GeneXus™

Grow thru knowledge

Visión General de GeneXus

Última actualización: 2012

GeneXus international
www.genexus.com

MONTEVIDEO - URUGUAY	Av. Italia 6201 - Edif. Los Pinos, P1	(598) 2601 2082
	Ruta 8 Km. 17.500 - Edif. @3 Of. 205	(598) 2518 2870
CHICAGO - USA	1143 W Rundell PL, Suite 200	(1 312) 836 9152
SÃO PAULO - BRASIL	Rua Samuel Morse 120 Conj. 141	(55 11) 5502 6722
CIUDAD DE MÉXICO - MÉXICO	Calle Leibnitz N° 20, desp. 801	(52 55) 5255 4733
TOKYO - JAPAN	Tk Gotanda Bldg. 303-2	(81 3) 5793 5481

Introducción

GeneXus es una herramienta inteligente, desarrollada por Artech, cuyo objetivo es asistir al analista y a los usuarios en todo el ciclo de vida de las aplicaciones.

El diseño y prototipo son realizados y probados en un ambiente Windows, Windows NT/2000/XP/7. Cuando el prototipo es totalmente aprobado por sus usuarios, la base de datos y los programas de aplicación son generados y/o mantenidos en forma totalmente automática, para el ambiente de producción.

La idea básica de GeneXus es automatizar todo aquello que es automatizable: normalización de los datos y diseño, generación y mantenimiento de la base de datos y de los programas de aplicación. De esta manera se evita que el analista deba dedicarse a tareas rutinarias y tediosas, permitiéndole poner toda su atención en aquello que nunca un programa podrá hacer: **entender los problemas del usuario**.

Como un subproducto, GeneXus ofrece una documentación rigurosa, autosuficiente y permanentemente actualizada.

Este documento tiene como objetivo ilustrar al lector sobre GeneXus y los problemas que resuelve.

Contenido de las siguientes secciones:

- El problema teórico: en este capítulo se hace una descripción comparada de las metodologías tradicionales de desarrollo de sistemas y el desarrollo incremental.
- Una implementación del desarrollo incremental: GeneXus.
- Características únicas de GeneXus.
- Quienes son los usuarios de GeneXus

El problema teórico

Metodologías tradicionales de desarrollo y problemas asociados

La forma tradicional de desarrollar aplicaciones parte de una premisa básica: **es posible construir un modelo de datos estable de la empresa**. Basándose en esa premisa, la primera tarea que se encara es el análisis de datos, donde se estudia la realidad en forma abstracta y se obtiene como producto el modelo de datos de la empresa. La segunda tarea es diseñar la base de datos. Es muy sencillo diseñar la base de datos partiendo del modelo de datos ya conocido.

Una vez que se ha estudiado la realidad desde el punto de vista de los datos, se hace lo propio desde el punto de vista de las funciones (análisis funcional). Sería deseable que el estudio de la realidad tuviera como producto una especificación funcional que dependiera sólo de dicha realidad. Lo que se hace en las metodologías más usadas, sin embargo, es obtener una especificación funcional que se refiere a los archivos de la base de datos (o bien a las entidades del modelo de datos, lo que es esencialmente equivalente).

Una vez que se tiene la base de datos y la especificación funcional, se pasa a la implementación de las funciones, existiendo tradicionalmente para ello varias opciones (lenguajes de 3ª. o 4ª generación, generadores, interpretadores).

Sin embargo, todas las formas de implementación vistas tienen un problema común: parten de la enunciada premisa: **es posible construir un modelo de datos estable de la empresa**, y esta premisa es **falsa**.

Realmente es **imposible** hacer, de una forma abstracta, un modelo de datos detallado de la empresa y con el suficiente nivel de detalle y objetividad, porque nadie la conoce como un todo.

Por ello es necesario recurrir a múltiples interlocutores, y cada uno de ellos proyecta sobre el modelo, su propia subjetividad. Una consecuencia de esto es que, durante todo el ciclo de vida de la aplicación, se producen cambios en el modelo.

Pero aún si se considerara la situación ideal, donde se conocen exactamente las necesidades y, entonces, es posible definir la base de datos óptima, el modelo no podrá permanecer estático porque deberá acompañar la evolución de la empresa.

Todo esto sería poco importante, si la especificación funcional y la base de datos fueran independientes. Sin embargo, dado que la especificación funcional se refiere a la base de datos, las inevitables modificaciones en ésta implican la necesidad de modificaciones (manuales) en aquella.

La mayor consecuencia de lo anterior está constituida por los muy altos costos de mantenimiento: en la mayoría de las empresas que trabajan de una manera convencional se

admite que el 80% de los recursos que teóricamente están destinados al desarrollo, realmente se utilizan para hacer mantenimiento de las aplicaciones ya implementadas.

Cuando se trata de aplicaciones grandes la situación es aún peor: este mantenimiento comienza mucho antes de la implementación, lo que hace que los costos de desarrollo crezcan en forma hiperlineal con respecto al tamaño del proyecto.

Dado que es muy difícil, en este contexto, determinar y propagar las consecuencias de los cambios de la base de datos sobre los procesos, es habitual que, en vez de efectuarse los cambios necesarios, se opte por introducir nuevos archivos redundantes, con la consiguiente degradación de la calidad de los sistemas y el incremento de los costos de mantenimiento.

Desarrollo Basado en Conocimiento y Metodología incremental

En los últimos años se ha hablado mucho en la industria de **Knowledge Management** y, dentro de este rótulo se han colocado muchas cosas que están bien distantes del Desarrollo Basado en Conocimiento a que nos queremos referir aquí.

Generalmente la industria se ha referido a maneras de organizar y/o acceder el conocimiento para ser utilizado de una forma tradicional por los seres humanos. Se trata de una versión actualizada, utilizando la tecnología actualmente disponible, de los libros (y que es de enorme utilidad para toda la humanidad): accedemos a un cierto conocimiento leyendo un libro y, en nuestra mente, hacemos razonamientos sobre ese conocimiento lo que, eventualmente, determina acciones. Los buscadores de texto inteligentes que están disponibles desde hace pocos años hacen que este conocimiento sea cada vez más accesible a los seres humanos.

Como característica general, este conocimiento no es "entendible" por una máquina y, en consecuencia, no es operable. Adicionalmente, como el razonamiento de los seres humanos puede lidiar razonablemente (dentro de ciertos límites) con la ambigüedad y, aún, con la inconsistencia, este conocimiento muchas veces no es riguroso.

Entonces, es bueno restringir el concepto de conocimiento que utilizaremos en nuestro Desarrollo Basado en Conocimiento. Se trata de conocimiento que cumple las siguientes condiciones:

- Riguroso
- Representable en forma objetiva
- Operable

Una nueva manera de resolver el problema del desarrollo de sistemas pasa por la sustitución de la premisa básica enunciada: asumir que **no es posible construir un modelo de datos estable de la empresa** y, en cambio, utilizar una **filosofía incremental y hacer un Desarrollo Basado en Conocimiento**. Un esquema incremental parece muy natural: no se encaran grandes problemas, sino que se van resolviendo los pequeños problemas a medida que se presentan.

¿Cuál será la repercusión de este tipo de esquema sobre los costos de mantenimiento?

Si se utilizaran, con este enfoque, las metodologías anteriormente reseñadas, esa repercusión sería muy grande: el modelo de datos se modificaría constantemente y los costos de mantenimiento serían aún mucho mayores que los enunciados.

Puede verse, sin embargo, lo siguiente: no se conoce la base de datos pero, cada usuario, conoce muy bien las visiones de los datos que él utiliza cotidianamente.

Esas visiones de los datos pueden ser de varios tipos: pantallas, diálogos, flujos de procesos, listados, etc. que componen el aspecto exterior de la aplicación: Aquello que es tangible para el usuario.

¿Cómo puede ayudar el conocimiento de estas visiones a obtener el modelo de datos?

¿Puede transformarse el asunto en un problema lógico/matemático? Si ello fuera posible, la lógica y la matemática podrían brindar una amplia gama de recursos para ayudar a resolverlo automáticamente y, como consecuencia, se simplificaría mucho la tarea del analista. Una reflexión interesante es la siguiente: si se conociera la base de datos, las visiones de los datos que tienen los diferentes usuarios deberían poder derivarse de ella. O, dicho de otra manera, la base de datos debe satisfacer a todas las visiones conocidas. Puede demostrarse que, dado un conjunto de visiones de datos de usuarios, existe siempre una base de datos mínima que las satisface, la cual, además, es única. En este estado, el problema se ha transformado en un problema lógico/matemático y, entonces es preciso resolverlo, para hallar esa base de datos.

¿Cómo se implementa esta teoría?

Se trata de capturar el conocimiento que existe en las visiones de los usuarios, y sistematizarlo en una base de conocimiento (todo ello en forma automática). La característica fundamental de esta base de conocimiento, que la diferencia de los tradicionales diccionarios de datos, es su capacidad de inferencia: se pretende que, en cualquier momento, se puedan obtener de esta base de conocimiento, tanto elementos que se han colocado en ella, como cualquier otro que se pueda inferir a partir de ellos.

Si este objetivo se logra, la base de datos y los programas de aplicación pasan a ser transformaciones determinísticas de dicha base de conocimiento y ello permite:

- Generarlos automáticamente
- Ante cambios en las visiones de los usuarios determinar el impacto de dichos cambios sobre datos y procesos y propagar esos cambios generando:
    - los programas necesarios para convertir los datos;
    - los programas de la aplicación afectados por los cambios;
    - Aquellos programas de aplicación que no han sido afectados por los cambios pero que, ahora, podrían ser sustituidos por otros más eficientes.

GeneXus: El desarrollo incremental hecho realidad

GeneXus implementa esta teoría.

GeneXus es una herramienta que parte de las "visiones de los usuarios"; captura su conocimiento y lo sistematiza en una base de conocimiento. A partir de su base de conocimiento, GeneXus es capaz de diseñar, generar y mantener de manera totalmente automática la estructura de la base de datos y los programas de la aplicación (los programas necesarios para que los usuarios puedan operar con sus visiones).

GeneXus está construido sobre un sólido fundamento matemático.

Si nos preguntamos cuál es la principal fortaleza de GeneXus, la respuesta es: **una excelente administración del conocimiento de los sistemas de negocios**.

GeneXus trabaja con conocimiento puro, lo que le permite realizar varias cosas: generar programas (software tradicional), entender ese conocimiento de los seres humanos (no necesita documentación adicional –que nunca estaría actualizada), y operar automáticamente con ese conocimiento (integrándolo con otro proveniente de otras fuentes, difundiéndolo, otorgando licencias a terceros para que lo integren a sus aplicaciones). En definitiva, GeneXus hace posible el "**negocio del conocimiento**", como un paso adelante respecto al "**negocio del software**".

Otra ventaja del trabajo con conocimiento puro es la posibilidad de generar aplicaciones para múltiples plataformas y múltiples arquitecturas y, muy especialmente, el poder contar con cierto tipo de "seguro" ante los cambios tecnológicos: por ejemplo, los usuarios GeneXus que desarrollaron aplicaciones hace 8 o 10 años para AS/400 con pantallas de texto y tecnologías bastante primitivas, pueden ahora aprovechar el conocimiento sobre el desarrollo de esas aplicaciones que GeneXus salvó para desarrollar aplicaciones Java y/o .NET con facilidad, a pesar de que cuando aquellas aplicaciones fueron desarrolladas, nadie pudo pensar en algo tan diferente respecto al ambiente en el cual ellas trabajaban.

Cuando una aplicación se desarrolla con GeneXus la primera etapa consiste en hacer el **Diseño** de la misma registrando las visiones de usuarios (a partir de las cuales el sistema captura y sistematiza el conocimiento).

Posteriormente se pasa a la etapa de **Prototipación** en donde GeneXus genera la base de datos (estructura y datos) y programas para el ambiente de prototipo. Una vez generado el Prototipo debe ser puesto a prueba por el analista y los usuarios.

Si durante la prueba del Prototipo se detectan mejoras o errores se retorna a la fase de Diseño, se realizan las modificaciones correspondientes y se vuelve al Prototipo. Llamaremos a este ciclo de Diseño/Prototipo.

Una vez que el Prototipo está aprobado, se pasa a la etapa de **Implementación**, en donde GeneXus genera, también automáticamente, la base de datos y programas para el ambiente de producción.

En resumen, una aplicación comienza con un Diseño, luego se Prototipa, luego se Implementa o pone en producción y en cualquiera de los pasos anteriores se puede regresar al Diseño para realizar modificaciones.

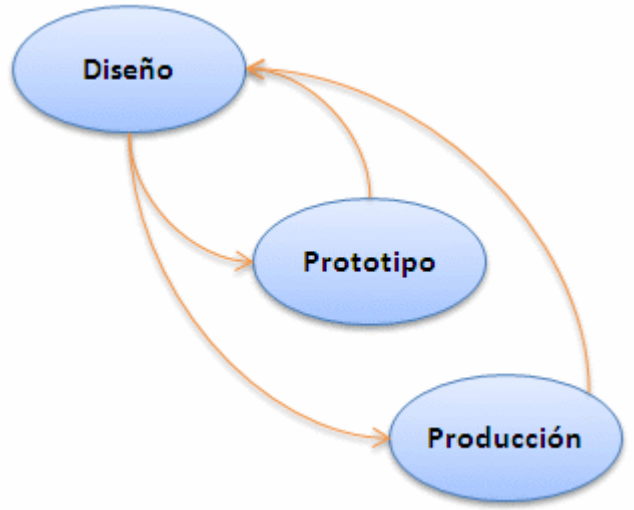

Figura 1 - Ciclos Diseño-Prototipación y Diseño-Producción

A continuación se describe cada una de estas tareas.

Diseño

Esta tarea es realizada conjuntamente por el analista y el usuario, y consiste en identificar y describir las visiones de datos de los usuarios.

El trabajo se realiza en el ambiente del usuario. Este esquema permite trabajar con un bajo nivel de abstracción, utilizando términos y conceptos que son bien conocidos por el usuario final.

Una consecuencia muy importante, es que la actitud del usuario se transforma en francamente participativa. El sistema pasa a ser una obra conjunta y, como el usuario sigue permanentemente su evolución, su calidad es mucho mejor que la habitual.

De acuerdo a lo visto, GeneXus captura el conocimiento por medio de visiones de objetos de la realidad del usuario. Los tipos de objetos soportados por GeneXus son, entre otros: **Transacciones, Reportes, Procedimientos, Work Panels, Web Panels, Data Views, Transacciones de BI**.

La tarea de diseño consiste, fundamentalmente, en identificar y describir estos objetos. A partir de estas descripciones, y automáticamente, GeneXus sistematiza el conocimiento capturado y va construyendo, en forma incremental, la Base de Conocimiento.

Esta Base de Conocimiento es un repositorio único de toda la información del diseño, a partir de la cual GeneXus crea el modelo de datos físico (tablas, atributos, índices, redundancias, reglas de integridad referencial, etc.), y los programas de aplicación.

Así, la tarea fundamental en el análisis y diseño de la aplicación se centra en la descripción de los objetos GeneXus.

Veamos en detalle las clases de objetos GeneXus más importantes:

- **Transacciones**

  Una transacción es un proceso interactivo o pantalla (Win o Web) que permite a los usuarios crear, modificar o eliminar información de la base de datos.

  Ejemplos:

  - Pantalla para crear, modificar o eliminar los Clientes de la Empresa.
  - Pantalla de facturación: proceso que permite a un usuario crear facturas e incluso imprimirlas.

  Una pantalla permite al usuario tomar diferentes acciones como insertar, actualizar, eliminar, imprimir sin tener que volver al menú para hacerlo.

  La transacción tiene elementos esenciales como la estructura de datos de la pantalla, reglas del negocio y fórmulas y elementos cosméticos como la forma de las pantallas (en este caso el desarrollador puede darle con los editores disponibles la forma que quiera u optar por utilizar la automáticamente inferida por el sistema).

- **Reportes**

  Un reporte es un proceso que permite visualizar los datos de la base de datos. La salida del listado puede ser enviada a pantalla o a la impresora (y con ello tenemos un listado convencional).

  Con este objeto se pueden definir desde listados simples (por ejemplo, listar los clientes) hasta muy sofisticados, en donde existan varios cortes de control, múltiples lecturas a la base de datos y parametrización.

  Un reporte no puede actualizar la base de datos.

  Se dispone además de una herramienta GXquery para realizar reportes dinámicos sobre la base de datos. Para más información ver: www.genexus.com/gxquery

- **Procedimientos**

  Este objeto tiene todas las características de los Reportes, y además permite actualizar la base de datos. Los Procedimientos son comúnmente usados para tres tipos de procesos:

  - *Procesos batch de actualización*. Por ejemplo: eliminar todas las facturas de fecha anterior a una fecha dada y que ya fueron pagadas

- *Subrutinas de uso general*. Por ejemplo: rutina de monto escrito en donde, dado un importe se devuelve un literal con el importe en letras (1010 => 'Mil diez')
- *Procesos a ejecutar en un servidor de aplicaciones o servidor de base de datos*: procesos (generalmente escritos en Java o .NET) para una Multi Tier Architecture, para ser ejecutados en un servidor de aplicaciones o de bases de datos.

- **Work Panels**

  Un Work Panel es una pantalla que permite al usuario realizar consultas interactivas a la base de datos. Cuanto más los usuarios utilizan el computador para su trabajo, se torna más necesaria la utilización de diálogos sofisticados, que le permitan sentarse a pensar frente al mismo. Los Work Panels permiten diseñar este tipo de diálogos del usuario.

  Por ejemplo: Un Work Panel que muestra la lista de clientes y que permite (a elección del usuario) ver cuales son sus facturas o su deuda.

- **Web Panels**

  Son similares al grupo de Work Panels pero requieren un navegador de aplicaciones (Browser) para ser ejecutados en ambientes Internet/Intranet/Extranet.

- **Data Views**

  Permiten considerar correspondencias entre tablas de bases de datos preexistentes y tablas GeneXus y tratar aquellas con la misma inteligencia como si fueran objetos GeneXus.

Desarrollo basado en el conocimiento

Partiendo de los objetos descritos, el modelo de datos físico es diseñado con base en la Teoría de Bases de Datos Relacionales, y asegura una base de datos en tercera forma normal (sin redundancia). Esta normalización es efectuada automáticamente por GeneXus.

El analista puede, sin embargo, definir redundancias que, a partir de ello, pasan a ser administradas (controladas o propagadas, según corresponda), automáticamente por GeneXus.

El repositorio de GeneXus mantiene las especificaciones de diseño en forma abstracta, o sea que no depende del ambiente objeto, lo que permite que, a partir del mismo repositorio, se puedan generar aplicaciones funcionalmente equivalentes, para ser ejecutadas en diferentes plataformas.

Múltiples plataformas / arquitectura de múltiples capas

Como consecuencia de lo anterior es posible, por ejemplo, que un usuario de una aplicación IBM AS/400 centralizada desarrollada 100% con GeneXus, quizás hace 15 años, pueda hacerla funcionar total o parcialmente en un ambiente JAVA o .NET sin tener que modificar los objetos originales.

En los últimos años se ha vuelto imperioso generar aplicaciones multi-plataforma, es decir, de ejecutar la misma aplicación en varios ambientes. Por ejemplo, la aplicación de un sistema bancario debe poder correr en un servidor iSeries o Linux en la oficina central y en una red de PCs en las sucursales del banco.

Pero eso no ha sido todo; con el uso progresivo de los ambientes Cliente/Servidor e Internet/Intranet/Extranet, ha surgido una nueva necesidad: la misma aplicación debe tener alguna de sus partes corriendo en una plataforma determinada y otras corriendo en otras plataformas. En estos casos, es también indispensable que exista una correcta intercomunicación entre las distintas partes de la plataforma.

El desarrollar aplicaciones con GeneXus da la posibilidad de dividir una aplicación de manera tal que cada parte puede ser ejecutada en una plataforma diferente, utilizándose el lenguaje más apropiado para generar los programas en cada una de estas plataformas. Esto ha dado lugar al advenimiento de las arquitecturas de múltiples capas, que a la vez optimizan el uso de los recursos disponibles, como así también las nuevas tecnologías.

Prototipado

En las tareas de diseño están implícitas las dificultades de toda comunicación humana:

- El usuario olvida ciertos detalles.
- El analista no toma nota de algunos elementos.
- El usuario se equivoca en algunas apreciaciones.
- El analista interpreta mal algunas explicaciones del usuario.

Pero, además, la implementación de sistemas es, habitualmente, una tarea que insume bastante tiempo, por lo que:

- Como muchos de estos problemas sólo son detectados en las pruebas finales del sistema, el costo (tiempo y dinero) de solucionarlos es muy grande.
- La realidad cambia, por ello, no es razonable pensar que se pueden congelar las especificaciones mientras se implementa el sistema.
- la consecuencia de la congelación de las especificaciones, es que se acaba implementando una solución relativamente insatisfactoria.

El impacto de estos problemas disminuiría mucho si se consiguiera probar cada especificación, inmediatamente, y saber cual es la repercusión de cada cambio sobre el resto del sistema.

Una primera aproximación a esto, ofrecida por diversos sistemas, es la posibilidad de mostrar al usuario formatos de pantallas, informes, etc. animados por menús. Esto permite ayudar al usuario a tener una idea de qué sistema se le construirá pero, posteriormente, siempre se presentan sorpresas.

Una situación bastante diferente sería la de poner a disposición del usuario para su ejecución, inmediatamente, una aplicación funcionalmente equivalente a la deseada, hasta en los mínimos detalles.

Esto es lo que hace GeneXus: **Un prototipo GeneXus es una aplicación completa, funcionalmente equivalente a la aplicación de producción**.

La diferencia entre prototipación y producción consiste en que la primera se hace en un ambiente de microcomputador, mientras que la producción se realiza en el ambiente objeto del usuario (IBM iSeries, servidor Linux, Cliente / Servidor, JAVA, .NET, iOS, Android, BlackBerry

etc. El prototipo permite que la aplicación sea totalmente probada antes de pasar a producción. Durante estas pruebas, el usuario final puede trabajar con datos reales, o sea que prueba, de una forma natural, no solamente formatos de pantallas, informes, etc. sino también fórmulas, reglas del negocio, estructuras de datos, etc.

La filosofía de GeneXus está basada en el concepto conocido como **desarrollo incremental**. Cuando se trabaja en un ambiente tradicional, los cambios en el proyecto hechos durante la implementación y, sobre todo, aquellos que son necesarios luego de que el sistema está implantado, son muy onerosos (y raramente quedan bien documentados). GeneXus resuelve este problema: construye la aplicación con una metodología de aproximaciones sucesivas que permite, una vez detectada la necesidad de cambios, prototiparlos y probarlos inmediatamente por parte del usuario, sin costo adicional.

Implementación

GeneXus genera automáticamente el código necesario para:

- Crear y mantener la base de datos;
- Generar y mantener los programas para manejar los objetos descritos por el usuario.

El proceso de generación puede ser considerado en dos etapas: ESPECIFICACIÓN y GENERACIÓN. La especificación es totalmente independiente del ambiente objeto, pero la generación no. Esto significa que se puede ejecutar el mismo modelo en las diferentes plataformas de ejecución para las que se ha generado y cada una de estas versiones generadas puede ser optimizada de acuerdo con el ambiente en el cual correrá.

Los ambientes y lenguajes más importantes actualmente soportados hasta la fecha de edición de este documento (ver portada) son:

Plataformas

Plataformas de ejecución
JAVA, Microsoft .NET, Microsoft .NET Compact Framework

Sistemas Operativos
IBM OS/400, LINUX, UNIX, Windows NT/2000/2003 Servers, Windows NT/2000/XP/CE y Windows Vista

Internet
JAVA, ASP.NET, Visual Basic (ASP), C/SQL, HTML, Web Services

Móviles iOS, Android, BlackBerry

Bases de Datos
IBM DB2 for iSeries y UDB, Informix, Microsoft SQL Server, MySQL, Oracle y PostgreSQL.

Lenguajes
JAVA, C#, COBOL, RPG, Visual Basic.

Servidores Web
Microsoft IIS, Apache, WebSphere, etc.

Múltiples Arquitecturas
Arquitecturas de múltiples capas, basadas en web, Cliente/Servidor, centralizadas (iSeries), móviles

Para conocer la lista completa de tecnologías soportadas hoy, visite:
http://www.genexus.com/technologies

Además GeneXus ofrece un conjunto de herramientas complementarias para:

- Workflow – GXflow (www.gxflow.com)
- Reporting – GXquery (www.gxquery.com)
- Business Intelligence – GXplorer (www.gxplorer.com)
- Portal Building – GXportal (www.gxportal.com)
- GeneXus Server (http://www.genexus.com/gxserver)
- Gxtest (http://www.genexus.com/gxtest)

Mantenimiento

Esta es una de las características más importante de GeneXus, y la que lo diferencia de manera más clara de sus competidores: el mantenimiento, tanto de la base de datos (estructura y contenido) como de los programas, es totalmente automático.

A continuación se explicará el proceso de mantenimiento, ante cambios en la descripción de algún objeto GeneXus (visión del usuario):

Impacto de los cambios sobre la base de datos

Análisis de impacto

Una vez descritos los cambios de las visiones de usuarios, GeneXus analiza automáticamente cual es el impacto de los mismos sobre la base de datos y produce un informe donde explica como debe hacerse la conversión de los datos y, si cabe, qué problemas potenciales tiene esa conversión (inconsistencias por viejos datos ante nuevas reglas, etc.). El analista decide si acepta el impacto y sigue adelante o no.

Generación de programas de conversión

Una vez que los problemas han sido solucionados o bien se ha aceptado la conversión que GeneXus sugiere por defecto, se generan automáticamente los programas para hacer la conversión (estructura y contenido) de la vieja base de datos a la nueva.

Ejecución de los programas de conversión

A continuación, se pasa al ambiente de ejecución que corresponda (prototipo, producción Internet, producción Cliente / Servidor, etc.) y se ejecutan los programas de conversión.

Impacto de los cambios sobre los programas

Análisis de impacto

GeneXus analiza el impacto de los cambios sobre los programas, y produce un diagnóstico informando qué programas deben generarse o re-generarse y proporcionando también, para el nuevo programa, o bien el diagrama de navegación o bien un pseudo-código, a elección del analista.

Generación de nuevos programas

A continuación el sistema genera o regenera automáticamente todos los programas.

Documentación

Todo el conocimiento provisto por el analista, o inferido por GeneXus, está disponible en un repositorio activo, que constituye una muy completa documentación online, permanentemente actualizada.

La documentación incluye la descripción de objetos específicos e información sobre la base de conocimiento resultante y sobre la base de datos diseñada.

La base de conocimiento de GeneXus no solamente le permite acceder al conocimiento que almacena siempre que el desarrollador lo desee sino que también le habilita el acceso a toda la información inferida lógicamente (una regla de integridad referencial, un mapa de navegación en la base de datos, un análisis de impacto de cambios, referencias cruzadas, diagramas E-R inferidos a partir del conocimiento almacenado, etc.).

Consolidación de varias aplicaciones y reutilización de conocimiento.

Varias aplicaciones pueden ser diseñadas y prototipadas simultáneamente, por diferentes equipos, utilizando GeneXus. Estos equipos pueden intercambiar especificaciones de diseño utilizando el GeneXus Knowledge Manager.

Este modulo le permite hacer lo siguiente automáticamente:

- Comenzar el diseño de una nueva aplicación basada en Objetos del Negocio, Patrones de Software, Dominios, Atributos y/o Estilos de un dominio público (consulte: http://www.gxopen.com.uy).
- Distribuir conocimiento desde una base de conocimiento corporativa a la base de conocimiento de otra aplicación.
- Verificar la concordancia entre la base de conocimiento de una aplicación y la corporativa.
- Consolidar dos aplicaciones (es especialmente útil consolidar el conocimiento de una aplicación dada a la base de conocimiento corporativa).

Esto permite una flexibilidad ideal: el analista trabaja con entera libertad en un ambiente de prototipo, con una pequeña base de conocimiento y, sólo cuando su aplicación está pronta desde el punto de vista del usuario, debe tomarse en cuenta la base de conocimiento corporativa, que generalmente será muy grande.

En ese momento, con poderosas ayudas automáticas, se establece el impacto que tendrá la nueva aplicación, o la modificación de la preexistente, sobre el modelo corporativo y, si es el caso, se hacen los cambios para asegurar la consistencia, de una manera muy simple.

Con este esquema es posible reutilizar, por ejemplo, Bases de Conocimiento licenciadas de terceras partes.

Al comienzo es necesario usar una nomenclatura común entre las diferentes bases de conocimiento involucradas en la consolidación. No obstante, la funcionalidad "Adapt From" le permite definir un mapeo para la conversión de nombres para adaptarse a la nomenclatura objetivo.

También es importante señalar que la casa de software que otorga la licencia sobre la base de conocimientos GeneXus puede mantener partes de la misma en privado; de esta manera podrá permitir su uso automático sin revelar sus fuentes.

Además, es posible que un objeto sea declarado como público o privado. Todos pueden ser usados automáticamente por GeneXus, pero en el caso de los objetos privados, solo el dueño puede ver y/o modificar la fuente de alto nivel de GeneXus.

Es de destacar la característica que permite sobre una misma base de conocimiento generar aplicaciones sobre varios idiomas lo que colabora mucho para que las aplicaciones puedan utilizarse internacionalmente.

Características únicas de GeneXus

GeneXus tiene algunas características únicas que lo distinguen de sus competidores. Entre ellas pueden destacarse:

- El diseño parte de las visiones proporcionadas por los usuarios. Debido a sus actividades diarias, ellos son quienes saben como deben y como no deben funcionar las cosas.
- La descripción de cada objeto es totalmente independiente de la de los demás por lo que, en el caso de que se deba modificar la descripción de uno, ello no implicará la necesidad de modificar manualmente la descripción de cualquier otro. Esta característica exclusiva de GeneXus (ortogonalidad de las descripciones) es la que permite un mantenimiento totalmente automático de las aplicaciones.
- La curva de aprendizaje es corta.
- El diseño, creación y mantenimiento de la base de datos son totalmente automáticos.
- La aplicación (base de datos y programas) tiene siempre, sean cuales sean las modificaciones que haya sufrido, la mejor calidad:
    - La base de datos es siempre la óptima (tercera forma normal).
    - No se modifican programas: cuando ya no son adecuados, se generan otros nuevos, óptimos y no remendados, que los sustituyen.
- Utilización los archivos o bases de datos preexistentes como propios de GeneXus.
- Lenguajes poderosos y de muy alto nivel para la definición de Procesos, Work Panels y Web Objects. En estos lenguajes las descripciones de los procesos se hacen sin referirse a los archivos involucrados, los que son inferidos automáticamente en tiempo de generación. Esta característica permite una total independencia entre los datos y dichas especificaciones. Como consecuencia, las especificaciones de alto nivel de GeneXus no necesitan modificaciones ante modificaciones de la base de datos.
- Mantenimiento 100% automático: El conjunto de estos elementos permite a GeneXus generar y mantener automáticamente el 100% de los programas en aplicaciones de negocios (comerciales, administrativas, financieras, industriales, etc.).
- GeneXus funciona en PCs, dejando al entorno de producción totalmente libre para el procesamiento de las aplicaciones.
- Fácil distribución del conocimiento corporativo para facilitar el desarrollo de nuevas aplicaciones.
- Soluciones de Reportes y Data Warehousing simples y potentes.
- Verificación automática de consistencia, y consolidación, entre aplicaciones desarrolladas separadamente.
- Independencia de plataforma y arquitectura.
- Simplicidad: GeneXus utiliza los recursos más avanzados de la inteligencia artificial para que el analista y los usuarios, puedan usarlo de una forma muy simple.

¿Quiénes son los usuarios de GeneXus?

Más de 5.500 clientes utilizando más de 50.000 licencias en el mundo usan GeneXus para crear e integrar aplicaciones de misión crítica que fácilmente se adaptan a los implacables cambios del negocio. La tecnología GeneXus permite que sus clientes usen el know-how exclusivo de su negocio en las plataformas tecnológicas líderes del mercado en cada momento.

Los clientes corporativos van de empresas medianas a muy grandes en una gran variedad de industrias. Hoy representan el 65% de la facturación Genexus.

Las casas de software comprenden pequeñas, medianas y grandes empresas de software que construyen sus soluciones utilizando la tecnología GeneXus. Este segmento representa actualmente el 35% de la facturación y está creciendo rápidamente.

Algunas empresas que eligieron GeneXus:

Más casos de éxito en: www.genexus.com/exito