Modelo de gestión del conocimiento aplicado a un sistema complejo: Desarrollo de fábricas de software

Ticsiana Lorena Carrillo Bovea
Universidad Distrital "Francisco José de Caldas", Bogotá D.C., Colombia
aiscit@yahoo.com

Víctor Hugo Medina García
Universidad Distrital "Francisco José de Caldas", Bogotá D.C., Colombia
vmedina@udistrital.edu.co

RESUMEN

Este artículo pretende mostrar el punto de conexión entre la teoría de la complejidad y la gestión del conocimiento. Se proyecta encontrar mecanismos para comunicar y compartir conocimientos en forma eficiente, mejorando la gestión de conocimiento en un sistema complejo como lo es el desarrollo de fábricas de software. Como resultado, se plantea un modelo de gestión de conocimiento, teniendo en cuenta nociones de teoría de complejidad aplicadas a un sistema de desarrollo de fábricas de software.

Pabras claves: Gestión de conocimiento, teoría de la complejidad, teoría del caos, desarrollo de software, fábrica de software.

ABSTRACT

This article shows the connection between complexity theory and knowledge management. It is planned to find mechanisms to communicate and share knowledge efficiently, improving knowledge management in a complex system such as the development software factories. As a result, this article will unveil a model of knowledge management, taking into account concepts of complexity theory applied to a development system software factories.

Keywords: knowledge management, complexity theory, chaos theory, software development, software factory.

1. INTRODUCCIÓN

Antes de comenzar, es necesario saber porque se llega a aplicar dos temas complejos por sí mismos, y complejos mutuamente. El primer concepto, gestión del conocimiento, enmarca el mundo actual, el conocimiento como fuente de poder y el mejor aliado a la hora de desarrollar estrategias organizacionales. Pero si ya las organizaciones poseen el conocimiento, cuál es la razón para que ahora se introduzca la palabra gestión?

Entonces viene a la mente, el plano organizacional, en donde se necesitan conocimientos técnicos, sociales y sobre todo procesos-métodos. Qué ocurre si no existe un apoyo en la transmisión y generación del conocimiento?, si cada vez que entra un nuevo funcionario o programador tiene que tratar de desarrollar la rueda nuevamente para aprender la forma de hacer las cosas? Lógicamente la empresa o el proyecto en sí, tendrá una ampliación en el tiempo de ejecución de tareas y por ende un aumento de gastos, y una disminución en su rendimiento.

Por otro lado el término complejidad existe y ha existido por siempre, incluso viene incrustado con el hombre, con el mundo y la naturaleza misma. Miles de procesos tan simples a nuestra vista son complejos y al mismo tiempo fascinantes.

La complejidad y el conocimiento viven juntos e incluso según la hipótesis de Seth Lloyds (Lloyd, 2010) "Everything that is worth under-standing about complex systems can be under-stood in terms of how it processes information", la complejidad puede tomarse en términos de procesamiento de información, es decir, conocimiento.

Entonces, se puede aprovechar las ventajas de las nociones y de las diferentes aplicaciones de la teoría de la complejidad para ofrecer un modelo de gestión de conocimiento para el desarrollo de un sistema complejo.

2. CONCEPTUALIZACIÓN

1.1 GESTIÓN DEL CONOCIMIENTO

Para poder desarrollar o tratar de aplicar la gestión de conocimiento es necesario conocer diferentes nociones sobre ella, por ejemplo se puede explicar en términos de transferencia de conocimiento: La Gestión o Administración del Conocimiento es un concepto utilizado en las empresas, que pretenden transferir el conocimiento y experiencia existente en los empleados, de modo tal que pueda ser utilizado como un recurso disponible para otros en la organización. La Gestión del Conocimiento pretende poner al alcance de cada empleado la información que necesita en el momento preciso para que su actividad sea efectiva (lagestiondelconocimiento, 2010).

También se puede establecer con base a la satisfacción de necesidades: "El proceso de administrar continuamente conocimiento de todo tipo para satisfacer necesidades presentes y futuras, para identificar y explotar recursos de conocimiento tanto existentes como adquiridos y para desarrollar nuevas oportunidades" (Quintas, 1997).

Otras definiciones se enfocan a partir del capital intelectual: "El conjunto de procesos y sistemas que permiten que el capital intelectual de una organización aumente de forma significativa, mediante la gestión de sus capacidades de resolución de problemas de forma eficiente (en el menor espacio de tiempo posible), con el objetivo final de generar ventajas competitivas sostenibles en el tiempo" (Carrión, 2001). Y la generación de valor a través de ella: "Crear valor a partir de los activos intangibles de la organización" (Sveiby, 2000).

Del mismo modo se encuentran nociones orientadas a la competitividad: Dirección planificada y continua de actividades y procesos para potenciar el conocimiento e incrementarla competitividad a través del mejor uso y creación de recursos de conocimiento individual y colectivo (Forum, 2004).

Pero en últimas, todos los conceptos de gestión de conocimiento apuntan a la manera como se obtiene, clasifica, distribuye, trasmite y se guarda el conocimiento: Es un concepto utilizado en las empresas, que pretende transferir el conocimiento y experiencia existente en los empleados, de modo que pueda ser utilizado como un recurso disponible para otros en la organización. La Gestión del Conocimiento pretende poner al alcance de cada empleado la información que necesita en el momento preciso para que su actividad sea efectiva (gestion-conocimiento.com, 2010).

Con base a las anteriores nociones se puede establecer que los objetivos de la gestión del conocimiento son:

- Formular una estrategia de alcance organizacional para el desarrollo, adquisición y aplicación del conocimiento.
- Implantar estrategias orientadas al conocimiento.
- Promover la mejora continua de los procesos de negocio, enfatizando la generación y utilización del conocimiento.
- Monitorear y evaluar los logros obtenidos mediante la aplicación del conocimiento.
- Reducir los costos asociados a la repetición de errores.
- Reducir tiempos de ciclos en el desarrollo de nuevos productos.

Y se puede definir como un soporte tecnológico: Un "Sistema de Gestión del Conocimiento", es una plataforma de tecnologías de información y comunicaciones (TICs) que soporta los procesos de GC en la organización. Se

centra en crear, recopilar, organizar y difundir el "conocimiento" de una organización, en lugar de la información o los datos (Medina, 2010).

1.2 TEORÍA DE LA COMPLEJIDAD

Muchos son los fenómenos y hechos que se encuentran intrínsecos dentro de la teoría de la complejidad: La teoría de la relatividad de Albert Einstein, el Principio de incertidumbre o indeterminación, fractales, inteligencia artificial, teoría del caos, agentes, emergencia, auto-organización (Figura 1); lo cual hace que encontrar una definición única sea casi imposible. Por lo general se asocia a la teoría de la complejidad con el estudio de sistemas complejos.

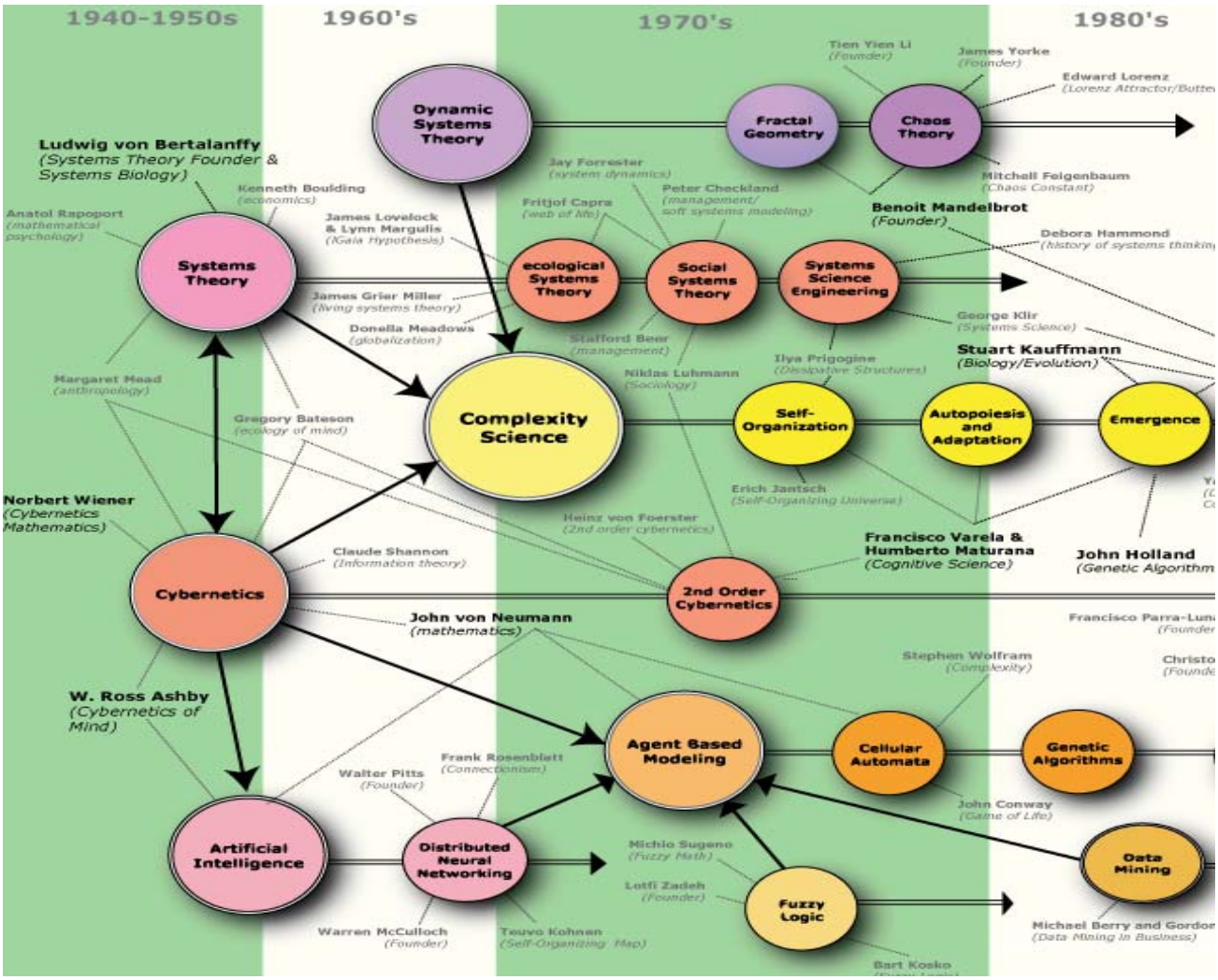

Figura 1: Una historia de la ciencia de la complejidad. Fuente: Wikipedia

Un sistema es complejo cuando se compone de muchas partes que se interconectan de forma intrínseca (Moses, 2001). Un sistema presenta complejidad dinámica cuando la causa y efecto son sutiles, con el tiempo (Senge, 1995). Un sistema es complejo cuando se compone de un grupo de unidades relacionadas (subsistemas), para lo cual el grado y naturaleza de las relaciones es imperfectamente conocidos (Sussman, 2007).

1.3 FÁBRICAS DE SOFTWARE

En estos momentos se necesitan sistemas de software que puedan ser desarrollados en tiempos cortos, es decir, disminuir el tiempo que transcurre de la especificación de requerimientos a la implementación. También se debe ser capaz de reciclar un conjunto de procedimientos, procesos, diseños, estrategias, alternativas y experiencias que permitan mejorar los procesos de desarrollo de software. Además estadísticamente se considera que el 64% de los proyectos informáticos son un fracaso ya sea porque no se efectúan dentro del tiempo / presupuesto planeado o porque no están dentro del alcance proyectado, todo esto nos confirma que es necesario automatizar el proceso de producción de software, mediante líneas de producción, similar al proceso de ensamblaje de automóviles.

Para la siguiente década la demanda de software aumentará de manera significativa, promocionada por los cambios económicos mundiales, las nuevas aplicaciones tecnológicas, al auge de los dispositivos móviles, el internet, la inteligencia artificial. En otras palabras *no es posible* que la oferta de desarrollo de software pueda cubrir toda esta demanda, entonces ¿cómo se obtendrá la capacidad para poder satisfacer la anterior demanda? La posible solución no es aumentar el número de desarrolladores, porque esto podría traer más problemas que ventajas. Lo que se debe lograr es cambiar radicalmente los métodos de desarrollo de software para que las personas implicadas no realicen tareas repetitivas.

Otro aspecto que se le agrega a este escenario es la constante evolución de la tecnología, la cual adiciona complejidad tanto al proyecto como al proceso y al producto, al mismo tiempo que coexiste con la propia evolución del dominio del problema (Greenfield, 2005). Esto aumenta la exigencia en el desarrollo de soluciones, forzando a las empresas de desarrollo de software a buscar opciones que les permitan tener procesos agiles y eficientes, disminuir el consumo de recursos y conservando niveles altos de calidad (Pohl, 2005).

Hoy en día, aún se puede considerar el desarrollo de software como un proceso artesanal fundamentado en las habilidades y conocimientos de los ingenieros involucrados. Cada nuevo producto exigido a la empresa de desarrollo de software es una solución nueva, única y ajustada a la medida del cliente.

Las industrias han multiplicado su capacidad pasando de la artesanía, donde todo son productos creados a partir de cero por individuos o pequeños equipos, a la industria, donde una amplia gama de variantes del producto son ensambladas rápidamente a partir de componentes reutilizables creados por múltiples proveedores o por la misma industria.

Con el pasar del tiempo los clientes necesitaban un mayor número de soluciones, una mayor calidad, una disminución del tiempo de desarrollo, por lo tanto se hace necesario cambiar el enfoque de solución a producto y centrarse en el proceso como fuente central del producto. Al encaminarse hacia el proceso de desarrollo del producto aparecen las fábricas de software con el fin de mejorar la productividad, estás a través de un banco de conocimientos, componentes, herramientas y procesos utilizan la reutilización para otorgar al cliente una mayor calidad, flexibilidad, menores precios y tiempos de entrega (Crnkovic, 2000). Mediante la reutilización se puede aprovechar las habilidades de varios desarrolladores en varios proyectos al mismo tiempo, esto se ha considerado como el enfoque principal para lograr importantes mejoras en la productividad y en la calidad de la ingeniería de software (Estublier, 2005), (Barachisio, 2004). El desarrollo se enfoca hacia la utilización de módulos ya existentes.

Una fábrica de software puede definirse como “una línea de producción de software que configura herramientas extensibles, procesos y contenido mediante una plantilla de fábrica de software basada en un esquema de fábrica de software para automatizar el desarrollo y mantenimiento de las variantes de un producto arquetípico mediante la adaptación, montaje y configuración de marco basado en componentes” (Short, 2004). Otros en forma más práctica la definen como “una línea de producto que configura herramientas de desarrollo extensible como Microsoft Visual Studio Team System (VSTS) con contenido empaquetado y orientación, cuidadosamente diseñado para construir tipos de aplicaciones específicas”.

Este tipo de proyecto se puede conducir en el marco de un diseño experimental (Kleppe, 2003), mediante la metodología MSF for agile Software Development de Microsoft (Pressman, 2006), (Lano, 2005), esta soporta desarrollo iterativo con aprendizaje y refinamiento continuo. La definición del producto, el desarrollo y las

pruebas ocurren en interacciones incrementales (Figura 2). El principal principio es la adición de funcionalidad interactiva, que mejora no solo la calidad del software, sino también la forma como se construye (Warmer, 2003). Se puede optar por esta metodología por el desarrollo en espiral y por la facilidad de integración con Visual Studio.net.

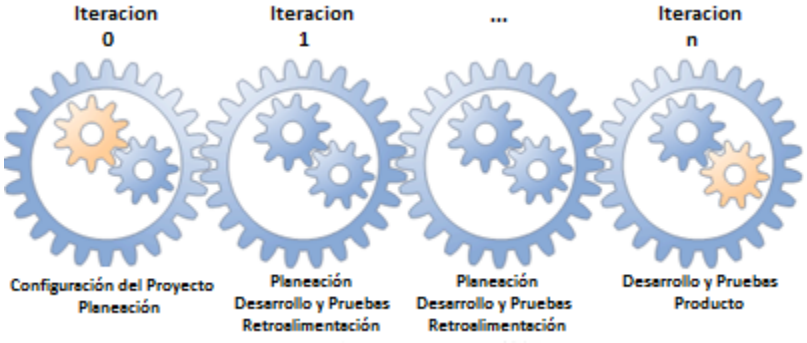

Figura 2: Ciclos e interacciones. Fuente: Microsoft Developer Network

Cada una de las interacciones tiene las siguientes fases: previsión, planeamiento, desarrollo, estabilización, e implementación.

3. MODELO

En una sociedad que busca incansablemente los métodos que faciliten los procesos para producir sus bienes y servicios surge la imperante necesidad de implementar y desarrollar procesos de transferencia de conocimiento, a través de herramientas que resulten rápidas, confiables y económicas.

Al tratar de aplicar un modelo de gestión de conocimiento a un sistema complejo, es posible que el modelo adopte las cualidades de los sistemas complejos y se vuelva también complejo. Además si se toma como base las características de los sistemas complejos como son emergencia, auto-organización y caos se puede tener un modelo organizado y retroalimentado de manera emergente (Figura 3).

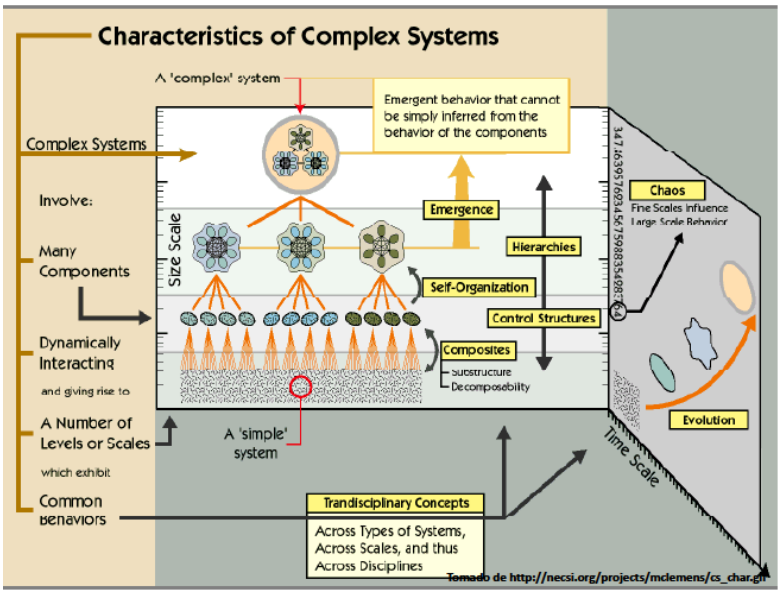

Figura 3: Características de Sistemas Complejos. Fuente: Instituto de Sistemas Complejos de N. Inglaterra

La mejor forma de sostener un desarrollo de fábrica de software es mediante un proceso de aprendizaje y refinamiento continuo, el cual se puede lograr a través de las cualidades de un sistema complejo, entonces en la creación de componentes se podría comenzar a aplicar cada uno de estos conceptos:

3.1 ORGANIZACIÓN

Para el desarrollo de cada componente de software es necesario organizar el grupo de trabajo en estructuras que interactúan y cambian con respecto al objetivo inmediato, esto a razón que puede resultar conveniente distribuir, jerarquizar o contraer grupos de trabajo de acuerdo a las capacidades de las personas, la cantidad de horas/persona, o el tipo de producto a obtener (Figura 4). Con esta cambiante distribución, los flujos de conocimiento se vuelven realmente dinámicos y fuertes, porque el conocimiento posee varias direcciones y llega a todos los individuos de manera natural.

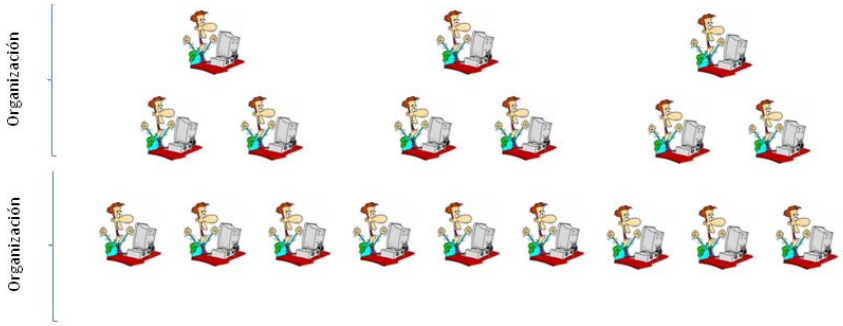

Figura 4: Formas de organización. Fuente: Los autores

3.2 INTERACCIÓN

Además de la interacción que se logra entre cada una de las personas que realizan el proyecto, se obtienen otros tipos de interacciones como la generada a través de conversaciones con clientes, directivos, compañeros e incluso con personas ajenas al proyecto y eventos del mundo exterior (Figura 5), lo que hace que el sistema sea un sistema abierto en el cual la inspiración para resolver inconvenientes puede provenir tanto del mismo sistema como de eventos exteriores. La distribución de información aumenta las capacidades de las personas para solucionar problemas.

Figura 5: Tipos de interacción. Fuente: Los autores

3.3 FEEDBACK

Cada uno de los eventos generados y las interacciones produce una nueva retroalimentación que apoya todos los procesos (Figura 6).

Figura 6: Eventos y retroalimentación. Fuente: Los autores

3.4 ADAPTACIÓN

El sistema, basándose en la retroalimentación consigue como respuesta la adaptación tomando decisiones y permitiendo un mejor acoplamiento a partir de modificaciones o cambios en los métodos, en los procesos y en el entorno. Los eventos externos e internos y la retroalimentación funcionan como incentivos para lograr armonía entre la forma actual de desarrollo de componentes y la mejor forma teniendo en cuenta todas las variables del entorno que intervienen en la sociedad (Figura 7). Los flujos de conocimientos se adaptan de igual forma, permitiendo que se tenga el conocimiento acerca del porqué del cambio y más tarde se pueda consultar una bitácora para entender las decisiones tomadas. Los eventos, la retroalimentación, las capacidades y métodos actuales sufren una adaptación que causa un nuevo proceso mejorado.

Figura 7: Proceso de adaptación. Fuente: Los autores

3.5 EVOLUCIÓN

Todos los cambios generados por la organizaciòn, la interacción, la retroalimentación y la adaptaciòn forja con el pasar del tiempo la optimizaciòn del sistema como tal y de los procesos de gestiòn del conocimiento, logrando un mejoramiento continuo que conlleva a la evolución (Figura 8).

Figura 8: Mejoramiento continuo. Fuente: Los autores

3.6 ETAPAS Y PAQUETES

Cada uno de los procesos vistos anteriormente se aplican a la gestión de conocimiento de la elaboración de los paquetes o componentes, y este conjunto de procesos se pueden repetir una y otra vez en el desarrollo de dicho paquete (Figura 9), por lo cual se forma diferentes etapas en donde en las ultimas, se verá con mayor precisión un aumento en la evolución.

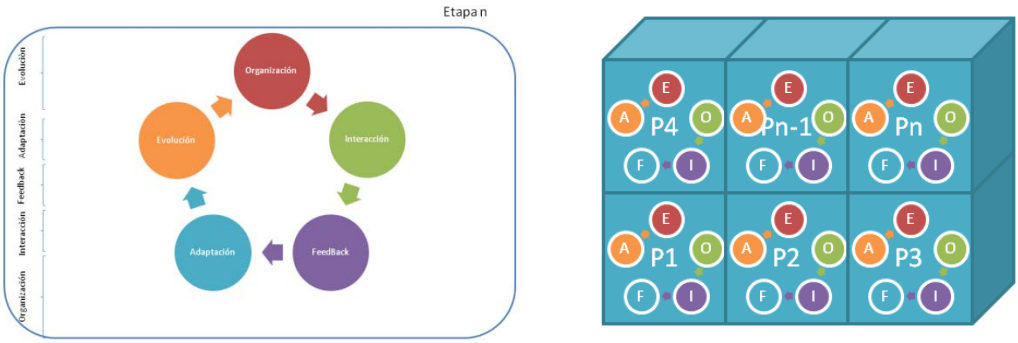

Figura 9: Procesos de la generación de paquetes y componentes. Fuente: Los autores

3.7 CONFORMACIÓN DEL MODELO

Al revisar cada uno de los anteriores conceptos es claro que en el desarrollo de un sistema complejo como lo es la fábrica de software, estos conceptos aplican en el desarrollo de un todo (Producto de Software) y a la vez en un elemento (Paquete). Cuando se comienza a desarrollar un paquete la organización, la interacción, el feedback, la adaptación y la evolución se adhieren a este proceso, y esto vuelve ocurrir cuando los conjuntos de paquetes se agrupan en una funcionalidad, permitiendo que el sistema llegue al punto de convertirse en complejo.

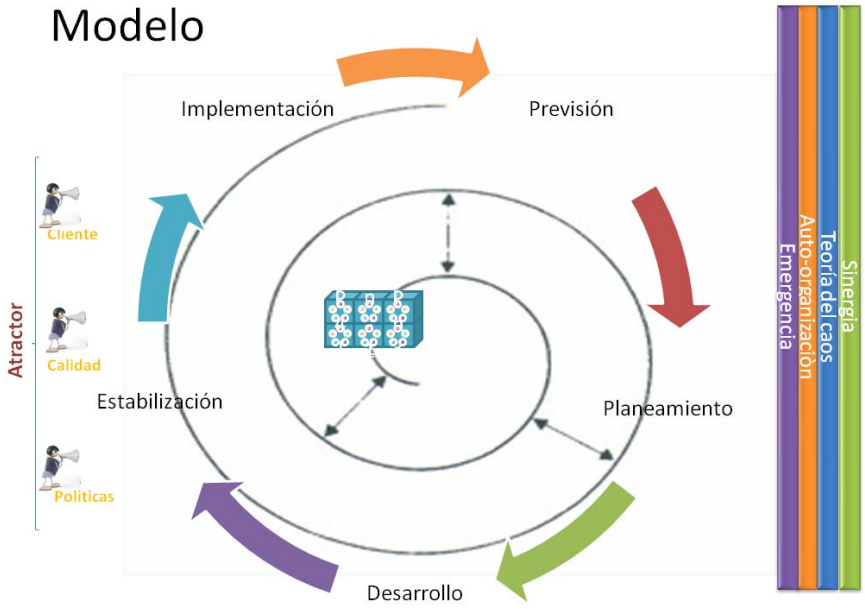

Figura 10: Modelo de gestión del conocimiento aplicado a un sistema complejo: Desarrollo de fábricas de Software. Fuente: Los autores

Una vez que ya se tengan cada uno de los paquetes, estos pasan a implementarse en la metodología MSF for agile Software Development de Microsoft bajo un enfoque de espiral, la cual contempla previsión, planeamiento, estabilización e implementación. Varios paquetes juntos representan una funcionalidad y esta a su vez unida con otras pasan a formar el producto de software, cuando todo esto ocurre es notable que existe un auto-organización basada en los eventos y retroalimentación obtenida anteriormente la cual puede responder a cualquier estimulo. Al

igual que la funcionalidad de cada paquete en particular puede convertirse en el logro de un objetivo basado en la colectividad, lo cual puede llamarse *Emergencia* y por ende al obtener cada función o procedimiento por separado nunca va a dar el mismo valor que si se tuviera el proyecto completo, porque 4 + 4 = 5 (Sinergia).

Pero aún cuando ya se puede ver toda la estructura, pueden aparecer algunos *atractores* como los *clientes, la calidad y las políticas,* formas y métodos con las que trabaja la organización, los cuales constituyen el piso firme y devuelven el horizonte al proyecto. Sin embargo es claro tener presente que debido a que el enfoque metodológico es un espiral, cualquier decisión tomada en las primeras fases repercute en las últimas y por consiguiente en el fracaso o éxito del proyecto, por esto se puede ver reflejada en este modelo la teoría del caos.

El modelo obtenido muestra que la gestión del conocimiento se implementa en los sistemas complejos porque para llegar a ser complejos se debe obtener, clasificar, distribuir, trasmitir y guardar información sacando el mejor provecho de la organización, interacción, feedback, adaptación y evolución (Figura 10).

4. CONCLUSIONES

Tanto la gestión del conocimiento como las fábricas de software son sistemas complejos, lo cual hace que la elaboración de un modelo de gestión de conocimiento para un sistema complejo, también sea complejo. El aplicar los conceptos de complejidad a un proceso de desarrollo de software nos muestra que los flujos de conocimiento pueden ser dinámicos, abiertos y distribuibles.

Los conceptos complejos fundamentales aplicados a un modelo de gestión de conocimiento del sistema complejo, desarrollo de fábrica de software son:

- La suma de las partes es mayor que el todo. Es un conjunto o combinación de cosas o partes, que forman un todo complejo o unitario. Sinergia.
- Está compuesta de varios elementos que reunidos pueden generar determinado comportamiento. Auto-organización-Emergencia.
- El desarrollo iterativo e incremental, que esta contiene, puede llevar a que decisiones erróneas en la primera interacción repercutan en todo el sistema. Teoría del caos.

REFERENCIAS

Lloyd, Seth. (2010). *http://en.wikipedia.org/wiki/Seth_Lloyd. Consultado noviembre de 2010*

La gestión del conocimiento. (2010) [En línea] http://www.gestion-conocimiento.com/. Consultado 2 de octubre de 2010.

Quintas Paul, Lefrere Paul and Jones Geoff.(1997). *Knowledge Management: a Strategic Agenda* : Long Range Planning. pág. 385 a 391. Vols. Vol 30, No.3.

Carrión, J.(2001). *Nuevos modelos en Internet para gestionar el talento Conjugación eficiente de las competencias estratégicas, técnicas y conductuales, con el entorno y la organización, para el logro de un desempeño superior. y el conocimiento.*

Sveiby, Karl Erik. *(2000). Capital Intelectual: La nueva riqueza de las empresas. Cómo medir y gestionar los activos intangibles para crear valor.* s.l. : Gestión 2000.

Forum, (2004). The European Knowledge Management**.**

gestion-conocimiento.com. (2010). [En línea] www.gestion-conocimiento.com.

Medina, Victor. (2010). Sistemas de Gestión del Conocimiento. Notas de clase. Universidad Distrital Francisco José de Caldas.

Moses, Joel (2001). "Complexity and Flexibility"**.***Professor of Computer Science and Engineering, MIT/ESD.*

Senge, Peter (1995). "The Fifth Discipline**"**.*Senior Lecturer, Organization Studies, MIT/ESD.*

Sussman, Joseph. (2007). "The New Transportation Faculty"**.** *Professor of Civil and Environmental Engineering, MIT/ESD.*

Greenfield, J. (2005). *Software Factories.*

Pohl, K., G. Bockle, and F.v.d. Linden. (2005). *Software Product Line Engineering.* s.l. : Springer.

CrnkovicIvica, LarssonMagnus.(2000). *A Case Study: Demands on Component-based Development.* s.l. : ACM. International Conference on Software Engineering:Proceedings of the 22nd international conference on Software engineering.

EstublierJacky and Grenoble France**.**(2005). *Reuse and Variability in Large Software Applications. Foundations of Software Engineering: Proceedings of the 10th European software engineering conference held jointly with 13th.* Lisbon, Portugal : ACM.

Barachisio Liana Lisboa, Marques NascimentoLeandro, Eduardo Santana de Almeida, Silvio Romero de Lemos Meira.(2004). *A Case Study in Software Product Lines: An Educational Experience. Federal University of Pernambuco and C.E.S.A.R, Recife, PE, Brazil. s.l. : 21.* Brazil : s.n.

Short, Jack Greenfield and Keith. (2004).*Software Factories-Assembling Applications with Patterns, Models, Frameworks, and Tools.* s.l. : Wiley Publishin.

Kleppe, A., J. Warmer, and W. Bast**.**(2003). *MDA Explained The Model Driven Architecture: Practice and Promise.* s.l. : Addison Wesley.

Pressman, R. (2006). *Ingenieria de Software un enfoque práctico.* McGraw Hill.

Lano, K. *(2005). Advanced System Design with Java, UML and MDA* . s.l. : Elsevier.

Warmer, J. and A. Kleppe. (2003). *The Object Constraint Language, getting your models ready for MDA.* s.l. : A. Wesley.

Autorización y Renuncia

Los autores autorizan a LACCEI para publicar el escrito en los procedimientos de la conferencia. LACCEI o los editores no son responsables ni por el contenido ni por las implicaciones de lo que esta expresado en el escrito

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.