REDISEÑO AL PARADIGMA DE LA INGENIERÍA DE SOFTWARE: EL CASO DEL SOFTWARE LIBRE

Ramón Marín Solís, Agustín Gutiérrez Tornés,
Miguel Ángel Mora Espinosa[1]
Centro de Investigación en Computación, SEPI-ESIA Tecamachalco[1]
Instituto Politécnico Nacional
UPALM Av. Juan de Dios Bátiz y Av. De las Torres
Col. Nueva Industrial Vallejo. México, DF CP 07738
Teléfono: 57296000 Ext. 56554.
e-mail: rmarin [EN]ipn.mx, atornes[EN]cic.ipn.mx, mora[EN]itesm.mx

RESUMEN
La convergencia entre las distintas disciplinas que atienden la problemática a la que se enfrentan las personas involucradas en el desarrollo de sistemas de información y software, es cada día más notoria. La ingeniería de software se ha constituido como una disciplina fundamental en este ámbito, pero el paradigma que ha desarrollado se encuentra rebasado por la *complejidad*, el *caos* y la *crisis*. Además se encuentra enfocada, casi en su totalidad, en aspectos estructurales (tecnológicos y organizacionales).

El movimiento del software libre propone desarrollar una sociedad ética, el presente trabajo muestra distintos conceptos que pueden aplicarse dentro del software libre para lograrlo, como el uso de la hermenéutica analógica para establecer un modelo de comunicación entre los distintos participantes en el proceso de desarrollo de software y los aspectos éticos del diseño tecnológico.

INTRODUCCIÓN

La convergencia entre la ingeniería de software y la ingeniería de sistemas es, cada vez, más evidente. Por ejemplo, ha llevado al Comité de estándares de ingeniería de software de la IEEE (SESC- Software Engineering Standards Committee) a ampliar sus objetivos y cambiar su nombre al "Comité de estándares de ingeniería de software y sistemas" (S2ESC- IEEE Systems and Software Engineering Standards Committee) para incluir la ingeniería de sistemas cuando se usa software intensivamente [1]. Y están buscando la incorporación, a este Comité, de la organización de profesionales de la ingeniería de software INCOSE (International Council on Systems Engineering – Concejo Internacional de Ingeniería de Sistemas) y de la AFEI (Association For Enterprise Integration – Asociación para la Integración Empresarial) [2].

La misión del S2ESC [3] es:

"-Desarrollar y mantener una familia de estándares de ingeniería de software y sistemas, que sea relevante, coherente, comprensible y efectiva en su uso. Esos estándares serán para el uso de profesionales, organizaciones y educadores y tienen como finalidad: mejorar la efectividad y la eficiencia de sus procesos de ingeniería de software, mejorar las comunicaciones entre los clientes y proveedores, y mejorar la calidad del software y los sistemas que contienen software.
-Desarrollar conocimiento que ayude a los profesionales, organizaciones y educadores en la comprensión y aplicación de estos estándares.
-Soportar y promover: un cuerpo de conocimiento (*Body of Knowledge*) de Ingeniería de Software, así como, mecanismos de certificación para los profesionales de la ingeniería de software."

Destaca dentro de la misión, el integrar la ingeniería de software y de sistemas, así como, mejorar la comunicación entre los clientes y proveedores.

EL PARADIGMA DE LA INGENIERÍA DE SOFTWARE

La ingeniería de software es:"La aplicación de un enfoque sistemático, disciplinado y

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005*. Febrero. México

cuantificable[1] para el desarrollo, operación y mantenimiento del software, esto es, la aplicación de la ingeniería al software; así como el estudio de esos enfoques" [4]
Según Sommerville [5] es: "una disciplina que comprende **todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema**[2], hasta el mantenimiento de éste, después de que se utiliza".
Cuando propone la forma de trabajo refiere que: "en general los ingenieros de software adoptan un enfoque **sistemático y organizado** en su trabajo, ya que es la forma más efectiva de producir **software de alta calidad**. Sin embargo, aunque **la ingeniería consiste en seleccionar el método más apropiado** para un conjunto de circunstancias, **un enfoque informal y creativo de desarrollo podría ser [más] efectivo en algunas circunstancias**. Y muestra un ejemplo de un caso: "El desarrollo informal es apropiado para el desarrollo de sistemas de comercio electrónico basados en Web que requieren una **mezcla de habilidades** de software y de diseño gráfico". [5]

Al tratar de diferenciarla de otras disciplinas, señala que, la diferencia entre Ingeniería de Software y Ciencias de la Computación es que esta última "se refiere a las **teorías y métodos** subyacentes a las computadoras y los sistemas de software, mientras que la Ingeniería de Software se refiere a los **problemas prácticos** de producir software [de alta calidad]... los ingenieros de software a menudo utilizan enfoques *ad hoc* para desarrollar el software. Las elegantes teorías de la Ciencia de la Computación no siempre pueden aplicarse a **problemas reales y complejos** que requieren [de] una solución de software". [5]

Cuando intenta establecer la diferencia entre la Ingeniería de Software y la Ingeniería de Sistemas, confunde a esta última con una de sus áreas de estudio –los Sistemas de Información- dentro de la cual se incrusta la Ingeniería de Sistemas Basados en Computadoras, que "se refiere a todos los aspectos del desarrollo y de la evolución de sistemas complejos donde el software juega un papel principal. Por lo tanto la Ingeniería de Sistemas [Basados en Computadoras] comprende el desarrollo de hardware, políticas y procesos de diseño y distribución de sistemas, así como la Ingeniería de Software. Los ingenieros de sistemas están involucrados en la especificación del sistema, en la definición de su arquitectura y en la integración de las diferentes partes para crear el sistema final. Están menos relacionados con la ingeniería de los componentes del sistema (hardware, software, etc.). La Ingeniería de Sistemas es más antigua que la de software, por más de 100 años, las personas han especificado y construido sistemas industriales complejos, como trenes y plantas químicas. Sin embargo, puesto que se ha incrementado el porcentaje de software en los sistemas, las técnicas de la ingeniería de software se utilizan en el proceso de Ingeniería de Sistemas [Basados en Computadoras]" [5].

LA PROBLEMÁTICA EN LA INGENIERÍA DE SOFTWARE

Como demuestran los hechos, el objetivo de producir software de alta calidad está lejos de cumplirse. Tepper [6] cita a un reportero de tecnología que en 1997 escribió: 'el software se hace muy mal, a menudo esta terriblemente mal hecho. Es desarrollado de una forma despreocupada e irresponsable, que sería inmoral si se aplicara a la construcción de puentes, automóviles y quizás hasta en trabajos de plomería'. Y señala que cinco años más tarde (2002) la situación en lugar de mejorar ha empeorado. Dentro de las características que hacen que el desarrollo de software sea deficiente destaca: la creciente complejidad del mismo; que los desarrolladores de software no consideraren las necesidades del usuario promedio (ponerse en el lugar del otro) ya que para el usuario final el software es sólo un medio para desarrollar una tarea y no un fin en sí, el desdén por los usuarios llega hasta el punto de considerarlos estúpidos, perjudicándolos ya que obtienen software repleto de características que no usan y no tienen las que necesitan.

Por su parte Ambler [7] es más tajante, y nos dice que "dependiendo de la fuente, la industria de Tecnologías de la Información tiene un rango de fracaso en proyectos de

[1] Subrayado añadido
[2] Énfasis y subrayado añadidos. En lo sucesivo, el énfasis y subrayado añadido, conservará el mismo estilo.

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005*. Febrero. México

misión crítica del 65% al 85%. Y aún el panorama más alentador, del 65%, que publica el Standish Group en su reporte del caos, es aterrador. Lo más frustrante es que el panorama no ha mejorado con el transcurso de los años, sino que ha empeorado. Y esto es una clara evidencia de que **el paradigma tradicional del desarrollo de software no funciona**".

En cuanto a la forma en la que se desarrolla el software, Fowler [8], remarca que "la mayoría del desarrollo de software es una **actividad caótica**, caracterizada a menudo por la frase 'codifica y arregla'. El software se escribe sin basarse en un plan, y el diseño del sistema es improvisado de decisiones en el corto plazo, lo cual funciona si el sistema es pequeño, pero conforme el sistema crece la dificultad para agregar nuevas características al mismo. Además los errores se incrementan y son más difíciles de arreglar".

Donaldson y Siegel [9] apuntan, que la "'forma' en la que la industria de software desarrolla sus productos debe cambiarse, esta afirmación no es sólo una opinión, ya que es un punto de vista que comparten muchos en la industria. Las siguientes solicitudes y reclamos, son ejemplos que hacen eco de esta necesidad de cambio:

> Entrega el software a tiempo y dentro del presupuesto
> Deja de hacer solicitudes de ultimo minuto para nuevas características [del software]
> Haz que el software haga lo que pedí
> Ayúdame –**define lo que realmente quieres**
> Deja de darme soluciones a corto plazo para problemas de largo plazo
> Deja de aplazar la fecha de entrega
> Proporciónanos los recursos para una prueba adecuada
> Establece una arquitectura en la cual podamos trabajar
> Reduce la dependencia de los individuos en nuestra organización
> Desarrolla nuevos sistemas con costos más bajos
> Proporciona los recursos para implementar una nueva forma de hacer negocios

Los ejemplos anteriores suenan como partes de diálogos cliente-desarrollador o de diálogos dentro de una organización que se dedica al negocio del desarrollo de software.

Los clientes quieren que el software satisfaga ciertos criterios, quieren sistemas que: "1) hagan lo que se supone que tienen que hacer, 2) que se entreguen a tiempo, y 3) que se entreguen en el precio acordado. De la misma forma, los vendedores (por ejemplo las compañías que desarrollan software) quieren que los sistemas que desarrollan 1) hagan lo que los clientes quieren, 2) se entreguen antes de tiempo o a tiempo, y 3) generen una utilidad razonable. Creemos que el software 'bueno' [de calidad] es aquel que satisface el criterio del cliente y del vendedor, y que ambos quieren que sus criterios se satisfagan de forma *consistente* [**confiable-constante**]". [9]

El desarrollo exitoso del software significa la capacidad de producir software de calidad de forma confiable.

E indican "que la industria de software ha demostrado claramente que la **falla en la comunicación cliente-vendedor** provoca la mayoría de los problemas en el desarrollo de software" [9]

Los empresarios de la industria de desarrollo de software requieren de personal competente en el área, y dicen que "En estos días, no puedes pagar a empleados que no entiendan el negocio como un todo. En la compañía, estamos tratando de contratar desarrolladores que comprendan, que están tratando de solucionar problemas de negocios" [10]

En base a lo anterior, se hace patente que el paradigma actual de la ingeniería de software, aplicado en problemas complejos, en un entorno de constante cambio, y con participantes en el proceso de desarrollo con intereses diversos, y más aún, de distintas culturas, ha quedado rebasado. Para rediseñarlo se propone, hacerlo bajo un enfoque sistémico, lo cual requerirá de una investigación transdisciplinar.

DISCIPLINAS QUE CONCURREN EN LOS SISTEMAS DE INFORMACIÓN

"Un paradigma surge en el marco de una comunidad, de una cultura" [11] por lo que para realizar el rediseño al paradigma de la ingeniería de software se requiere un acercamiento conceptual y del lenguaje entre las disciplinas que convergen en el desarrollo de software y de sistemas de información.

INGENIERÍA DE SISTEMAS

"Concierne a la ingeniería de sistemas el ofrecer diseños de sistemas aptos para resolver las tres faenas esenciales del quehacer gerencial, a saber, *formular políticas, tomar decisiones y controlar operaciones*[3]*;* a su vez, el quehacer gerencial aludido tiene lugar en sistemas de una especie singular a los que denominaremos *organizaciones,* las que se conforman de dos categorías de componentes, la primera referente a su *estructura* y la segunda a su *cultura... Se c*onsidera que el diseño es una actividad humana primitiva y espontánea, que se desglosa en tres disciplinas: aprendizaje, liderazgo, previsión; estas tres disciplinas están en concordancia con tres dimensiones o categorías preexistentes, esto es, anteriores e independientes del hombre: la complejidad, el caos, la crisis. [12]

La ingeniería de sistemas ha pasado por distintos enfoques, de acuerdo al contexto (complejidad) y a la naturaleza (unitaria, pluralista, coercitiva) de los participantes del sistema. [13]

El Enfoque Analítico: La Era de la Máquina

"El universo era una *máquina que fue creada por Dios para realizar Su obra*. Se esperaba que el hombre, como parte de esa máquina, cumpliera con los designios de Dios, que hiciera Su voluntad. De aquí se infirió que el *hombre tenía que crear máquinas para que hicieran su trabajo*. La revolución industrial fue un producto de esa inferencia. Los hombres en esa época, confrontaron la naturaleza con un temor reverente, y con la admiración y la curiosidad de un niño, e intentaron descifrar sus misterios analíticamente." [14]

La Era de los Sistemas

"La Segunda Guerra Mundial, sacó a la ciencia y a los científicos de sus laboratorios y los metió en el 'mundo real' en un esfuerzo por resolver importantes problemas que surgían en organizaciones grandes y complejas –militares, gubernamentales y corporativas. Los científicos descubrieron que los problemas que enfrentaban no podían descomponerse en otros que encajaran exactamente en una disciplina particular y que la interacción de las soluciones particulares de las partes separadas era de mayor importancia que las soluciones consideradas por separado. Esto llevó a su vez a la creación de trabajos interdisciplinarios. A finales de los años treinta del siglo XX, surgió de la institución militar británica la 'investigación de operaciones', una actividad interdisciplinaria, dirigida a resolver los problemas en la administración y el control de sus complejas operaciones.

Para los años cincuenta proliferaban las actividades científicas interdisciplinarias. Éstas incluían las ciencias de la administración, las ciencias de las decisiones, las ciencias de la computación, las ciencias de la información, la cibernética, las ciencias políticas y muchas otras. Los intereses compartidos y las similitudes en sus prácticas llevaron a la búsqueda de un tema común a todas ellas, y éste fue el comportamiento de los *sistemas*. "[14]

LA HERMENÉUTICA ANALÓGICA Y SU CONTRIBUCIÓN EN EL ENFOQUE SISTÉMICO

La hermenéutica o el arte de interpretar textos, es una "teoría de la verdad y el método que expresa la universalización del fenómeno interpretativo desde la concreta y personal historicidad"[4].

"El hombre es un ser biológico pero también un ser simbólico o simbológico, incluso más simbológico que biológico, un ser en permanente discurrir de la univocidad (lo biológico) a la equivocidad (lo simbológico), porque en ésta reside la creación de cultura. En la pugna de cultura versus natura se requiere de una hermenéutica *analógica* porque asume la mediación de contextos permitiendo el predominio del símbolo en la cultura, sin desmedro de lo biológico."

El símbolo tiene dos componentes: el metonímico [funda el discurso científico. Nos da la relación efecto-causa; partes al todo] y el

[3]Referencia original: Beer, Stafford, Ciencia de la Dirección, Ed. Ateneo, Argentina, 1972; p. 28.

[4] Diccionario de la Real Academia de la Lengua Española.

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005* . Febrero. México

metafórico [acto de habla simbólica, nos conduce a la utopía, y funda la poesía].

Beuchot [15] expresa que "los límites de mi mundo, son los límites de mi lenguaje y a dónde no llega mi intelección llega el símbolo. La analogía es un conocimiento no completo, reducido, pero suficiente." Y apunta que "el lenguaje es una red sistematizada de metáforas".

La hermenéutica analógica nos proporciona un modelo teórico de interpretación y una metodología para ganar comprensión sobre la realidad y poder explicarla [11], lo cual es de suma importancia en cualquier proceso de ingeniería.

REDISEÑO AL PARADIGMA DE LA INGENIERÍA DE SOFTWARE

"Los Sistemas de Información están en la edad de piedra, el paradigma que manejan está fuera de toda realidad, ya que se basan en ideas sobre la organización de los 60's. En las cuales se presenta el modelo de la administración como la toma de decisiones en la búsqueda de objetivos establecidos, un modelo del que cualquiera con experiencia real en la vida organizacional sabe que es absurdo e inadecuado. Debe haber serios problemas para el campo de los Sistemas de Información en los que hay confusiones conceptuales y que está atado a un modelo caduco acerca de la naturaleza de la administración y las organizaciones... En donde se considera que la función principal de los Sistemas de Información es servir como herramientas en la toma de decisiones". [16]

La preocupación por el atraso en las metodologías para el desarrollo de sistemas, con respecto de la tecnología, se señalaba de esta forma en 1995: Con el avance tecnológico de los últimos años, la capacidad para desarrollar aplicaciones se encuentra un poco atrasada respecto a la disponibilidad de tecnología. Se cuenta con el equipo, pero no con los principios metodológicos para elegirlos y explotarlos al máximo, así mismo, las herramientas para el desarrollo de aplicaciones exigen nuevas habilidades y conocimientos para operar óptimamente. Dentro de este contexto la teoría de los sistemas cliente/servidor nos proporciona una visión clara para el máximo aprovechamiento de todos los recursos de las empresas e instituciones, ya que índica la forma en la que podemos realizar la estructura de nuestros sistemas de cómputo, su creación, integración y explotación; además de que su prioridad es la productividad de los usuarios. [17]

La preocupación por el desarrollo de los principios metodológicos para la elección y la explotación de la tecnología continúan hasta el día de hoy. Y hay que notar que "en un campo dominado por la tecnología dónde la tecnología cambia rápidamente, la teoría siempre estará detrás de la práctica". [16]

EL CAMBIO COMO GENERADOR DE CRISIS

El cambio siempre ha sido acelerado. Esto no es ninguna novedad, sin embargo hay factores que hacen que los cambios que estamos experimentando nos afecten de una forma considerable, primero:

> **Los cambios habían ocurrido con la lentitud suficiente para permitir que la gente se adaptara,** haciendo pequeños ajustes ocasionales o bien acumulando la necesidad de hacerlos y transfiriéndola a la siguiente generación.
>
> El ritmo actual es tan rápido que los retrasos para responder a él pueden ser muy costosos, incluso desastrosos. Todos los días hay compañías y gobiernos que desaparecen por que no han podido adaptarse a él o por que lo hicieron con demasiada lentitud.
>
> La adaptación a los acelerados cambios actuales, requiere **ajustes frecuentes y extensos en lo que hacemos y cómo lo hacemos**.
>
> Los seres humanos buscan la estabilidad y son miembros de grupos, organizaciones, instituciones y sociedades que buscan estabilidad. Puede decirse que su objetivo es la 'homeostásis' pero el mundo en el que se persigue ese objetivo es más dinámico e inestable.
>
> Debido a la interconexión y la interdependencia creciente entre los individuos, grupos, organizaciones, instituciones y sociedades que se derivan de los cambios de las comunicaciones y el transporte, nuestro

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005*. Febrero. México

> entorno se ha hecho más extenso, más complejo y menos predecible; en resumen se ha vuelto más turbulento. [14]

Otra característica única de los cambios que experimentamos es que:

> Conforme se incrementa el ritmo del cambio, aumenta también la complejidad de los problemas que nos confrontan. Entre mayor sea la complicación de estos problemas, más tiempo tomará resolverlos
> Entre más rápido sea el ritmo del cambio, mayores serán las variaciones de los problemas que nos confrontan y más corta será la vigencia de las soluciones que encontremos para ellos.
> Por lo tanto, para cuando se tiene la cura a muchos de los problemas que nos confrontan, generalmente los más importantes, éstos ya han cambiado tanto que sus arreglos han dejado de ser relevantes o efectivos; han nacido muertos. En otras palabras, muchas de las soluciones son para dificultades que ya no existen en la forma en que se resolvieron.
> En consecuencia, nos estamos rezagando cada vez más.

El cambio más importante que está teniendo lugar, se da en la forma en la que intentamos comprender el mundo, y en nuestra concepción de su naturaleza. [14]

Hay que tomar en cuenta, que en un mundo con cambios constantes y rápidos, la información sobre los sucesos no la obtenemos de forma instantánea (en tiempo real), ya que existe un retraso, y si la esta no se encuentra actualizada las decisiones que tomemos serán erróneas.

LAS INSUFICIENCIAS DEL PARADIGMA SISTEMÁTICO

Ackoff [18] muestra algunas ideas obvias, sobre el paradigma sistemático (ordenado, analítico y disciplinar), que son erróneas:

> Que al mejorar el rendimiento de las partes de un sistema, tomadas por separado, se mejorará necesariamente el rendimiento del todo.
> Los problemas caen naturalmente en una disciplina. La investigación efectiva no es disciplinaria, interdisciplinaria o multidisciplinaria; es transdisciplinaria.
> Lo mejor que se puede hacer con un problema es solucionarlo, falso. Lo mejor es disolverlo, rediseñar la entidad que lo tiene o su entorno, para eliminar el problema.
> La mayoría de los sistemas sociales están buscando objetivos diferentes a los que proclaman, y éstos están equivocados.

De aquí que el enfoque sistémico y transdisciplinar, aplicado en la disolución de problemas, que tome en cuenta el punto de vista de todos los participantes sea la clave para el rediseño del paradigma de la ingeniería de software.

“La relevancia del pensamiento de sistemas para el trabajo en sistemas de Información se muestra de forma retórica en los trabajos sobre el área, pero aún no hay un esfuerzo persistente para forjar el enlace entre el pensamiento de sistemas y la provisión organizada de información en las organizaciones que es el rol de los Sistemas de Información. Checkland [16] destaca la relación de cuatro disciplinas relevantes para ello: Sistemas de Información, Pensamiento de Sistemas, Tecnologías de la Información y Teoría Organizacional. “
“La principal área del campo [Sistemas de Información] debe de ser: proveer datos e información dentro de la organización usando TI, y que esa información sea relevante para las actividades siempre cambiantes de la organización y/o de sus miembros. “[16]

> “La investigación de la realidad social debe de convertirse en un descubrimiento organizado de cómo los agentes humanos *toman conciencia* de sus mundos que perciben, cuáles son sus significados compartidos, y como estas percepciones

cambian a través del tiempo y difieren de una persona o grupo a otro." [16]

Y es en esta transmisión de significados entre individuos y culturas, cuando el uso de la hermenéutica analógica cobra sentido ya que "Cuando se cambia un paradigma, tiene que hacerse conmensurable con le paradigma anterior, so pena de no ser entendido. Se ha dicho que la conmensurabilidad se da al traducir elementos de un paradigma a los de otro. Precisamente en ese trabajo de traducción es donde más se da la analogía. Hay una traducción univocista [analítica-positivista], otra equivocista [subjetiva-relativista] y otra analógica. La traducción unívoca es inalcanzable las más de las veces; la traducción equivoca es inútil; sólo queda la traducción analógica, aproximativa." [11]

ÉTICA Y ESTÉTICA

Un campo de investigación polémico, es el de la aplicación de la ética (y la estética) en la ingeniería. En su más reciente publicación Ackoff [19], aborda el tema del Rediseño de la Sociedad, y ahí expresa que "los conceptos 'correcto y 'equivocado'; 'verdadero' y 'falso' no son aplicables a un diseño, pero a uno puede gustarle o desagradarle un diseño. Así, el concepto prueba no es relevante, lo es la **preferencia.** Un diseño debe de ser **evaluado** de acuerdo a **qué lo hace posible y qué lo hace imposible**. **La evaluación de un diseño es materia de gusto**; y el gusto es materia de **estética y en lo que concierne al gusto no hay disputa**."

El papel de los ingenieros de software y de sistemas dentro de la sociedad es de una gran importancia, acuden a nosotros personas y organizaciones de todo tipo, con problemas diversos, nuestros actos deben de guiarse bajo un comportamiento ético, buscando el bien común: el de aquellos que acuden a nosotros y el bien propio. Como se revisó a lo largo del presente trabajo, la complejidad y los distintos intereses de los participantes en los proyectos de desarrollo de software requieren de habilidades nuevas. Debemos de estar atentos ya que "la Sociedad adora a sus nuevos ídolos: la productividad, la competitividad y la competencia; está limitada por tremendas desigualdades, requiere de obediencia –lo cual coarta la libertad, la dignidad y a veces la justicia, lo que nos desalienta y nos lleva a la renuncia de **rediseñar la sociedad** y darle solución a la problemática de ésta". [20]

LA INGENIERÍA DE SISTEMAS Y EL SOFTWARE LIBRE

El campo del diseño ético, es dónde el software libre cobra relevancia, ya que la finalidad última del proyecto GNU/FSF es el de construir una **sociedad ética**. [21] [22].

La ingeniería de sistemas puede colaborar en el movimiento del software libre, proporcionando conceptos, metodologías y herramientas, que conduzcan al rediseño de la sociedad, ya que se requiere una transformación radical, no sólo tecnológica. Esta transformación se debe de comunicar a los políticos y las personas con el poder de decisión. [23].

El modelo participativo del software libre es una ventaja que debemos aprovechar. Hay que elaborar mecanismos que nos permitan el desarrollo sustentable de nuestras comunidades.

CONCLUSIONES

El paradigma sistemático que se aplica en la ingeniería de software ha quedado rebasado, los cambios constantes en el entorno de las organizaciones no permiten planearlos y predecirlos. Cada vez más los usuarios demandan ser escuchados, por lo que se requiere de un lenguaje común entre todos los involucrados en el desarrollo de software. Aún con el acercamiento entre las Ingenierías de Software y de Sistemas, no se ha logrado trasladar los conceptos, modelos y metodologías claves, que nos lleven al desarrollo de sistemas viables, que logren adaptarse a los cambios constantes en el entorno, y que sean capaces de sobrevivir en escenarios de crisis. El presente trabajo es el punto de partida de una investigación transdisciplinar que busca lograr ese objetivo. Y para ello se propone emplear el paradigma sistémico en el desarrollo de software, utilizando disciplinas de la ingeniería de sistemas y de la filosofía, como la hermenéutica analógica.

La ingeniería de sistemas, cuenta con herramientas que pueden ayudar al desarrollo de la sociedad ética, propuesta por el

movimiento de software libre. La sociedad requiere una transformación radical, y no sólo tecnológica.

Al mostrar la problemática y cómo el paradigma actual no puede resolver este conjunto de problemas, se pretende iniciar el diálogo entre la ingeniería de sistemas y la de software, para cumplir con la promesa de brindarles a los usuarios productos y servicios de calidad, en tiempo y a un precio accesible, en una sociedad que respete los valores fundamentales del hombre, y en donde la cooperación sea la base para el desarrollo sustentable de nuestras comunidades.

BIBLIOGRAFÍA

1. IEEE S2ESC. 2004. *Minutes of Meeting # 106*. http://standards.computer.org/sesc/s2esc_excom/minutes/2004-02/Minutes-Onsite-MBExCom-2004Feb.htm 26 y 27 Febrero 2004 (26/ago/04)
2. IEEE S2ESC. 2004. *Minutes of Meeting # 111*. http://standards.computer.org/sesc/s2esc_excom/minutes/2004-07/Minutes-Telecon-ExCom-2004Jul.htm 6 Julio 2004. (28/ago/04)
3. IEEE S2ESC. 2004. *Background*. http://standards.computer.org/sesc/s2esc_geninfo/S2ESC-Background.doc (28/ago/04)
4. IEEE. 1990. "IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990". En *IEEE Standards Software Engineering: Volume One, Customer and Terminology Standards*. 1999 Edition. EUA:IEEE. 1999. pp.82
5. Sommerville, Ian (2000). *Ingeniería de Software*. México: Pearson Educación. 6ª edición 2002. pp. 692
6. Tepper, Michele (2002). "Why Software Still Stinks". *The Last Word*. ACM Press: New York, NY, USA. Septiembre 2002.
7. Ambler, Scott (2004). *Agile Data Home Page: Bringing data professionals and application developers together*. http://www.agiledata.org/:EUA (12/05/2004)
8. Fowler, Martin (2003). *The New Methodology*. http://www.martinfowler.com/articles/newMethodology.html:EUA. (27/08/2003)
9. Donaldson, Scott y Siegel, Stanley (2001). *Successful Software Development 2nd Edition*. EUA: Prentice Hall PTR. 2001. pp. 701
10. McLaughlin, Laurianne (2003)."An Eye on India: Outsourcing Debate Continues". *IEEE SOFTWARE (Vol. 20, No. 3)*. EUA: IEEE Computer Society. May/June 2003.
11. Beuchot, Mauricio (1997). *Tratado de Hermenéutica Analógica: Hacia un Nuevo Modelo de Interpretación*. Itaca: México. Segunda Edición 2000. pp. 204
12. Mora, Miguel Ángel (2004). *SUMA SISTÉMICA. Claves heurísticas para el rediseño conceptual de la Ingeniería de Sistemas en escenarios de Crisis.* Tesis de Doctorado. México: DEPFI-UNAM. 2004.
13. Jackson, Michael C. (2003). *Systems Thinking: Creative Holism for Managers*. Inglaterra: John Wiley & Sons. 2003. pp. 352
14. Ackoff, Russell L. (1999). *El Paradigma de Ackoff. Una Administración Sistémica.* México: Limusa. 2002. pp. 367
15. Beuchot, Mauricio (2003). "La filosofía del hombre o antropología filosófica según la hermenéutica analógica". Ponencia ofrecida en la *VII Jornada de Hermenéutica*. México, DF: UNAM Facultad de Filosofía y Letras. 2 de Julio del 2003.
16. Checkland, Peter y Holwell, Sue (1998). *Information Systems: Making Sense of the Field.* Inglaterra: John Wiley & Sons. 1998. pp. 262
17. Marín Solís, Ramón. *Propuesta para la integración del Sistema Red de Control Escolar Institucional para el Instituto Politécnico Nacional.* Tesis de Licenciatura. México, DF.: IPN-UPIICSA. 1995
18. Ackoff, Russell L. (1999). "On Passing through 80". En Proceedings: *Russell L. Ackoff and the Advent of Systems Thinking Conference.* Villanova, Paris, Francia: The College of Commerce and Finance. 1999. http://ackoff.villanova.edu/public_html/ackoff99.pdf (05 de Abril de 2003, 03:20:28 a.m
19. Ackoff, Russell L. (2003). *Redesigning Society.* EUA: Stanford Business Books –Stanford University Press. 2003. pp. 184
20. Sánchez Vázquez, Adolfo (2003). *Dimensión Político-Moral del Compromiso Intelectual. Clase Magistral Ética y Política*. México: UNAM. 30 de Octubre 2003.
21. Marín Solís, Ramón. (2004). Arquitectura de cómputo para el diseño de soluciones informáticas usando software libre. *Third international workshop: graphs-operands-logic, with sessions on relativity and computation*. 9 de Febrero, 2004, Universidad Nacional Autónoma de México. Estado de México.
22. Stallman, Richard M. *Free Software: Freedom and Cooperation*. Transcripción discurso. NY, EUA: New York University in New York, New York. 29 May 2001 http://www.gnu.org/events/rms-nyu-2001-transcript.txt (02/08/2004)
23. Ackoff, Rusell L. (2004). *Transforming the Systems Movement*. EUA: Universidad de Pensilvania. 2004. http://www.acasa.upenn.edu/RLAConfPaper.pdf (5 Julio 2004)

CURRICULUM VITAE

Ramón Marín Solís.

Doctorando en Ciencias de la Computación IPN-CIC. Candidato a Maestro en Ciencias, con

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005*. Febrero. México

especialidad en Ingeniería de Sistemas, por el Instituto Politécnico Nacional, en la Sección de Estudios de Posgrado e Investigación de la ESIME-Zacatenco. Lic. en Ciencias de la Informática por el IPN-UPIICSA.

Miembro de la *Association for Computing Machinery*- ACM (Capítulos: Management Information Systems –MIS- y Software Engineering -SIGSOFT) y de la *IEEE* (Capítulos: Cómputo; Educación; y Sistemas).
Áreas de investigación: Metodologías Sistémicas, Sistemas de Información, Diseño de Soluciones de TI, Internet, y GNU/Linux.

Agustín Gutiérrez Tornés
Dr. en Ciencias con Especialidad en Informática. SSGW. Varsovia Polonia.
Profesor Investigador del Centro de Investigación en Computación del IPN, encargado del Laboratorio de Tecnología de Software. Tutor principal del programa de Posgrado en Ciencias de la Computación, línea de investigación Ingeniería de Software.
Áreas de investigación:
Informática, Calidad de Software, Metodología de Desarrollo.

Miguel Ángel Mora Espinosa
Dr. en Ingeniería con especialidad en Investigación de Operaciones DEPFI-UNAM.
Profesor Investigador en la Sección de Posgrado e Investigación de la Escuela Superior de Ingeniería y Arquitectura del IPN.
Áreas de investigación:
Ingeniería de Sistemas en escenarios de crisis. Ética y hermenéutica analógica aplicadas a la Ingeniería.

Citar como: Marín Solís, Ramón. Gutiérrez Tornés, Agustín. Mora Espinosa, Miguel. (2005). Rediseño al paradigma de la ingeniería de software: El caso del software libre. *CONSOL 2005*. Febrero. México