Modelo Teórico para la Identificacion del Antipatron "Stovepipe System" en la Etapa de la Implementacion de una Arquitectura de Software

Jakeline Cristina Candia Peñaloza
Postgrado en Informática
Universidad Mayor de San Andrés - UMSA
La Paz, Bolivia
Jakeline_candia@hotmail.com

***Resumen*— El presente documento pretende mostrar la forma en que el anti patrón STOVEPIPE SYSTEM puede perjudicar la implementación de una arquitectura de software.**

Para la identificación del STOVEPIPE SYSTEM se plantea un modelo teórico que sea una guía para las entidades de desarrollo de software y contribuya con la toma de decisiones.

Palabras clave*— *Implementación; Software; Antipatron

I. INTRODUCCIÓN

Los anti patrones son soluciones negativas que presentan más problemas que los que solucionan , si bien una Arquitectura de Software se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático para que los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado puede ocurrir que la solución escogida no obtenga los resultados esperados y lleve al proyecto a posibles retrasos o simplemente al fracaso. (Campo, 2009)

El fracaso de los programas de software deriva en la pérdida de la eficiencia en la planificación de la información y los recursos de las instituciones, además de la pérdida de tiempo destinado a la programación y manejo de los programas.

Esta realidad se magnifica en instituciones de grandes dimensiones con presencia en distintas zonas, como por ejemplo las instituciones públicas de nivel central del Estado, que tienen oficinas de representación en los distintos departamentos de Bolivia, tanto en ciudades capitales como en ciudades intermedias o, en el caso de instituciones de presencia en áreas rurales, como la policía y las fuerzas armadas, en regiones alejadas de los núcleos urbanos.

Los problemas empeoran cuando los sistemas a implementar no pueden relacionarse de forma adecuada y se pierde el control de las aplicaciones más aun cuando un sistema está ya heredado y en producción.

Este fenómeno corresponde al anti patrón STOVEPIPE SYSTEM (tubería caliente) que se presenta en una arquitectura cuando los componentes de los sistemas son tan complejos que no se pueden interrelacionar entre si y tampoco se los puede identificar realmente para poderlos comunicar.

Para identificar el anti patrón "STOVEPIPE SYSTEM" se plantea elaborar un modelo teórico en la etapa de implementación como en las arquitecturas ya implementadas, buscando plantear soluciones desde la realidad misma de los proyectos con la ayuda de una red neuronal para la toma de decisiones.

II. PROBLEMÁTICA

El estudio de los anti patrones es importante para comprender en mayor medida la evolución de los proyectos de software en el transcurso del tiempo, para identificar las malas experiencias que llevan a la suspensión o fracaso de los mismos, lo que genera deficiencias en el rendimiento de las instituciones.

La búsqueda de una mejor calidad en el software que permita una mayor aplicabilidad de los programas informáticos que mejoren su eficiencia llevaron con el tiempo a la constitución de dos grandes grupos de experiencias recolectadas, por un lado patrones de diseño, que corresponden a las buenas experiencias porque reúnen resoluciones a problemas, y por otro lado los anti patrones, que corresponden a las malas experiencias, ya que reúnen soluciones que han producido efectos negativos (Campo, 2009).

El anti patrón STOVEPIPE SYSTEM se presenta en una arquitectura donde la interrelación entre los sistemas es escasa o nula, por lo que los componentes están aislados y la arquitectura no se implementa de forma correcta o no se ha implementado como una arquitectura robusta que este abierta a cambios para el bien estar del proyecto de software,

Asimismo, el manejo de información es relevante y la interoperabilidad entre los sistemas brinda una ventaja competitiva a la institución y evita duplicación de esfuerzos pudiendo reutilizar componentes .

En caso de que el anti patrón afecte el desenvolvimiento del proyecto de software y terminen colapsando el mismo, la planificación necesaria en la institución se ve afectada, generando pérdidas de recursos humanos y materiales, así como tiempo.

Este problema es de mayor envergadura en las instituciones públicas, que deben por normativa legal (Ley 1178) contar con herramientas de planificación que permitan una mayor optimización de los recursos, mismos que son sometidos a procesos de auditoria interna y externa por parte de unidades de las propias instituciones y por entes externos,.

El estudio de los anti patrones equivale a un aprendizaje de las experiencias fallidas, para que puedan encontrarse las soluciones aplicables en el tiempo a dichas experiencias negativas y así poder mejorar la calidad de los proyectos.

De ahí se parte que el desarrollo de patrones de diseño permite una mejora en la capacidad pero que para ello se debe tener un aprendizaje de las experiencias de otros, lo cual implica una adecuada retroalimentación de las experiencias pasadas para poder mejorar sustancialmente en los patrones de diseño a futuro.

Puntualmente, el anti patrón Stovepipe System comprende un conjunto de elementos interrelacionados que están tan estrechamente unidos entre sí que los elementos individuales no se pueden diferenciar, actualizar o refactorizar. El sistema se debe mantener hasta que pueda ser sustituido en su totalidad por un nuevo sistema. Por lo tanto, con el tiempo este sistema normalmente también se convierte en un sistema heredado.

III. OBJETIVOS

El objetivo del pesente trabajo es mostrar el planteamiento de la elaboración de un modelo teórico que utilice las técnicas de redes neuronales para identificar el anti patrón "STOVEPIPE SYSTEM" en la implementación de una arquitectura de software.

OBJETIVOS ESPECÍFICOS

- Estudiar la Arquitectura de Software y sus componentes principalmente el proyecto.
- Delimitar arquitecturas orientadas a servicios y al modelo vista controlador.
- Identificar problemas con la arquitectura de software.
- Estudiar el anti patrón Stovepipe System.
- Utilizar técnica de inteligencia artificial como las redes neuronales para colaborar en la toma de decisiones.
- Ilustrar sobre la solución al caso del anti patrón Stovepipe System

IV. HIPÓTESIS

Debido a que las redes neuronales son dinámicas y realizan un aprendizaje con ciertas entradas y con un proceso de retroalimentación se ha escogido esta técnica para colaborar con la elaboración del modelo teórico debido a que en la retroalimentación y en el entrenamiento de esta red se puede contar con casos y circunstancias distintas basándose en que cada institución y cada proyecto es diferente.

Es así que se ha planteado la siguiente hipótesis:

Mediante un modelo teórico que utilice las técnicas de redes neuronales se contribuye a la identificación del anti patrón Stovepipe System en la implementación de una arquitectura de software

V. MARCO TEORICO

A. LOS ANTI PATRONES

El concepto de anti patrón tiene foco en la identificación de errores comunes realizados en proyectos de software. Los anti patrones describen una solución común a un problema que lleva a tener consecuencias negativas durante su aplicación, generalmente por no lograr comprender el problema, por no tener el conocimiento y la experiencia suficiente para resolver el tipo de problema en particular, o simplemente por aplicar un patrón en un escenario que no es el adecuado.

W. Brow, B. Malvean, S. MacCormick y T. Mowbray (1998) definen al anti patrón como la forma para capturar la experiencia de los desarrolladores y que la misma pueda ser asimilada más fácilmente por otros desarrolladores.

Los anti patrones examinan las causas, los síntomas y las consecuencias de implementar una solución incorrecta y ofrecen una solución refactorizada, la cual satisface la necesidad requerida por el proyecto de desarrollo de software. Son métodos que permiten realizar un mapeo eficiente entre una situación general y una clase de solución específica. Los anti patrones proveen experiencia real en el reconocimiento de problemas que son recurrentes en la industria del software y dan una solución para la mayoría de estos problemas.

Los anti patrones al igual que los patrones se basan en un vocabulario común para identificar los problemas y discutir sus soluciones en forma efectiva, de forma que sea sencillo comunicar las ideas y soluciones complejas. Los anti patrones dan soporte a la resolución de conflictos, utilizando recursos organizacionales de diferentes niveles donde es necesario, articulando la colaboración entre las distintas fuerzas de diferentes niveles de la gestión y desarrollo del proyecto.

Los anti patrones son de tres tipos:

- Antiptrones de desarrollo de software
- Anti patrones de arquitectura de software
- Anti patrones de gestión de proyectos de software

Desarrollo de Software: Se centran en problemas asociados al desarrollo de software a nivel de aplicación.

Arquitectura de Software: Se centran en la estructura de las aplicaciones y componentes a nivel de sistema y empresa.

Gestión de Proyectos de Software: En la ingeniería del software, más de la mitad del trabajo consiste en comunicación entre personas y resolver problemas relacionados con éstas. Los anti patrones de gestión de proyectos de software identifican algunos de los escenarios clave donde estos temas son destructivos para el proceso de software (Brow, 1998).

A1. Anti patrones de desarrollo de Software

- The Blob ("Clases gigantes"): Este anti patrón se da en objetos con muchos atributos o muchas operaciones. Esto rompe las ventajas de la programación OO, ya que estas clases tan grandes son muy difíciles de mantener, de reusar y de probar. Suele aparecer por un diseño malo o debido a sistemas heredados.
- Lava Flow ("Código muerto"): Este anti patrón se da cuando se entrega software antes de ser terminado o suficientemente probado que tiene un código no óptimo y, al ser expuesto, sus características no pueden ser modificadas.
- Poltergeists ("No se sabe bien lo que hacen algunas clases"): Esta mala práctica se refiere a objetos de un ciclo de vida corto cuya única función suele ser invocar métodos de otros objetos. Esto hace que crezca la dificultad para mantener el sistema.
- Golden Hammer ("Para un martillo todo son clavos"): Este anti patrón se refiere al uso de una tecnología, patrón, arquitectura etc. para cualquier problema o situación incluso cuando es evidente que no va ser útil.
- Spaghetti Code ("Muchos if ó switch"): se refiere a un código mal estructurado.
- Cut & Paste programming ("Cortar y pegar código").

A2. Anti patrones de Arquitectura de Software

Autogenerated Stovepipe (Estufa Auto generada): Este atipatron se presenta cuando se realizan migraciones la misma arquitectura se presenta para cualquier proyecto emergente.

Stovepipe Enterprise (Estufa de Empresa): Se establece por una falta de coordinación y planificación de un conjunto de sistemas se caracteriza por una arquitectura que inhibe el cambio.

Jumble (Revoltijo): Cuando components horizontales y verticals de mezclan, se establece una arquitectura inestable.

Stovepipe System (Estufa de Sistema): Son subsistemas que tienen soluciones ad hoc utilizando múltiples mecanismos, y todos están integrados como punto a punto. La integración entre subsistemas no es realizada correctamente. The Stovepipe System se basa en cómo se integran los subsistemas.

Cover Your Assets (cubra sus activos): La documentación de los procesos muchas veces no sirve de mucho para la elaboración de los sistemas. Con el fin de evitar cometer errores los autores plantean multiples alternativas a la hora de la toma de decisiones.

Vendor Lock−In: Vendor Lock−In (Dependencia de un proveedor): Ocurre en sistemas que dependen de arquitecturas propietarias.

Wolf Ticket (ticket lobo): Declarar compatibilidad con un estándar cuando ésta no existe, o bien cuando el estándar solo incluye recomendaciones no obligatorias que, de hecho, no se siguen.

Design by Committee: Creación de Arquitecturas extremadamente complejas.

A3. Anti patrones de Gestion

Blowhard Jamboree: Las opiniones de los expertos influencian mucho en las decisiones tecnológicas, lo mismo que las opiniones de los ejecutivos de la empresa.

Analysis Paralysis: La lucha contra la perfección en la fase de análisis crea embotellamiento con lluvia de requisitos y de modelos.

Viewgraph Engineering: Los desarrolladores se enfrascan en realizar fichas visuales y documentos en lugar de desarrollar software.

Death by Planning: La planificación excesiva lleva a horarios complejos que perjudican el desenvolvimiento del proyecto

B. ANTIPATRON STOVEPIPE SYSTEM

El anti patrón STOVEPIPE SYSTEM radica en la falta de coordinación y planificación en un conjunto de sistemas. Este antipatron radica en como se coordinan los subsistemas dentro de un conjunto de sistemas.

El problema clave en este anti patrón es la falta de abstracciones en subsistemas comunes.

Los subsistemas están integrados de una manera ad hoc utilizando múltiples estrategias de integración y mecanismos. Todos los subsistemas están integrados punto a punto, como se muestra en la siguiente figura.

INTERFACES STOVEPIPE

Fuente: Broun (1998)

El enfoque de integración para cada par de subsistemas no se aprovecha adecuadamente. Además, la implementación del sistema es frágil porque hay muchas dependencias individuales de los subsistemas, la configuración del sistema, detalles de la instalación, y el estado del sistema. El sistema es difícil de extender, como se integra cada nueva capacidad y la alteración, la complejidad del sistema aumenta a lo largo del ciclo de vida del sistema la ampliación y mantenimiento del sistema se torna más complejo.

Síntomas y Consecuencias

- Gran diferencia semántica entre la arquitectura y la documentación del software implementado; la documentación no corresponde con la implementación del sistema.
- Arquitectos no están familiarizados con los aspectos clave de la solución de integración.
- Existe un exceso en el presupuesto y se ha sobrepasado el cronograma de trabajo.
- Los cambios son costosos de implementar y el mantenimiento del sistema genera costos sorprendentes.
- El sistema puede cumplir con la mayoría de los requisitos de papel, pero no cumple con las expectativas del usuario.
- Los usuarios deben inventar soluciones para hacer frente a las limitaciones del sistema.
- Los procedimientos de instalación de los sistemas son complejos.
- La Interoperabilidad con otros sistemas no es posible, y hay una incapacidad para apoyarla gestión del sistema integrado y las capacidades de seguridad entre sistemas disminuye.
- Los cambios en los sistemas se vuelven cada vez más difícil.
- Las modificaciones del sistema se vuelven cada vez más peligrosos para introducir nuevos errores graves.

Causas típicas

- Múltiples mecanismos de infraestructura utilizados para integrar los subsistemas; ausencia de un mecanismo común, hace la arquitectura difícil de describir y modificar.
- La falta de abstracción; todas las interfaces son únicos para cada subsistema.
- Insuficiente uso de metadatos; los metadatos no están disponibles para apoyar las extensiones del sistema y reconfiguraciones sin cambios de software .
- Estrecho acoplamiento entre clases implementadas por lo que requiere código de cliente que es excesivo.
- Falta de visión arquitectónica.

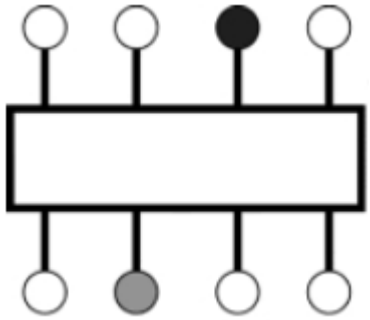
SOLUCIÓN

La solución refactorizada al STOVEPIPE SYSTEM es una arquitectura de componentes que proporciona flexibilidad para la sustitución de módulos de software. Subsistemas que se modelan de manera abstracta

La sustitución puede ser tanto estática (sustitución de componentes en tiempo de compilación) y dinámica

(en tiempo de ejecución de unión dinámica). La clave para la definición de las interfaces de los componentes es descubrir la abstracciones apropiadas . Las abstracciones de subsistemas, modelar las necesidades de interoperabilidad de la sistema sin exponer diferencias innecesarias entre subsistemas y detalles específicos de la implementación.

Con el fin de definir una arquitectura de componentes, usted debe elegir un nivel básico de funcionalidad que la mayoría de las aplicaciones apoyará. En general, ese nivel debe ser bajo y el foco en un solo aspecto de la interoperabilidad, tales como el intercambio o la conversión de datos.



Fuente: Broun(1998)

Las interfaces necesitan tener componentes en común y que sean robustos para poder desacoplar los subsistemas en el sistema integrado.

C. REDES NEURONALES

C1. Definiciones

Existen varias definiciones del término redes neuronales dentro del campo de la inteligencia artificial. En este contexto, el autor del trabajo de investigación Damián Jorge Matich (2001) señala como ejemplos de definiciones dadas a las redes neuronales las siguientes:

1) Una nueva forma de computación, inspirada en modelos biológicos.

2) Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.

3) Un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

4) Son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico (p. 8-9)

La primera definición refiere a la naturaleza de las redes neuronales en su constitución y su impacto en el mundo de la informática. En este segundo aspecto, las redes neuronales comprenden una nueva forma de entender la computación, porque van más allá de la concepción enteramente lógica del programador informático, de modelar de manera matemática el programa, mediante una solución al problema identificado, a través de algoritmos codificados cuyas propiedades se procede a intenta para resolver dicho problema.

Este paradigma convencional en la programación, diferente a la innovación de la red neuronal, y que se orienta a resolver problemas específicos identificados por los programadores mediante soluciones basadas de manera lógica en algoritmos seleccionados cuyas propiedades permitan esa solución matemática.

El otro elemento importante de la primera definición sobre las redes neuronales es su inspiración en modelos biológicos. Esa es la base del desarrollo de las redes neuronales, que comprenden una inspiración de la estructura del sistema nervioso de los seres vivos, que se compone de redes de neuronas que se encuentran interconectadas unas a otras en una compleja estructura, y que permite la transmisión de datos y el aprendizaje de las situaciones experimentadas para la adecuación del ser al entorno donde se desenvuelve.

Al respecto, Xabier Basogain Olabe (1998 p. 13) señala como fundamento del empleo de las redes neuronales de acuerdo a la insiración de la biologías, y en especial de la estructura compuesta por el sistema nerviosos humano, el siguiente:

> Las diferentes configuraciones y algoritmos que se diseñan para las redes neuronales artificiales están inspiradas en la organización del complejo sistema neuronal del cerebro humano. No obstante conviene aclarar que esta inspiración no supone que las ANN lleguen a emular al cerebro como algunos optimistas lo desean ya que entre otras limitaciones el conocimiento sobre el modo de funcionamiento y comportamiento del cerebro es bastante simple y reducido. De hecho los diseñadores de redes artificiales van más lejos del conocimiento biológico actual y prueban nuevas estructuras que presentan un comportamiento adecuado y útil.

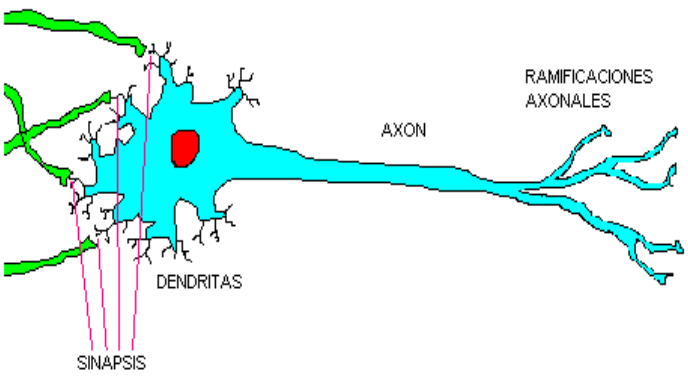
La experiencia alcanzada por la red neuronal es un elemento importante, porque se centra en la facultad de los seres vivos, y en especial en los seres humanos, de tener a la experiencia como medio de aprendizaje para poder comprender el proceso de desenvolvimiento que se tiene en un contexto dado, para resolver las dificultades y los problemas que se llegan a identificar con el tiempo.

Esta experiencia para la resolución de problemas no se puede establecer mediante un algoritmo, el cual es formal, y que está determinado para la resolución de un problema concreto en un contexto específico de condiciones invariables.

Otro autor quien trata el tema es, Damián Matich (2001 p. 4), quién indica en este sentido que:

> Las redes neuronales son más que otra forma de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Si se examinan con atención aquellos problemas que no pueden expresarse a través de un algoritmo, se observará que todos ellos tienen una característica en común: la experiencia.

Una red neuronal biológica tiene como base de su construcción a la neurona, que es la unidad de la res o sistema. A continuación se expone una figura que ejemplifica la forma de una neurona biológica



Fuente: MAGOMAR. UPV

Además de su carácter estructural, similar a las neuronas biológicas, la neurona artificial es un modelo matemático, cuya composición se sustenta en ecuaciones numéricas que dan forma a la estructura de red.

Otro concepto es el vertido por los autores Juan Bravo et al (2004 p. 2)

> Una red neuronal artificial es un sistema constituido por elementos de procesamiento interconectados llamados neuronas, los cuales están dispuestos en capas (una de entrada, una o varias intermediarias y una de salida) y que son responsables por la no-linealidad de la red, a través del procesamiento interno de ciertas funciones matemáticas.

C2. Características

Se concibe en las definiciones expuestas a las redes neuronales como sistemas, compuestas por redes interconectadas para procesar información bajo la inspiración del modelo neurológico de la biología. Partiendo de esa lógica se tienen como características de las redes neuronales las siguientes:28

- Elevada intercomunicación: Se requieren en ocasiones interconexiones específicas entre cada procesador.

[28] Las características de las redes neuronales señaladas se encuentran en el texto de aprendizaje con redes neuronales artificiales, de Graciani, M. Á. F., & Bonal, M. T. L. Francisco Javier Gómez Quesada (Quesada, F. J. G., Graciani, M. A. F., Bonal, M. T. L., & Díaz-Mata, M. A. (1994).

- Fuerte paralelismo: En gran escala se puede llegar a contar con cientos de miles de celdas en las redes neuronales.

- Representación distribuida: La fortaleza de las conexiones se llega a alterar para añadir nueva información.

- Tolerancia a fallos: Si una "neurona" falla, el sistema en su conjunto sigue funcionando, y la precisión se centra en el número de neuronas que fallen.

- Cálculo colectivo: Una red neuronal no ejecuta instrucciones particulares, sino en conjunto, de manera colectiva, resuelven problemas particulares.

- Auto-organización: Una red neuronal puede adaptar su estructura para descubrir nuevos patrones.

Un señalamiento más concreto de las características de las redes neuronales la efectúa el profesor de la Escuela Superior de Ingeniería de Bilbao Xabier Basogain Olabe (1998):

- Aprender: comprende el poder adquirir el conocimiento de una cosa por medio del estudio, ejercicio o experiencia., cambiando las redes neuronales su comportamiento en función del entorno, como por ejemplo llegando a mostrarles n conjunto de entradas y ellas mismas se ajustan para producir unas salidas consistentes.

- Generalizar: refiere a extender o ampliar una cosa, lo que en caso de las redes neuronales, éstas llegan a generalizan automáticamente su funcionamiento debido a su propia estructura y naturaleza, pudiendo ofrecer, dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a los efectos de ruido o distorsión.

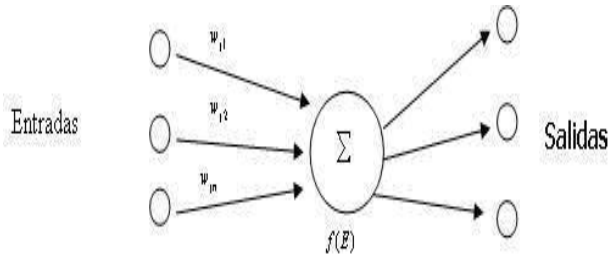
- Abstraer: Consiste en aislar mentalmente o considerar por separado las cualidades de un objeto, siendo en este sentido algunas redes neuronales capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes o relativos.

C3. Neurona artificial

El elemento fundamental de la estructura de las redes neuronales, es, por lógica, la neurona artificial.

Olabe (1998) señala que la neurona artificial fue diseñada para emular las características del funcionamiento básico de la neurona biológica, donde en esencia se llega a aplicar un conjunto de entradas a la neurona, cada una de las cuales representa una salida de otra neurona, y cada entrada se multiplica por su peso o ponderación, el cual es correspondiente análogo al grado de conexión de la sinapsis de la neurona en la red, derivando en que todas las entradas ponderadas se suman y se determina el nivel de excitación o activación de la neurona dentro de la red.

Figura 1. Estructura de una neurona artificial



Fuente: Disa (2006)

Al formar parte de una red, la neurona artificial no tiene sentido de existencia ni funcionalidad práctica si no se encuentra entrelazada con semejantes en una red, mismas que se establece en capas, que se agrupan en los siguientes tipos de capas:29

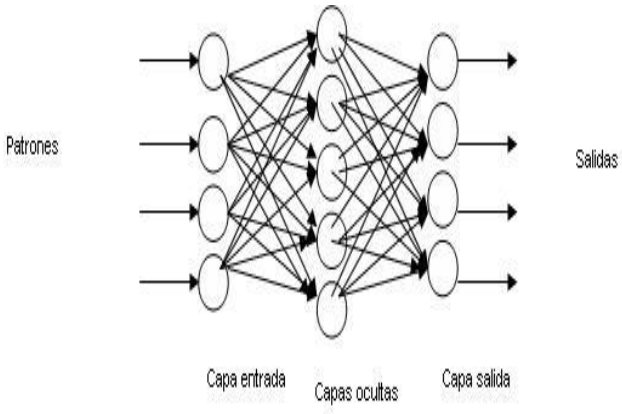
- Capa de entrada: actúa de buffer ante los datos de los patrones de entrada se introducen a la red.

- Capas ocultas: Puede haber una, varias o bien no existir dependiendo de la topología de la red y de la potencia de esta. Normalmente, las neuronas ocultas son las que realmente realizan el procesado de los datos, de forma que cuanto más complejo sea el problema, habrá que introducir más neuronas en estas capas.

- Capa de salida: Actúa de buffer de salida y ofrece la salida de la red.

Estas capas conforman en sus distintos grados de asociación la tred neuronal, que se conforma de manera análoga a la mostrada en la siguiente figura:

Figura 2. Arquitectura de red neuronal



Fuente: Disa (2006)

[29] La clasificación de las capaz proviene del texto de Axiel Zuzizarreta (2006)

La capa o nivel es un conjunto de neuronas cuyas entradas provienen de la misma fuente y cuyas salidas se dirigen al mismo destino. El profesor Héctor Alaiz Moretón (2013 p. 5) señala que las capas se constituyen de los siguientes componentes básicos, a partir de los cuales se estructura la red neuronal:

- Unidad de proceso: Existen tres tipos de unidades en cualquier sistema: entradas, salidas y ocultas. Las unidades de entrada reciben señales desde el entorno; las de salida envían la señal fuera de la red, y las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema.

- Estado de Activación: Los estados del sistema en un tiempo t se representan por un vector A(t). Los valores de activación pueden ser continuos o discretos, limitados o ilimitados. Si son discretos, suelen tomar un conjunto discreto de valores binarios, así un estado activo se indicaría con un 1 y un estado pasivo se representaría por un "0"

VI. MODELO TEÓRICO

Para la elaboración del modelo teórico se plantea la siguiente forma:

- Se debe evaluar la fase en la que se encuentra la arquitectura
- Se debe modelar los casos de estudio con UML.
- Se realizara la matriz de entrada para la red neuronal en base a lo modelado con UML.
- Se entrenara a la red neuronal con la matriz de ingresos y la retroalimentación que se tenga de los casos de estudio.
- Se obtendrá una salida de la red neuronal que verificará si el Stovepipe System está presente en la implementación de la arquitectura o estuvo presente durante su implementación.
- Se propondrá la posible solución de acuerdo a la teoría del Stovepipe System

VII. CONCLUSIONES

- En la implementación de las arquitecturas de software se presentan anti patrones que pueden colaborar a que las soluciones que en un momento parecían las adecuadas se conviertan en un impedimento para lograr los objetivos.
- El presente proyecto pretende elaborar un modelo para la identificación del anti patrón STOVEPIPE SYSTEM en el momento de la implementación de la arquitectura de software.
- Con la identificación del anti patrón se pueden tomar medidas que contribuyan a tener soluciones que eviten retrasos en los proyectos.
- Se utilizan las redes neuronales como ayuda para la identificación del anti patrón en una arquitectura, con ese resultado se pretende utilizar la teoría para dar solución a los posibles problemas ocasionados.

REFERENCIAS

[19] Campo, G. (2009). *Patrones de Diseño, Refactorización y Anti patrones.* Recuperado el 17 de Junio de 2014, de Ucasal: www.ucasal.edu.ar/.../4-p101-Campo-pdf

[20] Hernández, Roberto, Fernández, carlos y Baptista, Pilar. (2010). *Metodología de la investigación.* México DF: McGraw Hill.

[21] Junta de Andalucía. (2013). *Especificación de los Anti patrones de diseño de Arquitectura de Software.* Recuperado el 12 de Octubre de 2014, de Junta de Andalucía: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/820

[22] Junta de Andalucía. (2014). *Los anti patrones.* Recuperado el 12 de Junio de 2014, de Marco de Desarrollo de la Junta de Andalucía: www.juntadeandalucia.es/servicios/madeja/contenido/recurso/820

[23] Méndez, C. (2012). *Metodología.* Bogotá: McGraw-Hill.

[24] Morales, F. (2014). *Tipos de investigación.* Recuperado el 22 de Junio de 2014, de http://manuelgross.bligoo.com/conozca-3-tipos-de-investigacion-descriptiva-exploratoria-y-explicativa

[25] MUGPERÚ. (1 de Agosto de 2013). *Las Estadísticas de fracasos en los Proyectos TI.* Recuperado el 23 de Junio de 2014, de MUG Perú: http://blog.mugperu.com/index.php/2013/08/las-estadisticas-de-fracasos-en-los-proyectos-de-ti/#sthash.8RrX9NKg.dpuf

[26] Navegapolis. (2009). *Herramientas de uso libre para gestión de proyectos.* Recuperado el 16 de Junio de 2014, de Navegapolis: http://www.navegapolis.net/content/view/56/49/

[27] Procesos Águiles. (2014). *Retrospectiva (Sprint Retrospective).* Recuperado el 16 de Junio de 2014, de Prrocesos Águiles: http://www.proyectosagiles.org/retrospectiva-sprint-retrospective

[28] Proyectos Ágiles. (2013). *Iteración de procesos Scrum.* Recuperado el 12 de Junio de 2014, de Proyectos Águiles: http://www.proyectosagiles.org/planificacion-iteracion-sprint-planning

[29] Proyectos Ágiles. (2013). *Procesos Scrum.* Recuperado el 15 de Junio de 2014, de Proyectos Ágiles: http://www.proyectosagiles.org/fundamentos-de-scrum

[30] Quesada, F. J. G., Graciani, M. A. F., Bonal, M. T. L., & Díaz-Mata, M. A. (1994). *Aprendizaje con redes neuronales artificiales.* Recuperado el 16 de Octubre de 2014, de revista de la Facultad de Educación de Albacete: file:///C:/Users/Malaga/Downloads/Dialnet-AprendizajeConRedesNeuronalesArtificiales-2281678.pdf

[31] Quisbert Camacho, M. G. (Noviembre de 2010). *UDABOL.* Recuperado el 4 de Junio de 2014, de Buenas tareas: http://www.buenastareas.com/ensayos/Patrones-y-Frameworks/1230369.html

[32] W. Brow, B. Malvean, S. MacCormick y T. Mowbray . (1998). *Los anti patrones.* Recuperado el 13 de Junio de 2014, de UNAM: http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/anti patrones.html

[33] Wordpress. (16 de Septiembre de 2013). *Patrones Arquitectónicos.* Recuperado el 12 de Junio de 2014, de Ingenio DS : http://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/