Universidad Nacional Autónoma de México

Programa de Posgrado en Ciencias de la Administración

T e s i s

Modelo de ingeniería de software con base en directrices de administración del conocimiento

Que para obtener el grado de:

Maestro en Administración (Administración de la Tecnología)

Presenta: Rolando Lima Maciel

Tutor: Carlos Eduardo Puga Murguía

México, D.F. **2012**

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

Esta tesis trata temas relacionados con el conocimiento y un aspecto fundamental que aprendí en mis estudios de maestría es que el conocimiento realmente adquiere valor cuando se comparte, porque permite enriquecerlo y hacerlo crecer, por lo tanto el esfuerzo en la realización de este trabajo de nada serviría si se quedara archivado, razón que me lleva agradecer en primera instancia a todas las personas que por un motivo o por otro, han leído, están leyendo o algún día leerán este trabajo, ya que para todos ustedes es para quién fue creado.

A mis padres: Silvia y Darío, a mis hermanos: Silvia y Enrique, a mis abuelos, tíos y primos, les estoy profundamente agradecido por ser un soporte fundamental durante la realización de este proyecto, convirtiéndose en una motivación para materializar el trabajo.

A mis amigos, compañeros y otras personas con las que he podido compartir el recorrido de un camino llamado "vida", les quiero pedir una disculpa por no enlistarlos ya que no quisiera cometer la falta de omitir a alguno de ustedes, además de que creo que tengo tantas cosas que agradecerles que necesito escribir un libro completo para transmitirles todo lo que han representado en mi mundo, sobre todo durante estos últimos 4 años los cuales han sido un periodo de transición constante en para mi, y ustedes han estado presentes para brindarme su apoyo, paciencia, ejemplo y enseñanzas que me han llevado disfrutar mucho de mi andar en el tiempo así como a superar los momentos complicados que se han presentado.

Al Doctor Carlos Eduardo Puga Murguía le quiero agradecer por aceptar dirigir esta tesis así como por todas sus aportaciones que fungieron como guía para lograrla. Así mismo a mis sinodales: Dra. Graciela Bribiesca Correa, Dr. Luis Alfredo Veldéz Hernandez, M.A. María del Rocío Huitrón Hernández y M.A. Stephen García Garibay, les quiero agradecer su tiempo invertido para revisar y mejorar este trabajo.

Antes de llegar a la versión final de esta tesis que les entrego tuve el apoyo de diversos amigos y compañeros para revisar desde la estructura y ortografía hasta para desarrollar el contenido de la misma mediante la revisión y evaluación del modelo que propongo a todos ustedes les agradezco de manera especial por sus aportaciones tan significativas a: Silvia, Fabiola, Sandra, Ireri, David, Jose Luis, Moy, Lalito, Carry y Gabriel.

A mis jefes y compañeros de trabajo quiero reconocerles las facilidades que me brindaron para lograr mis metas personales y académicas.

A los responsables del Segundo Congreso de Alumnos de Posgrado de la UNAM así como a los del XVII Congreso Internacional de Contaduría Administración e Informática, les agradezco por la oportunidad de haberme permitido presentar la investigación derivada de esta tesis ya que resultó una experiencia muy valiosa mediante la cual me fue posible obtener retroalimentación del trabajo realizado.

Hay una persona que resultó un factor determinante para que yo decidiera ingresar a la UNAM y que además me enseñó a valorarla, respetarla y amarla desde cuando yo sólo era un niño e inclusive aún no tenía conciencia de la grandeza de la institución, esta persona fue el Arquitecto Antonio Attolini Lack, quién hace sólo algunos meses dejó de acompañarnos en este mundo para convertirse en leyenda, aportando un gran legado con todas sus obras con las cuales siempre puso en los primeros planos del país a la UNAM, a usted le quiero agradecer donde quiera que se encuentre por ser mi mayor referente del orgullo y dignidad con la que siempre hay que difundir a nuestra máxima casa de estudios que tanto me ha dado a mi y todo el país: LA UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

Rolando Lima Maciel

14 de Octubre de 2012

Índice

Índice de figuras

Índice de tablas

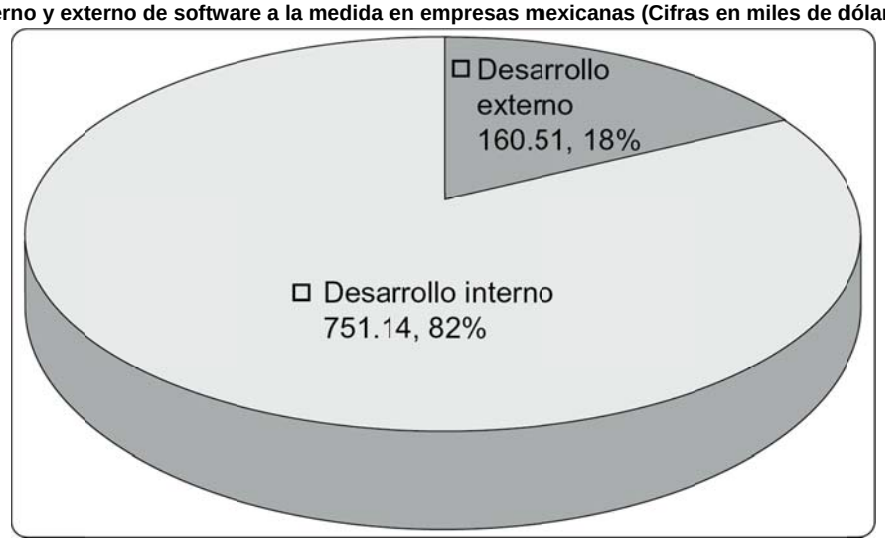
Introducción

Los sistemas de software desarrollados a la medida (adaptación de software estandarizado a las necesidades de los usuarios) se han convertido en un elemento primordial en las operaciones diarias de las organizaciones. Tal como lo mencionan Salem Ben y Mouna Ben (2009), a la fecha se han dedicado gran cantidad de trabajos orientados a los sistemas de información e ingeniería de software con el propósito de analizar los motivos de los retrasos en proyectos, presupuestos arriba de lo planeado, productos de software de baja calidad e inclusive con funcionalidad inadecuada. Desde los años 60's se emplea la expresión *crisis de software*, para hacer referencia a los problemas anteriores. Y se han realizado propuestas para aumentar la productividad y calidad del software al nivel requerido por el entorno en el que se encuentran inmersas actualmente las organizaciones.

Las soluciones aplicadas hasta el momento están centradas en mejoras al ciclo de vida del software con base en herramientas de desarrollo, métodos o procesos. Sin embargo, el alcance de estas soluciones suele estar limitado por enfocarse en problemas específicos y los beneficios que aportan no resultan significativos debido a que no toman en cuenta el hecho de que el software es resultado de la acumulación del conocimiento poseído por las personas que forman las organizaciones, dejando de lado que la *crisis del software* se puede mitigar disminuyendo la brecha de conocimiento existente entre lo que saben las personas que forman una organización y el conocimiento contenido en el software, por lo cual se requiere un modelo que permita integrar de forma eficiente la mayor cantidad del *know-how* de las organizaciones, manejando el proceso de desarrollo de software desde una perspectiva de la administración del conocimiento.

Los resultados presentados por Mochi Alemán (2006) en el estudio de "La industria del software en México en el contexto internacional y latinoamericano", elaborado por el Centro Regional de Investigaciones Multidisciplinarias de la UNAM, indican que la industria del software en México se encuentra centrada fundamentalmente en la producción de software a la medida, por lo tanto el sector está ligado por su propia naturaleza a las actividades de servicios, pero no prevalecen empresas especializadas en desarrollo de software para brindar tal servicio sino que la mayor parte de las necesidades de los grandes usuarios (sector público y empresas) son resueltas por el autoconsumo de los mismos usuarios a partir de departamentos internos de software abocados a esas tareas. Esta situación se puede apreciar en la Figura I donde se muestra que de un total de 911.65 millones de dólares invertidos para el desarrollo de software a la medida, el 82% fue implementado por áreas internas de desarrollo de software y el 18% mediante contratación de empresas especializadas. Tal disparidad se puede explicar debido a que existe la tendencia de considerar que con áreas internas para la implementación de software a la medida es posible llevar a cabo los proyectos en menor tiempo debido a que al contratar compañías externas se debe considerar tiempo de la curva de aprendizaje necesaria para que se involucren con las reglas de negocio de cada empresa y por lo tanto se acentúa el hecho de la importancia de administrar el conocimiento de las organizaciones.

Figura I. Desarrollo interno y externo de software a la medida en empresas mexicanas (Cifras en miles de dólares).



Fuente: Elaboración propia a partir de datos de Mochi Alemán, P. O. (2006*). La industria del software en México en el contexto internacional y latinoamericano*. Cuernavaca: Centro Regional de Investigaciones Multidisciplinarias, UNAM.

Dada esta realidad del país, el modelo propuesto tiene una orientación hacia el desarrollo de software a la medida, no sólo por la necesidad inherente de administrar el conocimiento relacionado con el mismo, sino también por ser en el que se invierte más en México. En cuanto al mercado de las empresas en las que el modelo será aplicable se tiene la restricción de que sean organizaciones con áreas internas para el desarrollo de software, o bien sean compañías que se dediquen completamente al desarrollo de software, lo anterior también tomando en cuenta que los principales mercados a los que se destina la implementación de software a la medida en México, según lo indicado por Mochi Alemán (2006), son muy diversos entre sí (como se puede apreciar en la tabla I), por lo tanto no se considera orientar el modelo a un mercado específico además de que los procesos de ingeniería de software y administración del conocimiento pueden ser estructurados de manera independiente al giro de las empresas donde sean aplicados.

Tabla I. Distribución de mercados que implementan software a la medida en México.

Mercados	%
Manufactura/ Extracción	26.8
Informática y telecomunicaciones	19.8
Seguros y servicios financieros	19.8
Comercio / Distribución	15.0
Gobierno	7.9
Otros servicios	7.7
Otros mercados	3.0

Fuente: Mochi Alemán, P. O. (2006). *La industria del software en México en el contexto internacional y latinoamericano*. Cuernavaca: Centro Regional de Investigaciones Multidisciplinarias, UNAM.

Respecto al tamaño de las empresas en las que el modelo será aplicable se están considerando organizaciones medianas y grandes, para las cuales el impacto de no administrar el conocimiento embebido en el software resulta mayor, pero no implica un impedimento para ser utilizado en organizaciones micro o pequeñas realizando los ajustes para simplificarlo de forma que sea funcional para este tipo de empresas.

Conforme a lo anterior la investigación desarrollada es no experimental, exploratoria, transversal con uso de un instrumento cualitativo de recolección de información. Las etapas en que considera su desarrollo son: identificación del problema, análisis bibliométrico, investigación documental, formulación de hipótesis, elaboración del marco teórico, diseño de la propuesta del modelo de ingeniería de software con base en directrices de administración del conocimiento, evaluación del modelo por expertos y ajustes al modelo.

Antes de presentar la propuesta del modelo, se tratarán los elementos teóricos y conceptuales que conformarán la estructura del mismo, dentro del Capítulo 1 se abordan: los temas correspondientes a la **administración del conocimiento** comenzando con los elementos básicos que dan lugar al conocimiento, así como una clasificación del mismo; algunos de los autores más representativos y sus aportaciones; una serie de definiciones del concepto de la administración del conocimiento para identificar los elementos comunes más relevantes que serán considerados para el planteamiento del modelo, las escuelas del conocimiento que aportan una clasificación de diversas estrategias de apoyo para iniciar un proyecto de implantación de administración del conocimiento, y algunas propuestas de procesos involucrados con el conocimiento de forma que sean seleccionados los que incluirá el modelo.

El Capítulo 2 considera las cuestiones relacionadas con la **ingeniería de software:** la definición de software, tanto desde una perspectiva *clásica* con base en elementos técnico como desde un enfoque de administración del conocimiento; una clasificación de los diversos tipos de software, las fases de la ingeniería de software, algunos de los modelos más conocidos para ejecutarla y los elementos de las metodologías ágiles de desarrollo de software.

El Capítulo 3 contiene los componentes de la metodología para la presente investigación, comenzado por el planteamiento del problema, preguntas y objetivos así como la hipótesis de trabajo, posteriormente se expone la taxonomía del estudio así como la explicación de las etapas de la investigación, después se abordará el proceso de elaboración de la propuesta del modelo de ingeniería se software con base en directrices de administración del conocimiento, definiendo el concepto de modelo y un procedimiento para construirlo, para finalmente presentar la propuesta inicial del modelo y la evaluación del mismo por expertos en el área de desarrollo de software a la medida.

Los puntos más destacables de la retroalimentación realizada por los expertos son la buena aceptación que tubo el modelo por su parte, ya que mencionaron que pudieron comprender los elementos teóricos sin mayor problema y consideran que el enfoque que plantea es apropiado

y necesario de aplicar en sus ambientes laborales, siempre y cuando se tome en cuenta las posibles barreras culturales debido a la resistencia al cambio y a compartir el conocimiento.

Lo anterior llevó a la consideración de algunos elementos complementarios que se presentan dentro del Capítulo 4 en el que se plantea la versión final del modelo aplicando las demás observaciones realizadas por los expertos, entre ellas hacer uso de la técnica de redundancia de información repitiendo algunos conceptos tratados en Capítulos anteriores para permitir una comprensión fácil y sencilla del modelo final, así mismo con esto se permite que los lectores que sólo estén interesados conocer el modelo final puedan abordarlo directamente dejando a su consideración la revisión del marco teórico completo en el que se fundamenta el modelo así como el proceso implicado para su construcción.

Posteriormente se presentan las conclusiones de la investigación, la bibliografía consultada y una serie de apéndices con información complementaria al trabajo desarrollado.

La importancia del tipo de investigación considerada en este trabajo es que el modelo final propuesto fue resultado de un proceso enriquecido por personas que cuentan con una experiencia relevante en el campo profesional y que diariamente trabajan en el ambiente en el que se presentan las dificultades provocadas por el problema estudiado, lo cual permite reconocer y explotar su conocimiento tácito. Por otra parte, el hecho de que al sustentarse dicho conocimiento de estos sujetos por el sistema de valores en el que cada individuo está inmerso permitió contar con aportaciones desde puntos de vista diversos e identificar los puntos de convergencia; por ejemplo la aceptación del modelo en distintos contextos que a su vez tienen el factor común de la relevancia en México del software desarrollado a la medida por el porcentaje de inversión que se realiza en el mismo, ya que los informantes laboran en empresas bancarias, de consultoría de software, telecomunicaciones y comercio.

Por la flexibilidad y la propia naturaleza del tipo de investigación empleada es posible estudiar el problema desde otras perspectivas u horizontes, por ejemplo cambiar el perfil de los expertos entrevistados o bien realizar el seguimiento de la aplicación del modelo en alguna empresa, situaciones que debido a la limitación de recursos, sobre todo en cuanto al tiempo para elaborar el presente trabajo, no fue posible abordarlas, pero se deja el planteamiento abierto para que los lectores interesados puedan mejorar la investigación realizada.

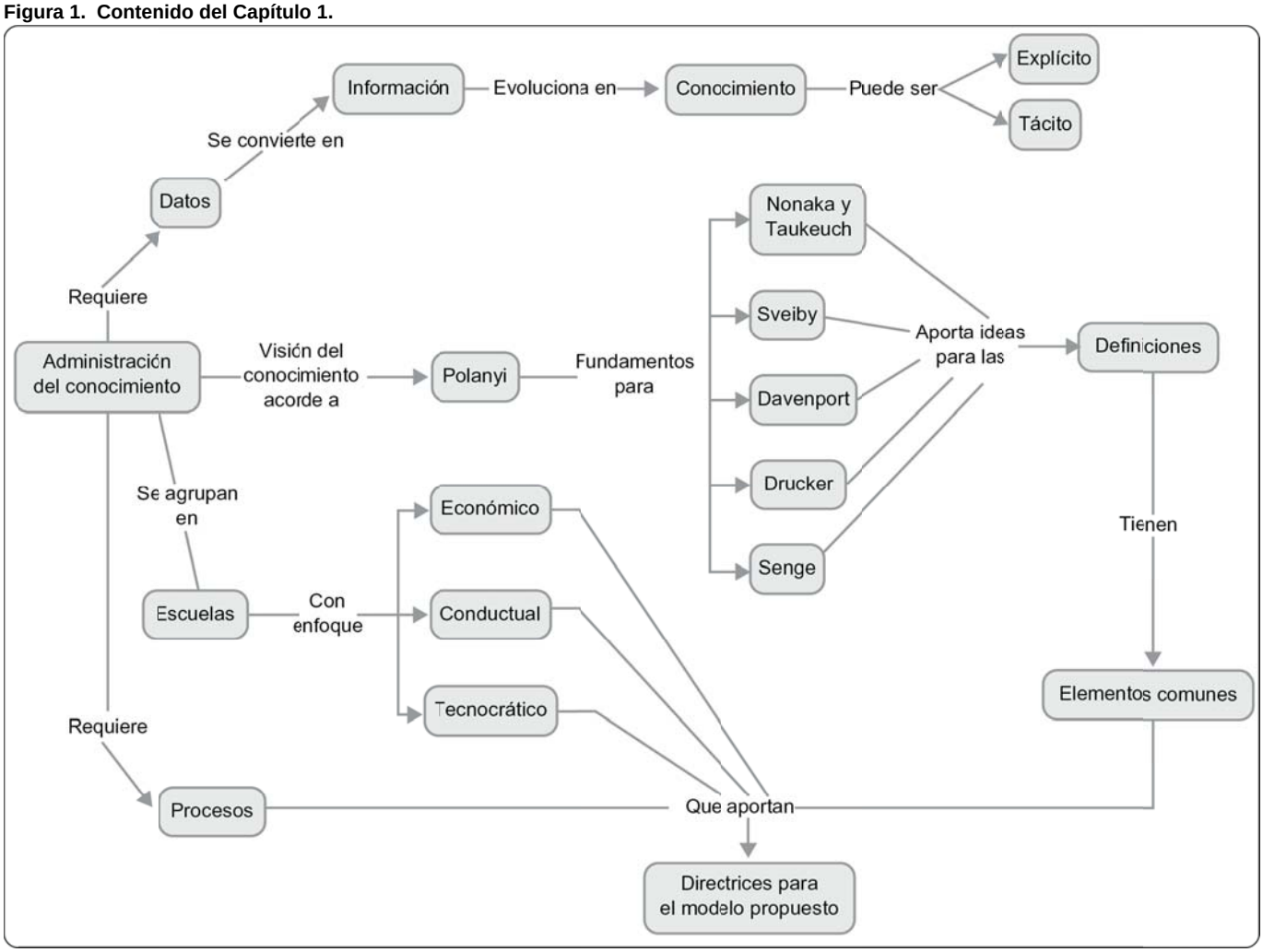
1. Administración del conocimiento

Si conoces a los demás y te conoces a ti mismo, ni en cien batallas correrás peligro; si no conoces a los demás, pero te conoces a ti mismo, perderás una batalla y ganarás otra; si no conoces a los demás ni te conoces a ti mismo, correrás peligro en cada batalla.

Sun Tzu

La intención de este Capítulo es presentar la estructura de la administración del conocimiento, inicialmente se aborda el tema del conocimiento partiendo de los conceptos básicos que lo componen como son datos e información y la manera en que se relacionan para obtener el conocimiento, así mismo se plantea el concepto de lo que se entenderá como conocimiento en este trabajo y dos formas de clasificarlo. Posteriormente, son presentadas las aportaciones de los autores más representativos en la administración del conocimiento, para después listar algunas definiciones que se manejan de su concepto e identificar los elementos clave de la componen. Como se podrá apreciar en la sección 1.5, no hay un consenso generalizado de lo que es la administración del conocimiento, por ello también presentamos una taxonomía en escuelas con distintos enfoques para poder tener una guía que permita seleccionar la estrategia de conocimiento más adecuada al modelo propuesto. El Capítulo finaliza presentando una serie de conjuntos de procesos considerados por algunos autores como los procesos que se deben seguir para ejecutar la administración del conocimiento, y de esos conjunto será seleccionado el que será aplicado al modelo. De forma gráfica los temas de este primer Capítulo y las relaciones entre ellos se presentan en la Figura 1.

Figura 1. Contenido del Capítulo 1.



Fuente: Elaboración propia.

1.1. Datos

Los datos son un conjunto discreto de hechos objetivos acerca de eventos. En el contexto empresarial pueden ser descritos como registros estructurados o transacciones. Carecen de sentido, porque describen sólo parcialmente lo que sucede y no proporcionan juicio, ni interpretación, tampoco permiten la toma de decisiones. Los datos en bruto no dicen lo que se tiene que hacer (Valhondo, 2003).

Realizando una analogía los datos serían tabiques apilados en bloques sin una secuencia o fin determinado, listos para ser empleados en las paredes que formarán una casa, una barda, una fuente, etc.

1.2. Información

Drucker (2000) considera a la información como datos dotados de relevancia y propósito y que a su vez otros investigadores la describen como mensaje, normalmente en forma de documento o comunicación visible o audible, así pues el mensaje debe informar.

A diferencia de los datos, la información tiene sentido, no sólo tiene el potencial de modelar al receptor, sino que en sí misma tiene forma, está organizada con algún propósito. Los datos se convierten en información cuando estos son:

- Contextualizados: Se sabe para qué propósito fueron recolectados.
- Categorizados: Se conocen las unidades de análisis o los componentes clave de los datos.
- Calculados: Los datos han sido analizados matemática o estadísticamente.
- Corregidos: Se han eliminados datos erróneos.
- Condensados: Los datos han sido resumidos, es decir, son más concisos.

Siguiendo con la analogía de la sección 1.1, la información representa una serie de paredes con un tamaño, forma y propósito establecido, construidas con los tabiques (datos) acomodados en una secuencia definida que constituye la estructura de una casa, estos tabiques fueron elegidos de manera que se eliminaron aquellos que no cumplen con las características necesarias para formar parte de las paredes o bien fueron modificados para que se ajustaran al diseño requerido.

1.3. Conocimiento

Davenport y Laurence (1998) mencionan que el conocimiento es una mezcla fluida de experiencias, valores, información contextual y apreciaciones expertas que proporcionan un marco para su evaluación e incorporación de nuevas experiencias e información. Se origina y

aplica en las mentes de los conocedores. En las empresas está presente no sólo en los documentos y bases de datos, sino también en las rutinas organizacionales, en los procesos, prácticas y normas.

El conocimiento se deriva de la información, y ésta se deriva de los datos mediante las siguientes acciones que tienen lugar en las mentes de las personas:

- Comparación: ¿Cómo se ajusta la información en la situación dada, comparada con otras situaciones conocidas?
- Consecuencias: ¿Qué implicaciones tiene la información para la toma de decisiones y acción?
- Conexiones: ¿Cómo se relacionan los fragmentos de conocimiento?
- Conversación: ¿Qué piensan otras personas acerca de la información?

El conocimiento sería la casa formada de las paredes mencionadas en la analogía del apartado 1.2, las cuales en conjunto definen los espacios que dan lugar a las habitaciones con cierta función, aunando a que las personas infieren su uso con base en la experiencia reconociendo los elementos que contienen cada una para establecer si es la concina, comedor, sala, etc.

1.3.1. Conocimiento explicito

El conocimiento explícito será aquel almacenado en medios físicos, codificado formalmente en bases de datos, documentos, correos electrónicos, esquemas, webs, etc. (Valhondo, 2003).

1.3.2. Conocimiento tácito

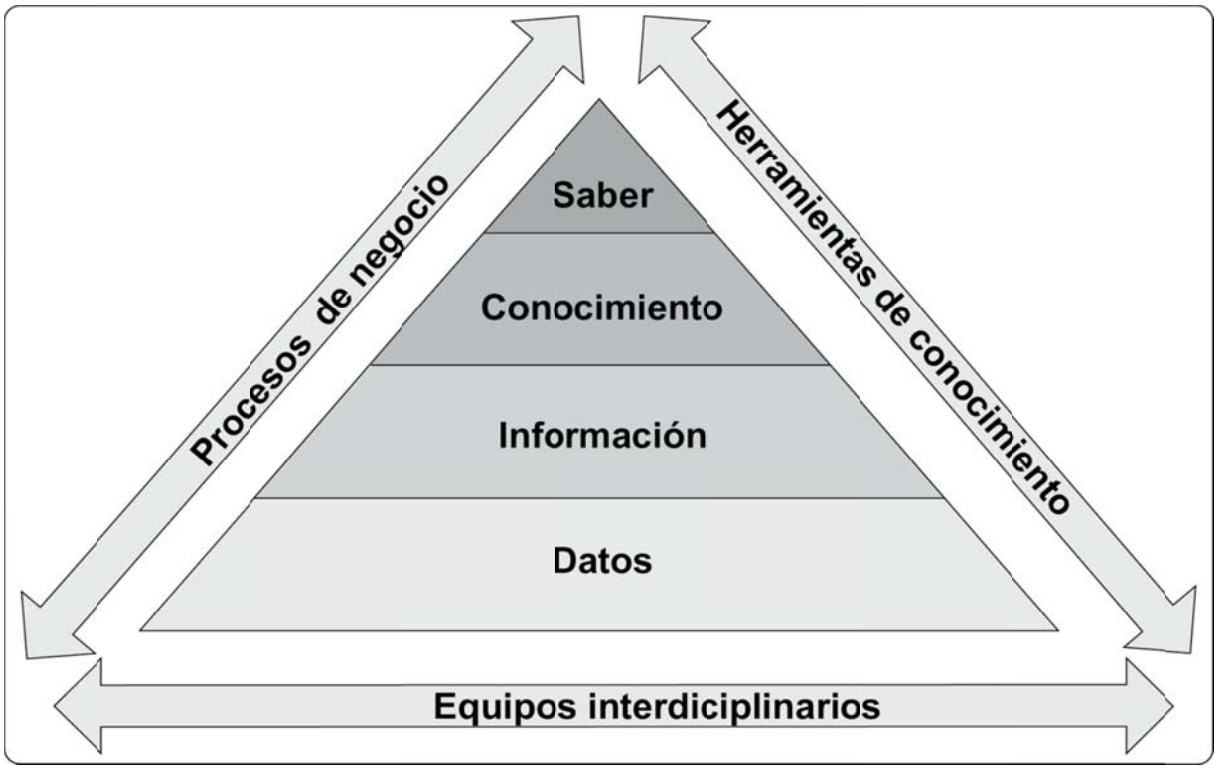
El conocimiento de tipo tácito es personal, almacenado en las mentes de los individuos, difícil de formalizar, registrar y articular, se desarrolla mediante un proceso de prueba y error que va conformando el conocimiento del individuo sobre diversos campos (Valhondo, 2003). Y es adquirido por las personas a través del tiempo mediante la combinación de la experiencia, conocimientos explícitos y la interacción otras personas.

1.4. Sabiduría

Valhondo (2003) menciona que algunos autores extienden la secuencia de datos, información y conocimiento hasta un nivel superior: Saber, conceptualizando la capacidad de comprender principios, como contraposición al conocimiento, que comprende patrones, y la información que comprende relaciones y cuya acumulación puede dar lugar, en términos más prácticos, al capital intelectual.

En relación a lo anterior *Petrotechnical Open Software Corporation* (POSC) plantea una pirámide mostrada en la Figura 2 que establece la relación entre datos, información, conocimiento y saber.

Figura 2. Pirámide de los datos al saber.



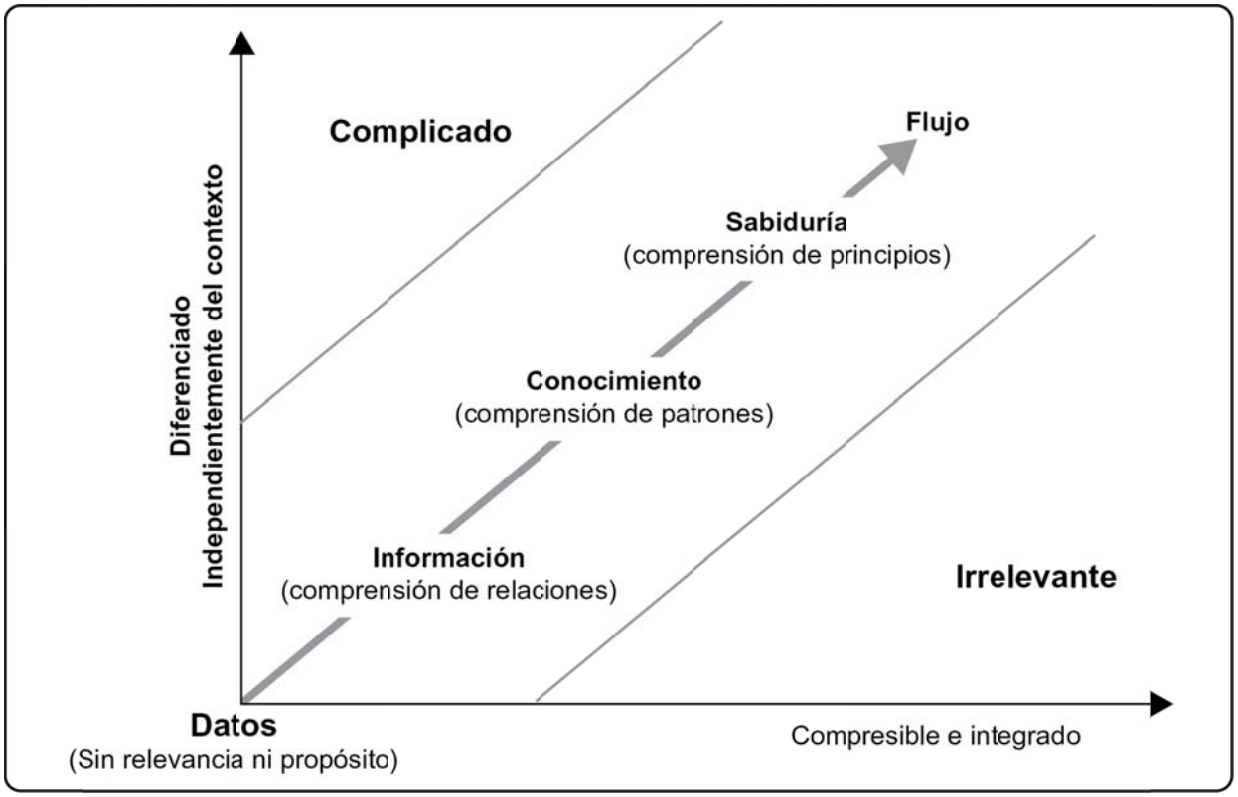
Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

Los tres niveles superiores de la pirámide en la Figura 2 responden a las siguientes preguntas:

- Saber: ¿Por qué?
- Conocimiento: ¿Cómo?
- Información: ¿Qué?, ¿Quién?, ¿Cuándo?, ¿Dónde?

Bellinger, Castro y Mills (2004) plantean que la secuencia ascendente desde los datos a la sabiduría se estructura como indica la Figura 3, con base en dos ejes uno de los cuales establece la diferenciación del conocimiento independientemente del contexto y el otro indica en qué grado es compresible e integrado, de forma que como ya se mencionó, los datos no tienen relevancia ni propósito hasta que son transformados en información permitiendo la compresión de patrones que prosperan en conocimiento el cual propicia el entendimiento de principios para “evolucionar” en sabiduría.

Figura 3. Secuencia de los datos a la sabiduría según Gene Bellenguer.



Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

1.5. Principales autores

En la sección 1.3 se mencionaron algunos autores para exponer los conceptos relacionados con el conocimiento, así mismo mediante el análisis bibliométrico (ver apéndice A) fueron identificados los que son citados con mayor recurrencia, tomando como referencia estos puntos y aunado a lo presentado por Valhondo (2003), a continuación se muestra un listado de los autores más relevantes en la administración del conocimiento con una breve descripción de sus respectivas contribuciones al tema.

1.5.1. Michael Polanyi

El erudito húngaro–británico Michael Polanyi fue el primero en plantear el conocimiento de la forma como se trabaja actualmente en la administración del conocimiento estableciendo las bases para las teorías actuales, sustentando su concepto en tres tesis claves:

i. Un descubrimiento auténtico no es explicable por un conjunto de reglas articuladas o de algoritmos.

ii. El conocimiento es público, pero también en gran medida es personal, ya que al estar construido por seres humanos contiene un aspecto emocional.

iii. Bajo el conocimiento explicito está el tácito. Todo conocimiento es tácito o está enraizado en el tácito.

Polanyi (1964) menciona que en cada actividad hay dos niveles o dimensiones del conocimiento complementarias:

- Conocimiento focal: Conocimiento sobre el objeto o fenómeno que observamos.
- Conocimiento tácito: Conocimiento utilizado como instrumento o herramienta para manejar o mejorar la interpretación de lo observado.

Polanyi (1964) identificó tres mecanismos sociales tácitos para la transferencia del proceso de conocer: imitación, identificación y aprendizaje por práctica, los cuales representan medios de traspaso directo del conocimiento, ya que delegan hechos, reglas y datos sin previo almacenamiento en un medio.

Según Polanyi (1964) la tradición como sistema de valores externos al individuo, describe cómo se transfiere el conocimiento en contexto social, ya que establece los modelos de acción, reglas, valores y normas, definiendo un orden social.

Estas ideas permiten resaltar la importancia del contexto e interacción social para desarrollar el conocimiento, pues no se compone únicamente de un conjunto estructurado de ideas y leyes comprobables, sino que para que el conocimiento pueda realmente denominarse como tal, se requiere del componente de la experiencia, de las personas redundando principalmente en la construcción del conocimiento tácito, el cual representa la base para cualquier tipo de concomimiento.

1.5.2. Ikujiro Nonaka y Hirotaka Takeuchi

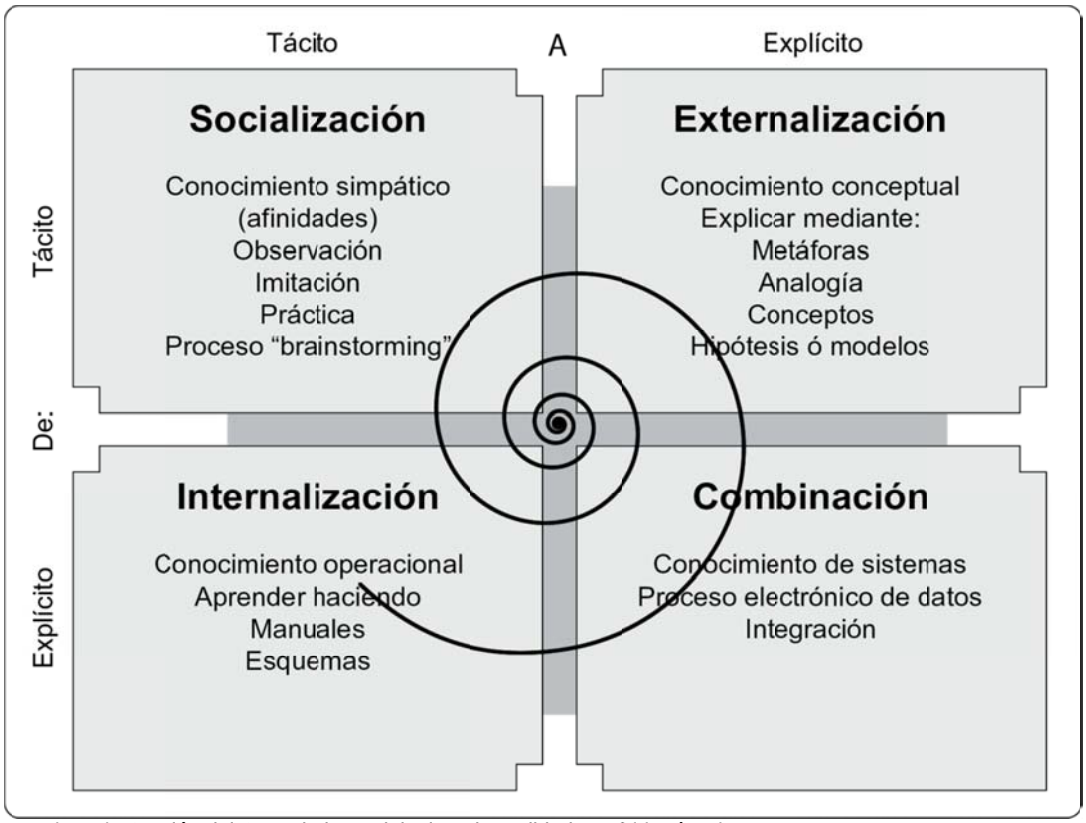
Los auntores japoneses Nonaka y Takeuchi (1995) profundizaron en los conceptos de conocimiento tácito y explícito, así como en la formulación de un esquema para crear conocimiento a través de un modelo basado en la interacción entre ambos tipos de conocimiento, planteando que todo conocimiento se produce a partir de los 4 procesos mostrados en la Figura 4, los cuales permiten transformar conocimientos tácitos a explícitos o viceversa, teniendo inclusive la posibilidad de enriquecer el conocimiento tácito de dos individuos mediante la socialización entre ambos o bien incrementar el conocimiento explícito por medio de la combinación de conocimientos codificados.

Adicionalmente, Nonaka y Takeuchi (1995) proponen que la construcción del conocimiento tiene comportamiento en forma de espiral ya que pasa por cada uno de los 4 procesos de manera iterativa haciendo crecer tanto el conocimiento personal como el público.

En los 4 procesos mostrados en la Figura 4, resalta la importancia de la interacción entre las personas para la construcción del conocimiento, siendo evidente sobre todo en lo que se refiere a la socialización y externalización, procesos que permiten ya sea de manera tácita o explicita poner a disposición de otras personas nuevos conocimientos. En lo referente a la internalización es posible visualizar una perspectiva individual en la que cada persona

selecciona de un conjunto de conocimientos codificados los que resultan relevantes para realizar determinadas tareas y los asimilan por medio de la experiencia para crear nuevos conocimientos tácitos que permitan realizar de manera exitosa actividades. La combinación plantea entes externos a las personas para integrar y procesar datos electrónicamente pero aun así no se puede crear conocimiento sin alguien que emplee los datos procesados para contextualizarlos e incorporarlos a conocimientos codificados con los cuales tengan sentido y sean enriquecerlos.

Figura 4. Espiral del conocimiento.



Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

1.5.3. Sveiby

El sueco Karl Erich Sveiby es considerado como unos de los padres de la administración del conocimiento ya que Sveiby (1997) se concentró en impulsar la administración del conocimiento desde una visión práctica en lugar que teórica, enfocada completamente al ámbito empresarial, y manifestó su convicción de que la administración del conocimiento es el arte de crear valor a partir de los activos intangibles.

Este mismo autor definió a las organizaciones del conocimiento como aquellas que están totalmente adaptadas a sus clientes. En estas empresas el servicio surge del proceso continuo de resolución de problemas entre los clientes y los equipos de expertos. Tratan a los clientes individualmente sin forzarlos a adaptarse al producto desarrollado, sino adaptando los productos a los clientes.

El personal clave de estas organizaciones es el que posee el conocimiento, tiende a ser muy competente, con amplia formación y/o experiencia profesional, lo cual refleja la nueva realidad de los empleos hoy en día, ya que para cumplir con sus actividades estos trabajadores requieren de mayor preparación y un ambiente apropiado para adquirir y promover el conocimiento del negocio, así como flexibilidad para realizar la adaptación de lo que requiere el cliente.

1.5.4. Tomas H. Davenport y Prusak Laurence

Davenport y Laurence (1998) en sus publicacions para *Harvard Business Pres* dedicaron especial atención a la distinción entre datos, información y conocimiento presentada en la sección 1.3. Por otro lado, consideraron el impacto en la administración del conocimiento de las tendencias actuales de la economía, como son la globalización, el cambio organizacional, la convergencia de productos y servicios, etc. enfocando gran atención a las personas.

Como se mencionará en la sección 1.6, un elemento primordial de la administración del conocimiento es el éxito de los negocios, es decir el aspecto económico, lo anterior debido a la nueva realidad para producir riqueza, ya que históricamente se ha transformado acorde a la estructura económica que dominaba el hombre, en un inicio el poder económico fue representado por la cantidad de tierras que una persona poseía, posteriormente fueron los medios de producción que permitían la creación de productos o servicios los que cobraron importancia y en la actualidad el conocimiento es lo que les permite a las empresas crear valor con mayor relevancia y dado que el conocimiento se encuentra en las personas es por lo que se comenzó a considerarlas con mayor atención siendo el motivo por el cual también resulta necesario administrar el conocimiento para que no se quede únicamente en las mentes de los empleados sino también en las organizaciones por lo cual cobran relevancia las ideas planteadas por Davenport y Prusak Laurence en cuanto al impacto de la administración del conocimiento a la economía.

1.5.5. Peter F. Drucker

El austriaco Peter F. Drucker asignó una gran importancia a las personas en las organizaciones, Drucker (2000) introdujo el concepto de trabajadores del cocimiento, definiéndolos cómo individuos que dan más valor a los productos y servicios de una compañía aplicando su conocimiento, siendo responsables de aportaciones que afecten la capacidad de la organización para realizar y obtener resultados, los puntos clave que definen a los trabajadores del conocimiento son:

- Se gestionan a sí mismos ya que necesitan tener autonomía.
- La innovación continua debe ser parte de su trabajo.
- Necesitan formación y aprendizaje continuo.

- Su productividad no se basa tanto en la cantidad como en la calidad.
- Debe de tratarse como activo de la empresa en lugar de un como gasto.

Estos planteamentos implican un cambio de paradigma respecto a la manera tradicional de manejar las empresas y se necesita gran madurez por todos los involucrados, pero también responden a la realidad actual en la cual los oficios requirieren personal cada vez más preparados para responder al dinamismo en que operan las organizaciones.

1.5.6. Peter Senge

Peter Senge es un personaje muy reconocido por su libro titulado: *La quinta disciplina,* este autor introdujo el concepto de organizaciones que aprenden en las que los empleados desarrollan su capacidad de crear los resultados que realmente desean y donde se propician nuevas formas de pensar, entendiendo la empresa como un proyecto común de manera que los empleados están constantemente aprendiendo a aprender. Estableció 8 características clave que una organización que aprende debe tener para lograr efectividad y éxito:

a) Gran compromiso con el aprendizaje.
b) Cultura de aprendizaje, desaprendizaje y reaprendizaje continúo.
c) Se permite pensar a las personas por sí mismas.
d) Observan el entorno para anticiparse al mercado.
e) Empleo de las tecnologías de información como herramienta facilitadora.
f) Fomentan el aprendizaje en equipo.
g) Traducen lo aprendido a la práctica.
h) Se hace notar que los esfuerzos de cada persona influyen en el futuro de la empresa.

Senge (2006) afirma que una empresa que aprende es aquella que está organizada de forma consistente con la naturaleza humana, y que desarrolla cinco disciplinas:

a) Pensamiento sistémico.
b) Dominio personal.
c) Modelos mentales.
d) Visión compartida.
e) Aprendizaje en equipo.

Cualquier organización es formada por personas, por lo tanto a partir del término de empresas que aprenden se puede inferir que: se requieren personas comprometidas con el aprendizaje no sólo individual sino colectivo, es decir, se necesitan trabajadores del conocimiento para desarrollar una empresa que aprende. Por otro lado, así como las tecnologías de información representan un elemento importante en diversas actividades diarias

del ser humano hay que considerarlas a partir de lo mencionado por Senge (2006) como herramientas para aumentar la eficiencia en los procesos del conocimiento.

1.6. Definiciones de administración del conocimiento

Hasta el momento ha sido presentados los conceptos relacionados con el conocimiento (sección 1.3) así como diversas aportaciones realizadas al tema (sección 1.5), pero no ha sido expuesta una definición formal de la administración del conocimiento, se puede decir que en la actualidad hay tantas definiciones como la cantidad de personas que han abordado el tema, es por ello que a continuación se presenta una recopilación con base en lo expuesto por Valhondo (2003) de la forma como conceptualizan algunos autores la administración del conocimiento, con la intención de resaltar los elementos comunes y más representativos de cada una de ellas:

- *Domingo Valhondo:* La administración del conocimiento consiste en las diligencias relacionadas con el conocimiento, conducentes al logro de un negocio.
- *Gene Meieran:* La administración del conocimiento tiene que ver con el uso de las computadoras y comunicaciones para ayudar a la gente a recopilar y aplicar sus datos, información, conocimiento y sabiduría colectivos, con el fin de tomar las mejores, más rápidas y efectivas decisiones.
- *Matthias Bellmann:* La administración del conocimiento es la transformación del conocimiento en negocios, aprendiendo mediante la transformación de información en conocimiento.
- *Karl Eric Sveiby*: La administración del conocimiento es el arte de crear valor mediante el afianzamiento de los activos intangibles. Por lo cual hay que ser capaz de visualizar la organización como algo que no es más que conocimiento y flujos de mismo.
- *Charles Armstrong:* La administración del conocimiento, tiene que ver con elevar la conductividad de la organización para mejorar la capacidad de enlazar con el mundo exterior y los clientes. Esto requiere crear el lugar, el tiempo y el ambiente apropiado para promover trabajo reflexivo y la efectividad de interacciones.
- *Robert K. Logan:* La administración del conocimiento está relacionada con el uso de la información estratégica para conseguir objetivos de negocio. Es la actividad organizacional de creación del entorno social e infraestructura para que el conocimiento pueda ser accedido, compartido y creado.
- *Gartner Group:* La administración del conocimiento es una disciplina que promueve el enfoque integrado de la creación, compartición y aplicación de información en una empresa. En lo que se refiere a la compartición incluye los procesos de captura, organización y acceso.
- *Bill Gates:* La administración del conocimiento no es más que gestionar los flujos de la información y dirigir la correcta a las personas que la necesitan de manera que sea posible hacer algo con ella.

A partir de la atención que enfocó el sector empresarial a la administración del cocimiento es que el tema cobró fuerza e importancia, por lo cual es posible apreciar que la mayoría de las definiciones anteriormente expuestas insinúan que el hecho de administrar el conocimiento tiene el fin de crear valor o bien generar negocios exitosos. Por otro lado, en lo referente al proceso del conocimiento resalta la importancia que se da al "flujo" del mismo así como a la información, de manera que se encuentre disponible para las personas que pueden transformarla y crear valor, lo cual tiene la consecuencia de aportarle ventajas competitivas a las empresas, situación que invariablemente lleva a tener mayor probabilidad de éxito en los negocios. Y finalmente, otro punto de recurrencia en las definiciones es que se enuncian procesos de creación, acceso, compartición y utilización del conocimiento. De acuerdo a lo anterior, los elementos clave de la administración del conocimiento que estarán presentes en el modelo de la sección 3.8.3 son:

- Perseguir la finalidad principal de aportar ventajas competitivas a las organizaciones, para aumentar el éxito en los negocios.
- Atender los procesos para crear, acceder, compartir y utilizar el conocimiento, o bien cualquiera de los conjuntos de procesos presentados en el apartado 1.8.
- Asegurar que el flujo de información y conocimiento llegue hasta las personas que pueden crear valor para las empresas.

1.7. Escuelas de administración del conocimiento

Earl (2001) elaboró una taxonomía de los enfoques de administración del conocimiento organizándolos en las escuelas mostradas en la Tabla 1. Menciona este autor, que el propósito principal de esta clasificación es tener una guía que permita definir el punto de partida en los proyectos de administración del conocimiento, acorde a la meta establecida, contando con las bases de la(s) escuela(s) que mejor cumplan los objetivos trazados, manejando el conocimiento como el producto principal del proceso de innovación, de manera que sea posible mejorar la toma de decisiones y renovación organizacional.

Cada una de las escuelas representa una orientación particular, con diferente tipo de intervención organizacional y a pesar de que son planteadas como una conceptualización idealizada no implica que sean excluyentes una con otra.

Como se mencionó en la sección 1.6 el interés hacia el que se ha orientado la administración del conocimiento son los negocios de las empresas y acorde a taxonomía de Earl (2001) se tiene la flexibilidad de enfocarse en un área o bien definir estrategias especificas, para lograr resultados en varios ámbitos. La Tabla 1 permite identificar los elementos fundamentales que se tienen que considerar para lograr el éxito de una escuela del conocimiento, la cual puede elegirse fácilmente de acuerdo al objetivo o enfoque que tenga una organización y de manera sencilla es posible saber cual será la unidad fundamental para el

esquema elegido así como los factores a considerar para el éxito y la filosofía que hay que seguir.

Tabla 1. Escuelas del conocimiento.

Escuela / Atributo	Tecnocrática			Económica	Conductual		
	Sistemas	Cartográfica	Ingeniería	Comercial	Organizacional	Espacial	Estratégica
Enfoque	Tecnología	Mapas	Procesos	Ingresos	Redes	Espacio	Actitud
Objetivo	Bases de conocimiento	Directorios de conocimiento	Flujos de conocimiento	Activos del conocimiento	Integración del conocimiento	Intercambio de conocimiento	Capacidad de conocimiento
Unidad	Dominio	Empresa	Actividad	*Know-how*	Comunidades	Lugar	Negocio
Ejemplo	Xerox Shorko Films	Brain& Co. AT&T	HP Frito-lay	Dow chemical IBM	BP Amoco Shell	Skandia British Airways	Skandia Unilever
Factores críticos de éxito	Validación de contenido Incentivos para aportación de contenido	Cultura o incentivos para compartir el conocimiento Redes para conectar a las personas	Conocimiento aprendido e información Distribución sin restricciones	Equipos especializados Procesos institucionalizados	Intermediarios para una cultura sociable del conocimiento	Diseño para estimular el conocimiento	Artefactos de retorica
Principal contribución de las TIC´s	Sistemas basados en conocimiento	Perfiles y directorios en intranets	Bases de datos compartidas	Activos intelectuales Sistemas de registro y procesamiento	Groupware Intranets	Herramientas de acceso y representación	Ecléctica
Filosofía	Codificación	Conectividad	Capacidad	Comercialización	Colaboración	Interacción directa	Conciencia

Fuente: Earl, M. (2001). *Knowledge Management Strategies: Toward a Taxonomy*. Journal of Management Information Systems, 215-233.

La Tabla 1 será considerada como guía para la selección de la estrategia del conocimiento empleada en el modelo propuesto en el Capítulo 3. A continuación se presentan las generalidades establecidas por Earl (2001) para las escuelas, cada una aporta ventajas desde un punto de vista muy particular de la administración del conocimiento y para realmente lograr una estrategia efectiva se requiere combinarlas de forma que se cubran completamente las necesidades acorde a un contexto particular.

1.7.1. Escuelas tecnocráticas

Acorde al planteamiento de Earl (2001) la escuela tecnocrática está compuesta por otras escuelas cuyas bases son las tecnologías de la información o la administración, apoyan en

menor o mayor grado a los trabajadores del conocimiento, en sus tareas diarias. Aportan el enfoque para trabajar la administración del conocimiento desde una perspectiva de tecnología, mapas o procesos.

Este conjunto de escuelas ofrecen mayores ventajas a las empresas dedicadas completamente a la creación de productos, ya que al englobar los aspectos referentes a tecnología, mapas de conocimiento y procesos, les permite trabajar con más eficiencia debido al uso de herramientas que facilitan la creación de bienes con mayor calidad para el usuario final, mediante una correcta integración del conocimiento y procedimientos de las áreas de negocio involucradas en la producción.

1.7.1.1. Escuela de sistemas

Ésta es la escuela más antigua de la administración del conocimiento así mismo es el enfoque que se trabaja con mayor recurrencia en la actualidad, Earl (2001) plantea que el *know-how* está compuesto por datos teóricos combinados con experiencia técnica, por ello la idea fundamental de esta escuela es capturar el conocimiento de los especialistas en bases del conocimiento que alimenten a sistemas expertos o intranets, de forma que otros especialistas o personas calificadas puedan hacer uso del mismo. También se considera el almacenamiento de mejores prácticas, así como soluciones a los problemas a enfrentados.

Es mencionado por Earl (2001) que la recompensa de las personas en esta escuela podría ser la fama o el reconocimiento de ser el autor de una solución que otros pueden consultar a través de los medios mencionados anteriormente. Bajo este esquema los trabajadores operativos se convierten también en trabajadores del conocimiento, ya que ellos realizan las tareas que redundan en los productos y/o servicios de las organizaciones.

Los factores críticos para que este enfoque de administración del conocimiento funcione, es en primer lugar, no perder de vista que el conocimiento no sólo es la acumulación de datos e información, sino también experiencia obtenida mediante entrenamiento y práctica. Por otro lado, se requiere considerar esquemas para recompensar a las personas por sus aportaciones a las bases de conocimiento ya que no para todos será suficiente con el reconocimiento de los demás. Y, finalmente hay que mencionar que esta escuela depende completamente de las tecnologías de la información, ya que se requiere una estructura que permita capturar, almacenar, organizar y distribuir el conocimiento de manera eficiente.

1.7.1.2. Escuela cartográfica

En este caso el objetivo es realizar un mapa del conocimiento organizacional, especificando quiénes son las personas que poseen determinado conocimiento mediante la elaboración de directorios del conocimiento. La idea principal es que las personas con la experiencia en un tema determinado estén accesibles para otros en caso de que sea requerido un consejo,

realizar una consulta o intercambio de conocimiento. Se maneja de esta manera debido a la consideración de que la forma más efectiva para transmitir el conocimiento es la comunicación directa entre las personas aportándole más valor. A diferencia de la escuela de sistemas en este caso se pretende incentivar el intercambio conocimiento entre las personas en lugar de solamente capturarlo en algún software, por este motivo se requiere que la gente dentro de este esquema cuente con una cultura de apoyo mutuo y compartir conocimiento trabajando bajo un propósito común dictado por la organización.

Earl (2001) considera que un directorio donde se indique quién posee conocimiento de un tema es más eficiente y efectivo, que realizar el mapeo del conocimiento organizacional en diferentes dominios. En la filosofía de esta escuela se puede apreciar cómo conectar a las personas para que intercambien su conocimiento.

1.7.1.3. Escuela de ingeniería

El enfoque de esta escuela surge a partir de la reingeniería de procesos, fundamentándose en que el rendimiento de éstos puede ser optimizado, si se proporciona el conocimiento requerido por el personal operativo para llevar a cabo sus actividades diarias, por lo cual se conceptualizan los procesos como actividades intensivas de conocimiento en lugar simplemente de acciones racionalizadas del negocio (Earl, 2001). En consecuencia se enfatiza el conocimiento y la mejora continua a través de reutilización del conocimiento, reconociendo que las tareas tienen componentes que suelen ser repetitivos.

Earl (2001) enuncia que si es posible aprender mediante experiencia y los trabajadores tienen acceso al conocimiento e información requerida para llevar a cabo sus actividades, el rendimiento y adaptación de los procesos administrativos así como del negocio serán mejorados significativamente.

1.7.2. Escuela económica (comercial)

La escuela comercial es clasificada por Earl (2001) dentro del bloque de económica porque está explícitamente enfocada en proteger y explorar el conocimiento o activos intelectuales de las organizaciones para producir utilidades, estos activos intelectuales comprenden patentes, marcas, derechos de autor y el *know-how,* comúnmente hace referencia al listado anterior como propiedad intelectual.

Bajo este enfoque se pone más atención en explotar el conocimiento que en explorarlo, ya que trabaja con la idea de que la diferencia entre el valor de mercado y en los libros de las organizaciones radica en el capital intelectual que cada una posee.

Para hacer funcionar esta idea se debe contar con especialistas que manejen de manera adecuada la propiedad intelectual y evitar el error de concentrarse más en medir el capital intelectual en lugar de desarrollarlo y explotarlo.

En México es claro que este tipo de escuela tiene una orientación que presenta mayores beneficios al sector farmacéutico debido al manejo de patentes; otros sectores aún no se han percatado de los beneficios del correcto manejo de la propiedad intelectual inclusive en ámbitos académicos no se le ha dado la importancia que amerita, ya que de una u otra manera es un medio para diferenciar productos de forma que ninguna otra compañía pueda ofrecerlo.

1.7.3. Escuelas conductuales

Estas escuelas enfocan su atención en el estímulo hacia los empleados para crear, compartir y utilizar el conocimiento. Con el propósito de formar redes de personas, establecer los espacios adecuados para el intercambio del conocimiento o bien asegurar que los empleados cuenten con una actitud de conocimiento.

Considero que la orientación de este conjunto de escuelas presenta mayores ventajas para las empresas de servicios, pero sobre todo para organizaciones dedicadas a cuestiones académicas ya que establecen con bases sólidas mecanismos para intercambio de conocimiento mediante la interacción social.

1.7.3.1. Escuela organizacional

Mediante esta concepción se pretende describir el uso de la estructura organizacional o redes, de forma que sea posible compartir e integrar el conocimiento mediante las llamadas comunidades de conocimiento, las cuales por si mismas establecen redes de comunicación tanto tecnológicas como sociales, que permiten enlazar a las personas con preguntas con las personas poseedoras de las respuestas.

Debido a la complejidad de que en una base de conocimiento sea codificado la totalidad de conocimiento de una persona se pueden agregar registros que permitan rastrear a quienes poseen más experiencia en alguna temática en particular y establecer los mecanismos de interacción con la gente que requiere ese conocimiento.

Los elementos básicos para que esta escuela funcione correctamente requieren que las personas que forman las comunidades de conocimiento tengan una cultura de trabajo en redes así como establecer un moderador a cargo de controlar las interacciones entre los miembros de las comunidades del conocimiento (Earl, 2001).

1.7.3.2. Escuela espacial

Los elementos de esta escuela están centrados en uso del espacio físico o bien en establecer un diseño del mismo que facilite el intercambio de conocimiento, por lo tanto se enfoca en fomentar la socialización como un medio para lograrlo, con base en las interacciones formales e informales entre las personas (Earl, 2001).

1.7.3.3. Escuela estratégica

Mediante esta escuela se visualiza la administración del conocimiento como una dimensión competitiva de la empresa, inclusive considerándola como la esencia de cualquier estrategia de la misma, desarrollando las competencias de los empleados al mismo tiempo que se captura y comparte el aprendizaje así como el *know-how*. El objetivo de esas consideraciones es construir, nutrir y explotar completamente los activos del conocimiento por medio de sistemas, procesos y las personas que los convierten en productos y servicios (Earl, 2001).

Este enfoque se compromete en incrementar la conciencia acerca del valor del conocimiento como una fuente para desarrollar más posibilidades para las organizaciones.

1.7.4. Escuela de administración del conocimiento para el modelo propuesto

El punto de partida para desarrollar el modelo que se presentará en la sección 3.8.3 es la escuela de sistemas, ya que como será especificado en los Capítulos 2 y 3: se pretende organizar el conocimiento contenido en el software y enlazarlo con los procesos de negocio; el enfoque de esta escuela es la tecnología lo cual concuerda con el perfil de lo que representa el software; la filosofía manejada es la codificación.

Es importante indicar que se tiene la intención de que con el modelo sea posible codificar el conocimiento desde una perspectiva del *know-how* de negocio para establecer elementos claros que permitan identificar qué conocimiento está contenido en los diversos elementos de software con los que operan diariamente las organizaciones.

Al seleccionar la escuela de sistemas el modelo, se considerará de manera importante la interacción con las tecnologías de la información para cumplir su cometido, así como otro elemento relevante señalado en las secciones 1.5 y 1.6: la importancia de la interacción social directa entre las personas para propiciar el flujo del conocimiento como fue mencionado.

La taxonomía de la Tabla 1 permite combinar las diversas escuelas, por lo tanto también se considerará el enfoque de la escuela cartográfica para solventar la necesidad de establecer redes de conexión entre las personas y sea posible abarcar la espiral del conocimiento planteada en el apartado 1.5.2.

1.8. Procesos del conocimiento

Para que los procesos de administración del conocimiento tengan sentido se requiere de la existencia de al menos dos personas, ya que los procesos surgen de la necesidad de interacción entre seres humanos para la consecución de objetivos comunes, desencadenando que los individuos intercambien conocimiento.

A continuación se presentan una serie de procesos consideraros por algunas instituciones, si bien al igual que en el caso de las definiciones de administración del conocimiento no hay consenso universal, sí se tiene mayor similitud entre ellos y en algunos casos sólo cambia el hecho de la integración de ciertas actividades.

1.8.1. Guía europea.

García Garibay (2010), hace referencia a la guía europea de buenas prácticas de administración del conocimiento, donde se detallan las siguientes funciones:

- *Creación de conocimiento*: Este proceso se refiere a motivar a las personas a nivel personal y a las organizaciones a nivel estratégico, a reflexionar sobre lo que quieren conseguir y el conocimiento que se requiere para hacer que suceda se recomienda incluir un análisis de los conocimientos disponibles y el conocimiento que falta, con el fin de fomentar la reutilización de los conocimientos actuales.
- *Organización del conocimiento*: Implica el almacenamiento y se refiere a la integración del conocimiento dentro de la empresa. Una gran parte del conocimiento se almacena en los cerebros de las personas, el cual se ve reflejado en el trabajo en equipo y las rutinas organizacionales, conocimiento que no necesariamente se hace explicito, pero que es accesible para su reutilización en la organización y se le denomina memoria organizacional.
- *Transferencia del conocimiento*: Es distribuir el conocimiento en el lugar correcto, en el momento indicado con la calidad adecuada, con el fin de crear valor. La transferencia del conocimiento se puede realizar de diversas formas: utilizando herramientas tecnológicas como base de datos o sistemas informáticos; o con la interacción directa de las personas a través de la colaboración, talleres, entrenamiento, etc.
- *Utilización del conocimiento*: Consiste en asegurar que el esfuerzo de los procesos anteriores se lleve de forma eficaz y eficiente a aplicaciones concretas, además de determinar las necesidades de conocimiento que serán la referencia para la creación de nuevos conocimientos.

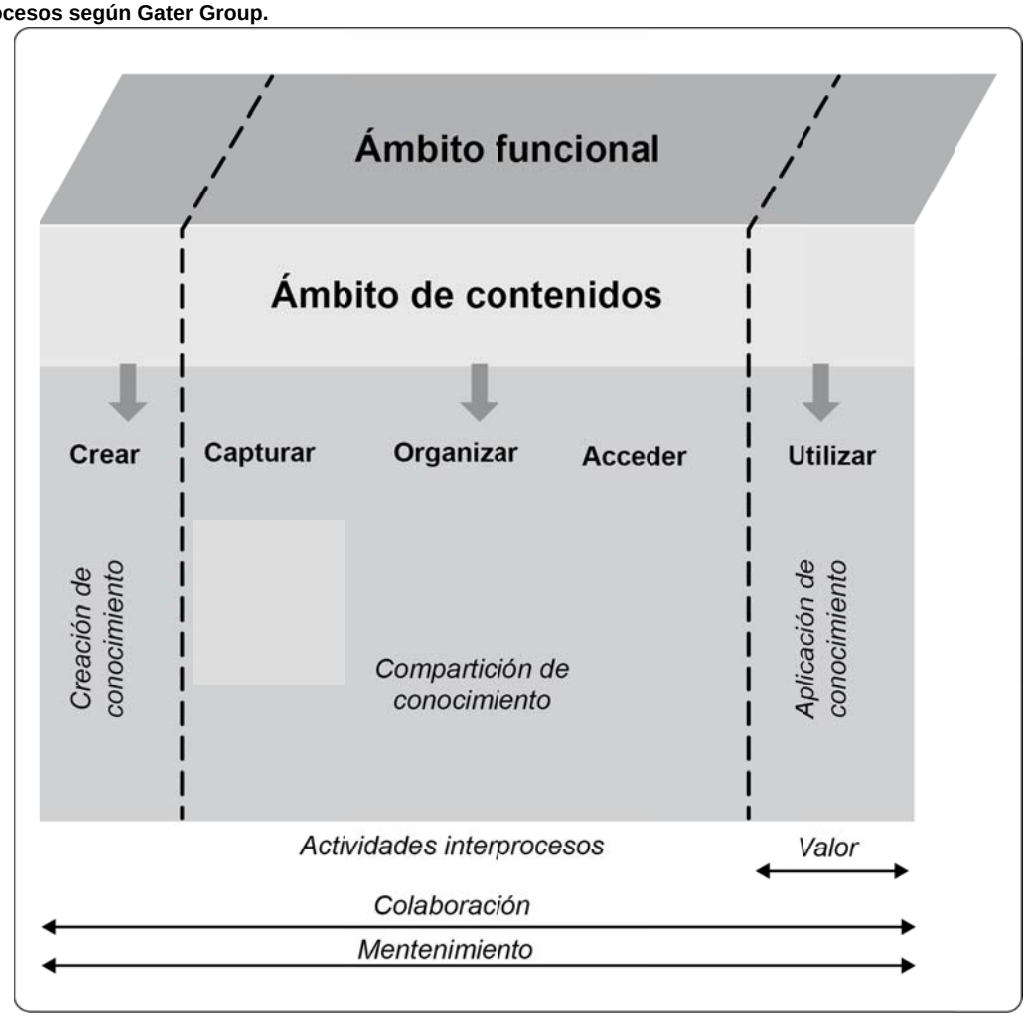
1.8.2. American Society for information

Por otro lado, Valhondo (2003) menciona que la *American Society for infomation*, en su reunión del año 2000 estructuró la innovación del conocimiento de la siguiente manera:

- *Descubrimiento, captura y creación del conocimiento*: Implica captura de cocimiento tácito: minería de datos, colaboración, directorios expertos, sistemas inteligentes que utilizan patrones, etc.
- *Clasificación y representación:* Diseño de interfaces, metadatos, visualización de información, taxonomías, clustering, indexación y vocabularios.
- *Recuperación de la información:* Motores de búsqueda, agentes inteligentes, browsing Vs búsqueda, navegación, arquitectura del conocimiento y de la información.
- *Diseminación de la información:* Comunicación, publicación (incluyendo: Internet, Intranet y Extranet).
- *Aspectos sociales, éticos, de comportamiento y legales*: Aceptación Vs rechazo de información, modificaciones del comportamiento, políticas y normas, evaluaciones de valores, búsqueda del conocimiento del comportamiento, formación para uso efectivo.

1.8.3. Garther Group

Figura 5. Procesos según Gater Group.



Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

Como se mencionó en apartado 1.6 *Garther Group*, considera los siguientes procesos:

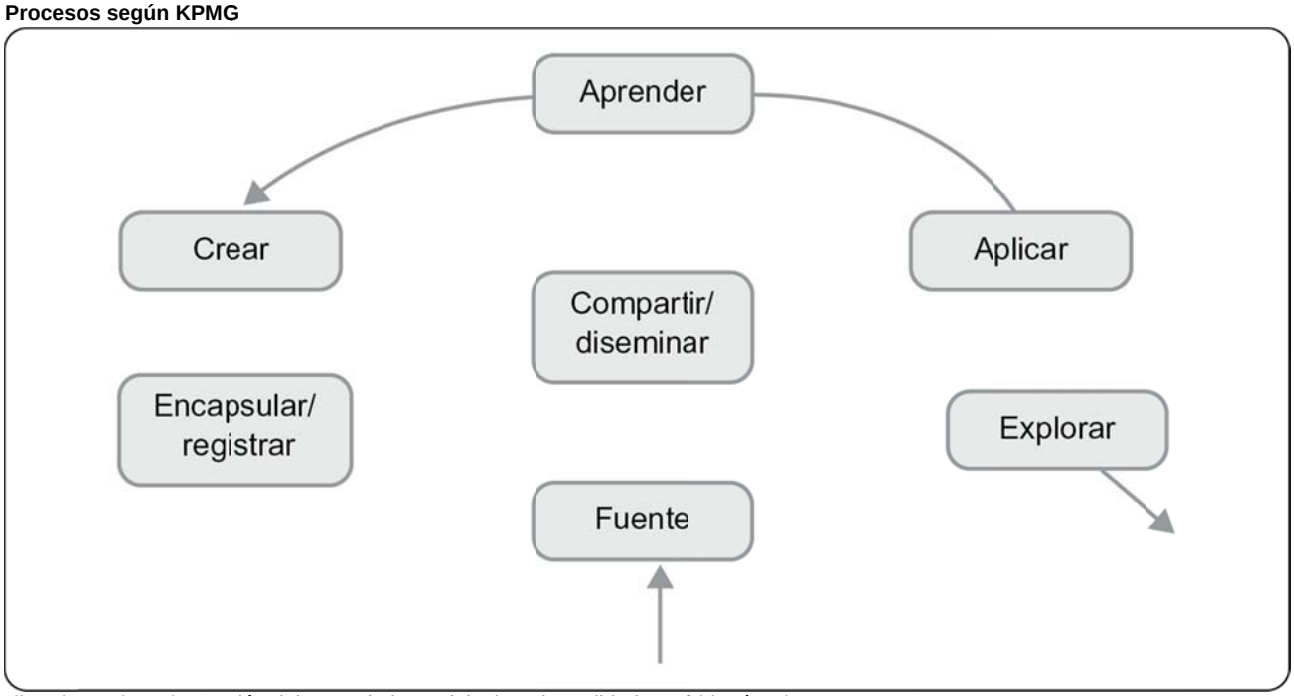
- *Crear.*
- *Capturar.*
- *Organizar.*
- *Acceder.*
- *Utilizar*

La representación gráfica de la interacción de estos procesos se muestra en la Figura 5, a partir de la cual es posible apreciar que los aspectos funcionales de contenidos de las empresas se sustentan en los procesos del cocimiento, los cuales requieren colaboración y mantenimiento de las personas para poder crear valor mediante la aplicación del conocimiento generado.

1.8.4. KPMG

KPMG, propone un modelo conformado por siete procesos básicos cuya representación se muestra en la Figura 6. La administración del conocimiento en este esquema se encarga de dar soporte y optimizar estos procesos en sus manifestaciones en el mundo real, destacando el papel relevante de compartición y diseminación, por lo cual ocupan el lugar central de la estrategia. Dicha estrategia considera explorar fuera de las organizaciones para complementar las fuentes internas de conocimiento y tenerlo disponible para a su aplicación que por medio del aprendizaje conducirá a la creación de nuevos conocimientos, y éstos a su vez se tendrán de registrar de alguna manera para llegar al proceso central de compartirlo.

Figura 6. Procesos según KPMG

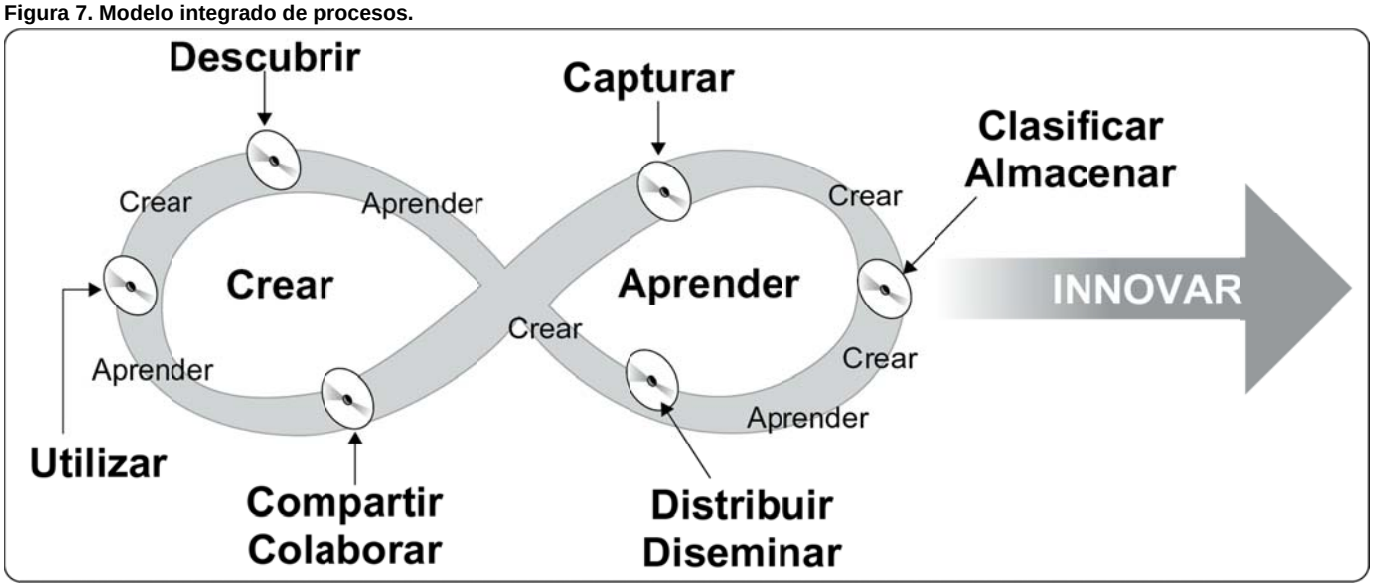


Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

1.8.5. Modelo integrado de procesos

El modelo de la Figura 7 es una síntesis de procesos propuesta por Valhondo (2003) que excluye la creación y el aprendizaje como procesos básicos ya que los considera como metaprocesos.

Figura 7. Modelo integrado de procesos.



Fuente: Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad*. Madrid: Díaz de Santos.

Según Valhondo (2003) la serie de procesos/metaprocesos de la Figura 7 tienen una interrelación espacial y temporal que no está dominada por alguno en particular, es decir, la interdependencia entre los procesos es múltiple y cruzada, las características generales de cada proceso son las siguientes:

- *Descubrir*: Este proceso implica identificar las fuentes de conocimiento para las organizaciones, tanto internas como externas así como establecer las herramientas necesarias para que sea extraído. Las fuentes van desde bases de datos, documentos y procesos organizacionales hasta las personas que forman la empresa por medio del conocimiento tácito.
- *Capturar:* Una vez localizado el conocimiento es preciso evaluar su utilidad y determinar de qué clase de conocimiento se trata ya que estos puntos determinarán la viabilidad y estrategia de captura para alimentar: bases de datos, directorios de expertos, repositorio de conocimientos, data warehouse o procesos.
- *Clasificar y almacenar:* La clasificación es un proceso interpretativo influido por los criterios de la persona que clasifica por lo cual es imperante que las organizaciones especifiquen las reglas bajo la cuales será clasificado el conocimiento antes de almacenarlo, y es precisamente esta clasificación que distingue la captura del almacenamiento ya que la captura solo implica codificar para que en el momento de que sea clasificado se ubique donde aporte valor para la organización.

- *Distribuir / Diseminar:* Hoy en día se requiere lidiar con la sobrecarga de información y si no es manejada de manera adecuada puede resultar un problema que cause los mismos efectos negativos que la carencia de la misma, por lo cual se debe establecer si el conocimiento será "arrojado" hacia las personas o simplemente se colocará en alguna herramienta tecnológica para que este disponible en el momento que se necesite.
- *Compartir / Colaborar:* Como se ha mencionado anteriormente el conocimiento se conforma de información estructurada en combinación con la experiencia de las personas y por lo tanto hay que fomentar que por iniciativa propia los empleados de una organización codifiquen esta experiencia para que este disponible para otros, además de colaborar con la socialización, externalización, internalización y combinación de conocimiento mediante la interacción adecuada con las demás personas.
- *Utilizar (innovación):* El objetivo de los procesos del conocimiento es que sean utilizados para conducir a innovaciones que permitan crecer a las organizaciones diferenciando sus productos y/o servicios por agregar valor para sus clientes.

El elemento adicional a la integración de los procesos que aporta Valhondo (2003) es el planteamiento de que la utilización del conocimiento no sólo redunda en la creación de más conocimiento y valor para las organizaciones sino que introduce a la innovación como producto de la administración del conocimiento, Senge (2006) menciona que se ha inventado una idea nueva cuando se demuestra que funciona en el laboratorio. La idea de transforma en innovación sólo cuando se puede reproducir sin contratiempo, en gran escala y a costos prácticos.

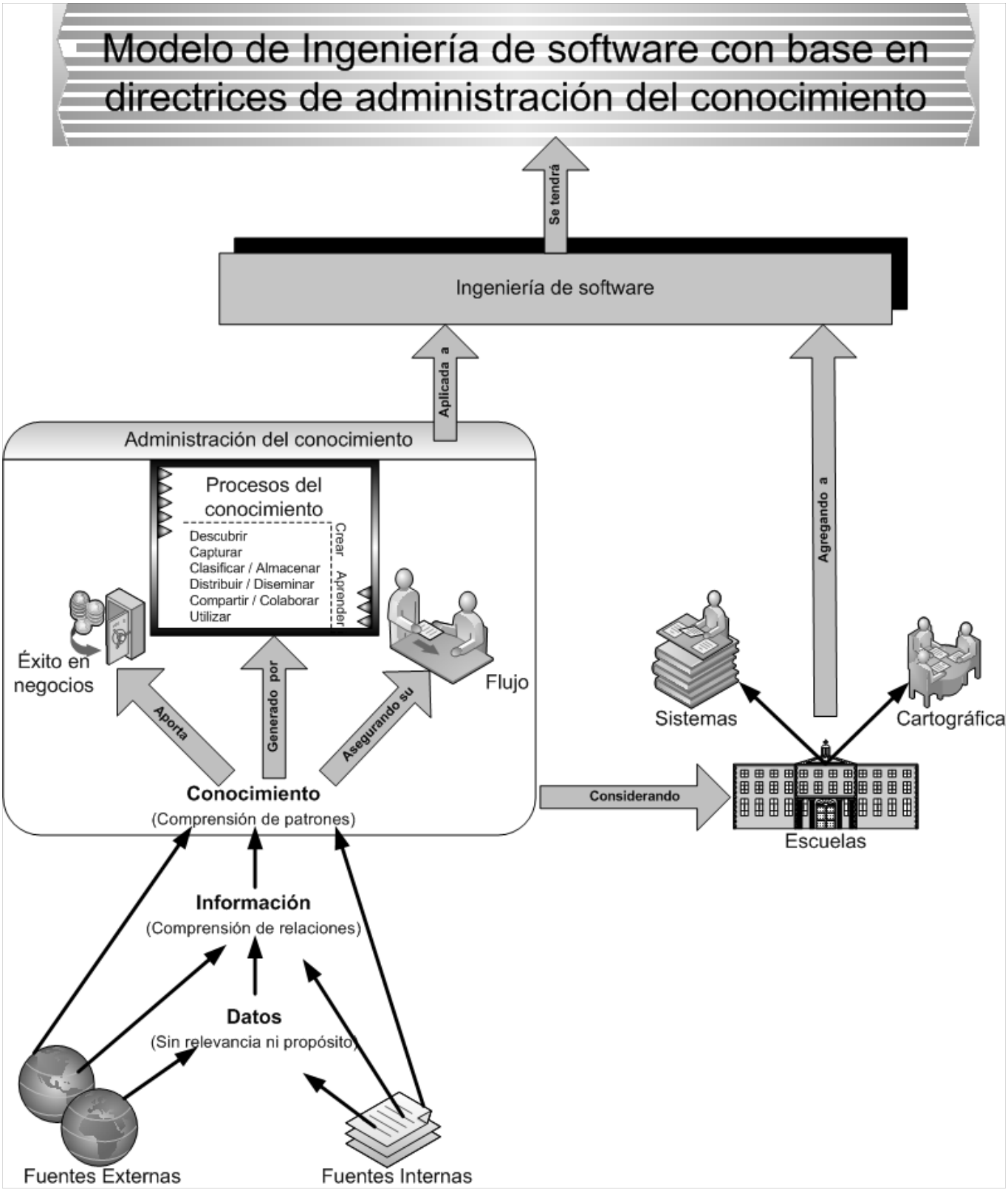
Por lo tanto, la innovación implica que un producto o servicio llegue al mercado y en consecuencia las empresas tengan beneficios económicos lo cual es uno de los elementos clave de la administración del conocimiento mencionados en las características clave de la sección 1.6.

1.8.6. Procesos seleccionados para el modelo

De los conjuntos de procesos listados de la sección 1.8.1 a la 1.8.5 fueron seleccionados, para el modelo propuesto, los considerados dentro del modelo integrado de procesos ya que, como su nombre lo indica, representa el compendio de los procesos empleados por otras organizaciones aunado a que se tienen implícitos la conceptualización del aprendizaje y la creación del conocimiento en los demás componentes, ahora bien cada uno de estos se empleará flexiblemente de forma que sean considerados únicamente los que realmente aporten una ventaja y valor.

1.9. Resumen de aportaciones al modelo

Figura 8. Directrices de administración del conocimiento para el modelo propuesto.



Fuente: Elaboración propia.

En este primer Capítulo se establecieron las bases de las directrices para la estructura de administración del conocimiento que será aplicada a la ingeniería de software, de forma que se pueda proponer un modelo que permita optimizarla. La Figura 8 indica gráficamente los elementos relevantes de este Capítulo, destacando la “evolución” de los datos a información y de ésta al conocimiento que aporta éxito en los negocios. Es por ello que surge la necesidad de asegurar su flujo y manejo adecuando por medio de la administración del conocimiento enfocada a las escuelas de sistemas y cartográfica para formar los cimientos que serán aplicados a la ingeniería de software.

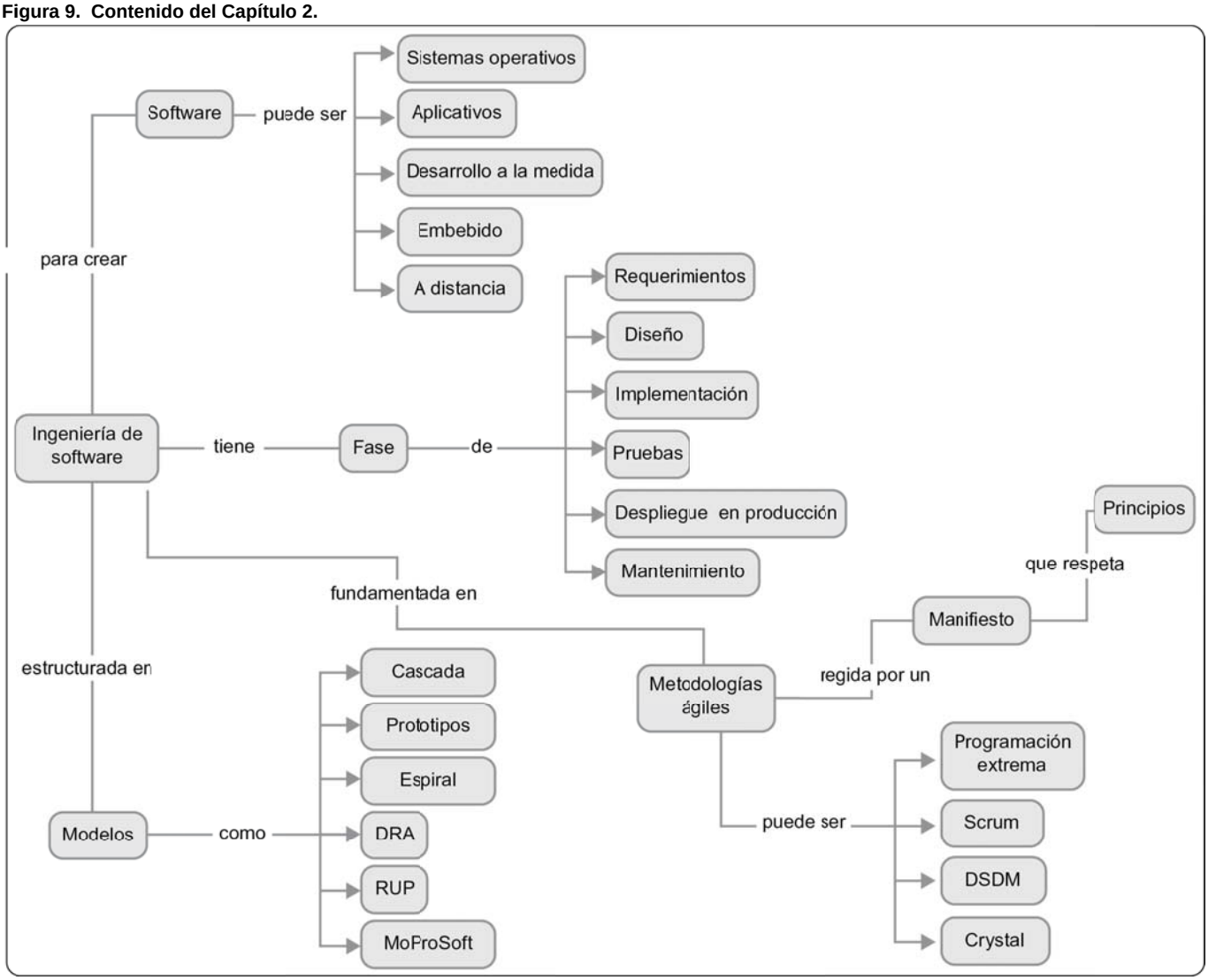
Enseguida, en el Capítulo 2 se presentarán los componentes que conforman el bloque de ingeniería se software mostrado en la Figura 8, que de momento se muestra sólo como una “caja negra” a la que se aplicarán las directrices de administración del conocimiento para obtener la propuesta del modelo

2. Ingeniería de software

En software, muy raramente partimos de requisitos con sentido. Incluso teniéndolos, la única medida del éxito que importa es si nuestra solución resuelve la cambiante idea que el cliente tiene de lo que es su problema.

Jeff Atwood

Figura 9. Contenido del Capítulo 2.



Fuente: Elaboración propia.

En el Capítulo 1 fueron presentadas las directrices de administración del conocimiento que serán aplicadas a la ingeniería de software, la cuál como lo indicaba la Figura 8, se mostró como una caja negra.

Ahora, en este segundo Capítulo se presenta el contenido de los componentes que están dentro de esa caja para ello presentaremos la definición del concepto de software para posteriormente presentar; una clasificación de tipo de software. Con estos elementos podremos entrar después de lleno a la ingeniería de software, por medio de su definición, sus procesos, y algunos de los modelos más recurridos para trabajarla. Entre éstos modelos daremos atención al modelo llamado *MoProsoft* el cual define a la norma mexicana para la industria de desarrollo y mantenimiento de software.

Otro ámbito de atención en el Capítulo será el de un conjunto de metodologías para desarrollo de software denominadas por Beck y otros (2001) como “ágiles” a las cuales en los

últimos años se les ha dirigido la atención debido a que proveen mayor flexibilidad para los procesos de ingeniería de software.

Al finalizar este Capítulo se tendrán perfectamente identificados los procesos de ingeniería se software y se contará con referencias de modelos y metodologías para construir el modelo de ingeniería de software con base en directrices de administración del conocimiento presentado en la sección 3.8.3, temas que representamos gráficamente en la Figura 9.

2.1. Definición de software.

En la actualidad el software está presente de una u otra manera en las actividades que realizamos diariamente, es empleado para computadoras personales o laptops, está embebido en una gran gama de dispositivos que van desde reproductores mp3, hasta automóviles y grandes maquinarias. La mayoría de las personas que interactúan con las tecnologías de la información y la comunicación tienen alguna noción aproximada al concepto de lo que es el software, algunas sólo por interacción con el mismo y otras con bases teóricas, si bien, las definiciones formales pueden tener alguna variación entre sí, la mayoría se fundamenta bajo ciertas premisas principales. La definición establecida por el *Institute of Electrical and Electronics Engineers (IEEE)* donde IEEE (1990) indica que el software consiste en Programas de computadora, procedimientos y la posible documentación asociada a los mismos así como los datos que pertenecen a la operación de un sistema informático.

Aunado a la definición anterior se tienen 2 definiciones más, estructuradas por IEEE (1990), las cuales se consideran de relevancia para los temas tratados más adelante:

1. “Aplicación de software: Software diseñado para cumplir con necesidades específicas de un usuario; por ejemplo, software para navegación, manejo de nómina o control de procesos”

2. “Sistema de software: Software diseñado para facilitar la operación y mantenimiento de un sistema de computadoras y sus programas asociados; por ejemplo, sistemas operativos, ensambladores y utilerías”

Para el modelo planteado en la sección 3.8.3 se considera no solamente la conceptualización técnica de software relacionada con las definiciones anteriores, sino también, como lo mencionan Salem Ben y Mouna Ben (2009) la consideración de que los artefactos de software representan la acumulación del conocimiento organizacional, es decir, el know-how, lo cual resalta la necesidad de manejar los procesos de desarrollo de software con una perspectiva de administración del conocimiento, considerando que alrededor del software hay dos clases de conocimiento: el conocimiento del negocio o know-how y el conocimiento relacionado con el software por sí mismo como es la documentación.

2.2. Tipo de software.

Mochi Alemán (2006) presenta la siguiente lista del tipo de software:

- *Sistemas operativos:* No se trata sólo de un programa, sino de un conjunto integrado de programas con diversos componentes. Metafóricamente es como la estructura física de una casa. Entre los sistemas operativos más conocidos podemos citar a CP/M, el sistema DOS, UNIX, Windows, MAC OS, OS/2, y Linux.

- *Software aplicativos o productos empaquetados de mercado masivo:* Son los programas que corresponden a necesidades e intereses particulares. Se conocen como productos empaquetados y se dirigen al mercado masivo, ya que son vendidos en paquetes confeccionados, de manera estándar. Generalmente su uso es más fácil y viene acompañado de manuales que explican todas sus funciones. Siguiendo con el ejemplo metafórico de la casa, el software aplicativo provee los componentes que hacen la casa habitable, respondiendo a las necesidades de cada usuario.

- *Soluciones empresariales o software desarrollado a medida, servicios informáticos:* Este tipo de programas exigen, de acuerdo con su complejidad, algún grado de personalización o adaptación a los requerimientos específicos de la organización en la cual van a ser implementados.

- *Software embarcado o embebido:* Viene incorporado en distintos tipos de maquinaria, equipos y dispositivos de consumo, es habitual que sea desarrollado *in house* por los propios productores de los bienes en los cuales se incorpora.

- *Software de servicios:* Incluye desarrollo de software a distancia, administración remota de aplicaciones, desarrollo y mantenimiento de aplicaciones y, en fases más avanzadas, administración integral de todo el departamento de sistemas de una empresa.

Acorde a la perspectiva hacia la que se orienta cada tipo de software, el modelo propuesto en el presente trabajo, se enfocará a las soluciones empresariales o software desarrollando a la medida, ya que a diferencia de las otras clasificaciones, para esta variante se involucra trabajo intensivo de conocimiento de diversos sectores debido a que requiere capturar el *know-how* de negocio de las empresas, surgiendo la necesidad de establecer una relación entre el software y de conocimiento contenido en el mismo.

2.3. Definición de ingeniería de software.

En relación a este tema IEEE (1990) presenta estas 2 definiciones:

1. “Ingeniería de software: Enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software.”

2. “Ambiente de ingeniería de software: Consiste en el hardware, software y firmware empleado para la ejecución de la ingeniería se software. Los componentes recurrentes son:

equipos de cómputo, compiladores, ensambladores, sistemas operativos, emuladores, herramientas para pruebas y documentación, así como manejadores de bases datos. ”

Varios autores, entre ellos Rusu, Russell, Robinson y Rusu (2010), complementan la definición de ingeniera de software desglosándola en las siguientes fases:

- *Fase de requerimientos:* Corresponde a las actividades que permiten especificar de la forma más detallada posible los requerimientos de los usuarios, de manera que las solicitudes no sean incompletas, ambiguas o contradictorias. Una vez que se cuentan con los elementos suficientes para realizar un análisis de los requerimientos se evalúan los mismos y se definen los que pueden ser alcanzables acorde a los recursos tanto técnicos como humanos disponibles.

- *Fase de diseño:* En esta etapa se diseña la solución que cubra el alcance de los requerimientos definido en la fase anterior, la cual incluye estructurar una arquitectura para los sistemas de software o bien modificaciones a una existente, prototipos de las interfaces gráficas con las que interactuará el usuario, mapeo de procesos, flujo de datos, diagramas e inclusive selección del lenguaje de programación.

- *Fase de implementación:* Consiste en *traducir* el diseño especificado anteriormente a un lenguaje de programación, de forma que se obtenga el producto de software que cumpla con la definición del alcance.

- *Fase de pruebas:* Una vez codificado el software se debe validar su adecuada funcionalidad, las pruebas se pueden dividir en las siguientes:

- Pruebas unitarias: La complejidad de los sistemas de software actuales implica que sean implementados por varias personas, por lo cual cada una de ellas debe asegurarse que la *porción* de construcción de código que le fue asignado trabaje correctamente.

- Pruebas integrales: El conjunto de programadores que participaron en la implementación del software deben asegurarse que el código construido sea compatible y cumpla adecuadamente su función dentro de las interacciones requeridas en el sistema de software.

- Pruebas en ambiente de *quality assurance*: Los dos tipos de pruebas mencionadas anteriormente son ejecutadas en un ambiente creado para desarrollar software, donde los desarrolladores suelen tener el control del mismo por lo cual pueden presentarse situaciones en las que el software no sea validado en algún escenario al que tenga que responder en condiciones de uso real. Por este motivo es requerido contar con un ambiente creado para ejecutar pruebas de software y es necesario migrar los componentes construidos por los programadores a este ambiente, lo que representa un *ensayo* del procedimiento a seguir cuando se traslade a un ambiente operativo. En este ambiente se emula un comportamiento operativo del nuevo software para garantizar que el sistema se comporte adecuadamente.

- *Fase de liberación o despliegue:* En caso de que sea detectada alguna falla en la fase de pruebas se debe volver a la fase de implementación para corregirla y llevar a

cabo una vez más las validaciones necesarias, hasta que sea posible certificar que el nuevo software cumple con el alcance definido y entonces se procederá a migrarlo al ambiente productivo para que los usuarios tengan acceso al mismo.

- *Fase de mantenimiento:* Una vez que el software es colocado en ambiente operativo puede darse el caso de que se presenten errores no detectados en la fase de pruebas o bien que se deban realizar modificaciones para atender nuevos requerimientos en cuyo caso se repetirían las fases tantas veces como sea solicitado por los usuarios.

La definición de ingeniería de software manejada por IEEE (1990) deja fuera a las personas y no toma en cuenta la administración del conocimiento que evidentemente esta está presente en este proceso, ya que como se puede apreciar en las características de las fases, implícitamente hay componentes de los procesos del conocimiento y en cada una de ellas se obtienen documentos que resultan de transformar conocimiento tácito en explicito, pero no se presta suficiente atención a manejarlos, y con ello se deja de lado que en un momento dado podrían resultar vitales para asegurar la calidad y correcta operatividad de los sistemas se software a largo plazo.

Los elementos empleados para el modelo de la definición de IEEE (1990), junto con lo mencionado por Rusu, Russell, Robinson y Rusu (2010), conformarán la perspectiva de que la ingeniería de software es un enfoque sistemático y disciplinado para la construcción de soluciones de software fundamentada en las fases de requerimientos, diseño, implementación, pruebas, despliegue en producción y mantenimiento.

2.4. Modelos de ingeniería de software.

Pressman (2002) menciona que para resolver los problemas reales de una industria, un ingeniero de software o un equipo de ingenieros deben incorporar: una estrategia de desarrollo que acompañe al proceso, método, capas de herramientas y a las fases de ingeniería de software. Esta estrategia recibe la denominación de modelo de proceso, paradigma de ingeniería del software o simplemente modelo de desarrollo de software. Se debe elegir un modelo acorde a la naturaleza del proyecto, tipo de aplicación, métodos, herramientas a utilizarse, así como a los controles y entregas que se requieren.

A continuación se presenta un breve resumen de algunos de los modelos de desarrollo de software que se emplean con mayor recurrencia.

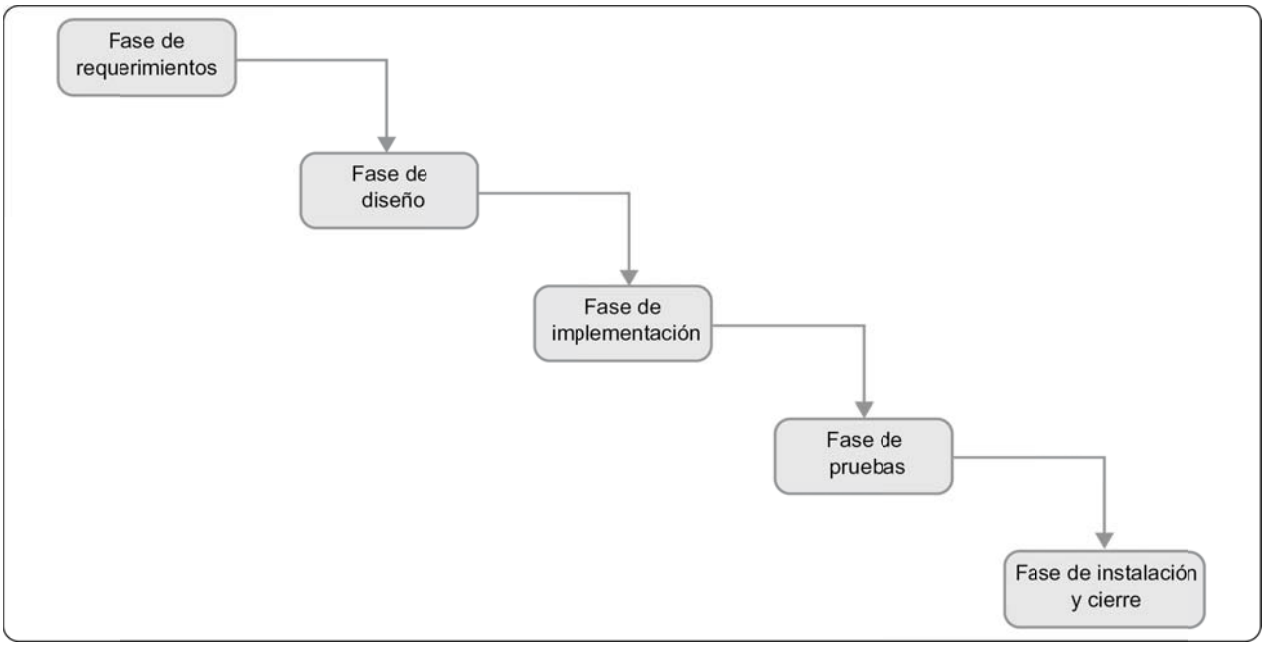
2.4.1. Modelo en cascada

El modelo más empleado en la industria del desarrollo de software es el modelo en cascada y por ello se hace referencia al mismo cómo el “modelo clásico”, el cual es definido por IEEE (1990) de la siguiente manera:

Modelo del proceso de desarrollo de software en el cual las actividades que lo componen, típicamente conceptualizadas como fase, fase requerimientos, fase de diseño, fase de implementación, fase de pruebas, fase de instalación y cierre, son ejecutadas en este orden, con posibilidad de que se superpongan pero con poca o nula repetición.

La representación gráfica de este modelo se representa en la Figura 10.

Figura 10. Modelo de cascada.



Fuente: Elaboración propia.

Los motivos por el que este modelo falla según Pressman (2002) son los siguientes:

- Los proyectos reales raras veces siguen el modelo secuencial que propone el modelo. Aunque puede acoplar interacción, lo hace indirectamente. Como resultado, los cambios pueden causar confusión cuando el equipo del proyecto comienza.
- A menudo es difícil que el usuario exponga explícitamente todos los requisitos.
- El usuario debe tener paciencia. Una versión de trabajo del(los) programa(s) no estará disponible hasta que el proyecto esté muy avanzado. Un grave error puede ser desastroso si no se detecta hasta que se revisa el programa.

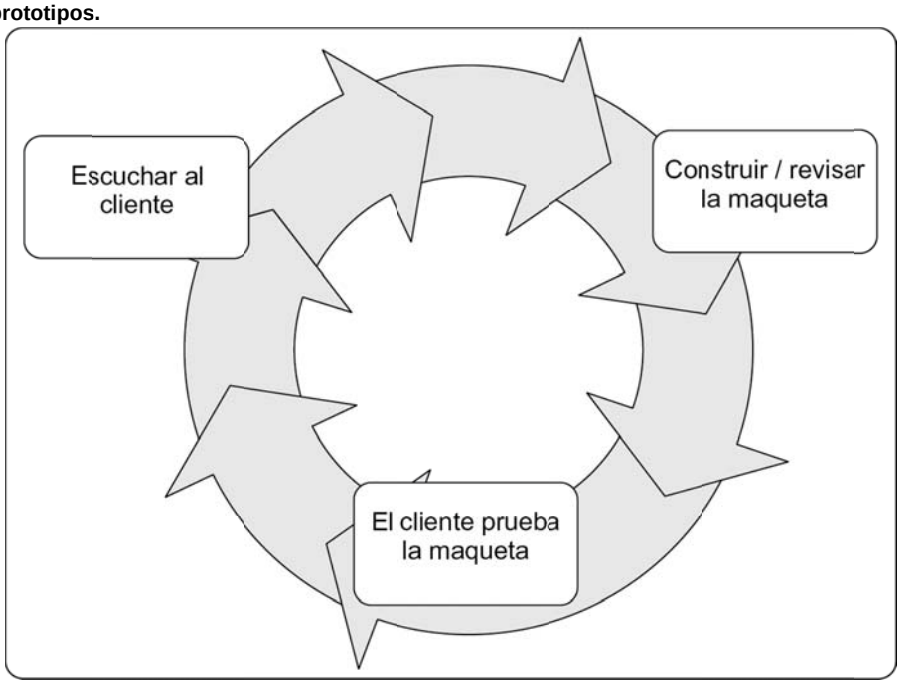
El modelo mencionado es el más usado porque su aplicación no representa gran complejidad tanto para los desarrolladores de software como para las demás personas involucradas, ya que al manejar fases de forma secuencial se tiene perfectamente definido en que punto termina cada una. Este modelo puede resultar útil para desarrollos pequeños o mantenimientos que impliquen pocos cambios, pero si se trata de nuevos proyectos o ajustes

significativos puede llegar a representar una gran lentitud para responder a nuevos requerimientos y por ende a las necesidades relacionadas con el negocio de la empresa.

2.4.2. Modelo de prototipos

La aplicación de este paradigma guía al modelo de prototipos, como lo indica Pressman (2002), comienza con la recolección de requisitos. El desarrollador y el usuario encuentran y definen los objetivos globales para el software, identifican los requerimientos conocidos y las áreas del esquema en donde es obligatoria más definición. Entonces aparece un *diseño rápido*. El diseño rápido se centra en una representación de esos aspectos del software que serán visibles para el usuario (por ejemplo: datos de entrada y formatos de salida). El diseño rápido lleva a la construcción de un prototipo. El prototipo lo evalúa el usuario y se utiliza para refinar los requisitos del software a desarrollar. La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer. Expresado de manera gráfica este modelo se visualiza en la Figura 11.

Figura 11. Modelo de prototipos.



Fuente: Pressman, R. S. (2002). *Ingeniería del software un enfoque práctico*. Madrid, España: McGraw-Hill.

El emplear este modelo permite poner las manos a la obra inmediatamente pero representa los siguientes problemas que menciona Pressman (2002):

- El usuario ve lo que parece ser una versión final del software, sin saber que con la prisa de hacer que funcione no se ha tenido en cuenta la calidad del software global o la facilidad de mantenimiento a largo plazo. Cuando se informa de que el producto se debe construir otra vez para que se puedan mantener los niveles altos de calidad, el usuario no suele entenderlo y pide

que se apliquen unos pequeños ajustes para que se pueda hacer del prototipo un producto final.

- El desarrollador, a menudo, hace compromisos de implementación para hacer que el prototipo funcione rápidamente. Se puede utilizar un sistema operativo o lenguaje de programación inadecuado simplemente porque está disponible y porque es conocido; un algoritmo deficiente se puede implementar simplemente para demostrar la capacidad. Después de algún tiempo, el desarrollador debe familiarizarse con estas selecciones, y olvidarse de las razones por las que son inadecuadas.

La mayor aportación de este modelo es que permite disminuir la ambigüedad de las solicitudes de los usuarios al promover que trabajen junto con los desarrolladores para realizar la definición de los requerimientos y al poder evaluar un prototipo es posible percatarse de mejoras o elementos importantes que se tengan que considerar para el producto final, pero para llevar a cabo lo anterior se requiere gran cantidad de tiempo, ya que para poner a punto una versión final del software regularmente serán requeridas varias revisiones aunado al hecho de que como en otras industrias a todo prototipo, a pesar de que sea funcional, se le tiene que depurar de forma que pueda ser entregado al usuario final.

2.4.3. Modelo en espiral

La definición de IEEE (1990) respecto a este modelo es la siguiente:

> Modelo del proceso de desarrollo de software en el cual las actividades que lo componen, típicamente análisis de requerimientos, diseño preliminar y detallado, codificación, integración y pruebas, son realizadas de manera repetida hasta que el software esta completo.

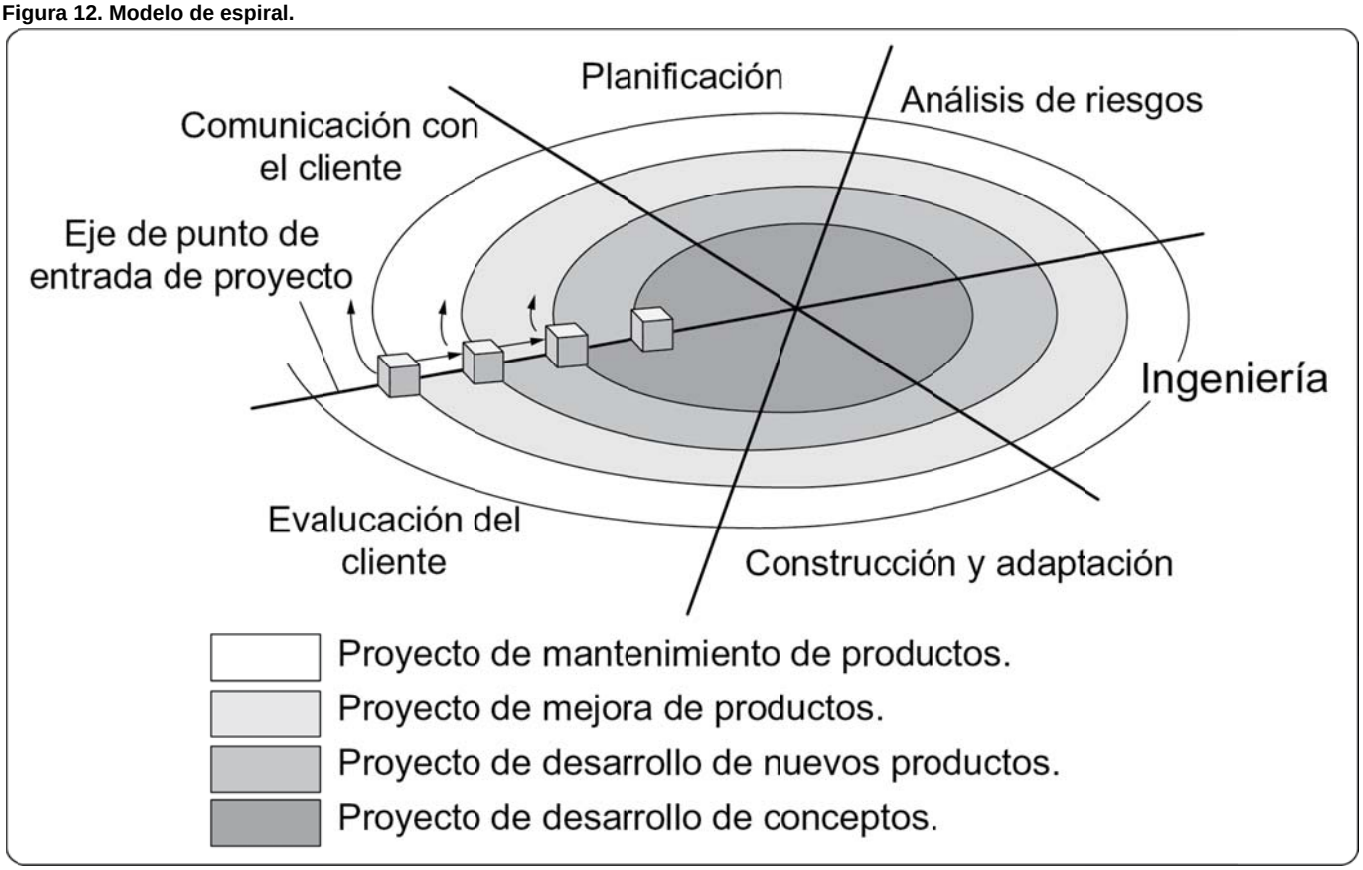
Pressman (2002) menciona que este modelo es de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal en cascada. En el modelo de espiral, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones la versión incremental podría ser un modelo en papel o un prototipo. En las últimas iteraciones se producen versiones cada vez más completas del sistema diseñado. Gráficamente el modelo se plantea en la Figura 12.

A diferencia del modelo en cascada que termina cuando se entrega el software, este modelo puede adaptarse y aplicarse a lo largo de la vida del software. Es un enfoque realista del desarrollo de software a gran escala. El software evoluciona, a medida que progresa el proceso, el desarrollador así como el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene

el enfoque sistemático de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo (Pressman, 2002).

Los problemas con este enfoque son que puede resultar difícil convencer a los usuarios de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la evaluación del riesgo. Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas (Pressman, 2002).

Figura 12. Modelo de espiral.



Fuente: Pressman, R. S. (2002). *Ingeniería del software un enfoque práctico*. Madrid, España: McGraw-Hill.

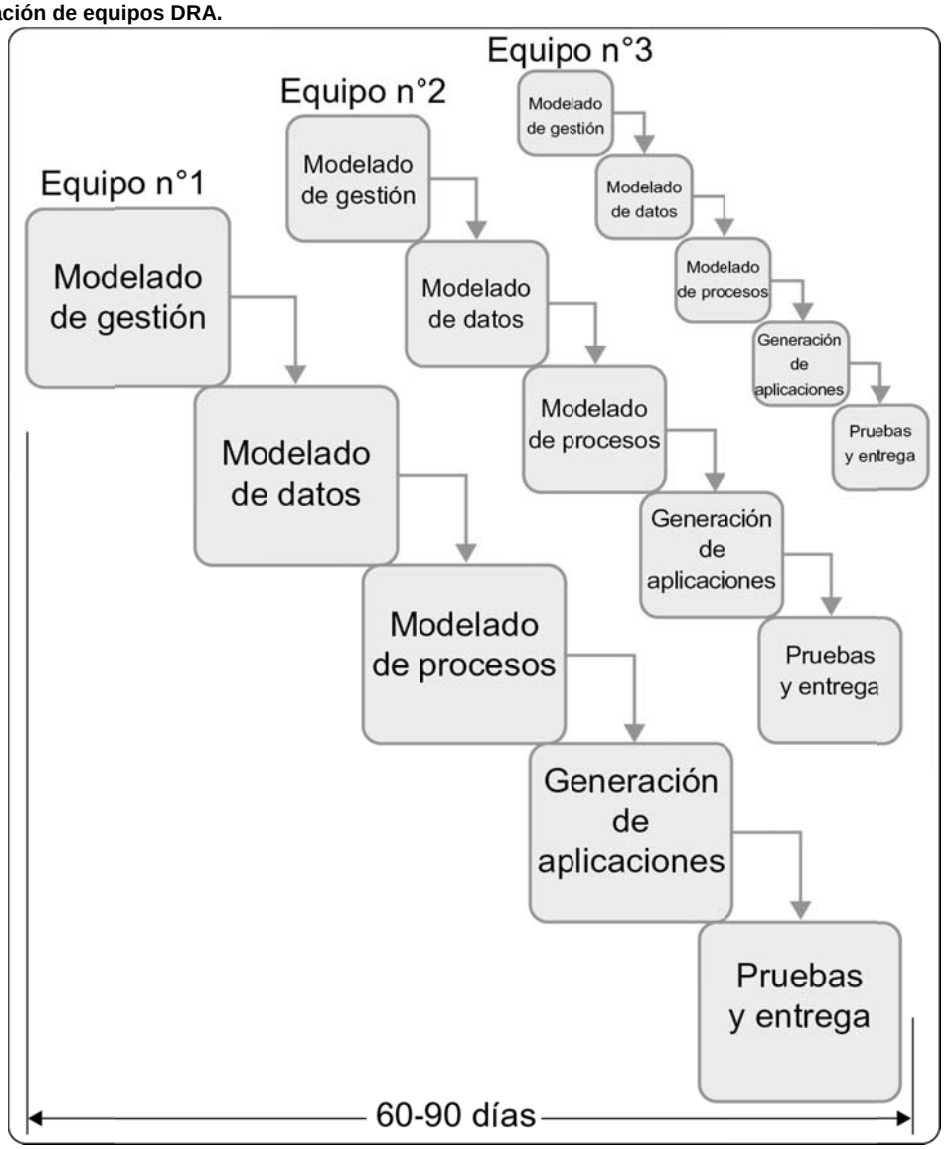
La ventaja principal en este modelo es el hecho de que se manejan entregas incrementales de funcionalidad, lo cual proporciona la flexibilidad para trabajar tanto con proyectos y mantenimientos pequeños como con los que implicarían funcionalidades grandes segmentándolas desde un principio en iteraciones cuyos componentes permitan un manejo controlado y alcanzable de cada entrega. Por otro lado, representa cierta complejidad aplicarlo debido a su estructura por lo cual las personas involucradas tanto en las áreas de negocio como de sistemas deben contar con gran experiencia para segmentar la funcionalidad general e identificar los riesgos en cada iteración.

2.4.4. Desarrollo Rápido de Aplicaciones (DRA)

Pressman (2002) lo clasifica como un modelo de proceso del desarrollo del software lineal y secuencial que enfatiza un ciclo de desarrollo extremadamente corto. Es una adaptación a alta velocidad del modelo en cascada, en el que se logra el desarrollo

rápido utilizando una construcción basada en componentes. Si se comprenden bien los requisitos y se limita el ámbito del proyecto, el proceso DRA permite al equipo de desarrollo crear un sistema completamente funcional dentro de períodos cortos de tiempo (por ejemplo: de 60 a 90 días). Cuando se utiliza principalmente para aplicaciones de sistemas de información, el enfoque DRA comprende las fases de: modelado de gestión, modelado de datos, modelado de procesos, generación de aplicaciones así como pruebas y entrega, tal como se puede apreciar en la Figura 13.

Figura 13. Segmentación de equipos DRA.



Fuente: Pressman, R. S. (2002). *Ingeniería del software un enfoque práctico*. Madrid, España: McGraw-Hill.

Los inconvenientes de este modelo listados por Pressman (2002) son los siguientes:

- Para proyectos grandes aunque por escalas, requiere recursos humanos suficientes para crear el número correcto de equipos.
- Requiere usuarios y desarrolladores comprometidos en las rápidas actividades necesarias para completar un sistema en un marco de tiempo abreviado. Si no hay compromiso por ninguna de las partes constituyentes, los proyectos fracasarán.

- No todos los tipos de aplicaciones son apropiados para DRA. Si un sistema no se puede modularizar adecuadamente, la construcción de los componentes necesarios será problemático.
- No es adecuado cuando los riesgos técnicos son altos. Esto ocurre cuando una aplicación hace uso de tecnologías nuevas, o cuando el software nuevo requiere un alto grado de interoperabilidad con programas de computadora ya existentes.

Este paradigma puede resultar atractivo para las empresas que o cuentan con gran cantidad de recursos, o bien, tienen organizados sus grupos de trabajo segmentados de manera que cada uno atiende un sistema distinto, porque en caso de que trabajen por áreas funcionales de negocio resultará complicado aplicarlo, debido a que en la actualidad es muy raro que solamente un sistema trabaje sin interacción con otros.

2.4.5. Rational Unified Process (RUP)

Es mencionado por Rational (1998) que RUP es un proceso de ingeniería de software que provee un enfoque disciplinado para la asignación de tareas y responsabilidades relacionadas con la implementación de software. Su meta es asegurar una alta calidad del software que satisfaga las necesidades de los usuarios finales, bajo un plan de trabajo y presupuesto que pueda ser predecible.

En la especificación de este modelo se maneja el concepto de bases de conocimiento con las cuales se pretende asegurar que todos los integrantes de un equipo de trabajo manejen el mismo lenguaje común, procesos y visión de como desarrollar el software.

Las actividades del RUP están orientadas a crear y mantener modelos que resultan ser representaciones semánticas de los sistemas se software, en lugar de enfocarse en producir grandes cantidades de documentos en papel, Rational (1998). Para lo anterior se proporciona una guía de cómo usar de manera efectiva el lenguaje unificado de modelado (UML por sus siglas en inglés).

(Rational, 1998) desglosa este modelo en las siguientes 6 mejores prácticas que según menciona, fueron las más efectivas identificadas en la industria de desarrollo de software:

1. *Desarrollo iterativo de software:* Dada la complejidad de los sistemas de software que se tienen hoy en día, resulta muy tardado seguir el modelo clásico por lo cual se requiere un enfoque iterativo que permita desarrollar una comprensión incremental del problema depurando de manera sucesiva y logrando una solución efectiva después de múltiples iteraciones.
2. *Administración de requerimientos:* Describe como obtener, organizar y documentar la funcionalidad solicitada así como sus restricciones; rastrear y documentar decisiones; capturar y comunicar requerimientos de negocio.
3. *Emplear arquitecturas basadas en componentes:* Plantea la manera de diseñar una arquitectura resistente que a la vez sea flexible, que permita clasificar los cambios, entendible de manera intuitiva y promueva la reutilización de

software. Lo anterior auxiliándose del desarrollo de software por componentes, en donde por componente se entiende un módulo de software no trivial, es decir subsistemas que cumplen perfectamente con una función específica.

4. *Modelado visual del software:* Este proceso muestra como estructurar de manera visual la arquitectura y componentes del software así como su comportamiento y que por medio de las abstracciones visuales sea más sencillo comunicar los diversos aspectos del software, indicando la forma en que los elementos del sistema encajan entre sí, ayudando a verificar que la implementación del código se ajusta al diseño.
5. *Verificación de la calidad de software:* Se menciona la criticidad que se debe considerar para validar que el software creado cumpla con los requerimientos solicitados bajo correctos parámetros de confiabilidad, funcionalidad, rendimiento tanto de la aplicación como del sistema. RUP define una estructura para establecer la planeación, diseño, implementación, ejecución y evaluación de las pruebas.
6. *Controlar lo cambios del software:* Hace referencia a la manera de controlar, rastrear y monitorear los cambios para que sea posible realizar exitosamente el desarrollo interactivo.

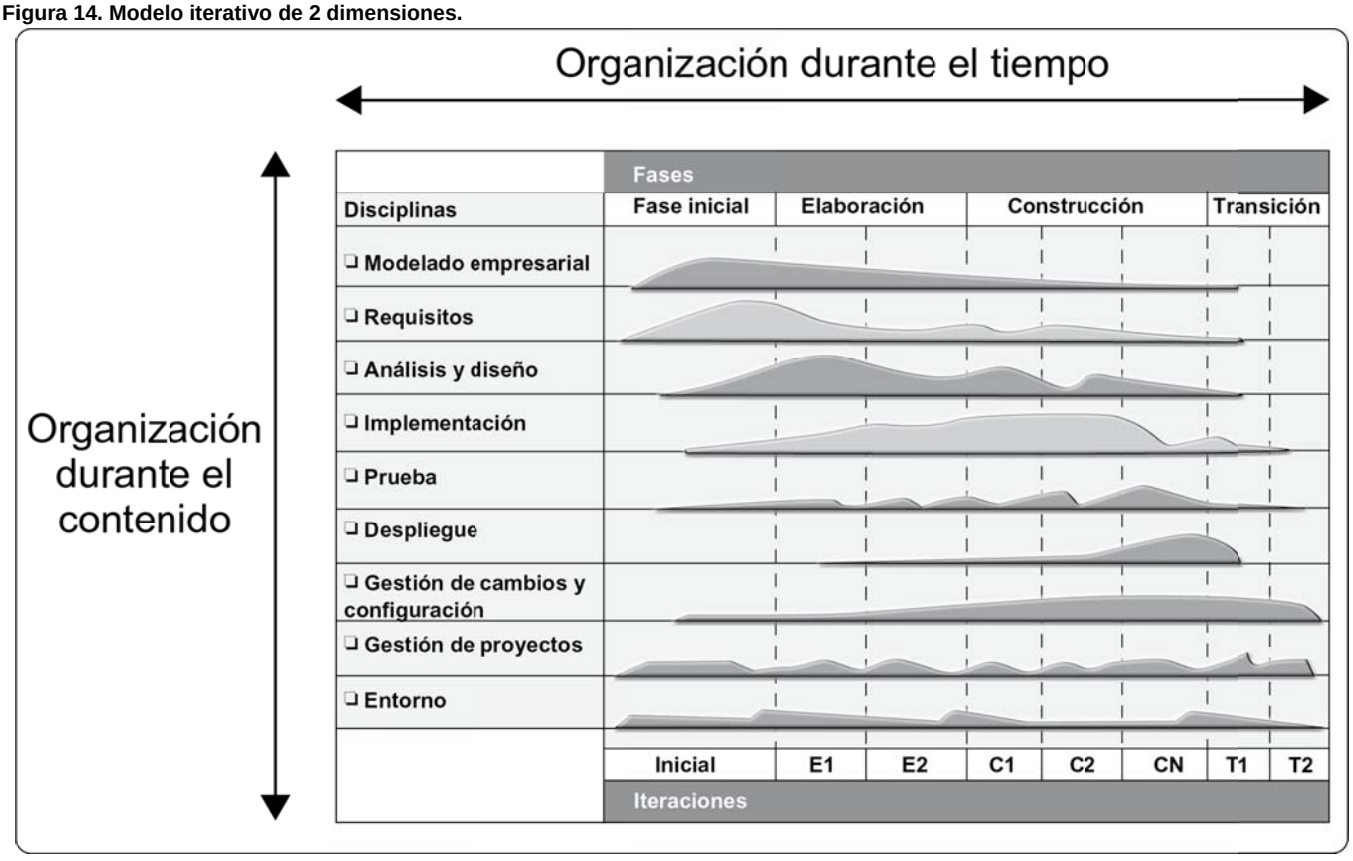
El listado anterior se encuentra embebido en el proceso iterativo de RUP, el cual acorde a Rational (1998) es descrito en dos dimensiones, una en cada eje:

- El eje horizontal representa el tiempo y muestra el aspecto dinámico del proceso y es expresado en términos de ciclos, fases, iteraciones e hitos, a su vez esta estructurado por:
  - *Fase inicial*: Identificación de los casos de negocio para el sistema y se delimita el alcance del proyecto.
  - *Elaboración*: Análisis del dominio del problema para establecer los fundamentos de la arquitectura, desarrollar el plan del proyecto y eliminar los elementos de mayor riesgo.
  - *Construcción*: Se crean los componentes de la aplicación y se integran al producto final, para que lleven a cabo las pruebas respectivas.
  - *Transición*: Se entrega el software al usuario, y recurrentemente surgen solicitudes de más requerimientos asociados al mismo o bien solución de errores.
- Por su cuenta el eje vertical representa el aspecto estático del proceso, en el que se describen las actividades, artefactos, trabajadores y flujos de trabajo. Y esta compuesto por:
  - *Modelo empresarial*: Documentación de los procesos de negocio por medio de casos de uso.
  - *Requisitos*: Se define lo qué el software debe de hacer.
  - *Análisis y diseño*: Se establece la forma de cómo el sistema cumplirá con los requerimientos.

- *Implementación:* Se construye el código del software.
- *Prueba*: Se realizan las pruebas pertinentes para asegurar la calidad del software.
- *Despliegue:* Se realiza la liberación del software en ambientes productivos para que los usuarios finales tengan acceso al mismo.
- *Configuración y gestión de cambios:* Indica la forma de controlar los artefactos de software creados por todos los desarrolladores, para evitar confusión y asegurar que el producto final no tenga conflictos de versiones.
- *Gestión de proyectos*: Consiste en balancear los objetivos, administrar los riesgos y cumplir con los propósitos del proyecto.
- *Entorno:* Se refiere a que se cuente con procesos y herramientas adecuadas para que el equipo de desarrollares puedan realizar sus funciones.

La representación gráfica de dimensiones del RUP se presenta en la Figura 14.

Figura 14. Modelo iterativo de 2 dimensiones.



Fuente: Rational. (1998). *Rational Unified Process Best Practices for Software Development Teams*. Cupertino, California, USA.

Este modelo ya plantea ciertas consideraciones para manejar el conocimiento relacionado con el software, pero aún no establece algún esquema para ligarlo con el conocimiento del negocio.

2.4.6. Modelo de Procesos para la Industria de Software (MoProSoft)

Los modelos mencionados anteriormente son propuestas estandarizadas a nivel internacional; en México la Secretaría de Economía solicitó a la Facultad de Ciencias de la Universidad Nacional Autónoma de México un modelo que sirviera como base para establecer la norma mexicana para la industria de desarrollo y mantenimiento de software a partir de lo cual surgió MoProSotf.

Por su parte Oktaba y otros (2005) mencionan que el propósito de MoProSoft es fomentar la estandarización de las operaciones de la industria de software en México a través de la incorporación de las mejores prácticas en la gestión de ingeniería de software, pretendiendo que la adopción del modelo permita elevar la capacidad de las organizaciones para ofrecer servicios con calidad y alcanzar niveles internacionales de competitividad.

El modelo combina principalmente elementos de ISO 9000:2000 y del *Capability Maturity Model Integration* (CMMI), no esta enfocado completamente a los procesos de ingeniería de software sino que según lo que es indicado por NYCE (2011) comprende los siguientes niveles:

- Alta Dirección:
  - *Gestión de Negocio*: Establecer la razón de ser de la organización, sus objetivos y las condiciones para lograrlos, para lo cual es necesario considerar las necesidades de los clientes, así como evaluar los resultados para poder proponer cambios que permitan la mejora continua.
- Gerencia:
  - *Gestión de Procesos*: Estructurar los procesos de la organización, en función de los identificados en el plan estratégico. Así como definir, planificar, e implantar las actividades de mejora en los mismos.
  - *Gestión de Proyecto*: Asegurar que los proyectos contribuyan al cumplimiento de los objetivos y estrategias de la organización.
  - *Gestión de Recursos*: Conseguir y dotar a la organización de los recursos humanos, infraestructura, ambiente de trabajo y proveedores, así como crear y mantener la base de conocimiento de la organización. Su finalidad es apoyar el cumplimiento de los objetivos del plan estratégico de la organización.
- Operación:
  - *Administración de Proyectos Específicos*: Definir y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados.
  - *Desarrollo y Mantenimiento de Software*: Realización sistemática de las actividades de obtención de requisitos, análisis, diseño, construcción, integración y pruebas de productos de software nuevos o modificados cumpliendo con los requisitos solicitados.

Como se puede apreciar este modelo pretende establecer un marco de referencia para la operación completa de las organizaciones y en lo que se refiere al proceso de ingeniería de software, sólo representa una pequeña parte del mismo. Oktaba y otros (2005) plantean su manejo acorde al enfoque clásico del modelo en cascada. Por otra parte, si bien se menciona la importancia del conocimiento en relación con el software, se maneja de manera aislada el conocimiento referente al negocio del conocimiento embebido con el software.

2.5. Metodologías ágiles

Este tipo de metodologías surgieron a mediados de los años 90's como reacción a la excesiva documentación y rigidez establecida por los enfoques de ingeniería de software estructurados hasta ese entonces, dando como resultado que se creara un movimiento cuyos fundadores se auto denominaron "La alianza ágil" quienes plasmaron su filosofía en el *Manifiesto para el Desarrollo Ágil de Software* publicado por Beck y otros (2001) el cual enuncia lo siguiente:

> Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar: Individuos e interacciones sobre procesos y herramientas Software funcionando sobre documentación extensiva Colaboración con el cliente sobre negociación contractual Respuesta ante el cambio sobre seguir un plan esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

Beck y otros (2001) también publicaron los 12 principios en los que fundamentaron el manifiesto:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el ambiente y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para ajustar y perfeccionar su comportamiento.

Uno de los creadores del manifiesto, Highsmith (2001), menciona que las metodologías ágiles no sólo dicen que las personas son el activo más importante en el proceso de ingeniería de software, sino que realmente actúan posicionando como referencia central a las personas para entregar un buen producto. Así mismo plantea que *movimiento ágil* no está en contra de la documentación o la estructuración de una metodología formal, pero si pretende contrarrestar la tendencia a la excesiva burocratización en la que ha sido atrapada la ingeniería de software. Por los motivos anteriores el modelo propuesto en la sección 3.8.3 estará construido bajo la filosofía del manifiesto ágil y la guía de los 12 principios indicados por Beck y otros (2001), ya que resultan una alternativa efectiva para resolver los problemas relacionados con la ingeniería de software.

A continuación se presentan algunas generalidades de metodologías ágiles más representativas.

2.5.1. Programación Extrema (Extreme Programming)

El autor de esta metodología es Don Wells, en el sitio oficial de esta metodología Wells (2009) menciona que el primer proyecto bajo este enfoque comenzó el 6 de marzo de 1996, puntualizando que su éxito radica en priorizar la satisfacción de los usuarios proporciona las herramientas necesarias a los desarrolladores para responder a las solicitudes de cambio inclusive a pesar de que sean realizadas en etapas finales del ciclo de desarrollo de software.

La programación extrema (XP por sus siglas en inglés), acorde a lo mencionado por Wells (2009) enfatiza que el trabajo en equipo, los líderes de proyecto, usuarios y desarrolladores se manejan como iguales dentro de equipos autodirigidos, mejorando los proyectos de software con base en cinco valores que moldean una forma de trabajar en armonía:

- *Simplicidad:* No se realizará más de lo que es necesario y solicitado, lo anterior maximizará la inversión. Se darán pequeños pasos para lograr la

funcionalidad completa y mitigar las fallas, creando algo de lo que se pueda estar orgulloso y sea posible mantener a largo plazo bajo costos razonables.

- *Comunicación:* Todos son parte del equipo, y la comunicación se realiza cara a cara. El trabajo se realiza en conjunto desde la definición de los requerimientos hasta la codificación del software. Se creará la mejor solución al problema que sea definida por el equipo.
- *Retroalimentación:* Cada iteración se realiza con un compromiso serio mediante la entrega de software funcional, del cual se realizan demostraciones previas para realizar los cambios que se consideren pertinentes. Los procesos se adaptan al proyecto y no en el sentido inverso.
- *Respeto:* Cada individuo da y siente el respecto que merece como miembro del equipo. Todos aportan valor al proyecto inclusive si sólo es con su entusiasmo. Los programadores respetan la experiencia de los usuarios y viceversa. El líder de proyecto respeta el derecho de los demás de aceptar la responsabilidad del trabajo y tener autoridad sobre el mismo.
- *Coraje:* Se dirá la verdad acerca del progreso del trabajo y los estimados de tiempo. No se manejarán excusas para justificar las fallas. No se tendrá miedo a algo porque nadie trabaja solo. Se tendrá adaptabilidad a los cambios cada que se presenten.

Los valores anteriores son plasmados en que los programadores mantienen una comunicación constante con los usuarios, así como otros programadores del equipo, realizan un diseño simple y limpio, obtienen retroalimentación realizando pruebas del software desde el primer día, entregan el sistema a los usuarios tan pronto como sea posible implementando los cambios que sean solicitados y en que la XP tiene la visión de que cualquier éxito depende del respeto al trabajo que cada miembro del equipo realiza, dando como resultado que los programadores puedan enfrentar con coraje los cambios tanto a requerimientos como a tecnología (Wells, 2009) .

Las reglas definidas por Wells (2009) para XP son:

- Planeación:
  - Escribir las experiencias de los usuarios.
  - Crear el plan de trabajo a partir del plan de liberación.
  - Realizar entregas pequeñas pero frecuentes.
  - Dividir el proyecto en iteraciones.
  - Iniciar cada iteración con la planeación de la misma.
- Dirección:
  - Proporcionar al equipo un espacio abierto de trabajo.
  - Establecer un ritmo sostenido.
  - Iniciar cada día con una junta de estatus.
  - Medir la velocidad del proyecto.

- Mover a la gente.
- Reparar XP cuando se rompe.

➢ Diseño:

- Simplicidad.
- Elegir una metáfora para el sistema.
- Utilizar tarjetas de clase, responsabilidad y colaboración para sesiones de diseño.
- Crear soluciones sencillas para disminuir los riegos.
- Ninguna funcionalidad es agregada demasiado tarde.
- Restructurar siempre que sea posible.

➢ Codificación:

- El usuario debe estar siempre disponible.
- El código debe apegarse a los estándares establecidos.
- Realizar pruebas unitarias.
- Generar el código por programación en parejas.
- Sólo una pareja integra el código.
- Respaldar el código fuente frecuentemente en un repositorio.
- Definir una computadora para integración.
- Realizar el trabajo en equipo.

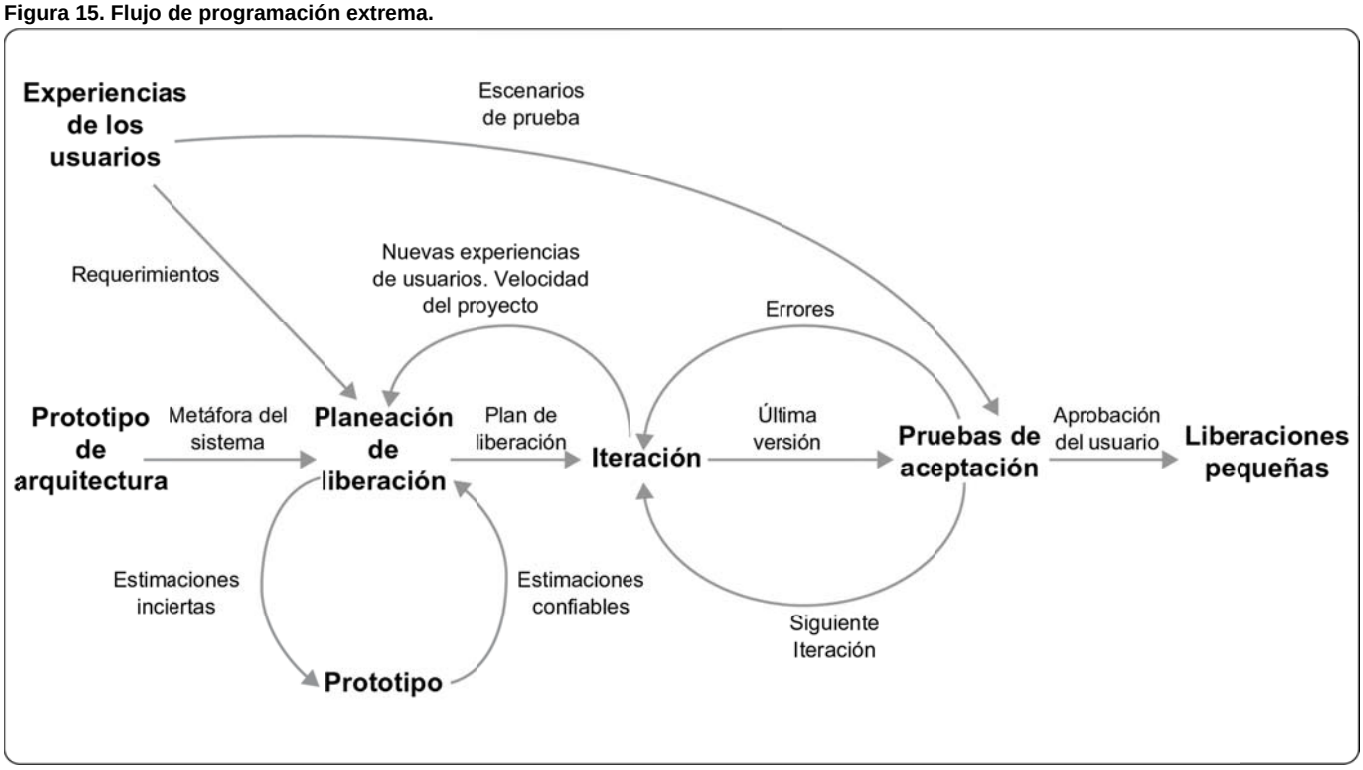
➢ Pruebas:

- Realizar pruebas unitarias a todo el código.
- Las pruebas unitarias debe ser exitosas antes de liberar.
- Documentar los errores encontrados.
- Realizar pruebas de aceptación en base a las experiencias del usuario y publicar los resultados

Las reglas anteriores se encuentran plasmadas en el diagrama de flujo de XP presentado en la Figura 15.

La programación extrema es una metodología muy flexible, requiere que las personas involucradas cuenten con gran disciplina para llevarla a cabo. Es digno de resaltar la comunicación directa y constante planteada como el manejo de entregas incrementales. Tal vez su único punto débil es la propuesta de programación por parejas, es decir que un par de programadores estén sentados frente a una misma computadora para codificar el software, debido a que los recursos con los que cuentan las empresas suelen ser limitados y no se cuenta con los empleados necesarios para que sea posible realizar esta actividad.

Figura 15. Flujo de programación extrema.



Fuente: Traducción de Wells, D. (28 de Septiembre de 2009*). Extreme Programming*. Recuperado el 08 de Marzo de 2012, de extremeprogramming.org: http://www.extremeprogramming.org/.

2.5.2. Scrum

Ken Schwaber y Jeff Sutherland (2011) crearon esta metodología, respecto a la cual mencionan que se fundamenta, en la teoría de control empírico de procesos, lo cual implica que el conocimiento proviene de la experiencia y toma de decisiones con base en lo que es conocido, además maneja un enfoque iterativo e incremental para optimizar la predictibilidad y controlar riesgos.

Según Ken Schwaber y Jeff Sutherland (2011) la teoría de control empírico de procesos está constituida por tres pilares:

- *Transparencia:* Los aspectos significativos de los procesos deben estar visibles para los responsables de los resultados.
- *Inspección:* Se deben realizar inspecciones frecuentes de los artefactos y progreso de las metas para detectar cualquier posible desviación.
- *Adaptación:* Si se detecta algún aspecto del proceso que se encuentra fuera de los límites aceptables provocando afectación al producto final, se deben realizar los ajustes necesarios para corregir la desviación.

Scrum se rige por un conjunto reglas que establecen las relaciones entre los siguientes elementos que mencionan (Schwaber & Sutherland, 2011):

- *El equipo Scrum:* Consiste en equipos de trabajo autodirigidos y multifuncionales para optimizar la flexibilidad, creatividad y productividad. Los equipos están constituidos por:
    - Dueño del producto: Es la persona encargada de maximizar el valor del producto creado por el equipo de desarrollo, incluyendo la definición del mismo.
    - Equipo de desarrollo: Consiste en un equipo de profesionales que realizan el trabajo necesario para entregar un incremento de funcionalidad al producto al final de cada *sprint*.
    - Scrum master: Se responsabiliza de asegurar que la metodología Scrum sea entendida y ejecutada, vigilando que todos los miembros del equipo se apeguen a la teoría, prácticas y reglas.
- *Eventos Scrum:* Se manejan eventos predefinidos para crear regularidad y minimizar la necesidad de reuniones que no sean definidas por Scrum. Cada evento tiene una duración máxima de forma que se asegura que se invierte el tiempo apropiado para planear. Los tipos de eventos que se manejan son los siguientes:
    - El Sprint: consiste en un periodo de tiempo de un mes o menos durante el cual es desarrollada funcionalidad especifica del producto sin que se agreguen cambios que afecten la meta inicial y los integrantes del equipo se mantienen constantes. Cada que se termina un Sprint se inicia inmediatamente otro.
    - Cancelación del Sprint: Este evento se presenta si el dueño producto considera pertinente cancelar un Sprint antes de que se termine el periodo establecido para ser completado.
    - Junta de planeación del Spring: El trabajo que tiene que ser completado durante el Spring es planeado durante este evento por todos los integrantes del equipo de trabajo.
    - Scrum diario: Consiste en una reunión de 15 minutos donde los integrantes del equipo de desarrollo sincronizan sus actividades y establecen el plan para las siguientes 24 horas.
    - Revisión del Sprint: Al final de cada Sprint se realiza una revisión para inspeccionar el incremento de funcionalidad del producto y definir si se requiere algún ajuste al mismo.
    - Retrospectiva del Sprint: Es una oportunidad para el equipo del Scrum de realizar un autoanálisis y crear un plan para mejorar en el siguiente Sprint.
- *Artefactos Scrum:* Son representaciones del trabajo o valor en diversas formas que resultan útiles para agregar transparencia así como oportunidades para la inspección y adaptación, que redundan en información clave para que los equipos Scrum completen exitosamente la entrega de incremento de funcionalidad. Estos artefactos son los siguientes:

- Cartera de producto: Consiste en una lista de todo lo que se requiere para tener completado el producto, convirtiéndose en la única fuente de requerimientos para cualquier cambio que se desee hacer al producto.
- Cartera de Sprint: Es un conjunto de carteras de producto, seleccionadas para un Sprint junto con el plan para la entrega de la funcionalidad incremental hasta obtener la meta final.
- Incremento: Es el conjunto todos los elementos de las carteras de producto completados tanto durante el Sprint actual como los anteriores.

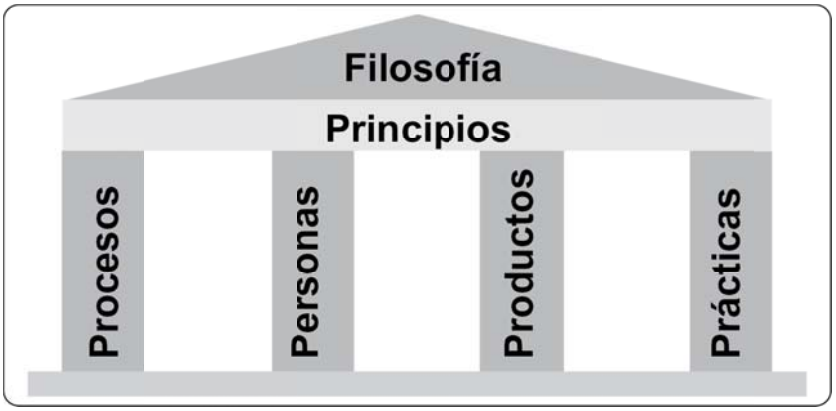
Esta metodología es planteada para ser empleada no sólo para la ingeniería de software sino para cualquier tipo de proyecto, por lo cual deja de lado algunas ventajas importantes presentadas por la Programación Extrema de la sección 2.5.1, y, si bien trabaja el concepto de entregas incrementales, se puede percibir cierta rigidez y burocracia en su estructura.

2.5.3. Método de Desarrollo de Sistemas Dinámicos (Dynamic Systems Development Method)

Esta metodología, al igual que Scrum del apartado 2.5.2, esta planteada no solamente para desarrollo de software, sino que se enfoca a cualquier tipo de proyecto, DSDM Consortium (2008) menciona que se puede combinar con otras metodologías de administración de proyectos, inclusive con otros esquemas ágiles, esta perspectiva toma como elementos fijos el tiempo, los costos y la calidad a partir de los cuales se define el alcance de la funcionalidad que será entregada. Y se fundamenta en esta filosofía: "Todo proyecto debe estar alineado a objetivos estratégicos claros y enfocándose en entregas prontas que beneficien de manera real al negocio."

De forma gráfica DSDM Consortium (2008) presenta que la estructura de la metodología queda plasmada en la Figura 16.

Figura 16. Estructura DSDM.



Fuente: Traducción de DSDM Consortium. (2008). *DSDM Atern Teh Handbook*. Chestfiels: DSDM Consortium.

DSDM Consortium (2008) menciona que los principios sobre los que reside la filosofía anteriormente enunciada son los siguientes:

- *Enfocarse en las necesidades de negocio:* Cada decisión que sea tomada durante el proyecto debe ser ejecutada con base en los objetivos del mismo, los cuales deben estar estructurados acorde a lo que se necesita para el negocio.
- *Entregar a tiempo:* Esta cuestión es primordial en cualquier tipo de proyecto, dado que la mayor parte de las veces los proyectos están relacionados con una nueva oportunidad de mercado o pueden implicar regulaciones legales.
- *Colaborar:* Trabajando en equipo se tiene un mayor entendimiento del problema y más rapidez para resolverlo, lo cual implica que el equipo tenga un mejor rendimiento que la suma de esfuerzos individuales.
- *Nunca comprometer calidad:* El nivel de calidad debe ser establecido desde el inicio del proyecto y todo el equipo del proyecto debe abocarse en no alterar ese nivel, inclusive aunque la calidad sea aumentada no se debe permitir la desviación.
- *Realizar entregas incrementales:* De esta manera es posible que los usuarios tengan un producto funcional en menor tiempo, lo cual permite que proporcionen retroalimentación del mismo y se realicen las modificaciones pertinentes.
- *Desarrollar iterativamente:* Dentro del entorno tan cambiante como en el que estamos inmersos actualmente, se requiere un esquema que facilite responder a estos cambios para entregar el producto correcto con las iteraciones que sean necesarias.
- *Comunicación continúa y clara:* Recurrentemente la causa principal de que los proyectos fracasen es que tienen una pobre comunicación. Por lo cual se debe contar con esquemas que aumenten la efectividad de la comunicación tanto entre equipos como entre individuos.
- *Demostrar control:* Es necesario tener proactividad para monitorear y controlar los proyectos, de forma que se alcancen los objetivos trazados.

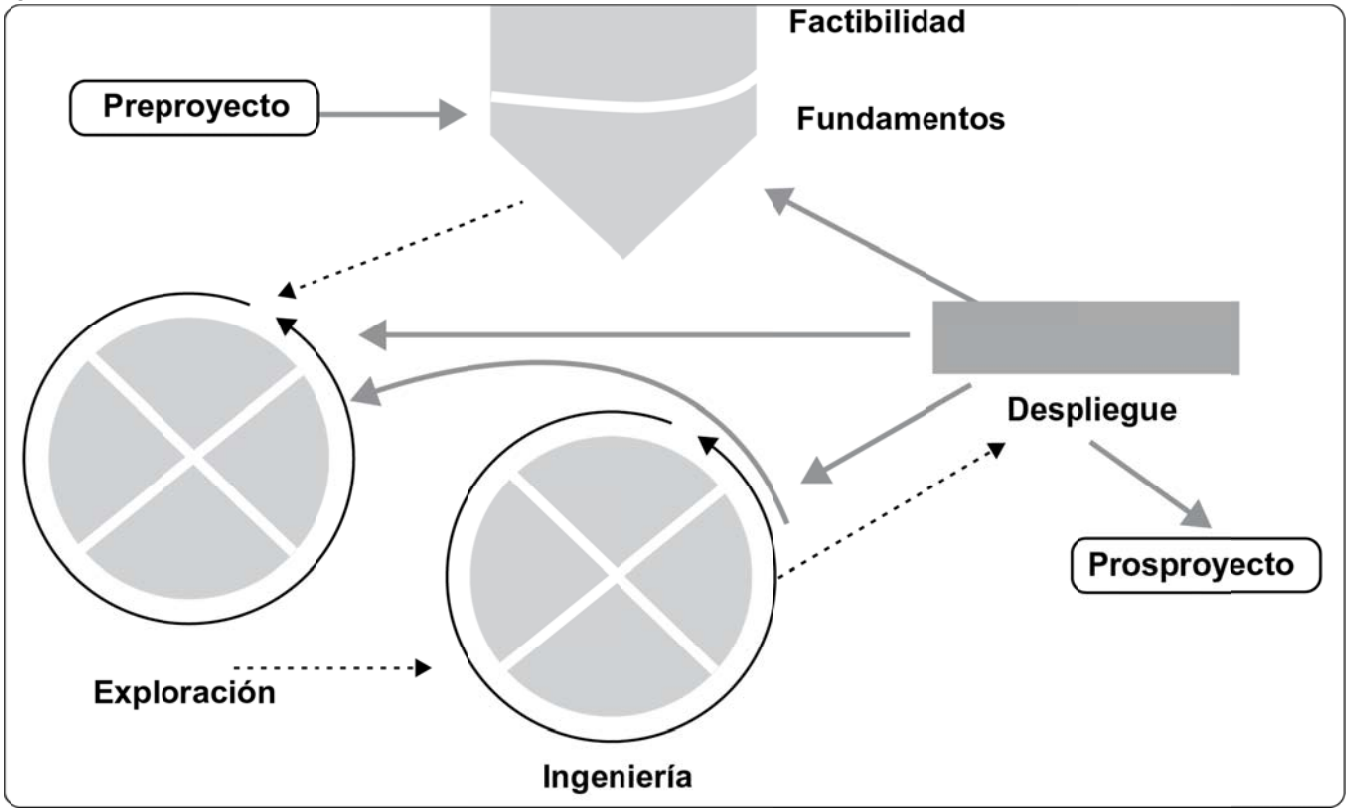
El ciclo de vida para esta metodología especificado por DSDM Consortium (2008), comprende las siguientes etapas:

- *Preproyecto:* Formalización de la propuesta del proyecto y su clasificación dentro del contexto correcto dentro de la organización.
- *Factibilidad*: Evaluación del proyecto para establecer que es viable, tanto desde el punto de vista de negocio como técnico, mediante una investigación de alto nivel de las posibles soluciones, costos y estimados de tiempo.
- *Fundamentos*: Definición de las directrices del proyecto en lo que se refiere al negocio, la solución y la administración del mismo, de forma que se estructure correctamente un enfoque robusto pero flexible.
- *Exploración*: Investigación iterativa e incremental de los requerimientos detallados de negocio para definir una solución viable.

- *Ingeniería*: Desarrollo también iterativo e incremental de la solución definida en la etapa de exploración para lograr el cumplimiento completo de los requerimientos.
- *Despliegue*: Poner la solución desarrollada al alcance de los actores interesados para que sea empleada.
- *Posproyecto*: Evaluación del valor real aportado por el proyecto al negocio, se debe realizar tan pronto como se tengan parámetros para ser evaluado.

Gráficamente la interacción de las etapas se muestra en la Figura 17.

Figura 17. Ciclo de vida DSDM.



Fuente: Traducción de DSDM Consortium. (2008). *DSDM Atern Teh Handbook*. Chestfiels: DSDM Consortium.

Esta metodología resalta un punto importante que debe cumplir cualquier proyecto en las organizaciones y es que debe estar completamente enfocado a las necesidades del negocio ya que en caso contrario la ejecución del mismo se podría cuestionar ampliamente, por otra parte al parecer deja de lado el principio número 2, del manifiesto ágil enlistado en la sección 2.5, el cual menciona que son aceptables los cambios de requerimientos para proporcionar una ventaja competitiva al cliente, pero en los principios de esta metodología mencionados por DSDM Consortium (2008), se menciona que no se debe permitir un cambio en la calidad inicial definida para el producto sin importar si el cambio es para aumentar la calidad.

2.5.4. Metodologías Cristal (Crystal)

Cristal (o *Crystal* por su nombre en inglés) es el nombre para una familia de metodologías las cuales, como menciona Alistair Cockburn (2001), al igual que los cristales geológicos cada una tiene un diferente color y dureza acorde con la criticidad y tamaño de los proyectos, siendo la filosofía Cristal la siguiente: "El desarrollo de software es visualizado como un juego cooperativo de invención y comunicación, con la meta principal de entregar software útil y funcional, y la meta secundaria de prepararse para el siguiente juego"

Dos consecuencias de esta filosofía son 1) que proyectos diferentes tienen que ser ejecutados de maneras distintas, y 2) la cantidad de modelado así como comunicación que las personadas necesitan tener será la que sea necesaria para que se muevan juntos a través del juego. Lo valores intrínsecos de las metodologías Cristal son:

- Atención a las personas y la comunicación entre ellas.
- Alto grado de tolerancia.

Los principios mencionados por Alistair Cockburn (2001) de Cristal son:

- El equipo puede reducir trabajo intermedio produciendo código funcional de manera frecuente, así como empleando planes de comunicación más eficientes.
- Debido a que cada proyecto es diferente en cuanto al tiempo y la forma, las convenciones que el equipo de trabajo adopte tienen que ser moldeadas acorde al contexto.
- Los cuellos de botella en el sistema determinan el trabajo concurrente y la necesidad de información.

En cuanto a las reglas de Cristal Alistair Cockburn (2001) lista las dos siguientes:

- El proyecto debe emplear desarrollo incremental, con entregas cada 4 meses o menos (de preferencia entre uno y tres meses).
- El equipo debe mantener talleres de análisis antes y después de entregar un incremento (con la recomendación de que también se lleve a cabo a la mitad del proceso).

Y, finalmente, las técnicas que presenta Alistair Cockburn (2001) para cristal son:

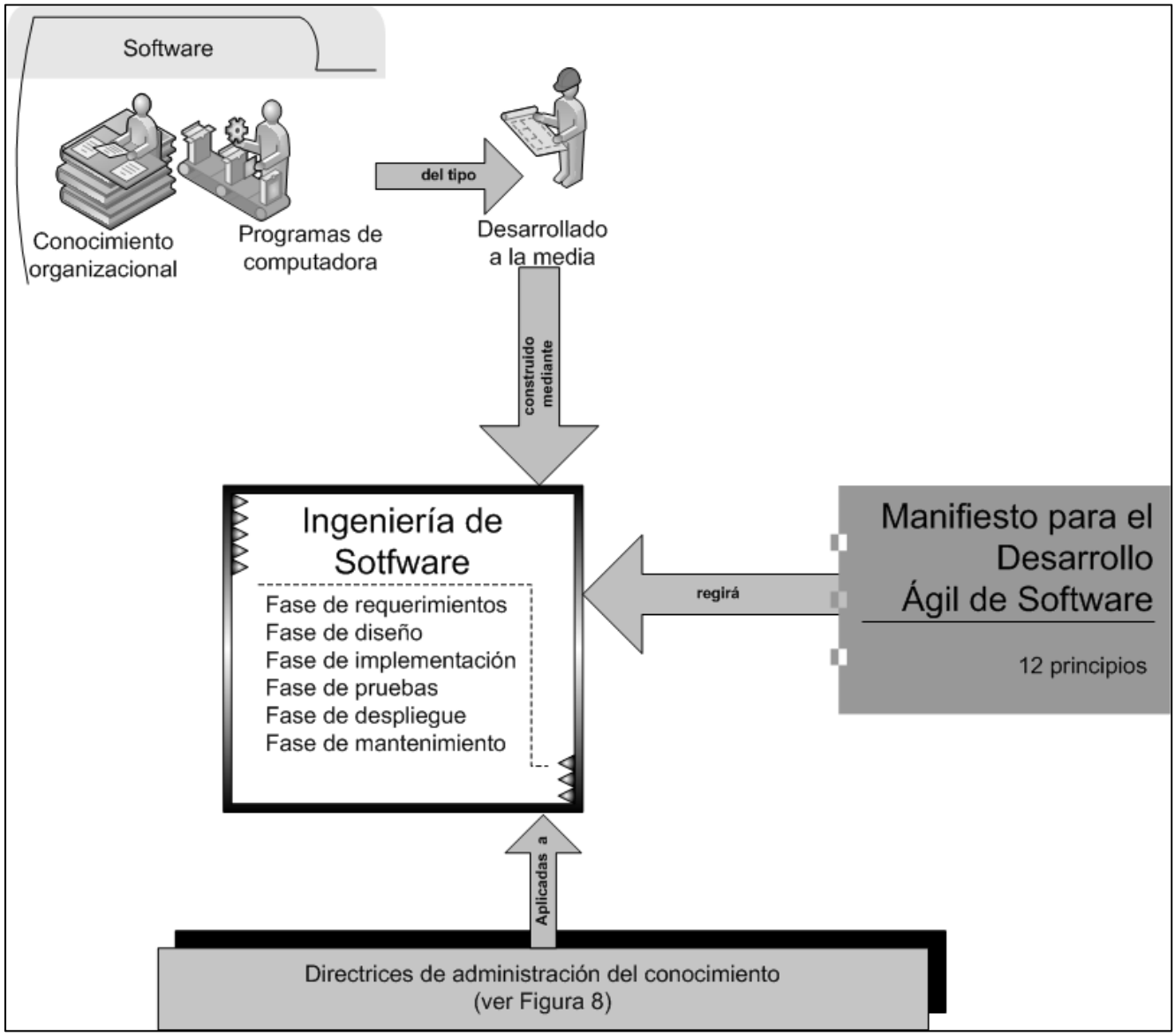
- Técnica de ajustes a la metodología: Realizando entrevistas y talleres con el equipo de trabajo para trasformar una metodología base en otra que se ajuste al proyecto.
- La técnica ya mencionada de los talleres de análisis.

A partir de los detalles anteriores se define cada una de las metodologías que forman parte de la familia cristal, cuyos detalles no se presentan por quedar afuera del alcance del trabajo. Las metodologías cristal están orientadas completamente a la ingeniería de

software, pero implican un alto grado de complejidad al tener que variar entre un proyecto y otro el tipo de metodología que será empleado ya que si de por si es complejo llevar a cabo el control y seguimiento de una metodología dentro de las empresas, teniendo un número variable de las mismas aumentará la dificultad.

2.6. Resumen de aportaciones al modelo

Figura 18. Elementos clave relacionados con la ingeniera de software para el modelo propuesto.



Fuente: Elaboración propia.

En este segundo Capítulo fue establecido que el concepto de software consiste en programas de computadora, procedimientos y posible documentación asociada, que representan la acumulación de conocimiento organizacional. Lo anterior orientado al software clasificado como soluciones empresariales o desarrollado a la medida, mediante un proceso de ingeniería de software que comprende las fases de requerimientos, diseño, implementación, pruebas, despliegue y mantenimiento. Fases a las cuales serán aplicadas

las directrices de administración del conocimiento establecidas en el Capítulo 1 para construir el modelo de la sección 3.8.3, el cual estará regido por el Manifiesto para el Desarrollo Ágil de Software junto con sus 12 principios, tal y como se representa en la Figura 18.

En el siguiente Capítulo se presentará la metodología para la creación del modelo propuesto el cual integrará los elementos establecidos en la Figura 18.

3. Metodología

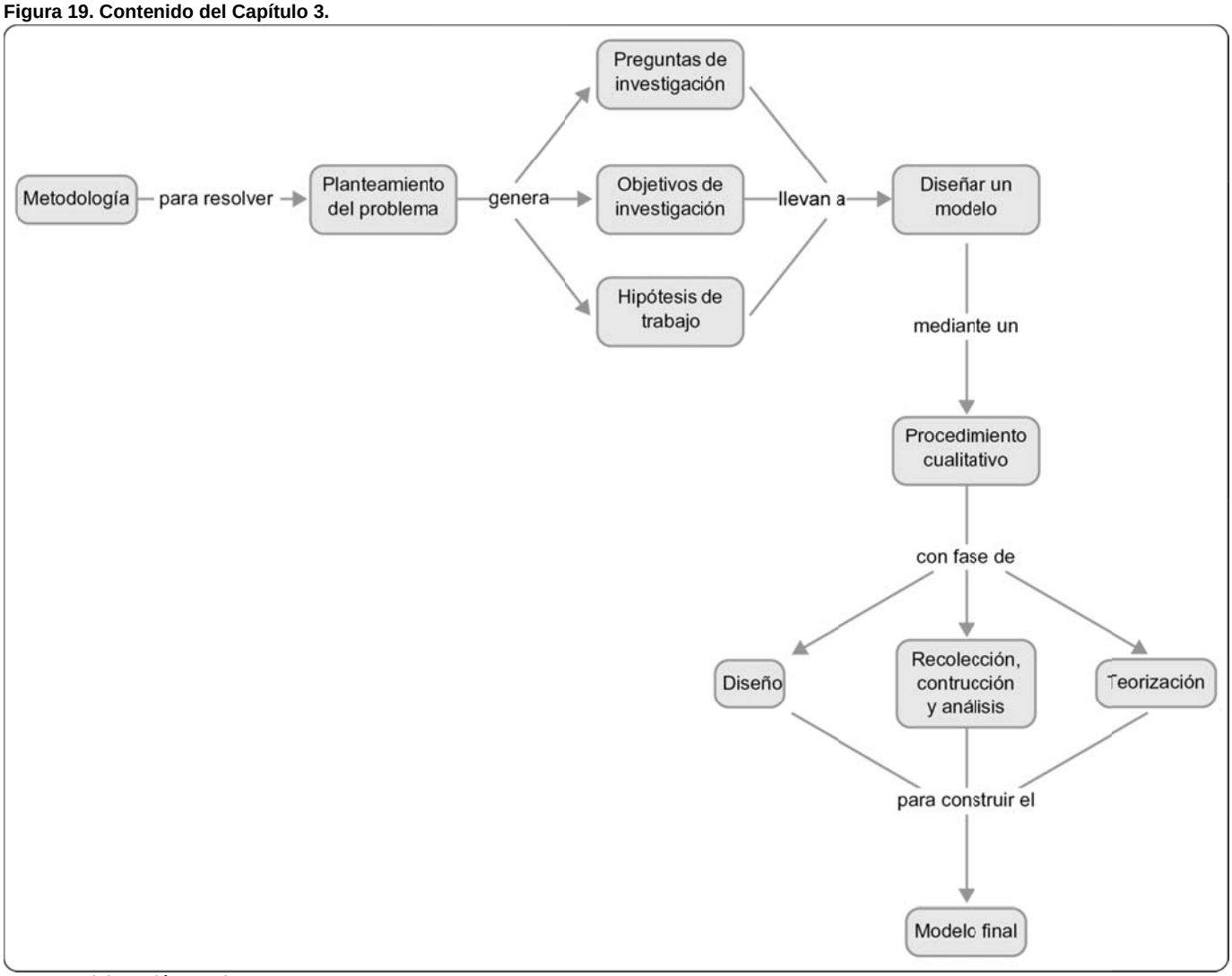
Es de importancia para quien desee alcanzar una certeza en su investigación, el saber dudar a tiempo.

Aristóteles

En el primer Capítulo se realizó una revisión de los conceptos y estructuras referentes a la administración del conocimiento, a partir de lo cual fueron establecidas las directrices mostradas en la Figura 8 que serán empleadas para aplicarse a los elementos clave relacionados con la ingeniería de software, plasmados en la Figura 18. Y para que finalmente en la sección 3.8.3 de este tercer Capítulo sea construida la propuesta del modelo de ingeniería de software con base en directrices de administración del conocimiento. Pero antes de llegar a ese punto será realizada la presentación del planteamiento formal del problema que dio origen a la tesis, un listado de preguntas que lo desglosa, los objetivos de investigación, el paso a la hipótesis de trabajo y la contrastación de los elementos (mediante el cuadro de congruencia).

Una vez realizado lo anterior se abordará el diseño del modelo comenzando con el concepto de lo que es un modelo, y la metodología empleada para crearlo, y enseguida la presentación de la propuesta del modelo, la cual una vez que fue evaluada por expertos, se le realizaron los ajustes pertinentes para el planteamiento final. En la Figura 19 se muestra de manera gráfica los temas contenidos en este Capítulo y sus relaciones.

Figura 19. Contenido del Capítulo 3.



Fuente: Elaboración propia.

3.1. Planteamiento del problema

Los sistemas de software desarrollados a la medida se han convertido en un elemento primordial en las operaciones diarias de las organizaciones. Tal y como lo mencionan Salem Ben y Mouna Ben (2009), a la fecha se han dedicado gran cantidad de trabajos orientados a los sistemas de información e ingeniería de software con el propósito de analizar los motivos de los retrasos en: proyectos, presupuestos arriba de lo planeado, productos de software de baja calidad e inclusive con funcionalidad inadecuada. Desde los años 60's se ha empleado la expresión *crisis de software*, para hacer referencia a los problemas antedichos, se han realizado varias propuestas de manera que sea posible aumentar la productividad y calidad del software al nivel requerido por el entorno en el que se encuentran inmersas actualmente las organizaciones.

Las soluciones aplicadas hasta el momento están centradas en mejoras al ciclo de vida del software con base en herramientas de desarrollo, métodos o procesos. Sin embargo, el alcance de estas soluciones suele estar limitado por enfocarse en problemas específicos y los beneficios que aportan no resultan significativos debido a que no se ha tomado en cuenta el hecho que el software es resultado de la acumulación del conocimiento poseído por las personas que forman las organizaciones, dejando de lado que la crisis del software se puede mitigar disminuyendo la brecha de conocimiento existente entre lo que saben las personas que forman una organización y el conocimiento contenido en el software. Por lo anterior, se requiere un modelo que permita integrar de forma eficiente la mayor cantidad del *know-how* de las organizaciones, manejando el proceso de desarrollo de software desde una perspectiva de la administración del conocimiento.

Algunos estudios previos relacionados con esta problemática son los siguientes:

- Mochi Alemán (2006) realizó un estudio de la industria del software en México en el contexto internacional y latinoamericano, en el que se presenta el panorama del software a nivel mundial puntualizando algunas características relevantes del sector y contrastándolas con las que prevalecen en México.

- Zapata Cant, Rialp I Criado y Rialp I Criado (2007) presentaron una investigación que examina los procesos de generación y transferencia del conocimiento, en las Pymes del sector de las tecnologías de la información con base en un modelo conceptual diseñado a partir de un estudio de casos y la validación mediante un sistema de ecuaciones estructurales.

- Ming (2009) llevó a cabo un estudio de la manera cómo Lenovo realiza su proceso de administración del conocimiento en el desarrollo de software exponiendo los éxitos y las fallas a la que se han enfrentado.

- Salem Ben y Mouna Ben (2009) elaboraron una estructura conceptual que proporciona la definición de conocimiento basada en arquitectura de sistemas de información, y describen un proceso de desarrollo de software orientado al conocimiento que ayude a las organizaciones a reducir los problemas del conocimiento asociado a sistemas de software.

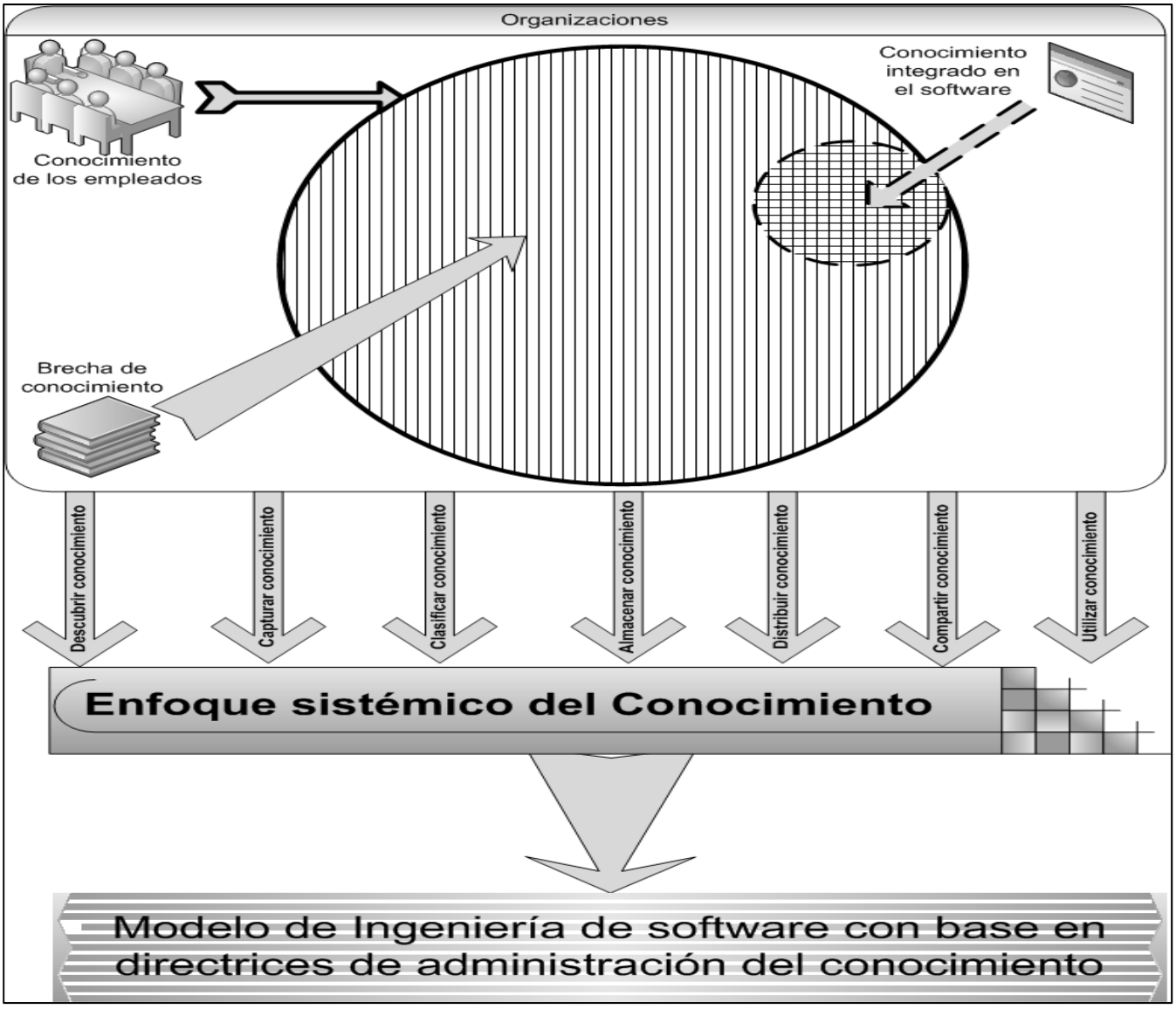
- García Garibay (2010) realizó una medición de la existencia de procesos de administración del conocimiento y el tipo tecnologías Web 2.0 en Pymes de la industria de TIC pertenecientes al clúster ProSoftware de la ciudad de México.

- Tor, Tore y Torgeir (2010) exploraron los beneficios y retos para mejorar la redundancia de conocimiento mediante la rotación de puestos con desarrolladores de software.

- Barreto Nunes y Bessa Albuquerque (2010) desarrollaron un trabajo en el que resumen la aplicación de un proceso para certificar la seguridad del software y realizan una propuesta para administrar el conocimiento generado durante este proceso.

Tomando en cuenta los detalles de la problemática, aunado a las referencias de estudios previos y la delimitación del problema, la representación gráfica del problema se presenta en la Figura 20.

Figura 20. Cuadro de análisis del problema.



Fuente: Elaboración propia.

3.2. Preguntas de investigación

3.2.1. General

¿Qué elementos debe tener un modelo de ingeniería de software con base en directrices de administración del conocimiento?

3.2.2. Específicas

a) ¿Cuáles son las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento?

b) ¿Qué actividades se deben ejecutar en el proceso de ingeniería de software para integrar eficientemente conocimiento de los empleados en el mismo?

c) ¿Es posible considerar redundancia de información para integrar el conocimiento al software?

d) ¿Mediante qué herramientas es posible almacenar el conocimiento tácito de los desarrolladores de software?

3.3. Objetivos de investigación

3.3.1. General

Diseñar un modelo para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento.

3.3.2. Específicos

a) Definir las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento.

b) Establecer el conjunto de actividades en el proceso de ingeniería de software para integrar eficientemente el conocimiento de los empleados en el mismo.

c) Evaluar la viabilidad de emplear redundancia de información para integrar el conocimiento en el software.

d) Identificar las herramientas que permitan almacenar conocimiento tácito.

3.4. Hipótesis de trabajo

H: "Es posible diseñar un modelo para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento"

3.5. Cuadro de congruencia

Tabla 2. Cuadro de congruencia.

Objetivo general	
Diseñar un modelo para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento.	
Pregunta de investigación	
¿Qué elementos debe tener un modelo de ingeniería de software con base en directrices de administración del conocimiento?	
Hipótesis de trabajo	
H: Es posible diseñar un para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento.	
Objetivos específicos	**Preguntas de investigación**
a) Definir las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento.	a) ¿Cuáles son las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento?
b) Establecer el conjunto de actividades en el proceso de ingeniería de software para integrar eficientemente el conocimiento de los empleados en el mismo.	b) ¿Qué actividades se deben ejecutar en el proceso de ingeniería de software para integrar eficientemente conocimiento de los empleados en el mismo?
c) Evaluar la viabilidad de emplear redundancia de información para integrar el conocimiento en el software.	c) ¿Es posible considerar redundancia de información para integrar el conocimiento al software?
d) Identificar las herramientas que permitan almacenar conocimiento tácito.	d) ¿Mediante qué herramientas es posible almacenar el conocimiento tácito?

Fuente: Elaboración propia.

3.6. Taxonomía del estudio

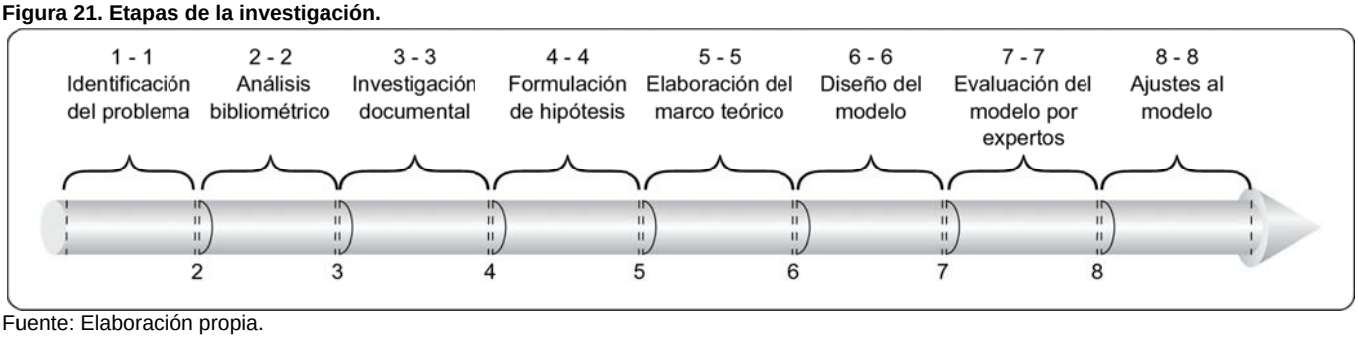
- No Experimental: No se realizó manipulación deliberada de variables.
- Exploratorio: Fue identificada la relación, poco estudiada hasta el momento, entre conceptos clave de la ingeniería de software y la administración del conocimiento, plasmando dicha relación en la propuesta del modelo de ingeniería de software con base en directrices de administración del conocimiento, para que se lleven a cabo investigaciones futuras en las que la línea de investigación sea el estudio de la aplicación del modelo.
- De corte transversal: La información fue recolectada sólo una vez sin pretender evaluar su evolución a través del tiempo.
- Cualitativo: La validación del modelo propuesto se llevo a cabo mediante entrevistas cualitativas a un grupo de expertos en el desarrollo de software a la medida, los detalles respecto a lo anterior se encuentran contenidos en la sección 3.8.4.

3.7. Diseño de la investigación

3.7.1. Etapas

Las etapas consideradas para la investigación se muestran en la Figura 21 acorde a la secuencia en la que se deben de ejecutar.

Figura 21. Etapas de la investigación.



Fuente: Elaboración propia.

3.7.2. Procedimiento

De manera general las actividades consideradas para cada una de las etapas son las siguientes:

- *Identificación del problema:* Definición de un problema actual, cuya relevancia aporte la solución de una situación importante para las organizaciones, el cual fue presentado en la sección 3.1.
- *Análisis bibliométrico:* Aplicación de métodos matemáticos y estadísticos para el análisis de publicaciones científicas relacionadas con el problema identificado, los resultados se presentan en el apéndice A.
- *Investigación documental:* Búsqueda de referencias relacionadas con el problema identificado, ya sea mediante tesis, artículos o estudios, así como teorías y modelos para la resolución del problema, las cuales son presentadas a lo largo del presente trabajo mediante la alusión a las diversas fuentes de información presentadas en la bibliografía.
- *Formulación de hipótesis:* Para el objetivo establecido se plantea una hipótesis de su posible cumplimiento, la cual se muestra en la sección 3.4.
- *Elaboración del marco teórico:* Establecimiento del contexto teórico y conceptual bajo el cual será establecido el modelo propuesto, desarrollado en los Capítulos 1 y 2.
- *Diseño del modelo:* Elaboración de una propuesta de modelo para llevar a cabo el proceso de ingeniería de software con base en directrices de administración del conocimiento, lo cual es desarrollado a través de la sección 3.8.3.
- *Evaluación del modelo por expertos:* El modelo propuesto se presenta a personas con 5 o más años experiencia en la industria de desarrollo de

software en puestos gerenciales o directivos así como desarrollares de software con experiencia de 10 o más años, teniendo la finalidad de recibir retroalimentación del modelo, los detalles de esta etapa son enunciados en la sección 3.8.4.

- *Ajustes al modelo:* A partir de la retroalimentación recibida por los expertos se llevan a cabo los ajustes pertinentes al modelo propuesto, este punto es desarrollado en el Capítulo 4.

3.8. Diseño del modelo

Una vez que ya han tratado los elementos teóricos y conceptuales necesarios para realizar la construcción del modelo, en las siguientes secciones de este aparado será presentado el producto principal para cumplir con el objetivo establecido en la sección 3.3.1, comenzando por detallar el concepto de lo que es un modelo para posteriormente explicar el procedimiento usado para crearlo y finalmente realizar el desarrollo correspondiente del mismo.

3.8.1. ¿Qué es un modelo?

En el *Diccionario del* español *de México*, creado por el Colegio de México (2010), en su cuarta acepción la palabra “modelo” se define como:

> Presentación esquemática, abstracta, más pequeña o más simple que se hace de un objeto complejo para facilitar su comprensión, por ejemplo: un modelo matemático, un modelo a escala, el modelo del átomo, el modelo del ADN.

MiTecnologico.com (2007) menciona que un modelo es una estructura conceptual que sugiere un marco de ideas para un conjunto de descripciones que de otra manera no podrían ser sistematizadas y que el modelo concebido en esta forma impulsa la inteligibilidad y ayuda a la comprensión de los fenómenos, ya que proporciona los canales de interconexión entre hechos que sin la existencia de los lazos inferenciales permanecerían aislados e independientes unos de otros.

Bajo estas dos visiones, el modelo propuesto es una representación esquemática que permite definir la estructura conceptual para ejecutar el proceso de ingeniería de software enlazándola con directrices de administración del conocimiento.

3.8.2. Procedimiento para construcción del modelo

Bolsegui y Fuguet (2006) plantearon un procedimiento para la construcción de un modelo conceptual por medio de la investigación cualitativa, justificándose en: que las características para el estudio de fenómenos sociales son la relevancia contextual, énfasis en todo el proceso

y no solamente en el control y experimentación, ya que es necesario orientarse en el desarrollo de teorías que expliquen datos, y no en la búsqueda de datos que correspondan a una teoría previa. Bajo estas premisas el ser humano es el instrumento principal para la recolección de datos, ya que se reconoce la construcción del conocimiento tácito de los investigadores así como que la investigación se sustenta por el sistema de valores que caracterizan al investigador, el informante, el paradigma que se elija y la teoría sustantiva que se relaciona, la cual puede orientar tanto los métodos de recolección y análisis como las formas de interpretar los hallazgos.

Debido a que tanto en los procesos de ingeniería de software como en la construcción del conocimiento necesariamente se involucran personas, se pueden considerar esos procesos como fenómenos sociales para los cuales la investigación cualitativa representa una opción viable.

El rol de investigador es una función que Bolsegui y Fuguet (2006) mencionan que es el principal instrumento de la investigación cualitativa, y no los métodos, como en la investigación tradicional, ya que es su capacidad hermenéutica la que define su facilidad para establecer vínculos entre las conexiones y los sujetos, redundando en que la intuición y creatividad se convierten en elementos claves del proceso de investigación.

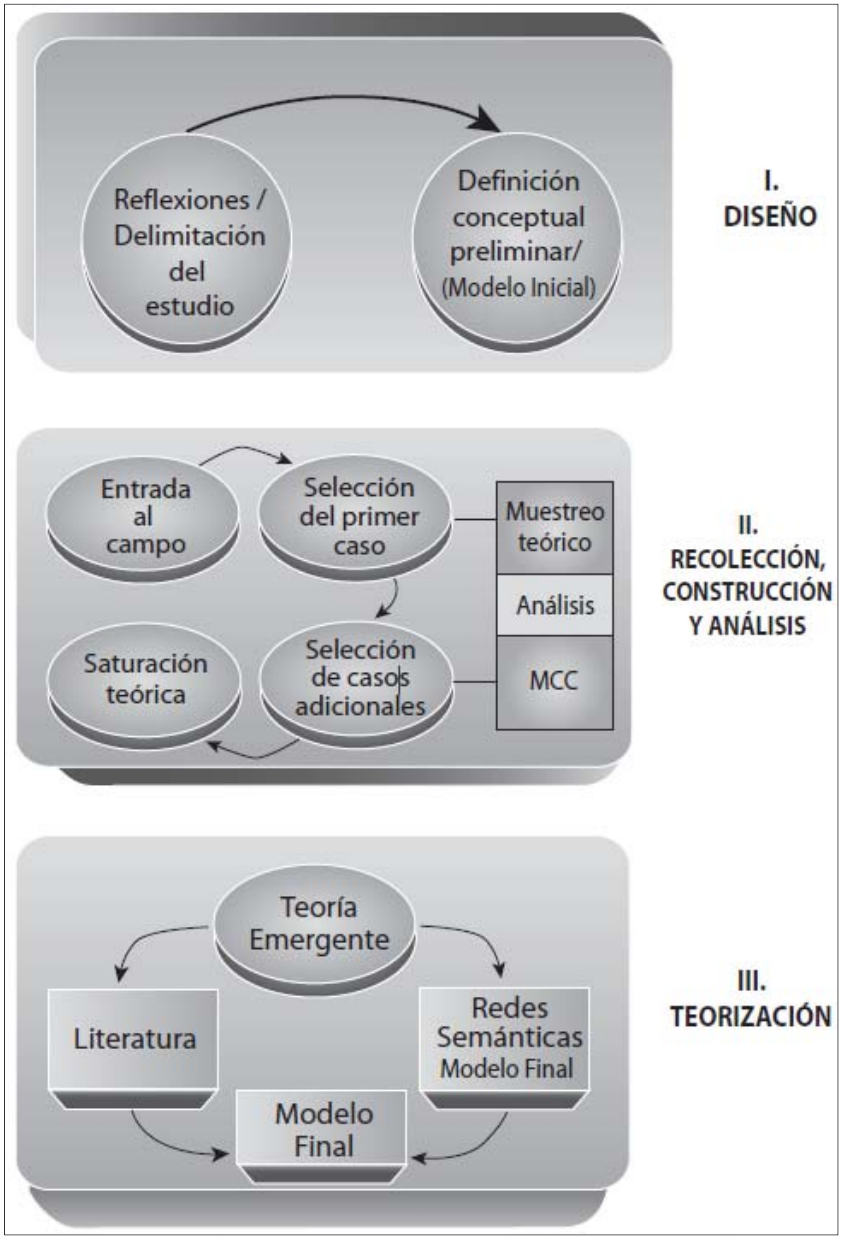
El proceso de la investigación cualitativa propuesto por Bolsegui y Fuguet (2006) se compone de las fases de: (1) diseño, (2) recolección, (3) construcción, (4) análisis (5) teorización y (6) construcción del modelo. Los componentes de cada fase se muestran en la Figura 22 y serán la estructura base para el diseño de la investigación cualitativa que permitirá evaluar el modelo.

3.8.2.1. Fase de diseño

De acuerdo con la propuesta de Bolsegui y Fuguet (2006) el diseño de una investigación cualitativa es emergente y en cascada, se elabora en la medida que avanza la investigación, es decir, es un diseño flexible. El problema de investigación da lugar a un cuestionamiento y reformulación constante, que guía y establece los límites de la investigación. El diseño surge en la medida que la investigación se desarrolla. En esta fase se define el objeto de estudio, se formulan las preguntas de investigación, se precisan los objetivos, generándose un entorno reflexivo sobre las motivaciones de la investigación iniciándose un marco referencial preliminar que es enriquecido con apoyo de la literatura especializada para permitir un esbozo del modelo inicial.

Una vez iniciada la investigación, se van decidiendo otros aspectos y escenarios a estudiar. La incorporación progresiva de nuevos informantes o escenarios se relaciona con los intereses de la investigación, en términos de lo que se ha aprendido. En la Figura 23 se muestra el esquema del diseño general de la investigación de Bolsegui y Fuguet (2006).

Figura 22. Fases del proceso de investigación cualitativa.



Fuente: Bolsegui, M., & Fuguet , S. A. (2006). *Construcción de un modelo conceptual a través de la investigación cualitativa*. Sapiens, 207 - 229.

3.8.2.2. Recolección, construcción y análisis

En la investigación cualitativa el análisis y la interpretación son procesos vinculados, el propósito es el de identificar el contenido de los relatos acordes con un paradigma o modelo semántico, por ello se considera que el análisis se desarrolla durante toda la investigación.

En lo que se refiere a la recolección de datos el investigador mantiene una interpretación fundamentada, extrayendo sus conclusiones a partir de las observaciones que realiza y de otros datos, intentando preservar las realidades múltiples en las diferentes, y a veces, contradictorias visiones de lo que sucede. Se recomienda que en el proceso de análisis de

entrevistas, en lugar de permitir que los datos hablen por si mismos, el analista documente empíricamente el proceso de construcción de significados y describa las actividades discursivas complejas, de forma que no sólo resuma y organice lo expresado por el entrevistado, sino que deconstruya los significados, es decir: que el investigador secuencie la acción, categorice propiedades, y posteriormente haga recuentos para sumarlos de forma intuitiva. (Bolsegui y Fuguet, 2006)

Figura 23. Diseño General de Investigación.

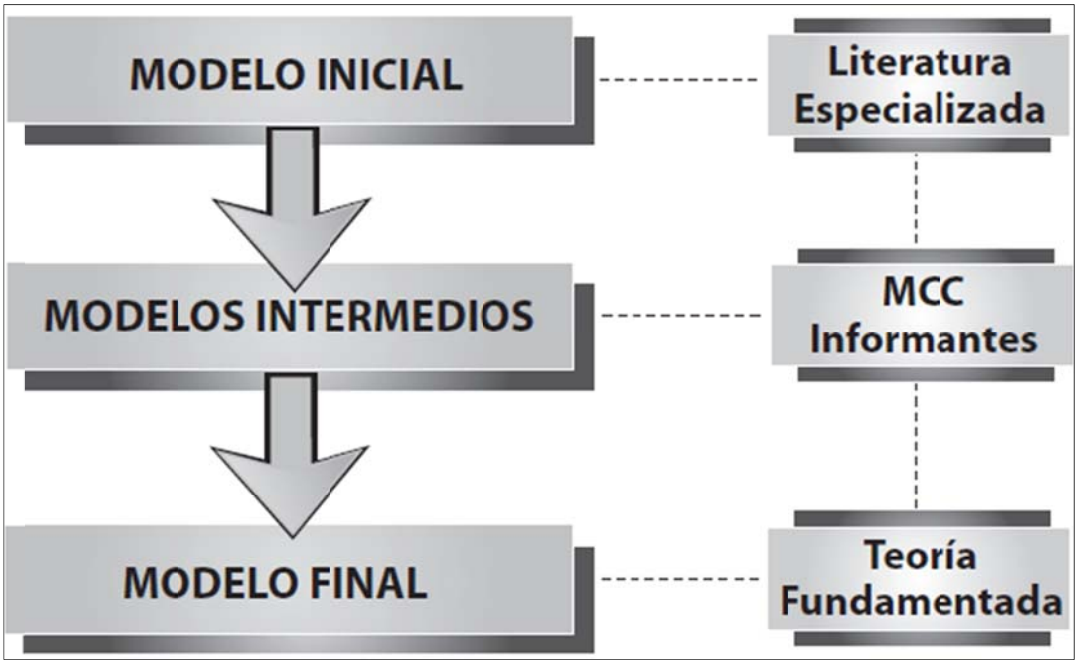
Fuente: Bolsegui, M., & Fuguet , S. A. (2006). *Construcción de un modelo conceptual a través de la investigación cualitativa*. Sapiens, 207 - 229.

3.8.2.3. Teorización y construcción del modelo

Acorde a lo expuesto por Bolsegui y Fuguet (2006), contamos con tres niveles de modelos como se muestra en la Figura 24. El primero de ellos consiste en un modelo inicial propuesto

por el investigador con base en la literatura especializada, posteriormente se tienen los modelos intermedios que surgen de la aplicación del método de comparativo continuo (MCC), el cual, a grandes rasgos, consiste en que al aplicar cada entrevista se contraste la visión del entrevistado con el modelo propuesto y/o el modelo ajustado por una entrevista previa; de forma que se integren las aportaciones y se formule una teoría emergente que nos llevará al modelo final sustentando por una teoría fundamentada. Teoría que resulta del trabajo y creatividad del investigador para establecer las respectivas comparaciones en términos de similitudes y diferencias.

Figura 24. Niveles de modelos.



Fuente: Bolsegui, M., & Fuguet , S. A. (2006). *Construcción de un modelo conceptual a través de la investigación cualitativa*. Sapiens, 207 - 229.

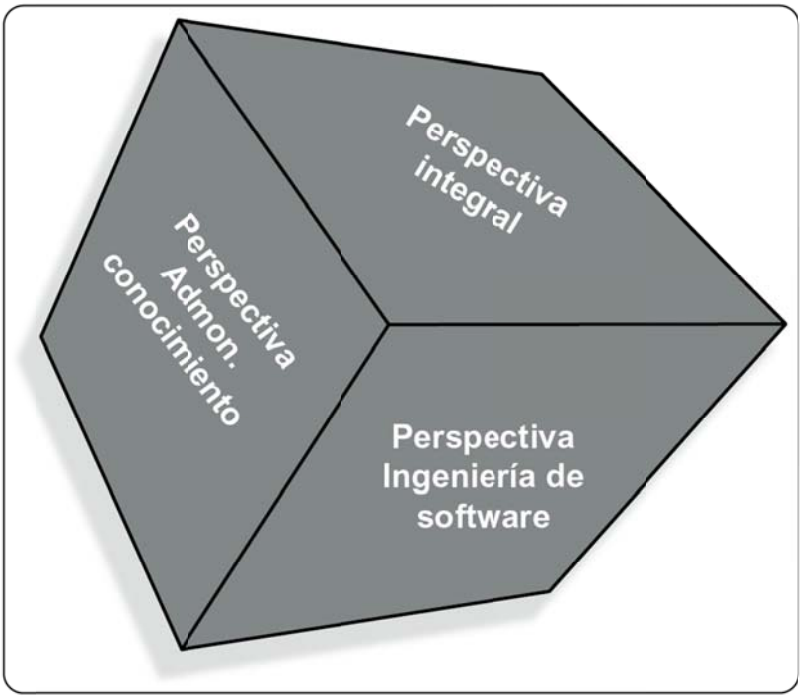
3.8.3. Modelo propuesto

En la Figura 22 se muestra gráficamente que, en la fase de diseño de un modelo la definición conceptual preliminar que da lugar al modelo inicial se fundamenta en las reflexiones y delimitación del estudio, tal definición es estructurada por medio del planteamiento del problema de investigación, así como en los objetivos y preguntas relacionadas con la misma, (elementos que fueron tratados y contrastados del apartado 3.1 al 3.5). Por otro lado, en la Figura 24 se establece una relación del modelo inicial con la literatura especializada, cuya revisión fue llevada a cabo de forma inicial con el protocolo de tesis para posteriormente complementar y enriquecer el respectivo marco teórico con los aspectos que conforman los Capítulos 1 y 2, por lo tanto, tomando como base esta serie de elementos el modelo inicial propuesto consta de tres perspectivas lo que se muestra en la Figura 25.

Un cubo es posible apreciarlo desde cada una de sus 6 caras y no por el hecho de que alguna de sus caras muestre algún color o diseño distinto de las demás, dejan de formar parte

del mismo cubo por lo cual a partir de esa idea es que se plantea un modelo que sea posible abordar con un enfoque hacia la ingeniería software otro para la administración del conocimiento así cómo la integración de ambos elementos. La intención de mostrar el modelo con tres perspectivas es que sea posible detallar por separado algunos aspectos relevantes para la ingeniería de software y la administración del conocimiento que serían complicados visualizar en la perspectiva integral de la Figura 26.

Figura 25. Perspectivas del modelo propuesto.

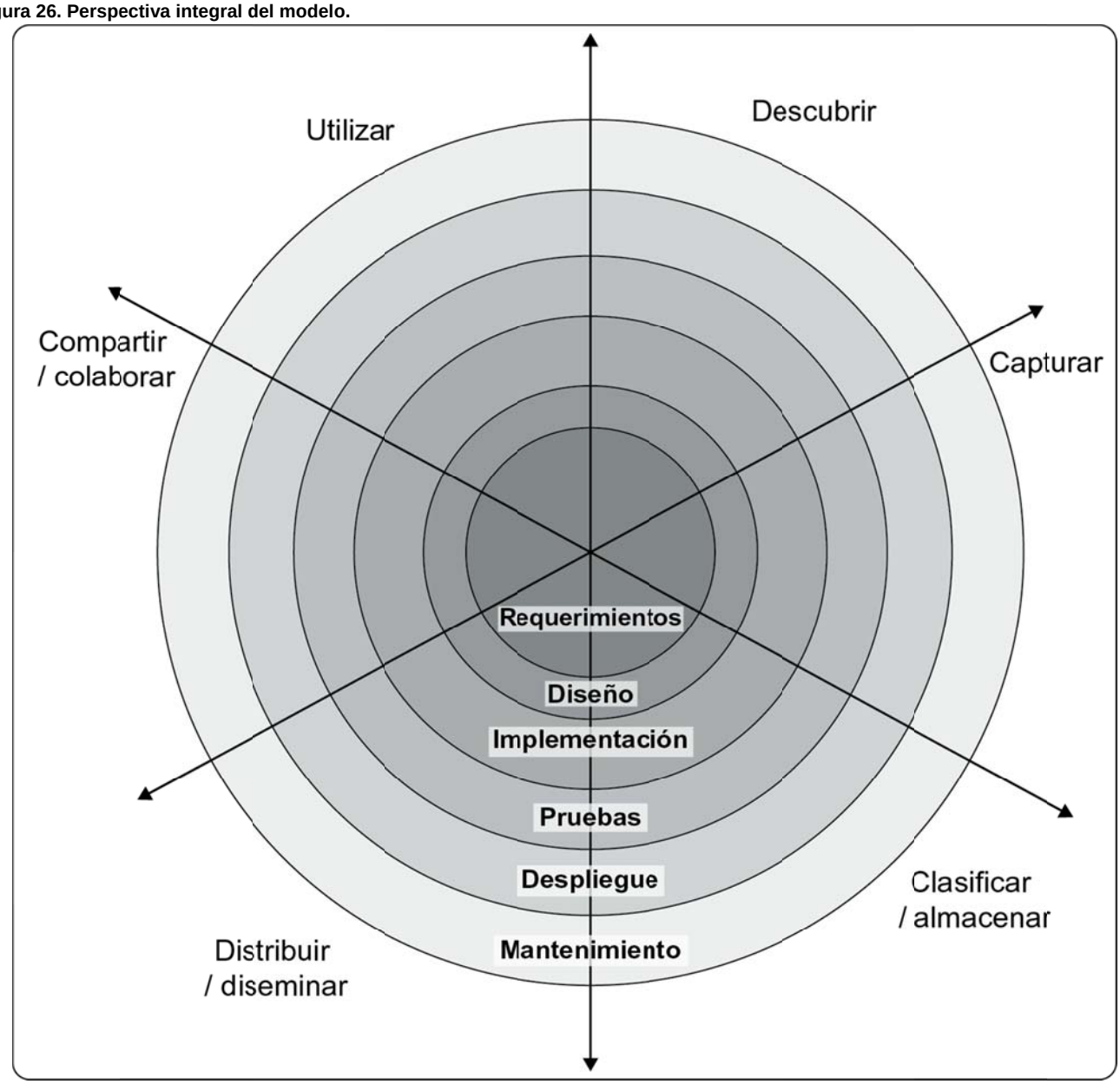


Fuente: Elaboración propia.

3.8.3.1. Perspectiva integral

Lo que se pretende transmitir con la perspectiva integral del modelo, presentada de manera gráfica en la Figura 26, es la conjunción de las fases de ingeniería de software representadas por los círculos concéntricos, con los procesos del conocimiento, mostrados en las secciones delimitadas mediante flechas en cada uno de los círculos. De tal forma que en las fases de ingeniería de software se ejecuten formalmente los procesos del conocimiento que sean requeridos. Es decir, no se deben realizar los procesos del conocimiento de manera cíclica para pasar a la siguiente fase, sino simplemente aquellos que ayuden a obtener de manera exitosa los entregables definidos o bien los que se requieran para almacenar los conocimientos generados en la base de conocimientos establecida en la perspectiva de administración del conocimiento del modelo (sección 3.8.3.3).

Figura 26. Perspectiva integral del modelo.



Fuente: Elaboración propia.

La formalización del uso de los procesos del conocimiento en cada una de las fases de ingeniería de software queda estructurada de la siguiente manera:

- *Utilizar:* Este proceso estará presente en todas las fases ya que para obtener los entregables que sean definidos para cada una, no hay otra manera más que utilizando el conocimiento que poseen las personas, el generado en cualquiera de las fases o almacenado en la base de conocimientos. Todos los entregables deben contener la referencia de los elementos empleados en la base de conocimientos. A su vez, en la base se requiere listar los entregables que han empleado los conocimientos que contiene. Por lo tanto, si algún conocimiento no está registrado en la base es necesario capturarlo para cumplir con lo anterior.

- *Capturar:* La ejecución de este proceso sucederá en cualquiera de las fases en las que se descubran y/o creen nuevos conocimientos.
- *Clasificar/Almacenar, Distribuir/Diseminar:* Se deben ejecutar estos procesos de manera conjunta en las fases en las que se capturen conocimientos nuevos en la base.
- *Compartir/Colaborar:* Si en una fase determinada se descubre que el conocimiento necesario para obtener los entregables sólo está disponible por medio de otras personas dentro empresa es cuando se deben realizar estos procesos de intercambio.
- *Descubrir: La presencia de este proceso* dependerá de las siguientes condiciones para las siguientes fases:
    - Fase de requerimientos: En caso de que no se cuente con el conocimiento necesario para evaluar la factibilidad de las solicitudes.
    - *Fase de diseño:* Si no se tienen los conocimientos para realizar el diseño del software.
    - Fase de implementación: Cuando los programadores no cuenten con el conocimiento requerido para desarrollar el software.
    - Fase de pruebas: Cuando las personas encargadas de ejecutar las pruebas no conozcan las condiciones del comportamiento correcto del software.
    - Fase de despliegue: En caso de que los responsables de migrar el software a un ambiente productivo no sepan el procedimiento para completar su tarea.
    - Fase de mantenimiento: Para saber qué conocimientos ya están contenidos en el software y valorar los ajustes que se requieren.

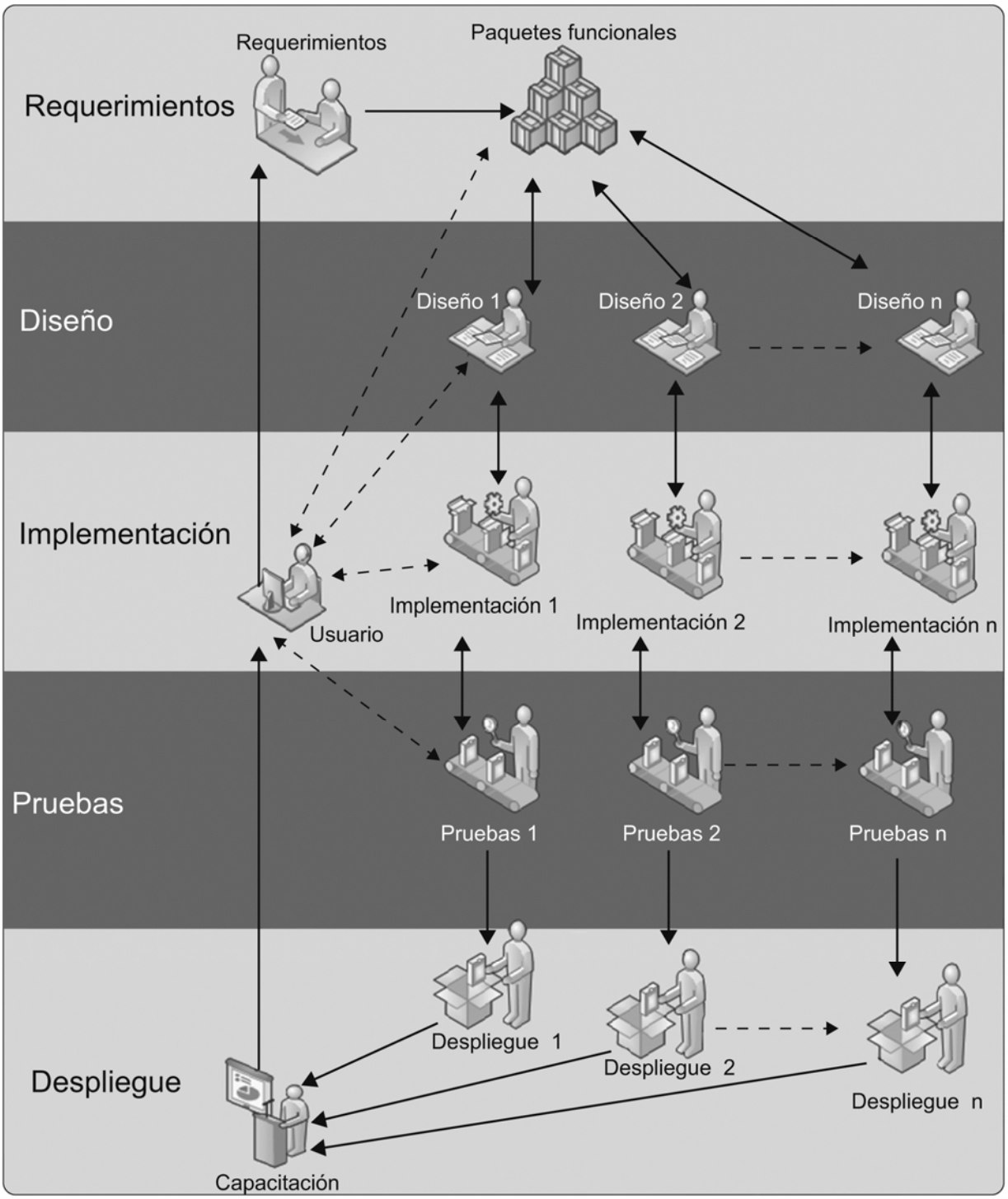
Con la intención de que no se presenten confusiones es prudente recordar que con la palabra “conocimiento” en la lista anterior, se hace referencia al conocimiento de negocio o procesos internos de la empresa, no a cuestiones técnicas relacionadas con la construcción del software.

3.8.3.2. Perspectiva de ingeniería de software

En la visión de la perspectiva de ingeniería de software mostrada en la Figura 27, la fase de mantenimiento no es enunciada de manera explícita, ya que las actividades para llevarla a cabo implican un nuevo ciclo de las fases que son consideradas en dicha Figura. Otro elemento implícito que se debe tomar en cuenta en esta perspectiva es la validación de los procesos de administración del conocimiento implicados en cada una de las fases acorde con las reglas y condiciones especificadas en la perspectiva integral de la sección 3.8.3.1, de tal forma que se cuente con el conocimiento necesario en cada una de las fases para obtener los entregables requeridos, o realizar la codificación de nuevos conocimientos y documentar la relación entre estos, así como los existentes con los entregables.

Pasando de lleno a los detalles de la perspectiva de ingeniería de software; la primer regla a cumplir es que cualquier interacción entre personas se debe realizar cara a cara, o en su defecto con el medio más directo que sea posible (video conferencia, conversación telefónica, etc.) y simplemente se deben formalizar los acuerdos, responsabilidades futuras y/o hallazgos mediante una minuta o un correo electrónico breve que debe almacenarse en un repositorio de documentación destinado a cada solicitud.

Figura 27. Perspectiva de ingeniería de software.



Fuente: Elaboración propia.

Adicional a lo mencionado en el apartado 2.3 respecto a las fases de ingeniería de software, las particularidades de cada una, bajo el modelo propuesto, son las siguientes:

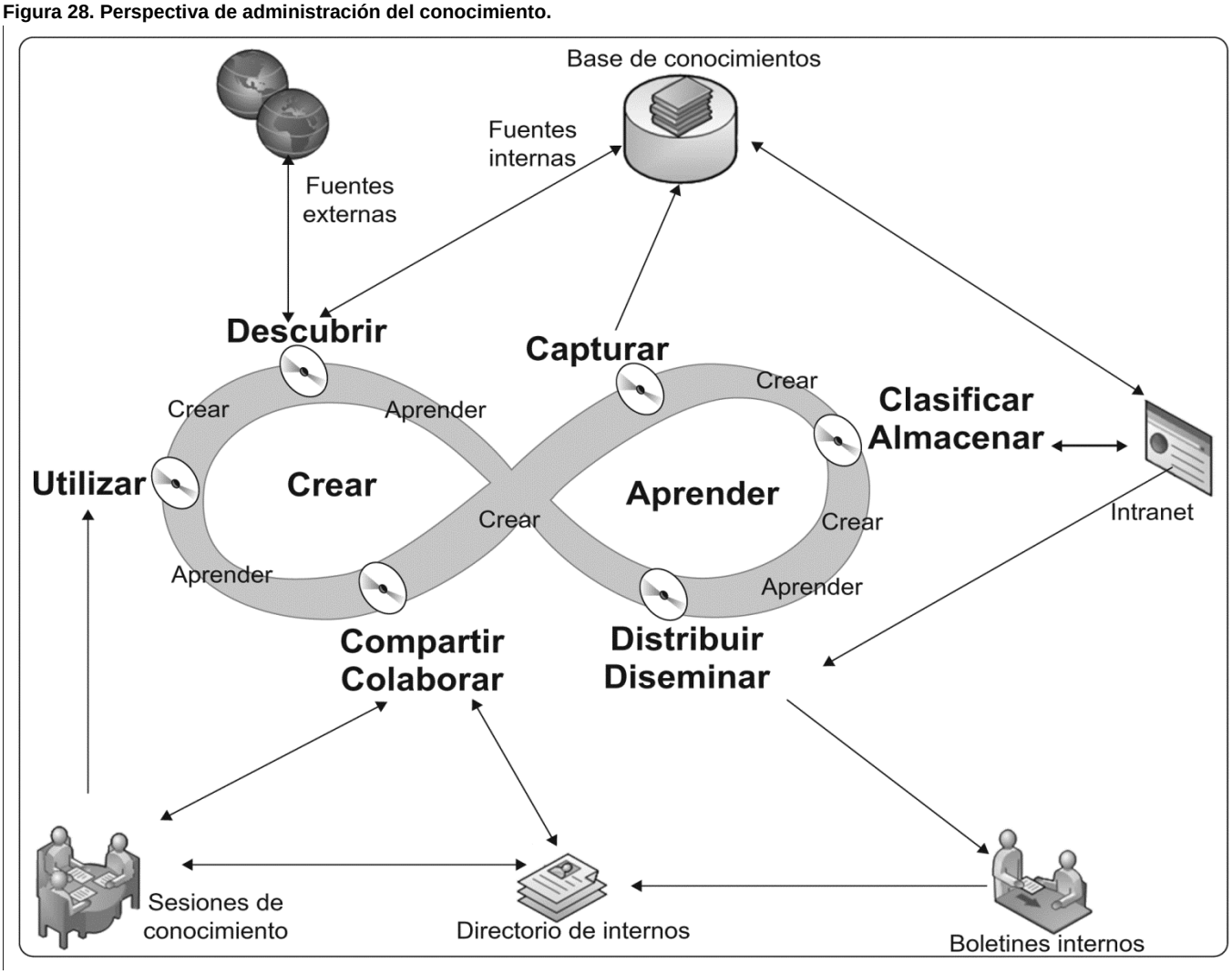
- *Fase de requerimientos:* Se deben identificar las diferentes funcionalidades individuales involucradas para cumplir con el alcance definido y resolver la solicitud del usuario, de manera que sea posible establecer conjuntos de requerimientos que puedan desarrollarse de forma independiente, formando lo que se denominarán como “paquetes funcionales”. La intención de manejar estos paquetes es que en el tiempo más corto posible el usuario pueda tener disponible la funcionalidad que considere más importante, por lo cual una vez que se hayan definido los paquetes es necesario consultarle respecto a que establezca una priorización de los mismos para establecer el orden en que le será entregado cada paquete, o en caso de que la organización cuente con recursos suficientes aplicar el modelo de Desarrollo Rápido de Aplicaciones (explicado en la sección 2.4.4).
- *Fases de diseño, implementación y pruebas:* Al final de cada una de estas fases se debe revisar con el usuario los entregables para certificar la inclusión de los elementos necesarios para que el software que esta siendo producido en cada paquete funcional cumpla sus expectativas y mantenga la prioridad adecuada para continuar con el proceso y en caso de no ser así, se debe replantear la implementación, el diseño o composición de los paquetes funcionales junto con su prioridad para satisfacer las expectativas del usuario. Es muy importante que lo anterior se maneje plasmando de manera clara los impactos en costo, tiempo y recursos que tendrán, pero de ninguna manera cerrándose a los cambios, ya que, si bien como se acaba de mencionar se presentará afectación respecto a los planes originales, es menos costoso realizar los ajustes necesarios a tiempo, que entregar un software que no le sea útil al usuario o inclusive le ocasione más problemas de los que le resuelva. Los entregables que se presenten al usuario deben ser aquellos que no tengan implicaciones técnicas y le sean entendibles de manera que pueda realizar la respectiva validación.
- *Fase de despliegue:* Cada vez que el software correspondiente a un paquete funcional sea desplegado en ambientes productivos, donde el usuario tenga acceso al mismo para sus actividades diarias. Además de los manuales de usuario que indiquen los detalles del uso del software se requiere llevar a cabo reuniones de capacitación para explicarle el manejo adecuado del software y dejar abierto algún canal de comunicación para resolver dudas.

El hecho de realizar la segmentación de los requerimientos en paquetes funcionales permitirá ejecutar un *mantenimiento activo* del software, ya que no es necesario esperar a que el alcance de la solicitud se complete al 100% para detectar puntos de mejora o errores, debido a que el usuario obtendrá entregas incrementales del software teniendo como resultado que le sea posible trabajar con el mismo, en condiciones de operación diaria que ayuden a identificar nuevas necesidades o ratificar las definidas previamente en su requerimiento.

3.8.3.3. Perspectiva de administración del conocimiento

La tercera perspectiva del modelo, corresponde a la administración del conocimiento. Ya en la sección 3.8.3.1 se explicó la forma como se interrelacionan los procesos del conocimiento con las fases de la ingeniería de software, y en este apartado, se pretende resaltar las herramientas que pueden dar soporte a la administración del conocimiento y permiten establecer una interconexión entre los procesos y las personas, lo que se muestra en la Figura 28.

Figura 28. Perspectiva de administración del conocimiento.



Fuente: Elaboración propia.

Las herramientas propuestas son las siguientes:

- *Fuentes externas de conocimiento:* En caso de que dentro de la empresa no se identifiquen fuentes que permitan obtener el conocimiento para cumplir el alcance definido para una solicitud de desarrollo de software se puede recurrir a entes externos a la empresa: fuentes confiables en internet, organizaciones y revistas especializadas, instituciones gubernamentales, empresas de consultoría, etc.

- *Base de conocimientos:* Constituye el activo más preciado en el que convergen las tres perspectivas que conforman el modelo, ya que dentro de ella se encontrará codificado el conocimiento poseído por las personas que forman parte de las organizaciones y será el instrumento que permita establecer la relaciones entre el software y el conocimiento organizacional en ambos sentidos, pues, por un lado, dentro de los documentos que contenga la base de conocimientos se deben especificar los entregables de cada fase de la ingeniería de software (incluyendo los artefactos de software) que usan o implementan determinados conocimientos, y, por otro lado, dentro de los entregables se debe considerar un apartado para indicar los conocimientos empleados mediante algún tipo de clave que permita acceder a ellos por medio de la base de forma sencilla. La base de conocimientos se puede estructurar de varias formas: carpetas compartidas en computadora dentro de la empresa a la que todos los empleados tengan acceso, alguna aplicación de software para el manejo de repositorios de archivos, o bien, software construido explícitamente para el manejo de bases de conocimientos.
- *Intranet:* Pueden emplearse como medio para manejar la clasificación y almacenamiento de los componentes contenidos en la base de conocimientos, de manera que sea más fácil localizarlos y represente un punto de partida para su distribución. Así mismo pueden emplearse como filtro de acceso para conocimientos sensitivos de las organizaciones.
- *Boletines internos:* Ayudan a difundir periódicamente nuevos conocimientos relevantes de la operación diaria de las organizaciones y a inducir una cultura que fomente en los empleados compartir el conocimiento personal que han adquirido a través de la experiencia.
- *Directorio de expertos*: Resulta complicado que en etapas iniciales de implantación de proyectos de administración del conocimiento, las bases donde se almacenen puedan contener todos los detalles requeridos, o bien, no se encuentren codificados. Es por ello que mediante un directorio de los expertos de las áreas de negocio en la empresa, puede ser posible establecer las relaciones entre las personas que tienen preguntas con las que poseen las respuestas.
- *Sesiones de conocimiento:* Son reuniones cara a cara, o en su defecto, mediante el canal de comunicación más directo que se pueda establecer para que los expertos en un tema apoyen a personas que requieran el conocimiento que poseen y no se encuentre codificado dentro de la base.

3.8.4. Validación del modelo por expertos

3.8.4.1. Instrumento de medición

Apegado a lo expuesto por Bolsegui y Fuguet (2006), el instrumento para medir y mejorar el modelo propuesto, es la entrevista cualitativa, porque dada la naturaleza del problema así como la solución propuesta, se considera que la mejor manera de enriquecer el modelo es tomando en cuenta de forma abierta los diversos puntos de vista de los expertos (informantes), reconociendo la valía de su experiencia y conocimiento tácito, situaciones que resultan

complicadas obtener empleando un instrumento formal que se concentre en aspectos definidos previamente.

Se realizó la invitación de participar en la investigación a 10 informantes y a las personas que confirmación su participación, se les envió, vía correo electrónico, un resumen con los elementos teóricos más significativos del proyecto de tesis (contenido en el apéndice B), los Capítulos 1 y 2 (para las personas que desearan profundizar en la información), y el modelo inicial planteado en la sección 3.8.3, dejando abierta la comunicación para resolver cualquier duda respecto a los documentos enviados.

Con los elementos anteriores se solventa la validez de contenido, ya que acorde, con lo mencionado por Hernández Sampieri, Fernández Collado, y Baptista Lucio (1997), es acotado el dominio a la ingeniería de software en conjunto con la administración del conocimiento. Una vez enviada la información fue acordada una entrevista con cada uno de los informantes para obtener la retroalimentación del modelo.

Las entrevistas realizadas se condujeron desde una perspectiva cualitativa, definidas por Bolsegui y Fuguet (2006) como una conversación de carácter profesional guiada por un tema central, en la que el entrevistador supone que el entrevistado elabora significados en relación con otras personas, por cuanto el sujeto entrevistado pertenece a distintas redes donde se habla el tema abordado. Otra característica de la entrevista es que fue semiestructurada, es decir, que hubo cierto grado de control acerca de los temas sobre los que se conversaron con base en un guion (mostrado en el Apéndice C) abarcando el modelo propuesto y sus distintos subtemas, pero de manera flexible, más como apoyo para ejecutar las entrevistas que como una regla restrictiva.

Estas entrevistas ayudaron comprobar la validez del constructo, representado por el modelo en sí mismo, el cual establece y especifica la relación teórica entre los dominios de la ingeniería de software y la administración del conocimiento, planteando la correlación entre ambos conceptos y mediante los resultados que se presentan en la sección 3.8.4.3 se obtuvo la evidencia empírica para validarlo, por lo cual se cumple con lo establecido por Hernández Sampieri, Fernández Collado, y Baptista Lucio (1997) en referencia a la validez de constructo.

3.8.4.2. Selección de los informantes

Bolsegui y Fuguet (2006) mencionan que en el muestreo cualitativo, a diferencia de otros tipos de muestreo, la estructura definitiva de los grupos no se define con anterioridad a la recolección de la información, ya que en la medida que se desarrolla el proceso de recolección, codificación y análisis, el investigador decide cuáles datos conservará para la próxima comparación y por otro lado en la consideración de los informantes finales se requiere tomar en cuenta criterios tales como: conocimiento y experiencia, habilidad para reflexionar, disponibilidad de tiempo así como disposición a participar en la investigación.

En la presente investigación, el criterio principal de inclusión para invitar a los informantes fue que trabajaran ya sea como programadores o en puestos gerenciales dedicados al desarrollo de software a la medida debido a que el modelo propuesto se diseñó para dicho tipo de software.

En relación a los programadores se tenía la intención de que contaran con 10 o más años de experiencia ya que se pretendía que fueran personas con amplia experiencia en el desarrollo de software, pero debido a la disponibilidad y/o disposición de las personas que cumplen con esta característica no fue posible considerarlos para el estudio, y en su lugar se tomó en cuenta la participación de programadores con un rango menor de años de experiencia, situación que al momento de realizar las entrevistas no afectó en absoluto la calidad y valía de las aportaciones de los informantes.

En cuanto a los informantes en puestos gerenciales se consideraron a personas que contaran con por lo menos 5 años de experiencia, esto debido a que en esos años ya es posible que hayan enfrentado suficientes problemas derivados de la crisis del software.

Respecto a su ubicación geográfica, ya que las entrevistas se pretendían realizar en persona con cada uno de los informantes, sólo se consideraron a los que laboran dentro del Distrito Federal y en cuanto al giro de la empresa no se estableció alguna restricción pero si fue tomando en cuenta que fueran compañías catalogadas como grandes y con una presencia relevante en su respectivo mercado, lamentablemente por motivos de confidencialidad no es posible proporcionar el nombre de los informantes ni de las empresas en las que laboran.

Como ya fue mencionado se invitó a 10 personas a realizar la evaluación del modelo, el perfil de los informantes con los que fue posible completar el proceso se muestra en la tabla 3.

Tabla 3. Perfil de los informantes.

	Puesto	Años de experiencia	Giro de la empresa
Informante 1	Diseñador y Desarrollador de Sistemas.	6	Banco
Informante 2	Desarrollador de sistemas (Consultor)	9	Consultoría de software.
Informante 3	Desarrollador de sistemas	8	Telecomunicaciones
Informante 4	Gerente de sistemas de facturación y mediación	14	Telecomunicaciones
Informante 5	Desarrollador de sistemas	4	Comercio
Informante 6	Coordinador de oficina de proyectos e inteligencia de negocios	23	Telecomunicaciones

Fuente: Elaboración propia.

De las 10 personas consideradas 9 aceptaron la invitación pero con 3 de ellas no fue posible llevar cabo la entrevista; por ello la Tabla 3 sólo muestra la información de 6 informantes. A continuación en la sección 3.8.4.3 se presentan los resultados más relevantes de la retroalimentación recibida por parte de los informantes por medio de las entrevistas.

3.8.4.3. Resultados de las entrevistas

Las entrevistas realizadas a los informantes permitieron identificar los siguientes aspectos:

- A pesar de que uno de los informantes mencionó cierta dificultad inicial para comprender los conceptos relacionados con la administración del conocimiento, a final de cuentas los seis comentaron haber entendido los conceptos y teorías presentados en el resumen que les fue enviado.
- Se presentó un consenso general de aceptación al planteamiento de que el software tiene conocimiento embebido, inclusive uno de los informantes mencionó que la relación entre el software y el conocimiento es natural pero no se le ha dado el valor y la formalidad necesaria acorde al entorno actual en el que se encuentran las organizaciones, por lo cual se requiere crear conciencia entre los empleados para que realicen sus actividades diarias bajo este esquema.
- En referencia a las directrices de administración del conocimiento aplicadas a la ingeniería de software, los informantes mencionaron haber identificado de manera directa los procesos del conocimiento, pero no reconocieron la conceptualización de que el software tiene conocimiento, ni a las características clave de la administración del conocimiento o a las escuelas del conocimiento como directrices del modelo.
- En cuanto a la administración del conocimiento como concepto, uno de los informantes recomendó presentar una definición propia bajo la cual se plantee el modelo, en lugar de sólo establecer una relación con las características clave de la administración del conocimiento.
- El planteamiento del modelo sólo se le dificultó a uno de los informantes ya que le resultó confuso distinguir la perspectiva integral. Otro informante no identificó las diferencias de la perspectiva integral respecto del modelo en espiral de ingeniería de software (presentado en la sección 2.4.3), los restantes cuatro informantes mencionaron que fueron claras y comprensibles cada una de las perspectivas y uno de esos cuatro sugirió reordenar la secuencia en las que se presentan de la siguiente manera: ingeniera de software, administración del conocimiento y al final la perspectiva integral.
- Fortalezas que los informantes consideran que tiene el modelo propuesto son:
    - Manejo de entregas incrementales por medio de los paquetes funcionales.
    - La base de conocimientos ayuda a invertir menos tiempo y esfuerzo en la planeación de proyectos futuros así como a detectar mejoras.
    - Es posible modularizar el software por áreas funcionales de negocio.

- Las personas involucradas en un proyecto de software estarían conscientes de todas las implicaciones que tiene debido a que se tiene una visión integral tanto de los procesos de ingeniería de software como de negocio.
- Perspectiva de que el software tiene conocimiento embebido.
- Empleo práctico de una base de conocimientos.
- Los empleados se pueden ir de vacaciones sin que sean buscados en el periodo que estén ausentes ya que el conocimiento esta en la base de conocimientos.
- Contacto directo con los usuarios.
- Facilita la contratación de outsourcing debido a que el conocimiento de negocio estaría codificado.
- Evita la monopolización del conocimiento dentro de las empresas.
- Eliminar la carga de trabajo de ciertas personas saturadas de solicitudes debido a que nadie más posee determinados conocimientos para realizar sus actividades.

➢ Debilidades del modelo propuesto detectadas por los informantes son:

- El manejo de la base de conocimientos implica más trabajo para las personas.
- Usos y costumbres de los empleados que derivan en la falta de disposición de compartir su conocimiento.
- Los desarrolladores de software no suelen tener la actitud de investigar las necesidades del negocio.
- Dificultad de trasladar el modelo a una metodología.
- Requiere un cambio de paradigma en la forma de trabajar de las empresas y por lo tanto hay que enfrentar la resistencia al cambio.
- Falta de disponibilidad de las personas para la interacción cara a cara.

➢ Las mejoras que los informantes sugirieron realizar la modelo son:

- Considerar un esquema para la definición y manejo de los requerimientos del usuario debido a que al ser la base del proceso de ingeniería de software representan un elemento al que se debe tratar con mayor atención.
- Tomar en cuenta estímulos para que las personas compartan el conocimiento que poseen.
- Incluir algún esquema para asegurar que todos los paquetes funcionales sean entregados.

➢ Sólo un informante consideró que el modelo no era aplicable en su entorno laboral debido a que argumentó que en su empresa se realiza la planeación a corto plazo y mientras se resuelva el problema no importa planear a futuro. Los restantes cinco informantes mencionaron considerar que el modelo es aplicable en sus empresas, siempre y cuando se cuente con apoyo de la alta dirección.

- Cinco informantes mencionaron estar dispuestos en trabajar bajo la estructura que plantea el modelo y la persona que expresó no estar dispuesta mencionó que no le gustaría realizar el trabajo extra que implica.
- Las recomendaciones en cuanto a la forma como se presentó el modelo son las siguientes:
  - Mostrar la información completa al presentar el modelo en lugar de hacer referencias a otros apartados.
  - Presentar un mapa conceptual en el que pueda apreciar la interacción entre las diversas perspectivas del modelo.
- Las sugerencias en referencia a la información contenida en el modelo son:
  - Proporcionar más detalles sobre el concepto y manejo de los paquetes funcionales.
  - Trasladar el modelo a una metodología.
  - Considerar que debido a que la ingeniería de software se desarrolla dentro del contexto de un proyecto y como tal hay elementos como el control de cambios o influencias externas que se requieren tomar en cuenta.
  - Sugerir indicadores de desempeño para medir la efectividad del modelo.

3.8.5. Ajustes al modelo

A partir de la retroalimentación obtenida en las entrevistas, los ajustes que se realizaron a la propuesta inicial del modelo son los siguientes:

- En la medida de lo posible no hacer referencia a otros apartados del texto y presentar la información completa al explicar el modelo, a pesar de que implique repetir ideas y conceptos correspondientes a otros Capítulos.
- Se cambió el planteamiento de los paquetes funcionales para tratar de presentarlo con mayor claridad.
- Plantear una definición de administración del conocimiento.
- Diferenciar la imagen del cubo donde se muestran las tres perspectivas respecto a la perspectiva integral lo cual incluye cambiar el nombre de “Perspectiva integral” a “Perspectiva de integración”.
- En la perspectiva de ingeniería de software incluir el proceso de ingeniería de requerimientos, sugerido por uno de los informantes, lo cual será realizado con base en el estándar de la IEEE.
- Debido a que en planteamiento general del modelo inicial sólo se usan tres caras del cubo, las restantes se emplean para enunciar elementos cuyos detalles quedan fuera del alcance del presente trabajo pero debido a la inquietud de los informantes se consideran para darles visibilidad dentro del modelo, estos elementos son: manejo de resistencia al cambio, manejo de resistencia a compartir el conocimiento y dirección de proyectos empleando las buenas prácticas establecidas por el *Project Management Institute* (PMI).

- Cambiar la secuencia de presentación de las perspectivas en este orden: ingeniera de software, administración del conocimiento, perspectiva de integración (anteriormente perspectiva integral) y posteriormente las nuevas perspectivas de manejo de resistencia al cambio, compartir conocimiento así como la dirección de proyectos.
- Presentar un mapa del modelo que permite apreciar la interrelación entre las diferentes perspectivas.

Las modificaciones anteriores fueron incorporadas al planteamiento inicial del modelo y el resultado se muestra en el siguiente Capítulo 4.

4. Modelo Final

La vida es un rompecabezas, cada meta alcanzada representa una pieza más que nos hace sentir mas completos.

Anónimo

Los sistemas de software desarrollados a la medida se han convertido en un elemento primordial en las operaciones diarias de las organizaciones. Tal y como lo mencionan Salem Ben y Mouna Ben (2009), a la fecha se han dedicado gran cantidad de trabajos orientados a los sistemas de información e ingeniería de software con el propósito de analizar los motivos de los retrasos en proyectos, presupuestos arriba de lo planeado, productos de software de baja calidad e inclusive con funcionalidad inadecuada. Desde los años 60's se ha empleado la expresión *crisis de software*, para hacer referencia a los problemas anteriores, realizándose varias propuestas de manera que sea posible aumentar la productividad y la calidad del software al nivel requerido por el entorno en el que se encuentran inmersas actualmente las organizaciones.

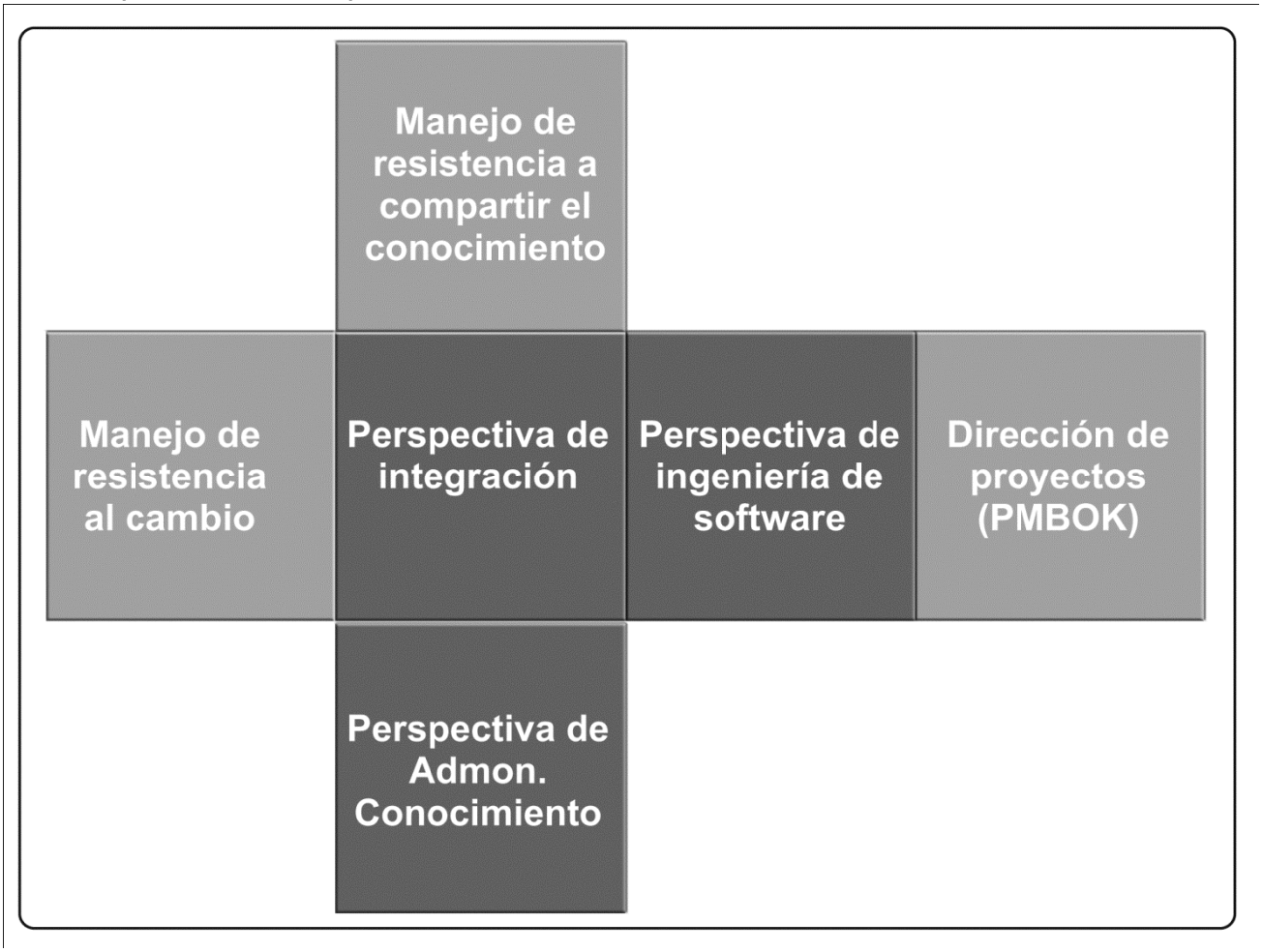
Las soluciones aplicadas hasta el momento están centradas en mejoras al ciclo de vida del software con base en herramientas de desarrollo, métodos o procesos, sin embargo, el alcance de estas soluciones suele ser limitado por enfocarse en problemas específicos y los beneficios que aportan no resultan significativos debido a que no se ha tomado en cuenta el hecho que el software es resultado de la acumulación del conocimiento poseído por las personas que forman las organizaciones, dejando de lado que la crisis del software se puede mitigar disminuyendo la brecha de conocimiento existente entre lo que saben las personas que forman una organización y el conocimiento contenido en el software, por lo cual se requiere un modelo que permita integrar de forma eficiente la mayor cantidad del *know-how* de las organizaciones, manejando el proceso de desarrollo de software desde una perspectiva de la administración del conocimiento permitiendo aprovechar el potencial del conocimiento poseído por las personas que forman las organizaciones e integrarlo al software.

A partir de los elementos desarrollados en los Capítulos anteriores los componentes que conforman la propuesta final del modelo de ingeniería de software con base en directrices de administración del conocimiento son: una perspectiva de ingeniería software, otra de administración del conocimiento y una más respecto a la integración de ambos elementos. Dichas perspectivas se complementan con otra de manejo de: resistencia al cambio y a compartir el conocimiento, y una para dirección de proyectos. Los detalles de estás últimas tres quedan fuera del alcance del presente trabajo, todas las perspectivas se pueden apreciar en la Figura 29.

Si consideramos que la Figura 29 representa la estructura de las 6 caras de un cubo, el cual al construirlo y observarlo desde dos ángulos distintos, podríamos apreciar lo que se presenta en la Figura 30, la cual constituye la perspectiva general del modelo propuesto.

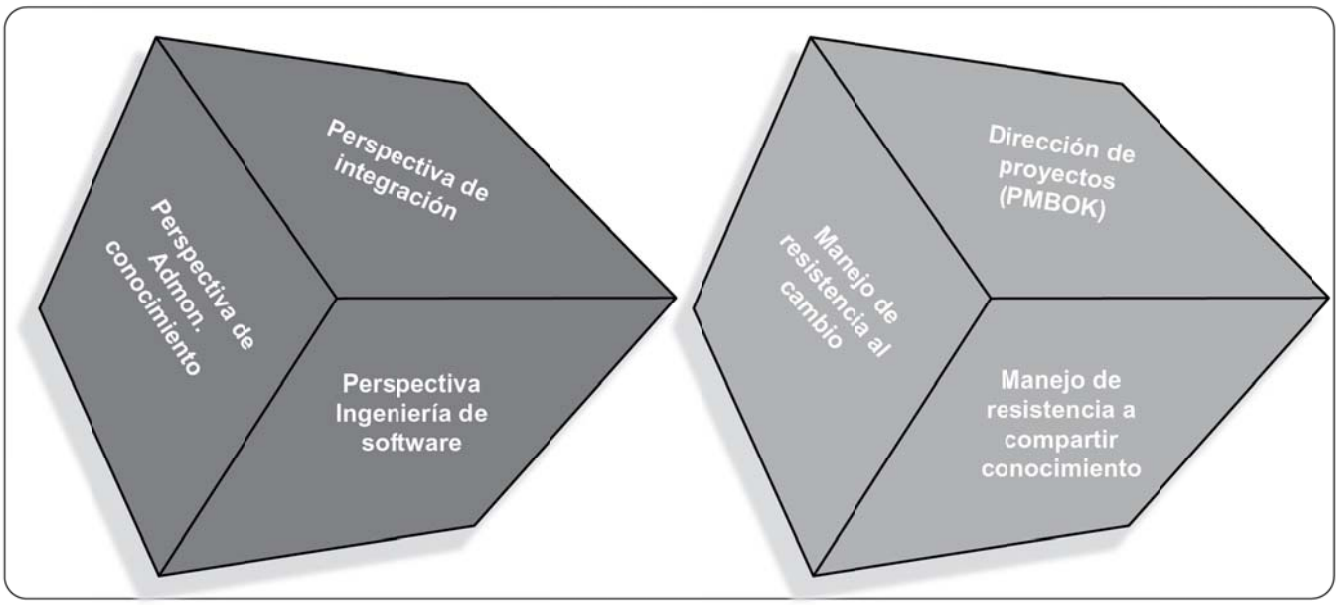
Fue elegido un cubo para representar la perspectiva general del modelo debido a que no por el hecho de que alguna de sus caras muestre algún color o diseño distinto de las demás, dejan de formar parte del mismo cubo, por lo tanto a partir de esa idea es sencillo abordar el modelo desde cualquiera de las perspectivas (caras del cubo).

Figura 29. Perspectivas consideradas para el modelo final.



Fuente: Elaboración propia.

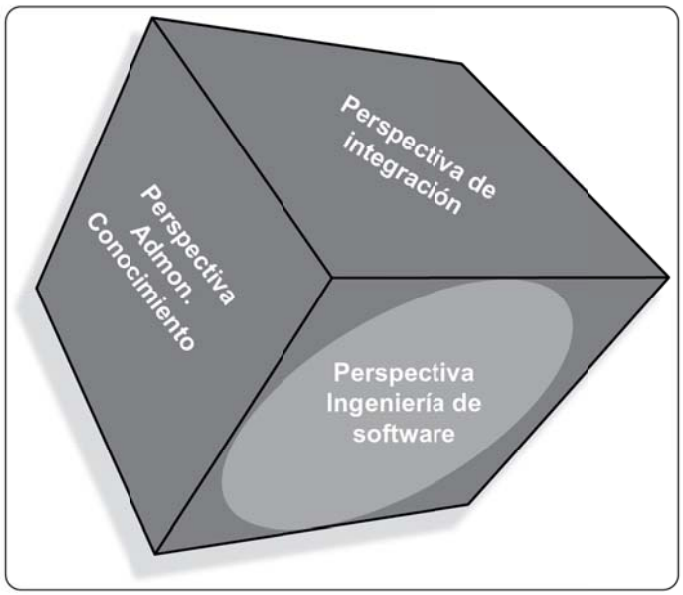
Figura 30. Perspectiva general del modelo propuesto.



Fuente: Elaboración propia.

4.1. Perspectiva de ingeniería de software

Figura 31. Perspectiva general del modelo propuesto (Ingeniería de Software).



Fuente: Elaboración propia.

Como se resalta en la Figura 31, la perspectiva abordada en esta sección corresponde a la cara de ingeniería de software, la estructura de este enfoque tiene como base la visión tradicional de las fases de ingeniería de software que son:

- Fase de requerimientos: Corresponde a las actividades que permiten especificar de la forma más detallada posible los requerimientos de los usuarios, de manera que las solicitudes no sean incompletas, ambiguas o contradictorias. Una vez que se cuentan con los elementos suficientes para realizar un análisis de los requerimientos se evalúan los mismos y se definen los que pueden ser alcanzables acorde a los recursos disponibles tanto técnicos como humanos.

- Fase de diseño: En esta etapa se diseña la solución que cubra el alcance de los requerimientos definidos en la fase anterior, la cual incluye estructurar una arquitectura para los sistemas de software o bien modificaciones a una existente, prototipos de las interfaces gráficas con las interactuará el usuario, mapeo de procesos, flujo de datos, diagramas e inclusive selección del lenguaje de programación.

- Fase de implementación: Consiste en traducir el diseño especificado anteriormente a un lenguaje de programación, de forma que se obtenga el producto de software que cumpla con la definición del alcance.

- Fase de pruebas: Una vez codificado el software se debe validar su adecuada funcionalidad, las pruebas se pueden dividir en las siguientes:

  - *Pruebas unitarias:* La complejidad de los sistemas de software actuales implica que sean implementados por varias personas, por lo cual cada una de ellas debe asegurarse que la porción de construcción de código que le fue asignado trabaje correctamente.

- *Pruebas integrales:* El conjunto de programadores que participaron en la implementación del software deben asegurarse que el código construido sea compatible y cumpla adecuadamente su función dentro de las interacciones requeridas en el sistema de software.
- *Pruebas en ambiente de quality assurance*: Los dos tipos de pruebas mencionadas anteriormente son ejecutadas en un ambiente creado para desarrollar software, donde los desarrolladores suelen tener el control del mismo por lo cual pueden presentarse situaciones donde el software no sea validado en algún escenario al que tenga que responder en condiciones de uso real. Por este motivo, es requerido contar con un ambiente creado para ejecutar pruebas de software, lo cual hace necesario migrar los componentes construidos por los programadores a este ambiente, y representa un ensayo del procedimiento a seguir cuando se traslade a un ambiente operativo. En este ambiente se emula comportamiento operativo del nuevo software para garantizar que el sistema se comporte adecuadamente.

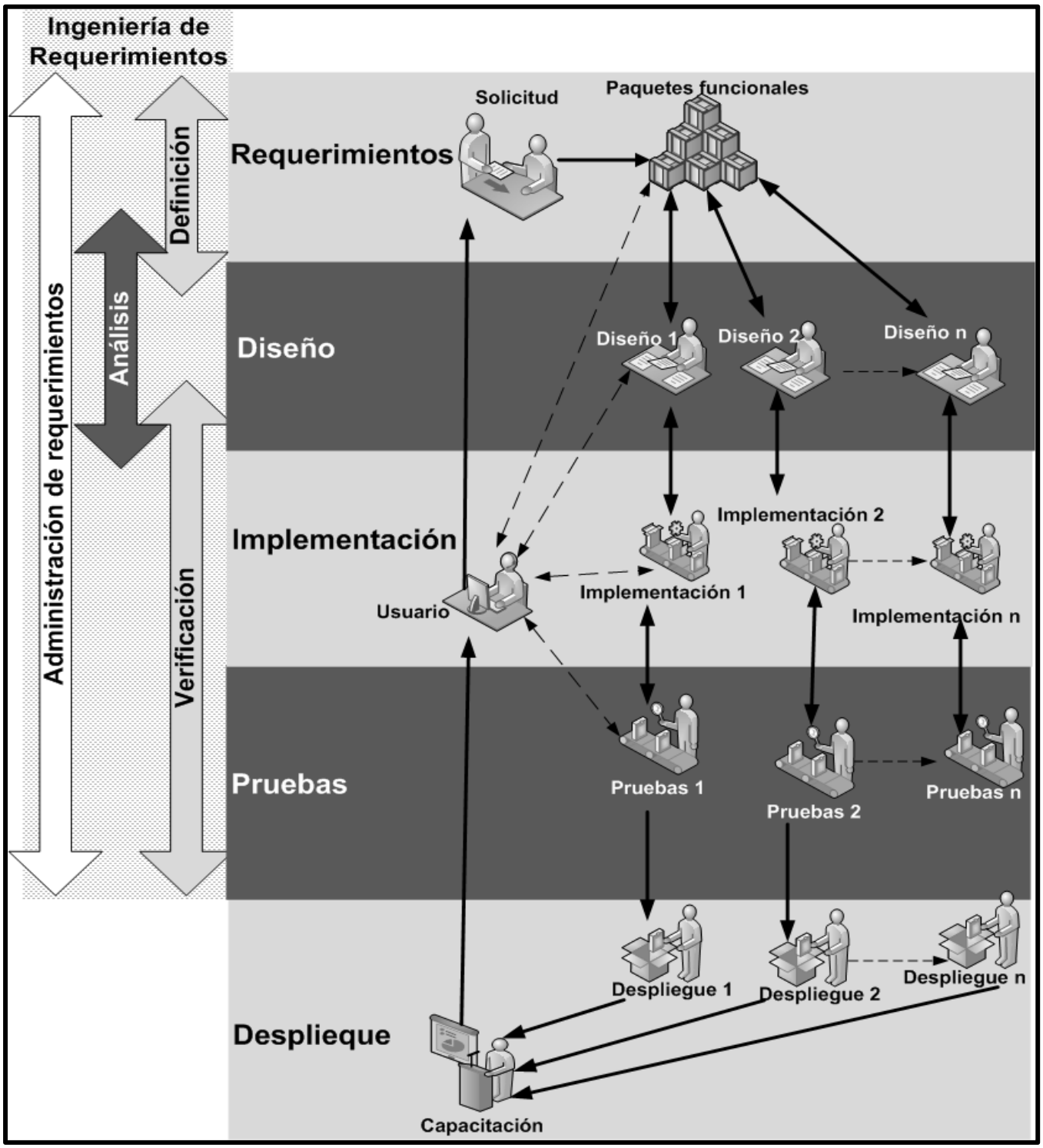
➢ Fase de liberación o despliegue: En caso de que sea detectada alguna falla en la fase de pruebas se debe volver a la fase de implementación para corregirla y llevar a cabo una vez mas las validaciones necesarias hasta que sea posible certificar que el nuevo software cumple con el alcance definido y entonces se procederá a migrarlo al ambiente productivo para que los usuarios tengan acceso al mismo.

➢ Fase de mantenimiento: Una vez que el software es colocado en ambiente operativo puede darse el caso de que se presenten errores no detectados en la fase de pruebas o bien que se deban realizar modificaciones para atender nuevos requerimientos en cuyo caso se repetirían las fases tantas veces como sea solicitado por los usuarios.

La Figura 32 muestra de manera gráfica el manejo que debe hacerse a las fases de ingeniería de software de acuerdo al modelo que se esta proponiendo, en lo que se refiere a la fase de mantenimiento no es presentada de manera explicita ya que las actividades para llevarla a cabo implican un nuevo ciclo de las fases que son consideradas en dicha Figura. Un elemento implícito que se debe tomar en cuenta en esta perspectiva es validar los procesos de administración del conocimiento implicados en cada una de las fases acorde con las reglas y condiciones especificadas en la perspectiva de integración (sección 4.3), de forma tal que se cuente con el conocimiento necesario en cada una de las fases para obtener los entregables requeridos o bien realizar la codificación de nuevos conocimientos y documentar la relación entre estos así como los existentes con los entregables.

Cualquier interacción entre personas se debe realizar cara a cara, o en su defecto con el medio más directo que sea posible (video conferencia, conversación telefónica, etc.) y simplemente se deben formalizar los acuerdos, responsabilidades futuras y/o hallazgos mediante una minuta o un correo electrónico breve, que debe almacenarse en un repositorio de documentación destinado a cada solicitud.

Figura 32. Perspectiva de ingeniería de software (modelo final).



Fuente: Elaboración propia.

De forma adicional a los aspectos mencionados de las fases de la ingeniería de software respecto al enfoque tradicional, las particularidades de cada una, bajo el modelo propuesto son las siguientes:

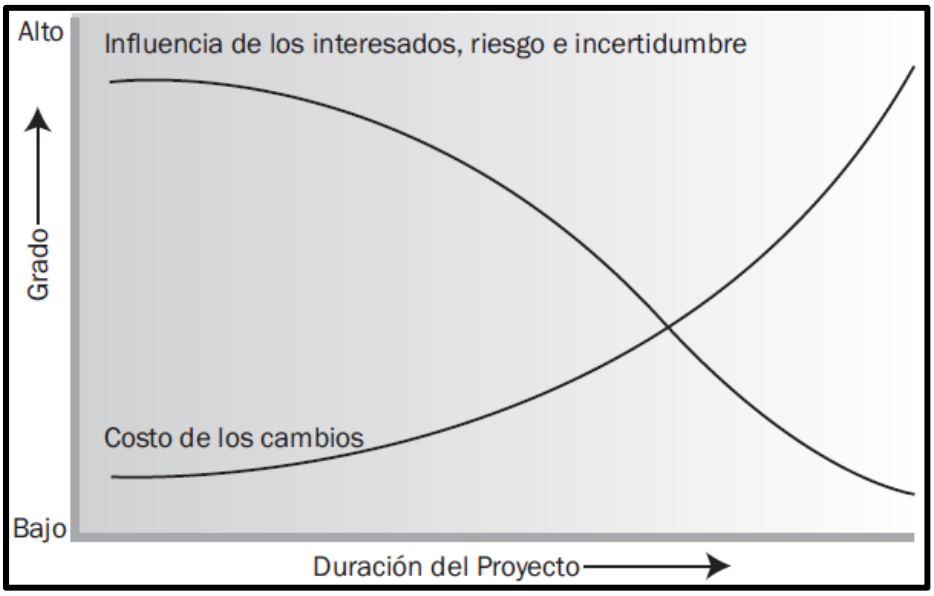
- *Fase de requerimientos:* El equipo de trabajo establecerá “paquetes funcionales”, cada uno de los cuales consiste en un conjunto lo más pequeño posible de requerimientos agrupados de manera tal que cada conjunto (paquete funcional) pueda ser implementado independientemente uno de otro. La intención de manejar estos paquetes es que en el tiempo más corto posible el usuario pueda tener disponible la funcionalidad que considere más importante. Para tal efecto, una vez que se hayan definido los paquetes es necesario consultar al usuario para que establezca una priorización de los mismos acorde a sus necesidades y se defina el orden en que será implementado cada uno, lo que permitirá realizar entregas incrementales de funcionalidad o bien desarrollarlos simultáneamente en caso de que la organización cuente con recursos suficientes aplicar el modelo de Desarrollo Rápido de Aplicaciones (explicado en la sección 2.4.4). Si en un momento dado, no es posible segmentar la entrega de la solución se tendrá un sólo paquete funcional compuesto por la totalidad de requerimientos solicitados. Como se aprecia en la Figura 32, para lograr tener una correcta definición y seguimiento de los requerimientos estas actividades se ejecutarán mediante los procesos comprendidos por la ingeniería de requerimientos cuyos detalles se explican más adelante en la sección 4.1.1.

- *Fases de diseño, implementación y pruebas:* Al final de cada una de estas fases se debe revisar con el usuario los entregables para certificar la inclusión de los elementos necesarios para que el software que está siendo producido en cada paquete funcional cumpla sus expectativas y mantenga la prioridad adecuada para continuar con el proceso. Y en caso de que no se alcancen las expectativas del usuario, se debe replantear la implementación, el diseño o la composición de los paquetes funcionales junto con su prioridad para satisfacer las expectativas del usuario. Es muy importante que lo anterior se maneje plasmando de manera clara los impactos en costo, tiempo y otros recursos que tendrán apoyándose en los elementos de la perspectiva de dirección de proyectos, pero de ninguna manera cerrándose a los cambios, ya que si bien, (como se acaba de mencionar) se presentará afectación respecto a los planes originales, es menos costoso realizar los ajustes necesarios lo más pronto posible (como lo indica la Figura 33).Los entregables que se presenten al usuario deben ser aquellos que no tengan implicaciones técnicas y le sean entendibles de manera que pueda realizar la respectiva validación.

- Fase de despliegue: Cada vez que el software correspondiente a un paquete funcional sea desplegado en ambientes productivos, donde el usuario tenga acceso al mismo para sus actividades diarias. Además de los manuales de usuario que indiquen los detalles del uso del software se requiere llevar a cabo reuniones de capacitación para explicarle el manejo adecuado del software y dejar abierto algún canal de comunicación para resolver dudas.

En caso de que sea posible segmentar los requerimientos en paquetes funcionales implícitamente se ejecutará un *mantenimiento activo* del software ya que no es necesario esperar a que el alcance de la solicitud se complete al 100% para detectar puntos de mejora o errores, debido a que el usuario obtendrá entregas incrementales del software, situación que le permitirá trabajar con el mismo en condiciones de operación real que ayuden a identificar nuevas necesidades o ratificar las definidas previamente en su requerimiento.

Figura 33. Impacto del Costo de cambios acorde al tiempo en el que son aplicados.



Fuente: Project Management Institute. (2008). *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)*. Pennsylvania, USA: Project Management Institute.

4.1.1. Ingeniería de requerimientos

IEEE (2011) menciona que la ingeniería de requerimientos es una función interdisciplinaria, que permite mediar entre los intereses de los solicitantes y proveedores para establecer y gestionar los requerimientos que debe lograr un sistema, software o servicio. La ingeniería de requerimientos se ocupa de descubrir, obtener, desarrollar, analizar, determinar métodos de verificación, validar, comunicar, documentar y manejar las exigencias directivas relacionadas con los requerimientos, teniendo como resultado una jerarquía que:

- permite establecer un acuerdo entre las personas involucradas en una solicitud.
- es validada contra necesidades reales cuya solución sea posible implementar, y
- proporciona una base para verificar diseños y establecer criterios de aceptación de las soluciones.

Los procesos implicados en la ingeniería de requerimientos acordes con IEEE (2011) son:

- *La Definición de los requerimientos:* que tiene la finalidad de establecer los requerimientos para que el sistema sea capaz de proporcionar los servicios necesitados por los usuarios. Se identifican las personas (involucrados) que serán afectadas por los requerimientos, definiendo sus necesidades expectativas y deseos, que son trasformadas en un conjunto de requerimientos que expresen la intención de las funciones que debe cumplir el sistema en un ambiente operativo y contra los cuales será validado. Dentro

del modelo presentado en la Figura 32, se plantea que este proceso debe ser ejecutado en la fase de requerimientos de la ingeniería de software así como al inicio de la fase de diseño, en la cual se suelen definir algunos requerimientos técnicos para solventar la solución.

- *Los Análisis de los requerimientos*: transforman los requerimientos establecidos desde el punto de vista de los usuarios a una visión de los elementos técnicos implicados para solventarlos. En este proceso se construye una representación del sistema que resolverá los requerimientos teniendo como resultado que pueda ser posible establecer métricas para validar los requerimientos así como las características que debe cumplir el sistema para satisfacer los requerimientos. Este proceso, tal y como es mostrado en la Figura 32, comienza en la fase de requerimientos para que se establezcan las métricas que permitan validarlos, posteriormente estará presente en la fase de diseño para realizar la transformación de los requerimientos de negocio a una solución técnica y finalmente en la fase de implementación se lleva a cabo un análisis que lleve a la correcta construcción de las soluciones.
- *La Verificación de los requerimientos*: que se encarga de confirmar que los requerimientos sean complemente resueltos por el sistema. En la Figura 32 se puede apreciar que este proceso tiene presencia desde la fase de diseño para certificar que la solución conceptual cumpla con los elementos necesarios para resolver el problema, posteriormente en las fases de construcción y pruebas se debe hacer el control de que el software implementado cumpla con los criterios establecidos en la definición de requerimientos y por lo tanto los artefactos de software puedan ser entregados a los usuarios en los ambientes operativos.

IEEE (2011) menciona que la gestión de los procesos anteriores es realizada por medio de la administración de requerimientos, la cual es vigente en cualquiera de las fases de ingeniería de software en la que tenga presencia cualquiera de los tres procesos de ingeniería de requerimientos como se muestra en la Figura 32.

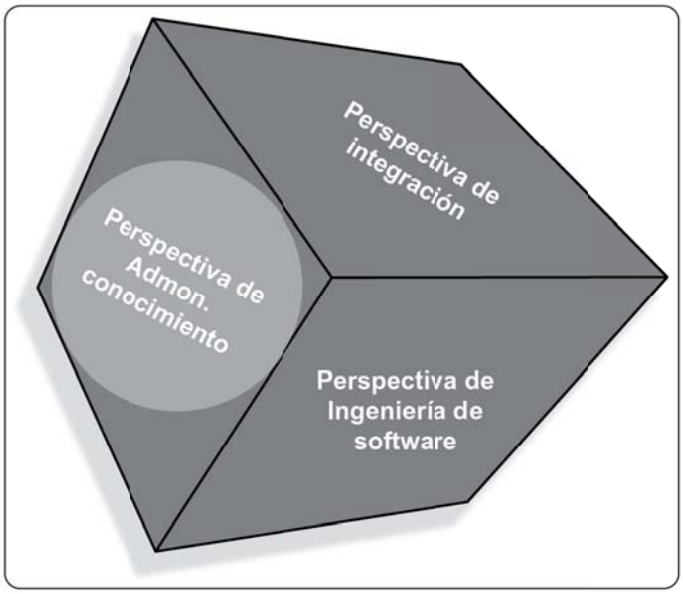
Según lo expuesto por IEEE (2011), las características que debe cumplir un requerimiento son:

- verificabilidad.
- que esté solventado por algún sistema que permita resolver el problema expuesto por el usuario.
- que sea cuantificable y medible.
- restricciones de operación.
- rendimiento del sistema cuando se ejecute la funcionalidad asociada.
- que sea necesario (o que satisfaga una necesidad).
- claridad
- consistencia
- integridad.

- que sea alcanzable y rastreable.

4.2. Perspectiva de administración del conocimiento

Figura 34. Perspectiva general del modelo propuesto (Administración del Conocimiento).



Fuente: Elaboración propia.

Como lo indica la Figura 34 este apartado está enfocado a los detalles que corresponden con la perspectiva de administración del conocimiento, presentada de manera gráfica en la Figura 35, los procesos del conocimiento que en ella se muestran se manejan acorde con lo expuesto por Valhondo (2003), quien indica que la serie de procesos/metaprocesos tienen una interrelación espacial y temporal que no está dominada por alguno en particular. Es decir, la interdependencia entre los procesos es múltiple y cruzada. Las características generales de cada proceso son las siguientes:

- *Descubrir*: Este proceso implica identificar las fuentes de conocimiento para las organizaciones, tanto internas como externas, así como establecer las herramientas necesarias para que sea extraído. Las fuentes van desde bases de datos, documentos y procesos organizacionales hasta las personas que forman la empresa por medio del conocimiento tácito.

- *Capturar:* Una vez localizado el conocimiento es preciso evaluar su utilidad y determinar de qué clase de conocimiento se trata, ya que estos puntos determinarán la viabilidad y estrategia de captura para alimentar: bases de datos, directorios de expertos, repositorio de conocimientos, data warehouse o procesos.

- *Clasificar y almacenar:* La clasificación es un proceso interpretativo influido por los criterios de la persona que clasifica, por lo cual es imperante que las organizaciones especifiquen reglas bajo las cuales será clasificado el conocimiento antes de almacenarlo, y es precisamente esta clasificación que distingue la captura del almacenamiento ya que la captura solo implica codificar para que en el momento de que sea clasificado se ubique donde aporte valor para la organización.

- *Distribuir / Diseminar:* Hoy en día se requiere lidiar con la sobrecarga de información y si no es manejada de manera adecuada puede resultar un problema que cause los mismos efectos negativos que la carencia de la misma. Es por ello que se debe establecer si el conocimiento será "arrojado" hacia las personas o simplemente se colocará en alguna herramienta tecnológica para que esté disponible en el momento que se necesite.

- *Compartir / Colaborar:* El conocimiento se conforma de información estructurada en combinación con la experiencia de las personas, para ello se debe fomentar que por iniciativa propia los empleados de una organización codifiquen esta experiencia para que este disponible para otros, además de colaborar con la socialización, externalización, internalización y combinación de conocimiento mediante la interacción adecuada con las demás personas.

- *Utilizar:* El objetivo de los procesos del conocimiento es que sea utilizado para conducir a innovaciones que permitan crecer a las organizaciones diferenciando sus productos y/o servicios agregando valor para sus clientes.

La administración del conocimiento consiste en atender los procesos anteriores para asegurar que el flujo de información y conocimiento llegue hasta las personas que pueden crear valor para las empresas con la finalidad principal de aportar ventajas competitivas a las organizaciones y aumentar el éxito en los negocios.

Algunas herramientas que pueden dar soporte a la administración del conocimiento y permiten establecer una interconexión entre los procesos y las personas, como se muestra en la Figura 35, son las siguientes:

- *Fuentes externas de conocimiento:* En caso de que dentro de la empresa no se identifiquen recursos que permitan obtener el conocimiento para cumplir el alcance definido para una solicitud de desarrollo de software se puede recurrir a entes externos a la empresa como: fuentes en internet, organizaciones y revistas especializadas, instituciones gubernamentales, empresas de consultoría, etc.

- *Base de conocimientos:* Constituye el activo más preciado en el que convergen las tres perspectivas principales del modelo, porque dentro de esta base se encontrará codificado el conocimiento poseído por las personas que forman parte de las organizaciones y será el instrumento que permita establecer la relaciones entre el software y el conocimiento organizacional en ambos sentidos. Dentro de los documentos que contenga la base de conocimientos se deben especificar los entregables de cada fase de la ingeniería de software (incluyendo los artefactos de software) que usan o implementan determinados conocimientos. En los entregables se debe considerar un apartado para indicar los conocimientos empleados mediante algún tipo de clave que permita acceder a ellos por medio de la base de forma sencilla. La base de conocimientos se puede estructurar de varias formas, puede ser: carpetas compartidas en alguna computadora dentro de la empresa a la que todos los empleados tengan acceso, alguna aplicación de software para el manejo de repositorios de archivos o bien software construido explícitamente para el manejo de bases de conocimientos.

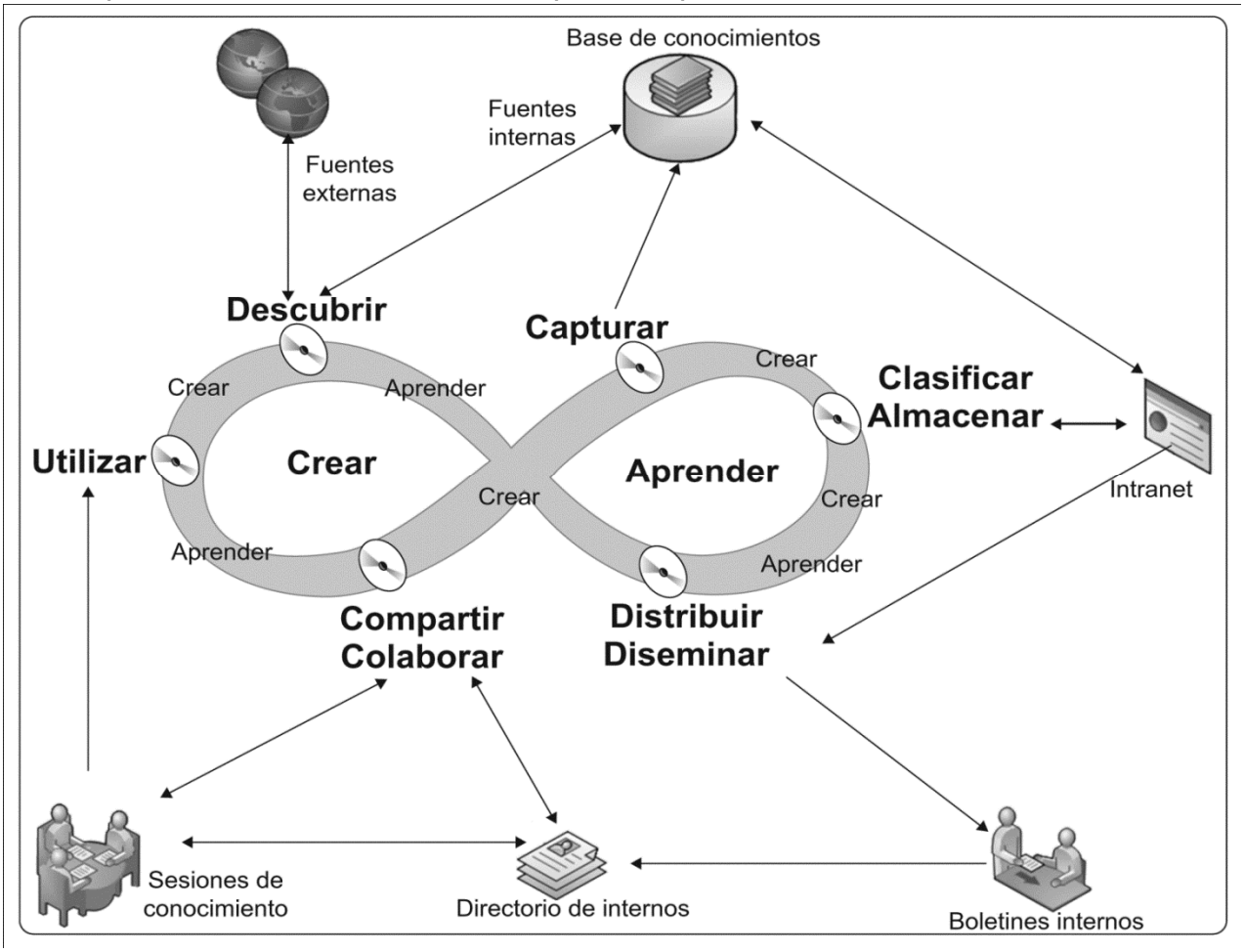
➢ *Intranet:* Pueden emplearse como medio para manejar la clasificación y almacenamiento de los componentes contenidos en la base de conocimientos, de manera que sea fácil localizarlos y represente un punto de partida para su distribución. También pueden emplearse como filtro de acceso para conocimientos sensitivos de las organizaciones.

➢ *Boletines internos:* Ayudan a difundir periódicamente nuevos conocimientos relevantes de la operación diaria de las organizaciones, y a inducir una cultura que fomente en los empleados compartir el conocimiento personal que han adquirido a través de la experiencia.

➢ *Directorio de expertos:* Resulta complicado que, en etapas iniciales de implantación de proyectos de administración del conocimiento, las bases donde se almacenen los conocimientos puedan contener todos los detalles requeridos o no se encuentren codificados. Es por ello que mediante un directorio de los expertos de las áreas de negocio en la empresa puede hacer posible que se establezcan enlaces entre las personas que tienen preguntas con las que poseen las respuestas.

➢ *Sesiones de conocimiento:* Son reuniones cara a cara, o en su defecto mediante el canal de comunicación más directo que se pueda establecer, para que los expertos en un tema apoyen a personas que requieran el conocimiento que poseen y no se encuentre codificado dentro de la base.

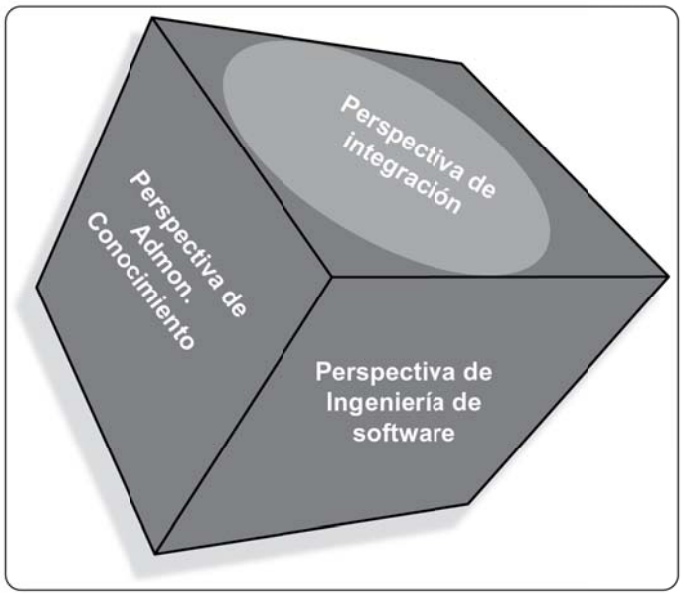
Figura 35. Perspectiva de administración del conocimiento (modelo final).



Fuente: Elaboración propia.

4.3. Perspectiva de integración

Figura 36. Perspectiva general del modelo propuesto (Integración).



Fuente: Elaboración propia.

Como es señalado en el cubo de la Figura 36 se abordará la perspectiva que integra los dos elementos clave del presente trabajo: la ingeniería de software y la administración del conocimiento. El enlace de estos enfoques se fundamenta con la consideración de que el software además de poder definirse como programas de computadora, procedimientos y la posible documentación asociada a los mismos así como los datos que pertenecen a la operación de un sistema informático acorde a lo indicado por IEEE (1990), también representa la acumulación del conocimiento organizacional, es decir el know-how como lo plantean Salem Ben y Mouna Ben (2009) ya que contiene conocimiento embebido. Por lo tanto, al reconocer formalmente que el software contiene y requiere del conocimiento que posee la organización y las personas que la forman, en las fases de ingeniería de software es imperante involucrar los procesos de la administración del conocimiento. Esta integración es presentada de manera gráfica en la Figura 37 en la cual las fases de ingeniería de software se representan por los círculos concéntricos y los procesos del conocimiento son las secciones delimitadas mediante flechas en cada uno de los círculos, de tal forma, que en las fases de ingeniería de software se ejecuten formalmente los procesos del conocimiento que sean requeridos, es decir no se deben realizar los procesos del conocimiento de manera cíclica para pasar a la siguiente fase, sino simplemente aquellos que ayuden a obtener de manera exitosa los entregables definidos o bien los que se requieran para almacenar los conocimientos generados en la base de conocimientos establecida en la perspectiva de administración del conocimiento del modelo (detallada anteriormente en la sección 4.2).

La formalización del uso de los procesos del conocimiento en cada una de las fases de ingeniería de software queda estructurada de la siguiente manera:

- *Utilizar:* Este proceso estará presente en todas las fases ya que para obtener los entregables que sean definidos para cada una, no hay otra manera más que utilizando el conocimiento que poseen las personas, el generado en cualquiera de las fases o el almacenado en la base de conocimientos, todos los entregables deben contener la referencia de los elementos empleados en la base de conocimientos y a su vez en la base se requiere listar los entregables que han empleado los conocimientos que contiene, por lo tanto si algún conocimiento no está registrado en la base es necesario capturarlo para cumplir con lo anterior.

- *Capturar:* La ejecución de este proceso sucederá en cualquiera de las fases en las que se descubran y/o creen nuevos conocimientos.

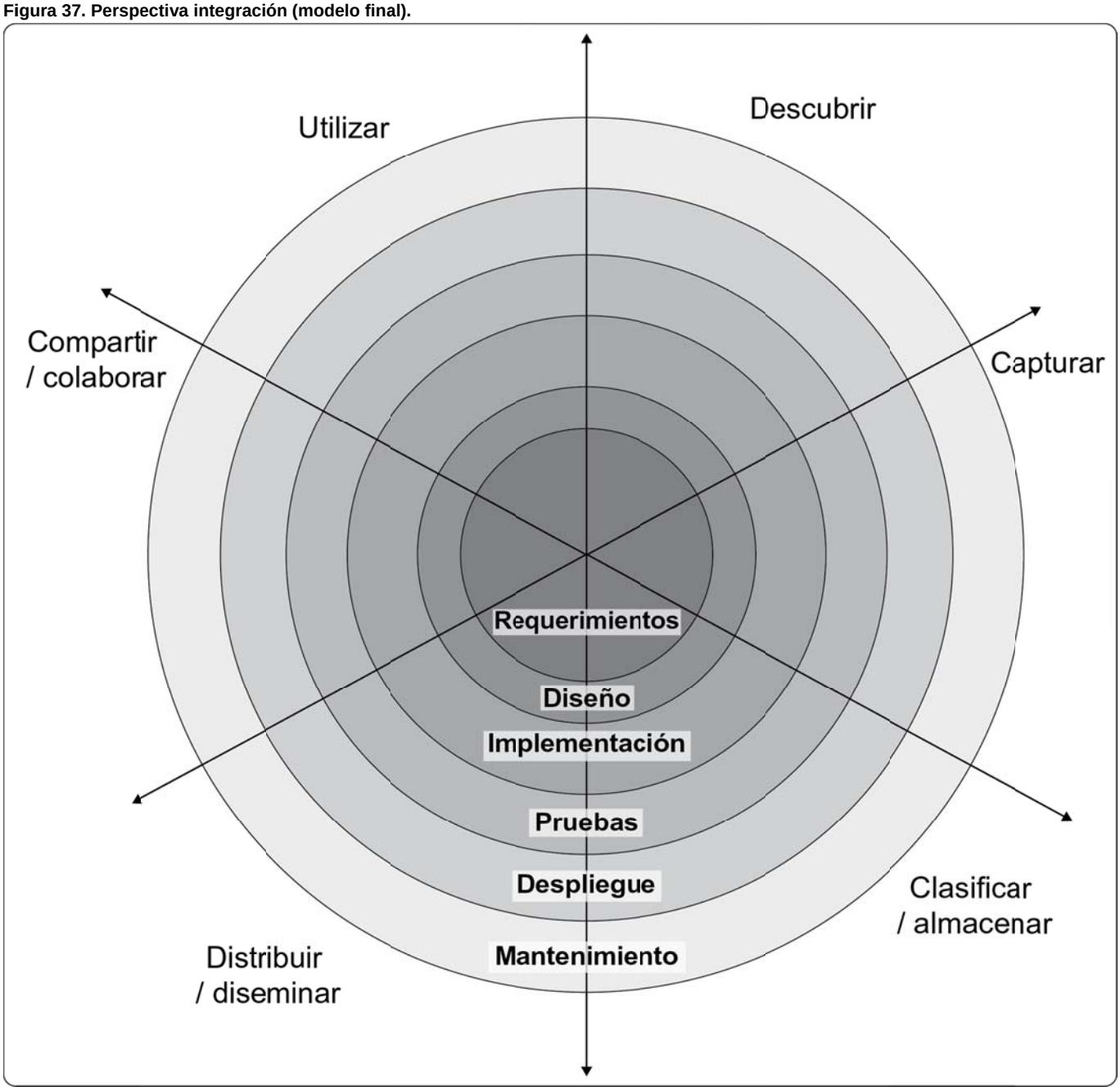
- *Clasificar/Almacenar, Distribuir/Diseminar:* Se deben ejecutar estos procesos de manera conjunta en las fases en las que se capturen conocimientos nuevos en la base.

- *Compartir/Colaborar:* Si en una fase determinada se descubre que el conocimiento necesario para obtener los entregables sólo está disponible por medio de otras personas dentro empresa es cuando se deben realizar estos procesos.

- *Descubrir:* La presencia de este proceso dependerá de las siguientes condiciones para cada fase:
  - Fase de requerimientos: En caso de no se cuente con el conocimiento necesario para evaluar y analizar la factibilidad de los requerimientos.
  - Fase de diseño: Si no se tienen los conocimientos para realizar el diseño del software.
  - Fase de implementación: Cuando los programadores no cuenten con el conocimiento requerido para desarrollar el software.
  - Fase de pruebas: Cuando las personas encargadas de ejecutar las pruebas no conozcan las condiciones del comportamiento correcto del software.
  - Fase de despliegue: En caso de que los responsables de migrar el software a un ambiente productivo no sepan el procedimiento para completar su tarea.
  - Fase de mantenimiento: Para saber qué conocimientos ya están contenidos en el software y valorar los ajustes que se requieren.

Con la intención de que no se presenten confusiones es prudente recordar que con la palabra “conocimiento” en la lista anterior, se hace referencia al conocimiento de negocio o procesos internos de la empresa, no a cuestiones técnicas relacionadas con la construcción del software.

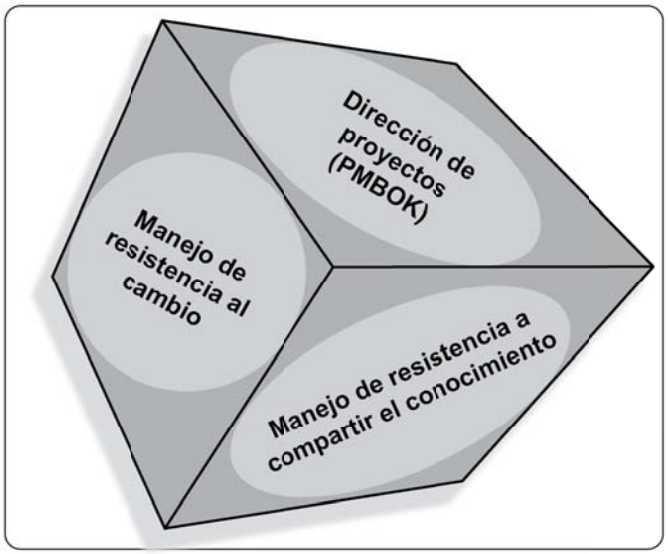
Figura 37. Perspectiva integración (modelo final).



Fuente: Elaboración propia.

4.4. Perspectivas complementarias

Figura 38. Perspectiva general del modelo propuesto (Complementarias).



Fuente: Elaboración propia.

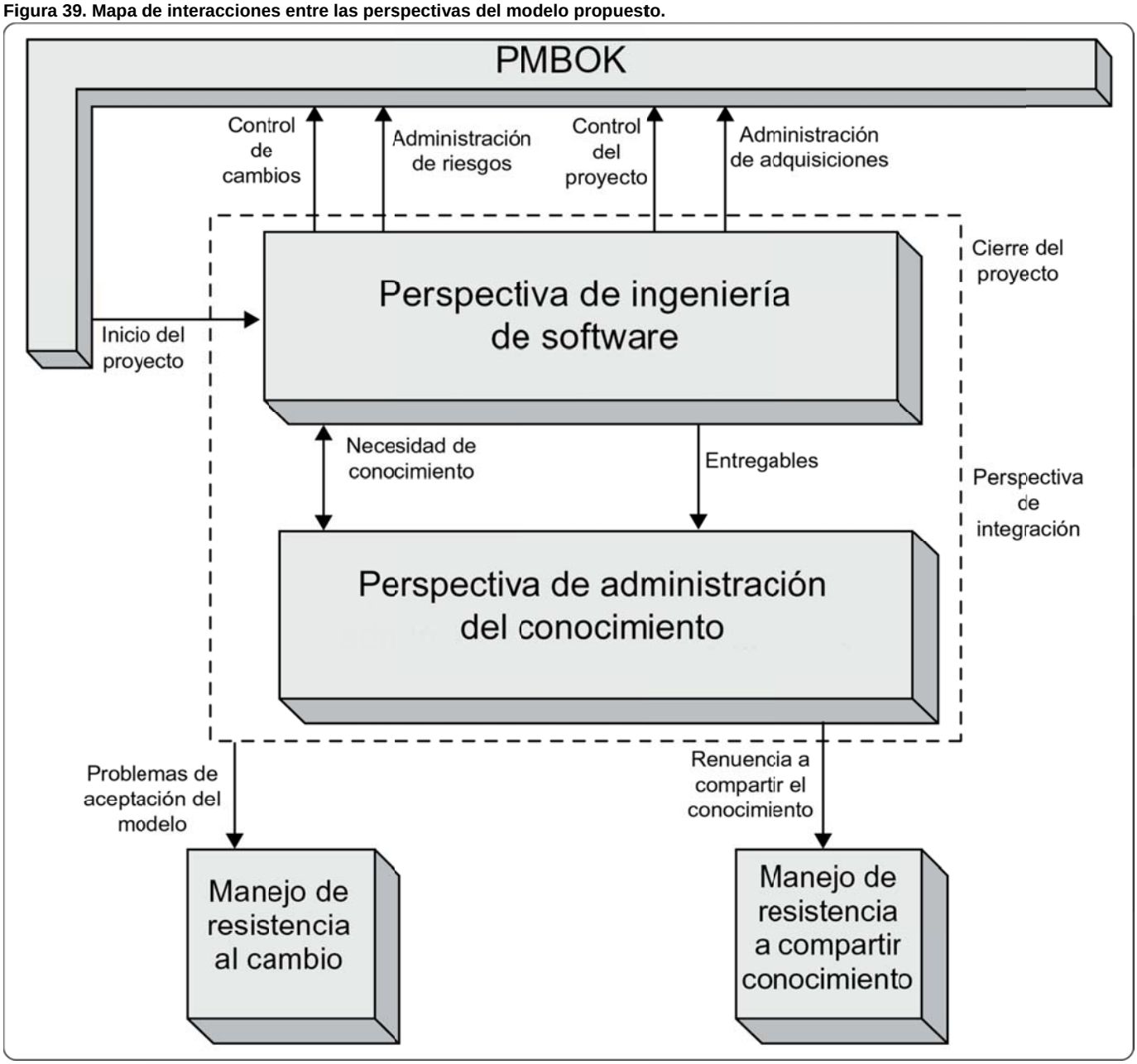
La inclusión de las perspectivas mostradas en la Figura 38 surgió a partir de las inquietudes expresadas por las personas (informantes) que realizaron la evaluación de la propuesta inicial del modelo (para más detalles consultar las secciones 3.8.4 y 3.8.5). A diferencia de las 3 perspectivas que se explicaron anteriormente, para las complementarias no se presentan detalle, debido a que su perfil queda fuera del alcance y los objetivos del presente trabajo, pero se incluye la referencia a las mismas para cubrir algunos "huecos" identificados en las perspectivas principales (ingeniería de software, administración del conocimiento e integración), esto debido a que por un lado la ingeniería de software es un proceso que se ejecuta como una reacción de las organizaciones a necesidades dictadas por el entorno en el que interactúan y el desarrollo de software a la medida constituye por si mismo un proyecto el cual hay ocasiones que forma parte de un programa o portafolio de proyectos con fines de mejorar y agregar valor a los productos y servicios ofertados por las empresas. Por este motivo es que se requiere tener una referencia para el manejo de situaciones relacionadas con la dirección de proyectos que pueden influenciar a la ingeniería de software, por ejemplo la gestión de los cambios de los requerimientos en cualquiera de las sus fases. Y, por otro lado, los informantes manifestaron recurrentemente la necesidad de trasladar el modelo a una metodología aspecto sobre el que fue recurrente el comentario de que en el mundo del desarrollo de software: "no hay que intentar reinventar la rueda", por lo tanto como metodología para la implementar el modelo se puede emplear sin problema alguno, las buenas practicas establecidas por el *Project Management Institute(PMI)* en su guía de *Project Management Body of Knowledge(PMBOK)* el cual es un estándar ampliamente aceptado a nivel internacional.

Adicionalmente, el trabajar con un enfoque orientado hacia la administración del conocimiento representa un cambio de paradigma que puede causar conflictos entre las personas que forman las organizaciones derivando en resistencia tanto al cambio en la

forma de trabajar como a compartir el conocimiento. Es por ello que para aplicar este el modelo se requiere el apoyo de altos directivos y concientizar a las personas en mandos medios para que se pueda permear a niveles inferiores la nueva forma de trabajar más por convicción que por imposición, para lo cual hay que aplicar técnicas y habilidades acordes con la cultura organizacional de cada empresa.

4.5. Mapa del modelo

Figura 39. Mapa de interacciones entre las perspectivas del modelo propuesto.



Fuente: Elaboración propia.

La Figura 39 permite apreciar el mapa de las interacciones entre las 6 perspectivas que conforman el modelo, las cuestiones propias de la gestión del proyecto asociado a las fases contenidas en la ingeniería de software serán canalizadas a la perspectiva del

PMBOK así mismo ésta indicará el inicio de las actividades de la ingeniería de software una vez autorizado el inicio del proyecto y oficializará el cierre del mismo en el momento que se hayan entregado la totalidad de los paquetes funcionales definidos en la fase de requerimientos. En cualquier instante en el que sean requeridos conocimientos para completar alguna de las fases de ingeniería de software será necesario trasladarse a la perspectiva de administración del conocimiento para obtenerlo. Los entregables de cada fase deben registrarse en la base de conocimientos por lo cual también en tales casos se crea interacción con la administración del conocimiento. Ahora bien, estas interacciones están perfectamente establecidas en la perspectiva de integración de la ingeniería de software con la administración del conocimiento, y si se llega a dar el caso que en la perspectiva integral se presenten problemas de aceptación del modelo se requiere acudir a la perspectiva que permite realizar el manejo de la resistencia al cambio. Finalmente, cuando en alguno de los componentes de la administración del conocimiento se presente el hecho de que las personas se nieguen a compartir lo que saben, es necesario establecer el enlace con el manejo de la renuencia a compartir el conocimiento.

Conclusiones

La vida es el arte de sacar conclusiones suficientes a partir de datos insuficientes.

Samuel Butler

En relación a las preguntas de investigación planteadas para este trabajo la respuesta a cada interrogante queda de la siguiente manera:

a) ¿Cuáles son las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento?

   Respuesta: Requerimientos, diseño, implementación, pruebas, despliegue y mantenimiento.

b) ¿Qué actividades se deben ejecutar en el proceso de ingeniería de software para integrar eficientemente conocimiento de los empleados en el mismo?

   Respuesta: Los procesos del conocimiento contenidos en el modelo integrado de procesos: descubrir, capturar, clasificar/almacenar, distribuir/diseminar, compartir/colaborar y utilizar.

c) ¿Es posible considerar redundancia de información para integrar el conocimiento al software?

   Respuesta: No se encontró evidencia de que mediante la redundancia de información se obtuvieran beneficios para integrar el conocimiento en el software.

d) ¿Mediante qué herramientas es posible almacenar el conocimiento tácito de los desarrolladores de software?

   Respuesta: Con una base de conocimientos para permitir establecer la relación explícita entre los artefactos de software y los conocimientos que contiene.

En cuanto a la respuesta de la pregunta general de la investigación planteada: ¿Qué elementos debe tener un modelo de ingeniería de software con base en directrices de administración del conocimiento?; la resolución es: la perspectiva de ingeniería de software, la perspectiva administración del conocimiento y la perspectiva integración.

Las respuestas anteriores junto con el modelo final presentado en el Capítulo 4 permiten establecer que se logró cumplir completamente con el objetivo general de la investigación: Diseñar un modelo para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento. De la misma forma se logró cumplir íntegramente cada uno de los objetivos específicos:

a) Definir las etapas clave en el proceso de ingeniería de software en las que se debe integrar la administración del conocimiento.
b) Establecer el conjunto de actividades en el proceso de ingeniería de software para integrar eficientemente el conocimiento de los empleados en el mismo.
c) Evaluar la viabilidad de emplear redundancia de información para integrar el conocimiento en el software.
d) Identificar las herramientas que permitan almacenar conocimiento tácito.

Por otro lado, la evaluación del modelo mediante entrevistas cualitativas representó un proceso que le agregó valor de una manera evidente que se puede apreciar al comparar la propuesta inicial del Capitulo 3 con la final presentada en el Capítulo 4. Entre las bondades de

haber empleado esta herramienta resalta que, como lo mencionan Bolsegui y Fuguet (2006), se considera al ser humano como el instrumento principal para la recolección de datos, ya que se reconoce la construcción del conocimiento tácito del investigador así como que la investigación se sustenta por el sistema de valores que caracterizan al investigador, el informante, el paradigma que se elija y la teoría sustantiva que se relaciona, estos conceptos se reflejaron al involucrar personas con diferentes puestos y grados de experiencia como informantes permitiendo enriquecer el modelo desde distintos puntos de vista.

Algunos ajustes realizados al modelo se podrían catalogar como obligatorios debido a que prácticamente todos los informantes que evaluaron la propuesta inicial coincidieron en la necesidad de incluirlos, pero por otro lado también fueron indicados algunos elementos que no era posible considerar, debido a que fueron expuestos de manera aislada por algún informante y no agregaban valor al modelo.

Entre los informantes tuvo una gran aceptación la idea de reconocer que el software desarrollado a la medida contiene conocimiento, desencadenando en el hecho de que la ingeniería de software sea integrada con la administración del conocimiento, ya que todos manifestaron que este enfoque resulta necesario para el dinamismo del entorno actual en el que se encuentran involucradas las organizaciones.

El giro en la conceptualización del software dio como resultado el modelo propuesto, el cual es la contribución más importante de esta investigación a la estrategia empresarial debido a que permitirá responder de manera más rápida y efectiva a las condiciones que imponen los mercados actuales en las situaciones en que la solución involucre al software desarrollado a la medida permitiendo una mejor la toma de decisiones, además se realiza una aportación al conocimiento de la administración particularmente en relación al área funcional de informática, debido a que representa un nuevo esquema para coordinar recursos humanos y tecnológicos de las organizaciones, la implementación para fines prácticos de lo anterior se logra a través de la aplicación del modelo propuesto.

Los inconvenientes más fuertes que tiene que superar el modelo, sobre todo en México, son derivados de cuestiones culturales como la resistencia al cambio y la renuencia a compartir el conocimiento, porque hay una tendencia a que la mayoría de las personas sientan amenazados sus puestos de trabajo si no son los únicos en poseer algún conocimiento.

La relación evidente de la presente investigación con la Maestría en Administración con campo de conocimiento en administración de la tecnología, se puede apreciar en la interacción que se estableció entre administración del conocimiento y la ingeniería de software, lo cual también implica la gestión de los conocimientos teóricos y empíricos que se desarrollan en las empresas, debido a que están completamente involucrados con los elementos tecnológicos que operan ya que esos conocimientos son los que definen los procesos que hacen trabajar diariamente a las organizaciones, y a la vez, se está influyendo la manera en como se trabajarán los proyectos de administración de la tecnología cuya base esté formada por el software resultando en la modificación de dichos procesos bajo un enfoque tecnológico.

Queda pendiente para otras investigaciones la aplicación del modelo y medir su desempeño tomando como referencia tres indicadores clave: el dinero invertido, el tiempo requerido para completar y entregar el software final, así como la satisfacción de los usuarios, debido a que no fue posible realizar la investigación de la aplicación del modelo en este trabajo ya que no se contaron con los recursos necesarios.

Otro último un aspecto que enriquecería este el modelo es tomar en cuenta informantes con otros perfiles involucrados en el proceso de ingeniería de software, como por ejemplo desde áreas funcionales de negocio.

Bibliografía

Alistair Cockburn. (2001). *Agile Software Development, Draft version: 3b.* Cockburn * Highsmith Series Editors.

Barreto Nunes, F. J., & Bessa Albuquerque, A. (2010). A Secure Software Development Supported by Knowledge Management. *International Joint Conference on Computer, Information, Systems* (págs. 291-296). Berlin: Springer Science+Business Media B.V.

Basri, S., & O'Connor, R. (2010). Evaluation on Knowledge Management Process in Very Small Software Companies: A Survey. *5th International Conference on Knowledge Management* (págs. 623-631). UTARI, MALAYSIA: UNIV UTARI MALAYSIA-UUM.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., y otros. (11 de Febrero de 2001). *agilemanifesto.* Recuperado el 06 de Marzo de 2012, de http://www.agilemanifesto.org/iso/es/manifesto.html

Bellinger, G., Castro, D., & Mills, A. (2004). *Systems Thinking*. Recuperado el 31 de Marzo de 2012, de Data, Information, Knowledge, and Wisdom: http://www.systems-thinking.org/dikw/dikw.htm

Bjornson, F., & Dingsoyr, T. (2009). A Survey of Perceptions on Knowledge Management Schools in Agile and Traditional Software Development Environments PT S. *10th International Conference on Agile Processes in Software Engineering* (págs. 94-103). BERLIN, Alemania: SPRINGER-VERLAG.

Boden, A., Avram, G., Bannon, L., & Wulf, V. (2009). Knowledge Management in Distributed Software Development Teams – Does Culture Matter? *Fourth IEEE International Conference on Global Software Engineering* (págs. 18-27). Limerick, Irlanda: IEEE.

Bolsegui, M., & Fuguet , S. A. (2006). Construcción de un modelo conceptual a través de la investigación cualitativa. *Sapiens*, 207 - 229.

BRINT Institute LLC. (21 de Febrero de 2012). *Resource for Business, Information Technology, & Finance Risk Management Research*. Recuperado el 21 de Febrero de 2012, de http://www.brint.com

Calvo Manzano, J., Cuevas, G., Munoz , M., San Feliu, T., Alvaro, R., & Sanchez, A. (2010). Identification of best practices of the organization of Software Development through knowledge management. *5th Iberian Conference on Information Systems and Technologies* (págs. 231-237). RIO TINTO, PORTUGAL: AISTI-ASSOC IBERICA SISTEMAS & TECNOLOGIAS INFORMACAO.

Colegio de México. (2010). *Diccionario del español de México*. Recuperado el 23 de Abril de 2012, de http://dem.colmex.mx/GridView.aspx?txtPalabra=modelo

Davenport, T., & Laurence , P. (1998). *Working knowledge: how organizations manage what they know.* Boston: Harvard Business Pres.

Domínguez Jardines, A. L., García Fernández, F., & Sánchez Aldape, A. (2007). ¿Cómo gestionar el conocimiento? Propuesta de un modelo de gestión del conocimiento en las organizaciones. *XI congreso de investigación en ciencias administrativas ACACIA.* Guadalajara, Jalisco: Academia de Ciencias Administrativas, A. C.

Drucker, P. (2000). *El management del siglo XX.* Barcelona: EDHASA.

DSDM Consortium. (2008). *DSDM Atern Teh Handbook.* Chestfiels: DSDM Consortium.

Earl, M. (2001). Knowledge Management Strategies: Toward a Taxonomy. *Journal of Management Information Systems*, 215-233.

García Garibay, S. (2010). *Tecnologías Web 2.0 para administrar el conocimiento de la PyME mexicana.* México D.F. Tesis de maestría, Universidad Nacional Autónoma de México.

Goldoni, V., & Oliveira, M. (2010). Knowledge management metrics in software development companies in Brazil. *JOURNAL OF KNOWLEDGE MANAGEMENT*, 301-313.

Havlice, Z., Kunstar, J., Adamuscinova, I., & Plocica, O. (2009). Knowledge in Software Life Cycle. *TH INTERNATIONAL SYMPOSIUM ON APPLIED MACHINE INTELLIGENCE* (págs. 153-157). NEW YORK, USA: IEEE.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (1997). *Metodología de la Investigación.* Colombia: McGRAW - HILL.

Highsmith, J. (2001). *agilemanifesto.org.* Recuperado el 08 de Marzo de 2012, de http://agilemanifesto.org/history.html

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. New York, NY, USA.

IEEE. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. New York, NY, USA.

Jaakkola, H., Heimbürger, A., & Petri , L. (2010). Knowledge-oriented software engineering process in a multi-cultural context. *Software Qual*, 299–319.

Knowledge Research Institute. (2 de Agosto de 2010). *Knowledge Research Institute*. Recuperado el 20 de Enero de 2012, de http://www.krii.com/

Levy, M., & Hazzan, O. (2009). Knowledge Management in Practice: The Case of Agile Software Development. *ICSE'09 Workshop*, 60-65.

Liu, Y., Wu, J., Liu, X., & Gu, G. (2009). Investigation of Knowledge Management Methods in Software Testing Process. *International Conference on Information Technology and Computer Science* (págs. 90-94). LOS ALAMITOS, USA: IEEE COMPUTER SOC.

Matturro, G., & Silva, A. (2010). A Model for Capturing and Managing Software Engineering Knowledge and Experience. *Journal of Universal Computer Science,*, 479-505.

Ming, C. (2009). Research on Knowledge Management of Software Enterprises. *2nd IEEE International Conference on Computer Science and Information* (págs. 291-294). New York, USA: IEEE.

MiTecnologico.com. (17 de Diciembre de 2007). *Concepto de Modelo Desarrollo Software.* Recuperado el 02 de Marzo de 2012, de MiTecnologico: http://www.mitecnologico.com/Main/ConceptoDeModeloDesarrolloSoftware

Mochi Alemán, P. O. (2006). *La industria del software en México en el contexto internacional y latinoamericano.* Cuernavaca: Centro Regional de Investigaciones Multidisciplinarias, UNAM.

Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company.* New York: Oxford University Pres.

NYCE. (2011). *Normalización y Certificación Electrónica A.C*. Recuperado el 13 de Marzo de 2012, de NMX-I-059/02-NYCE-2011: http://www.moprosoft.com.mx/contenido.aspx?id_pagina=8

Oktaba, H., Alquicira Esquivel, C., Su Ramos, A., Martínez Martínez, A., Quintanilla Osorio, G., Ruvalcaba López, M., y otros. (2005). Modelo de Procesos para la Industria de Software. *MoProSoft*. México, D.F., México: Secretaría de Economía.

Polanyi, M. (1964). *Personal Knowlegde.* New York: Harper & Row.

Pressman, R. S. (2002). *Ingeniería del software un enfoque práctico.* Madrid, España: McGraw-Hill.

Project Mangement Institute. (2008). *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK).* Pennsylvania, USA: Project Mangement Institute.

Ramírez Solís, E. R. (2009). *La percepción de las tácticas políticas, el aprendizaje organizacional y el compromiso laboral en el sector calzado de Jalisco: hacia una propuesta metodológica para la administración del conocimiento en la MIPYME.* México D.F.: Tesis de doctorado, Universidad Nacional Autónoma de México.

Ramos Ayllon, ,. J. (2002). *Inteligencia competitiva y administración del conocimiento. Estudio de viabilidad para crear una área de inteligencia competitiva y administración de conocimiento, como empresa autosustentable, dentro de una compañía productora y comercializadora de telev.* México D.F.: Tesis maestría, Universidad Nacional Autónoma de México.

Rational. (1998). Rational Unified Process Best Practices for Software Development Teams. Cupertino, California.

Rusu, A., Russell, R., Robinson, J., & Rusu, A. (2010). Learning Software Engineering Basic Concepts using a Five-Phase Game. *40th ASEE/IEEE Frontiers in Education Conference* (págs. S2D-1 - S2D-6). IEEE.

S.C. Shang, S., & Shu-Fang , L. (2009). Understanding the effectiveness of Capability Maturity Model Integration by examining the knowledge management of software development processes. *Total Quality Management*, 509–521.

Salem Ben, D. D., & Mouna Ben, C. (2009). The Knowledge-Gap Reduction in Software. *The IEEE International Conference on Researh Challenges in Informational Sciene.* New York: IEEE.

Schwaber, K., & Sutherland, J. (Octubre de 2011). The Scrum Guide. *The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org.

Senge, P. (2006). *La quinta disciplina.* Buenos Aires: Granica.

Serna Montoya, E. (2010). Ontological approach to knowledge management in software maintenance. *Rev. Fac. Ing. Univ. Antioquia*, 184-193.

Sveib, K. E. (1997). *The new organizational wealth.* California: Berrett-Koehler.

Taweel, A., Delaney, B., & Zhao, L. (2009). Knowledge Management in Distributed Scientific Software Development. *Fourth IEEE International Conference on Global Software Engineering* (págs. 299-300). Limerick, Irlanda: IEEE .

Tor, E. F., Tore, D., & Torgeir, D. (2010). Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work. *Information and Software Technology*, 1118–1132.

Valhondo, D. (2003). *Gestión del Conocimeinto del mito a la realidad.* Madrid: Díaz de Santos.

Wells, D. (28 de Septiembre de 2009). *Extreme Programming*. Recuperado el 08 de Marzo de 2012, de extremeprogramming.org: http://www.extremeprogramming.org/

Zapata Cant, L., Rialp I Criado, J., & Rialp I Criado, Á. (2007). La generación y la transferencia de conocimiento en PyMES del sector de las tecnologías de la información. *XI congreso de investigación en ciencias administrativas ACACIA.* Guadalajara, Jalisco: Academia de Ciencias Administrativas, A. C.

Glosario

Arquitectura para los sistemas de software: Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

Compiladores: Programas o herramientas encargadas de traducir un texto (código fuente) escrito en un lenguaje de alto nivel a un lenguaje comprensible por las computadoras (código objeto).

Embebido: Software usado para controlar equipos, operación de maquinarias o plantas industriales completas. El término "embebido" (también se lo conoce como "incrustado") está caracterizado por que es una parte integral del sistema en que se encuentra. Lo interesante de que un software sea "embebido" es que puede estar incrustado de tal forma, que su presencia no resulta obvia sin el dispositivo que lo contiene.

Emuladores: Software que sirve para tener compatibilidad entre distintos tipos de plataformas mediante aplicaciones. Una emulación intenta respetar el funcionamiento y visualización del original, pero no siempre se logra; suele haber problemas en el rendimiento o en los gráficos.

Ensambladores: Herramientas que traducen un archivo de código fuente escrito en lenguaje ensamblador, a un archivo objeto que puede ser ejecutado por la computadora.

Firmware: Combinación de instrucciones de un dispositivo de hardware e instrucciones y datos de computadora que residen como software de solo lectura en ese dispositivo.

Hermenéutica: Pretensión de explicar las relaciones existentes entre un hecho y el contexto en el que acontece.

Manejadores de bases datos: Conjunto de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad

Portafolio de proyectos: Conjunto de proyectos o programas y otros tipos de trabajos que se agrupan para facilitar la dirección eficaz de ese trabajo para cumplir con los objetivos estratégicos de negocio.

Programa de proyectos: Grupo de proyectos relacionados y administrados de forma coordinada para obtener beneficios y control, que no se obtendrían si se gestionaran en forma individual.

Utilerías: Son programas que sirven de apoyo al procesamiento de los trabajos.

Apéndice A, análisis bibliométrico

A.1. Introducción

El análisis bibliométrico es una técnica que consiste en aplicar métodos matemáticos y estadísticos para realizar un estudio de publicaciones científicas relacionadas con un determinado tema, las búsquedas que permitieron conocer la tendencia que prevalece en cuanto a la relación de la administración del conocimiento con el desarrollo de software fueron realizas a partir de la base de datos "*Web of knowledge*" por medio del acceso que brinda la biblioteca digital de la UNAM, a partir de las siguientes combinaciones:

a. "knowledge management"
b. "software development"
c. "tool business software"
d. "knowledge management" y "software development"
e. "knowledge management" y "tool business software"
f. "software development" y "tool business software"
g. "knowledge management" y "software development" y "tool business software"

Los resultados de cada una de las búsquedas fueros exportados en archivos de texto plano y tratados de forma que fuera posible analizarlos mediante una versión de prueba del software *Matheo Analizer*, obteniendo las conclusiones presentadas en las siguientes secciones de este apéndice A.

A.2. Búsqueda de *"knowledge management"*

Para esta búsqueda se realizó un filtrado de información especificando como título "*knowledge management*" y además se acotó al área de ciencias de la computación o ingeniería y con las categorías de sistemas de información o ingeniería eléctrica electrónica del año 2009 al 2012. La búsqueda arrojó 2,281 resultados a partir de los cuales fue posible inferir:

- Los autores con mayor número de publicaciones se indican en la tabla A1.

Tabla A1. Autores más publicaciones para knowledge management.

Autor	Publicaciones
Wang, J	10
Zhang, Y	9
Liu, L	9
Chen, YJ	8

Fuente: Elaboración propia.

- Los autores más citados son listados en la tabla A2:

Tabla A2. Autores más citados para knowledge management.

Autor	Ocasiones citado
Nonaka I	258
Davenport TH	193
Alavi M	147
Grant RM	81
Gruber TR	69
Markus ML	66
Hanse MT	65

Fuente: Elaboración propia.

- Como se aprecia en la tabla A2 el tipo de autores más citados son personas.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (consultar Figura A1), no se detecta presencia de relación entre administración del conocimiento y desarrollo de software.
- La distribución de los porcentajes de publicaciones acorde al año se muestra en la tabla A3.

Tabla A3. Porcentaje de publicaciones por año para knowledge management.

Año	% de publicaciones
2009	52.7
2010	30.47
2011	16.31
2012	0.53

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones es China, como se puede apreciar en la tabla A4.

Tabla A4. Porcentaje de publicaciones por año para knowledge management.

País	Lugar	Publicaciones	%
China	1°	486	32.79
USA	2°	341	23.01
Taiwán	3°	169	11.4
México	33°	14	0.94

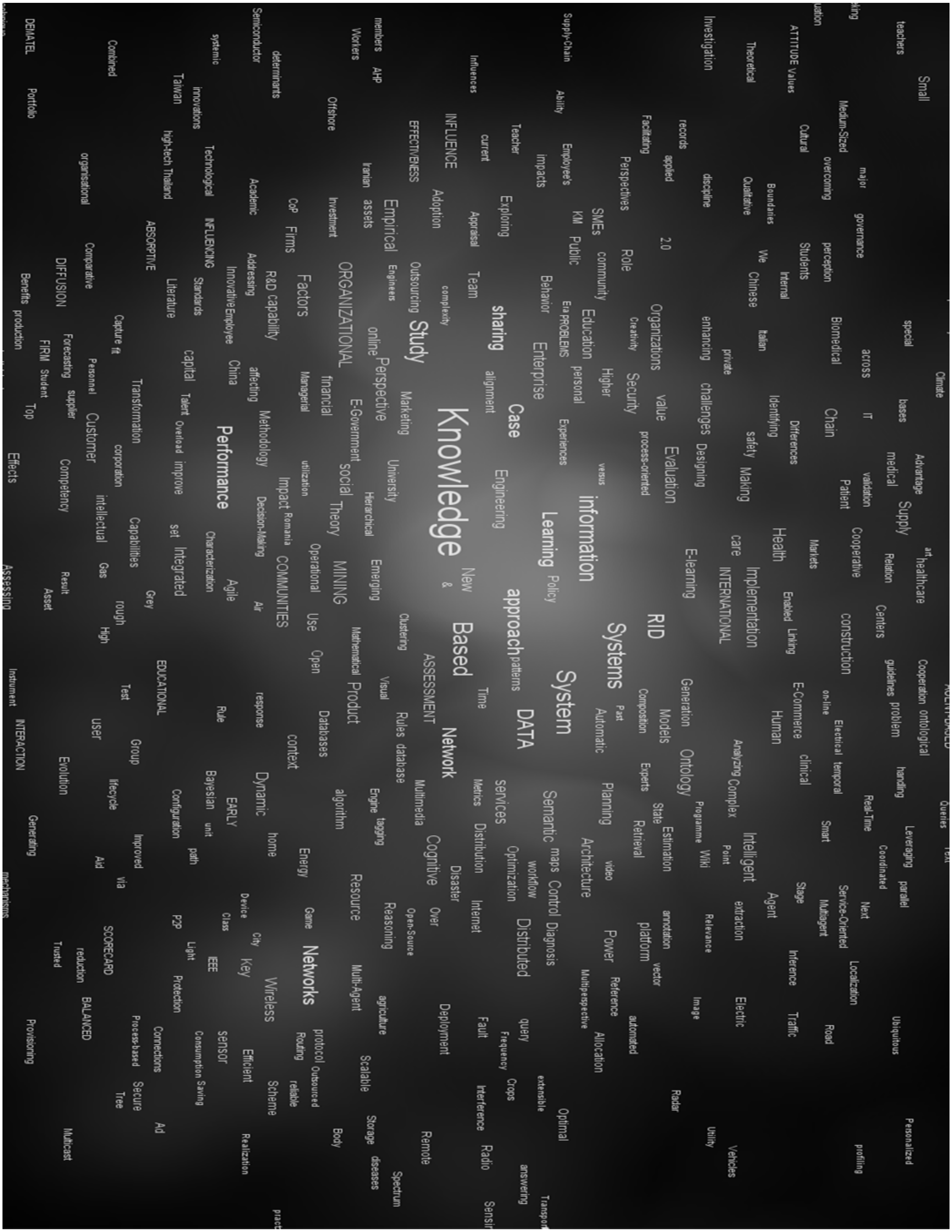
Fuente: Elaboración propia.

- De los países listados en la tabla A4 todos salvo México cuentan con publicaciones en los años 2009, 2010 y 2011.
- Las publicaciones realizadas en México fueron temas relacionados con:
    - Investigación de operaciones y ciencias de la administración.
    - Ciencias de la computación.
    - Energía y combustibles.
    - Ingeniería.
    - Ciencias de la información.

- Economía y negocios.

- La tendencia del tipo de instituciones que publican en México, son universidades tanto públicas como privadas.
- El diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda mostrados en la Figura A1, no revela alguna tendencia o clúster notable de relación entre la administración del conocimiento y el desarrollo de Software.

Figura A1. Diagrama de densidad de palabras clave para knowledge management.



Fuente: Elaboración propia.

A.3. Búsqueda de *"software development"*

Se realizó un filtro de búsqueda especificando como título "software development" y además se acotó a las categorías de: métodos de las teorías de las ciencias de la computación, ingeniería de software en ciencias de la computación así como aplicaciones interdisciplinarias de las ciencias de la computación y se consideró sólo el área de ciencias de la computación, del año 2009 al 2012. La búsqueda arrojó 1,908 resultados a partir de los cuales fue posible inferir:

- Los autores con mayor número de publicaciones son presentados en la tabla A5.

Tabla A5. Autores más publicaciones para software development.

Autor	Publicaciones
Babar, MA	11
Piattini, M	11
Fernandez-Medina, E	11
Hassan, AE	9

Fuente: Elaboración propia.

- Los autores más citados se listan en la tabla A6.

Tabla A6. Autores más citados para software development.

Autor	Ocasiones citado
BECK K	139
*OMG	138
Boehm B	134
BOEHM BW	106
GAMMA E	98
FOWLER M	97
BASILI VR	92
*IEEE	84

Fuente: Elaboración propia.

- En temas de desarrollo de software se toma de manera importante las referencias de instituciones como se puede apreciar en la tabla A6.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A2) no se detecta presencia de relación entre administración del conocimiento y desarrollo de software.
- La distribución de los porcentajes de publicaciones en los años analizados se indica en la tabla A7.

Tabla A7. Porcentaje de publicaciones por año para software development.

Año	% de publicaciones
2009	54.77
2010	23.27
2011	21.28
2012	0.68
2012	0.82

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA, como se parecía en la tabla A8.

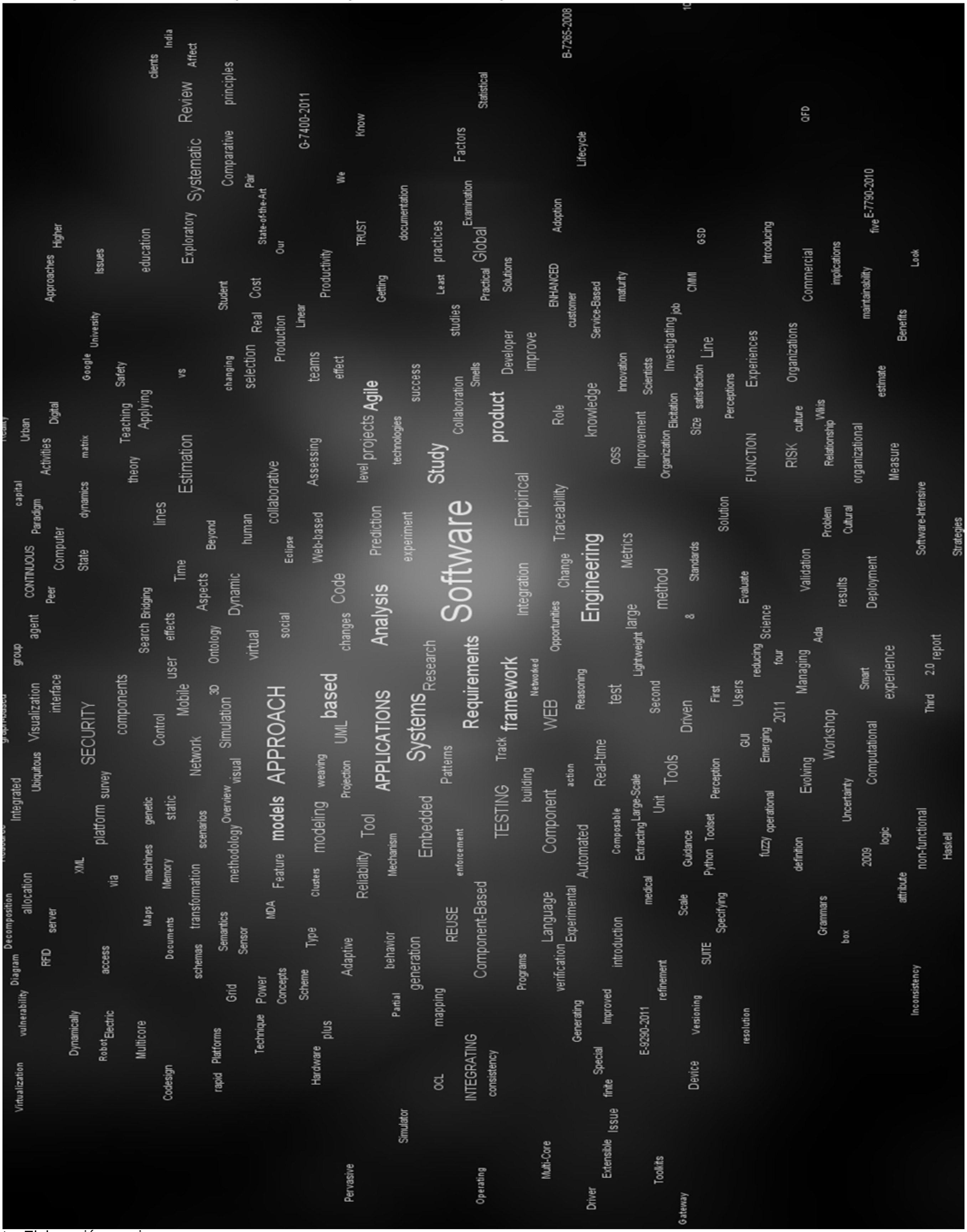
Tabla A8. Porcentaje de publicaciones por año para software development.

País	Lugar	Publicaciones	%
USA	1°	393	31.26
China	2°	141	11.22
Alemania	3°	141	11.22
España	4°	135	10.74
México	42°	8	0.64

Fuente: Elaboración propia.

- De los países presentados en la tabla A8 anteriormente USA, China, España y México han publicado en los años 2009, 2010, 2011 y 2012, Alemania no cuenta con publicaciones en el 2012.
- Las publicaciones realizadas en México fueron temas relacionados con:
  - Ciencias de la computación.
  - Ingeniería.
- La tendencia del tipo de instituciones que publican en México, son universidades tanto públicas como privadas.
- El diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda presentado en la Figura A2 no revela alguna tendencia o clúster notable de relación entre la administración del conocimiento y el desarrollo de Software.

Figura A2. Diagrama de densidad de palabras clave para software development.



Fuente: Elaboración propia.

A.4. Búsqueda de *"tool business software"*

El filtro de búsqueda especificando como título "*tool Business software*", del año 2009 al 2012, devolvió 369 resultados a partir de los cuales fue posible inferir lo siguiente:

- Los autores con mayor número de publicaciones son presentados en la tabla A9.

Tabla A9. Autores más publicaciones para tool Business software.

Autor	Publicaciones
Daniel, F	3
Zhang, J	2
Wu, CR	2
West, AA	2

Fuente: Elaboración propia.

- Los autores citados más frecuentemente son los mostrados en la tabla A10.

Tabla A10. Autores más citados para tool Business software.

Autor	Ocasiones citado
BECK K	139
*OMG	138
Boehm B	134
BOEHM BW	106
GAMMA E	98
FOWLER M	97
BASILI VR	92
*IEEE	84

Fuente: Elaboración propia.

- Como se aprecia en la tabla A10 se tiene una presencia importante de referencias a instituciones.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A3) no se aprecia una posible relación entre administración del conocimiento y desarrollo de software.
- La distribución de los porcentajes de publicaciones en los años analizados se presenta en la tabla A11.

Tabla A11. Porcentaje de publicaciones por año para tool Business software.

Año	% de publicaciones
2009	44.44
2010	38.48
2011	16.26
2012	0.82

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA, como se muestra en la tabla A12:

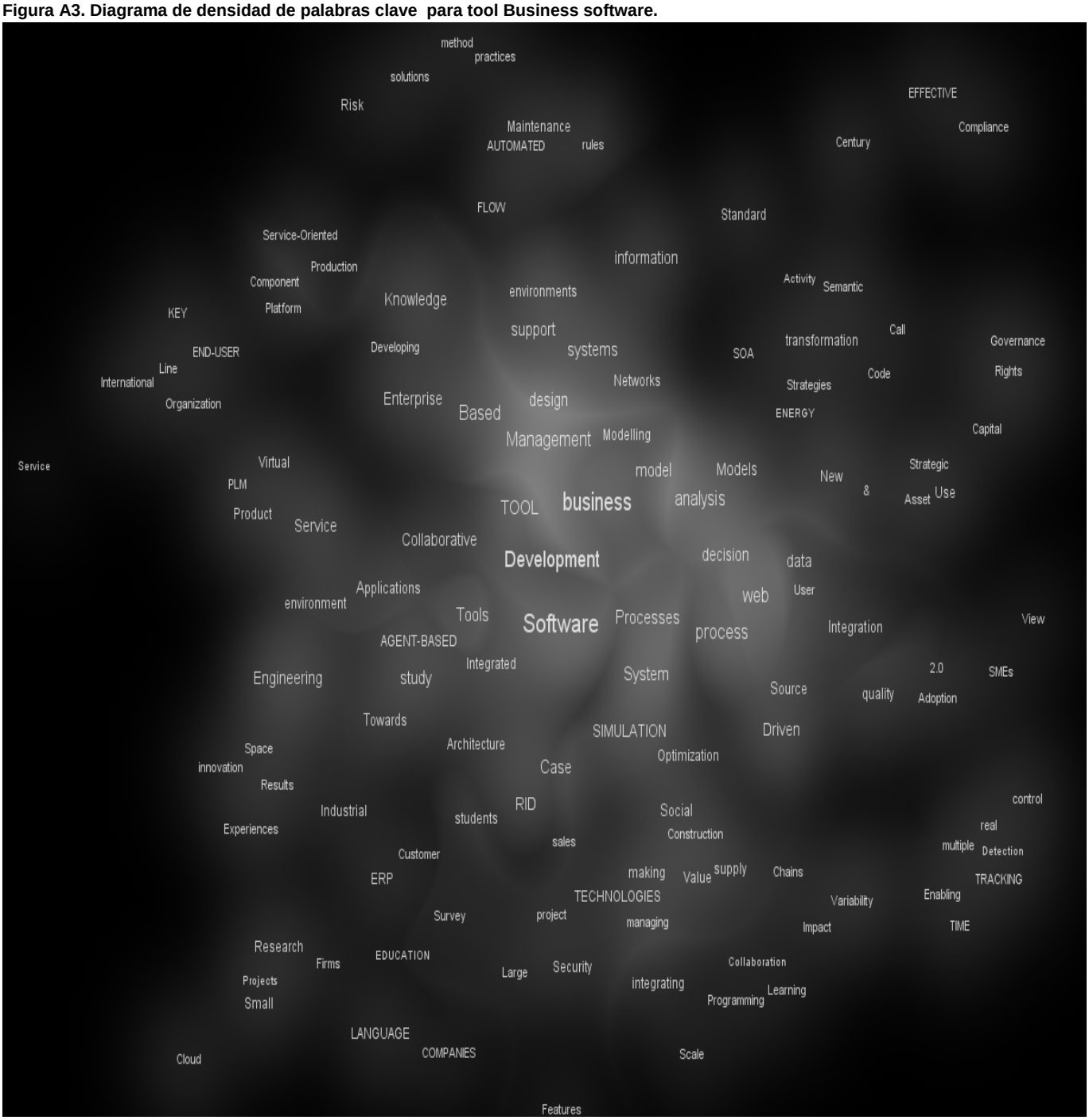
Tabla A12. Porcentaje de publicaciones por año para tool Business software.

País	Lugar	Publicaciones	%
USA	1°	59	26.94
Alemania	2°	38	17.35
Italia	3°	21	9.59
Inglaterra	4°	21	9.59
México*	35°	3	1.37

Fuente: Elaboración propia.

- De los presentados en la tabla A8 USA, Alemania, Italia e Inglaterra han publicado en los años 2009, 2010, 2011, México cuenta con publicaciones en el 2009 y 2010.
- Las publicaciones realizadas en México fueron temas relacionados con:
  - Ciencias de la computación.
- La tendencia del tipo de instituciones que publican en México, son universidades públicas.
- El diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda mostrado en la Figura A3 no revela alguna tendencia o clúster notable de relación entre la administración del conocimiento y el desarrollo de Software.

Figura A3. Diagrama de densidad de palabras clave para tool Business software.



Fuente: Elaboración propia.

A.5. Búsqueda de *"knowledge management" y "software development"*

Se realizó un filtro de búsqueda especificando como título "knowledge management" y "software development", del año 2009 al 2012, obteniendo 436 resultados a partir de los cuales fue posible inferir:

- Los autores con mayor número de publicaciones son los mostrados en la tabla A13.

Tabla A13. Autores más publicaciones para knowledge management" y software development.

Autor	Publicaciones
Mentzas, G	3
von Krogh, G	3
Assawamekin, N	3
Sunetnanta, T	3

Fuente: Elaboración propia.

- Los autores más citados se presentan en la tabla A14

Tabla A14. Autores más citados para knowledge management" y software development.

Autor	Ocasiones citado
NONAKA I	46
DAVENPORT TH	24
Eisenhardt KM	24
Faraj S	21
BOEHM B	21
Herbsleb JD	10
CARMEL E	19
Markus ML	19

Fuente: Elaboración propia.

- Como se aprecia en la tabla A14 el tipo de autores citados que dominan son personas.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A4) es posible inferir una relación entre administración del conocimiento y desarrollo de software.
- El porcentaje de publicaciones en cada año analizado se muestra en la tabla A15.

Tabla A15. Porcentaje de publicaciones por año para knowledge management" y software development.

Año	% de publicaciones
2009	41.74
2010	38.53
2011	19.5
2012	0.23

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA, como se presenta en la tabla A16.

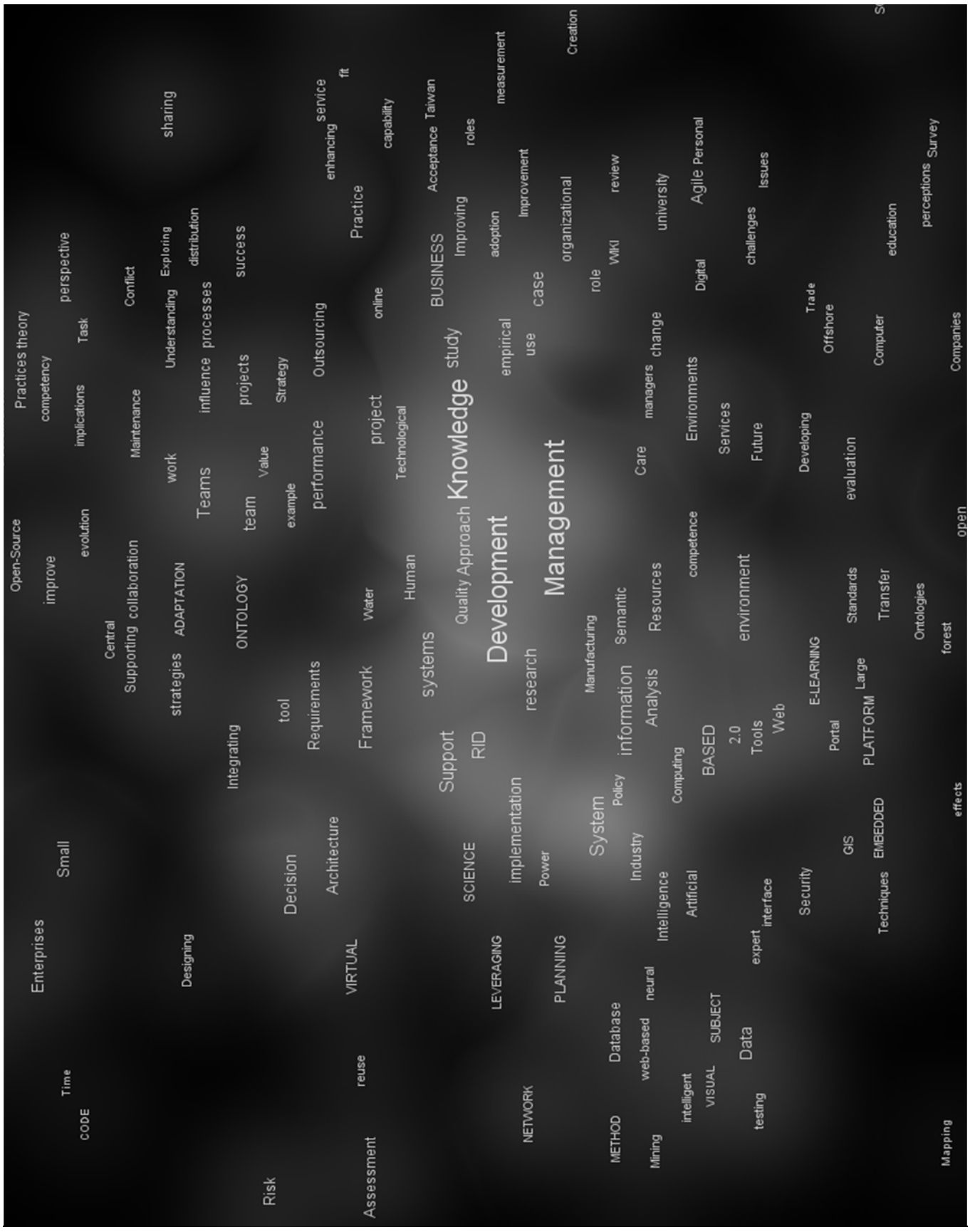
Tabla A16. Porcentaje de publicaciones por año para knowledge management" y software development.

País	Lugar	Publicaciones	%
USA	1°	79	26.6
Alemania	2°	48	16.16
China	3°	36	12.12
España	4°	29	9.76
Inglaterra	5°	26	8.75

Fuente: Elaboración propia.

- Los países listados en la tabla A16 han publicado en los años 2009, 2010, 2011.
- No se localizo participación de México.
- En el diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda presentado en la Figura A4 se aprecia relación entre la administración del conocimiento y el desarrollo de Software ligada a posibles clústeres acerca de:
    - Mejora en procesos de negocios e innovación.
    - Desempeño de equipos de trabajo.
    - Métodos de implementación de sistemas de información de negocios.

Figura A4. Diagrama de densidad de palabras clave para knowledge management" y software development.



Fuente: Elaboración propia.

A.6. Búsqueda de *"knowledge management" y "tool business software"*

El filtro de para esta búsqueda se especificó como título "knowledge management" y "tool business software", del año 2009 al 2012, obteniendo 45 resultados a partir de los cuales fue posible inferir:

- Los autores con mayor número de publicaciones se presentan en la tabla A17.

Tabla A17. Autores más publicaciones para knowledge management y tool business.

Autor	Publicaciones
Capatina, A	2
van Der Merwe, A	2
Moser, T	1
Biffl, S	1

Fuente: Elaboración propia.

- Los autores más citados en las publicaciones son listados en tabla A18.

Tabla A18. Autores más citados para knowledge management y tool business

Autor	Ocasiones citado
NONAKA I	5
WIEGERS KE	4
Davenport TH	4
KAPLAN RS	4
Alavi M	4
KRUCHTEN P	3
REGNELL B	3
SVEIBY KE	3

Fuente: Elaboración propia.

- Como se aprecia en la tabla A18 el tipo de autores citados que dominan son personas.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A4) es posible inferir una relación entre administración del conocimiento y desarrollo de software.
- La distribución de los porcentajes de publicaciones en los años analizados es mostrado en la tabla A19.

Tabla A19. Porcentaje de publicaciones por año para knowledge management" y tool business

Año	% de publicaciones
2009	51.11
2010	35.56
2011	11.11
2012	2.22

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA, como se presenta en la tabla A20.:

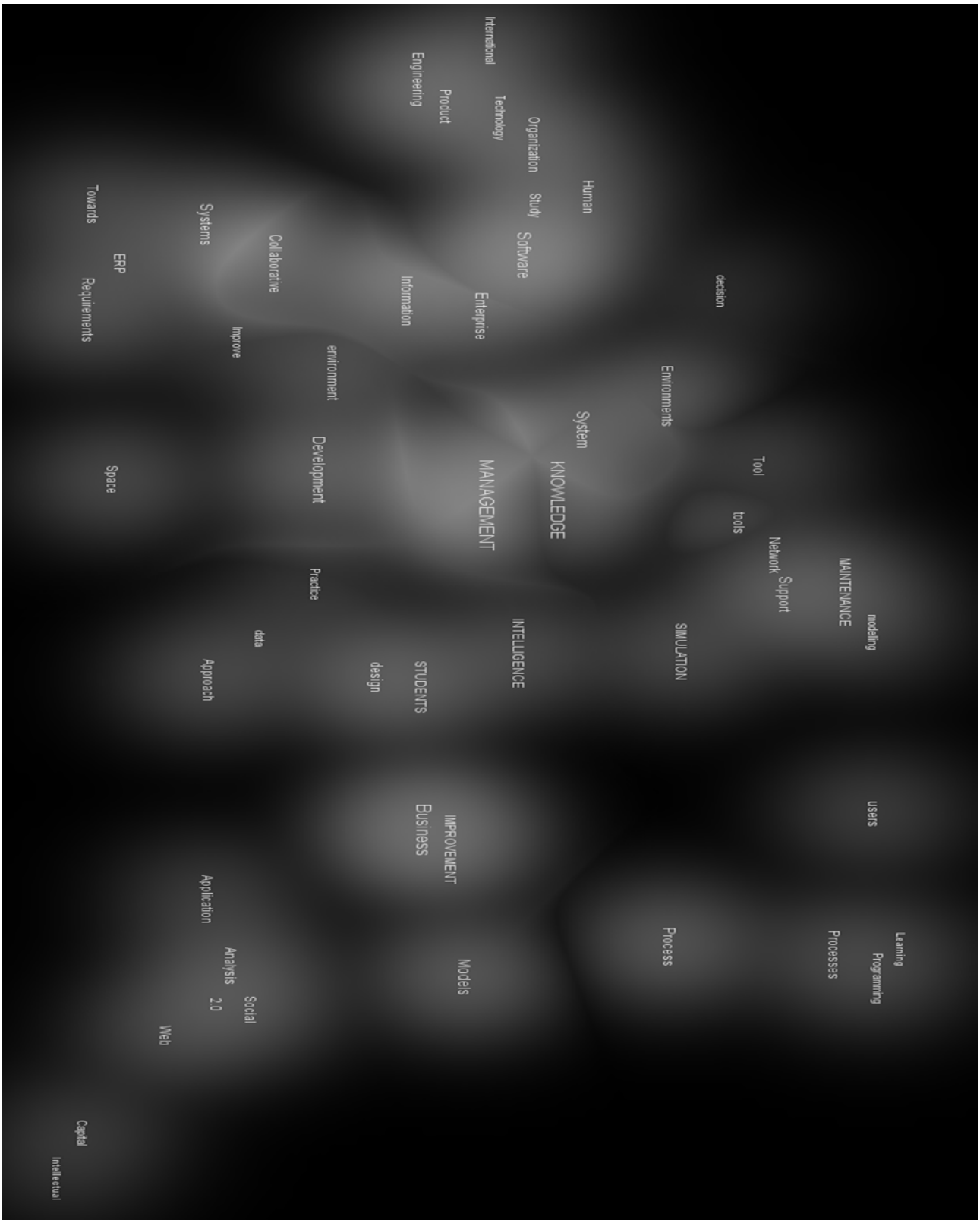
Tabla A20. Porcentaje de publicaciones por año para knowledge management y tool business

País	Lugar	Publicaciones	%
USA	1°	7	20
Italia	2°	6	17.14
Alemania	3°	5	14.29
Rumania	4°	4	11.43
Inglaterra	5°	3	8.57

Fuente: Elaboración propia.

- De los países contenidos en la tabla A20 USA e Italia han publicado en los años 2009, 2010 y 2011, Italia y Rumania en los años 2009 y 2010 e Inglaterra solamente en 2011.
- No se localizo participación de México.
- El diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda, mostrado en la Figura A5, se aprecia relación entre la administración del conocimiento y el desarrollo de Software ligada a posibles clústeres acerca de:
    - Requerimientos en el desarrollo de sistemas.
    - Herramientas para la administración de la ingeniería de Software.
    - Procesos de aprendizaje de los usuarios.
    - Modelos para mejora de negocios.

Figura A5. Diagrama de densidad de palabras clave para knowledge management y software development.



Fuente: Elaboración propia.

A.7. Búsqueda de *"software development" y "tool business software"*

En este caso fue configurado un filtro de búsqueda especificando como título "software development" y "tool business software", del año 2009 al 2012, y se obtuvieron 45 resultados a partir de los cuales fue posible inferir:

- Los autores con mayor número de publicaciones se muestran en la tabla A21:

Tabla A21. Autores más publicaciones para software development y tool business software.

Autor	Publicaciones
Ren, WJ	2
Cebon, D	2
Arnold, SM	2
Kokash, N	2

Fuente: Elaboración propia.

- Los autores más citados son presentados en la tabla A22.

Tabla A22. Autores más citados para software development y tool business software.

Autor	Ocasiones citado
*OMG	15
*ISO IEC	9
*ISO	8
GAMMA E	8
*OBJ MAN GROUP	8
BOEHM B	8
*OASIS	7
BECK K	7

Fuente: Elaboración propia.

- Como se aprecia en la tabla A22 el tipo de autores citados que dominan son instituciones.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A5) no es posible inferir una relación entre administración del conocimiento y desarrollo de software.
- El porcentaje de publicaciones dentro de los años analizado es mostrado en la tabla A23.

Tabla A23. Porcentaje de publicaciones por año para software development y tool business software.

Año	% de publicaciones
2009	45.45
2010	35.18
2011	15.15
2012	1.21

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA, como se puede apreciar en la tabla A24.

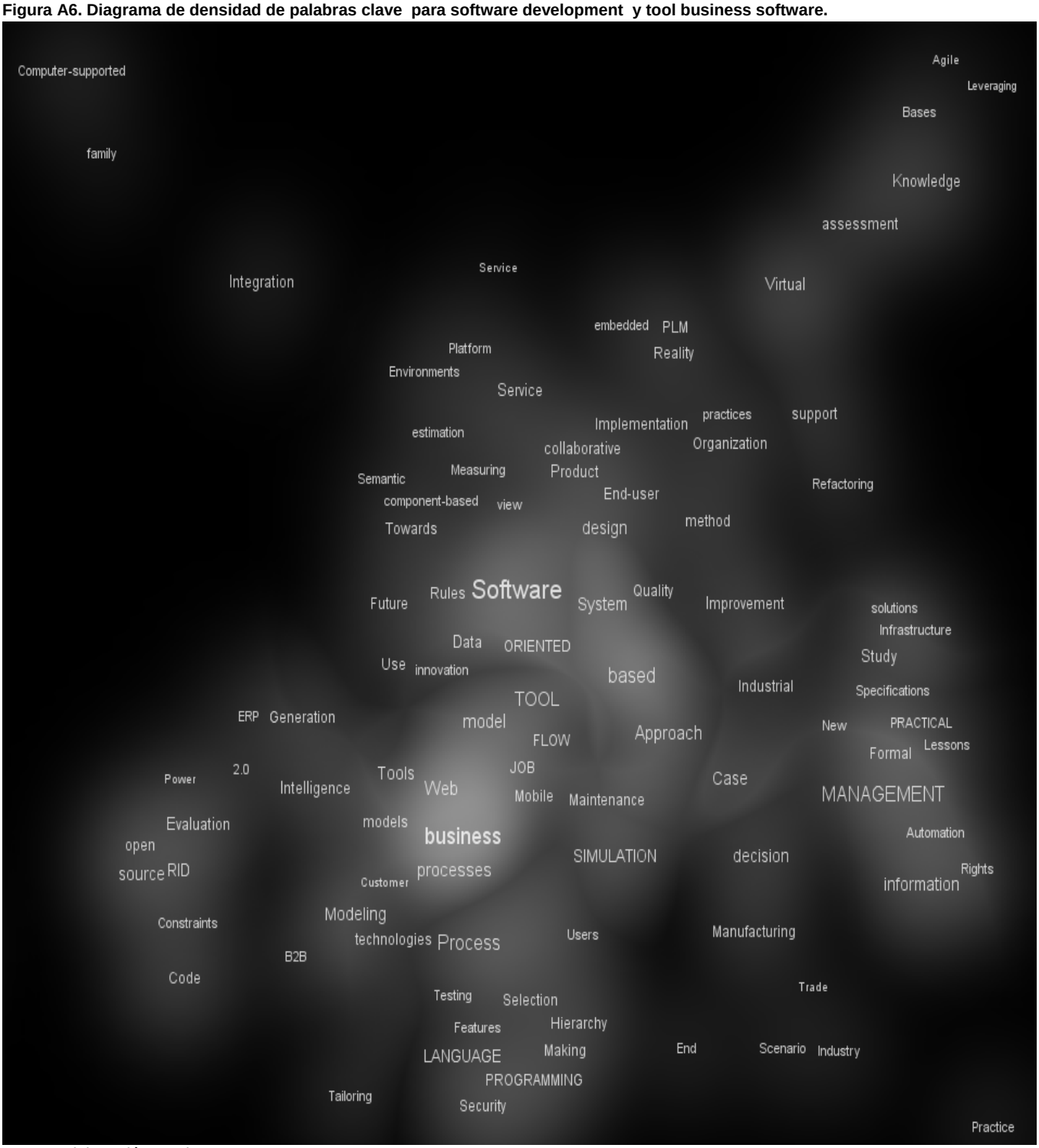
Tabla A24. Porcentaje de publicaciones por año para software development y tool business software.

País	Lugar	Publicaciones	%
USA	1°	26	27.61
Alemania	2°	17	14.17
España	3°	12	10
Italia	4°	9	7.5
México*	19°	3	2.5

Fuente: Elaboración propia.

- De los países listados en la tabla A24 USA, Alemania, España e Italia han publicado en los años 2009, 2010 y 2011, México tiene publicaciones en el 2009 y 2010.
- Las publicaciones realizadas en México fueron temas relacionados con:
  - Ciencias de la computación.
- La tendencia del tipo de instituciones que publican en México, son universidades públicas.
- En el diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda presentado en la Figura A6 se aprecia un posible clúster de:
  - Bases para el soporte de activos del conocimiento.

Figura A6. Diagrama de densidad de palabras clave para software development y tool business software.



Fuente: Elaboración propia.

A.8. Búsqueda de *"knowledge management" y "software development" y "tool business software"*

Se realizó un filtro de búsqueda especificando como título "knowledge management", "software development" y "tool business software", del año 2009 al 2012, la búsqueda arrojó 21 resultados a partir de los cuales fue posible inferir:

- No se identificaron autores que destaquen por su número de publicaciones.
- Los autores más citados se listan en la tabla A25.

Tabla A25. Autores más citados para knowledge management, software development y tool business software.

Autor	Veces citado
WIEGERS KE	4
Boehm B	3
KRUCHTEN P	3
REGNELL B	3
SOMMERVILLE I	3
Ahmed S	2
ALAVI M	2
BERNSTEIN PA	2

Fuente: Elaboración propia.

- Como se aprecia en la tabla A25 el tipo de autores citados que dominan son personas.
- De acuerdo a las palabras clave que prevalecieron en las publicaciones (ver Figura A6) es posible inferir una relación entre administración del conocimiento y desarrollo de software.
- El porcentaje de publicaciones en cada año analizado es presentado en la tabla A26.

Tabla A26. Porcentaje de publicaciones por año para knowledge management, software development y tool business software.

Año	% de publicaciones
2009	38.1
2010	38.1
2011	19.05
2012	4.76

Fuente: Elaboración propia.

- El país que ha realizado el mayor número de publicaciones del tema es USA como se muestra en la tabla A27.

Tabla A27. Porcentaje de publicaciones por año para para knowledge management, software development y tool business software.

País	Lugar	Publicaciones	%
USA	1°	4	18.18
Italia	2°	3	13.64
Inglaterra	3°	2	9.09
Finlandia	4°	2	9.09
Francia	5°	2	9.09

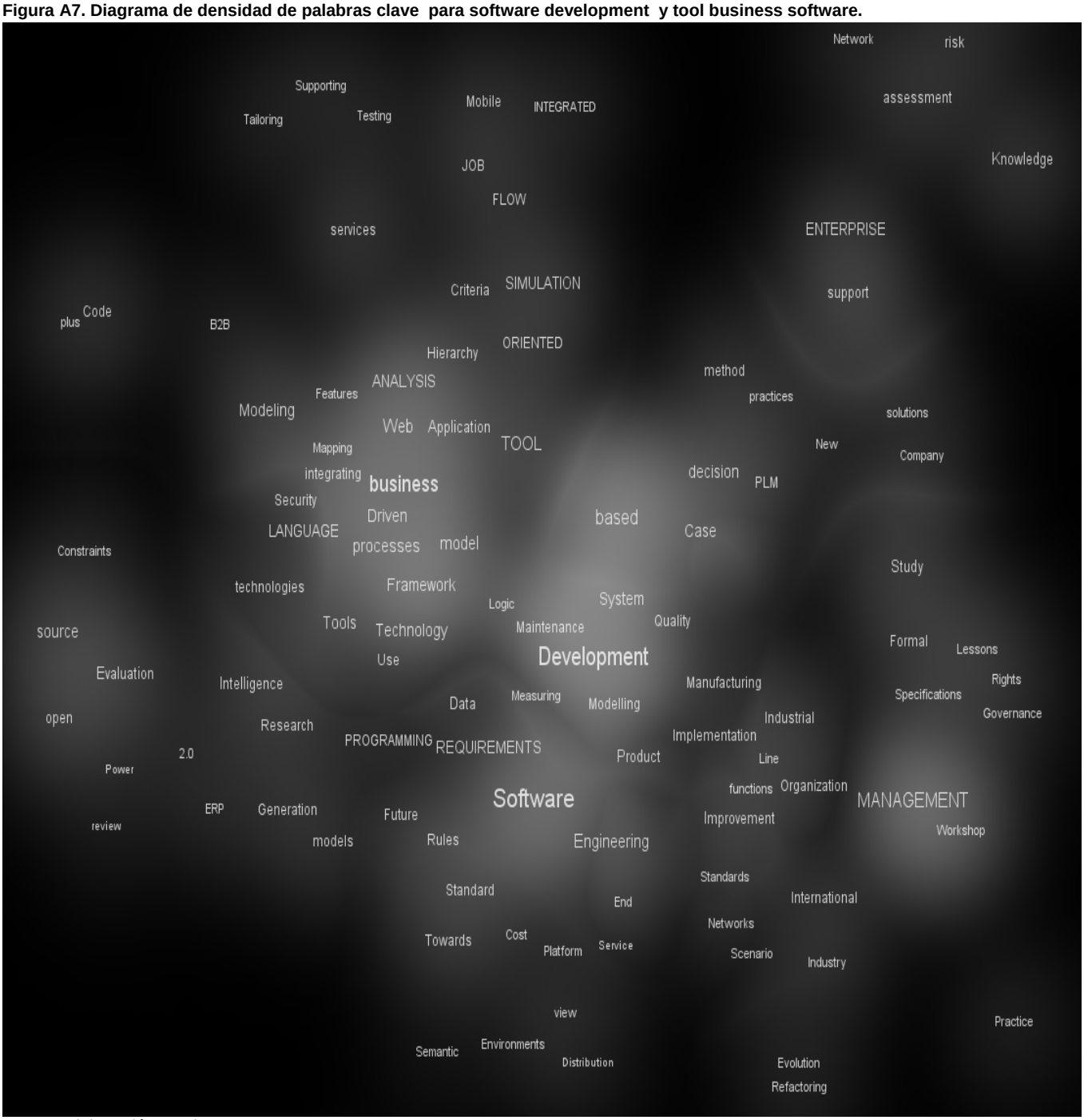
Fuente: Elaboración propia.

- No se localizo participación de México.
- El diagrama de densidad por títulos de las publicaciones arrojadas en la búsqueda, mostrado en la Figura A7, se aprecia relación entre la

administración del conocimiento y el desarrollo de Software ligada a posibles clústeres acerca de:

- Soporte profesional a sistemas empresariales.
- Requerimientos para ERP.
- Ingeniería de software.
- Administración del conocimiento en diseño y desarrollo.
- Estandarización de productos.
- Herramientas/modelos servicios al negocio y usuarios.

Figura A7. Diagrama de densidad de palabras clave para software development y tool business software.



Fuente: Elaboración propia.

A.9. Conclusiones generales del análisis bibliométrico

Los puntos más representativos del análisis realizado son:

- China es el país que más ha contribuido en los últimos años en el campo de la administración del conocimiento.
- En temas relacionados con administración del conocimiento el tipo de autores que más se usan como referencia son personas, a diferencia de las cuestiones de software en las que recurre más a instituciones.
- El número de publicaciones para las búsquedas disminuye progresivamente cada año.
- USA prevaleció con el país con más publicaciones durante los años analizados salvo en el tema de administración del conocimiento.
- Los autores más referenciados que dominaron a través de las diversas búsquedas son:
  - Nonaka I
  - Davenport TH
  - Alavi M
  - KRUCHTEN P
  - BECK K
  - *OMG
  - *IEEE
- Fue posible localizar algunas publicaciones relacionadas el tema tratado en el presente trabajo.

Filtrando los resultados de las búsquedas limitándolos a publicaciones cuyo título contengan las palabras “knowledge” así como “software” y que se pudiera inferir que tienen relevancia para el tema de la tesis, dio como resultado que de cada una de las búsquedas se tomara el número de artículos mostrados en la tabla A28.

Tabla A28. Numero de artículos seleccionados por cada búsqueda en relación a resultados totales.

Búsqueda	Resultados totales	Resultados seleccionados
“knowledge management”	2,281	23
“software development”	1,908	29
“tool business software”	369	7
“knowledge management” y “software development”	436	30
“knowledge management” y “tool business software”	45	3
“software development” y “tool business software”	165	2
“knowledge management”, “software development” y “tool business software”	21	2

Fuente: Elaboración propia.

Finalmente se realizó un cruce entre los resultados elegidos y se pudo apreciar algunas publicaciones recurrentes en las diferentes búsquedas las cuales son presentadas en la tabla A29, donde las letras del encabezado representan las siguientes combinaciones:

a. “knowledge management”
b. “software development”
c. “tool business software”
d. “knowledge management” y “software development”
e. “knowledge management” y “tool business software”
f. “software development” y “tool business software”
g. “knowledge management”, “software development” y “tool business software”

Tabla A29. Publicaciones recurrentes en las búsquedas

Título de la publicación	a	b	c	d	e	f	g
The Knowledge-Gap Reduction in Software Engineering	X	X	X	X	X	X	X
Knowledge Management in Global Software Development	X		X	X	X	X	X
A Secure Software Development Supported by Knowledge Management	X	X		X			
Design guidelines for software processes knowledge repository development	X	X		X			
Improving General Knowledge in Agile Software Organizations Experiences with job rotation in customer support	X	X		X			
Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work	X	X		X			
Investigating the relationship between schedules and knowledge transfer in software testing	X	X		X			
Levering software reuse with knowledge management in software development	X	X		X			

Fuente: Elaboración propia.

Finalmente con base en su recurrencia en las búsquedas y al interés de su posible contenido se procedió a revisar como referencia base para el trabajo las siguientes 19 publicaciones:

- Barreto Nunes, F. J., & Bessa Albuquerque, A. (2010). A Secure Software Development Supported by Knowledge Management. *International Joint Conference on Computer, Information, Systems* (págs. 291-296). Berlin, Alemania: Springer Science+Business Media B.V.
- Basri, S., & O'Connor, R. (2010). Evaluation on Knowledge Management Process in Very Small Software Companies : A Survey. *5th International Conference on Knowledge Management* (págs. 623-631). UTARI, MALAYSIA: UNIV UTARI MALAYSIA-UUM.
- Bjornson, F., & Dingsoyr, T. (2009). A Survey of Perceptions on Knowledge Management Schools in Agile and Traditional Software Development Environments PT S. *10th International Conference on Agile Processes in Software Engineering* (págs. 94-103). BERLIN, Alemania: SPRINGER-VERLAG.

- Boden, A., Avram, G., Bannon, L., & Wulf, V. (2009). Knowledge Management in Distributed Software Development Teams – Does Culture Matter? *Fourth IEEE International Conference on Global Software Engineering* (págs. 18-27). Limerick, Irlanda: IEEE.
- Calvo Manzano, J., Cuevas, G., Munoz , M., San Feliu, T., Alvaro, R., & Sanchez, A. (2010). Identification of best practices of the organization of Software Development through knowledge management. *5th Iberian Conference on Information Systems and Technologies* (págs. 231-237). RIO TINTO, PORTUGAL: AISTI-ASSOC IBERICA SISTEMAS & TECNOLOGIAS INFORMACAO.
- Earl, M. (2001). Knowledge Management Strategies: Toward a Taxonomy. *Journal of Management Information Systems*, 215-233.
- Goldoni, V., & Oliveira, M. (2010). Knowledge management metrics in software development companies in Brazil. *JOURNAL OF KNOWLEDGE MANAGEMENT*, 301-313.
- Havlice, Z., Kunstar, J., Adamuscinova, I., & Plocica, O. (2009). Knowledge in Software Life Cycle. *TH INTERNATIONAL SYMPOSIUM ON APPLIED MACHINE INTELLIGENCE* (págs. 153-157). NEW YORK, USA: IEEE.
- Jaakkola, H., Heimbürger, A., & Petri , L. (2010). Knowledge-oriented software engineering process in a multi-cultural context. *Software Qual*, 299–319.
- Levy, M., & Hazzan, O. (2009). Knowledge Management in Practice: The Case of Agile Software Development. *ICSE'09 Workshop*, 60-65.
- Liu, Y., Wu, J., Liu, X., & Gu, G. (2009). Investigation of Knowledge Management Methods in Software Testing Process. *International Conference on Information Technology and Computer Science* (págs. 90-94). LOS ALAMITOS, USA: IEEE COMPUTER SOC.
- Matturro, G., & Silva, A. (2010). A Model for Capturing and Managing Software Engineering Knowledge and Experience. *Journal of Universal Computer Science,*, 479-505.
- Ming, C. (2009). Research on Knowledge Management of Software Enterprises. *2nd IEEE International Conference on Computer Science and Information* (págs. 291-294). New York, USA: IEEE.
- Rusu, A., Russell, R., Robinson, J., & Rusu, A. (2010). Learning Software Engineering Basic Concepts using a Five-Phase Game. *40th ASEE/IEEE Frontiers in Education Conference* (págs. S2D-1 - S2D-6). IEEE.
- S.C. Shang, S., & Shu-Fang , L. (2009). Understanding the effectiveness of Capability Maturity Model Integration by examining the knowledge management of software development processes. *Total Quality Management*, 509–521.
- Salem Ben, D. D., & Mouna Ben, C. (2009). The Knowledge-Gap Reduction in Software. *The IEEE International Conference on Researh Challenges in Informational Sciene.* New York, USA: IEEE.

- Serna Montoya, E. (2010). Ontological approach to knowledge management in software maintenance. *Rev. Fac. Ing. Univ. Antioquia*, 184-193.
- Taweel, A., Delaney, B., & Zhao, L. (2009). Knowledge Management in Distributed Scientific Software Development. *Fourth IEEE International Conference on Global Software Engineering* (págs. 299-300). Limerick, Irlanda: IEEE .
- Tor, E. F., Tore, D., & Torgeir, D. (2010). Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work. *Information and Software Technology*, 1118–1132.

Apéndice B, Resumen general enviado a los informantes.

Modelo de ingeniería de software con base en directrices de administración del conocimiento

Resumen

Los sistemas de software “hechos/configurados a la medida” se han convertido en un elemento primordial en las operaciones diarias de las organizaciones, a la fecha se han dedicado gran cantidad de trabajos con el propósito de analizar los motivos de los retrasos en proyectos, presupuestos arriba de lo planeado, productos de software de baja calidad e inclusive con funcionalidad inadecuada. Las soluciones más aplicadas hasta el momento están centradas en mejoras al ciclo de vida del software con base en herramientas de desarrollo, métodos o procesos, sin embargo, el alcance de estas soluciones suele estar limitado por enfocarse en problemas específicos y los beneficios que aportan no resultan significativos debido a que no se ha tomado en cuenta el hecho de que el software es resultado de la acumulación del conocimiento poseído por las personas que forman las organizaciones, dejando de lado que los problemas relacionados con el software se pueden mitigar disminuyendo la brecha de conocimiento existente entre lo que saben las personas que forman una organización y el conocimiento contenido en el software, por lo cual se requiere un modelo que permita integrar de forma eficiente la mayor cantidad del *know-how* de las organizaciones, considerando el proceso de desarrollo de software desde una perspectiva de la administración del conocimiento.

Palabras clave: Administración del Conocimiento, Desarrollo de Software.

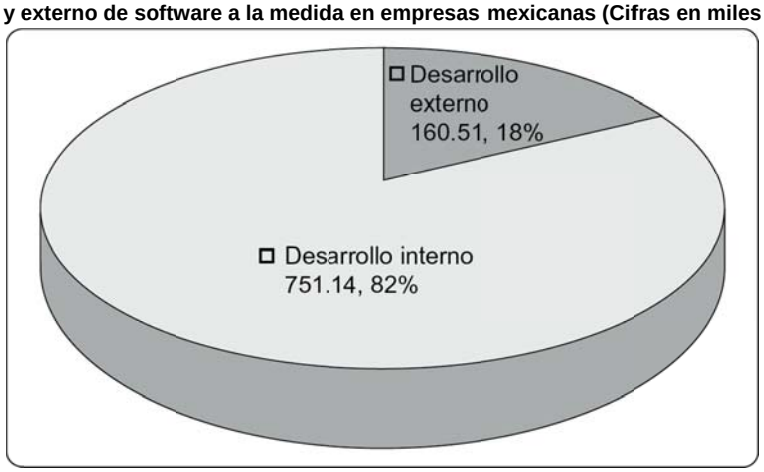
Introducción

Los sistemas de software desarrollados a la medida se han convertido en un elemento primordial en las operaciones diarias de las organizaciones. Tal y como lo mencionan Salem Ben y Mouna Ben (2009), a la fecha se han dedicado gran cantidad de trabajos orientados a los sistemas de información e ingeniería de software con el propósito de analizar los motivos de los retrasos en proyectos, presupuestos arriba de lo planeado, productos de software de baja calidad e inclusive con funcionalidad inadecuada. Desde los años 60's se ha empleado la expresión *crisis de software*, para hacer referencia a los problemas anteriores, realizándose varias propuestas de manera que sea posible aumentar la productividad y calidad del software al nivel requerido por el entorno en el que se encuentran inmersas actualmente las organizaciones.

Las soluciones aplicadas hasta el momento están centradas en mejoras al ciclo de vida del software con base en herramientas de desarrollo, métodos o procesos, sin embargo, el alcance de estas soluciones suele estar limitado por enfocarse en problemas específicos y los beneficios que aportan no resultan significativos debido a que no se ha tomado en cuenta el hecho de que el software es resultado de la acumulación del conocimiento poseído por las personas que forman las organizaciones, dejando de lado que la crisis del software se puede mitigar disminuyendo la brecha de conocimiento existente entre lo que saben las personas que forman una organización y el conocimiento contenido en el software, por lo cual se requiere un modelo que permita integrar de forma eficiente la mayor cantidad del *know-how* de las organizaciones, manejando el proceso de desarrollo de software desde una perspectiva de la administración del conocimiento.

Acorde a los resultados del presentados por Mochi Alemán (2006), la industria del software en México se encuentra centrada fundamentalmente en la producción de software a la medida (adaptación de software estandarizado a las necesidades de los usuarios), por lo tanto el sector esta ligado por su propia naturaleza a las actividades de servicios, pero no prevalecen empresas especializadas en desarrollo de software para brindar tal servicio sino que la mayor parte de las necesidades de los grandes usuarios (sector público y empresas) son resueltas por el autoconsumo de los mismos usuarios, a partir de departamentos internos de software abocados a esas tareas, esta situación se puede apreciar en la Figura B1 donde se muestra que de un total de 911.65 millones de dólares invertidos para el desarrollo de software a la medida, el 82% fue implementado por áreas internas de desarrollo de software y el 18% mediante contratación de empresas especializadas, este fenómeno se puede explicar debido a que existe la tendencia de considerar que con áreas internas para la implementación de software a la medida, es posible llevar a cabo los proyectos en menor tiempo debido a que al contratar compañías externas se debe considerar tiempo de la curva de aprendizaje necesaria para que se involucren con las reglas de negocio propias de cada empresa.

Figura B1. Desarrollo interno y externo de software a la medida en empresas mexicanas (Cifras en miles de dólares).



Fuente: Elaboración propia a partir de datos de Mochi Alemán (2006).

Dada esta realidad del país, el modelo que se pretende diseñar estará orientado al desarrollo de software a la medida, no sólo por la necesidad de administrar el conocimiento relacionado con el mismo, sino también por que es en el que se invierte más en México. En cuanto al mercado de las empresas el modelo no tendrá una orientación particular, la restricción radica en que sean organizaciones que cuenten áreas internas para el desarrollo de software, o bien sean compañías que se dediquen completamente al desarrollo de software, lo anterior también tomando en cuenta que los principales mercados a los que se destina la implementación de software a la medida en México según lo indicado por Mochi Alemán (2006), son muy diversos entre sí como se puede apreciar en la tabla A2.1.

Tabla B1: Distribución de mercados que implementan software a la medida en México.

Mercados	%
Manufactura/ Extracción	26.8
Informática y telecomunicaciones	19.8
Seguros y servicios financieros	19.8
Comercio / Distribución	15.0
Gobierno	7.9
Otros servicios	7.7
Otros mercados	3.0

Fuente: Mochi Alemán, P. O. (2006). *La industria del software en México en el contexto internacional y latinoamericano.* Cuernavaca: Centro Regional de Investigaciones Multidisciplinarias, UNAM.

Por lo tanto no se considera orientar el modelo a un mercado específico además de que el proceso de ingeniería se software y administración del conocimiento pueden ser estructurados de manera independiente al giro de las empresas donde sean aplicados.

Respecto al tamaño de las empresas en las que el modelo será aplicable se están considerando organizaciones medinas y grandes, para las cuales el impacto de no

administrar el conocimiento embebido en el software resulta mayor, pero no implica un impedimento para ser utilizado en organizaciones micro o pequeñas realizando los ajustes para simplificarlo de forma que sea funcional para este tipo de empresas.

Objetivo

Diseñar un modelo con los elementos necesarios para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento.

Hipótesis de trabajo

Es posible diseñar un modelo con los elementos necesarios para ejecutar el proceso de ingeniería de software con base en directrices de administración del conocimiento

Antecedentes

Algunos trabajos previos que han tratado el tema del software y/o su relación con la administración del conocimiento son los siguientes:

- García Garibay (2010) realizó una medición de la existencia de procesos de administración del conocimiento y el tipo tecnologías Web 2.0 en Pymes de la industria de TIC pertenecientes al clúster ProSoftware de la ciudad de México.
- Zapata Cant, Rialp I Criado y Rialp I Criado (2007) presentaron una investigación que examina los procesos de generación y transferencia del conocimiento, en las pymes del sector de las tecnologías de la información con base en un modelo conceptual diseñado a partir de un estudio de casos y la validación mediante un sistema de ecuaciones estructurales.
- Salem Ben y Mouna Ben (2009) elaboraron una estructura conceptual que proporciona la definición de conocimiento basada en arquitectura de sistemas de información y describen un proceso de desarrollo de software orientado al conocimiento que ayude a las organizaciones a reducir los problemas del conocimiento asociado a sistemas de software.
- Tor, Tore y Torgeir (2010) exploraron los beneficios y retos para mejorar la redundancia de conocimiento mediante la rotación de puestos con desarrolladores de software.
- Barreto Nunes y Bessa Albuquerque (2010) desarrollaron un trabajo en el que resumen la aplicación de un proceso para certificar la seguridad del software y realizan una propuesta para administrar el conocimiento generado durante este proceso.

- Ming (2009) llevó a cabo un estudio de caso de la manera como Lenovo realiza su proceso de administración del conocimiento en el desarrollo de software exponiendo los éxitos y las fallas a la que se han enfrentado.

- Mochi Alemán (2006) realizo un estudio de la industria del software en México en el contexto internacional y latinoamericano, en que el presenta el panorama del software a nivel mundial puntualizando algunas características relevantes del sector y contrastándolas con las que prevalecen en México.

Marco teórico

¿Qué es el software?

En la actualidad el software esta presente de una u otra manera en las actividades que realizamos diariamente, hoy en día además de ser empleado para computadoras personales o laptops, esta embebido en una gran gama de dispositivos que van desde reproductores mp3, hasta automóviles y grandes maquinarias. La mayoría de las personas que interactúan con las TIC's tienen alguna noción aproximada al concepto de lo que es el software, algunas sólo por interacción con el mismo y otras con bases teóricas, si bien las definiciones formales pueden tener alguna variación entre sí, la mayoría se fundamenta bajo ciertas premisas principales, a continuación se presenta la definición establecida por IEEE (1990):

> *"Software: Programas de computadora, procedimientos y la posible documentación asociada a los mismos así como los datos que pertenecen a la operación de un sistema informático"*

Por otro lado Salem Ben y Mouna Ben (2009) mencionan que los artefactos de software representan la acumulación del conocimiento organizacional, es decir el *know-how*, lo cual resalta la necesidad de manejar los procesos de desarrollo con una perspectiva de administración del conocimiento, considerando que alrededor del software hay dos clases de conocimiento: el conocimiento del negocio o *know-how* y el conocimiento relacionado con el software por si mimo como la posible documentación asociada, mencionada en la definición de IEEE (1990) .

Tipo de software

Mochi Alemán (2006) presenta la siguiente lista del tipo de software:

- *Sistemas operativos:* No se trata sólo de un programa, sino de un conjunto integrado de programas con varios componentes, metafóricamente es como la estructura física de una casa. Entre los sistemas operativos más conocidos podemos citar a CP/M, el sistema DOS, UNIX, Windows, MAC OS, OS/2.

- *Software aplicativos o productos empaquetados de mercado masivo:* Son los programas que corresponden a nuestras necesidades e intereses, se conocen como productos empaquetados y se dirigen al mercado masivo, ya que son vendidos en paquetes confeccionados, de manera estándar; generalmente su uso es más fácil y viene acompañado de manuales que explican todas sus funciones. Siguiendo con el ejemplo metafórico, el software aplicativo provee los componentes que hacen la casa habitable, respondiendo a las necesidades de cada usuario.

- *Soluciones empresariales o software desarrollado a medida, servicios informáticos:* Este tipo de programas exigen, de acuerdo con su complejidad, algún grado de personalización o adaptación a los requerimientos específicos de la organización en la cual van a ser implementados.

- *Software embarcado o embebido:* Viene incorporado en distintos tipos de maquinaria, equipos y dispositivos de consumo, es habitual que sea desarrollado *in house* por los propios productores de los bienes en los cuales se incorpora.

- *Software de servicios:* Incluye desarrollo de software a distancia, administración remota de aplicaciones, desarrollo y mantenimiento de aplicaciones y, en fases más avanzadas, administración integral de todo el departamento de sistemas de una empresa.

Ingeniería de Software

En relación a este tema la IEEE (1990) presenta esta definición:

> *"Ingeniería de software: Enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software."*

Varios autores, entre ellos Rusu, Russell, Robinson y Rusu (2010), complementan la definición de ingeniera de software desglosándola en las siguientes fases:

- *Fase de requerimientos:* Corresponde a las actividades que permiten especificar de la forma más detallada posible los requerimientos de los usuarios, de manera que las solicitudes no sean incompletas, ambiguas o contradictorias. Una vez que se cuentan con los elementos suficientes para realizar un análisis de los requerimientos

se evalúan los mismos y se definen los que pueden ser alcanzables acorde a los recursos tanto técnicos como humanos disponibles.

- *Fase de diseño:* En esta etapa se diseña la solución que cubra el alcance de los requerimientos definido en la fase anterior, la cual incluye estructurar una arquitectura para los sistemas de software o bien modificaciones a una existente, prototipos de las interfaces gráficas con las interactuará el usuario, mapeo de procesos, flujo de datos, diagramas e inclusive selección del lenguaje de programación.

- *Fase de implementación:* Consiste en *traducir* el diseño especificado anteriormente a un lenguaje de programación, de forma que se obtenga el producto de software que cumpla con la definición del alcance.

- *Fase de pruebas:* Una vez codificado el software se debe validar su adecuada funcionalidad, las pruebas se pueden dividir en las siguientes:

- Pruebas unitarias: La complejidad de los sistemas de software actuales implica que sean implementados por varias personas, por lo cual cada una de ellas debe asegurarse que la *porción* de construcción de código que le fue asignado trabaje correctamente.

- Pruebas integrales: El conjunto de programadores que participaron en la implementación del software deben asegurarse que el código construido sea compatible y cumpla adecamente su función dentro de las interacciones requeridas en el sistema de software.

- Pruebas en ambiente de quality assurance: Los dos tipos de pruebas mencionadas anteriormente son ejecutadas en un ambiente creado para desarrollar software, donde los desarrolladores suelen tener el control del mismo por lo cual pueden presentarse situaciones donde el software no sea validado en algún escenario al que tenga que responder en condiciones de uso real, por este motivo es requerido contar con un ambiente creado para ejecutar pruebas de software para lo cual es necesario migrar los componentes construidos por los programadores a este ambiente lo cual representa un *ensayo* del procedimiento a seguir cuando se traslade a un ambiente operativo. En este ambiente se emula el comportamiento operativo del nuevo software para garantizar que el sistema se comporte adecuadamente.

- *Fase de liberación:* En caso de que sea detectada alguna falla en la fase de pruebas se debe volver a la fase de implementación para corregirla y llevar a cabo de nuevo las validaciones necesarias hasta que sea posible certificar que el nuevo software cumple con el alcance definido y entonces se procederá a migrarlo al ambiente productivo para que los usuarios tengan acceso al mismo.

- *Fase de mantenimiento:* Una vez que el software es colocado en un ambiente operativo puede darse el caso de que se presenten errores no detectados en la fase de pruebas o bien que se deban realizar modificaciones para atender nuevos requerimientos en cuyo caso se repetirían las fases tantas veces como sea solicitado por los usuarios.

Datos

Son un conjunto discreto de hechos objetivos acerca de eventos. En el contexto empresarial pueden ser descritos como registros estructurados o transacciones. Carecen de sentido, porque describen sólo parcialmente lo que sucede y no proporcionan juicio ni interpretación, tampoco permiten la toma de decisiones. Los datos en bruto no dicen lo que se tiene que hacer (Valhondo, 2003).

Realizando una analogía los datos serían tabiques apilados en bloques sin una secuencia o fin determinado, listos para ser empleados en las paredes que formarán una casa.

Información

Drucker (2000) considera la información como datos dotados de relevancia y propósito y que a su vez otros investigadores la describen como mensaje, normalmente en forma de documento o comunicación visible o audible, así pues el mensaje debe informar.

A diferencia de los datos, la información tiene sentido, no sólo tiene el potencial de modelar al receptor, sino que en sí misma tiene forma, está organizada con algún propósito. Los datos se convierten en información cuando estos son:

- Contextualizados: Se sabe para qué propósito fueron recolectados.
- Categorizados: Se conocen las unidades de análisis o los componentes clave de los datos.
- Calculados: Los datos han sido analizados matemática o estadísticamente.
- Corregidos: Se han eliminados datos erróneos.
- Condensados: Los datos han sido resumidos, es decir, son más concisos.

Siguiendo con la analogía, la información representa una serie de paredes con un tamaño, forma y propósito establecido, construidas con los tabiques (datos) acomodados en una secuencia definida que constituye la estructura de una casa, estos tabiques fueron elegidos de manera que se eliminaron aquellos que no cumplen con las características necesarias para formar parte de las paredes o bien fueron modificados para que se ajustaran al diseño requerido.

Conocimiento

El conocimiento es un tema un tanto complejo el cual ha sido estudiado a través de la historia del hombre, por ejemplo los antiguos griegos se plantearon una discusión en cuanto a la estructura del mismo y la manera como es creado. Y por otro lado puede ser visto o estudiado desde diferentes perspectivas y contextos como por ejemplo los procesos internos del cerebro involucrados con el conocimiento o las implicaciones psicológicas que se tienen, para mi tesis me interesa el conocimiento desde un enfoque empresarial.

Davenport y Laurence (1998) mencionan que el conocimiento es una mezcla fluida de experiencias, valores, información contextual y apreciaciones expertas que proporcionan un marco para su evaluación e incorporación de nuevas experiencias e información. Se origina y aplica en las mentes de los conocedores. En las empresas está presente no sólo en los documentos y bases de datos, sino también en las rutinas organizacionales, en los procesos, prácticas y normas.

El conocimiento se deriva de la información, cómo esta se deriva de los datos mediante las siguientes acciones que tienen lugar en las mentes de las personas:

- Comparación: Cómo se ajusta la información en la situación dada, comparada con otras situaciones conocidas.
- Consecuencias: ¿Qué implicaciones tiene la información para la toma de decisiones y acción?
- Conexiones: ¿Cómo se relacionan los fragmentos de conocimiento?
- Conversación: ¿Qué piensan otras personas acerca de la información?

Para finalizar la analogía el conocimiento sería la casa formada de las paredes (infomación), las cuales en conjunto definen los espacios que dan lugar a las habitaciones con cierta función, aunando a que las personas infieren su uso en base a la experiencia reconociendo los elementos que contienen cada una para establecer si es la concina, comedor, sala, etc.

Conocimiento explicito

Es el conocimiento almacenado en medios físicos, codificado formalmente en bases de datos, documentos, correos electrónicos, esquemas, webs, etc. (Valhondo, 2003).

Conocimiento tácito

Este tipo de conocimiento es personal, almacenado en las mentes de los individuos, difícil de formalizar, registrar y articular, se desarrolla mediante un proceso de prueba y error que va conformando el conocimiento del individuo sobre diversos campos (Valhondo, 2003).

Es adquirido por las personas a través del tiempo mediante la combinación de la experiencia, conocimientos explícitos y la interacción con otras personas.

Administración del conocimiento

Se puede decir que en la actualidad hay tantas definiciones de la administración del conocimiento, como la cantidad de personas que han abordado el tema, es por ello que a continuación se presenta una recopilación con base en lo expuesto por Valhondo (2003) de la forma como conceptualizan algunos autores la administración del conocimiento, con la intención de resaltar los elementos comunes y más representativos de cada una de ellas:

- *Domingo Valhondo:* La administración del conocimiento consiste en las diligencias relacionadas con el conocimiento, conducentes al logro de un negocio.
- *Gene Meieran:* La administración del conocimiento tiene que ver con el uso de las computadoras y comunicaciones para ayudar a la gente a recopilar y aplicar sus datos, información, conocimiento y sabiduría colectivos con el fin de tomar las mejores, más rápidas y efectivas decisiones.
- *Matthias Bellmann:* La administración del conocimiento es la transformación del conocimiento en negocios, aprendiendo mediante la transformación de información en conocimiento.
- *Karl Eric Sveiby*: La administración del conocimiento es el arte de crear valor mediante el afianzamiento de los activos intangibles. Por lo cual hay que ser capaz de visualizar la organización como algo que no es más que conocimiento y flujos de mismo.
- *Charles Armstrong:* La administración del conocimiento, tiene que ver con elevar la conductividad de la organización para mejorar la capacidad de enlazar con el mundo exterior y los clientes. Esto requiere crear el lugar, el tiempo y el ambiente apropiado para promover trabajo reflexivo y la efectividad de interacciones.
- *Robert K. Logan:* La administración del conocimiento está relacionada con el uso de la información estratégica para conseguir objetivos de negocio. Es la actividad organizacional de creación del entorno social e infraestructura para que el conocimiento pueda ser accedido, compartido y creado.
- *Gartner Group:* La administración del conocimiento es una disciplina que promueve el enfoque integrado de la creación, compartición y aplicación de

información en una empresa. En lo que se refiere a la compartición incluye los procesos de captura, organización y acceso.

- *Bill Gates:* La administración del conocimiento no es más que gestionar los flujos de la información y llevar la correcta a las personas que la necesitan de manera que sea posible hacer algo con ella.

A partir de la atención que enfocó el sector empresarial a la administración del cocimiento es que el tema cobró fuerza e importancia, por lo cual es posible apreciar que la mayoría de las definiciones anteriormente expuestas insinúan que el hecho de administrar el conocimiento tiene el fin de crear valor o bien generar negocios exitosos en las empresas. Por otro lado en lo referente al proceso del conocimiento resalta la importancia que se da al "flujo" del mismo así como a la información de forma que se encuentren disponibles para las personas que pueden transfórmalos y crear valor, lo cual tiene la consecuencia de aportarle ventajas competitivas a las empresas, situación que invariablemente lleva a tener mayor probabilidad de éxito en los negocios. Y finalmente otro punto de recurrencia en las definiciones es que se enuncian procesos de creación, acceso, compartición y utilización del conocimiento. Con base en lo anterior los elementos que considero clave de la administración del conocimiento son:

- Perseguir la finalidad principal de aportar ventajas competitivas a las organizaciones, para aumentar el éxito en los negocios.
- Atender los procesos para crear, acceder, compartir y utilizar el conocimiento.
- Asegurar que el flujo de información y conocimiento llegue hasta las personas que pueden crear valor para las empresas.

Procesos del conocimiento

Según Valhondo (2003) el conjunto de procesos/metaprocesos del conocimiento tienen una interrelación espacial y temporal que no está dominada por alguno en particular, es decir, la interdependencia entre los procesos es múltiple y cruzada, excluye la creación y el aprendizaje como procesos básicos ya que los considera como metaprocesos cuyas características generales son las siguientes:

- *Descubrir*: Este proceso implica identificar las fuentes de conocimiento para las organizaciones, tanto internas como externas así como establecer las herramientas necesarias para que sea extraído. Las fuentes van desde bases de datos, documentos y procesos organizacionales hasta las personas que forman la empresa por medio del conocimiento tácito.
- *Capturar:* Una vez localizado el conocimiento es preciso evaluar su utilidad y determinar de qué clase de conocimiento se trata ya que estos puntos determinarán la viabilidad y estrategia de captura para alimentar: bases de datos, directorios de expertos, repositorio de conocimientos, data warehouse o procesos.

- *Clasificar y almacenar:* La clasificación es un proceso interpretativo es influido por los criterios de la persona que clasifica por lo cual es imperante que las organizaciones especifiquen las reglas bajo la cuales será clasificado el conocimiento antes de almacenarlo, y es precisamente esta clasificación que distingue la captura del almacenamiento ya que la captura solo implica codificar para que en el momento de que sea clasificado se ubique donde aporte valor para la organización.
- *Distribuir / Diseminar:* Hoy en día se requiere lidiar con la sobrecarga de información y si no es manejada de manera adecuada puede resultar un problema que cause los mismos efectos negativos que la carencia de la misma, por lo cual se debe establecer si el conocimiento será "arrojado" hacia las personas o simplemente se colocará en alguna herramienta tecnológica para que esté disponible en el momento que sea necesitado.
- *Compartir / Colaborar:* Como se ha mencionado anteriormente el conocimiento se conforma de información estructurada en combinación con la experiencia de las personas y por lo tanto hay que fomentar que por iniciativa propia los empleados de una organización codifiquen esta experiencia para que esté disponible para otros, además de colaborar con la socialización, externalización, internalización y combinación de conocimiento mediante la interacción adecuada con demás personas.
- *Utilizar (innovación):* El objetivo de los procesos del conocimiento es que sea utilizados para conducir a innovaciones que permitan crecer a las organizaciones diferenciando sus productos y/o servicios agregando valor para sus clientes.

Metodología

La clasificación del tipo de estudio queda de la siguiente forma:

- *No Experimental*: ya que no se lleva a cabo algún proceso para manipular deliberadamente alguna variable.
- *Exploratorio*: por que se investiga una solución desde una perspectiva poco estudiada en el proceso de desarrollo de software.
- *De corte transversal*: por que la información será recolectada sólo una vez en un momento dado sin pretender evaluar.
- *Cualitativo*: Los ajustes el análisis del modelo será llevado a cabo desde una perspectiva cualitativa para la recolección de información.

Las etapas consideradas para la investigación son:

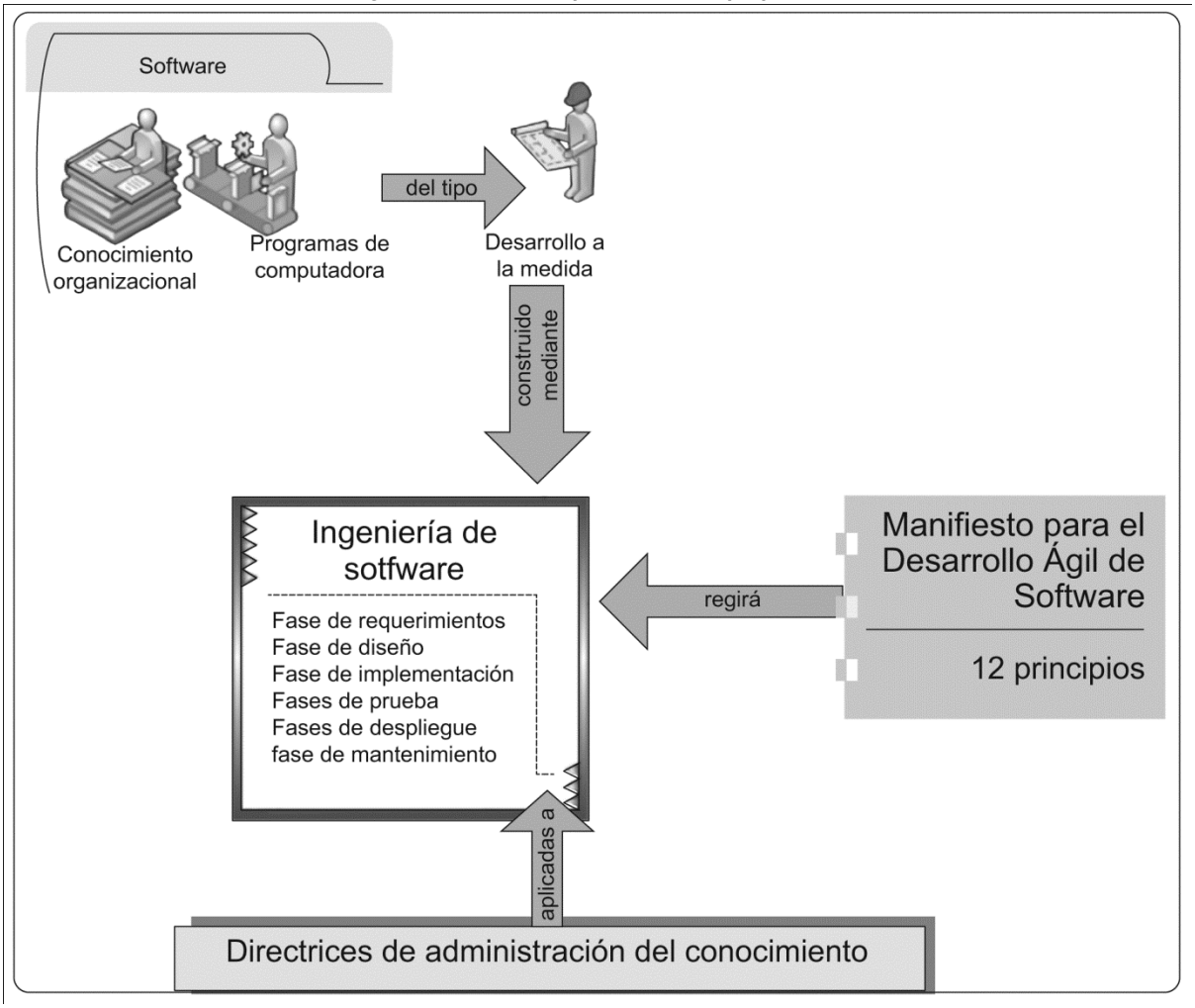
1. *Identificación del problema*: Definición de un problema actual, cuya relevancia aporte la solución de una situación importante para las organizaciones.

2. *Análisis bibliométrico*: Aplicación de métodos matemáticos y estadísticos para el análisis de publicaciones científicas.
3. *Investigación documental:* Búsqueda de referencias relacionadas con el problema identificado, ya sea mediante tesis, artículos o estudios, así como teorías y modelos para la resolución del problema.
4. *Formulación de hipótesis:* Para cada uno de los objetivos establecidos se planteará una hipótesis de su cumplimiento.
5. *Elaboración del marco teórico*: Establecimiento del contexto teórico y conceptual bajo el cual será establecida la propuesta modelo de desarrollo de software aplicando administración del conocimiento
6. *Diseño del modelo:* Elaboración una propuesta de modelo para llevar a cabo el desarrollo de software con una perspectiva de administración del conocimiento.
7. *Evaluación del modelo por expertos:* El modelo propuesto será presentado a 2 personas con 5 o más años experiencia en la industria de desarrollo de software en puestos gerenciales o directivos y 2 desarrollares de software con experiencia de 10 o más años, teniendo la finalidad de recibir retroalimentación del modelo.
8. *Ajustes al modelo:* Con base en los comentarios recibidos por los expertos se llevaran a cabo los ajustes pertinentes al modelo propuesto.

Resultados preliminares.

Con base en lo desarrollado al momento en cuanto a los temas de ingeniería de software y administración del conocimiento, puedo inferir que una alternativa para solucionar los problemas relacionados con la implementación de software, consiste en considerar como elemento principal que dentro del software está embebido el conocimiento organizacional, presentándose esta situación con mayor importancia en el caso de que es implementado a la medida, el cual al ser elaborado mediante la ingeniería de software, se necesita aplicar directrices de administración del conocimiento a cada una de sus fases de forma que sea posible tener el manejo adecuado del conocimiento que se encuentra alrededor del software, estableciendo las conexiones con el "*know how*" o conocimiento de negocio y por lo tanto al tener un adecuado manejo del mismo será posible controlar las demoras en la entrega de proyectos así como lograr que se cumplan los presupuestos planeados y que tanto la calidad como la funcionalidad sean las adecuadas para que los usuarios lleven a cabo sus tareas diarias e inclusive se mejore su ejecución y sea posible aportarle nuevas ventajas competitivas a las empresas, lo anterior se ejemplifica gráficamente en la Figura B2.

Figura B2. Elementos relacionados con la ingeniera de software para el modelo propuesto.

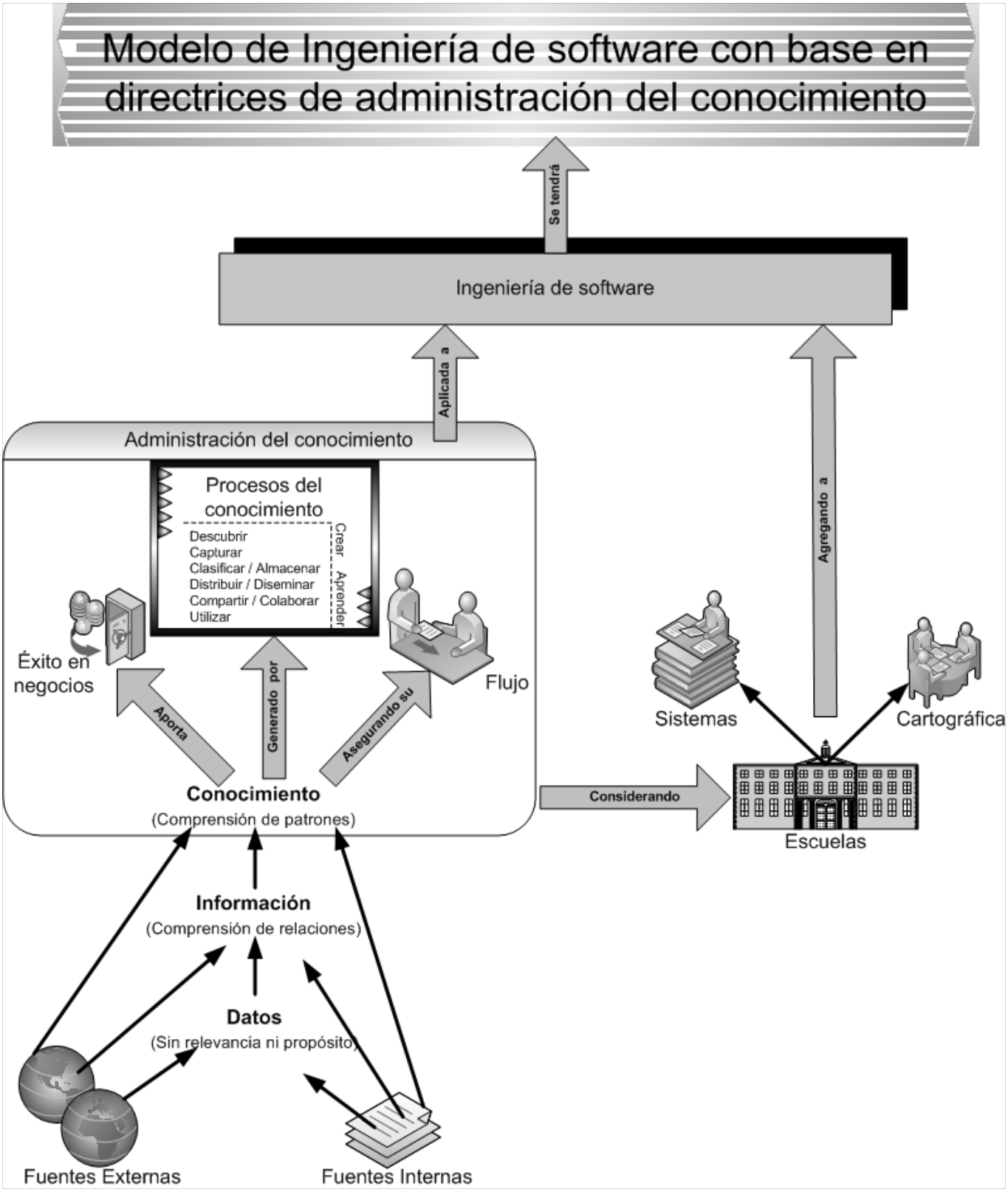


Fuente: Elaboración propia.

Otro elemento que será considerado es el manifiesto para el desarrollo ágil de software el cual se fundamenta en 12 principios, los cuales de forma general conducen a priorizar la interacción entre las personas involucradas en las fases de ingeniería se software así como la entrega de software útil para los usuarios en los periodos de tiempo más pequeños posibles.

La ejemplificación de las directrices del conocimiento que se deben aplicar a la ingeniería de software se presenta gráficamente en la Figura B3.

Figura B3. Directrices de administración del conocimiento.



Fuente: Elaboración propia.

Bibliografía

Barreto Nunes, F. J., & Bessa Albuquerque, A. (2010). A Secure Software Development Supported by Knowledge Management. *International Joint Conference on Computer, Information, Systems* (págs. 291-296). Berlin, Alemania: Springer Science+Business Media B.V.

Davenport, T., & Laurence , P. (1998). *Working knowledge: how organizations manage what they know.* Boston: Harvard Business Pres.

Drucker, P. (2000). *El management del Siglo XX.* Barcelona: EDHASA.

García Garibay, S. (2010). *Tecnologías Web 2.0 para administrar el conocimiento de la PyME mexicana.* México D.F.: Tesis de maestría, Universidad Nacional Autónma de México.

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology*. New York.

Ming, C. (2009). Research on Knowledge Management of Software Enterprises. *2nd IEEE International Conference on Computer Science and Information* (págs. 291-294). New York: IEEE.

Mochi Alemán, P. O. (2006). *La industria del software en México en el contexto internacional y latinoamericano.* Cuernavaca: Centro Regional de Investigaciones Multidisciplinarias, UNAM.

Rusu, A., Russell, R., Robinson, J., & Rusu, A. (2010). Learning Software Engineering Basic Concepts using a Five-Phase Game. *40th ASEE/IEEE Frontiers in Education Conference* (págs. S2D-1 - S2D-6). IEEE.

Salem Ben, D. D., & Mouna Ben, C. (2009). The Knowledge-Gap Reduction in Software. *The IEEE International Conference on Researh Challenges in Informational Sciene.* New York: IEEE.

Tor, E. F., Tore, D., & Torgeir, D. (2010). Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work. *Information and Software Technology*, 1118–1132.

Valhondo, D. (2003). *Gestión del Conocimiento del mito a la realidad.* Madrid: Díaz de Santos.

Zapata Cant, L., Rialp I Criado, J., & Rialp I Criado, Á. (2007). La generación y la transferencia de conocimiento en PyMES del sector de las tecnologías de la información. *XI congreso de investigación en ciencias administrativas ACACIA.* Guadalajara, Jalisco: Academia de Ciencias Administrativas, A. C.

Apéndice C, Guión empleado en la entrevista.

1. Preguntar al informante si tiene alguna duda respecto a los elementos teóricos del modelo.
    a. En caso afirmativo resolver las dudas al respecto.
2. Consultar al informante su opinión de que se conceptualice el software desde una perspectiva en la que se reconozca que contiene conocimiento embebido.
3. Saber si las directrices de administración del conocimiento que son aplicadas a la ingeniería de software son claras y compresibles para el informante.
4. Averiguar si fue complejo para el informante comprender el modelo propuesto.
    a. En caso afirmativo investigar los motivos por que se dificultó su compresión.
5. Indagar las fortalezas y debilidades que tiene el modelo propuesto desde el punto de vista del informante.
6. Preguntar sobre las mejoras que el informante considera necesarias para el modelo.
7. Consultar al informante si considera aplicable el modelo a su entorno laboral actual.
8. Saber si el informante se sentiría cómodo trabajando bajo el modelo propuesto en su vida profesional y conocer los motivos de su respuesta.
9. Recapitular los elementos tratados en la entrevista para garantizar que se hayan comprendido completamente las ideas del informante.