CAPÍTULO II

2 MARCO TEÓRICO

2.1 INTRODUCCIÓN

En este capítulo, se explican los fundamentos y las bases para la aplicación y el uso de la firma digital. Cubre los conceptos de infraestructura de claves publicas (PKI), que apoya el uso de la criptografía de clave pública en entornos abiertos, firmas electrónicas como formatos que definen la estructura y la información de la tecnología de firmas digitales, las políticas de firma electrónica , que permiten establecer los requisitos para una firma que se considerará válida en un contexto de transacción en particular, y una breve reseña de la legislación nacional.

Al ser un sistema que se desarrollara con la metodología de desarrollo ágil denominado XP (Programación Extrema), mismo que interactuará con el modelo UML (Lenguaje unificado de Modelado), se debe entender la estructura general que tienen estos, los cuales utilizaremos para la solución de problemas.

2.2 CRIPTOGRAFÍA

La criptografía es la creación de técnicas para el cifrado de datos. Teniendo como objetivo conseguir la confidencialidad de los mensajes. Si la criptografía es la creación de mecanismos para cifrar datos, el criptoanálisis son los métodos para “romper” estos mecanismos y obtener la información.

Una vez que nuestros datos han pasado un proceso criptográfico decimos que la información se encuentra cifrada.

La finalidad de la criptografía es, en primer lugar, garantizar el secreto en la comunicación entre dos entidades (personas, organizaciones, etc.) y, en segundo lugar, asegurar que la información que se envía es auténtica en un doble sentido: que el remitente sea realmente quien dice ser y que el contenido del mensaje enviado, habitualmente denominado

criptograma, no haya sido modificado en su tránsito. Se pueden distinguir tres tipos de criptografía: simétrica, asimétrica e híbrida.

2.2.1 Criptografía simétrica

La criptografía Simétrica es un método criptográfico mono-clave, esto quiere decir que se usa la misma clave para cifrar y descifrar. Esto supone un grave problema a la hora de realizar el intercambio entre el emisor y el receptor, dado que si una tercera persona estuviese escuchando el canal podría capturar la clave, siendo inútil el cifrado.

Es importante que la clave sea difícil de adivinar y el método de cifrado empleado sea adecuado. Hoy en día, con la capacidad computacional disponible, si se emplean los algoritmos adecuados, dependiendo del método de cifrado empleado se puede obtener una clave en cuestión de tiempo reducida. Algunos ejemplos de algoritmos simétricos son 3DES, AES, Blowfish e IDEA.

2.2.2 Criptografía asimétrica

La criptografía asimétrica, también conocida como de clave pública es un sistema que emplea una pareja de claves. Esta pareja de claves pertenecen a la misma persona. Una es de dominio público y cualquiera puede tenerla y la otra es privada. El funcionamiento de este sistema es el siguiente: El remitente usa la clave pública del destinatario y sólo con la clave privada se podrá descifrar el mensaje. De esta forma se consigue que sólo el destinatario pueda acceder a la información.

De la misma forma si el propietario usa su clave privada para cifrar un mensaje sólo se podrá descifrar con la clave pública. La mayor ventaja de este sistema es que la distribución de claves es más fácil y segura que usando clave simétrica. Algunos ejemplos de algoritmos asimétricos son: Diffie-Hellman, RSA, DSA, ElGamal, Criptografía de curva elíptica.

El más extendido de los sistemas de clave pública es el RSA, que fue desarrollado por Rivest, Shamir y Adleman, este algoritmo se basa en escoger dos números primos grandes elegidos de forma aleatoria y mantenidos en secreto. La principal ventaja de este algoritmo desde el punto de vista de seguridad radica en la dificultad a la hora de factorizar números

grandes. RSA es reversible, es decir, además de permitir cifrar con la clave pública y descifrar con la privada, permite cifrar con la clave privada y descifrar con la clave pública.

En la normativa Boliviana se usa la criptografía asimétrica, usando el algoritmo RSA para la generación de la clave publica como privada. Esto esta regulado por la Autoridad de Regulación y Fiscalización de Telecomunicaciones y transporte (ATT).

2.2.3 Criptografía híbrida

Este tipo de criptografía utiliza tanto el cifrado simétrico como el asimétrico. Emplea el cifrado de clave pública para compartir una clave para el cifrado simétrico. El mensaje que se envía en el momento, se cifra usando la clave única (cifrado asimétrico) y se envía al destinatario. Tanto PGP como GnuPG usan sistemas de cifrado híbridos.

2.3 CERTIFICADOS

2.3.1 Certificado digital

Los sistemas de control de acceso basados en criptografía utilizan un concentrado de información denominado, por Kohnfelder, certificado digital[1], que se usa para demostrar la identidad y los atributos de su poseedor antes de permitirle el acceso a un sistema en Internet.

El objetivo principal de un certificado digital es restringir el acceso a un sistema basado en un proceso de autorización para evitar la suplantación de un usuario. Un certificado digital permite también detectar si una transacción ha sido alterada durante la transmisión, consiguiendo de este modo garantizar la integridad de un mensaje[2].

2.3.2 Tipos de certificados

2.3.2.1 Certificados de identidad

Dos entidades que poseen claves privadas y que desean intercambiar datos con

1 Stefan A. Brands. Rethinking Public Key Infrastructures and Digital Certificates. MIT Press, Cambridge, Massachusetts, August 2000.

2 Talens-Oliag, Sergio. Introducción a los certificados digitales. http://www.uv.es/~sto/articulos/BEI-2003-11/certificados_digitales.html

confianza mediante un medio no fiable pueden asociar esas claves con una clave pública e integrarla en un certificado digital de identidad. Los certificados de identidad son estructuras de datos que tienen un contenido datos usado para reconocer a un sujeto (persona, objeto o máquina) y tienen la propiedad de conectar una entidad con su clave pública[3].

Los certificados de clave pública son emitidos por autoridades de certificación (AC) y representan una evidencia que asegura el vínculo (pertenencia) de la clave pública con los datos de identidad declarados en el mismo, evidencia que puede ser demostrada (probada) mediante un proceso de verificación técnica que consiste en la presentación de una clave privada o una afirmación hecha por el sujeto. Las autoridades de certificación son organizaciones seguras que administran las firmas digitales de clave pública y proporcionan servicios de consulta. Estos servicios permiten la verificación de firmas para asegurar que una entidad sea considerada legítima o no niegue su identidad (X.509).

Los certificados de identidad de clave pública son utilizados en un proceso de control de acceso para legitimar a su propietario (autentificación). La distribución de las claves públicas y los certificados requieren de una infraestructura denominada de clave pública (Public Key Infraestructure, PKI). La ITU mediante el estándar X.509 define y describe esta forma de administración de claves.

2.3.2.2 Certificados de atributo

El proceso de control de acceso puede usar la información contenida en estructuras de datos, denominados certificados de atributo, no sólo para comprobar la identidad de un sujeto sino también sus roles. Los certificados de atributo tienen una estructura de datos similar a la de un certificado de identidad[4]. La diferencia está en que los certificados de atributo no contienen una clave pública, en lugar de ella incluyen atributos que especifican

3 Mavridis, Ioannis; Georgiadis, Christos; Pangalos, George; Khair, Marie. Access Control based on Attribute Certificates for Medical Intranet Applications. Aristotle University of Thessaloniki, Greece.

4 Farrell, S. and R. Housley. An Internet Attribute Certificate Profile for Authorization. Internet Draft draft-ietf-pkix-ac509prof-06, January 2001.

información de control de acceso asociado con el poseedor del certificado. En el proceso de autorización las decisiones no sólo se basan en la verificación de identidad sino también en la verificación de roles, reglas y control de acceso basado en el rango. Los certificados de atributo permiten asociar información de identidad con información de autorización que no es de identidad[5].

Esta forma de control de acceso permite restringir de acuerdo al perfil de los usuarios y así agregar a los sistemas mayor grado de fiabilidad. La información de control de acceso puede utilizarse en un proceso de autorización para validar dinámicamente un certificado y prescindir de la revocación de certificados manejando cortos períodos de vida de un certificado.

Las autoridades de atributos son entidades responsables de emitir certificados de atributo al igual que las autoridades de certificación de certificados de clave pública.

2.3.2.3 Otros tipos de certificado

Los sistemas basados en la identidad son una opción pero no una solución al problema de dar confianza, existen otras propiedades además de la identidad (edad, dirección, nacionalidad, estado civil y otros) que son relevantes para establecer confianza entre las partes involucradas. Estos son los sistemas de credencial digital[6]. Stefan Brands introduce el concepto de credenciales digitales como certificados de atributo de privacidad-mejorada[7]. Considerando que las infraestructuras de certificados de clave pública ignoran la privacidad de la identidad de las personas, Zero-Knowledge Systems en noviembre de 2000 publicó su visión de las credenciales privadas. También existen otros modelos conceptuales que son SPKI (Simple Public Key Infraestructure) y PGP (Pretty God Privacy). Una comparación de sistemas de certificación se presenta en el trabajo de E. Gerck, quien afirma que los métodos de certificación absoluta son lógicamente imposibles, porque un certificado no

5 Mavridis, Ioannis; Georgiadis, Christos; Pangalos, George; Khair, Marie. Access Control based on Attribute Certificates for Medical Intranet Applications. Aristotle University of Thessaloniki, Greece.

6 Seamons, Kent E. Using Digital Credentials to Establish Trust between Strangers. http://isrl.cs.byu.edu/pres/seamons.CERT1999.pdf

7 Stefan A. Brands. Rethinking Public Key Infrastructures and Digital Certificates. MIT Press, Cambridge, Massachusetts, August 2000.

puede certificarse así mismo[8].

2.3.3 Autoridad de certificación

"Una autoridad de certificación (AC) es definida como una autoridad que ha recibido confianza de uno o más usuarios para crear y asignar certificados" [X.509]. Las AC tienen la facultad de certificar la correspondencia entre una entidad y una clave pública. Sin embargo, semánticamente una AC no es capaz de denotarla [3]. La AC se constituye en la tercera parte confiable, frente a las entidades que se comunican (emisor y receptor). Entre las autoridades de certificación más conocidas se tienen a: Verisign, Thawte, GeoTrust, RapidSSL y DigiCertSSL.

Todas las Autoridades de Certificación deben mantener una base de datos de nombres distinguidos (ND) para usuarios o AC subordinadas y tomar las medidas para asegurar que ninguna autoridad emita duplicados de ND. Las funciones más importantes que realizan las autoridades de certificación son:

- Registro de usuarios: tienen la responsabilidad de gestionar la información de identidad de los usuarios.
- Emisión de certificados: deben generar los certificados que enlacen a un usuario con una clave pública.
- Administración de certificados: además de registrar deben controlar atributos de los certificados para tomar decisiones de revocación, renovación y suspensión.
- Servicio de consulta: deben ofrecer servicios a los usuarios para facilitar el seguimiento sobre el estado de los certificados.
- Administración de las firmas: deben ofrecer mecanismos para la generación de claves usando algoritmos de cifrado de mensajes.

8 Gerck, E. MCG. Overview of Certification Systems: X.509, CA, PGP and SKIP. http://www.mcg.org.br/cert.htm (verificado Abril 1997)

2.3.3.1 Jerarquía de certificación

La jerarquía de autoridades de certificación se define en el documento RFC 1422[9]. Este estándar establece una estructura jerárquica rígida de AC. En la estructura se definen tres tipos de autoridades de certificación:

- Internet Policy Registration Authority (IPRA): Esta autoridad es la más alta (raíz) de la jerarquía de certificación PEM. La actuación de esta autoridad es a nivel 1 y sólo se le está permitido emitir certificados para el siguiente nivel de autoridad (PCA). Todo proceso de certificación comienza en una autoridad IPRA.
- Policy Certification Authorities (PCA): las autoridades PCA actúan a nivel 2 de la jerarquía. Cada autoridad PCA debe estar certificada por una autoridad IPRA. Una autoridad PCA debe establecer y declarar su política respecto a los usuarios o subautoridades de certificación. Está permitida la existencia de distintas autoridades PCA para responder necesidades específicas de los usuarios. En el caso boliviano este rol lo cumplirá la Autoridad de Regulación y Fiscalización de Telecomunicaciones y Transportes (ATT).
- Certification Authorities (CA). las autoridades CA están ubicadas a nivel 3 y pueden funcionar a niveles inferiores. Las autoridades que están a nivel 3 tienen que recibir la certificación de una autoridad del nivel 2. En el caso boliviano la entidad certificadora sera la Agencia para el Desarrollo de la Sociedad de la Información en Bolivia (ADSIB).

Una regla de designación de nombres también está definida en RFC 1422, además, ésta establece que una autoridad CA sólo puede emitir certificados para entidades cuyos nombres se subordinan al nombre de la misma autoridad CA. A partir de esta regla se puede hacer un seguimiento de encadenamiento de autoridades de certificación.

9 Administración de claves basada en certificados: Mejoramiento de Privacidad de Internet de correo electrónico: Parte II

2.3.3.2 CACert

CAcert.org es una Autoridad de certificación administrada por una comunidad que otorga gratuitamente certificados de clave pública. Estos certificados pueden ser usados para firmar y cifrar correo electrónico, identificar y autorizar usuarios conectados a sitios web y transmisión segura de datos en Internet. Cualquier aplicación que soporte Secure Socket Layer (SSL) puede usar certificados firmados por CAcert, tal como lo puede hacer cualquier aplicación que use certificados X.509, por ejemplo para cifrar o firmar documentos digitalmente.

El procedimiento de expedición de certificados es muy riguroso en cuanto a la comprobación de documentos de identidad, y exige apersonación ante más de un agente de verificación de identidad que tiene funciones de Autoridad de Registro.

2.3.4 Revocación de certificados

Después de la emisión de un certificado por parte de una autoridad de certificación, es posible que se haya puesto en peligro la clave privada del titular del certificado o que se haya utilizado información falsa para solicitar el certificado. En estos y otros casos surge la necesidad de dar a las autoridades de certificación la facultad de retirar un certificado ya emitido.

2.3.4.1 Listas de revocacion de certificado

Las listas de revocación de certificados (CRL) son un mecanismo mediante el cual la CA publica y distribuye información acerca de los certificados anulados a las aplicaciones que los emplean. Una CRL es una estructura de datos firmada por la CA que contiene la fecha y hora de su publicación, el nombre de la entidad certificadora y los números de serie de los certificados anulados que aún no han expirado. Cuando una aplicación trabaja con certificados debe obtener la última CRL de la entidad que firma el certificado que está empleando y comprobar que su número de serie no está incluido en dicha lista.

Existen varios métodos para la actualización de CRLs:

- Muestreo de CRLs. Las aplicaciones acceden a la CA o a los almacenes de archivos y copian el último CRL en intervalos regulares.
- Anuncio de CRLs. La entidad certificadora anuncia que ha habido un cambio en el CRL a las aplicaciones. El problema de este enfoque es que el anuncio puede ser muy costoso y no se sabe qué aplicaciones deben ser informadas.
- Verificación en línea. Una aplicación hace una consulta en línea a la CA para determinar el estado de revocación de un certificado. Es el mejor método para las aplicaciones, pero es muy costoso para la CA

2.3.4.2 Protocolo de estado de certificados en linea

Es un método para determinar el estado de revocación de un certificado digital X.509 usando otros medios que no sean el uso de CRL. Este protocolo se describe en el RFC 2560[10] y está en el registro de estándares de Internet.

OCSP fue creado para solventar ciertas deficiencias de las CRL. Cuando se despliega una PKI (Infraestructura de Clave Pública), es preferible la validación de los certificados mediante OCSP sobre el uso de CRL por varias razones:

- OCSP puede proporcionar una información más adecuada y reciente del estado de revocación de un certificado.
- OCSP elimina la necesidad de que los clientes tengan que obtener y procesar las CRL, ahorrando de este modo tráfico de red y procesado por parte del cliente.
- El contenido de las CRL puede considerarse información sensible, análogamente a la lista de morosos de un banco.
- Un "OCSP responder" puede implementar mecanismos de tarificación para pasarle el coste de la validación de las transacciones al vendedor, más bien que al cliente.

10 Infraestructura de clave Pública de Internet para Online Certificate Status Protocol

- OCSP soporta el encadenamiento de confianza de las peticiones OCSP entre los "responders". Esto permite que los clientes se comuniquen con un "responder" de confianza para lanzar una petición a una autoridad de certificación alternativa dentro de la misma PKI.
- Una consulta sobre el estado de un certificado sobre una CRL, debe recorrerla completa secuencialmente para decir si es válido o no. Un "OCSP responder" en el fondo, usa un motor de base de datos para consultar el estado del certificado solicitado, con todas las ventajas y estructura para facilitar las consultas. Esto se manifiesta aún más cuando el tamaño de la CRL es muy grande.

2.3.5 Tipos de certificados de clave publica

Existen cuatro tipos de certificados de clave pública: certificados de autoridad, certificados de servidor, certificados de usuario (personales) y certificados de productores de software:

- Certificados de autoridad. Las entidades emisoras de certificados raíz tienen la capacidad de asignar certificados a certificados de autoridad. Corresponden a entidades que certifican. Los certificados raíz son los únicos auto-firmados y son los que inician una cadena de certificación de acuerdo a la jerarquía definida en el estándar X.509.
- Certificado de servidor. Certifica que un servidor es de la empresa que dice ser y que el identificador del servidor es correcto. Los certificados de servidor identifican a servidores que participan en comunicaciones seguras con otros equipos mediante la utilización de protocolos de comunicaciones. Estos certificados permiten al servidor probar su identidad ante los clientes.
- Certificados personales. Los certificados personales aseguran que una dirección de correo y clave pública corresponden a una persona. Estos certificados identifican a personas y se pueden utilizar para autenticar usuarios con un servidor.
- Certificados de productores de software. Se utilizan para "firmar" el software y

asegurar que no ha sido modificado. Esto no implica que se pueda ejecutar con seguridad, pero informa al usuario que el fabricante de software participa en la infraestructura de compañías y entidades emisoras de certificados de confianza. Estos certificados se utilizan para firmar el software que se distribuye por Internet.

2.3.5.1 Componentes de un certificado de clave publica

Los componentes de un certificado X.509 son: el descriptor del certificado, la firma digital y un valor de firma. Los elementos del descriptor son:

- Versión. Contiene el número de versión del certificado codificado. Los valores aceptables son 1, 2 y 3.
- Número de serie. Es un entero asignado por la autoridad certificadora. Cada certificado emitido por una CA debe tener un número de serie único.
- Identificador del algoritmo de firmado. Identifica el algoritmo empleado para firmar el certificado. Nombre del emisor. Identifica la CA que ha firmado y emitido el certificado.
- Periodo de validez. Indica el periodo de tiempo durante el cual el certificado es válido. Nombre del sujeto. Identifica el nombre del usuario para el que se emite el certificado.
- Nombre del sujeto. Indica el nombre del usuario para el cual se emite el certificado.
- Información de clave pública del sujeto. Información de la clave pública del usuario para el que se emite el certificado (nombre, algoritmo, etc.).
- Identificador único del emisor. Es un campo opcional que permite reutilizar nombres de emisor.
- Identificador único del sujeto. Es un campo opcional que permite reutilizar nombres de sujeto.
- Extensiones. Otros campos específicos de cada protocolo que están sujetos a sus

propias regulaciones.

Los componentes de un certificado emitido por una CA en Bolivia esta establecida por la ATT, que contienen los mismos elementos mencionados.

2.3.5.2 Propiedades de los certificados de clave publica

Las características más importantes de los certificados digitales son:

- **Autenticación**. Para el receptor de un documento, la autenticación implica asegurar que los datos recibidos han sido enviados por quien declara ser poseedor de la identidad contenida en la firma digital.
- **Confidencialidad**. La confidencialidad implica asegurar información enviada no podrá ser interceptada por terceros.
- **Integridad**. La integridad de los documentos implica tanto para el remitente como para el destinatario asegurar que la información enviada no será modificada por terceros.
- **Privacidad**. La privacidad de los mensajes implica que los datos sólo podrán ser leídos por el destinatario por contener elementos cifrados.
- **No repudio**. El no repudio implica para el receptor de un mensaje asegurar que el emisor no negará haber enviado la información recibida.

a) Autenticación

La autenticación de claves asimétricas permite que un mensaje cifrado con una clave privada sólo pueda haber sido enviado por el propietario de la misma.

b) Confidencialidad

Para lograr la confidencialidad, el remitente (emisor) de un mensaje debe cifrarlo con la clave pública del destinatario (receptor), que puede obtenerse de su Certificado Digital. De esta forma el emisor se asegura que el mensaje sólo podrá ser descifrado con la clave privada del receptor, es decir, sólo podrá ser leído por el destinatario.

c) Integridad

Para lograr la confidencialidad, el remitente (emisor) de un mensaje debe cifrarlo con la clave pública del destinatario (receptor), que puede obtenerse de su Certificado Digital. De esta forma el emisor se asegura que el mensaje sólo podrá ser descifrado con la clave privada del receptor, es decir, sólo podrá ser leído por el destinatario.

d) No repudio

También como consecuencia directa del concepto de firma digital, la sola existencia del mensaje "firmado" por su clave privada, una vez comprobada su integridad, impide al emisor el repudio del mensaje, ya que el mismo no podría haberse generado por otra vía. El receptor conserva el documento firmado como comprobante de la operación.

2.4 SELLADO DE TIEMPO (TSA)

El sellado de tiempo (TSA, Timestamping Authority). es un método para probar que un conjunto de datos existió antes de un momento dado y que ninguno de estos datos ha sido modificado desde entonces. El sellado de tiempo proporciona un valor añadido a la utilización de firma digital ya que ésta por sí sola no proporciona ninguna información acerca del momento de creación de la firma, y en el caso de que el firmante la incluyese, ésta habría sido proporcionada por una de las partes, cuando lo recomendable es que la marca de tiempo sea proporcionada por una tercera parte de confianza.

2.5 FUNCIÓN HASH

Las funciones criptográficas hash juegan un papel fundamental en la criptografía moderna, hay muchas funciones hash, comúnmente usadas en aplicaciones no criptográficas.

Una función hash, o función resumen, toma un mensaje como entrada y produce una salida que llamamos resultado hash. La idea básica de las funciones criptográficas hash es que los valores hash obtenidos con ellas sirven como una imagen representativa y compactada de una cadena de entrada, y pueden usarse como un posible identificador único de esa cadena de entrada: ese valor hash obtenido del mensaje de entrada suele llamarse resumen del

mensaje o huella digital del mensaje.

Las funciones hash se emplean en criptografía junto con los criptosistemas de firma digital para otorgar integridad a los datos. A la hora de firmar digitalmente un documento o mensaje, es práctica habitual hacer la firma sobre la huella digital del mensaje y no sobre la totalidad del mensaje a firmar.

Los algoritmos has mas utilizados son:

- MD5 (Message Digest; en castellano, Resumen de mensaje), el MD5 crea, a partir de un texto cuyo tamaño es elegido al azar, una huella digital de 128 bits procesándola en bloques de 512 bits. Es común observar documentos descargados de Internet que vienen acompañados por archivos MD5: este es el hash del documento que hace posible verificar su integridad.
- SHA (Secure Hash Algorithm; en castellano, Algoritmo Hash Seguro), crea una huella digital que tiene 160 bits de longitud. SHA-1 es una versión mejorada de SHA que produce una huella digital de 160 bits a partir de un mensaje que tiene una longitud máxima de 264 bits y los procesa en bloques de 512 bits. Por otra parte el SHA-2 es un conjunto de funciones hash criptográficas (SHA-224, SHA-256, SHA-384, SHA-512), SHA-2 incluye un significante número de cambios respecto a su predecesor, SHA-1; y consiste en un conjunto de cuatro funciones hash de 224, 256, 384 o 512 bits.

2.6 FIRMA DIGITAL

Podemos definirlo de la siguiente forma:

“Es la firma electrónica que identifica únicamente a su titular, creada por métodos que se encuentren bajo el absoluto y exclusivo control de su titular, susceptible de verificación y está vinculada a los datos del documento digital de modo tal que cualquier modificación de los mismos ponga en evidencia su alteración.”[11]

11 Articulo 6, párrafo 4, definición 5, Ley General de telecomunicaciones, tecnologías de información y comunicación

De esta definición podemos decir que firma digital es el resultado de aplicar cierto algoritmos matemáticos sobre documentos digitales, que permiten garantizar la identidad del firmante así como la integridad de la información, la confidencialidad de los datos y el no repudio de la información.

La firma digital debe ser avanzada, cumpliendo con los siguientes requisitos:

- requerir información de exclusivo conocimiento del firmante, permitiendo su identificación unívoca;
- ser creada por medios que el firmante pueda mantener bajo su exclusivo control;
- ser susceptible de verificación por terceros;
- estar vinculada a un documento electrónico de tal modo que cualquier alteración subsiguiente en el mismo sea detectable; y
- haber sido creada utilizando un dispositivo de creación de firma técnicamente seguro y confiable y estar basada en un certificado reconocido válido al momento de la firma “.

Por otra parte, el software de firma digital debe efectuar varias validaciones, entre las cuales podemos mencionar:

- Vigencia del certificado digital del firmante.
- Revocación del certificado digital del firmante.
- Sello de tiempo.

2.6.1 Análisis comparativo entre firma manuscrita y firma digital

Analizaremos las diferentes características que habíamos establecido para la firma manuscrita a los efectos de establecer que aplicando el principio de Derecho Informático de equivalencia funcional, la firma electrónica cumple la misma función .

a) Suplantación; Cuando trabajamos en el medio electrónico, en especial en Internet,

existe un alto riesgo de que la persona con la que interactuamos no sea quien dice ser, por lo que es imprescindible verificar la autenticidad y la garantía de origen en los entornos electrónicos. Esta verificación de capacidad del firmante, en el medio electrónico, lo hace el prestador de servicio de certificación, quien garantizara dicha información.

b) Alteración; Los documentos electrónicos y la información contenida en ellos, por lo general, son susceptibles de ser modificados o alterados al viajar en la red. Esto trae como consecuencia que la integridad del documento electrónico puede verse comprometida. Esta alteración, si el documento tiene firma digital avanzada, es detectada técnicamente, por lo que el receptor del mismo recibirá el documento firmado y un aviso de su modificación.

c) Pérdida de confidencialidad; Este atributo implica que la información sólo sea compartida entre las personas u organizaciones autorizadas. La no pérdida de la confidencialidad es un atributo imprescindible en las comunicaciones electrónicas con importantes efectos jurídicos y legales.

d) Rechazo o no repudio; Es el riesgo jurídico de rechazo de la autoría o de la integridad de información trasmitida por medio electrónicos. Una vez firmado el documento electrónicamente, quien lo hace, no puede rechazar su autoría.

e) Conflictos en la fecha y hora; La fecha y hora en la generación, envío y recepción de información electrónica, juegan un papel de importancia en materia probatoria. Dejarlo liberado a la información del equipo en el cual se está trabajando podría determinar un error en la misma, ya sea porque la fecha es errónea o por motivos de modificación de la misma. De allí que la solución a este problema es el sellado de tiempo, con lo cual la fecha y hora de expedición del documento no se podrán modificar.

2.6.2 Formato de firma digital

Las normas TS 102 778[12] definidas por la ETSI (European Telecommunications Standards Institute) definen los formatos técnicos de la firma electrónica. PKCS (Public-Key Cryptography Standards) se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California.

- El PKCS#7 es una estándar publicada como RFC 2315[13] que describe la sintaxis de encapsulación para la protección de datos. Permiten incluir firmas de diferentes firmantes mediante dos modalidades: encadenada y mancomunada. La firma propiamente dicha se compone de datos formales referidos al tipo de firma, así como de distintos atributos, como el tipo de contenido, certificados, hash, fecha y hora, número de firmas.
- El PKCS#11 esta norma también se conoce como "Cryptoki", que es una fusión de "interfaz de señal criptográfica" y se pronuncia como "cripto-key". Es una API que define una interfaz genérica para tokens criptográficos, como los módulos de seguridad de hardware (HSM), llaves USB y tarjetas inteligentes.
- El PKCS#12 es una norma que define un formato de archivo utilizado para almacenar las claves privadas con el acompañamiento de certificados de clave pública, protegido con una clave simétrica basada en contraseña. En la práctica, estos archivos tienen la extensión .p12 o .pxf predecesor de PKCS12. Es utilizable como formato para el almacén de claves de Java.

2.7 INFRAESTRUCTURA DE CLAVE PÚBLICA (PKI)

Una infraestructura de clave pública (o, en inglés, PKI, Public Key Infrastructure) es una combinación de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con garantías de operaciones criptográficas como el cifrado, la firma digital o el no repudio de transacciones electrónicas.

12 PAdES (PDF Advanced Electronic Signature) perfila el soporte para firmas digitales del formato PDF 1.7 (ISO 32000-1)

13 Sintaxis de mensajes criptográficos versión 1.5

El término PKI se utiliza para referirse tanto a la autoridad de certificación y al resto de componentes, como para referirse, al uso de algoritmos de clave pública en comunicaciones electrónicas.

2.7.1 Componentes de una infraestructura de clave pública

Los componentes más habituales de una infraestructura de clave pública son:

- La autoridad de certificación (CA).
- La autoridad de registro (o, en inglés, RA, Registration Authority): es la responsable de verificar el enlace entre los certificados (concretamente, entre la clave pública del certificado) y la identidad de sus titulares.
- Los repositorios: son las estructuras encargadas de almacenar la información relativa a la PKI. Los dos repositorios más importantes son el repositorio de certificados y el repositorio de listas de revocación de certificados (CRL) o servidor OCSP (protocolo de estado de certificados en linea).
- La autoridad de validación (o, en inglés, VA, Validation Authority): es la encargada de comprobar la validez de los certificados digitales.
- La autoridad de sellado de tiempo (TSA).
- Los usuarios y entidades finales son aquellos que poseen un par de claves (pública y privada) y un certificado asociado a su clave pública. Utilizan un conjunto de aplicaciones que hacen uso de la tecnología PKI (para validar firmas digitales, cifrar documentos para otros usuarios, etc.)

2.8 INGENIERÍA DEL SOFTWARE

La ingeniería de software es una disciplina de la ingeniería que comprende todos los aspectos de producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza.

El proceso de Ingeniería del Software se basa en modelos, métodos y herramientas que

sirven como una guía para los desarrolladores de software durante el proceso de desarrollo, con la finalidad de mejorar la calidad de los proyectos, procesos y productos mediante la evaluación y medición de los mismos. El objetivo de las organizaciones desarrolladoras de estos modelos, procesos y metodologías es que en las empresas desarrolladoras de software se los ponga en práctica para ver las mejoras en los procesos de cada una de las fases de desarrollo. Otro tema importante son los modelos del ciclo de vida del software, los cuales se basan en diferentes técnicas y fases pero todos tienen un mismo fin.

La ingeniería de software no solo comprende los procesos técnicos de desarrollo de software, sino también con actividades tales como la gestión de proyectos de software y desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

2.8.1 Ingeniería

La ingeniería es el estudio y la aplicación de las distintas ramas de la tecnología, para la resolución de problemas que afectan a la actividad cotidiana de la sociedad.

La ingeniería también supone la aplicación de la inventiva y del ingenio para desarrollar una cierta actividad. Esto, por supuesto, no implica que no se utilice el método científico para llevar a cabo los planes. De esta forma la ingeniería es la actividad de transformar el conocimiento en algo práctico.

Otra característica que define a la ingeniería es la aplicación de los conocimientos científicos a la invención o perfeccionamiento de nuevas técnicas. Esta aplicación se caracteriza por usar el ingenio principalmente de una manera más pragmática y ágil que el método científico, puesto que la ingeniería, como actividad, está limitada al tiempo y recursos dados por el entorno en que ella se desenvuelve.

2.8.2 Software

Según la definición de la de IEE “Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.”

e considera que el software es el equipamiento lógico e intangible de un ordenador. En otras palabras, el concepto de software abarca a todas las aplicaciones informáticas, el software es desarrollado mediante distintos lenguajes de programación que permiten controlar el comportamiento de un dispositivo electrónico.

Los lenguajes de programación consisten en un conjunto de símbolos, reglas sintácticas y semánticas que definen el significado de sus elementos y expresiones, así mismo con los lenguajes de programación el desarrollador puede especificar de forma precisa, sobre que datos debe operar el dispositivo electrónico.

2.9 METODOLOGÍA DE DESARROLLO

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de un sistema de información. Una determinada metodología no es necesariamente aplicable a todo tipo de proyectos, al contrario cada tipo de proyecto tiene una metodología a la cual se adapta mejor.

Una metodología de desarrollo de software consiste en una filosofía de desarrollo de software con una base de procesos de desarrollo de software; múltiples herramientas, modelo y métodos para asistir en el proceso de desarrollo de software; suele estar documentada de una forma formal; suele estar promovida por algún tipo de organización publica o privada que promueve dicha metodología.

Cada metodología de desarrollo tiene tiene su enfoque de en lo que debería de consistir un proyecto de desarrollo de software, pero todas ellas se basan en una serie de enfoques generalistas como ser: lineal (Waterfall model), iterativo (prototyping), combinacion de iterativo y lineal (incremental y spiral) y iterativo (rapid application development).

2.9.1 Metodologías de desarrollo ágiles

De las metodologías tradicionales se genera un nuevo enfoque denominado métodos ágiles, que lograban permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos

por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Se creó The Agile Alliance3, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume la filosofía "ágil".

Los principios de este manifiesto son:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Entre las principales metodologías se encuentran el XP (eXtremeProgramming), Scrum, Iconix, Cristal Methods, AUP entre otras.

2.9.2 Comparación metodologías ágiles y tradicionales

Vamos a enumerar las principales diferencias de una Metodología Ágil respecto de las Metodologías Tradicionales. La Tabla 2.1 recoge estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Mas artefactos. El modelo es esencial, mantenimiento de modelos.
Pocos roles, mas genéricos y flexibles	Mas roles, mas específicos.
No existe un contrato tradicional, debe ser bastante flexible.	Existe un contrato prefijado.
Cliente es parte del equipo de desarrollo (ademas in-situ).	El cliente interactua con el equipo de desarrollo mediante reuniones.
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (menor de 10 integrantes) y trabajando en el mismo sitio.	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas usadas en proyectos grandes y con equipos posiblemente disperso.
La arquitectura se va definiendo y mejorando a lo largo del proyecto.	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo.	Énfasis en la definición del proceso , roles, actividades y artefactos.
Basadas en heurísticas provenientes de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Tabla 1: Tabla comparativa entre metodologías ágiles y metodologías tradicionales

2.10 METODOLOGÍA DE DESARROLLO XP (XTREM PROGRAMMING)

Es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en

realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP, sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea.

2.10.1 Fases de la metodología XP

Esta metodología se basa en la retro alimentación entre cliente y el equipo de desarrollo, con una comunicación fluida entre todos los participantes, simplicidad en la soluciones implementadas y audacia para enfrentar los cambios repentinos.

La metodología XP abarca reglas y practicas que están en el contexto de 4 fases de trabajo:

- Planificación
- Diseño
- Desarrollo
- Pruebas

2.10.1.1 Planificación

a) Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente

comprensible y delimitada para que los programadores puedan implementarla en unas semanas[14].

Respecto de la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción[15] o sólo una descripción[14], más quizás una estimación de esfuerzo en días[16].

No hay que preocuparse si en un principio no se identifican todas las historias de usuario. Al comienzo de cada iteración estarán registrados los cambios en las historias de usuario y según eso se planificará la siguiente iteración. Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración.

b) Plan de entregas

En este punto el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una

14 Jeffries, R., Anderson, A., Hendrickson, C. "Extreme Programming Installed". Addison-Wesley. 2001
15 Wake, W.C. "Extreme Programming Explored". Addison-Wesley. 2002.
16 Newkirk, J., Martin R.C. "Extreme Programming in Practice". Addison-Wesley. 2001.

fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

c) Velocidad del proyecto

Es una medida de capacidad que tienen el equipo de desarrollo para evacuar las historias de usuario en una determinada iteración. Esta medida se calcula totalizando el numero de historias de usuario realizadas en una iteración. Para la iteración siguiente se podrá implementar el mismo numero de historias de usuario que en la anterior iteración.

La velocidad de proyecto se usa para determinar cuantas historias de usuario pueden ser implementadas antes de una fecha dada, o cuanto tiempo es necesario para llevar a cabo un conjunto de historias de usuario. Cuando se realiza una planificación por alcance se divide el numero total de semanas entre la velocidad de proyecto para determinar cuantas iteraciones estarán disponibles.

d) Iteraciones

Esta etapa incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el

trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

e) Rotaciones

Las rotaciones evitaran que las personas se conviertan en si mismas en un cuello de botella. Las rotaciones permitirán que todo el mundo conozca como funciona el sistema.

f) Reuniones

Las reuniones son esenciales para cualquier tipo de metodología , es por ello que XP requiere una revisión continua del plan de trabajo a pesar de ser una metodología que evita la documentación exagerada, es muy estricta en la organización del trabajo.

2.10.1.2 Diseño

a) Metáfora del sistema

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema.

b) Tarjetas CRC

La principal funcionalidad que tienen las tarjetas CRC (Clase – Responsabilidad - Colaboración) es ayudar a dejar el pensamiento procedimental para incorporarse al enfoque orientado a objetos. Cada tarjeta representa una clase con su nombre en la parte superior, en

la sección inferior izquierda están descritas las responsabilidades y a la derecha las clases que le sirven de soporte.

En el proceso de diseñar el sistema por medio de las tarjetas CRC como máximo dos personas se ponen de pie adicionando o modificando las tarjetas, prestando atención a los mensajes que éstas se transmiten mientras los demás miembros del grupo que permanecen sentados, participan en la discusión obteniendo así lo que puede considerarse un diagrama de clases preliminar.

c) Soluciones puntuales

En muchas ocasiones los equipos de desarrollo se enfrentan a requerimientos de los clientes (en este caso historias de usuario) los cuales generan problemas desde el punto de vista del diseño o la implementación. Spike Solution, es una herramienta de XP para abordar este inconveniente.

Se trata de una pequeña aplicación completamente desconectada del proyecto con la cual se intenta explorar el problema y propone una solución potencial. Puede ser burda y simple, siempre que brinde la información suficiente para enfrentar el problema encontrado.

d) Funcionalidad mínima

Los desarrolladores tienden a predecir las necesidades futuras e implementarlas antes. Según mediciones, esta es una práctica ineficiente, concluyendo que tan solo el 10% de las soluciones para el futuro son utilizadas, desperdiciando tiempo de desarrollo y complicando el diseño innecesariamente.

En XP sólo se analiza lo que se desarrollará en la iteración actual, olvidando por completo cualquier necesidad que se pueda presentar en el futuro, lo que supone uno de los preceptos más radicales de la programación extrema.

e) Refactorización

Como se trató al principio de este apartado, el diseño es una tarea permanente durante toda la vida del proyecto y la refactorización concreta este concepto. Como en cualquier

metodología tradicional en XP se inicia el proceso de desarrollo con un diseño inicial. La diferencia es que en las metodologías tradicionales este diseño es tan global y completo como se es posible tomando generalmente mucho tiempo en lograrse y con la creencia de que si se ven forzados a modificarlo será un fracaso para el grupo de desarrollo. El caso de XP es el opuesto. Se parte de un diseño muy general y simple que no debe tardar en conseguirse, al cual se le hacen adiciones y correcciones a medida que el proyecto avanza, con el fin de mantenerlo tanto correcto como simple.

La refactorización en el código pretende conservarlo tan sencillo y fácil de mantener como sea posible. En cada inspección que se encuentre alguna redundancia, funcionalidad no necesaria o aspecto en general por corregir, se debe rehacer esa sección de código con el fin de lograr las metas de sencillez tanto en el código en sí mismo como en la lectura y mantenimiento.

Estas prácticas son difíciles de llevar a cabo cuando se está iniciando en XP por varios motivos. En primer lugar debido el temor que genera en los equipos de desarrollo cambiar algo que ya funciona bien sea a nivel de diseño o implementación. Sin embargo si se cuenta con un esquema de pruebas completo y un sistema de automatización para las mismas se tendrá éxito en el proceso. El otro motivo es la creencia que es más el tiempo que se pierde en refactoring que el ganado en sencillez y mantenimiento. Según XP la ganancia obtenida en refactoring es tan relevante que justifica suficientemente el esfuerzo extra en corrección de redundancias y funcionalidades innecesarias.

2.10.1.3 Codificación

La codificación es un proceso que se realiza en forma paralela con el diseño y la cual está sujeta a varias observaciones por parte de XP consideradas controversiales por algunos expertos tales como la rotación de los programadores o la programación en parejas. Además de los mencionados temas, describiremos los temas a continuacion:

a) **Cliente siempre presente.**

Uno de los requerimientos de XP es que el cliente esté siempre disponible. No solamente

para solucionar las dudas del grupo de desarrollo, debería ser parte de éste. En este sentido se convierte en gran ayuda al solucionar todas las dudas que puedan surgir, especialmente cara a cara, para garantizar que lo implementado cubre con las necesidades planteadas en las historias de usuario.

b) Codificar primero la prueba

Cuando se crea primero una prueba, se ahorra mucho tiempo elaborando el código que la haga pasar, siendo menor el tiempo de hacer ambos procesos que crear el código solamente. Una de las ventajas de crear una prueba antes que el código es que permite identificar los requerimientos de dicho código. En otras palabras, al escribir primero las pruebas se encuentran de una forma más sencilla y con mayor claridad todos los casos especiales que debe considerar el código a implementar. De esta forma el desarrollador sabrá con completa certeza en qué momento ha terminado, ya que habrán pasado todas las pruebas.

c) Programación en parejas

Todo el código debe ser creado por parejas de programadores sentados ambos frente a un único computador lo que en principio representa una reducción de un 50% en productividad, sin embargo, según XP no es tal la pérdida. Se entiende que no hay mucha diferencia, en lo que a la cantidad se refiere, entre el código producido por una pareja bajo estas condiciones que el creado por los mismos miembros trabajando en forma separada, con la excepción que uno o ambos programadores sean muy expertos en la herramienta en cuestión. Cuando se trabaja en parejas se obtiene un diseño de mejor calidad y un código más organizado y con menores errores que si se trabajase solo, además de la ventaja que representa contar con un compañero que ayude a solucionar inconvenientes en tiempo de codificación, los cuales se presentan con mucha frecuencia. Se recomienda que mientras un miembro de la pareja se preocupa del método que se está escribiendo el otro se ocupe de cómo encaja éste en el resto de la clase.

d) Integración secuencial

Uno de los mayores inconvenientes presentados en proyectos de software tiene que ver con

la integración, sobre todo si todos los programadores son dueños de todo el código. Para saldar este problema han surgido muchos mecanismos, como darle propiedad de determinadas clases a algunos desarrolladores, los cuales son los responsables de mantenerlas actualizadas y consistentes. Sin embargo, sumado al hecho que esto va en contra de la propiedad colectiva del código no se solucionan los problemas presentados por la comunicación entre clases.

XP propone que se emplee un esquema de turnos con el cual solo una pareja de programadores integre una vez. De esta forma se tiene plena seguridad de cuál es la última versión liberada y se le podrán hacer todas las pruebas para garantizar que funcione correctamente. A esto se le conoce como integración secuencial.

2.10.1.4 Pruebas

XP enfatiza mucho los aspectos relacionados con las pruebas, clasificándolas en diferentes tipos y funcionalidades específicas, indicando quién, cuándo y cómo deben ser implementadas y ejecutadas. Del buen uso de las pruebas depende el éxito de otras prácticas, tales como la propiedad colectiva del código y la refactorización. Cuando se tienen bien implementadas las pruebas no habrá temor de modificar el código del otro programador en el sentido que si se daña alguna sección, las pruebas mostrarán el error y permitirán encontrarlo. El mismo criterio se aplica a la refactorización. Uno de los elementos que podría obstaculizar que un programador cambie una sección de código funcional es precisamente hacer que esta deje de funcionar. Si se tiene un grupo de pruebas que garantice su buen funcionamiento, este temor se mitiga en gran medida.

a) **Pruebas unitarias**

Estas pruebas se aplican a todos los métodos no triviales de todas las clases del proyecto con la condición que no se liberará ninguna clase que no tenga asociada su correspondiente paquete de pruebas. Uno de los elementos más importantes en estas es que idealmente deben ser construidas antes que los métodos mismos, permitiéndole al programador tener máxima claridad sobre lo que va a programar antes de hacerlo, así como conocer cada uno

de los casos de prueba que deberá pasar, lo que optimizará su trabajo y su código será de mejor calidad.

Deben ser construidas por los programadores con el empleo de algún mecanismo que permita automatizarlas de modo tal que tanto su implementación y ejecución consuman el menor tiempo posible permitiendo sacarles el mejor provecho.

EL empleo de pruebas unitarias completas facilitan la liberación continua de versiones por cuanto al implementar algo nuevo y actualizar la última versión, solo es cuestión de ejecutar de forma automática las pruebas unitarias ya creadas para saber que la nueva versión no contiene errores.

b) Pruebas de aceptación

Las pruebas de aceptación, también llamadas pruebas funcionales son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario seleccionadas por el cliente deberá tener una o más pruebas de aceptación, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos.

Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia.

Es importante resaltar la diferencia entre las pruebas de aceptación y las unitarias en lo que al papel del usuario se refiere. Mientras que en las pruebas de aceptación juega un papel muy importante seleccionando los casos de prueba para cada historia de usuario e identificando los resultados esperados, en las segundas no tiene ninguna intervención por ser de competencia del equipo de programadores.

c) Cuando se encuentra un error

Al momento de encontrar un error debe escribirse una prueba antes de intentar corregirlo. De esta forma tanto el cliente logrará tener completamente claro cuál fue y dónde se

encontraba el mismo como el equipo de desarrollo podrá enfocar mejor sus esfuerzos para solucionarlo. Por otro lado se logrará evitar volver a cometerlo. Si el error fue reportado por el cliente y este creó la correspondiente prueba de aceptación junto al equipo de desarrollo, el programador encargado podrá a su vez producir nuevas pruebas unitarias que le permita ubicarla sección específica donde el error se encuentra.

2.10.2 Roles XP

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado describiremos los roles de acuerdo con la propuesta original de Beck.

- **Programador**. El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
- **Cliente**. El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.
- **Encargado de pruebas (Tester)**. El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (Tracker)**. El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las

restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

- **Entrenador (Coach)**. Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor**. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.
- **Gestor (Big boss)**. Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

2.11 LENGUAJE UNIFICADO DE MODELADO

El UML es una de las herramientas mas emocionantes en el mundo actual del desarrollo de sistemas. Esto se debe a que permite a los desarrolladores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas con otras personas.

Es la creación de de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados “Los tres amigos” cada uno diseño su propia metodología para el análisis y diseño orientado a objetos, mas tarde los tre empezaron a compartir ideas y decidieron desarrollar su trabajo en conjunto.

2.11.1 Diagramas del UML

Es un lenguaje gráfico para la especificación, visualización, construcción y documentación de piezas de información usadas o producidas durante el proceso de desarrollo de software. A estas piezas de construcción se les conoce como Artefactos. UML provee un marco arquitectónico de diagramas para trabajar sobre análisis y diseño orientado a objetos. UML es un lenguaje simbólico para expresar modelos orientados objetos y no una metodología

para desarrollarlos.

2.11.1.1 Tipos de diagramas

Un diagrama es una representación gráfica de un conjunto de elementos, la mayoría de las veces mostrados como grafo conexo de vértices (cosas) y arcos (relaciones). Los buenos diagramas hacen el sistema que se está desarrollando, más comprensible y cercano a los objetivos.

El lenguaje unificado de diagrama o notación sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente le ayuda a visualizar el diseño y a hacerlo más accesible para otros. UML está diseñado para su uso con software orientado a objetos, y tiene un uso limitado en otro tipo de cuestiones de programación.

2.11.1.2 Diagramas estructurales

Los diagramas estructurales en UML existen para visualizar, especificar, construir y documentar los aspectos estáticos del sistema. Los diagramas estructurales están organizados sobre grupos de cosas (u objetos) que se encontrarán cuando se esté modelando un sistema.

- **Diagramas de clases.** un diagrama de éste tipo muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones.
- **Diagramas de objetos**. Muestra un conjunto de objetos y sus relaciones. A diferencia de los diagramas anteriores, estos diagramas se enfocan en la perspectiva de casos reales o prototipos.
- **Diagramas de componentes**. Muestra el conjunto de componentes y sus relaciones y se utilizan para ilustrar la vista de la implementación estática de un sistema.
- **Diagramas de implantación**. Muestra un conjunto de nodos y sus relaciones; se usan para ilustrar la vista de implantación estática de un sistema.

a) **Diagramas de clases**

En UML el diagrama de clases es uno de los tipos de diagramas o símbolo estático y tiene como fin describir la estructura de un sistema mostrando sus clases, atributos y relaciones entre ellos.
Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas informáticos, en donde se intentan conformar el diagrama conceptual de la información que se manejará en el sistema.

Los diagramas de clases tiene las siguientes características:

- Las clases define el ámbito de definición de un conjunto de objetos.
- Cada objeto pertenece a una clase.
- Los objetos se crean por instanciación de las clases.

2.11.1.3 Diagrama de comportamiento

Los diagramas de comportamiento se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema.

Los aspectos dinámicos de un sistema de software involucran cosas tales como el flujo de mensajes a lo largo del tiempo y el movimiento físico de componentes en una red.

- Diagramas de casos de uso.
- Diagramas de estado. Un estado es una condición durante la vida de un objeto, de forma que cuando dicha condición se satisface se lleva a cabo alguna acción o se espera por un evento.
- Diagramas de secuencia. Muestra una interacción ordenada según la secuencia temporal de eventos y el intercambio de mensajes.
- Diagramas de colaboración. Es una forma alternativa al diagrama de secuencias a la hora de mostrar un escenario.

- Diagramas de distribución. Permiten comprender cómo estarían conectadas las unidades entre sí y dónde se ejecutarían los programas.

a) Diagramas de casos de uso

Los Casos de Uso no forma parte de la llamada Fase de Diseño, sino parte de la fase de Análisis, respondiendo el interrogante ¿Qué?. De forma que al ser parte del análisis ayuda a describir que es lo que el sistema debe hacer.

Estos diagramas muestran operaciones que se esperan de una aplicación o sistema y como se relaciona con su entorno, es por ello que se ve desde el punto de vista del usuario. Describen un uso del sistema y como éste interactúa con el usuario.

Los casos de usos se representan en el diagrama por una elipses la cual denota un requerimiento solucionado por el sistema.

El conjunto de casos de usos representa la totalidad de operaciones que va a desarrollar el sistema. Por último a estos elipses lo acompaña un nombre significativo de manera de rótulo.

Otro elemento fundamental de estos diagramas son los actores la cual representa a un usuario del sistema, que necesita o interactúa con algún caso de uso, la que también es acompañado por un nombre. Por último tenemos los flujos de eventos que corresponde a la ejecución normal y exitosa del caso de uso.