GeneXus™

Grow thru knowledge

Preparando su organización para el futuro

Ken Orr

Si quieres conocer tu pasado, mira tu situación presente.

Si quieres conocer tu futuro, mira tus acciones presentes.

Proverbio Budista

"Futureproofing" y cambio

Definición de Futureproofing: "el acto de asegurarnos de que las decisiones que tomamos hoy tendrán sentido en el futuro."

"Futureproofing" – ¡Que fantástico concepto! ¿No sería grandioso que las decisiones que tomáramos hoy en materia de negocios y tecnología permanecieran válidas a pesar de la andanada de cambios constantes? ¿Que mantuvieran su sentido en un futuro lejano? ¿Qué tal si fuera posible construir sistemas tan flexibles que pudieran adaptarse fácil y rápidamente a todos los cambios comerciales o tecnológicos que tuvieran lugar? ¿Qué tal si fuera realmente posible "preparar el futuro" de su negocio? Quizás no sea tan descabellado como puede parecer. Este documento trata sobre como "preparar el futuro" de su organización usando un conjunto probado de tecnologías de desarrollo y mantenimiento de software.

Considere la idea de "preparar el futuro". Futureproofing se basa totalmente en entender real y profundamente el cambio y aprender a manejarlo. Resulta claro que en el mundo de hoy el cambio es el factor más crítico para cualquier tipo de negocio, tanto en las organizaciones públicas como en las privadas, en las grandes como en las pequeñas. Hoy en día, cualquiera sea el lugar en que se lleve a cabo un negocio, éste deberá enfrentarse con los cambios. Todos los gerentes, independientemente de su grado de responsabilidad, deben tomar decisiones críticas Y estas decisiones deben perdurar, a pesar de los cambios acelerados que vayan a enfrentar. Y en ningún área es este hecho más contundente que en el área de la informática.

Semanalmente, casi podríamos decir diariamente, se incorporan al mercado nuevos dispositivos y nuevo software. Esto crea importantes oportunidades pero, al mismo tiempo, vuelven obsoletas las gigantescas inversiones efectuadas en los dispositivos y/o software ya existente. Del mismo modo, casi a diario tienen lugar desarrollos empresariales que cambian el panorama de los negocios. El éxito de muchas organizaciones o, más aún, su existencia misma en los próximos años, dependerá en gran parte de su habilidad para gestionar el cambio –preparar sus organizaciones- y de esto es de lo que se trata este documento.

Megatendencias: ¿de dónde vienen todos estos cambios?

Pero el cambio no es algo nuevo, y ni aún el cambio masivo lo es. Durante el último siglo hemos visto la introducción de las radios, los aviones, los televisores, la energía atómica, los circuitos integrados, los viajes espaciales, la ingeniería genética y la ciertamente no menos importante, computadoras. Al comienzo del siglo veinte, la mayoría de la gente viajaba a pie o a caballo; hacia finales de siglo viajaba en autos de alta velocidad y en aviones. Al comienzo del siglo veinte los teléfonos eran algo fuera de lo común, hacia finales de siglo los teléfonos celulares eran algo tan común como las billeteras o los portafolios.

Pero la clase de cambio que está teniendo lugar en el siglo veintiuno es de una nueva dimensión. Existen una serie de megatendencias que están empujando nuestro mundo hacia delante, cada vez más aceleradamente, haciendo que la toma de decisiones, aún de las más mundanas, sea cada vez más riesgosa. Las más importantes de estas megatendencias son las siguientes:

- Megatendencia 1: la historia se está acelerando
- Megatendencia 2: el futuro es altamente impredecible
- Megatendencia 3: el futuro no es lineal

Megatendencia 1: la historia se está acelerando

Actualmente, las presiones de los negocios y la tecnología son de tal magnitud que las organizaciones y las personas son víctimas de un estrés cada vez mayor para seguir adelante. Lo más probable es que un empresario lleve hoy consigo un teléfono celular, un PDA y frecuentemente también una laptop. Hace 50 años los teléfonos estaban sujetos a la pared. Solo un grupo de acaudalados podía tener teléfonos en sus autos o en sus barcos. En la actualidad, cada vez más personas, desde acaudalados a trabajadores, pueden costearse un teléfono celular. Cada día se anuncian más y más dispositivos que hacen posible la comunicación desde casi cualquier lugar del planeta.

Y el cambio también se está acelerando en el mundo del software. La duración media de las plataformas y lenguajes está disminuyendo exponencialmente. Hace ocho años nadie pensaba en Java. Hace cinco años nadie pensaba en .NET. Y hace tres años nadie pensaba en Linux.

Y en el mundo del hardware, los cambios ocurren aún con mayor rapidez. Hace tres años nadie pensaba en iPods y hoy en día todo el mundo lo hace. Actualmente, las disqueteras, la memoria y los CPU son tan pequeños y de tan bajo costo que pueden ir en casi cualquier dispositivo. Con cada generación de nuevos dispositivos hay una mayor integración y miniaturización y los costos se reducen significativamente.

Pero las personas no cambian con la misma rapidez que los negocios y la tecnología, lo que resulta en que las organizaciones tienen dificultades para mantenerse al día. Hace 30 años los gerentes de informática podían tomar decisiones y esperar que fueran válidas por cinco o diez años como mínimo. Actualmente la duración media de cualquier nueva tecnología o herramienta o plataforma de software, aún de las más prometedoras, es extremadamente limitada.

Megatendencia 2: el futuro es cada vez menos predecible

La predicción es muy difícil, especialmente la del futuro. – Niels Bohr

Desde el comienzo del siglo veintiuno ya hemos presenciado enormes cambios: el final de la "burbuja .COM", el ataque del 11 de setiembre al World Trade Center y al Pentágono, el explosivo crecimiento de las economías de China y de la India, la creciente tercerización de puestos de trabajo y productos hacia el Extremo Oriente, la Guerra de Afganistán y de Irak y el devastador sismo del tsunami en el Océano Índico que se llevó 250.000 vidas humanas en unas pocas horas.

Solo los analistas e historiadores más perceptivos podrían haber adelantado lo que ha ocurrido en los últimos cinco años. Y solo los más temerarios querrían predecir que es lo que puede pasar en los próximos cinco años, y menos aún en los próximos veinte años.

Históricamente, los futuristas y pronosticadores adjudicaban cierto nivel de riesgo o error a sus predicciones. Hoy en día, las personas que reciben remuneración por "leer" el futuro se resisten cada vez más a aumentar los niveles de certeza que atribuyen a sus predicciones, aún para periodos de tiempo limitados; el futuro no es ya tan predecible como solía ser: ¡el mundo se está moviendo demasiado deprisa!

Megatendencia 3: el futuro no es lineal

Muchas personas que se ganan la vida intentando predecir el futuro también les dirán que una de sus herramientas más importantes es un *patrón lineal*. Han llegado a la conclusión de que muchas tendencias son lineales. Pero el futuro ya no es lineal. Aún una guía tan históricamente confiable como la Ley de Moore está comenzando a tambalearse. Desde mediados de la década del 70 y hacia principios del siglo veintiuno, los computadores personales y toda la tecnología que hay detrás de ellos trajo una enorme ola de innovación y expansión tecnológica de acuerdo con la Ley de Moore. Pero los computadores personales ya no son quienes impulsan la tecnología; en realidad, el mundo de los PC está luchando por sobrevivir. Y la Ley de Moore puede estar flaqueando.

El mundo está atravesando un periodo de discontinuidad. Mientras que los computadores personales e Internet fueron los motores que impulsaron los últimos años del siglo veinte, lo que impulsa al mundo actualmente es algo diferente, quizás los teléfonos inteligentes o los dispositivos personales para disfrutar de entretenimientos o la comunicación inalámbrica. Y en tanto que en la era de los computadores personales los pronosticadores hablaban en términos de millones de dispositivos, en la era de los teléfonos inalámbricos inteligentes hablan en términos de miles de millones de dispositivos. No es de sorprenderse entonces si las organizaciones, al igual que las personas, tienen problemas para adaptarse a los cambios en una escala semejante.

Con todos estos cambios, lo que las organizaciones necesitan en la arena del software es una nueva generación de herramientas, o más precisamente, una nueva manera de desarrollar y mantener las aplicaciones de negocios y un conjunto de herramientas que soporten esta nueva modalidad para que pueda acompañar a los cambios del siglo veintiuno. El resto de este documento trata justamente sobre la clase de tecnología necesaria para hacer esto posible.

Problemas asociados con "preparar el futuro" de nuestros sistemas de software

No obstante, antes de discutir cuales podrían ser las importantes soluciones de "Futureproofing", debemos considerar algunos de los temas principales que deberemos abordar:

- La necesidad de nuevas aplicaciones y nuevas capacidades excederá en mucho a nuestra capacidad para proporcionarlos.
- La próxima generación de aplicaciones requerirá de un conjunto totalmente nuevo de capacidades.
- Existe la necesidad de crear un ambiente estable que se enfoque más en los problemas / oportunidades de los nuevos negocios y menos en resolver los problemas tecnológicos.

La necesidad de nuevas aplicaciones y capacidades excederá en mucho a nuestra capacidad para proporcionarlos

Hasta ahora hemos estado hablando sobre el cambio; en esta sección debemos discutir en términos de sistemas *hiper-adaptables*. Estos sistemas hiper-adaptables deben enfrentarse a lo siguiente:

- Cambios en las necesidades del negocio.
- Cambios en las oportunidades tecnológicas.
- Ambientes de desarrollo cada vez más complejos.
- Falta de personal capacitado.
- Una significativa renovación de personal debido al retiro de los "baby boomers".

Desde el colapso de la burbuja .COM, un gran número de organizaciones han estado recortando sus gastos en información tecnológica, especialmente en el área de desarrollo de nuevos sistemas. Las organizaciones han marcado una fuerte tendencia a: (1) tercerizar sus departamentos de desarrollo de aplicaciones junto con sus operaciones de computación, (2) grandes paquetes integrados para reemplazar porciones de sus sistemas legacy y (3) desarrollar front end basados en la web para sus aplicaciones de mainframe de misión critica. Estas tendencias están a punto de cambiar.

El mundo de los negocios del 2005 es mucho más competitivo de lo que era hace una década, e inclusive que hace cinco años. El mundo está mucho más interrelacionado e integrado y es mucho más conocedor de la tecnología. Al mismo tiempo, los sistemas en los que las organizaciones confían para su sobrevivencia diaria están cada vez más obsoletos, y la gente que desarrolla estos sistemas, si es que aún trabajan, ya están cada vez más cerca de retirarse.

En la próxima década, las organizaciones más grandes tendrán que reemplazar o desarrollar nuevamente muchas de sus aplicaciones más críticas –sus aplicaciones medulares. Y en muchos casos, las aplicaciones que más necesiten no estarán disponibles en el mercado; deberán ser desarrolladas y esto deberá hacerse muy rápidamente.

La próxima generación de aplicaciones requerirá de un conjunto totalmente nuevo de capacidades

¿Cómo deberá ser esta nueva generación de aplicaciones hiper-adaptables? Bueno, lo cierto es que será cada vez más importante que estas aplicaciones puedan moverse rápidamente de una plataforma a otra. Si bien se ha elogiado mucho a los "sistemas abiertos", la mayor parte de las aplicaciones, aún hoy en día, son desarrolladas para una sola plataforma, un solo lenguaje y un solo sistema de base de datos. Las preocupaciones relativas a costos y desarrollo seguramente cambiarán esta perspectiva mucho más rápidamente de lo que hubiera podido imaginarse, aún hace unos pocos años. Por ejemplo, actualmente no hay solamente dos plataformas que compiten entre sí (Java y .NET), hay por lo menos tres (Java, .NET, y Linux). ¿Y cuántas podrá haber en un futuro cercano? Nadie lo sabe.

Y mientras que las organizaciones se han vuelto muy competentes en la construcción de sistemas que trabajan sobre escritorios y laptops conectadas a Internet, la necesidad de soportar dispositivos inalámbricos con pequeñas pantallas con conexiones de bajo ancho de banda representa un nuevo desafío. También lo es el desarrollo de aplicaciones que deben trabajar con dispositivos que no están siempre conectados. ¿Quién sabe cuál será el próximo desafío? Quizás sea el RFID, o quizás los dispositivos GPS, o quizás todo lo anterior. Lo que sabemos con certeza es que la cantidad y las distintas clases de dispositivos se multiplican, aún cuando pensamos solo en términos de nuestras posibilidades actuales. Y cada nuevo dispositivo representa un nuevo desafío. Los dispositivos creados para un propósito se modificarán y serán usados para algo más. Y se crearán dispositivos con los que nadie ha soñado, y estos deberán ser soportados.

Entonces, la nueva generación de aplicaciones de computación deberán poderse mover fácilmente de una plataforma a otra, de un lenguaje a otro y de un equipo de hardware a otro. Y deberán poder soportar diferentes sistemas de gestión de base de datos. Como les dirán los programadores, esta no es tarea fácil. Aún cuando las principales bases de datos relacionales son muy similares, de acuerdo a como se construyen las cosas hoy, hay suficientes diferencias como para que aún la conversión de una base de datos relacional a otra base de datos relacional sea una tarea nada trivial. Y lo mismo ocurre con los lenguajes. Aunque Java y C# sean muy similares, la conversiones de uno al otro tampoco son una tarea sencilla.

Finalmente, la próxima generación de aplicaciones deberá ser: (1) de fácil comunicación, (2) más tolerante a las fallas, y (3) muchísimo más segura. Debido a que las computadoras se adentran más y más como tendencia prevaleciente en la vida diaria, los sistemas deberán diseñarse no para los técnicos sino para los abuelos y abuelas. La razón por la que la Computadora Apple ha podido sobrevivir y reaparecer como competidora en el mundo de la alta tecnología es en gran parte el resultado de la atención que Apple ha prestado siempre al diseño de la interfase humana, haciendo las cosas fáciles de usar.

La necesidad de crear un ambiente estable que se enfoque más en los problemas / oportunidades de los nuevos negocios y menos en resolver los problemas tecnológicos

Si hemos de desarrollar una solución para "preparar el futuro" de nuestras organizaciones, deberemos concentrarnos cada vez más en enfoques que nos permitan pasar la mayor tiempo de nuestro tiempo con el usuario, explorando cual es la mejor manera de solucionar los problemas de su negocio. Actualmente, los sistemas se esfuerzan demasiado en descubrir como utilizar la base tecnológica inmensamente rica que existe.

En el futuro, para tener éxito deberemos llegar a soluciones que nos permitan concentrar la mayor parte de nuestros esfuerzos de desarrollo en entender y resolver los problemas de nuestro negocio, empleando un porcentaje cada vez menor de nuestro tiempo en la tecnología.

Estrategias para "Preparar el Futuro"

A esta altura, sin embargo, recién hemos establecido el escenario para preparar el futuro de nuestras organizaciones. Hemos expuesto la importancia de "Futureproofing", pero no hemos visto aún cuales son las alternativas serias. Resulta claro que desde el comienzo de la era de la computación hemos estado intentando resolver el "problema del cambio". El desarrollo de un software con el que se pueda trabajar ha sido siempre un desafío, pero el desafío de mantener o modificar ese software para que pueda hacer cosas para las que no había sido creado en su origen ha constituido un desafío aún mayor. En esta sección vamos a ver las soluciones a este problema, tanto a corto como a largo plazo.

Soluciones a Corto Plazo

Hoy en día en el mundo de los negocios hay una serie de soluciones conocidas que apuntan a ayudar a las organizaciones a hacer frente a los efectos del cambio. Las más conocidas son las siguientes:

- Paquetes comerciales
- Tercerización
- Desarrollo ágil
- Componentes reutilizables

Paquetes Comerciales

Cada vez más organizaciones en todo el mundo han decidido que "no estaban en el negocio del desarrollo de software". En términos prácticos, han decidido comprar software crítico en lugar de desarrollarlo. En lugar de intentar mantener su propio y muy costoso personal de expertos en software, muchas empresas han elegido comprar paquetes de software comercial a vendedores. Esto ha trasmitido la pesada carga de mantener el software actualizado al vendedor. Los cambios en la tecnología y en las reglas del negocio y las innovaciones se entregan como parte del costo del paquete y, por supuesto, el mantenimiento del paquete. Si una organización compra un paquete de software, esto significa que no necesita un gran número de entrenados analistas, diseñadores y desarrolladores, y que las organizaciones solo deben preocuparse acerca de qué paquete comprar, ¿o no?

El problema es que los paquetes tienen su lado oscuro. Ningún paquete, sin importar lo bien concebido que esté, podrá satisfacer las necesidades de una organización de grandes dimensiones tal como viene de fábrica. Entonces, las empresas que compran paquetes se

enfrentan a un dilema: ¿(1) adaptar el paquete para que satisfaga las necesidades únicas de su organización o (2) mantener el paquete tal como está y cambiar la organización para que se adapte al él?

Cualquiera de las dos soluciones presenta problemas. En los primeros tiempos de los paquetes de software, las organizaciones tendían a adaptarlos extensivamente. Esto traía como consecuencia que debían enfrentar serios problemas al cambiar de una versión del paquete a la próxima. Después de un par de crecientemente difíciles cambios de versión, estas organizaciones usualmente cambiaban a una estrategia de mínima adaptación de los paquetes y crecientes cambios en la organización para que se adaptara a ellos. Pero esto tampoco funcionaba, y debían plantearse nuevamente más adaptaciones. La historia demuestra que el péndulo continúa moviéndose hacia atrás y hacia adelante.

Pero los paquetes presentan elementos aún más oscuros que la adaptación. Con el tiempo, las organizaciones se han encontrado atadas a vendedores específicos. Es extremadamente difícil salirse del compromiso adquirido con un gran paquete en menos de 8 a 10 años. Y en ese tiempo, el panorama competitivo puede haber cambiado drásticamente. Peoplesoft compra J.D. Edwards, Oracle compra Peoplesoft, etc. Las empresas de software se salen del negocio o son adquiridas todo el tiempo, y repentinamente las organizaciones se encuentran atadas a vendedores de software que nunca eligieron explícitamente.

Tercerización

Otra forma en que las organizaciones han intentado manejar los problemas del cambio de software es delegándolo a alguna organización externa. En los últimos años, la tercerización se ha convertido en un método común para manejarse con el software y los cambios de software. En lugar de mantener su propio staff de expertos en software, lo que se argumenta es: ¿por qué no pasarle este trabajo a empresas que se especializan en software? Las publicaciones de negocios están llenas de anuncios sobre la tercerización que hacen grandes organizaciones de su desarrollo de software y/o de sus operaciones de computación.

Normalmente, tal como la compra de paquetes de software, la tercerización representa un compromiso a largo plazo. Con frecuencia el personal interno de la empresa es transferido a la empresa de tercerización y en el corto plazo, la mayoría del personal con excepción del gerente y los profesionales del departamento de tecnología, retienen lo mismo de la relación, solo que ahora las discusiones entre los analistas, diseñadores y desarrolladores y los usuarios del negocios se vuelve mucho más formal y, por supuesto, mucho más costosa. Y como ocurre con los paquetes, la tercerización presenta inconvenientes definitivos.

La tercerización crea frecuentemente barreras entre los usuarios del negocio y los desarrolladores de software. Después de un tiempo, el staff original que una vez trabajó para la organización cliente comienza a cambiar, y la próxima generación de personal de tercerización tiene menos apego e interés en el negocio para el que están construyendo software. Además, las empresas de tercerización tienen frecuentemente un interés creado en minimizar los cambios en el software o las plataformas subyacentes. Los grandes cambios significan con frecuencia un nuevo desarrollo, o la adquisición de paquetes, actividades que amenazan el acuerdo de tercerización.

Probablemente el problema más serio que enfrenta la tercerización es la pérdida de conocimiento por parte de las persona claves del negocio. Las organizaciones que tercerizan todo o casi todo su desarrollo de software terminan frecuentemente careciendo de personal que sepa lo suficiente como para manejar la relación de tercerización. Típicamente, cuanto más tiempo pasa, las organizaciones de tercerización ganan más y más poder y comienzan las fricciones entre ambas organizaciones. Frecuentemente, las organizaciones que intentan volver a tener el control de las relaciones previamente tercerizadas tienen que reconstruir la estructura de la organización que habían dejado en manos del tercerizador, un proceso que normalmente toma muchos años, y a veces décadas.

Reutilización: Objetos, Componentes y Servicios

La instalación de paquetes y/o la tercerización son grandes estrategias que las organizaciones han estado empleando en los últimos años en un intento por bajar sus costos y en gran medida para disminuir su exposición a los cambios. Ambas estrategias involucran básicamente transferir la carga del desarrollo y mantenimiento de software a otros. Pero en muchos casos, existen grandes volúmenes de software que aún deben ser desarrollados en la empresa. Algunos proyectos son demasiado estratégicos o demasiado puntuales en términos de tiempo o demasiado críticos como para dejarlos en otras manos.

Para aquellas organizaciones que aún hacen desarrollo de software, la palabra santa durante los últimos 10 o 15 años ha sido la reutilización. De acuerdo con muchos expertos en el tema, el software es tan caro y propenso a errores primeramente porque, a diferencia de otras formas de productos, el software aún no ha evolucionado a una disciplina de ingeniería basada en “partes reutilizables”. En gran medida, la “revolución orientada a objetos” ha sido vendida en base al concepto de reutilización. Se ha intentado diseñar los objetos de forma tal que puedan ser fácilmente utilizados en un gran número de programas diferentes.

Durante los últimos 10 o 15 años, ha surgido toda una gama de soluciones reutilizables, primeramente objetos y componentes y ahora servicios. La idea detrás de lo que ahora se

llama “desarrollo basado en componentes” es que, si los componentes son diseñados correctamente, se convertirán en un nicho de mercado en el que, en lugar de desarrollar programas de cero, los desarrolladores y los ingenieros podrán elegir componentes estándar de un catálogo en Internet y simplemente “armarlos” para crear nuevos programas.

Algunas partes de esta agenda de reutilización han funcionado. Los objetos y componentes son ampliamente utilizados por varios integrantes de la comunidad de programadores hoy en día. Por ejemplo, se explotan ampliamente los objetos en el diseño de interfaces de usuario gráficas (GUI). Las GUI son tradicionalmente complejas y difíciles de crear, y los objetos resultan ser una manera natural de construir interfaces complejas. Otras áreas incluyen aplicaciones de presentación, como traducir números en gráficos o diagramas. Pero el concepto de “objetos de negocio” no ha sido para nada tan exitoso. Los objetos/componentes no han penetrado tanto en el dominio de los negocios como lo han hecho en el caso de las GUI o de la lógica de presentación.

Parte del problema que afecta a los componentes reutilizables es su naturaleza estática. Desde el comienzo, se ha asumido que los objetos de software son similares a los repuestos de un automóvil o un refrigerador, y esto no ha resultado para nada así. Por un motivo, las partes del software han resultado ser mucho más complejas que la mayoría de las partes físicas. Y por otro motivo, los diseñadores de objetos/componentes se han olvidado que incluir un elemento fundamental para el diseño del producto en su ecuación: la estructura del producto.

En el mundo real, los ingenieros de productos pasan gran parte del tiempo diseñando la estructura general del producto. Los diagramas de diseño más importantes muestran la estructura general y el lugar donde va cada parte. Desafortunadamente, los diseñadores de software han ignorado la estructura del producto (o han subestimado su importancia). El resultado ha sido que la mayoría de los componentes disponibles como componentes estándar son estáticos. Y una de las consecuencias de este descuido es que los componentes terminan siendo mucho más complejos de lo que deberían ser, porque los diseñadores han pensado en todos los posibles usos que pueden darse a un componente estático.

Las consecuencias de estas presunciones sobre la reutilización han sido que el desarrollo de software orientado a objetos se ha vuelto más complejo y más costoso de lo que se había previsto y el software producido se ha vuelto más difícil (y costoso) de mantener. Es un problema especialmente molesto ya que más de la mitad del costo de cualquier aplicación de software está relacionado con el mantenimiento y actualización de la aplicación.

Entonces, falta mucho por aprender de la observación de la ingeniería de productos en un sentido más amplio. Durante los últimos 30 o 40 años, todas las formas de ingeniería han avanzado grandes pasos al usar computadoras y software de computación para asistirlos en el diseño, ingeniería, prototipacion y fabricación de sus productos. El Diseño Asistido por Computadora (CAD), la Ingeniería Asistida por Computadora (CAE), la Prototipacion Asistida por Computadora (CAP) y la Fabricación Asistida por Computadora (CAM) son uniformemente usados en todo el mundo de la ingeniería.

Actualmente, los ingenieros pueden pasar de los dibujos conceptuales a la prototipacion de partes en cuestión de horas. Los dibujos CAD pueden analizarse y simularse usando software CAE. Pueden identificarse problemas y efectuarse cambios, y se pueden fabricar nuevas partes usando software CAP o CAM, todo desde productos digitalizados y virtuales almacenados como datos meta en computadores gráficos de alta velocidad.

A diferencia de los productos de software, los productos digitales y las partes almacenadas en computadores en el espacio aéreo, la electrónica y la arquitectura se pueden reconfigurar instantáneamente para producir productos finales. Desafortunadamente, el desarrollo de software en las organizaciones de desarrollo de software más avanzadas aún requiere una enorme intervención manual. Los desarrolladores de software argumentan que el software es demasiado complejo para ser generado por computadora, pero el resto del mundo de la ingeniería ya ha superado esa idea. *Si las organizaciones van realmente a "preparar su futuro", deben volcarse a tecnologías probadas en base a la otra forma más avanzada de ingeniería: las herramientas de prototipacion para la generación de diseños de acuerdo a requerimientos dinámicos.*

Desarrollo Ágil

Finalmente, hay otra solución a corto plazo para los problemas de cambio de software; algo llamado "desarrollo ágil". A través de toda la historia del software, el enfoque predominante ha sido una "estrategia en cascada", en la cual los proyectos se dividen en fases (planeamiento, requerimientos, construcción e instalación). Cada fase es completada antes de que comience la próxima fase. Este enfoque se tomó de estrategias para la construcción de productos de hardware en gran escala empleadas durante la Segunda Guerra Mundial e inmediatamente después de esta. La gran fortaleza de este enfoque es que permite la especialización y es fácil de entender.

Pero aunque resulte atractiva, la estrategia en cascada tiende a presentar un gran número de problemas. Debido a que el software no es una disciplina madura, no hay estándares acordados para la especificación, diseño y prueba de programas, bases de datos y

especialmente reglas de negocios. En proyectos muy grandes, cada una de las fases principales puede tomar meses y aun años. Como resultado, cuando los proyectos que han llevado un largo tiempo son finalmente implementados ya se han desactualizado. Cuanto más grande sea el proyecto más probabilidades hay de que fracase.

A mediados de los noventa, varios expertos en software comenzaron a atacar este obstáculo para el desarrollo de software. Comenzaron a preguntarse si no sería mejor diseñar, construir y probar el software en incrementos pequeños, trabajando directamente con el usuario final. Entonces, después de que el usuario evaluara el incremento actual y que ellos afinaran sus ideas y definieran el próximo grupo de requerimientos, los desarrolladores podrían rediseñar el producto agregando las nuevas piezas y corrigiendo los posibles errores o efectuando cambios al grupo de requerimientos previo. Entonces podrían crear un nuevo incremento para ser revisado por el usuario, nuevamente en un periodo muy corto. Este proceso en su totalidad se dio a conocer como "desarrollo ágil". Las organizaciones que lo emplearon informaron obtener increíbles ganancias. Pero como en todo lo demás, también el desarrollo ágil tiene su lado oscuro.

Una de las razones por la que el desarrollo ágil se hizo tan popular fue que tanto los usuarios finales como los desarrolladores son inmediatamente recompensados. No existe retraso entre el requerimiento y la respuesta ni esfuerzos inútiles. Los requerimientos se transforman directamente en códigos. No obstante, como sabe todo aquel que haya hecho desarrollo de software, el software complejo requiere frecuentemente un diseño y un análisis muy serio.

Entonces, el lado oscuro del desarrollo ágil tiene que ver con la falta de documentación del diseño y con el costo de mantenimiento en el tiempo. Una de las reglas de oro del desarrollo ágil es que los equipos de proyecto deben evitar el papeleo siempre que sea posible. Muchos extremistas del desarrollo ágil creen inclusive que "el código es la única documentación que realmente se necesita". Ahora, esta es una afirmación que se ha hecho muchas veces en la historia del software, principalmente por parte de los hackers, pero nunca demostró ser correcta. El mantenimiento, especialmente el mantenimiento realizado por personas que no han trabajado en el software original, siempre ha requerido de documentación. La escasa documentación equivale a software difícil (es decir, costoso) de mantener.

Pero el desarrollo ágil tiene una serie de elementos para ser recomendado. Por ejemplo, el desarrollo incremental es una gran idea. Históricamente, uno de los mayores problemas con el desarrollo de software es el resultado de intentar proteger los requerimientos actuales durante demasiado tiempo en el futuro. En realidad, este es uno de los problemas más

importantes enfrentados por el diseño orientado a objetos; el deseo de diseñar el objeto perfecto. Entonces, el desarrollo incremental es una buena idea.

Y el trabajar junto a los usuarios en el desarrollo de prototipos es también una excelente ideal. La experiencia ha demostrado cuanto más rápidamente puedan trasladarse los requerimientos a los programas que están corriendo mejor. Y muchos usuarios se manejan mejor con cosas tangibles como prototipos que con dibujos abstractos de estas pantallas.

Y el rediseño de programas, es decir el hecho de realmente rediseñarlos para cada incremento es también una excelente idea. En el mundo del desarrollo ágil, esto se ha dado en llamar "refabricación". Aunque suene muy sofisticado, "refabricación" significa simplemente revisar todo el diseño y código (clase objeto), la base de dato para asegurarse de que los nuevos requerimientos y cambios se han construido como parte del sistema ("built-in") en lugar de haber sido simplemente agregados ("added-on"). El único problema con la refabricación, especialmente con la refabricación manual, es que cuanto mayor es el sistema o aplicación más tiempo lleva realizarla. Entonces, a medida que un programa o aplicación se agranda más, mayores serán los esfuerzos que insumirá cada nueva refabricación. Esto significa que o bien las iteraciones de desarrollo se hacen cada vez más lentas o la refabricación se tira por el resumidero.

Aprender de las Soluciones a Corto Plazo

A mediados de los ochenta había una gran movida hacia la "automatización del desarrollo de software" para producir herramientas de Ingeniería de Software Asistida por Computadora (CASE). Desafortunadamente, la tecnología no estaba preparada. Había demasiadas cosas para ser inventadas e integradas, demasiadas cosas que la industria del software simplemente no había imaginado. Pero el objetivo de automatizar el desarrollo del software era correcto; en otras palabras, definir al software así como los ingenieros o arquitectos de productos definen los productos o las construcciones, y después hacer que la computadora generara la solución final en segundos o minutos en lugar de semanas o meses. En realidad, cuando se piensa sobre ello, parecería que el software es el dominio ideal para el desarrollo de esta idea, porque, para comenzar, los productos de software son esencialmente digitales por naturaleza.

Para realmente "preparar el futuro" de nuestras organizaciones, deberemos resolver este problema de automatizar el proceso de desarrollo de software. Necesitamos soluciones hoy que nos permitan definir nuestros procesos de negocios, actividades, pantallas/reportes y reglas de negocio y después dejar que la computadora genere automáticamente las bases de datos y programas que corran contra esas bases de datos para producir aplicaciones ejecutables. Y esto no puede ser un proceso de solo un ciclo. La computadora no deberá

solamente ser capaz de generar la primera iteración desde nuestros procesos de negocios, etc. También deberá poder hacer todo tipo de cambios a esos elementos y rediseñar, regenerar y reimplementar los nuevos programas en segundos o minutos. Solo entonces será posible que nuestras organizaciones se adapten a las nuevas circunstancias de negocios y a las nuevas tecnologías ("preparar su futuro") sin miedo y sin enormes costos.

Soluciones a Largo Plazo para Futureproofing

Como hemos visto, hay muchas soluciones posibles a corto plazo para "preparar el futuro" de nuestras organizaciones, pero todos presentan inconvenientes importantes. El problema fundamental de todas ellas es que aceptan el actual paradigma del software: que el software es fundamentalmente un proceso manual. Para tener una estrategia que sea realmente a largo plazo para "preparar su futuro", las organizaciones deberán estudiar que otras disciplinas de la ingeniería se han empleado para mejorar drásticamente el desarrollo y mantenimiento de nuevos productos. Para construir la realidad del futuro, las organizaciones deben cambiar la manera en que ven al desarrollo de software. Y esto implica dos cosas:

- Adoptar un modelo de ingeniería moderno para el desarrollo del software.
- Utilizar herramientas de diseño automático y generación automática de código.

Adoptar un Modelo de Ingeniería Moderno para el Desarrollo de Software

La ingeniería ha existido en sus diversas formas durante cientos de años y en algunos casos, como el de la ingeniería civil y la arquitectura, durante miles de años. Como resultado, estas disciplinas han creado sólidos métodos y herramientas para documentar e implemenmtar sus diseños. Durante cientos de años han existido métodos estandarizados para documentar los diseños de manera que otros pudieran implementarlos. En el siglo veinte se dieron en llamar "proyectos estándar". Estos estándares, desarrollados durante décadas para asegurar que lo que se había diseñado se contruyera exactamenmte como había sido diseñado, se beneficiaron mucho del poder de las computadoras.

Con el tiempo, los ingenieros en computación pudieron tomar lo que los diseñadores hicieron al crear proyectos y transformaron los diseños en programas gráficos CAD. Actualmente, gran parte de la arquitectura e ingeniería modernas comienza con herramientas CAD. Las herramientas CAD han reemplazado a un gran número de los empleados en firmas de ingenieros por un pequeño número de técnicos expertos en CAD. Los que quedaron deben ser capaces de hacer el diseño tradicional pero fundamentalmente poder trabajar fácilmente con las herramientas CAD.

Pero la ingeniería y la arquitectura modernas no se detuvieron simplemente en la automatización del proceso de diseño; esto hubiera equivalido a "dibujar programas" como VISIO o PowerPoint. La tecnología de la moderna ingeniería ha producido la ingeniería asistida por computadora (CAE) y herramientas para la fabricación asistida por computadora (CAM), herramientas que comienzan con los dibujos CAD y después aplican la inteligencia asistida por computadora que simula el objeto diseñado para después producirlo realmente.

El mismo proceso se está empleando hoy en el software, pero este "modelo de ingeniería moderna" no ha sido tan ampliamente adoptado como debería. Este modelo de ingeniería moderna involucra el compromiso con los siguientes principios básicos:

- Separar las áreas principales.
- Dotar a las herramientas de una creciente inteligencia.
- Aumentar la capacidad para relacionarse directamente con el usuario.
- Dejar que la computadora realice el seguimiento de los detalles.

Las disciplinas de la ingeniería moderna *separan a las áreas principales*, requerimientos, ingeniería, diseño y construcción, considerándolas como fases diferentes. Cada una de estas fases tiene una delineación clara y claras interfaces. Estas interfaces cuentan con una documentación estándar. Es debido a la separación de áreas y a las claras interfaces que se ha logrado un gran progreso en la ingeniería computarizada.

Casi desde el principio, los ingenieros comprendieron el potencial que tenían las computadoras para su trabajo. Pudieron imaginarse usando la computadora para manejar "productos virtuales" dentro de ella misma. También pudieron imaginarse partiendo de esos productos virtuales para crear los productos reales. Por supuesto, ellos tenían en cuenta el gran beneficio representado por la reducción de costos de diseño, pero la ganancia real venía de su capacidad para trabajar más y más cerca de sus usuarios. Utilizando la computadora a través del espectro total, desde la conceptualización a la fabricación, los ingenieros han podido reducir drásticamente el costo de desarrollar nuevos productos. Pero también pudieron reducir drásticamente el tiempo involucrado en ir del concepto al producto, algo que ha revolucionado los mercados desde los automóviles hasta los aviones, desde las computadoras a los teléfonos celulares.

Dotar a las herramientas de una creciente inteligencia implica que se cometen menos errores y que las cosas se hacen con la rapidez de las computadoras. Y más aún, cuando deben hacerse cambios, la computadora puede hacerlos en los lugares adecuados, debido a metadatos integrados y sofisticados.

Aumentar la capacidad para relacionarse directamente con el usuario es una de las características fundamentales de las "herramientas inteligentes". Muy pocas personas, aún usuarios de negocios muy inteligentes y experimentados en un campo, pueden visualizar realmente lo que van a obtener. Es por este motivo que los ingenieros y arquitectos han invertido muchísimo tiempo y dinero durante décadas en la construcción de modelos. Cuanto más real sea el modelo de un nuevo edificio o de un nuevo auto o aparato de televisión, más sencillo será para los futuros compradores decir si les gustará o no cuando realmente se construido. Las herramientas de diseño por computadora son usadas universalmente en la actualidad para ayudar a los usuarios finales a "ver" lo que van a obtener una vez que sea producido.

Dejar que la computadora realice el seguimiento de los detalles es otro de los principios básicos de las herramientas de diseño de la ingeniería y la arquitectura modernas. Durante cientos de años, por ejemplo, los arquitectos e ingenieros han podido producir dibujos en perspectiva realistas de sus productos mediante diseños de ingeniería regular, pero el desarrollo de diseños en perspectiva lleva mucho tiempo. Como resultado, solo uno o dos de estos diseños eran presentados rutinariamente en un ciclo de ingeniería. Actualmente es posible construir objetos tridimensionales realistas instantáneamente en la computadora y permitir que los usuarios vean estos "edificios o productos virtuales" desde distintas perspectivas. En realidad, en el caso de los edificios, es posible producir "caminatas virtuales" a través de ellos, como si se estuviera viendo una película de un tour por un edificio completamente nuevo.

Y el software debe poder hacer lo mismo que la ingeniería y la arquitectura. Los desarrolladores de software deben poder definir las nuevas aplicaciones de software en una computadora y después instantáneamente dejar que el usuario vea por sí mismo como luce el software y como se comporta. Y tal como ocurre con el software de ingeniería y arquitectura, los ingenieros en computación deberían poder cambiar aspectos de los requerimientos y el diseño de software y después hacer que la computadora instantáneamente volviera a diseñar y desplegar las aplicaciones modificadas. Y la única forma de hacer esto es contando con el diseño automático y la generación automática de código.

El uso del diseño automático y la generación automática de código

El uso del diseño automático y la generación automática de código es una extensión natural del diseño de código asistido por computadora para construir aplicaciones de software. La tecnología del software se ha vuelto tan compleja que los programadores e incluso los super programadores tienen dificultades para mantener todas las opciones apropiadas en su cabeza. Si las organizaciones van a "preparar su futuro", necesitarán sistemas de fácil

desarrollo, libres de errores y fáciles de mantener. Esto se torna aún más importante en el mundo de Sarbanes-Oxley. Actualmente, más que nunca antes, las organizaciones van a tener que probar que sus sistemas funcionan y que sus controles están libres de fallas. Para hacer esto, deberán contar cada vez más con el diseño automático y la generación automática de código.

Los generadores de código han existido durante décadas, pero en la mayoría de los casos han estado limitados por sus metadatos limitados. *Para que los generadores de código sean realmente efectivos deben basarse en un grupo muy rico de metadatos que unifique los datos y las reglas del negocio consistentemente*. La generación de código de última generación usa bases de conocimiento que contienen metadatos muy ricos, lo suficientemente ricos como para soportar a todas las relaciones que tienen lugar en una aplicación compleja.

Durante décadas, los programadores han argumentado que la solución para construir software complejo involucra lenguajes aún más sofisticados y complejos. La experiencia ha demostrado que ningún lenguaje, sistema operativo ni sistema de administración de bases de datos podrá resolver realmente la complejidad del problema. La complejidad del software tiene que ver con el conocimiento de todas las piezas relevantes y las relaciones entre ellas. Los generadores de código correctos trabajan de la mano con herramientas de diseño automático que, a su vez, son alimentadas por ricas bases de datos de metadatos.

Probablemente el aspecto más significativo de una verdadera solución de "futureproofing" es la disponibilidad de bases de conocimiento de todos los sistemas y de toda la empresa. Hay demasiadas herramientas de software trabajando solo a nivel del programa. Pero las programaciones sofisticadas siempre tienen que ver con sistemas y frecuentemente con múltiples sistemas. El diseño de una sola tabla para un solo programa no es problema. Pero diseñar una base de datos que trabajará con cientos (o miles) de programas es algo diferente. El mismo dato, por ejemplo, puede existir en muchas pantallas y muchos programas dentro de múltiples aplicaciones. La misma formula, por ejemplo, puede ser ejecutada cientos de veces en miles de programas. Una rica base de conocimientos de metadatos hace el seguimiento de todos estos usos, y cuando se hacen cambios en uno de ellos, esto se refleja en todo el sistema. Lo mismo ocurre con los cambios en la base de datos. Una rica base de datos de metadatos efectúa el seguimiento de todas las relaciones y de todos los atributos individuales, tablas y reglas del negocio.

Una rica base de conocimiento de metadatos es en realidad una base de conocimientos sofisticada que se actualiza constantemente para reflejar los últimos cambios. Y más importante aún: cuando un hecho o una relación cambian, la base de conocimiento debe

actualizarse consistentemente para asegurar que se efectúen todos los cambios importantes. El diseño automático y la generación automática de código son simples aplicaciones muy sofisticadas que corren en una base de conocimiento consistente.

Lentamente, la industria del software está comprendiendo este hecho. El reciente interés por la Model Driven Architecture (MDA) es un reconocimiento de que debe haber un vínculo entre las definiciones del negocio y el código. Hasta la fecha, MDA está en una etapa temprana en donde muchas, o casi todas, las transiciones se hacen todavía manualmente. Para ser una solución completa, MDA debe evolucionar y convertirse en una verdadera solución de software como CAD/CAE/CAP/CAM.

MDA está muy lejos, años y quizás décadas en el futuro. No obstante, hay soluciones de futureproofing que ya existen y que se han venido utilizando para construir miles de aplicaciones multi-plataforma y multi-base de datos a gran escala. Y entre ellas, la solución de más larga historia de éxito se llama GeneXus.

La Combinación del Negocio con el Conocimiento Tecnológico

Durante la década de los setenta y los ochenta, la mayoría de las aplicaciones eran desarrolladas por organizaciones que intentaban utilizar la tecnología para su propio beneficio. Estos sistemas eran muy adaptados, frecuentemente escritos para computadoras de mainframe, e involucraban la implementación de los procesos centrales del negocio de esas organizaciones. Durante la década del noventa, la tendencia en el desarrollo de aplicaciones cambió hacia la compra de paquetes de software en lugar de la construcción (desarrollo) de aplicaciones centrales en la empresa. Esta tendencia creó algunas disyuntivas interesantes para las organizaciones que intentaban llevar la delantera a su competencia. Si miran cuidadosamente la figura 1 notarán algunos de los problemas.

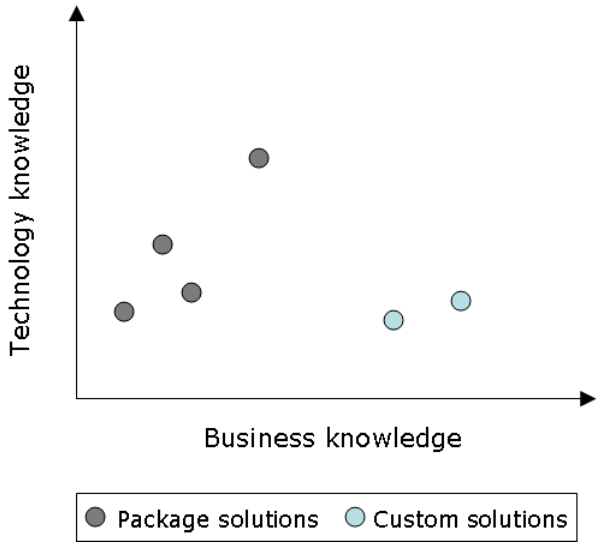

Figura 1 – Negocio versus conocimiento tecnológico

En general, las aplicaciones del cliente (puntos claros en la figura 1) tienden a contener un conocimiento superior del negocio, en tanto que las soluciones de paquete (puntos grises) tienden a contar con una tecnología mejor y más actual. En el pasado, las aplicaciones desarrolladas por el cliente, que contaban con un conocimiento profundo de las reglas del negocio y los procesos del negocio exclusivos de cada organización, tenían problemas para mantenerse al día con los cambios tecnológicos. Por ejemplo, bases de datos relacionales, Windows, cliente/servidor, Internet, soluciones de Paquetes[1]. Y las más nuevas tienen exactamente el problema inverso. Si bien usan siempre la última tecnología, tienen problemas para construir bases de conocimiento específicas en sus herramientas.

Como resultado de esta situación, las organizaciones enfrentaron un dilema: intentar construir sus propias aplicaciones de actualización para hacer modificaciones importantes a los paquetes que habían comprado y lograr que tuvieran un máximo de flexibilidad para adaptarse a su negocio, o adquirir un paquete de software y adaptar su negocio para que fuera apto para el paquete. Resulta claro que para "preparar el futuro" de su organización

[1] En el largo plazo, las aplicaciones de paquete tienen el mismo problema con la tecnología, en general, que las aplicaciones del cliente: se vuelven dependientes de su tecnología. Las empresas de paquetes que han tenido éxito deben luchar con las nuevas tecnologías tanto o más que sus clientes. Las empresas de paquetes han invertido mucho en nuevas técnicas de desarrollo que prometen minimizar el impacto de los cambios tecnológicos, pero solo han podido ser algo más exitosas porque existe un factor de mercado importante. Las empresas de paquetes de software no pueden darse el lujo de demostrar que no están al día con la tecnología.

de manera efectiva, deben encontrarse algunas soluciones que satisfagan tanto la necesidad de adaptarse rápidamente a los cambios del negocio como la necesidad de adaptarse rápidamente a los cambios tecnológicos. Debido a que las soluciones de paquete deben soportar miles de negocios en cientos de mercados diferentes, las mismas son limitadas en términos de sus posibilidades de adaptación a una sola organización o incluso a un solo mercado.

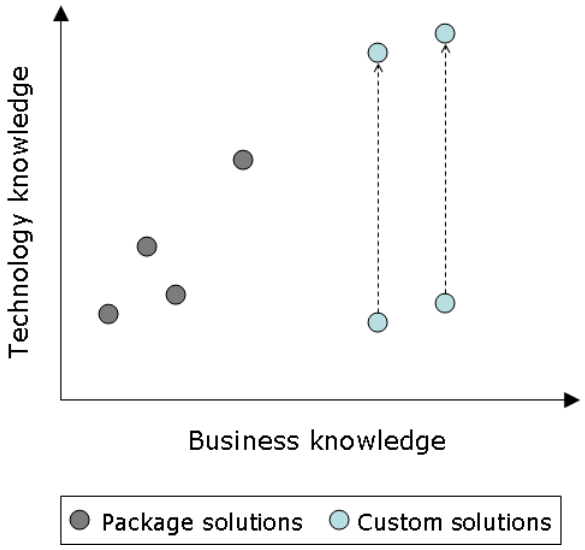

Figura 2 – Conocimiento del negocio versus conocimiento de la tecnología (la solución óptima)

La solución óptima para las organizaciones sería que pudieran de alguna manera construir aplicaciones que se pudieran ir adaptando sin esfuerzos tanto a las nuevas necesidades del negocio como a las nuevas tecnologías (ver figura 2). A largo plazo, el desarrollo verdaderamente ágil supondría que las organizaciones pudieran tener un software que estuviera en el ángulo superior derecho del diagrama conocimiento del negocio versus conocimiento tecnológico. Para que las organizaciones se vuelvan realmente ágiles, necesitarán moverse a este cuarto cuadrante de la "alta tecnología" y la "alta funcionalidad del negocio". En la próxima sección describiremos como es posible esto en la actualidad con las herramientas de última generación.

GeneXus: Una verdadera solución de futureproofing del siglo XXI

GeneXus es un ambiente de desarrollo de sistemas de última generación que se ha estado desarrollando desde hace casi dos décadas. Involucra la reunión de una serie de tecnologías avanzadas que incluyen: Inteligencia Artificial, diseño automático de bases de datos, diseño de interfase de usuario y diseño de las reglas del negocio. Como resultado de esta integración innovadora de tecnologías claves, GeneXus ha sido capaz de hacer evolucionar el ambiente de desarrollo de software permitiendo a las organizaciones cambiar de una arquitectura centralizada (ej. AS400) a una arquitectura cliente/servidor y de allí a una arquitectura orientada al servicio y basada en la red, manteniendo y actualizando el conocimiento crítico de su negocio. Esta cadena de éxitos se basa en algunas percepciones muy importantes.

- La primera de estas percepciones fue que era posible deducir una estructura de diseño de base de datos eficiente y normalizada a partir de un grupo de descripciones de estructura de datos de transacciones de negocios individuales y generar automáticamente las condiciones de navegación necesarias para correr en esta estructura de base de datos.
- La segunda de estas percepciones fundamentales advirtió que, al realizarse cambios en las especificaciones de las estructuras de datos mencionadas anteriormente, o al agregarse nuevas estructuras de datos, es posible rediseñar y renormalizar la base de datos, convertir la vieja estructura de la base de datos a la nueva y regenerar todo el código que opera en la nueva base de datos. [2]
- La tercera de estas percepciones fundamentales fue que este proceso puede llevarse a cabo en todos los lenguajes principales y en todas las principales bases de datos relacionales comerciales desarrollando un motor de generación común y adaptando cada conjunto de generadores a una plataforma dada.
- La cuarta de estas percepciones fue que cualquier sistema que pueda realmente mantenerse al día con los cambios del negocio y los cambios tecnológicos debería generar 100% del código para la mayoría de las aplicaciones.

Adoptando esta estrategia radical, los investigadores y desarrolladores de GeneXus pudieron crear un set de herramientas que opera igual que lo hace el software CAD/CAE/CAM en ingeniería y arquitectura. Como resultado, las organizaciones que usan GeneXus han podido

[2] Los desarrolladores saben que el diseño de la base de datos era lo más difícil de cambiar en cualquier sistema. La razón es que para la mayoría de los ambientes de desarrollo, una vez que los programadores comienzan a escribir código en un diseño de base de datos dado, el costo de cambiar ese diseño de base de datos sube drásticamente por todo el código de navegación de la base de datos que debe ser reescrito. GeneXus resuelve este problema generando él mismo todo el código de navegación.

crear aplicaciones de trabajo casi instantáneamente, en minutos o segundos, en lugar de en días o semanas. Y lo que es más importante, han podido revisar estas aplicaciones con la misma rapidez. Mucho antes de que nadie hubiera dado un nombre al desarrollo ágil, los desarrolladores de GeneXus ya lo estaban llevando a cabo.

Pero la prueba de toda tecnología está en sus resultados, no en sus reclamos. Actualmente, hay más de 30.000 licencias GeneXus en todo el mundo generando un estimado de 2 mil millones de líneas de código por año en una amplia gama de plataformas y bases de datos. La aplicación promedio tiene las siguientes características:

- Casi el 100% de generación automática.
- Entre 1 y 5 millones de líneas de código generado.
- La capacidad de manejar más de 500 tablas relacionales en una sola aplicación.
- Migración de una plataforma a otra con el mínimo esfuerzo.

El desarrollo de software resultó ser una actividad mucho más complicada de lo que nadie hubiera pensado en sus primeros tiempos. Mientras que con el correr de los años se ha venido prestando una enorme atención a los programas y al desarrollo de programas, mucha menos atención se ha prestado al desarrollo de sistemas integrados. Se ha asumido que los sistemas son simples colecciones de programas individuales. Pero los sistemas son mucho más complicados que los programas individuales, y en realidad descansan en un grupo común de datos y reglas del negocio. Es poco frecuente que un programa tenga más de 20 mil líneas de código, pero no es poco frecuente que un sistema incluya 10 e incluso 20 millones de líneas de código en la actualidad.

Históricamente, los desarrolladores de sistemas han intentado resolver el problema del desarrollo de sistemas mediante una interminable variedad de esquemas de gestión de proyectos. Esto implicó emplear enormes cantidades de tiempo en el diseño de sistemas y bases de datos y en las interfaces de los sistemas, de modo tal que los programas individuales pudieran ser escritos individualmente manteniendo la “integridad del sistema”.

GeneXus ha tenido un enorme éxito en su enfoque de la generación de código, porque trata a los sistemas como la unidad de trabajo básica. Una base de conocimiento dada, por ejemplo, incluye información sobre todos los datos en una aplicación dada (sistema). Por esta razón, el generador de código puede ser mucho más inteligente en la manera en que genera el código, y lo que es más importante, en la manera en que maneja y actualiza los datos.

El Futuro de Futureproofing

El mundo de los negocios de hoy en día es tan volátil como no lo ha sido nunca. La globalización, la intensa competencia y las nuevas tecnologías han contribuido en su conjunto al surgimiento de una urgente necesidad por parte de las organizaciones de ser cada vez más ágiles. Futureproofing cuenta con dos elementos fundamentales: (1) una concepción más seria sobre el futuro y sobre el impacto de las decisiones actuales en el futuro, y (2) la adquisición de herramientas y tecnología que permiten a las organizaciones alcanzar el máximo de flexibilidad.

Artech, desarrollador de GeneXus, cuenta con especialistas en "software de Futureproofing". Hace casi dos décadas, Artech vio la necesidad de una herramienta de desarrollo de software totalmente diferente, que permitiera a los desarrolladores de software pasar la mayor parte de su tiempo enfocados en las nuevas posibilidades para sus negocios en lugar de pasarlo ocupándose en adaptarse a los cambios tecnológicos. Artech también comprendió que, a largo plazo, los sistemas que han permitido un desarrollo incremental (ágil) serán mucho más productivos que aquellos que han permanecido en sus enfoques tradicionales de desarrollo en cascada.

También ha sido crédito de Artech permanecer en la vanguardia todo el tiempo, al reconocer la importancia de integrar cada nueva tecnología a medida que va surgiendo, como ha sido el caso de workflow y la tecnología inalámbrica, para mantenerse al tanto de los cambios tecnológicos. Pensando en el futuro, Artech ha hecho del "futureproofing" su principal negocio. Como resultado, ha producido un set de herramientas que puede ser usada por los negocios en todo el mundo para incrementar su tecnología, sacar ventaja a sus competidores y realmente "preparar el futuro de sus organizaciones."

"Probablemente en una o dos décadas aparezcan otras soluciones de Futureproofing.

Pero GeneXus está listo y disponible hoy, y hoy puede mostrarnos el camino."

Breogán Gonda, Fundador y Presidente de ARTech