programa durante pruebas
- Nivel de estrés del sistema durante las pruebas medido en el nivel de uso del CPU en término de número de horas de CPU por sistema por día durante pruebas de sistema.
- Número de veces que aborta el sistema o se queda colgado en pruebas de estrés o de sistema.

Otra métrica del producto que se considera importante es la complejidad ciclomática propuesta por (McCabe J. T. 1976) la cual se define como:

La complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el numero de caminos independientes del conjunto básico de un programa y nos da un limite superior para el numero de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Esta puede utilizarse en varias áreas incluyendo:

- Análisis de riesgo en desarrollo de código: Mientras el código esta en desarrollo, su complejidad puede ser medida para estimar el riesgo inherente.
- Análisis de riesgo de cambio durante la fase de mantenimiento: La complejidad del código tiende a incrementarse a medida que es mantenido durante el tiempo. Midiendo la complejidad antes y después de un cambio propuesto, puede ayudar a decidir cómo minimizar el riesgo del cambio.
- Planificación de Pruebas: El análisis matemático ha demostrado que la complejidad ciclomática indica el número exacto de casos de prueba necesarios para probar cada punto de decisión en un programa.
- Reingeniería: Provee conocimiento de la estructura del código operacional de un sistema. El riesgo involucrado en la reingeniería de una pieza de código está relacionado con su complejidad.

La Complejidad Ciclomática puede calcularse de la siguiente manera:

$$M = E - N + P + R$$

Donde:

M = Complejidad ciclomática.

E = Número de aristas del grafo. Una arista conecta dos vértices si una sentencia puede ser ejecutada inmediatamente después de la primera.

N = Número de nodos del grafo correspondientes a sentencias del programa.

P = Número de componentes conexos correspondientes a las diferentes subrutinas, funciones o métodos.

R = Número de salidas (returns) del programa

Una versión simplificada para el cálculo de la Complejidad Ciclomática es la siguiente:

M = Número de condiciones + 1

III. CARACTERÍSTICAS DE LA INVESTIGACIÓN

3.1 Justificación de la investigación

La presente investigación se justifica en virtud de que en la empresa se llevan algunas métricas aisladas para saber como se van comportando los proyectos, pero no se ha hecho una investigación más exhaustiva para detectar cuales son los factores e indicadores más significativos que ayuden a medir la productividad del área de software.

Esta investigación es importante hacerla para poder descubrir esas variables y empezar a medir la productividad de manera efectiva y así ayudar a detectar las áreas de mejora, es decir serviría entre otras cosas para:

- Incrementar el volumen de trabajo con el mismo número de ingenieros
- Hacer el mismo trabajo con menor número de ingenieros
- Desarrollar productos más complejos o de mayor valor de mercado con el personal actual.
- Evitar contratar personal adicional para incrementar la carga de trabajo
- Reducir el número de errores en los productos que se entregan al cliente.
- Reducir el esfuerzo para rectificar errores
- Identificar defectos del producto en etapas tempranas del desarrollo de software.
- Identificar cuellos de botella
- Recursos no utilizados adecuadamente,
- Incrementar la calidad del software
- Desarrollar mejores productos a menor precio

Si se logran identificar esas variables los resultados que arroje esta investigación puede ayudar a descubrir otras que ayuden a medir la productividad de diferentes áreas de la empresa como lo seria para el área de pruebas de software la cual está muy relacionada si no es que ligada al área de desarrollo de software.

3.2. Planteamiento del problema

Las métricas que se están llevando actualmente en el área de software están más relacionadas con lo que es el cálculo de estimados de tiempo para el desarrollo de los proyectos. Para medir la productividad no existe mucho al respecto sólo se cuenta con algunas métricas como el número de cambios debido a errores introducidos por los ingenieros o el número de entregas a tiempo a los clientes de las versiones de los proyectos.

Debido a que existe poca información para saber que tan productiva esta siendo el área de software es una buen momento para encontrar los conocimientos e información que ayuden a determinar los principales factores e indicadores que sirvan para medir la productividad en el área de software y así poder compararse con la competencia y detectar las áreas de mejora.

3.2.1 Preguntas de investigación

¿Cuales son los factores e indicadores más significativos en el proceso de desarrollo de software que ayuden a medir la productividad en el área de software de la empresa?

¿Existe alguna correlación entre esos factores e indicadores y la medición de la productividad en el desarrollo de software?

3.3 Definición de los objetivos

3.3.1 Objetivo general

- Identificar los factores e indicadores más significativos que ayuden a medir la productividad en el proceso de desarrollo de software en la empresa.

3.3.2 Objetivos particulares

- Identificar los procesos más significativos involucrados en el área de desarrollo de software.

- Determinar de los procesos más significativos los factores e indicadores relacionados con la productividad.
- Buscar si existe alguna relación entre las los factores e indicadores identificados como relacionadas con la productividad.

3.4 Definición del universo

La investigación esta enfocada al área de desarrollo de software. Existen diversos grupos de desarrollo de software dentro de la empresa conformados por ingenieros en electrónica, en sistemas computaciones, en electrónica y comunicaciones y en mecatrónica. Los grupos son:

RSE Restrain Systems conformada por 70 ingenieros

Infotainment (information and entertainment), integrada por 75 ingenieros

SDT (Software Development Tools) constituida por 5 ingenieros.

3.5 Tamaño y tipo de la muestra

Para trabajar en la investigación se va a tomar una muestra de al menos el 80% de los supervisores (10) un grupo de 12 de las diferentes áreas mencionadas anteriormente, es decir se trata de una muestra no probabilística por conveniencia de acuerdo a las características, tiempo y recursos de la investigación.

3.6 Identificación de variables

La variable independiente es: Los factores e indicadores involucrados en los procesos del desarrollo de software.

La variable dependiente es: La medición de la productividad en el desarrollo de software.

3.6.1 Definición operacional de las variables

Relación de los factores e indicadores involucrados en los procesos del desarrollo de software: Esta dada por la manera en que se hagan interactuar las diferentes variables identificadas en el proceso de desarrollo de software y que están relacionadas con la productividad.

Medición de la productividad: Se refiere a los valores identificados y registrados en el proceso de desarrollo del software.

3.7 Hipótesis

La medición de la productividad depende de la relación matemática de los factores e indicadores involucrados en los procesos del desarrollo de software.

IV. METODOLOGÍA

4.1 Diseño del estudio

En cuanto al diseño a emplear en el presente estudio es no experimental, ya que no se posee control directo de las variables independientes, debido a que sus manifestaciones ya han ocurrido. Se hacen inferencias entre las variables, sin intervención directa, de la variación concomitante de las variables independiente y dependiente. (Kerlinger 2002)

4.2 Tipo de estudio

Estudio de tipo correlacional, de campo, transversal de comprobación de hipótesis, ya que busca descubrir o revelar relaciones. (Kerlinger 2002)

4.3 Instrumento

El tipo de instrumento empleado en este estudio son formatos basados en la jerarquización de elementos (ver apéndice).

4.4 Procedimiento

El procedimiento para elegir los factores e indicadores que pueden ser de utilidad para la empresa se realizó de la siguiente manera:

1. Se hizo una primera ronda de selección de los factores e indicadores propuestos por varios autores y empresas mencionados en el marco teórico.
2. A los factores e indicadores se les dio el nombre de métricas. La selección fue realizada por el líder de equipo del comité de métricas de la empresa. Este sugirió que se dejaran aquellos que estaban relacionados con productividad pero que impactaran en la calidad, el costo y el servicio (la lista de métricas propuestas en la primera ronda se puede ver en el apéndice 1).
3. De las métricas listadas en la forma del apéndice 1 se filtraron algunas métricas y se eligieron otras que podían ser más factibles de ser implementadas en la empresa.
4. De las métricas que fueron elegidas se hizo otra forma (ver apéndice 2) que se paso a los supervisores de grupo para que eligieran de acuerdo a su criterio

cuales de las métricas propuestas consideran más factibles de ser implementadas en la empresa.

5. Las métricas fueron explicadas y discutidas en una junta con el comité y los supervisores de área.
6. Los resultados de la jerarquización fueron mostrados a los supervisores.
7. De ahí se discutieron los resultados y se eligieron las métricas más útiles o factibles de ser adoptadas por la empresa.

4.5 Procesamiento

Se realizó un análisis de frecuencias para ver que métricas fueron elegidas como las de mayor prioridad.

Una vez visto que métricas tuvieron la mayor prioridad, se discutieron para ver cuales podrían ser adoptadas para ser medidas en la empresa.

V. RESULTADOS, DISCUSIÓN DEL TEMA Y PROPUESTAS

5.1 Características de la población

Las formas fueron llenadas por los supervisores de las diferentes áreas de software y de IT&V. Un total de nueve supervisores de una población de doce fueron los que llenaron las formas. Siete de ellos cuentan con un nivel de licenciatura y dos cuentan con maestría. Su antigüedad en el puesto va de cuatro a diez años.

5.2 Resultados

A continuación se muestran los resultados obtenidos de los supervisores de las diferentes áreas de la empresa.

No.	Métrica	Respuestas de nueve supervisores por orden de prioridad por métrica								
1	Densidad de defectos	6	1	8	9	9	2	2	9	1
2	Problemas de usuario por mes	5	2	1	10	8	9	1	5	11
3	Complejidad ciclomática	8	6	10	8	4	10	10	10	10
4	Puntos de función	11	7	11	11	1	11	11	11	9
5	Patrón removedor de defectos basado en fase	7	9	7	7	5	6	3	4	8
6	Administración de carga de trabajo	2	4	4	1	10	7	5	6	7
7	Tiempo de respuesta para arreglar defectos	1	3	6	3	7	4	4	1	6
8	Porcentaje de reparaciones delincuentes	9	8	5	4	6	3	7	7	2
9	Calidad del defecto reparado	3	5	3	5	3	5	6	2	3
10	Métrica de seguimiento de valor de costo	4	10	2	2	2	1	8	3	4
11	Porcentaje de tiempo extra	10	11	9	6	11	8	9	8	5

Tabla 3
Resultados de 9 supervisores de distintas áreas.
Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores.

En la tabla anterior se ve el valor que cada supervisor asignó a cada métrica propuesta en escala del 1 al 11. La siguiente gráfica muestra el valor que cada una de los nueve supervisores dio a cada una de las métricas.

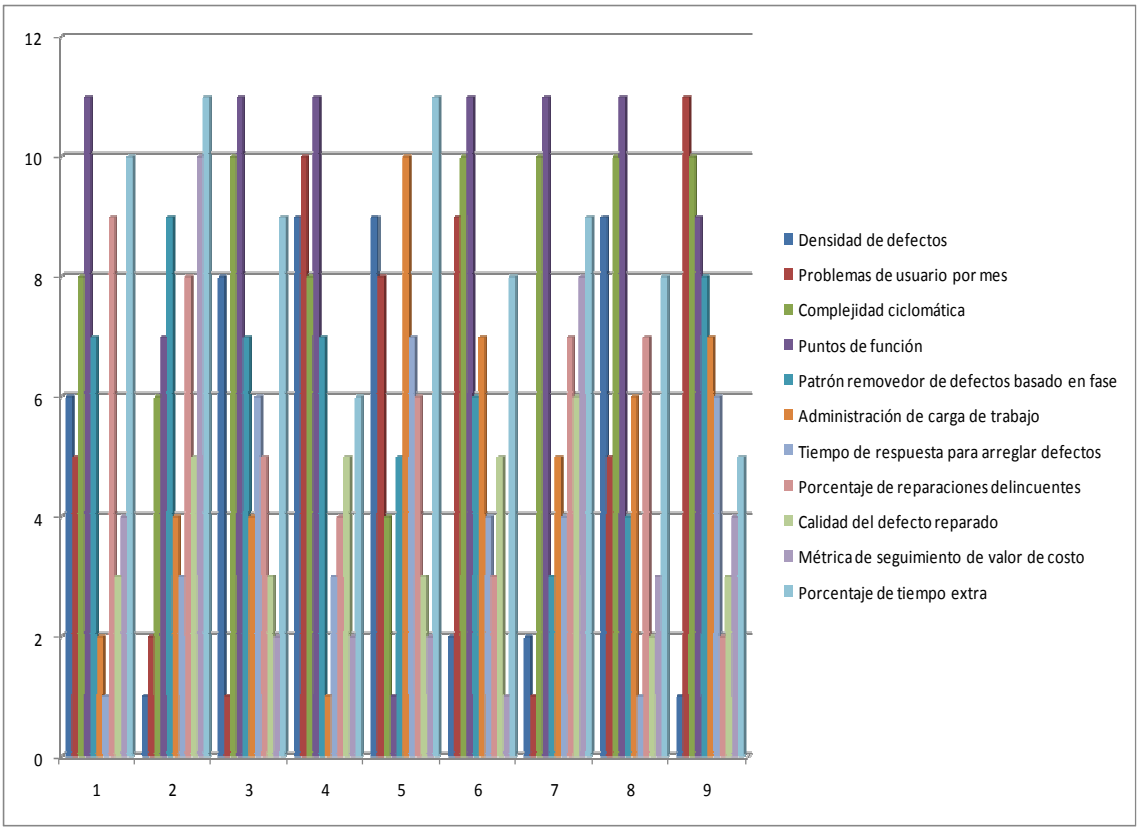


Figura 8

Gráfica de resultados.

Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores.

La siguiente tabla muestra las frecuencias para la prioridad en cada una de las métricas:

Frecuencia de prioridad por métrica como:	1	2	3	4	5	6	7	8	9	10	11
Métrica 1	**2**	2	0	0	0	1	0	1	**3**	0	0
Métrica 2	**2**	1	0	0	2	0	0	1	1	1	1
Métrica 3	0	0	0	1	0	1	0	**2**	0	**5**	0
Métrica 4	1	0	0	0	0	0	1	0	1	0	**6**
Métrica 5	0	0	1	1	1	1	**3**	1	1	0	0
Métrica 6	1	1	0	**2**	1	1	2	0	0	1	0
Métrica 7	**2**	0	2	**2**	0	**2**	1	0	0	0	0
Métrica 8	0	1	1	1	1	1	2	1	1	0	0
Métrica 9	0	1	**4**	0	**3**	1	0	0	0	0	0
Métrica 10	1	**3**	1	**2**	0	0	0	1	0	1	0
Métrica 11	0	0	0	0	1	1	0	**2**	2	1	2
Máximo número de frecuencia	**2**	**3**	**4**	**2**	**3**	**2**	**3**	**2**	**3**	**5**	**6**

Tabla 4
Frecuencia de métricas para cada prioridad.
Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores.

En la tabla anterior se puede observar el número de veces que cada una de las métricas fue elegida para un número de prioridad. Por ejemplo la métrica1 fue elegida por 2 supervisores como prioridad 1. La siguiente gráfica también muestra la frecuencia para cada una de las métricas (las métricas que no aparecen es que tienen un valor de 0)

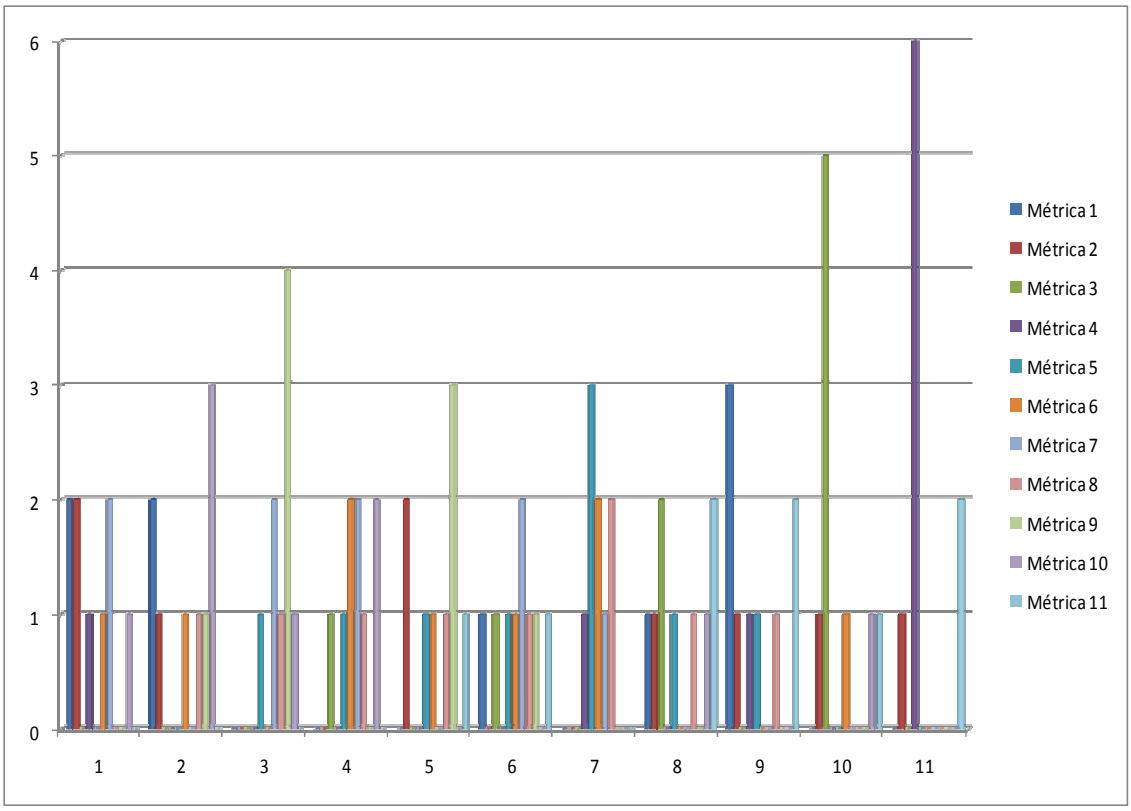


Figura 9
Gráfica de frecuencias.
Fuente: Diseño propio usando información obtenida de la retroalimentación de los supervisores.

Para obtener las métricas de mayor a menor prioridad, se contabilizaron el número de votos obtenidos para cada métrica en cada prioridad de la tabla de frecuencias. Así para obtener la métrica que fue elegida con prioridad 1, el mayor número de votos es 2 habiendo 3 métricas (1, 2 y 7) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 2, el mayor número de votos fue de 3 habiendo solo 1 métrica (métrica 10) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 3, el mayor número de votos fue de 4 habiendo solo 1 métrica (métrica 9) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 4, el mayor número de votos fue de 2 habiendo 3 métricas (métrica 6, 7 y 10) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 5, el mayor número de votos fue de 3 habiendo solo 1 métrica (métrica 9) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 6, el mayor número de votos fue de 2 habiendo solo 1 métrica (métrica 7) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 7, el mayor número de votos fue de 3 habiendo solo 1 métrica (métrica 5) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 8, el mayor número de votos fue de 2 habiendo solo 1 métrica (métrica 3) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 9, el mayor número de votos fue de 3 habiendo solo 1 métrica (métrica 1) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 10, el mayor número de votos fue de 5 habiendo solo 1 métrica (métrica 3) obtenido este valor.

Para obtener la métrica que fue elegida con prioridad 11, el mayor número de votos fue de 4 habiendo solo 1 métrica (métrica 4) obtenido este valor.

Los resultados no son muy claros durante la primera ronda pues hubo varias métricas que obtuvieron el mismo número de votos que otras. Al parecer lo que para algunos era importante para otros no lo era tanto y viceversa. Sin embargo al discutir los resultados con todos los supervisores se pudo llegar a un consenso para elegir las métricas que para las necesidades de la empresa se consideraron más importantes.

5.3 Discusión del tema

A pesar de haber muchas métricas que se pueden considerar medir para el desarrollo de software de acuerdo a lo encontrado en el marco teórico, unas son más útiles que otras, el tratar de medir todas es complicado y causa sobre carga de trabajo con

consecuencias en la productividad. Por lo que se determino empezar a medir unas pocas métricas.

A continuación se describen las once métricas en términos de su importancia en relación a la calidad, servicio y costo.

No.	MÉTRICA	DESCRIPCIÓN
	MÉTRICAS DE CALIDAD	
1	Densidad de defectos	Su importancia reside en que ayuda a determinar el porcentaje de defectos introducidos por cierto número de líneas de código.
2	Problemas de usuario por mes	Ayuda a dar seguimiento a los problemas reportados por el cliente.
3	Complejidad ciclomática	Ayuda a determinar la complejidad lógica de un programa para hacer un análisis de riesgo de desarrollo o planeación de pruebas.
4	Puntos de función	Ayuda a determinar que tan funcional (que tanto hace) es el producto que se entrega al cliente.
	MÉTRICAS DE SERVICIO	
5	Patrón removedor de defectos basado en fase	Ayuda a determinar en que fase del desarrollo de software se están introduciendo más errores
6	Administración de carga de trabajo	Ayuda a determinar que tan rápido se cierran los errores reportados por los clientes en un determinado tiempo.
7	Tiempo de respuesta para arreglar defectos	Determinar que tan rápido se están solucionando los problemas.
8	Porcentaje de reparaciones delincuentes	Determina cuantos defectos quedaron pendientes de reparar en un determinado tiempo.
9	Calidad del defecto reparado	Ayuda a determinar cuantos errores desencadeno la reparación de algún error reportado.
	MÉTRICAS DE COSTO	
10	Métrica de seguimiento de valor de costo	Ayuda a determinar el costo real de un proyecto a partir de un costo estimado. Se le va dando seguimiento a cada tarea y se pone el tiempo real que tomo en hacerla. Esto ayuda a determinar en que fases del desarrollo se esta fallando en la estimación para hacer estimaciones más exactas en futuras etapas del proyecto o en proyectos similares futuros.
11	Porcentaje de tiempo extra	Determina que tanto tiempo extra esta siendo requerido en los proyectos y ver las causas por las cuales se esta presentando.

Tabla 5
Descripción de métricas en términos de su importancia.
Fuente: Diseño propio usando información del marco teórico.

Según los resultados obtenidos después de haber jerarquizado los factores o métricas en el desarrollo de software, las métricas que la supervisión considero más importantes fueron:

Densidad de defectos

Problemas de usuario por mes

Tiempo de respuesta para arreglar defectos

Métrica de seguimiento de valor de costo

Calidad del defecto reparado

Estas métricas se discutieron en una junta posterior y se analizaron para ver si podían ser factibles de ser medidas de acuerdo a las necesidades de la empresa. Se descarto la métrica de densidad de defecto pues esta se considero de menos importancia además de que podía ser cubierta de alguna manera por las demás. También se descarto la métrica de tiempo de respuesta para arreglar defectos pues esta depende de la experiencia y complejidad del defecto lo cual habría también que determinar de alguna manera.

Entonces como un inicio para empezar a tomar mediciones se determino tomar una métrica por cada clasificación (calidad, servicio y costo), para no causar demasiada sobrecarga de trabajo a los líderes de proyecto e ingenieros de software. Entonces las métricas que se consideraran medir para una primera etapa de acuerdo a su clasificación son:

Calidad

Problemas de usuario por mes

Servicio

Calidad del defecto reparado

Costo

Métrica de seguimiento de valor de costo

La métrica de problemas de usuario por mes se tomo en cuenta para la primera etapa porque es importante estar al pendiente de los problemas que llegan al usuario final y darles seguimiento, determinar de que tipo son y ver en que es en lo que se esta fallando para evitarlo en las siguientes liberaciones de software.

La métrica de calidad del defecto reparado se eligió porque se puede ver cuantos errores son introducidos después de "reparar" algún defecto y en que etapas del proceso de desarrollo son detectados (diseño, codificación, verificación, pruebas externas, etc.)

La métrica de seguimiento de valor de costo se eligió porque se puede determinar el costo real de elaboración de un proyecto dando seguimiento a cada una las tareas que se desglosaron y su tiempo estimado. Entonces se puede ver del tiempo estimado de cada tarea cual fue le tiempo y costo real, de manera que ayude a ver en que partes del desarrollo del proyecto no se esta estimando de manera adecuada y hacer ajustes para las siguientes etapas de liberación del proyecto.

Estas métricas se van estar monitoreando constantemente para ver cuales son los datos que se arrojan.

Debido a que se necesita contabilizar las incidencias que ocurren en las métricas involucradas en el desarrollo de software se acepta la hipótesis:

La medición de la productividad depende de la relación matemática de los factores e indicadores involucrados en los procesos del desarrollo de software.

CONCLUSIONES

Después de haber investigado el trabajo de varios autores e instituciones involucrados en el proceso de desarrollo de software se encontró que existen muchos factores e indicadores que pueden ser medidos para determinar la productividad en el proceso de desarrollo de software, Sin embargo muchos de ellos solo pueden ser más informativos que útiles para la toma de decisiones importantes. Muchas veces estar midiendo tantos factores o indicadores puede causar sobre carga de trabajo, por lo que la productividad se puede ver afectada y entonces el objetivo de medir la productividad para ser más eficientes ya no se cumple. Por lo que es importante elegir de todos los factores e indicadores aquellos que puedan ser más útiles a las necesidades de la empresa.

Una vez elegidos los factores que se consideran más apropiados es conveniente determinar la forma de estarlos midiendo de manera más adecuada para obtener datos más exactos y ayuden a la toma de decisiones. Esto no es fácil ya que hay que considerar muchas variables tales como la experiencia del personal y la complejidad de los productos que se desarrollan Sin embargo una vez solucionado este problema los datos que se obtengan pueden ser de mucha utilidad para la toma de acciones correctivas y mejoras en los procesos.

Lo primero que se hará entonces en una primera etapa es medir las tres métricas propuestas anteriormente y checar los resultados que arrojen de su medición semanalmente para las métricas de calidad del defecto reparado y métrica de seguimiento del valor de costo y mensualmente para la de problemas de usuario reportados, e ir haciendo los ajustes y tomar acciones correctivas para ir mejorando en las siguientes etapas del desarrollo.

Posteriormente una vez que se haya madurado la medición de estas métricas se podrán ir añadiendo otras sin que esto cause una sobrecarga de trabajo para los ingenieros de software.

BIBLIOGRAFÍA

Adkins, G., Y. Liu, Yao, J.F, y G. Williams. 2007. Studying software metrics based on real-world. Pag 55. Extraído el 04 de Octubre de 2008 de la base de datos ACM.

Barra, P. C. n.d. Software e ingeniería de software. Extraído el 02 de Octubre de 2008 desde: http://www.revistamarina.cl/revistas/1998/1/barra.pdf

Barrachina, S. y R. Mayo n.d.. Tema 2 Funcionamiento del computador 1. Programación. Extraído el 30 de Septiembre de 2008 desde: http://lorca.act.uji.es/ig09/transparencias/2_1_programacion_p_2up.pdf

Freeman, P., L. Johnson, C. McClure, L. Tripp, D. Robinson, J. Ellis, W. Scachi, B. Scheff, F. Van Den Bosch, y A.Von Staa 1982. Evaluation of software development life cycle: methodology implementation, Software Engineering Notes, 46-48. Extraído el 04 de Octubre de 2008 de la base de datos ACM.

H. Kan, S. 2004. Metrics and models in software quality engineering 2nd edition, pags. 86-117

Institute for software research, universidad de California Irvine 2005. Understanding and improving software productivity. Extraído el 15 de Septiembre del 2008 desde: http://www.ics.uci.edu/~wscacchi/

Kerlinger, F.N. y H. B. Lee, 2002 "Investigación de comportamiento. Métodos de investigación en ciencias sociales", 4ta. Edición. Editorial Mc Graw Hil, México.

McCabe, J. T. 1976. IEEE Transactions on Software Engineering, Vol. se-2 (A Complexity Measure), pags. 308-320.

Real Academia Española. Extraído el 04 de Octubre de 2008 desde http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=software

Scacchi, W. 1995. Understanding Software Productivity. Extraído el 05 de Octubre del 2008 desde: http://cwis.usc.edu/dept/ATRIUM/Papers/Software_Productivity.html

Smith, D. A. 1988. Software development productivity tools and metrics, pags. 44-47. Extraído el 05 de Octubre del 2008 de la base de datos ACM.

Vosburg, J., B. C., R. Wolverton, B. A., H. Malec, S. Hoben, y Y. Liu, `Productivity Factors and Programming Environments', Proc. 7th. Intern. Conf. Soft. Engr., IEEE Computer Society 1984, pags. 143-152

APÉNDICE

APÉNDICE 1

Puesto:_____ Antigüedad en la empresa:_____
Antigüedad en el área de SW: _____ Área: _____

Jerarquiza las siguientes 26 métricas de acuerdo al grado de importancia según tu criterio:

___ **Frecuencia y distribución de cambios de requisitos** (Que tan rápido cambian los requisitos durante las etapas de desarrollo)

___ **Complejidad del diseño** (tasa componentes nuevos y reutilizados)

___ **Número de potenciales defectos de diseño detectados y removidos antes de codificar**

___ **Esfuerzo para codificar los módulos diseñados** (tasa de inconsistencias encontradas entre el diseño del modulo y la implementación de los codificadores)

___ **Densidad de defectos** (No. De defectos / No. De líneas de código)

___ **Líneas de código** (tamaño del proyecto)

___ **Puntos de función** (un punto de función es una medida compuesta de un número de atributos del programa, incluyendo el número de entradas, salidas, llamadas a función, acceso a archivos, etc., que son multiplicados por unos factores de peso)

___ **Complejidad ciclomática** (proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el numero de caminos independientes del conjunto básico de un programa y nos da un limite superior para el numero de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez)

___ **Problemas reportados por el usuario** (No. De problemas totales que el usuario reporto en un periodo de tiempo)

___ **Patrón removedor de defectos basado en fase** (dar seguimiento de los defectos en todas las fases del ciclo de desarrollo, incluyendo a las revisiones de diseño, inspecciones de código y verificaciones formales antes de las pruebas)

___ **Densidad de defectos durante pruebas formales**

___ **Efectividad para remover defectos** (defectos removidos durante una fase de desarrollo / Defectos latentes en el producto * 100 %)

___ **Calidad del defecto reparado** (número de defectos introducidos después de haber reparado un defecto)

___ **Carga de trabajo ajustada e índice de administración de carga de trabajo** (es una métrica relacionada con el mantenimiento y esta relacionada a la tasa de llegada de los defectos y la tasa de disponibilidad para arreglar los defectos. El índice de carga de trabajo se refiere a los problemas abiertos o no resueltos, esta dado por: No. Problemas cerrados durante el mes / No, de problemas recibidos durante el mes)

___ **Tiempo de respuesta para arreglar defectos**

___ **Porcentaje de reparaciones delincuentes** (No. De reparaciones que excedieron el criterio de tiempo de respuesta por nivel de severidad / No. De reparaciones entregadas en un tiempo especifico * 100)

___ **Fase de ciclo de vida y métrica de seguimiento a programa o itinerario** (dar seguimiento al programa basado en fases del ciclo de vida comparar lo actual con lo planeado)

___ **Seguimiento de valor de costo** (comparar el costo actual del proyecto contra el costo presupuestado)
___ **Seguimiento a diseño** (dar seguimiento al número de requisitos implementados en el diseño contra el número de requisitos escritos)
___ **Seguimiento por tipos de falla**
___ **Promedio de defectos reparados por día**
___ **Promedio de horas de ingeniería / número de defectos reparados**
___ **Promedio de defectos reportados por día**
___ **Ramificaciones cubiertas entre número total de ramificaciones** (cuando se ejecuta un programa indica el número de puntos de decisión que fueron ejecutados)
___ **Porcentaje de tiempo extra** (promedio de tiempo extra / No. De horas de trabajo a la semana)
___ **Número de veces que aborta el sistema o se queda colgado en pruebas de estrés o de sistema**

Si piensas que hay más métricas que hay que seria conveniente tomar en cuenta por favor descríbelas:_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

APÉNDICE 2

Puesto:_____ Antigüedad en la empresa:_____
Antigüedad en el área de SW: _____ Área: _____

Jerarquiza las siguientes métricas (del 1 al 11) de acuerdo al grado de importancia (1 mayor prioridad, 11 la de menor prioridad) según tu criterio:

MÉTRICAS DE CALIDAD

CALIDAD EN EL PRODUCTO

___***Densidad de defectos*** = No. De defectos / No. De líneas de código

___***Problemas de usuario por mes*** = No. De problemas totales que el usuario reporto en un periodo de tiempo (pueden ser problemas relacionados con la funcionalidad del programa o la documentación)

___***Complejidad ciclomática*** (proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el numero de caminos independientes del conjunto básico de un programa y nos da un limite superior para el numero de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Esta puede utilizarse en varias áreas incluyendo:
Análisis de riesgo en desarrollo de código
Análisis de riesgo de cambio durante la fase de mantenimiento
Planificación de Pruebas
Reingeniería)

___ ***Puntos de función*** (un punto de función es una medida compuesta de un número de atributos del programa, incluyendo el número de entradas, salidas, llamadas a función, acceso a archivos, etc., que son multiplicados por unos factores de peso)

CALIDAD EN EL PROCESO

___***Patrón removedor de defectos basado en fase*** (dar seguimiento de los defectos en todas las fases del ciclo de desarrollo, incluyendo a las revisiones de diseño, inspecciones de código y verificaciones formales, esta métrica nos podría dar una idea de en que fase del producto se están introduciendo más errores)

MÉTRICAS DE SERVICIO

___***Índice de administración de carga de trabajo*** = No. Problemas cerrados durante el mes / No. de problemas recibidos durante el mes. (Los problemas que quedaron pendientes pueden ser considerados como número de problemas recibidos para el mes siguiente)

___***Tiempo de respuesta para arreglar defectos*** (puede ser considerado por ingeniero, de acuerdo a su experiencia y complejidad del problema)

___**Porcentaje de reparaciones delincuentes** = (No. De reparaciones que excedieron el criterio de tiempo de respuesta / No. De reparaciones entregadas en un tiempo especifico) * 100

___***Calidad del defecto reparado*** (número de defectos introducidos después de haber reparado un defecto)

MÉTRICAS DE COSTO

___***Métrica de seguimiento de valor de costo***: Comparar el costo actual del proyecto contra el costo presupuestado.

___***Porcentaje de tiempo extra***: Promedio de tiempo extra / No. De horas de trabajo a la semana

Si tienes algún comentario en cuanto a las métricas aquí propuestas o piensas que hay otras métricas que deben considerarse por favor exprésalo en las siguientes líneas:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____