IX Congreso de Ingeniería de Organización
Gijón, 8 y 9 de septiembre de 2005

Proyectos Software desde una perspectiva cibernética

Julio César Puche Regaliza[1], José Manuel Pérez Ríos[1], Pablo Sánchez Mayoral[1]

[1] Dpto. de Organización de Empresas y Comercialización e Investigación de Mercados. Escuela Técnica Superior de Ingeniería Informática. Universidad de Valladolid. Campus Miguel Delibes s/n, 47011 Valladolid. puche@uva.es, rios@uva.es, mayoral@uva.es.

Resumen

Los Proyectos Software siguen sufriendo numerosos fracasos. Sus desarrolladores tienen dificultades para terminarlos dentro del plazo exigido, con costes no superiores a los estimados inicialmente y con la calidad que asegure la satisfacción de los usuarios. A pesar de los avances existentes en diferentes campos del desarrollo de Proyectos Software, en el producto, en el proceso de desarrollo, o incluso en la formación del personal, el número de fracasos sigue siendo desalentador. Proponemos un enfoque organizacional para intentar resolver estos problemas, en el cual esté implicada toda la organización que se dedica al desarrollo del Proyecto Software. Más concretamente, aplicaremos el Modelo de los Sistemas Viables para asegurar la viabilidad, es decir, la capacidad de existencia independiente, auto-regulación, aprendizaje y adaptación de la organización que hace software.

Palabras clave: Desarrollo de Proyectos Software, Modelo de Sistemas Viables, Cibernética Organizacional, Dinámica de Sistemas.

1. Introducción

La importancia del software en la sociedad actual es conocida. Como señala Pressman (2002):

"El software de computadora se ha convertido en el *alma mater*. Es la máquina que conduce a la toma de decisiones comerciales. Sirve de base para la investigación científica moderna y de resolución de problemas de ingeniería. Es el factor clave que diferencia los productos y servicios modernos. Está inmerso en sistemas de todo tipo: de transportes, médicos, de telecomunicaciones, militares, procesos industriales, entretenimientos, productos de oficina…, la lista es casi interminable. El software es casi ineludible en un mundo moderno. A medida que nos adentremos en el siglo XXI, será el que nos conduzca a nuevos avances en todo, desde la educación elemental a la ingeniería genética*"*.

Las inversiones en desarrollo de software no han cesado de aumentar. Se ha estimado que el gasto en Estados Unidos para el desarrollo y mantenimiento del software fue de 70 billones de dólares en 1985 (Boehm, 1987) y 225 billones en 1995, alcanzando en ese año, los 450 en todo el mundo (Boehm y Papaccio, 1998). Las cifras son similares en los primeros años del siglo XXI (Pressman, 2002). Son gastos suficientemente elevados como para prestar una atención especial a los proyectos que los generan.

Y es que el desarrollo de software sigue afectado por tres graves problemas: costes muy por encima de lo presupuestado, entrega fuera de plazo y falta de calidad del producto. El número de fracasos en el desarrollo de software no desciende suficientemente. Según Reel (en

Pressman, 2002), en 1988 los datos de la industria del software indicaron que el 26% de los proyectos software fallaron completamente y que el 46% experimentaron un desbordamiento en el plazo de ejecución y en el coste. Parece clara la necesidad de aportaciones que permitan construir software de manera que se asegure su calidad, fiabilidad, sencillez y robustez, a un coste y plazo de entrega razonables.

En el trabajo que presentamos proponemos un tratamiento de esta problemática basado en la Cibernética. Consideramos que este enfoque puede contribuir a reducir el índice de fracasos. Mostraremos inicialmente algunos conceptos básicos sobre Proyectos Software. En los dos siguientes apartados expondremos algunos elementos básicos de Cibernética, la Cibernética Organizacional y el Modelo de los Sistemas Viables. Posteriormente abordamos la posibilidad de aplicar estos conceptos al desarrollo de software. Terminamos el trabajo con unas conclusiones y referencias al futuro desarrollo de la investigación.

2. Proyecto Software y su gestión.

Toda acción planificada y que requiera de un control durante su ejecución puede denominarse proyecto. Si nos centramos en el marco de un desarrollo software, podemos definir Proyecto Software como un conjunto de acciones planificadas, no repetitivas, de duración determinada, formalmente organizadas, que utilizan unos determinados recursos, y que requieren de un control durante su ejecución con el objetivo de obtener unos resultados técnicos, en costes y en plazos, sin olvidar la necesidad de proporcionar la máxima satisfacción al cliente.

Surge entonces la idea de Gestión del Proyecto Software. Para realizar la actividad explicada en el párrafo anterior, es conveniente plantear un enfoque racional en la administración del proyecto, con todas las funciones típicas del proceso administrativo: planificación, organización, dirección y control. Para evitar el fracaso, el gestor del proyecto y los ingenieros que construyen el producto, deben eludir un conjunto de peligros comunes y comprender los factores críticos del éxito.

Existe numerosa bibliografía que hace referencia a la Gestión de Proyectos Software, en la que se aportan ideas o se intenta dar solución a los problemas que surgen en la realización de los mismos. A pesar de ello, la percepción común que se tiene de la industria del software no es precisamente de una alta calidad de sus productos y servicios, ni de eficiencia en sus procesos. Una de las referencias más citadas sobre el estado de los Proyectos Software es el denominado "*Reported Caos*" del *Standish Group* (en Zavala, 2004), en el que se clasifican los proyectos en tres tipos:

- *Successful*: el proyecto se completa a tiempo y dentro del presupuesto, con todas las características y funciones.
- *Challenged*: el proyecto se completa y es operacional, pero más allá del presupuesto, más allá del tiempo estimado y con pocas de las características y funciones que fueron inicialmente especificadas.
- *Failed*: el proyecto es cancelado antes de completarse.

Señala dicho informe que el 31,1% de los proyectos son *failed* y que el 52,7% de los proyectos cuestan el 189% más que sus estimaciones originales. Se pueden ver en Zavala (2004) éstas y otras estadísticas similares, con inclusión, en algunos casos, de los factores determinantes.

Según Ewsi-Mansan y Przanski (en Zavala, 2004), pocas organizaciones dedicadas al desarrollo de Proyectos Software parecen aprender de sus errores, ya que, en Estados Unidos, el 60% de este tipo de organizaciones fracasaron en más de un proyecto por las mismas causas, dándose además la circunstancia de que el 75% de estas organizaciones no conservaban registros adecuados de sus proyectos fallidos.

Esta repetición de fracasos puede estar provocada por cuestiones de tipo "político". Boehm (en Zavala, 2004) asegura que los gestores no asumen como fracasos, ni los califican como tal, aquellos proyectos que cancelan. Y lo hacen porque ello puede comprometer su carrera profesional. Sin embargo, los cancelan pues son conscientes de que continuar con el proyecto sería un desperdicio de recursos para la compañía. Al final, el resultado es que esos fracasos se ocultan y las causas de los mismos vuelven a aparecer.

Boehm etiqueta los proyectos cancelados como fracasados, estimando que el 31,1% de ellos se cancelaron por una gestión ineficiente. Esta afirmación debe ser tomada con precaución, pues como se podría ver en las estadísticas mencionadas anteriormente, uno de los principales problemas por los que un proyecto fracasa es la inexactitud de los requerimientos iniciales del proyecto que, además, en muchos casos, se modifican a lo largo del ciclo de vida del mismo.

Gran parte de los enfoques que abordan la solución de estos fracasos se centran en:

1. *El producto*: se intenta mejorar el nivel de la calidad del producto entregable mediante modelos, documentos, código. Como ejemplos, citar la utilización de Lenguaje Unificado de Modelado o filosofía orientada a objetos.
2. *El proceso* de desarrollo: mediante la adopción de modelos de ciclo de desarrollo, modelos de calidad y de estimación, que implican el aprendizaje de técnicas de gestión por parte de los gestores de los proyectos. Como ejemplo está la utilización de tecnologías como Proceso Unificado, *Capability Mature Model* o ciclo de vida evolutivo.
3. *El personal*: se intenta desarrollar un modelo de equipo de trabajo que permita manejar la complejidad del sistema a desarrollar. Aquí se pueden citar el *Team Software Process*, *Personal Software Process*, organización de equipo, liderazgo o motivación.

Cada uno de estos enfoques, conocidos como las 3 P's, ha tenido un éxito parcial y relativo en el desarrollo software, pero el índice de fracasos sigue sin descender suficientemente. Brooks (en Zavala, 2004) señala que: "No hay ningún tipo de desarrollo, en cualquier tecnología o con cualquier técnica de gestión, que por sí solo prometa una mejora de un orden de magnitud en cuanto a productividad, estabilidad y simplicidad".

Podemos pensar, por tanto, que se necesita un enfoque de gestión diferente a los tradicionales, que facilite la resolución de estos problemas y disminuya el número de fracasos en Proyectos Software. Nuestra aportación comienza aquí. Pretendemos utilizar la Cibernética, y más concretamente, la Cibernética Organizacional, con el objeto de lograr unos resultados técnicos, de costes y plazos que incrementen la satisfacción del cliente.

3. Cibernética Organizacional.

Después del éxito que habían tenido los grupos de trabajo interdisciplinares (que eventualmente se denominaron grupos de Investigación Operativa), durante la II Guerra

Mundial, en la resolución de problemas que trascendían a la disciplina particular de cada uno de ellos, al terminar ésta, se extendió el uso de tales grupos para la resolución de problemas no militares.

Uno de ellos fue creado por Norbert Wiener en Méjico. Este grupo reconoció la unidad esencial de un conjunto de problemas relacionados con la comunicación y el control, tanto en la máquina como en los organismos vivos. Su trabajo, así como el de científicos como Warren McCulloch, Walter Pitts, Ross Ashby, Grey Walter y otros, dio lugar a una nueva visión sobre la interacción de los sistemas complejos y las respuestas humanas a ellos. Fue Wiener el que dio el nombre de *Cibernética* a la nueva ciencia, a la que definió como ciencia de la comunicación y el control en el animal y en la máquina. En los años de la posguerra tiene lugar la publicación de numerosos artículos y libros sobre la materia, y empieza a conformarse una comunidad internacional interesada en ella. Entre esas aportaciones destacamos el libro de Wiener "*Cybernetics or the Control and Communication in the Animal and the Machine*" (1948), cuyas ideas fueron puestas en práctica por Stafford Beer, que ya formaba parte de dicha comunidad.

El Diccionario de la Real Academia de la Lengua Española ofrece como definición de Cibernética la siguiente: "Estudio de las analogías entre los sistemas de control y comunicación de los seres vivos y de las máquinas; y en particular, el de las aplicaciones de los mecanismos de regulación biológica a la tecnología". Como podemos comprobar, el concepto de control siempre aparece asociado a Cibernética, lo cual es lógico si pensamos en su procedencia del término griego "*κυβερνητικήʺ*, arte de gobernar una nave.

Stafford Beer, en su obra "*Cybernetics and Management*" (1959), da el primer paso en la Cibernética Organizacional, es decir en la aplicación de los principios de la ciencia Cibernética al estudio de las organizaciones. En él realiza una revisión histórica sobre el origen de la Cibernética como ciencia, y reivindica el concepto de sistema como alternativa al enfoque reduccionista dominante en la cultura occidental (el todo puede ser comprendido completamente si se entienden sus partes y la naturaleza de su suma). La descripción de las situaciones complejas como cajas negras y la noción de que los sistemas con una finalidad son definidos por el producto saliente de la caja negra (y no por deseos o intenciones) dieron lugar a la conocida afirmación de Beer de que "el propósito de un sistema es lo que hace". El libro constituye una defensa del "holismo" en el método científico (el todo es más que la suma de sus partes).

Sin embargo, a pesar de los 40 años que han transcurrido desde esta llamada a la aplicación de un enfoque sistémico al estudio de los problemas complejos, y del carácter global de los problemas que afectan a la humanidad, todavía prevalece en medios científicos, académicos, políticos, médicos o sociales, el enfoque reduccionista. Pero también es cierto que la importancia de la investigación sobre el control aumenta de día en día a medida que lo hacen las fuerzas fuera de control.

En ese libro Beer argumenta, por primera vez, la posibilidad de diseñar científicamente una organización para que constituya un sistema dotado de capacidad de aprendizaje, de adaptación y de evolución (Pérez Ríos, 2001).

Nosotros queremos aplicar sus principios a una organización que desarrolla software, y contribuir con ello a la resolución de sus problemas antes comentados. Como señalamos, a pesar de los avances realizados en lo que denominábamos las 3 P's, los problemas en el

desarrollo de software no disminuyen suficientemente. El enfoque sistémico propuesto por Beer nos parece un marco adecuado para ver el desarrollo de software desde otra perspectiva, de manera que podamos relacionar todos los elementos que participan en dicho desarrollo. Nuestro objetivo será diseñar científicamente la organización que desarrolla proyectos software como un sistema dotado de capacidad de aprendizaje, de adaptación y de evolución, que le permitan enfrentarse a los problemas que surjan y de este modo reducir las elevadas tasas de fracaso.

4. Modelo de los Sistemas Viables.

En 1985 y bajo el título "*Diagnosing the System for Organizations*", Beer publica un libro en el que expone los fundamentos conceptuales del Modelo de Sistemas Viables (MSV), que pudieran servir de guía para su aplicación.

El MSV es una de las aportaciones más conocidas y utilizadas de Beer en el ámbito de la Teoría de la Organización. En él establece las condiciones necesarias y suficientes para que un sistema sea "viable", es decir capaz de mantener una existencia independiente (Beer, 1985). Ello implica que dicho sistema estará dotado de las capacidades de regulación, aprendizaje, adaptación y evolución necesarias para garantizar su "supervivencia" ante los cambios que puedan producirse en su entorno a lo largo del tiempo (incluso aunque éstos no hayan sido previstos cuando el sistema fue diseñado).

Para que un sistema sea viable ha de ser capaz de hacer frente a la complejidad del entorno en el que opera. Desde el punto de vista cibernético, el manejo de la complejidad es la esencia de la actividad directiva. Una forma de medir la complejidad de un sistema es su "variedad", entendiendo por ella el número de estados posibles o modos de comportamiento que puede adoptar un sistema. Controlar una situación significa ser capaz de hacer frente a su complejidad, es decir a su variedad, y en este sentido la Ley de Ashby establece que "sólo la variedad puede absorber (destruir) la variedad", o bien, que el "control" sólo es posible si la variedad del "controlador" es equivalente a la variedad de la situación objeto de control (Ashby, 1956).

Desde el punto de vista del *management* esto implica que para que los directivos puedan hacer frente a la enorme variedad presente en el entorno, así como en las operaciones de las cuales son responsables, deben ser capaces de desarrollar la variedad requerida. La variedad del entorno es enormemente mayor que la variedad del sistema productivo encargado de proporcionar los productos o servicios al entorno, y la de éste es, a su vez, también muy superior a la variedad disponible en el sistema directivo encargado de controlarlo. La forma de equilibrar las variedades de los tres elementos (entorno, operaciones, dirección) es mediante el diseño de mecanismos de reducción de la variedad (del entorno con relación a las operaciones, y de éstas con relación a la dirección) y de amplificación de la variedad (de la dirección con relación a las operaciones y de las operaciones con relación al entorno). Este proceso se conoce como "ingeniería de la variedad".

Uno de los postulados fundamentales del MSV afirma que un sistema (p. ej. una empresa) es viable si y sólo si dispone de las cinco funciones caracterizadas por Beer como sistemas uno al cinco, y que de forma muy aproximada podemos asociar con "implementación", "coordinación", "integración", "inteligencia" y "política" (Figura 1). Veamos brevemente en qué consisten (Pérez Ríos *et al.*, 2003).

El Sistema Uno está constituido por los procesos productivos (operaciones) que hacen posible que la organización genere sus productos o servicios.

El resto de los sistemas, del dos al cinco, tienen como misión servir al sistema uno. Así, el Sistema Dos se ocupa de las actividades de coordinación, siendo su principal función amortiguar las oscilaciones que se producen como consecuencia del funcionamiento de las operaciones contenidas en el Sistema Uno y sus interacciones.

El Sistema Tres se ocupa del entorno interno del sistema, en tiempo real. Su misión es intervenir en la negociación de recursos con las operaciones primarias (Sistema Uno), transmitirles instrucciones, auditar su funcionamiento y eventualmente intervenir en él en aquellos casos en los que la coordinación ha sido incapaz de resolver el conflicto entre las operaciones. Se puede decir que la principal función del Sistema Tres es ocuparse del "aquí y ahora" de la organización. Su misión es vigilar el funcionamiento de la organización en el corto plazo.

El Sistema Cuatro representa la "inteligencia" del sistema viable. Ha de vigilar la evolución del entorno de la organización. Su principal misión es ocuparse del "exterior y futuro" de la organización, con la finalidad de mantener a ésta constantemente preparada para el cambio. El Sistema Cuatro idealmente estará formado por la "sala de operaciones", donde son explorados de forma continua diferentes escenarios futuros para ayudar a la toma de decisiones que incrementen la probabilidad de lograr el futuro deseado.

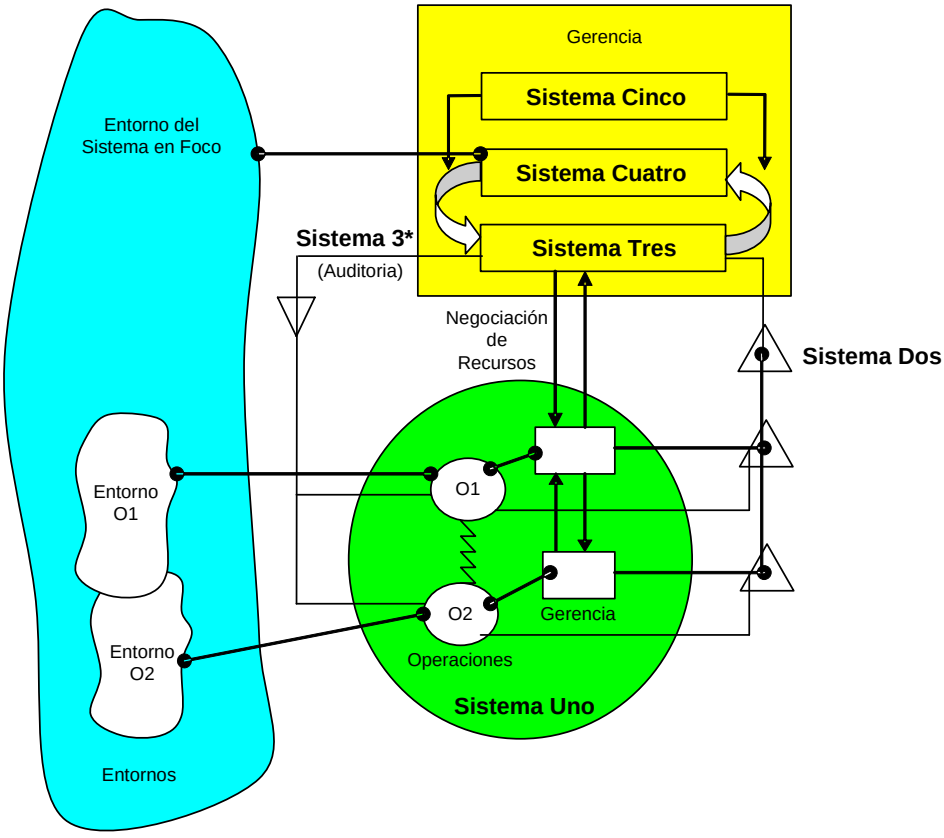

Figura 1. Modelo de Sistema Viable

Finalmente, el Sistema Cinco, que podríamos identificar con la "política" de la organización, se ocupa de los aspectos ideológicos, normativos y define la misión y el estilo de la

organización. Debe asegurar que ésta última se adapte al entorno manteniendo, al mismo tiempo, un grado adecuado de estabilidad interna.

Otro aspecto esencial del MSV es la consideración del carácter "recursivo" de los sistemas viables. Todo sistema viable contiene sistemas viables y, a su vez, forma parte de sistemas que son también viables. La consecuencia directa de esta recursividad es que cualquier sistema viable, sea cual sea el lugar que ocupe, ha de contener los cinco sistemas que caracterizan la "viabilidad". Es decir, la viabilidad del sistema requiere que las cinco funciones existan, de manera recursiva, en todos los niveles de la organización. Toda unidad (Sistema Uno) replica, en términos estructurales, el total en el que está contenida.

5. Proyecto Software desde una perspectiva Cibernética.

Una vez definidos los conceptos fundamentales relacionados con el Proyecto Software y con la Cibernética Organizacional, y más concretamente con el Modelo de Sistemas Viables, estamos en condiciones de aplicar las pautas que dicho modelo nos dicta. Pretendemos avanzar en la línea de convertir al desarrollo del Proyecto Software en un proceso viable, es decir, dotado de las capacidades de regulación, aprendizaje, adaptación y evolución necesarias para garantizar su "supervivencia" ante los cambios que puedan producirse en su entorno, de manera que, como vimos, reduzca la posibilidad de fracaso. Estas capacidades son fundamentales en entornos tan cambiantes como los que afectan al desarrollo de software.

Veamos la composición de los cinco subsistemas fundamentales de todo sistema viable en una organización desarrolladora de software. El Sistema Uno estaría formado por las actividades dedicadas a desarrollar el proyecto. Entre otras, podemos citar: análisis, diseño, implementación y control de calidad.

El Sistema Dos estaría compuesto por la planificación del proyecto, de manera que coordine todas las actividades que se desarrollan en el Sistema Uno.

El Sistema Tres se correspondería con el control y seguimiento de la ejecución del proyecto, mientras que las funciones correspondientes al Sistema 3* serían, entre otras, las auditorías técnicas y las de calidad.

El Sistema Cuatro se ocupa del futuro. Una de las formas de explorarlo es mediante la simulación. Existen variados procedimientos de simulación de sistemas sociotécnicos. Centrándonos en el campo del Proyecto Software, hemos identificado dos tendencias que pueden ser utilizadas por el Sistema Cuatro para llevar a cabo este cometido:

1. *Modelos de estimación estáticos*. Son empíricos y univariables. Se caracterizan por la utilización de fórmulas obtenidas empíricamente de proyectos reales, para predecir los datos requeridos. Entre los más utilizados está *COCOMO* (*COnstructive COst MOdel*), desarrollado por Boehm en 1981 (actualmente *COCOMO II*) (Boehm *et al.*, 2000) y considerado como uno de los modelos más completos. Ha sido formulado como una jerarquía de modelos desde *COCOMO* básico, *COCOMO* intermedio y *COCOMO* avanzado.

2. *Modelos de estimación dinámicos*. Se caracterizan por la aplicación de la Dinámica de Sistemas (Aracil y Gordillo, 1997) al desarrollo de Proyectos Software. Éstos son considerados como sistemas dinámicos sociotécnicos complejos, cuya evolución

temporal está determinada por su estructura interna, que incluye las relaciones establecidas entre las personas que trabajan en el proyecto. Esto permite el desarrollo de modelos dinámicos multivariables para describir el proceso mental seguido por los gestores del proyecto en la toma de decisiones. Con la utilización de modelos de simulación dinámicos, los gestores de proyectos pueden encontrar respuestas a preguntas del tipo: ¿Qué ocurriría si…? Uno de los modelos más completos realizado hasta la fecha sobre el campo que nos ocupa es el de *Abdel-Hamid* y *Madnick* (1991).

Para completar el MSV aplicado al desarrollo de Proyectos Software, mencionamos el Sistema Cinco, en el que debieran de estar representados los diferentes *stakeholders* para desempeñar la función que es propia de este sistema, y que abarca, entre otras, la definición de la ideología, normativa, política.

6. Conclusiones.

El desarrollo de Proyectos Software sigue afectado por múltiples problemas como el exceso de costes respecto a los inicialmente estimados, incumplimiento de plazos de entrega o insatisfacción de los clientes, y ello, a pesar de las diferentes aportaciones y mejoras orientadas al producto, al proceso o al personal (las 3 P´s).

Por esta razón proponemos en este trabajo un enfoque alternativo que, basado en la Cibernética de Beer y aplicando en particular el Modelo de Sistemas Viables, permita diseñar organizaciones que desarrollan software que sean viables, es decir, que estén dotadas de las capacidades de regulación, aprendizaje, adaptación y evolución necesarias para garantizar su "supervivencia" ante los cambios que puedan producirse en su entorno (incluso aunque éstos no hayan sido previstos al inicio de los proyectos). Defendemos en especial este enfoque, habida cuenta de la turbulencia del entorno en que precisamente se mueven este tipo de organizaciones y sus proyectos.

Finalmente, señalamos nuestra intención de profundizar en mayor medida en las asociaciones existentes entre los diferentes sistemas propuestos por el MSV y las actividades necesarias para el desarrollo de software, con el objetivo de ofrecer a las organizaciones unas pautas que les permita obtener unas características de viabilidad. Haremos especial hincapié en el Sistema Cuatro, haciendo una revisión de los principales modelos existentes, tanto estáticos como dinámicos, integrando algunas de las características fundamentales que nos aportan.

7. Referencias.

Abdel-Hamid, T.; Madnick, S. E. (1991). *Software project dynamics. An integrated approach.* Prentice Hall Software Series.

Aracil, J.; Gordillo, F. (1997). *Dinámica de sistemas.* Alianza Editorial.

Ashby, R. (1956). *An Introduction to Cybernetics*. Chapman & Hall.

Beer, S. (1972). *Brain of the firm.* The Pinguin Press.

Beer, S. (1959). *Cybernetics and Management.* English Universities Press.

Beer, S. (1966). *Decision and Control. The Meaning of Operational Research and Management Cybernetics.* John Wiley & Sons.

Beer, S. (1985). *Diagnosing the System for Organizations.* John Wiley & Sons.

Boehm, B. W. (1996). Anchoring the Software Process. *IEEE Software*, junio, Vol. 13, No. 4, pp. 73-82.

Boehm, B. W.; et al. (2000). *Software Cost Estimation with COCOMO II*. Prentice Hall PTR.

Boehm, B. W. (1987). Improving Software Productivity. *Computer*, septiembre, pp. 43-50.
Boehm, B. W.; Papaccio, P.N. (1988). Understanding and Controlling Software Cost. *IEEE Transactions on Software Engineering*, octubre, Vol. 14, No. 10, pp. 1462-1477.
Pérez Ríos, J. (2001). *Laudatio de Stafford Beer.* Investidura de Stafford Beer como "Doctor Honoris Causa" por la Universidad de Valladolid. Universidad de Valladolid.
Pérez Ríos, J.; Hernández, C.; del Olmo, R.; Sánchez, P. (2003). Redtemps: Red Temática de Pensamiento Sistémico. Una red evolutiva para la comunidad de sistemas. *V Congreso de Ingeniería de Organización*.
Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque práctico*. 5th ed. McGraw Hill.
Zavala Ruiz, J.J.M. (2004). ¿Por Qué Fracasan los Proyectos de Software?; Un Enfoque Organizacional. *Congreso Nacional de Software Libre*.
Wiener, N. (1948). *Cybernetics or the Control and Communication in the Animal and the Machine*. MIT Press.