la duración de la construcción del *software*. La estimación se hace difícil de realizar por la incertidumbre que existe antes de comenzar el proyecto, por la falta de datos históricos en los cuales basarse, falta de experiencia del estimador, o malas interpretaciones. El proyecto cuenta con muchas personas, y la influencia de muchos otros factores llamados *drivers* de costo o disparadores de costo. Más adelante en el tiempo aparecerán herramientas que ayudarán al informático a poder estimar con mayor precisión.

- **Expansión anárquica de las aplicaciones informáticas:** Esta etapa ocurre en la década del setenta y en ella se comienzan a recibir distintos pedidos -cada vez más complejos- de los usuarios, y de todos los departamentos del negocio. En esta década comienza a llamarse Departamento de Sistemas de Información, pero la barrera de comunicación con los directivos se mantiene y la selección de los proyectos (que se encuentra a cargo del personal de sistemas) no necesariamente concuerda con los objetivos de la organización por el simple motivo de no formar parte del nivel de toma de decisiones de la misma.

1975

Programación a gran escala (1975 en adelante): no solamente se prestó atención a las entradas y salidas, sino también al rendimiento, la fiabilidad y los estados transitados por el sistema. Se comenzó a centrar la atención en las interfaces y la gestión de los procesos de programación. Comenzó la preocupación por la integridad y la consistencia de las bases de datos. Los programas tenían que funcionar en forma constante. Las herramientas y técnicas acompañaron el cambio de la programación. Ahora comenzaba a trabajarse en grupos organizados de programadores.

1976

Firma digital (Diffie y Hellman): es la analogía electrónica de la firma manual.

La tercera era[7]

Comienza a mediados de la década del '70 y abarca hasta principios de la década del 90.

[7] Pressman, *Op. cit.*, 5.

Aparecieron las redes de área local y de área global, las comunicaciones digitales de gran ancho de banda, y la necesidad de acceder en forma rápida a los datos. Esto produjo mucha presión sobre los desarrolladores.

Hizo su aparición en el mercado la PC.

1979

Puntos de función FPA (Allan Albrech): Los puntos de función miden el *software* cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema.

El FPA tiene como objetivo medir lo que el usuario pide y lo que recibe, haciéndose esta medición independientemente de la tecnología utilizada; da un resultado que servirá para estimar el *software*, y proporcionar un factor de normalización para la comparación de distintos *software*. El FPA da como resultado un número que indica el tamaño que tendrá el *software*, número que sirve para la estimación del costo y el esfuerzo necesario para llevar a cabo el *software*.

1980

- Lenguaje de programación C++, lenguajes de cuarta generación, SQL.[8]
- **Coordinación SI-objetivo de empresa:** Transcurre la década del 80 y la alta dirección empieza a darse cuenta de la importancia del departamento de Sistemas en la organización, por lo tanto se le asignan los recursos necesarios para afrontar las peticiones de los usuarios—que son cada vez más diversas—, y de todos los departamentos. Además se comienza a tomar partida en las decisiones del departamento de Sistemas de Información. Entonces aparece en escena el Jefe de Sistemas de Información y es el punto de partida de la planificación que va acorde con los objetivos estratégicos de la organización.
- En esta época se inicia el cavado de los cimientos de la barrera que existía con la dirección y se comienza con una comunicación directa. Los planes y objetivos de la organización y los planes de Sistemas de Información se aúnan. Además se comienzan a fijar prioridades de proyectos. Sistemas de Información se convierte en un coordinador del

[8] Odell, *Op. cit.*, 70.

equipo interdepartamental que elabora las propuestas de sistemas que, luego de aprobadas, recibirán el presupuesto correspondiente.

1981

- **COCOMO** (Barry Boehm): método de estimación de *software*, por medio del cual obtenemos el esfuerzo "medio hombre" y la duración del proyecto. Esta herramienta proporciona tres niveles de modelos que son: básico, intermedio y detallado. Además se lo puede desarrollar en forma orgánica, semilibre y rígida, dependiendo del tipo, tamaño y limitaciones que posea el *software*.
- **La primera PC de IBM.**

1982

Ciclo de vida de prototipos: El prototipo es una técnica para proporcionar una versión del sistema *software* de funcionalidad, reducida en las fases iniciales de su desarrollo, para ser evaluadas por el usuario. Estas evaluaciones sirven como retroalimentación para refinar los diseños y especificaciones del sistema. La idea básica es que el prototipo ayude a comprender los requisitos al usuario.

El prototipo puede ser de varios tipos:

1. **Desechable**: se usa para ayudar al cliente a identificar los requisitos (aspectos que no están bien claros) del nuevo sistema, y luego se desechan.
2. **Maqueta**: es un ejemplo visual de las entradas y salidas, usando datos estáticos.
3. **Evolutivo**: se trabaja sobre el modelo propuesto, fácilmente modificable y ampliable, se representa una visión física de las partes claves del sistema antes de la implantación. Una vez definidos los requisitos, el prototipo evolucionará hasta transformarse en el sistema final.

1983

- **LAN**: las redes de área local.
- **Internet.**

1985

TCSEC (Departamento de Defensa de los EE.UU): Es un documento conocido como el Libro Naranja, que especifica los diferentes criterios de seguridad del *hardware* y *software*, así como metodologías de evaluación de sistemas informáticos con respecto de la seguridad.

1986

- **Ciclo de vida en espiral** (Barry Boehm): Este ciclo de vida es un enfoque dirigido por el riesgo para el análisis y estructuración de procesos *software*, como así también está dirigido por las especificaciones y el prototipado. El desarrollo es interactivo, en forma de espiral, teniendo en los ciclos internos análisis y prototipado precoz, y en los externos, el modelo clásico.

  Tiene como ventaja el permitir utilizar los modelos de proceso de construcción de *software* tradicionales, mientras su orientación al riesgo evita muchas dificultades. Hace uso del prototipado y del *software* existente.

- **IFPUG** (Grupo Internacional de Usuarios de Puntos de Función): que tiene como función transmitir los conocimientos sobre esta forma de medir el *software*.

- **Tipos de pruebas** (estándar IEEE 1012 - 1986): aunque las pruebas de caja blanca ya fueron mencionadas por Tom McCabe en 1976, el estándar de la IEEE aparece en 1986. Este estándar propone un conjunto mínimo de pruebas que se deben realizar; éstas son:

  1. **Prueba modular o unitaria**: Consiste en probar cada módulo en forma aislada del resto del sistema; pueden ser de dos tipos:

     - **Caja blanca**: El objeto que se va a probar puede ser visto en su interior, y se basa en el conocimiento que se tenga de la estructura del objeto que se desea probar.

     - **Caja negra**: La elección de los datos no se basa en la estructura del objeto, sino en el conocimiento acerca de la funcionalidad deseada.

2. **Prueba de integración**: se realiza a medida que los módulos se van integrando, con el objetivo de probar las interfaces entre los módulos.

3. **Prueba de sistema**: se realiza una vez integrados todos los módulos y su objetivo es comprobar si el sistema satisface los requisitos.

4. **Prueba de aceptación**: una vez que el sistema está implementado en su entorno real; se lleva a cabo esta prueba con el objetivo de demostrar al usuario que el sistema satisface sus necesidades.

- **Gestión de configuración del *software***: nos permite controlar la evolución de un sistema *software*; tiene como objetivo el establecimiento y mantenimiento de la integridad de los productos generados durante un proyecto de desarrollo de *software* y a lo largo de todo el ciclo de vida del producto. Esto implica la realización de tres actividades:

  1. Identificación de la configuración del *software* en ciertos momentos.

  2. Control sistemático de los cambios que se producen en la configuración.

  3. Mantenimiento de la integridad y seguimiento de la configuración a lo largo del ciclo de vida del producto *software*.

  La gestión de configuración del *software* está fuertemente ligada al mantenimiento del *software*; influye también en aspectos como el entorno de desarrollo, el modelo del proceso, la calidad del producto y la organización.

1987

Estándares militares y prácticas industriales (Fuerzas Armadas de los EE.UU): tienen algunas variaciones en las etapas del ciclo de vida en cascada y además especifican los documentos que deben entregarse en cada una de ellas, al igual que las revisiones que deben realizarse al producto. Además, para cada etapa del ciclo de vida se establece la gestión del desarrollo, ingeniería del *software*, cualificación formal y pruebas, evaluación de productos *software* y la gestión de configuración.

1989

- **Estándar IEEE** 1074 - 1989, sobre procesos de *software*: este estándar detalla las fases que conforman el proceso base para la construcción del *software*, y enumera el conjunto de actividades no ordenadas en el tiempo, documentación por emitir y herramientas por utilizar. Esta norma no define un ciclo de vida en particular, sino que debe ser elegido por el usuario.

  El estándar está dirigido a los gestores de proyectos, a los desarrolladores de *software*, a los responsables de la garantía de calidad, a quienes ejecutan tareas de apoyo, a los usuarios y al personal de mantenimiento.

  El proceso está compuesto por cuatro procesos, que son los siguientes:

  1. Proceso de selección de un modelo de ciclo de vida del producto.
  2. Proceso de gestión del proyecto.
  3. Procesos orientados al desarrollo del *software*.
  4. Procesos integrales del proyecto.

- **Garantía de calidad** (Estándar IEEE Std 730-1989): la calidad comprende todos aquellos aspectos o características de un producto o actividad que son significativos en cuanto a la satisfacción de los requisitos. La calidad es la suma de todos aquellos aspectos o características de un producto o servicio que influyen en su capacidad para satisfacer las necesidades. Se la podría definir como un conjunto de actividades de planificación, estimación y supervisión de las actividades de desarrollo, que se realiza de forma independiente del equipo de desarrollo, de tal forma que los productos *software* resultantes cumplen los requisitos establecidos.

  Las áreas que caen bajo la responsabilidad del grupo de garantía de calidad son las siguientes:

  1. Las metas y objetivos de la organización y de los usuarios.
  2. Los métodos: asegurarse que se sigan los procedimientos establecidos, que se ajusten a los estándares seleccionados, de acuerdo con las políticas de la organización.

3. Rendimiento: asegurarse de la optimización del *hardware* y *software*, para que sean eficaces y eficientes.

La cuarta era[9]

Desde fines de la década del '80 hasta la actualidad. Esta era se caracteriza por el impacto producido por las computadoras y el *software*. Las máquinas son mucho más potentes, controladas por *software* sofisticados, unidos a redes locales y globales; las estructuras informáticas tienden a entornos descentralizados cliente/servidor. Aparece la superautopista de información, surgen las grandes compañías de *software*, nuevas metodologías de desarrollo de *software*, sistemas expertos, inteligencia artificial, redes neuronales, realidad virtual, etc.

A pesar de lo mencionado, la crisis del *software* sigue siendo una realidad, debido a que no se lo puede crear tan rápido como se lo necesita en el mercado y con la calidad y fiabilidad que se requiere y, peor aún, no se cuenta con los recursos y las herramientas necesarias.

Aunque nos encontramos en la cuarta era de generación de *software* aún nos hallamos en plena crisis, no hemos aplicado el remedio eficaz para combatirla y estará con nosotros hasta que no nos concienticemos y apliquemos todos los métodos y herramientas que estén a nuestro alcance.

1990

- **CASE:** Generadores de código activados por las herramientas CASE mediante técnicas estructuradas tradicionales.

  Generadores de código activados por herramientas CASE orientadas a objetos.

  Las herramientas CASE son un complemento de la caja de herramientas del ingeniero del *software*. Estas herramientas le permitirán automatizar sus actividades manuales y mejorar su visión general de la ingeniería.

- **ITSEC** (Information Tecnology Security Evaluation Criteria, con el patrocinio de la Comisión de Comunidades Europeas): criterios de

[9] Pressman, *Op. cit.*, 5-6.

evaluación de seguridad informática, que definen los productos TI (Tecnologías de Información), como también aquellos equipos físicos o paquetes de programas que pueden trabajar en una diversidad de entornos. Se consideran sistemas TI a aquellos diseñados y construidos para las necesidades de un usuario específico y que funcionan en un entorno concreto. En el contexto del ITSEC, la seguridad de la TI es sinónimo de confidencialidad, integridad y disponibilidad.

- **Reingeniería** (Michael Hammer): la reingeniería cambió la forma en que se hacían las cosas en las organizaciones, modificando los procesos de negocios, otorgando facultades al personal de menor nivel jerárquico, eliminando pasos que no aportaban valor agregado al proceso ni al cliente y así se aprovechó la oportunidad de utilizar las nuevas tecnologías. Esto último hace que el *software* sufra un gran cambio, ya que éste suele ser la realización de las reglas de negocios; si estas reglas cambian, el *software* debe cambiar también.
- **Interdependencia estratégica de la compañía –TI/SI:** Se continuó avanzado y se integró a Sistemas de Información la tecnología de la información, formando la dupla TI/SI. A principios de la década aparece el gerente de sistemas, que es el máximo cargo del área informática, por así decirlo. Sobre el final de la década del '90 surge el CIO (*Chief Information Officer*) cuya tarea es similar a la del gerente de sistemas, pero está sobre él. El CIO podemos decir que tiene las siguientes misiones: dar solución a las distintas necesidades tecnológicas y de comunicación, dirigir el área tecnológica, determinando inversiones, desarrollando estrategias y políticas, y negociar.[10]

Los sistemas de información dejaron un perfil operativo y tomaron uno estratégico. Por lo tanto, cuando se instala un sistema en una organización, no sólo se debe tener en cuenta el aspecto tecnológico, sino también el factor de negocio; esa es la función del CIO, la de acompañar los cambios.

Sistemas de Información está dejando de ser un área prestadora de servicios para convertirse en un elemento que contribuye a la **estrategia del negocio**. Las áreas de sistemas deben involucrarse en todas las demás áreas del negocio para localizar oportunidades de contribuir con algo valioso.

[10] Walter Duer, "Poder CIO", *Compumagazine*, Año XI, Nº 118, (Mayo, 1998), 47.

El enfoque es cambiar desde la idea de la productividad del *backoffice* (esencialmente emplear la tecnología para hacer lo mismo más rápido y más eficientemente), hacia la idea de influir en la manera de hacer negocios.

1991

CMM (Modelo de Capacidad y de Madurez, Instituto de Ingeniería del *Software*): Es una manera ordenada mediante la cual las organizaciones pueden determinar las capacidades de sus procesos actuales y establecer prioridades de mejora. Esto lo realiza el marco de madurez, estableciendo cinco niveles de proceso de madurez y capacidad progresiva.

2000

Herramientas CASE que utilizan lenguajes y diagramas con sentido directo para los usuarios finales.

Algunos consejos

A continuación se describen brevemente diez puntos que nos ayudarán a superar las crisis del *software* en el futuro:

1. Comunicar al personal lo que se piensa hacer: se debe entender en forma clara lo que se quiere hacer, y esta comunicación se debe refrescar periódicamente.
2. Crear un área de Ingeniería del *Software*: se debe organizar un área de control de calidad que tendrá la función de controlar la aplicación de metodologías y estándares, control de la calidad y administración de la configuración.
3. Implantar un sistema de registro de requerimiento y control de proyectos a partir del reporte de horas diario del personal de desarrollo: por lo general los usuarios no pueden saber el grado de avance de los proyectos informáticos. Por lo tanto, se debe implementar un sistema que contenga las siguientes funciones: registro de los requerimientos, creación de proyectos y asignación de los requerimientos a proyectos, actualización del estado de los requerimientos y los proyectos, consultas para los usuarios sobre el estado de los requerimientos y los proyectos, reporte de horas diarias del personal de desarrollo a los proyectos

en curso, y generación de informes semanales sobre el avance de los proyectos para ser entregados a los usuarios.

4. Elegir un conjunto de técnicas simples para hacer análisis de requerimientos de *software* y dictar la capacitación al personal: una gran parte de los errores en los sistemas se debe a una incorrecta definición de los requerimientos del sistema. El grupo de ingeniería del *software* deben seleccionar este grupo de técnicas y dictar la capacitación.

5. Establecer la obligatoriedad del uso de técnicas de análisis en todos los proyectos importantes de la organización: las técnicas dan buenos resultados siempre y cuando se las use, por lo tanto, al principio, puede ser necesario que se las imponga, ya que siempre se tiene resistencia al cambio. La introducción de estas técnicas se la debe hacer con la ayuda del grupo de calidad.

6. Seleccionar un conjunto de técnicas simples para probar el *software*, que involucren al personal de sistemas y a los usuarios, y dictar la capacitación asociada: el área de metodología y estándares debe seleccionar técnicas de prueba y capacitar al personal de desarrollo. Mejorar la forma en las que se hace las pruebas redundará en beneficios de la calidad de los sistemas.

7. Imponer la obligatoriedad del uso de estas técnicas de prueba: se debe lograr vencer la resistencia al cambio obligando a los desarrolladores a usar las técnicas de prueba, con la ayuda del grupo de calidad.

8. Crear un sistema de registro y seguimiento de fallas: este sistema permitirá registrar todas los errores cometidos, esto nos permitirá mejorar.

9. Introducir un área que se ocupe de la prueba final de los sistemas antes de que pasen a producción, que escriba los manuales del usuario, dicte la capacitación y atienda la mesa de ayuda: esta área debe ser independiente, y el objetivo sería asegurar la calidad de los sistemas que entran en producción.

10. Fundar un grupo formado por representantes de las áreas de desarrollo, prueba y soporte técnico, para mejorar la definición,

administración y uso de los entornos de pruebas y el movimiento de componentes de *software* entre esos entornos.[11]

Conclusión

Hoy, a las puertas del año 2000, con todas las herramientas y metodologías de trabajo mencionadas, todavía no hemos superado la crisis del *software*, y peor aun, estamos esperando la peor crisis del *software* de la historia, el conocido problema del 2000, o el *Armagedón*, como lo llaman algunos. Éste será el punto crítico más profundo sufrido por el *software*. Tal vez esto sirva de lección para que apliquemos de una vez por todas las herramientas que tenemos a nuestro alcance y dejemos de generar *software* en forma artesanal, que pensemos con visión de futuro, que planifiquemos, estimemos, gestionemos, validemos y verifiquemos cada una de nuestras actividades de desarrollo de *software* para el próximo siglo.

Estos consejos están escritos en forma muy resumida. La idea de este artículo es sólo despertar en el lector la necesidad de realizar un cambio, y poder ubicarlo en el camino hacia la solución de su problema; esta solución la encontrará en la **ingeniería del *software***.

Juan Manuel Bournissen
Facultad de Ciencias Económicas y de la Administración
Universidad Adventista del Plata
Dirección: Houssay 445
3103 Libertador San Martín, Entre Ríos
E-mail: bournissen@uapar.edu

[11] Roberto Crespo Wajsman, "Acabemos con la crisis del *software*", *Sindicatura*, Año III-IV, Nº 9 (Abril, 1998), 49-54.