UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

ESCUELA DE POSTGRADO

TESIS

LA METODOLOGÍA GENEXUS DE GONDA Y LA TRADICIONAL EN EL APRENDIZAJE DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS EN LOS ESTUDIANTES DEL IV CICLO DE LA ESPECIALIDAD DE INFORMÁTICA DE LA UNIVERSIDAD NACIONAL DE EDUCACIÓN ENRIQUE GUZMÁN Y VALLE

PRESENTADA POR:
FLORENCIO FLORES CCANTO

para optar el Grado Académico de Doctor en Ciencias de la Educación.

ASESORA: Dra. Lidia CRUZ NEYRA

LIMA - PERÚ
2007

TESIS

"LA METODOLOGÍA GENEXUS DE GONDA Y LA TRADICIONAL EN EL APRENDIZAJE DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS EN LOS ESTUDIANTES DEL IV CICLO DE LA ESPECIALIDAD DE INFORMÁTICA DE LA UNIVERSIDAD NACIONAL DE EDUCACIÓN ENRIQUE GUZMÁN Y VALLE"

GRADUANDO : FLORENCIO FLORES CCANTO

SECCIÓN : DOCTORADO

MENCIÓN : CIENCIAS DE LA EDUCACIÓN

MIEMBROS DEL JURADO

DR. RUBÉN FLORES ROSAS
PRESIDENTE DEL JURADO

DR. ROLANDO VIZARRAGA RODRÍGUEZ
MIEMBRO DEL JURADO

DR. ROBERTO CASTRO GÓMEZ
MIEMBRO DEL JURADO

DR. LUIS RODRÍGUEZ DE LOS RÍOS
MIEMBRO DEL JURADO

DRA. LIDIA CRUZ NEYRA
ASESOR

DICIEMBRE – 2007
LIMA - PERÚ

"Saber lo que sabes y saber lo que no sabes
es característico de alguien que sabe."
CONFUCIO (55-479 a. C.)

DEDICATORIA:
A la memoria de mi madre.

AGRADECIMIENTO

A los Docentes de la Sección Doctoral de la Escuela de Postgrado de la Universidad Nacional de Educación Enrique Guzmán y Valle, por su valiosa enseñanza y permanente orientación en mis estudios de doctorado.

A la Srta. Dra. Lidia CRUZ NEYRA, por su asesoramiento en la realización de la presente investigación.

Al Dr. Raúl LOAYZA JAQUI, Mg. José Fernando DAVELOUIS MENDÍA, Mg. Jimmy ROSALES HUAMÁN, Mg. Bertila GARCÍA DÍAZ y Mg. Hugo Froilán VEGA HUERTA, por su invalorable ayuda en la validación de los instrumentos de recolección de datos (pretest y postest para los estudiantes).

A mi colega docente Lic. Renée Maquera Atencio, quien tuvo a cargo el dictado de la asignatura de Base de datos con el grupo control en la Facultad de Ciencias de la Universidad Nacional de Educación Enrique Guzmán y Valle.

A mis compañeros de la sección Doctoral, Promoción 2004, de la Escuela de Postgrado de la Universidad Nacional de Educación Enrique Guzmán y Valle, por su apoyo y aliento permanente.

Finalmente, mi reconocimiento a todas las personas que colaboraron de una u otra manera en la ejecución de la investigación.

El autor.

RESUMEN

La presente investigación se realizó sobre la base de muestra inquietud por hallar°respuesta al siguiente problema: ¿Cuáles son los efectos de la aplicación de la metodología GeneXus de Gonda y la metodología tradicional en el aprendizaje del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle?.

Este estudio es de tipo cuasi-experimental con un diseño de dos grupos apareados: experimental y de control, los cuales fueron medidos a un pretest y postest. El diseño consistió en aplicar la metodología GeneXus de Gonda en el grupo experimental, mediante el uso de clases expositivas en las horas de teoría y del laboratorio de informática para las horas de práctica, guiados con un manual; y la aplicación de la metodología tradicional al grupo control, la cual consiste en en estudiar el lenguaje de programación Visual Basic, el gestor de base de datos Microsoft Access y desarrollar software sobre base de datos. En ambos grupos se controla las variables intervenientes; solo varían en la aplicación de la metodología GeneXus de Gonda y la tradicional en el desarrollo de software sobre base de datos.

Los resultados fueron medidos y sometidos a pruebas estadísticas de U de Mann Whitney para el estudio del comportamiento de los puntajes de los grupos experimental y control, y W de Wilcoxon para el estudio de rangos y pares igualados (magnitud y dirección), tanto para el grupo experimental como para el grupo control.

Se estableció que el grupo experimental tuvo una diferencia de medias 8,842, superior al grupo control que tuvo una diferencia de medias 4,737 en los puntajes de conocimiento conceptual, en los puntajes de conocimiento procedimental, una diferencia de medias 32,105, superior al grupo control con diferencia de medias 25,579 en el conocimiento procedimental y una diferencia de medias 11,737, superior al grupo control con diferencia de medias 3,105 en el conocimiento actitudinal.

Se sabe que la aplicación de la metodología de desarrollo de software ha estado adaptándose a los cambios tecnológicos y al desarrollo de la ingeniería de software en nuestro país y el mundo. Recientemente se viene aplicando la metodología GeneXus de Gonda en el desarrollo de software sobre base de datos en el Perú; sin embargo, la metodología tradicional del desarrollo de software hace que el estudiante primero tenga que estudiar un lenguaje de programación y un gestor de base de datos, y, posteriormente, puede iniciar con el desarrollo de software. Para lo anterior, se debe considerar el ciclo de vida del software, que tiene las siguientes fases: estudio de los objetos de la realidad, estudio de los requerimientos, análisis de datos, diseño de base de datos, análisis funcional, diseño del software, programación o codificación y el programa integrado; mientras que con la aplicación de la metodología GeneXus de Gonda, se incorpora el concepto de base de conocimiento, el que se fundamenta en los datos y no en los procesos. Esta base se construye a partir de las visiones de usuarios y mediante las aproximaciones sucesivas, la que permite crear prototipos automáticamente en un lenguaje de programación y un gestor de base de datos.

En esta tesis se estudia que la metodología GeneXus de Gonda mejora el aprendizaje del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

En la determinación de los logros de aprendizaje del conocimiento conceptual, la media del grupo experimental es de 69,5% siendo el grupo control de 45,5 %. En cambio, en los logros de aprendizaje del conocimiento procedimental, la media del grupo experimental es de 60%, siendo el grupo control de 43,42 %; igualmente los logros de aprendizaje del conocimiento actitudinal, la media del grupo experimental es de 76,84%, siendo el grupo control de 60,89%.

Lo anterior nos lleva a la conclusión de que la aplicación de la metodología GeneXus de Gonda mejora significativamente el logro de los aprendizaje en el desarrollo de software sobre base de datos, en comparación con la aplicación de la metodología tradicional.

SUMMARY

The research was developer in order to get the answer to the problem: which are the effects of applying GeneXus de Gonda methodology and traditional methodology to the learning of the software development on database applied to 4th semester students of computer sciences from the National University of Education Enrique Guzmán y Valle?.

This is a cuasi experimental research with pattern of coupled groups, one experiment aproup and one control group, taking into account the results of a pretest and postest of both groups and the development of a manual which involves the GeneXus de Gonda methodology in learning of the software development on database. This was applied to 4th semester student of computer sciences selected because of the easy access to apply the project and for its relation with the curriculum developer on that speciality.

Statistics shows that the experimental group had much more meaningful learning with a difference of average of 8,842 which is superior to the control group who had a difference of average of 4,737. This means that there is a very meaningful statistical difference among the grades of the opening test and the post test on the learning of conceptual knowledgement of the software development on the database, so that the hypothesis was contrasted GeneXus de Gonda methodology gets better the learning of the software development on data base compared with the traditional methodology, applied to 4th semester student applied to 4th semester student of computer sciences speciality of the National University of Education Enrique Guzmán y Valle.

ÍNDICE

INTRODUCCIÓN

La presente investigación determina la validez de aplicación de la metodología GeneXus de Gonda para mejorar el logro de aprendizaje del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

La investigación parte de un marco teórico sobre conceptos generales del desarrollo de software y antecedentes de estudio; proceso de desarrollo del software sobre base de datos mediante la metodología tradicional y la metodología Genexus de Gonda, revisión de la literatura actualizada sobre el aprendizaje, según diversos autores.

En el capítulo segundo, se presenta la determinación del problema, formulación del problema y se señala la importancia, alcances y limitaciones de la investigación.

En el título segundo, Trabajo de campo, desarrollamos el capítulo cuarto que comprende los instrumentos de investigación (selección y validación), los resultados, la descripción de otras técnicas de recolección de datos, el tratamiento estadístico e interpretación de cuadros, resultados, tablas, gráficos, y concluye con la discusión de resultados.

Como en todo trabajo de investigación, finalizamos con las conclusiones, sugerencias y referencias bibliográficas. Asimismo, consignamos como anexo los cuestionarios aplicados a las unidades muestrales del estudio.

Para tal efecto, las hipótesis relacionan las dos variables en estudio: la metodología Genexus de Gonda y la tradicional en el aprendizaje del desarrollo de software sobre base de datos. La investigación es de tipo explicativo-experimental, y se trabajó con un diseño cuasi-experimental con dos grupos apareados, uno experimental y

otro de control, aplicándose el pretest y postest, como mediciones de entrada y salida, previamente validados a través del juicio de expertos y prueba piloto.

El tratamiento estadístico permite realizar la Prueba de Hipótesis de las variables del estudio, para determinar el nivel de influencia de la primera variable sobre la segunda.

La aplicación de la metodología GeneXus de Gonda en la enseñanza del desarrollo de software sobre base de datos es importante y necesaria, toda vez que apoya en las aproximaciones sucesivas del desarrollo, permite formar en los estudiantes un grado de seguridad en la resolución de problemas y los libera del temor a equivocarse, porque pueden corregir los errores rápidamente mediante el uso de prototipos.

PRIMERA PARTE: ASPECTOS TEÓRICOS
CAPÍTULO I: MARCO TEÓRICO

1.1 ANTECEDENTES Y MARCO TEÓRICO

1.1.1 ANTECEDENTES DE ESTUDIO

Durante la elaboración del proyecto y ejecución de este trabajo, se han revisado diversas investigaciones sobre estudios similares o afines realizados en las diferentes universidades del país y del extranjero y no se ha encontrado ninguna bajo la denominación de la *metodología Genexus de Gonda y la tradicional en el aprendizaje del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.* Se han encontrado diversos estudios sobre metodologías del desarrollo de software sobre base de datos, y no existe investigación alguna sobre experiencias en la enseñanza y aprendizaje de las metodologías de desarrollo de software en el nivel universitario mediante la aplicación de la metodología GeneXus de Gonda. Sin embargo, existen trabajos de investigación referidos a metodologías de enseñanza o, mediante el uso de diversos software educativos en las diferentes áreas del conocimiento.

A continuación se presentan algunas investigaciones referidas a metodologías de desarrollo de software, las que serán de motivo de análisis y comentario.

1.1.1.1 Investigaciones en el ámbito nacional

En las universidades del país existe poca literatura sobre trabajos relacionados a experiencias de uso de metodologías de desarrollo de software; no obstante, se han encontrado los siguientes trabajos de aproximación temática con el tema de investigación:

BUSTOS (2002)[1] informa, en la Universidad Nacional de Educación Enrique Guzmán y Valle, sobre la investigación *"Elaboración de software educativo sobre modificación de conducta"* y presenta entre sus conclusiones:

[1] BUSTOS, M. (2002). Informe de Investigación: *Elaboración de software educativo sobre modificación de conducta*. Lima, Instituto de Investigaciones de la UNE.

"1. La reproducción del software es una ventaja por la rapidez en relación en el texto impreso.

2. El software puede ser presentado mediante dispositivos como el data show para conferencias colectivas, resultando atractivo y motivante.

3. Cuando se emplea individualmente implica un aprendizaje personalizado, donde cada quien desarrolla de acuerdo con su propio ritmo."

LI (2002)[2] realizó un estudio con total de 48 docentes de centros educativos secundarios no estatales: Nuestra Señora de la Salud y Liceo Naval del distrito de Punchada, durante el año 2002. Entre las características más relevantes de los docentes se obtuvo la información que el 37,4% fueron docentes de 36 a más años de edad, 58,3% fueron del sexo femenino, 60,4% tuvieron de 11 a más años de tiempo de servicio, 35,4% tuvieron de 11 a más años de tiempo de permanencia en su centro educativo y el 72,9% tenían conocimientos de computación. La que presenta entre otras, la conclusión siguiente:

"3. Al realizar el análisis posterior a la aplicación del software se obtuvo los siguientes resultados: 79,2% de docentes manifestaron que no podía cometerse el error de desaprobar a un alumno estando aprobado, 81,0% manifestaron la facilidad para dejar el modo tradicional de evaluación por el automatizado, 69,0% manifestaron la seguridad para utilizar el software en las evaluaciones de sus alumnos, 92,0% manifestaron la existencia de muchas ventajas que representa el software para el desempeño del docente, 50,0% manifestaron que el tiempo que toma el software para corregir los exámenes es óptimo y el 96,0% manifestaron la posibilidad de implementar el software en sus centros educativos."

ALARCÓN (2003)[3], en su tesis: "*Los medios educativos computarizados y sus implicancias en el proceso de enseñanza y aprendizaje, en la Facultad de Ciencias de la*

---

[2] LI, C. A. (2002). Tesis: *"Propuesta de un sistema de registro de evaluación de los aprendizajes utilizando un software para centros educativos secundarios, Iquitos, 2002".* Iquitos: Universidad Nacional de la Amazonía Peruana.

[3] ALARCÓN, J. (2003). Tesis de Maestría: "*Los medios educativos computarizados y sus implicancias en el proceso de enseñanza y aprendizaje, en la Facultad de Ciencias de la Educación y Humanidades de la Universidad Nacional San Luis Gonzaga de Ica.*" ICA: UN "SLG", pág. 103.

Educación y Humanidades de la Universidad Nacional San Luís Gonzaga de Ica", reportó los resultados de los estudios realizados con los estudiantes evaluados en los cursos obligatorios entre los años 1999 y 2001. De 80 estudiantes entrevistados de la Facultad de Ciencias de la Educación de la Universidad Nacional San Luís Gonzaga, 61 tenían conocimientos en Tecnologías de Información y Comunicación (76,25%) y sólo 19 manifestaron que no (23,75%); y presenta, entre otras, las siguientes conclusiones:

> *"1. La principal dificultad del uso de medios educativos computarizados, es su alto costo económico y falta de apoyo de la UNICA."*
>
> *"2. Las nuevas tecnologías de información y comunicación favorecen el proceso de enseñanza-aprendizaje, alterando el esquema tradicional de enseñanza, en donde la relación alumno docente se ve mejorada y es de mayor beneficio para el estudiante porque permite un mejor aprendizaje."*

GARCÍA (2003)[4], en la investigación titulada: "*Estimulación de la creatividad en la Facultad de Ingeniería Industrial para el desarrollo y producción de software*", en la Universidad Nacional Mayor de San Marcos, referido al uso de las metodologías de desarrollo de software y aprendizaje de destrezas, presenta las siguientes conclusiones:

> *"1. Los docentes y estudiantes de la Facultad de Ingeniería Industrial requieren de un Programa de estímulos a través de diferentes talleres de aprendizaje de destrezas (habilidades y actitudes) orientados al desarrollo de la creatividad.*
>
> *2. Promover la contratación de especialistas del mas alto nivel en el desarrollo de talleres orientados a la producción de software.*
>
> *3. Los docentes y estudiantes requieren de un programa de estímulos a través de talleres de aprendizaje de nuevas metodologías de desarrollo de software y promover el aprendizaje de destrezas (habilidades y actitudes) orientados al desarrollo de la creatividad de software."*

En la tesis de García se puede encontrar los enfoques teóricos sobre la creatividad, la producción y el desarrollo de software, enmarcado dentro del constante y

[4] GARCÍA, Teonila (2003). *Estimulación de la creatividad en la Facultad de Ingeniería Industrial para el desarrollo y producción de software*. Universidad Nacional Mayor de San Marcos. CIDESOFT, pág.16-22.

acelerado proceso de intelectualización de la humanidad, que conlleva a formar personas con conductas éticas, conocimientos y mucha creatividad.

TRISTÁN (2003)[5], en la tesis "*Una metodología de análisis de factores que intervienen en el desarrollo de sistemas de información de apoyo para la toma de decisiones*", dentro de sus conclusiones, presenta las ventajas y desventajas del uso de las metodologías de desarrollo de sistemas, en la que recomienda realizar las etapas de análisis y diseño mediante el uso de las herramientas de cuarta generación, mediante el uso de la metodología incremental dentro de las metodologías ágiles.

TAPIA (2004)[6], en su tesis: "*Metodología de sistemas blandos y cuadro de mando integral para la gestión de la Universidad Tecnológica de los Andes de Apurímac*", afirma que "es sorprendente la falta de herramientas metodológicas que ayuden a las estrategias de la Universidad"; además, propone la "Integración de metodologías de sistemas blandos y el cuadro de mando integral para la gestión de la Universidad", y presenta las siguientes conclusiones:

> *"1. La metodología de sistemas blandos y cuadro de mando integral es una herramienta que le permitirá desarrollar el planeamiento estratégico, monitorear y llevar a cabo las acciones de control de la gestión de la Universidad Tecnológica de los Andes de Apurímac.*
>
> *2. Los aportes que brinda el presente trabajo demuestran claramente cómo la metodología de sistemas blandos puede ser un esquema adaptable al estudio de diversos temas organizacionales, el cual aumenta los conceptos de cuadro de mando integral, han permitido obtener un mando sistémico que permite el estudio de integral y sostenido de una situación problemática concreta.*
>
> *3. La metodología de sistemas blandos es una metodología, basada en el pensamiento sistémico, que sirve perfectamente para, en*

[5] TRISTÁN, L. A. (2003). Tesis: "*Una metodología de análisis de factores que intervienen en el desarrollo de sistemas de información de apoyo para la toma de decisiones*". Universidad Nacional Mayor de San Marcos. Lima, Perú, pág. 142.

[6] TAPIA, Toribio (2004). Tesis de Doctorado: "Metodología de sistemas blandos y cuadro de mando integral para la gestión de la Universidad Tecnológica de los Andes de Apurímac". Universidad Nacional Federico Villarreal. Lima, pág. 126.

forma simultánea, generar nuevas metodologías y solucionar situaciones problemáticas en las organizaciones."

TRINIDAD (2006)[7], en su tesis de Doctorado: *"Modelo de sistema de información gerencial en la Facultad de una universidad pública para ventajas competitivas en el tema decisiones"*, en la Universidad Nacional Federico Villarreal, realiza un estudio de los modelos clásicos de administración y sus funciones administrativas que son la planificación, organización, coordinación, decisión y control. Propone y aplica una nueva metodología de desarrollo de sistemas de soporte a la toma de decisiones, la cual debe responder a la dinámica mundial actual a los requerimientos de la nación. Presenta las siguientes conclusiones:

"1. La metodología propuesta debe ser adoptada por las facultades de la educación pública.

2. Las ITSA'S permiten estrategias que deberán ser evaluadas por la administración y dirección de la FIIS para su posterior puesta en marcha y modificaciones del modelo presentado.

2. El organigrama propuesto permitirá agilizar diversas operaciones que se realizan actualmente en la FIIS."

VICTORIO (2007)[8], en su tesis: "*Los módulos didácticos de ortografía a través de la multimedia y su eficacia en el aprendizaje significativo*", desarrollada en la Universidad Nacional de Educación Enrique Guzmán y Valle, concluye que:

"Los módulos didácticos de ortografía a través de multimedia son innovaciones y un aporte importante en la producción de software educativo. Además constituye un inicio significativo para la Universidad Nacional de Educación ya que los componentes y elementos en esta producción son preferentemente de nuestra institución y de nuestra nacionalidad peruana."

En el ámbito nacional se ha podido encontrar pocas investigaciones referidas al tema aprendizaje del desarrollo de software, y ninguna sobre la aplicación de la

---

[7] TRINIDAD, L. H.. (2006). Tesis de Doctorado: *Modelo de sistema de información gerencial en la facultad de una universidad pública para ventajas competitivas en el tema decisiones*. Universidad Nacional Federico Villarreal. Lima. Pág. 310.

[8] VICTORIO, J. (2007). Tesis Doctoral: *Los módulos didácticos de ortografía a través de la multimedia y su eficacia en el aprendizaje significativo*. Universidad Nacional de Educación Enrique Guzmán y Valle. Perú, pág.132.

metodología GeneXus de Gonda en el desarrollo de software sobre base de datos en la formación de futuros profesionales del área de informática.

1.1.1.2 Investigaciones en el ámbito internacional

En las instituciones académicas extranjeras se han encontrado trabajos relacionados al tema de investigación:

GONDA (1995)[9], en su trabajo de investigación titulado: "*Proyecto GeneXus*", plantea el siguiente problema: ¿se puede diseñar automáticamente el modelo de datos, y generar el esquema de base de datos correspondiente, para un SGBD? Como producto de su investigación, concluye que:

> *"Las soluciones incrementales son las más naturales para los humanos: estudiamos un problema pequeño y desarrollamos un sistema para soportarlo. Luego, el problema se modifica, describimos los cambios y, a esta altura, enfrentamos un gran obstáculo: la propagación de esos cambios, con recursos tradicionales, es extremadamente costosa. ... además...*
> *El sistema infiere, en tiempo de generación de cada programa, sobre qué archivo o archivos debe efectuarse cada operación. Como consecuencia, las especificaciones en este lenguaje no pierden validez por codificaciones en la base de datos, lo que también representa una característica exclusiva. Con esto se llegaba en 1992 a generar y mantener automáticamente el 100% de los programas."*

GONZALES (2000)[10], en el trabajo de investigación titulado: "*Modelo pedagógico para un ambiente de aprendizaje de NTIC*", afirma que estamos lejos de contar con un modelo pedagógico -por lo tanto teórico- que oriente con claridad las formas de enseñar y llevar a la práctica un proceso de enseñanza y de aprendizaje, caracterizado por el uso de medios informáticos y telemáticos, y hacerlas funcional para los profesores y estudiantes. El autor agrega que el uso

---

[9] GONDA, B. (1995). *Proyecto GeneXus*. Academia Nacional de Ingeniería. Premio Nacional de Ingeniería 1995. Uruguay. pág. 5.

[10] GONZALES, M.A. (2000). Tesis: "*Modelo pedagógicos para un ambiente de aprendizaje de NTIC*". En Conexiones, informática y escuela. Un enfoque global. Medellín Colombia. Ed. Universidad Pontificia Bolivariana. 1ra. ed. págs 45-62.

de las NTIC en el proceso enseñanza–aprendizaje presenta ventajas y desventajas:

> *"Ventajas:*
> - *Variedad de métodos.*
> - *Facilitan el tratamiento, presentación y comprensión de cierto tipo de información.*
> - *Facilitan que el alumno se vuelva protagonista de su propio aprendizaje.*
> - *Motivan y facilitan el trabajo colaborativo.*
> - *Abren la clase a mundos y situaciones fuera del alcance del alumno.*
>
> *Desventajas:*
> - *Pasividad, pues se percibe como un medio "fácil".*
> - *Abuso, uso inadecuado.*
> - *Inexistencia de estructura pedagógica en la información y multimedia.*
> - *Tecnófobos y tecnófilos.*
> - *Dificultades organizativas y problemas técnicos."*

ZANONI (2002)[11], en su tesis: "*Modelo de gerencia de proyecto de software: propuesta de extensión de los procesos de gestión DO PMI*", desarrollada en la Facultad de Ciencia de la Computación de la Pontificia Universidad Católica de Río Grande del Sur, Porto Alegre, Brasil, referido al aprendizaje de desarrollo de los procesos de software, concluye:

> *"Se identifica gran potencial de crecimiento en esta línea de investigación, donde los puntos fuertes de formación de futuros profesionales de informática involucran fuertes compromisos entre la academia y la industria, creando condiciones de experimentación y aprendizaje únicas. Esta línea de estudios demuestra que el mundo empresarial y las prácticas de negocio están andando muy frente de las teorías y modelos conceptuales de desarrollo de software."*

[11] ZANONI, A. (2002). *Las metodologías ágiles en la enseñanza de la Ingeniería de Software.* Facultad de Ciencia de la Computación de la Pontificia Universidad Católica de Río Grande del Sur, Porto Alegre, Brasil. pág. 9.

AGUILAR (2003)[12], en su tesis: *"Las metodologías ágiles en la enseñanza de la ingeniería de software"*, desarrollada en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, teniendo como objetivo el estudio de las metodologías de desarrollo de software, concluyó:

> *"1. Diversos estudios han demostrado los beneficios de introducir las metodologías ágiles en las aulas, no sólo por la conveniencia de enseñar métodos que han sido exitosos en la industria, sino porque aportan valores didácticos que incrementan las habilidades del estudiante en su desempeño académico y profesional".*
>
> *"2. Las metodologías ágiles en la enseñanza de la ingeniería de software no implica desechar las teorías convencionales, por el contrario, se complementan y es preciso saber en qué casos usar una u otra."*

VICHILO (2003)[13], en su tesis de Maestría: *"Bases para la instrumentación computacional del constructivismo, aplicado a las ciencias exactas en la enseñanza primaria"*, desarrollada en la Facultad de Ingeniería de la Universidad de las Américas Puebla, en la que concluye que:

> *"1). Es importante utilizar las metodologías de desarrollo de software para poder realizar la instrumentación computacional adecuada.*
>
> *2). Es necesario aplicar un examen antes de desarrollar cualquier juego para saber cuáles son los temas con mayor dificultad de aprendizaje y cuáles son los más conocidos.*
>
> *3). Se debe prestar especial atención en la realización de los exámenes a presentar, evitar ambigüedades ya que el constructivismo no permite aprender de manera formal, sino experimental."*

---

[12] AGUILAR, A. (2003). Tesis de Maestría: *Las metodologías ágiles en la enseñanza de la Ingeniería de Software*. Facultad de Ingeniería de Sistemas. Universidad Nacional Autónoma de México. pág. 89.

[13] VICHILO, César (2003). Tesis de Maestría: *Bases para la instrumentación computacional del constructivismo, aplicado a las ciencias exactas en la enseñanza primaria*. Universidad de las Américas Puebla. Escuela de Ingeniería, Departamento de Ingeniería en Sistemas Computacionales. Chulula, Puebla, México. pág. 85.

HOSSIAN (2003)[14], en su tesis: "*Sistema de asistencia para la selección de estrategias y actividades instruccionales*", para optar el Grado de Magíster en Ingeniería de Software en la Universidad Privada Instituto Tecnológico de Buenos Aires, referida al uso de las metodologías de desarrollo de software, recomienda que:

> *"Es fundamental que se construya un modelo de desarrollo de estructuración del conocimiento y se haga los experimentos en aula de uso de nuevos métodos y uso de nuevas herramientas de desarrollo de software que asista a los docentes de cátedras universitarias en la explicación y práctica del tema."*

MARÍ (2003)[15], en su tesis: "*Explotación de datos aplicada al ámbito universitario*", para optar el Grado de Magíster en Ingeniería de Software en la Universidad Privada Instituto Tecnológico de Buenos Aires, referida al caso de estudio de creación e implementación de un sistema académico, mediante el uso de metodologías de desarrollo de software de cuarta generación, concluye:

> *"1. Para el desarrollo de un software académico, es una buena combinación realizar con Métrica Versión 3 y el Proceso Unificado.*
>
> *2. Es un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software dentro del marco que permite alcanzar los siguientes objetivos:*
>
> - *Proporcionar o definir sistemas de información que ayuden a conseguir los fines de la organización mediante la definición de un marco estratégico para el desarrollo de los mismos.*
> - *Dotar a la organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos.*
> - *Mejorar la productividad de los departamentos de Sistemas y Tecnologías de la información y las comunicaciones, permitiendo una mayor capacidad de adaptación a los*

[14] MARÍ, Carlos Alberto (2005). Tesis de Magíster: "*Sistema de asistencia para la selección de estrategias y actividades instruccionales*". Universidad Privada Instituto Tecnológico de Buenos Aires. Buenos Aires. Argentina. pág 142-143.

[15] HOSSIAN, A. A. (2003). Tesis de Magíster: "*Sistema de asistencia para la selección de estrategias y actividades instruccionales*". Universidad Privada Instituto Tecnológico de Buenos Aires. Buenos Aires. Argentina. pág 339.

cambios y teniendo en cuenta la reutilización en la medida de lo posible.

- *Facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, teniendo en cuenta su papel y responsabilidad, así como las necesidades de todos y cada uno de ellos.*
- *Facilitar la operación, mantenimiento y uso de los productos software obtenidos."*

BARBERÁ (2004)[16], en su investigación titulada: "*La enseñanza a distancia y los procesos de autonomía en el aprendizaje*", desarrollada en la Universitat Oberta de Catalunya, Barcelona, propone los siguientes criterios para la evaluación de metodologías y software de educación a distancia:

"1. Referidas a la interacción profesor-estudiante: en términos generales, dicha interacción supondrá un ajuste pedagógico mutuo.

2. *Referidas a la interacción materiales- profesor: no contemplada, normalmente, pero importante en los formatos educativos virtuales en los que ha de darse una coherencia entre el diseñador-autor de los materiales y el profesor puesto que habitualmente no son los mismos. Hablaremos en este sentido sobre la coincidencia de objetivos y la complementariedad de visiones.*
3. *Referidas a la interacción materiales- estudiante: resaltaremos el proceso de toma de decisiones que potencia el material y el diálogo que con él se establece.*
4. *Referidas a la interacción entre estudiantes: temática más estudiada en contextos virtuales aunque no por ello resuelta de manera satisfactoria desde la perspectiva psicopedagógica en relación con la cooperación entre estudiantes."*

[16] BARBERÁ, Elena (2004). *"La enseñanza a distancia y los procesos de autonomía en el aprendizaje".* Uiversitat Oberta de Catalunya, Barcelona. España. pág. 121.

RIZZI (2005)[17], en su tesis: "*Sistema experto asistente de requerimiento*", presentada para optar el Grado de Magíster en Ingeniería de Software en la Universidad Tecnológica de Buenos Aires, referida al uso de la metodología IDEAL de desarrollo de software, concluye:

> *"1. La utilización de una metodología que promueve el desarrollo incremental de prototipos, como es la metodología IDEAL, procura que la culminación del proyecto se dé en tiempos y costos acordes a los estimados.*
> 2. *La herramienta Kappa-PC resultó adecuada para el tipo de sistema a construir, sobre todo en la modelización de las interfaces de usuario prototipadas en forma evolutiva con los expertos. Además la herramienta se adecuó perfectamente a la información recibida de la etapa de formalización."*

COLL (2006)[18], en el trabajo titulado: "*Análisis y resolución de casos-problema mediante el aprendizaje colaborativo*", presenta y discute una experiencia de innovación de la docencia universitaria en el ámbito disciplinar de la Psicología de la Educación, basada en una metodología de análisis y resolución de casos-problema en pequeños grupos colaborativos, y en el uso de Tecnologías de la Información y de la Comunicación (TIC). La experiencia, que se ha desarrollado a lo largo de dos cursos académicos, se fundamenta en una visión constructivista y sociocultural de los procesos de enseñanza y aprendizaje. Se describe en detalle el diseño instruccional desarrollado, que da prioridad a tres formas de uso de las TIC: 1) como apoyo al trabajo colaborativo en pequeño grupo de los estudiantes; 2) como soporte al seguimiento, el apoyo y la tutorización por parte del profesor, y 3) como apoyo a la reflexión y regulación de los estudiantes sobre su propio proceso de trabajo y aprendizaje.

Estas formas de uso extienden y amplifican la actividad presencial de profesor y estudiantes, y dan lugar a un contexto híbrido (presencial y virtual) de enseñanza y aprendizaje. La valoración global de la experiencia es muy positiva, tanto desde el punto de vista del rendimiento académico de los estudiantes como desde el de la satisfacción de éstos y de profesores. Con todo, se identifican también algunos aspectos susceptibles

---

[17] RIZZI, M. F. (2005). Tesis de Magíster: "*Sistema experto asistente de requerimiento*". Universidad Privada Instituto Tecnológico de Buenos Aires. Buenos Aires. Argentina. pág. 256.

[18] COLL, C., MAURI, T., ONRUBIA, J. (2006). "*Análisis y resolución de casos-problema mediante el aprendizaje colaborativo*". Revista de Universidad y Sociedad del Conocimiento (*RUSC*). Vol. 3, n.° 2. España. págs. 7-11.

de revisión y mejora; en particular, se señala la dificultad que supone integrar herramientas y espacios virtuales de enseñanza y aprendizaje en una "cultura institucional" y de los estudiantes centrada en la presencialidad, y se destaca la necesidad de ayudar y enseñar explícitamente a los alumnos habilidades específicas para el trabajo y el aprendizaje en entornos virtuales.

Las investigaciones relacionadas a las metodologías de construcción de software y el aprendizaje de los mismos están enmarcados dentro del paradigma de la aplicación de nuevas metodologías de cuarta generación, la que permitirá cubrir expectativas de las empresas educativas u otros con los profesionales formados por nuestra universidad, dotados de conocimientos innovadores. En las experiencias y en los antecedentes estudiados, no hemos podido encontrar trabajos en los que se experimenten el uso de la metodología GeneXus de Gonda en la enseñanza y aprendizaje del desarrollo de software sobre base de datos. Sin embargo, en la actualidad, los estudiantes de esta generación tienen nuevas exigencias acordes con los avances de la ciencia y tecnología. Por esta razón, en esta investigación se ha planteado abordar el aprendizaje del desarrollo de software sobre base de datos, mediante la aplicación de la Metodología GeneXus de Gonda.

1.1.2 EVOLUCIÓN DEL DISEÑO DE BASE DE DATOS

En el modelo lógico de datos de BACHMAN(1964)[19] (5), apareció el modelo diseño de base de datos, que consta de una serie de entidades y restricciones de integridad jerárquicas entre ellas. Los diagramas presentados por el autor son, aún, muy utilizados para visualizar partes de la estructura de la base de datos.

LUQUE (2002)[20], plantea la apareció la teoría de Edgar Frank CODD que puso énfasis en la "normalización" o "formas normales", concepto esencialmente matemático que, en definitiva, nos lleva a almacenar en la base de datos, únicamente, aquellos elementos que no podrían inferirse de otros allí existentes. En 1976, Peter CHEN introdujo el llamado modelo de Entidades y Relaciones (E-R).

---

[19] BACHMAN, C. N. (1964). *The Integrated Data Store, a General Purpose Programming System for Random Access Memories*. 1ra ed. Arizona: ACM Press. pág. 3.

[20] LUQUE, Irene y otros (2002). *Bases de datos desde Chen hasta Codd con Oracle*. México. 1ra ed. Alfaomega Ra-Ma. pág. 72.

SILBERSCHATZ y otros (2002)[21], plantea que, en el modelo de Entidades y Relaciones (E-R), existen entidades sustancialmente similares a las de Charles BACHMAN, en el esquema lógico de una base de datos relacional, se introdujeron las relaciones entre las tablas, tales como: relaciones *uno a uno*: 1-1 , relaciones *uno a muchos*: 1-n , y relaciones *muchos a muchos*: n-m y a dichas relaciones se asocian el nombre y el significado.

LUQUE (2002)[22], plantea que el modelo de Peter CHEN introdujo elementos semánticos que ayudan al usuario no técnico a acceder sin ayuda a la base de datos, y fue adoptado por la mayoría de la comunidad informática. Aún hoy, es el modelo tradicional metodológico que más se enseña en las universidades del país, y es el más utilizado en un conjunto de herramientas de ayuda al desarrollo de software sobre base de datos. Sin embargo, existen alrededor de este modelo algunas limitaciones. No se ha encontrado la manera de representar las relaciones n-m directamente, por lo que se termina implementándolos como una entidad ficticia, que soporta los datos de la relación, y las relaciones jerárquicas que subordinan esa entidad ficticia a las involucradas en la relación primitiva.

CASTAÑO (2000)[23], define que, el "modelo semántico se encarga de interpretar el significado de una oración. Se entiende que el significado de una expresión está dado en su forma primitiva y que las transformaciones no aportan cambio de significados"; y TÉLLEZ (2005)[24] afirma que "el análisis sintáctico e interpretación semántica tratan en primer término de identificar la forma en que las palabras se combinan para formar constituyentes a nivel sintáctico superior (los sintagmas); y posteriormente, generan bien una forma lógica o una plantilla parcial desde las sentencias analizadas de forma sintáctica"; es decir, un modelo, para que sea útil, y, podamos operar con él, sin intervención humana, debe ser un modelo sintáctico. O sea que existe también una necesidad previa a la implementación de representar elementos semánticos mediante elementos sintácticos.

---

[21] SILBERSCHATZ, A y otros (2002). *Fundamento de base de datos*. Madrid. MacGraw-Hill. pág. 39.
[22] LUQUE, Irene y otros (2002). Opus. cit. pág. 72.
[23] CASTAÑO, A. y otros (2000). *Fundamentos y modelos de Bases de Datos*. 2da ed. Madrid: Ra-Ma. pág. 38.
[24] TÉLLEZ, A. (2005). En la Tesis: *Extracción de información con algoritmos de clasificación,* para optar la Maestría en Ciencias Computacionales en el Instituto Nacional de Astrofísica, Óptica y Electrónica. México. pág. 32.

En términos prácticos, el modelo Entidad-Relación (E-R) de Peter CHEN se transforma en un modelo de Charles Bachman, porque éste sí se implementa directamente. Las diversas metodologías de análisis de datos *a priori* permiten analizar la organización e identificar en ella los objetos relevantes, sus relaciones y sus representaciones en términos de entidades y relaciones.

1.1.3 EL SOFTWARE Y SU INGENIERÍA

PRESSMAN (2006)[25] dice que "el software se forma con 1) las instrucciones (programas de computación) que al ejecutarse proporcionan las características, funciones y el grado de desempeño deseado; 2) las estructuras de datos que permiten que los programas manipulen información de manera adecuada; 3) los documentos que describen la operación y el uso de los programas".

PRESSMAN (2006)[26] define el software de sistemas como "una colección de programas escritos" y algunos programas de sistemas (como los compiladores, editores, utilerías para la administración de archivos) procesan estructuras de información complejas pero "determinadas" y el software es "determinado" si el orden y el ritmo de las entradas, el procesamiento y las salidas son predecibles; y el software es "indeterminado" si el orden y el ritmo de las entradas, el procesamiento y las salidas no se pueden predecir.

Asimismo, un sistema informático está compuesto por un conjunto de instrucciones que, cuando se ejecutan en un dispositivo físico (el hardware), produce resultados de acuerdo con los objetivos y función principal predeterminado. Dicho conjunto de instrucciones está organizado en estructuras de datos[27], las que permiten la manipulación de la información.

La estructura de datos asociada al tipo de hardware usado influye decisivamente en el desempeño del software. Así, pues, durante el diseño y programación de un software es importante seleccionar el tipo de estructura de datos más adecuado respecto al tipo de procesamiento previsto.

---

[25] PRESSMAN, R. (2006). *Ingeniería del Software: Un enfoque práctico*. 6ta ed. México: McGraw-Hill, pág. 5.
[26] PRESSMAN, R. (2006). Opus cit., pág. 8.
[27] Debe entenderse que la "estructura de datos es una representación de la relación lógica entre elementos de datos individuales y determina la organización, los métodos de acceso, el grado de asociatividad y las alternativas de procesamiento de información".

Para PRESSMAN (2006)[28], el concepto abstracto de software (parte lógica del sistema "computacional"[29]) como punto de partida, presenta algunas características, según se diferencia el software del hardware y ofrecen una buena comprensión de su significado:

1. *"El software se desarrollo o construye; no se manufactura en el sentido clásico.*
2. *El software no se "desgasta".*
3. *A pesar de que la industria tiene una tendencia hacia la construcción de componentes, la mayoría de los sistemas informáticos son desarrollados a medida."*

Para ZAHRAN (1998)[30], el desarrollo de software, más que un intento, es una empresa caracterizada por un proceso continuado de desafío. Además, la influencia del software dentro del mundo de los negocios y de la vida cotidiana de las personas se vuelve crítica debido no sólo a los problemas conceptuales y de diseño, sino también a los problemas técnicos y económicos generados durante el proyecto de software.

Las propuestas académicas y de las industrias para la reducción de estos problemas es el empleo de las herramientas CASE[31] y de las metodologías de desarrollo, que permitan el software evolutivo, a partir de las necesidades de los usuarios, y el empleo de una buena disciplina: la ingeniería de software. Esta disciplina debe incluir a la ingeniería de la usabilidad[32], que define estándares para el desarrollo de software de calidad basados en diversos indicadores como, por ejemplo, la corrección, la eficacia, la eficiencia, la fiabilidad, la facilidad de comprensión, de uso y mantenimiento, la interoperabilidad, la portabilidad, la reusabilidad y la robustez.

De acuerdo con SEFFAH (2003)[33], la ingeniería de software es:

---

[28] PRESSMAN, R. (2006). Opus cit., págs. 8-11.

[29] Se define un sistema "computacional" como un entorno de trabajo, estudio, entrenamiento u otra actividad, el cual está compuesto por tres elementos interrelacionados: el software (la parte lógica), el hardware (la parte física) y el peopleware (la parte humana).

[30] ZAHRAN, S. (1998). *Software Process Improvement.* Practical Guidelines for Business Success. SEI Series in Software Engineering. Massachussets. 1ra ed. Addison-Wesley Longman, pág.63.

[31] La sigla CASE es acrónimo de Computer-Arded Software Engineering. Esta herramienta consiste en un sistema compuesto por software, hardware y una base de datos que ofrece soporte a la ingeniería de software, permitiendo el análisis estructurado, la implementación y el test de sistemas informáticos.

[32] De acuerdo con a la Norma ISO 9241-11 (1998), el concepto de usabilidad consiste en "hasta qué punto un producto puede usarse por usuarios específicos para lograr los objetivos específicos con eficacia, eficiencia y satisfacción en un contexto específico de uso".

[33] SEFFAH, A. (2003). *Empowering Software Engineers in Human-Centered Design. in Proceedings of the ICSE.* III Conferencia en Portland, Oregon. IEEE Computer Society, págs. 653-658.

"el establecimiento y uso de sólidos principios de ingeniería con el propósito de obtener económicamente un software que sea fiable y que funcione eficientemente en máquinas reales."

Asimismo, los autores afirman que el desarrollo formal de software se caracteriza por la definición de los requerimientos de la aplicación, la especificación de los objetivos, el diseño iterativo, los procesos continuados de test y la implementación del software. Para cada una de las actividades se diseñan procesos (y subprocesos) que serán implementados a través de un lenguaje de programación.

1.1.4 EL PROCESO DEL SOFTWARE

Para lograr el entendimiento de proceso de software, es necesario, en primer lugar, presentar una definición de proceso:

HAMMER (1993)[34], presenta una definición adecuada para el ámbito de la presente investigación debido a su carácter general. Según los autores, un proceso es:

"Una colección de actividades que toman uno o más tipos de entradas y crea una salida que es de valor para el cliente."

En el contexto de la presente investigación, la colección de actividades se caracteriza por todos los procedimiento que componen el ciclo de vida del software (procesos de concepción, diseño, desarrollo, mantenimiento, documentación, control de calidad, test, gestión, entrenamiento, etc.) desde las fases iniciales hasta su uso por el cliente (el usuario final). Cabe señalar que un software se caracteriza por tres grandes actividades: entrada de datos, su procesamiento y salida de las informaciones procesadas.

ZAHRAN (1998)[35] propone que un proceso está compuesto por tres aspectos. El primer aspecto es la definición del proceso que generalmente está representado por el documento que especifica las actividades y procedimientos para el proceso. El segundo aspecto se refiere a la transferencia del conocimiento del proceso para aquellos que lo van a ejecutar. Finalmente, el tercer aspecto son los resultados del proceso después de su ejecución. Según el autor, el enfoque orientado a procesos permite:

---

[34] HAMMER, M. (1993). *Reengineering de corporation: A manifesto for Business Revolutions.* 1ra ed. Harper Collins. New Cork, pág.35.

[35] ZAHRAN, S. (1998) Opus cit., pág. 221.

"1.La sincronización y armonía entre las actividades desempeñadas por cada individuo y los objetivos comunes del equipo.

2.La garantía de la consistencia de las actividades y sus resultados.

3.El uso de técnicas objetivas de medición del desempeño de cada individuo respecto a las actividades asignadas.

4. Debido a la reducción de las dependencias en cada individuo, se consigue que el equipo pueda repetir un determinado proceso, aunque individuos ajenos sean asignados a la actividad."

BARRY (1988)[36] argumenta que "las funciones primarias de un modelo de proceso de software son determinar el orden de las fases involucradas en el desarrollo y evolución de software y establecer los criterios de transición para avanzar de una fase a la próxima".

1.1.5 LA INGENIERÍA DE SOFTWARE

PRESSMAN (2006)[37] define la ingeniería de software como un conjunto estructurado de pasos que representan los "paradigmas de la ingeniería de software". Estos paradigmas se basan en la naturaleza del proyecto y de la aplicación, en los métodos y herramientas que serán usados en el proyecto, los controles y los productos y servicios desarrollados.

1.1.6 CICLO DE VIDA DEL SOFTWARE

En las bibliografías especializadas podemos encontrar varios modelos de ciclo de vida de desarrollo de software. Todos ellos se basan en diversos tipos: la planificación, el análisis, el diseño e implementación del software. Cada uno de estos métodos está asociado a los marcos y paradigmas de ciclo de vida de desarrollo de software, a los paradigmas tecnológicos de su entorno y época.

A continuación, se presenta una revisión de los modelos de ciclos de vida de desarrollo de software:

---

[36] BARRY, B. (1988). *Applying process programming to the spiral Model.* Proc. Fourth software process. Work Shop. IEEE. UCLA. pág. 61.

[37] PRESSMAN, R. (2006). Opus. cit., págs. 23-24.

1.1.6.1 Modelo codificar–y-fijar

BARRY (1988)[38] comenta que en la época inicial del desarrollo de software se ha usado el modelo codificar-y-fijar (code-and-fix). Éste contiene dos pasos:

"- Escribir algún código.

- Fijar los problemas en el código."

De esta manera, inicialmente se implementaba algún código y, a continuación, se pensaba sobre los requerimientos, el diseño, los test y el mantenimiento. Las principales dificultades del modelo consisten en la estructuración insuficiente de código, la carencia de correspondencia con las necesidades del usuario y el coste excesivo debido a la deficiente preparación de los test y modificaciones. Ante esto, se observa la necesidad de reorganización del modelo en etapas, incorporando elementos de planificación, coordinación y control.

1.1.6.2 Modelo incremental

PRESSMAN (2006)[39] fundamenta que el modelo incremental combina elementos del modelo de la cascada en forma iterativa. Asimismo, McDERMID(1993)[40] dice que "por ejemplo el software de procesamiento de tratamiento de textos desarrollado con el paradigma incremental podría extraer funciones de gestión de archivos básicos y de producción de documentos en el primer incremento; funciones de edición más sofisticadas y de producción de documentos en el segundo incremento; corrección ortográfica y gramatical en el tercero; y una función avanzada de esquema de página en el cuarto. Se debería tener en cuenta que el flujo del proceso de cualquier incremento puede incorporar el paradigma de construcción de prototipos".

Cuando se utiliza un modelo incremental, el primer incremento a menudo es un producto esencial. Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer. El cliente utiliza el producto central (o sufre la revisión detallada). Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente y la

[38] BARRY, B. (1988).Opus. cit., pág. 104.
[39] PRESSMAN, R. (2006). Opus. cit., pág.52.
[40] McDERMID, G. J. (1993). *Software Development Process Model.* 1ra ed. En software engineer's Reference Book. CRC Press. págs.15-26.

entrega de funciones, y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

El modelo de proceso incremental (ver Gráfico N° 1), al igual que la construcción de prototipos y otros enfoques evolutivos, es iterativo por naturaleza. Pero a diferencia de la construcción de prototipos, el modelo incremental se centra en la entrega de un producto operacional con cada incremento. Los primeros incrementos son versiones "incompletas" del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

El desarrollo incremental es particularmente útil cuando la dotación de personal no está disponible. Los primeros incrementos se pueden implementar con menos gente. Si el producto esencial está bien definido, se agrega (si se requiere) más personal para implementar el incremento siguiente. Además, los incrementos se pueden planear para manejar los riesgos técnicos.

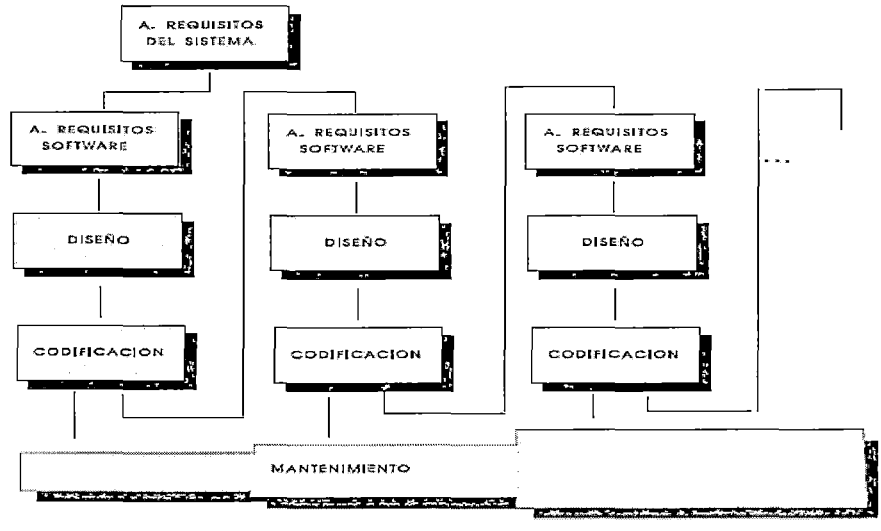


Gráfico N° 1.El modelo incremental.

1.1.6.3 Modelo de cascada

Este modelo, presentado por PRESSMAN (2006)[41], conocido como "ciclo de vida clásico", ha sido inicialmente presentado por ROYCE (1970)[42] y se caracteriza por un refinamiento del modelo de etapas, en el que se realiza los ciclos de retroalimentación entre las etapas con el objetivo de minimizar el coste de retrabado del proyecto. El modelo incorpora el

[41] PRESSMAN, R. (2006). Opus. cit., pág. 50.
[42] ROYCE, W. (1970). *Managing the Development Of Large Software System: Concepts and Techniques.* Proc. of IEEE. WESCON. pág. 60.

"prototipo" al ciclo de vida de desarrollo de software, tal como podemos ver en el Gráfico N° 2:

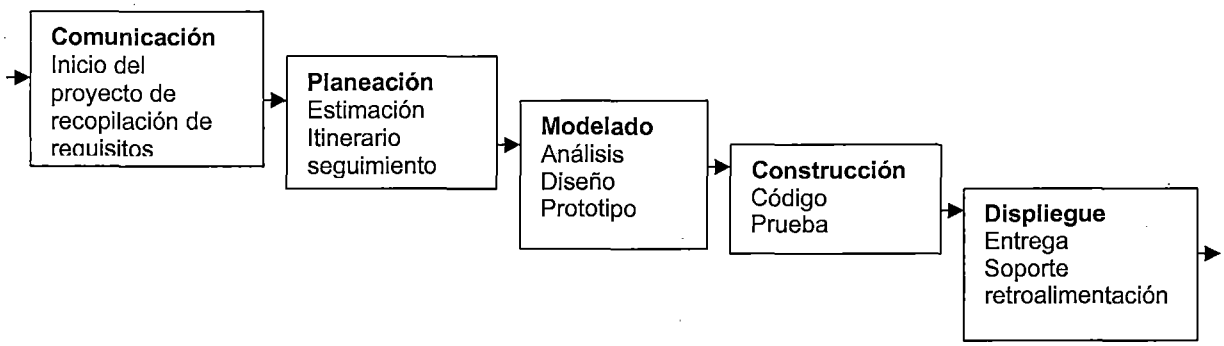


Gráfico N° 2. El modelo en cascada del proceso de software. Fuente: PRESSMAN (2006: 26-30 y 50).

Aunque los ciclos de retroalimentación permitan realizar extensas revisiones y refinamientos, se identifican algunas dificultades causadas por el énfasis en la elaboración de documentos completos como criterios de finalización para las etapas de requerimientos y diseño. Por otra parte, BARRY (1988)[43] dice "se observa la problemática de su aplicación a sistemas interactivos, debido, una vez más, al distanciamiento entre los usuarios y el equipo de desarrollo".

1.1.6.4 Modelo de desarrollo orientado a prototipos

Para PRESSMAN (2006)[44], el modelo consiste en un procedimiento que permite al equipo de desarrollo diseñar y analizar una aplicación que representa el sistema que será implementado. Dicha aplicación, llamada prototipo, está compuesta por los componentes que se desean evaluar (las funciones principales). Según el autor, las etapas del modelo son:

"1. Investigación preliminar.

2. Colecta y refinamiento de los requerimientos y proyecto rápido:

- *Análisis y especificación del prototipo.*
- *Diseño y construcción del prototipo.*
- *Evaluación del prototipo por el cliente.*
- *Refinamiento del prototipo.*

3. Diseño técnico.

4. Programación y test.

5. Operación y mantenimiento."

[43] BARRY, B. (1988). Opus. cit. pág. 245.
[44] PRESSMAN, R. (2006). Opus cit. pág. 55.

PRESSMAN (2006)[45] plantea que los problemas identificados en el modelo de desarrollo orientado a prototipos consisten, por una parte, en que "se ve lo que parece ser" (lo que el usuario examina es una representación del sistema con funcionalidades restringidas), por tanto, no se considera la calidad global del software, ni sus aspectos de mantenimiento. Por otra parte, el equipo de desarrollo hace concesiones de implementación basadas en el uso de sistemas operativos y lenguajes de programación impropios, sin importar la inadecuación posterior del producto (ver el Gráfico Nº 3).

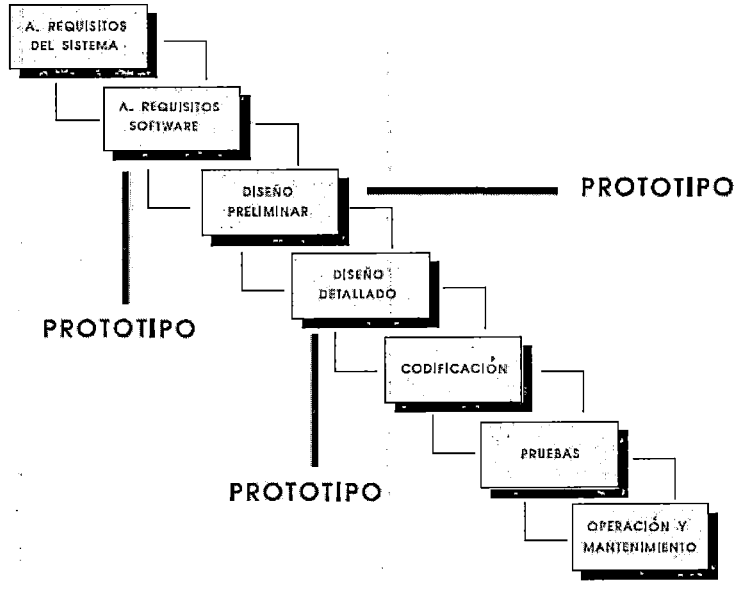


Gráfico Nº 3. El modelo de prototipo. Fuente: PRESSMAN (2006: 55)

1.1.6.5 Modelo de desarrollo evolutivo

De acuerdo con PRESSMAN (2006)[46], las etapas del modelo de desarrollo evolutivo "consisten en expandir los incrementos de un producto de software operativo, siendo las direcciones de evolución determinadas por la experiencia operativa". Se identifica, por tanto, la importancia en la obtención de un sistema de producción flexible y expansible, permitiendo la adaptación de los cambios de requerimientos que no han sido planeados. Se asocia este modelo a un lenguaje de cuarta generación, tales como la herramienta GeneXus y otros. En este sentido, SOMMERVILLE (1996)[47] comenta que este modelo es el más apropiado para el desarrollo de sistemas interactivos y de sistemas de inteligencia artificial.

[45] PRESSMAN, R. (2006). Opus cit., pág. 56.
[46] PRESSMAN, R. (2006). Opus cit., pág. 54.
[47] SOMMERVILE, L. (2001). *Software Engineering*, 6ta ed. New York: Addison Wesley- Reading. pág. 188.

"Las etapas básicas del modelo son:

- *Especificación inicial.*
- *Desarrollo del producto o servicio.*
- *Implementación.*
- *Uso.*
- *Evaluación.*
- *Nuevas versiones.*
- *Re-especificación."*

BARRY (1988)[48] expresa que se observan algunas dificultades técnicas como, por ejemplo, la problemática de la integración de aplicaciones independientes y el reemplazo de un nuevo software en un gran sistema existente.

1.1.6.6 Modelo de transformación

El modelo de transformación ha sido propuesto como una solución a las dificultades de código del modelo de desarrollo evolutivo y del modelo codificar-y-fijar analizado dentro del modelo en cascada. Desde la óptica de BARRY (1988)[49], el modelo:

> *"asume la existencia de una capacidad de convertir automáticamente una especificación formal de un producto de software en un programa que satisfaga la especificación".*

Según el autor, los pasos de este modelo son:

> *"1.Especificación formal del mejor entendimiento inicial del producto deseado.*
> *2.Transformación automática de la especificación en código.*
> *3.Un ciclo iterativo, si es necesario, para mejorar el desempeño del código resultante, lo que ofrece una guía de optimización al sistema de transformación.*
> *4. Ejercicio o evaluación del producto resultante.*
> *5.Un ciclo interactivo externo para ajustar las especificaciones basadas en el resultado de la experiencia operativa, y para rederivar, reoptimizar, y ejercitar o evaluar* el producto de software ajustado."

[48] BARRY, B. (1988). Opus cit., pág. 321.
[49] BARRY, B. (1988). Opus cit., pág. 63.

Las dificultades del modelo se caracterizaban por la carencia generalizada de capacidades de transformación automáticas sólo disponibles para pequeños productos, la posibilidad de evolución no planeada y los problemas de mantenimiento de base de conocimiento.

1.1.6.7 Modelo espiral

BARRY (1988)[50] presenta el modelo espiral del proceso de software (véase el Gráfico N° 4), el cual ha evolucionado basándose en los diversos refinamientos del modelo en cascada y del desarrollo orientado a prototipos. El modelo espiral se caracteriza por un conjunto de ciclos progresivos, en los cuales se identifican los objetivos de cada parte del producto que está siendo desarrollado, las propuestas alternativas de implementación y las restricciones impuestas por dichas alternativas al software.

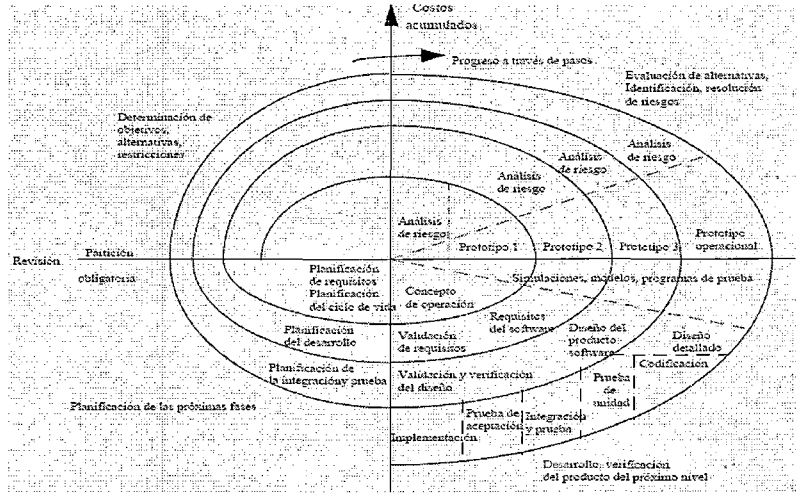


Gráfico N° 4.El modelo espiral del proceso de software. Fuente: Tesis de Bibiana ROSSI D. (2001)[51]

El modelo se divide en cuatro cuadrantes que representan las actividades principales para el ciclo de vida de desarrollo de software, resumidos por PRESSMAN (1995)[52] en:

"- Planeamiento.
- Análisis de riesgos.
- Ingeniería
- Evaluación del cliente."

[50] BARRY, B. (1988). Opus cit., pág. 73.
[51] ROSSI, B. (2001). Tesis de Magíster: *"Sistema experto de ayuda para la selección del modelo de ciclo de vida"*. Argentina. Universidad Privada Instituto Tecnológico de Buenos Aires, pág. 28
[52] PRESSMAN. R. (2006). Opus cit., pág. 58.

BARRY (1988)[53] presenta que el ciclo de vida del modelo espiral se basa en cuatro preguntas fundamentales:

"¿Cómo empieza la espiral?".

"¿Cómo uno rompe los ciclos de la espiral cuando es apropiado terminar un proyecto apenas iniciado?".

"¿Por qué la espiral se acaba bruscamente?".

"¿Qué pasa con la mejora (o el mantenimiento) del software?".

Debido a la reducción de riesgos determinada por el método evolutivo del modelo, éste representa un enfoque más apropiado para el desarrollo de grandes, complejos y ambiciosos sistemas de información. Además, el modelo espiral posee un nivel alto de capacidades de entorno de soporte de software y es flexible, lo que permite acomodar diversas alternativas técnicas y objetivos de usuario.

1.1.6.8 Capability Maturity Model for Software

PAULK (1993)[54] define el modelo de madurez de capacidad (Capability Maturity Model – CMM-) como un modelo que establece los niveles por los cuales las organizaciones de software hacen evolucionar sus definiciones, implementaciones, mediciones, controles y mejorías de sus procesos de software.

Además, el CMM permite la definición del grado de madurez de las prácticas de gestión y de ingeniería de software de dichas organizaciones. De esta manera, se puede determinar cómo madura una determinada organización y cuáles son las acciones de mejora prioritarias para sus procesos de software.

Para ZAHRAN (1998)[55], el enfoque inicial del CMM ha sido el proceso de software. Sin embargo, se puede encontrar aplicaciones del modelo en otros campos como por ejemplo CMM para personas (People CMM), CMM para ingeniería de sistemas (Systems Engineering CMM), CMM para la gestión de productos integrados (Integrated Product Management CMM) y otros.

---

[53] BARRY, B. (1988). Opus. cit., pág. 65.

[54] PAULK, M. y otros (1993). *Capability Maturity Model for Software: Version 1.1*. Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University. Pittsburg. Pennsylvania, pág. 202.

[55] ZAHRAN, S. (1998). Opus cit., pág. 76.

1.1.6.9 Estructura del Capability Maturity Model for Software (CMM)

PAULK y otros (1993)[56] presentan el esquema del modelo CMM, el que está compuesto por cinco niveles de madurez de acuerdo con la capacidad del proceso de software y definidos por los objetivos de los procesos que, cuando satisfechos, permiten evolucionar al próximo nivel ya que uno o más componentes importantes del proceso de software han sido estabilizados.

De esta manera, los niveles de madurez ayudan a las organizaciones a definir prioridades para sus esfuerzos de mejora. En el Gráfico N° 5, el autor presenta los cinco niveles de madurez del proceso de software:

> *"Nivel 1 - Inicial: se caracteriza como ad hoc o caótico. Pocos procesos son definidos.*
> *Nivel 2 - "Repetible" o repetición: se caracteriza como disciplinado. Se establecen procesos básicos de gestión.*
> *Nivel 3 -Definido: se caracteriza como estándar y consistente.*
> *Nivel 4 - Gestionado: se caracteriza como predicable. Hay una preocupación en la medición detallada de la calidad del proceso de software y del producto.*
> *Nivel 5 - Optimización: se caracteriza como mejora continua a partir de la realimentación (feedback) cuantitativa."*

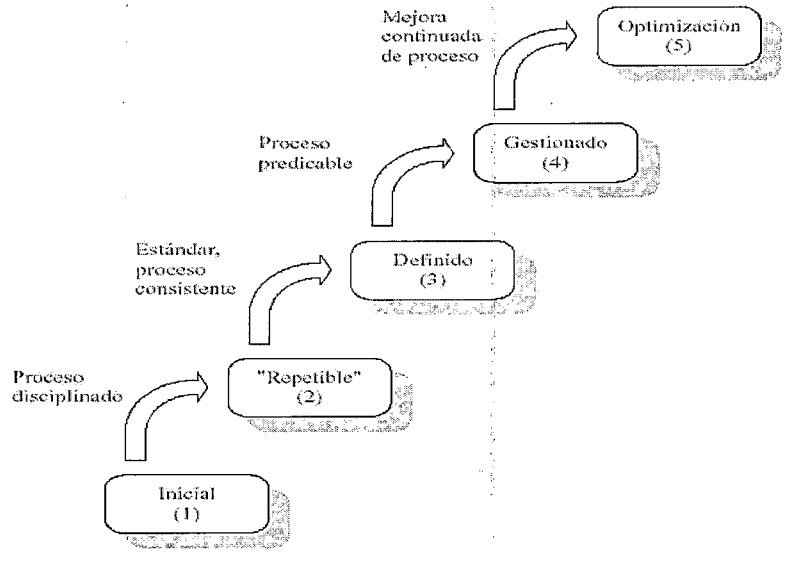


Gráfico N° 5. Los niveles de madurez del proceso de software del CMM. Fuente: Paulk y otros (1993)

[56] PAULK, M y otros (1993). Opus. cit., pág. 271.

Cada nivel de madurez tiene una estructura interna (véase el Gráfico N° 6) compuesta por los siguientes componentes:

- Nivel de madurez: Representa un indicador evolutivo que permite alcanzar la madurez del proceso de software.

- Áreas de proceso clave (Key Process Areas -KPA): Son las subestructuras de cada nivel que indican las áreas a las que una organización debería dirigir su atención con el propósito de mejorar su proceso de software. Se asigna cada conjunto de KPA a un nivel de madurez (excepto al nivel uno), como se muestra en la Figura N° 5. ZAHRAN (1998)[57] comenta que el CMM no detalla todas las áreas de proceso involucradas en el desarrollo y mantenimiento de software, sino que se enfoca en las áreas clave que contribuyen en la mayoría de las capacidades de proceso.

- Objetivos: Este componente delimita las KPA a través de la definición de su alcance, límites e intenciones. Los objetivos, por lo tanto, determinan las restricciones que deben ser superadas por la organización para que ésta pueda alcanzar mejores niveles de madurez.

- Características comunes: Son atributos que indican si la implementación e institucionalización de una KPA son eficaces, repetidas y duraderas.

Además, PAULK y otros (1993)[58] presentan cinco características comunes:

"*1. El compromiso en desempeñar las acciones que garantizan el establecimiento del proceso.*

2. La habilidad en desempeñar dichas acciones.

3. Las actividades desempeñadas para implementar las KPA.

4. La medición y el análisis del estado y eficacia de las actividades desempeñadas.

5. La implementación para la verificación de las actividades desempeñadas respecto a los procesos establecidos."

[57] ZAHRAN, S. (1998). Opus cit., pág. 62.
[58] PAULK , M y otros (1993) Opus cit., pág. 293.

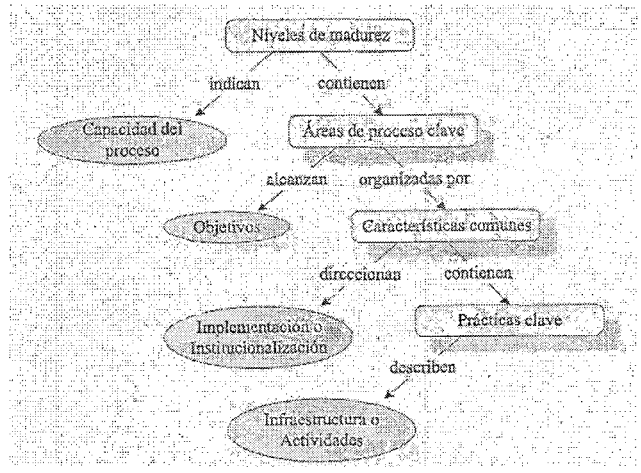


Gráfico N° 6.Estructura interna del CMM. Fuente: Paulk y otros (1993).

1.1.6.10 Bootstrap

El Bootstrap propuesto por KUVAJA (1993)[59] es una metodología de evaluación que mejora la capacidad de los modelos de procesos de desarrollo de software. Además, el modelo Bootstrap describe el proceso de evaluación, determina dónde se encuentra una organización respecto de los niveles de madurez, identifica los puntos fuertes y debilidades de la organización, y ofrece una guía para el proceso de mejora.

Este método es uno de los resultados del proyecto ESPRIT 5441 apoyado por la Comunidad Europea. El Bootstrap ha considerado como punto de partida varios estándares y metodologías como, por ejemplo, CMM, ISO /IEC 15504 y los estándares de ingeniería de software (Software Engineering Standards - SES) de la Agencia Espacial Europea (European Space Agency- ESA).

El modelo Bootstrap se basa en la tríada Organización, Metodología y Tecnología (OMT) presentada en la Gráfico N° 7.

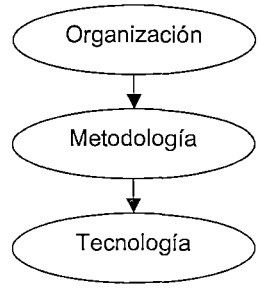


Gráfico N° 7. La triada Bootstrap. Fuente: ZAHRAN (1998)

[59] KUVAJA, P. y otros (1994), *Software Process Assessment and Improvement: The BOOTSTRAP Approach.* Oxford. Blackwell Business Publishers. pág. 419.

1.2 PROCESO DE TEST DE SOFTWARE

Los procesos de test se realizan tanto dentro del ámbito de la ingeniería de software (test de verificación y validación), como dentro del ámbito de la ergonomía (test de usabilidad). En este sentido, es necesario realizar un estudio sobre la formación del proceso de test en la ingeniería de software.

KIT (1995)[60] define el proceso de test de software como "los medios a través de los cuales se integran las personas, los métodos, las mediciones, las herramientas y los equipos con el objetivo de evaluar un producto de software".

A partir de esta definición, el autor presenta seis elementos esenciales para el proceso de test de software:

> *"1. La calidad del proceso de test (y, consecuentemente, la calidad del software producido) determina el éxito del esfuerzo de test.*
> *2. La prevención de la migración de defectos a través de la aplicación de técnicas de test en las fases iniciales del ciclo de vida de desarrollo de software.*
> *3. El uso inmediato de herramientas de test de software.*
> *4. Una persona debe responsabilizarse de la mejora de procesos de test.*
> *5. La realización de test (testing) es una disciplina profesional que requiere personas entrenadas y calificadas.*
> *6. Cultivar una actitud positiva de eliminación de defectos en el equipo de test."*

Según el estudio presentado por el autor, dentro de este contexto se observan avances continuados en la búsqueda de la calidad de software desde las perspectivas de la gestión y sobre la gestión de calidad de software orientada a procesos. Usando estas consideraciones como punto de partida, se presentan, a continuación, las actividades que permiten tanto verificar y validar los software, como evaluar su aceptabilidad y la satisfacción del usuario.

1.2.1 TEST DE VERIFICACIÓN Y VALIDACIÓN

El test de verificación es una de las fases del ciclo de vida de los procedimientos de test. Según KIT (1995)[61], el principal objetivo es detectar la mayoría de errores posibles. En

[60] KIT, E. (1995). *Software testing in the real world.* Boston, MA: Addison-Wesley Publishing Company, pág. 3.

este sentido, SCHULMEYER (2000)[62] comenta que la verificación permite garantizar la consistencia entre los productos desarrollados (los prototipos y la versión final del software) y sus requerimientos predeterminados en la fase de especificación, es decir, se necesita saber si se está desarrollando de manera correcta el producto.

Por otra parte, SCHULMEYER (2000)[63] menciona que el test de validación permite garantizar que el software final satisfaga los requerimientos del sistema, es decir, se necesita saber si se desarrolla el producto correcto.

Para una mejor comprensión de los tipos de test de verificación y validación, se presenta en el Gráfico Nº 8 el modelo "U en punto" (Dotted-U Model) propuesto por Software Development Technologies (SDT). Según KIT (1995)[64], el modelo "U en punto" integra el ciclo de vida de desarrollo y el ciclo de test.

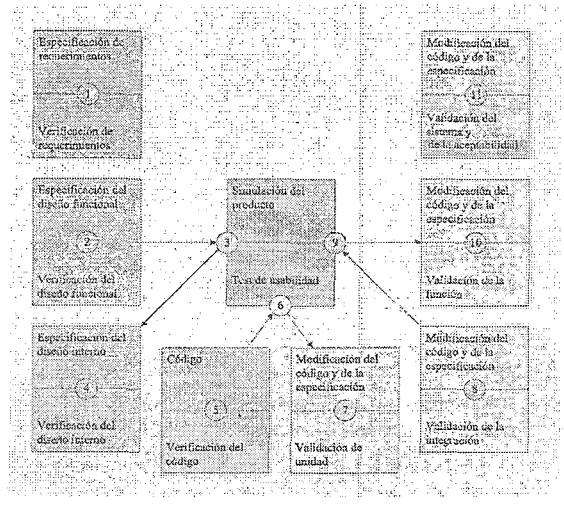


Gráfico Nº 8.Adaptación del Modelo U en punto SDT. Fuente: http//www.sdtcorp.com/trntest.htm y KIT (1995).

1.2.2 IMPORTANCIA DEL TEST DE SOFTWARE

La necesidad de garantizar la alta calidad de los sistemas informáticos (i.e. software) ha aumentado las horas de trabajo de los procedimientos de test respecto a los de análisis, diseño y programación. En los últimos años, diversos autores han publicado documentos en los cuales se constata la preocupación en transformar los procedimientos de test (i.e.

---

[61] KIT. E. (1995). Opus cit., pág. 25.
[62] SCHULMEYER, G. (2000). *Verification and Validation of Modern Software-Intensive Systems.* Boston. Prentice Hall, pág. 39.
[63] SCHULMEYER, G. (2000). Opus cit., pág. 42.
[64] KUVAJA (1994). Opus cit., pág. 78.

un proceso dentro del ciclo de vida de desarrollo de software) en estudios más profundizados.

SHEPARD (2001)[65] comenta sobre la importancia de las técnicas de verificación y validación de software como parte de una disciplina importante de ingeniería de software, las cuales tienen el objetivo de formar desarrolladores de software más efectivos.

1.3 EL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS, MEDIANTE EL USO DE LA METODOLOGÍA TRADICIONAL

El investigador DIEZ (2003)[66] experimenta el uso de la metodología IDEAL en la construcción de sistemas basados en el conocimiento. Sostiene que:

> *"el enfoque no pretende ser exclusivo y en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podrá ser su complemento, adaptándolo convenientemente."*

El autor presenta algunos de los beneficios esperados de la aplicación constante y rigurosa de la metodología y el enfoque presentados. Son los siguientes:

> *"1.El software tendrá menos defectos latentes; como consecuencia, se reducirá el esfuerzo y el tiempo durante las etapas de prueba y mantenimiento.*
> 2. *Se dará una mayor fiabilidad y, por tanto, una mayor satisfacción del cliente.*
> 3. *Se podrán reducir los costos de mantenimiento (un porcentaje sustancial de los costos totales del software).*
> 4. *El tiempo y el costo total del ciclo de vida del software disminuirá."*

REYNOSO (2006)[67] define los métodos ágiles, tales como Lean Development, eXtreme Programming, Adaptive Software Development y GeneXus, como:

---

[65] SHEPARD, T. (2000). *Task-directed software inspection technique: an experiment and case study.* Of the 2000 conference of the Centre for Advanced Studies on Collaborative research. November. Mississauga, Ontario, Canadá. pág.6.

[66] DIEZ, E. (2003). *Aseguramiento de la calidad en la construcción de sistema basados en el conocimiento.* Tesis de Maestría publicada electrónicamente. Instituto Tecnológico de Buenos Aires. Buenos Aires. pág. 141-142.

[67] REYNOSO, C. (2006). *Métodos heterodoxos en el desarrollo de software.* Tesis de Maestría publicada electrónicamente. Universidad de Buenos Aires. Buenos Aires, pág. 1.

"estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas, centradas en la gente o en los equipos, iterativas, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva, y que requieren que el negocio se involucre en forma directa".

La primera tarea que se realiza en el desarrollo de software generalmente es el análisis de datos, donde se estudia la realidad en forma abstracta, identificando los objetos existentes y sus relaciones, y se obtiene como resultado un modelo de datos con las entidades y relaciones encontradas (el modelo E-R). Es fácil ver que existe una correspondencia muy simple entre el modelo de datos y la base de datos necesaria para soportarlo. Por esta razón, lo que sigue es diseñar esa base de datos, partiendo del modelo de datos (ver el Gráfico Nº 9).

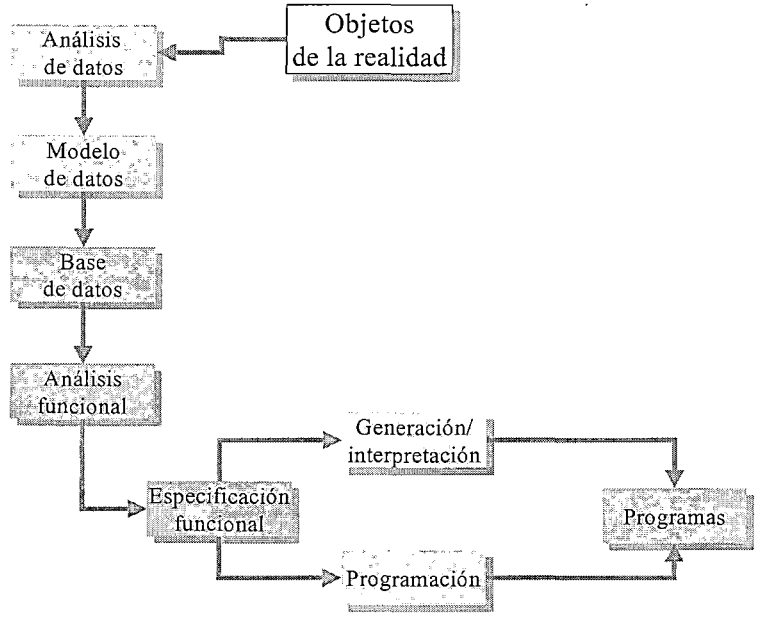


Gráfico Nº 9. Metodología tradicional del desarrollo de software sobre base de datos (elaborado por el autor).

MÁRQUEZ (2006)[68], en el estudio de la secuencia del desarrollo de software sobre base de datos con la metodología tradicional, presenta los pasos: una vez obtenido el modelo de datos, el siguiente es diseñar la base de datos que soporte el modelo de dato. Sin embargo, el modelo de dato no es suficiente para el desarrollo de software sobre base de datos, ya que él mismo describe los datos pero no los comportamientos.

[68] MÁRQUEZ, D. (2006). *Guía Práctica GeneXus. Desarrollo basado en conocimiento.* 1ra ed. Montevideo, Uruguay: Grupo Magró, pág. 6.

Entonces, es necesario recurrir a una tarea adicional llamada análisis funcional, mediante el cual se estudia la organización desde el punto de vista de las funciones existentes. El resultado de dicha tarea es una especificación funcional.

El autor plantea que, una vez que se cuenta con la base de datos y la especificación funcional, ya están dadas las condiciones para crear los software de aplicación sobre base de datos. Además afirma que "existen varias alternativas: 1) Lenguajes de tercera generación (3GL): COBOL, RPG, C, Pascal, ADA, FORTRAN y lenguajes de cuarta generación (4GL): CA IDEAL, INFORMIX 4GL, NATURAL 2, PROGRESS. 2) Programación y/o interpretación (compiladores e intérpretes) poseen un mayor aumento de productividad sobre los 3GL y 4GL. "Estadísticamente, en los lenguajes de cuarta generación se escribe diez veces menos códigos que en un lenguaje de tercera generación."

1.4. EL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS, MEDIANTE EL USO DE LA METODOLOGÍA GENEXUS DE GONDA.

MÁRQUEZ (2006)[69] sustenta que los estudios y experiencias consideran que es un acierto afirmar que "Utilizando GeneXus, la tarea básica del analista GeneXus es la descripción de la realidad". Por esta razón, es común referirse al analista del software que trabaja con GeneXus como "Analista GeneXus" en lugar de "Programador GeneXus". El analista GeneXus en el desarrollo de software sobre base de datos trabaja en alto nivel[70], en vez de realizar tareas de bajo nivel[71] como diseñar archivos, normalizar, diseñar programas, programar, buscar y eliminar los errores de los programas.

Para MÁRQUEZ (2006)[72] , en el aprendizaje del desarrollo de software sobre base de datos con GeneXus de Gonda, el primer paso consiste en crear un nuevo proyecto o base de conocimiento (en inglés *knowledge base*). Una vez creada una nueva base de conocimiento, el siguiente paso es describir las visiones de los usuarios. Para ello, se deben identificar los objetos de la realidad (prestando atención a los sustantivos que los usuarios mencionan en sus descripciones, como por ejemplo: estudiantes,

[69] MÁRQUEZ, D. (2006). Opus cit., págs. 9-10.
[70] Se dice que un analista trabaja en alto nivel, si trabaja con lenguajes de programación muy cercanos al lenguaje humano, tales como: CA IDEAL, INFORMIX 4GL, NATURAL 2, PROGRESS, etcétera.
[71] Se dice que un analista trabaja en bajo nivel, si trabaja con lenguajes de programación muy cercanos al lenguaje de máquina, tales como: COBOL, RPG, C, Pascal, ADA, FORTRAN, etcétera.
[72] MÁRQUEZ, D. (2006). Opus cit., págs. 10-11.

padres, cursos, profesores, clientes, productos, facturas, etc.) y pasar a definirlos mediante los objetos GeneXus.

En forma simplificada, podemos abstraer a la base de conocimiento como una especie de repositorio o contenedor de los objetos GeneXus, donde quedan sistematizados todas las visiones de los usuarios, maximizando la capacidad de inferencia. Esto quiere decir que, interrogando a la base de conocimiento, se puede obtener cualquier conocimiento que se haya registrado en ella o que pueda inferirse lógicamente a partir del conocimiento allí almacenado (ver figura N° 8).

La investigación realizada por los ingenieros Breogán GONDA y Nicolás JODAL (2006)[73] señala que "un diccionario de datos activo colocamos conocimiento, y luego podemos recuperar el conocimiento que hemos colocado, y otro que pueda asociarse a él, básicamente mediante referencias cruzadas. En una base de conocimiento también colocamos conocimiento (en el caso de la metodología GeneXus, en forma implícita, colocada en ella el conocimiento capturado en las visiones de los usuarios y podemos recuperar el conocimiento que hemos colocado, y otra que pueda ser inferido lógicamente de aquél. En particular, el modelo de datos, el esquema de la base de datos, los programas, el análisis de impacto de cambios, etc.)".

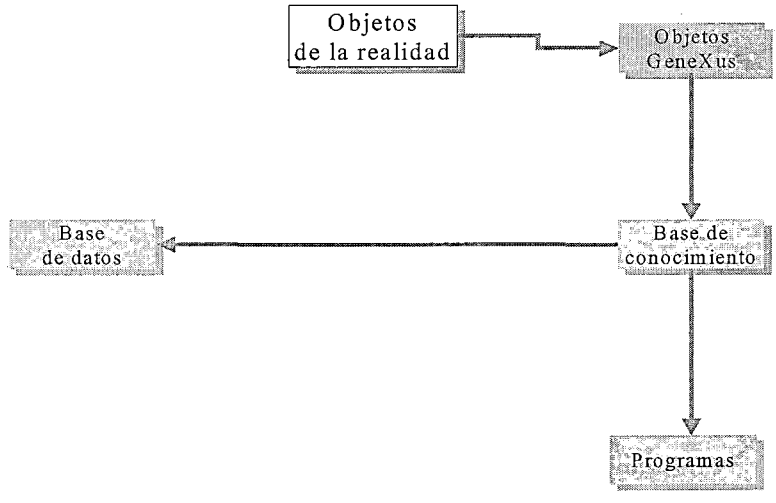


Gráfico N° 10. Metodología GeneXus de Breogán Gonda del desarrollo de software sobre base de datos (elaborado por el autor).

[73] GONDA, B. y JODAL, J. N. (1995). "*Proyecto GeneXus: Resumen del Trabajo Ganador del Premio Nacional de Ingeniería 1995*". Academia Nacional de Ingeniería. Montevideo, Uruguay. pág. 5.

Como se muestra en el Gráfico N° 10, si un objeto de la realidad cambia, o se identifican nuevas características de los objetos de la realidad, o se encuentran objetos de la realidad aún no modelados, el analista GeneXus debe reflejar dichos cambios en los objetos GeneXus que correspondan.

En el documento de Artech Inc-GeneXus The First Intelligent Toll (2004)[74], los generadores de base de datos y programas, mediante aproximaciones sucesivas, permiten automáticamente realizar las modificaciones que sean necesarias tanto en la base de datos como en los programas asociados.

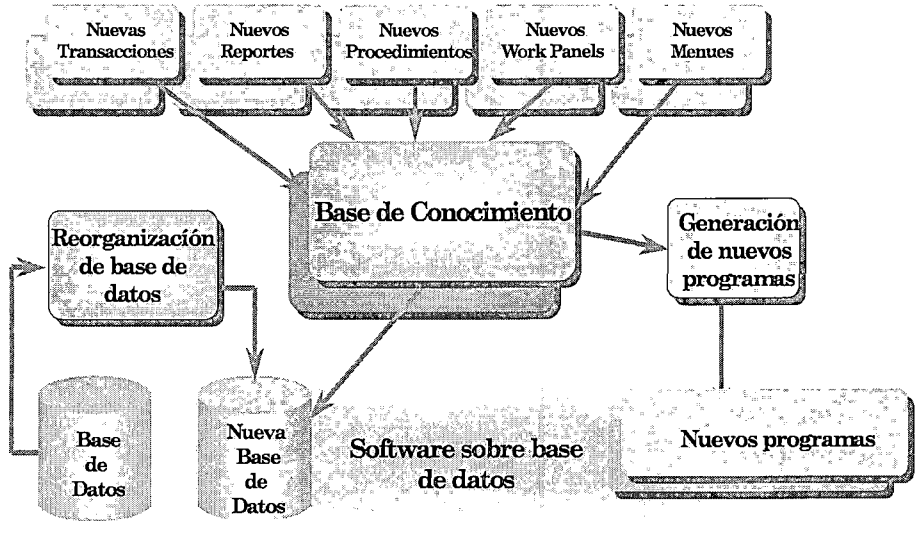


Gráfico N° 11. Filosofía del desarrollo de software sobre base de datos mediante la metodología GeneXus de Breogán Gonda (Fuente: *Manual de referencia de GeneXus.* Junio, 1999, Copyright (c) Artech Inc).

Para MARQUEZ (2006)[75], la metodología GeneXus de Gonda es una metodología incremental (ver Gráfico N° 11), pues parte de la base de la construcción de un software y se realiza mediante aproximaciones sucesivas. En cada momento, el analista GeneXus de desarrollo del software define el conocimiento que tiene; y si luego pasa a tener más conocimiento (o simplemente es diferente), lo refleja en la base de conocimiento, y GeneXus se ocupará automáticamente de todas las adaptaciones en la base de datos y programas. El autor argumenta que si la metodología no fuera capaz y permitiera realizar automáticamente las modificaciones en la base de datos y programas

[74] Artech Inc, GeneXus-GeneXus The First Intelligent Toll (2004). *Curso básico de GeneXus 8.0.* Uruguay: Montevideo, pág. 12.
[75] MÁRQUEZ, D. (2006). Opus cit., pág.15.

conforme se realicen cambios que así lo requieran, el desarrollo incremental sería inviable (ver Gráfico Nº 12).

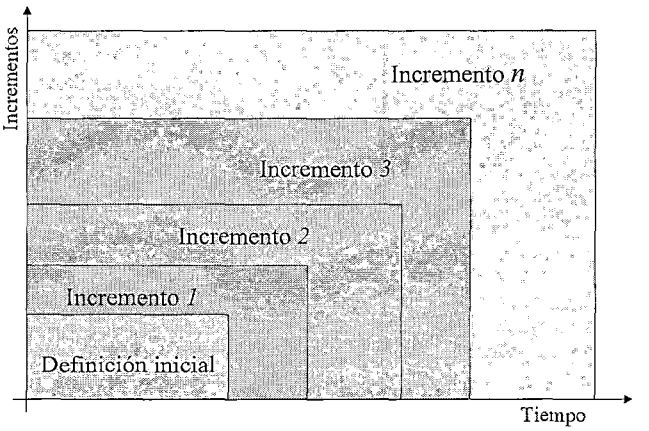


Gráfico Nº 12. Esquema de la metodología incremental (Fuente: *Manual de referencia de GeneXus*. Junio, 1999, Copyright (c) Artech Inc).

1.5. ANÁLISIS LEXICOLÓGICO DEL TÉRMINO COMPETENCIA

COROMINAS (1998)[76] presenta, desde el punto de vista etimológico, el origen del término "competencia" en el verbo latín "competere" que significa "ir al encuentro una cosa de otra, encontrarse", y se considera también las acepciones como "responder a, corresponder", "estar en buen estado", "ser suficiente", dando lugar a los adjetivos "competens-entis" (participio presente de competo) en la línea de competente, conveniente, apropiado para; y los sustantivos "competio-onis" competición en juicio y "competitor-oris" competidor, concurrente, rival.

COROMINAS expresa que nos encontramos con dos verbos en castellano "competir" y "competer" que proviniendo del mismo verbo latino ("competere") se diferencian significativamente, pero a su vez entrañan semánticamente el ámbito de la competencia.

a. "Competer": pertenecer o incumbir, dando lugar al sustantivo competencia y al adjetivo competente (apto, adecuado).
b. "Competir": pugnar, rivalizar, dando lugar también al sustantivo competencia, competitividad, y al adjetivo competitivo.

[76] COROMINAS, J. (1998). *Breve. diccionario etimológico de la Lengua Castellana*. 4ta ed. Madrid, España: Gredos S.A. pág. 163.

TEJADA (1999)[77] refiere la competencia como "capacitación", al grado de preparación, saber hacer, conocimientos y pericia de una persona como resultado del aprendizaje. En este caso, la competencia alude directamente a las capacidades y habilidades de una persona, que son necesarias desarrollar a través de la formación. También se puede considerar en este punto la competencia como cualificación, referida básicamente a la formación necesaria para tener la competencia profesional deseada.

1.5.1. CARACTERIZACIÓN DE LAS COMPETENCIAS

Actualmente el término de competencia nos pone en dificultad al intentar definirlo y más allá de esta dificultad, es necesario concretar y llegar a algunos puntos de síntesis de definición para los objetivos de la investigación.

FERRÁNDEZ (1997)[78] presenta una primera característica al sostener que el concepto de competencia "se comporta como todo un conjunto de conocimientos, procedimientos y actitudes combinados, coordinados e integrados", en el sentido que el individuo ha de "saber hacer" y "saber estar" para el ejercicio profesional. El dominio de estos saberes lo hacen "capaz de" actuar con eficacia en situaciones problemáticas de la vida profesional; y mediante el conjunto que forman las capacidades se logran las competencias a través de un proceso de aprendizaje.

1.6. REVISIÓN DE LA LITERATURA SOBRE APRENDIZAJE

1.6.1. TEORÍA DE APRENDIZAJE SIGNIFICATIVO

COLL (2000)[79] refiere que las teorías de aprendizaje son:

> *"un conjunto global de fundamentos, hechos, enfoques y perspectivas teóricas que intentan ofrecer explicaciones más o menos generales de los elementos o factores implicados en el proceso de cambio que las personas experimentan como resultado de su experiencia y de su relación con su medio; es utilizada con frecuencia, en un sentido más estricto, para designar un*

---

[77] TEJADA, J. (1999). *Acerca de las competencias profesionales.* Documento publicado en la "Revista Herramientas". Parte 1. España. pág.4.

[78] FERRÁNDEZ, A. (1997). *El perfil profesional de los formadores.* (Documento mimeografiado). Universitat Autónoma de Barcelona, Departamento de Pedagogía Aplicada, pág. 4.

[79] COLL, C. y otros (2000). *Psicología do ensino.* 2da Ed. Sau Paulo, Brasil: Editora Artes Médicas Sull Ltda., pág. 215.

subconjunto específico de marcos teóricos, que son caracterizados porque se inspiran, de manera más o menos directa, en la tradición conductista de la psicología".

El conjunto de los contenidos escolares específicos:

"- son saberes preexistentes.

- son saberes que permiten el desarrollo de determinadas capacidades de los alumnos.

- son saberes que constan en el currículo escolar porque requieren una ayuda específica para ser aprendidos.

- son saberes para los cuales se reclaman una apropiación significativa y con sentido por parte de los alumnos.

- son saberes que se ordenan en el seno de diversas áreas curriculares".

Según AUSUBEL (1983),

"Un aprendizaje se dice que es significativo cuando una nueva información (concepto, idea, proposición) adquiere significado para el aprendiz a través de una especie de anclaje en aspectos relevantes de la estructura cognitiva preexistente del individuo, o sea en conceptos, ideas, proposiciones ya existentes en su estructura de conocimientos (o de significados) con determinado grado de claridad, estabilidad y diferenciación".

Por lo tanto, en el aprendizaje significativo hay una interacción entre el nuevo conocimiento y el ya existente, en la cual ambos se modifican. En la medida en que el conocimiento sirve de base para la atribución de significados a la nueva información, él también se modifica, o sea, los conceptos van adquiriendo nuevos significados, tornándose más diferenciados, más estables. La estructura cognitiva está constantemente reestructurándose durante el aprendizaje significativo. El proceso es dinámico, por lo tanto el conocimiento va siendo construido. Este aprendizaje, según COLL (1997)[80], consiste en establecer jerarquías conceptuales que prescriben una secuencia descendente: partir de los conceptos más generales e inclusivos hasta llegar a los más específicos, pasando por los conceptos intermedios.

[80] COLL, C. y otros. (1987). *Psicología y Currículum* . 1ra Ed. México, Paidós Mexicana, pág. 32.

Asimismo, COLL (1987)[81] refiere que la investigación sobre el aprendizaje ha revelado que en diferentes áreas del conocimiento los sujetos combinan conceptos y procedimientos en forma de reglas para la acción cuando las condiciones de una tarea o trabajo así lo exigen. Este tipo de proceso tiene como resultado planes de acción alternativos para resolver problemas. Este tipo de conocimiento que se adquiere suele denominarse como "conocimiento estratégico", ya que involucra activar cuál, cuándo y por qué un determinado dominio del saber es aplicable

COLL (1987)[82] comenta los trabajos de Ausubel y colaboradores, en relación con su propuesta de análisis de contenido. Sostiene que "ésta consiste en establecer jerarquías conceptuales que prescriben una secuencia descendente: partir de los conceptos más generales e inclusivos hasta llegar a los más específicos, pasando por los conceptos intermedios".

PIAGET (1972)[83] sostiene que "conocer" es "actuar mediante la realidad que nos envuelve". El sujeto conoce en la medida que modifica la realidad "a través de los hechos". Los esquemas de conocimiento de Piaget hacen referencia a los aspectos generales de hechos, que consiste en reunir, comparar, separar, juntar, ordenar, etc.; y que pueden ser aplicados a cualquier realidad. Además, según Piaget, un esquema corresponde a un aspecto organizativo de una acción o hecho, la estructura que corresponde a ese hecho puede repetirse o ser repetido y ser aplicado con ligeras modificaciones en situaciones distintas para conseguir objetivos similares; y éstos denominan, según Piaget, "a aquellas acciones que permiten transportar, generalizar, o diferenciar de una u otra manera, o que es común a las diversas repeticiones o aplicaciones de la misma situación".

Para COLL (1987), la psicología de la enseñanza no dispone, en este momento, de una teoría única y acepta globalmente que pueda ofrecernos el poder explicar en forma completa y detallada los procesos del aprendizaje escolar; por el contrario, en la dimensión teórico-conceptual de la disciplina, coexisten diversas teorías y enfoques sobre el aprendizaje escolar.

---

[81] COLL, C. y otros. (1987). Opus cit., pág. 78.
[82] COLL, C. y otros. (1987). Opus cit., págs. 21-28.
[83] PIAGET ,J. (1972). La euilibración de las estructuras cognitivas. Problema central de desarrollo. Madrid: Siglo XXI. (Publicação original em francés, no ano de 1975), pág. 250 -251.

AUSUBEL (1983)[84] refiere que el aprendizaje del estudiante depende de la estructura cognitiva previa que se relaciona con la nueva información. Debe entenderse, por "estructura cognitiva", al conjunto de conceptos, ideas que un individuo posee en un determinado campo del conocimiento, así como su organización. En el proceso de orientación del aprendizaje, es de vital importancia conocer la estructura cognitiva del estudiante; no sólo se trata de saber la cantidad de información que posee, sino cuáles son los conceptos y proposiciones que maneja así como de su grado de estabilidad.

Los principios de aprendizaje propuestos por Ausubel ofrecen el marco para el diseño de herramientas metacognitivas que permiten conocer la organización de la estructura cognitiva del estudiante, lo cual permitirá una mejor orientación de la tarea educativa. Ésta ya no se verá como una tarea que deba desarrollarse con "mentes en blanco" o que el aprendizaje de los estuantes comience de "cero", pues no es así, sino que los educandos tienen una serie de experiencias y conocimientos que afectan su aprendizaje y pueden ser aprovechados para su beneficio.

AUSUBEL (1983)[85] menciona que el aprendizaje es significativo cuando "los contenidos son relacionados de modo no arbitrario y sustancial (no al pie de la letra) con lo que el estudiante ya sabe" y "por relación sustancial y no arbitraria se debe entender que las ideas se relacionan con algún aspecto existente específicamente relevante de la estructura cognoscitiva del estudiante, como una imagen, un símbolo ya significativo, un concepto o una proposición". Esto quiere decir que en el proceso educativo es importante considerar lo que el individuo ya sabe, de tal manera que establezca una relación con aquello que debe aprender. Este proceso tiene lugar si el estudiante tiene en su estructura cognitiva conceptos (ideas, proposiciones, estables y definidos), con los cuales la nueva información puede interactuar.

Para AUSUBEL (1983)[86], el aprendizaje significativo ocurre cuando una nueva información "se conecta" con un concepto relevante preexistente en la estructura cognitiva. Esto implica que las nuevas ideas, conceptos y proposiciones pueden ser aprendidos significativamente en la medida en que otras ideas, conceptos o proposiciones relevantes estén adecuadamente claras y disponibles en la estructura

---

[84] AUSUBEL, D. y otros (1983). *Psicología Educativa: Un punto de vista cognoscitivo.* 2da Ed. México: TRILLAS, pág. 9.
[85] AUSUBEL, D. y otros (1983). Opus cit., pág. 18
[86] AUSUBEL, D. y otros (1983). Opus cit., pág. 22.

cognitiva del individuo. La característica más importante del aprendizaje significativo es que produce una interacción entre los conocimientos más relevantes de la estructura cognitiva y las nuevas informaciones (no es una simple asociación), de tal modo que éstas adquieren un significado y son integradas a la estructura cognitiva de manera no arbitraria y sustancial, favoreciendo la diferenciación, evolución y estabilidad.

COLL (2000)[87] plantea que es importante considerar los aprendizaje desde la perspectiva de los conocimientos referidos a hechos, conceptos, principios y teorías; el aprendizaje de los procedimientos, referidos a conocer el manejo de determinados instrumentos, capacidad de una persona para realizar una secuencia de actividades, mostrar habilidad específica en la resolución de problemas de su vida diaria y profesional; y el aprendizaje actitudinal, referido a crear hábitos para el cuidado personal en el trabajo escolar u otro, participación y cooperación en grupo, fraternidad, solidaridad, integración intercultural y responsabilidad.

Para COLL (1997)[88], desarrollo, aprendizaje y enseñanza son tres elementos íntimamente relacionados entre sí, ya que el nivel de desarrollo afectivo condiciona los posibles aprendizajes que el individuo puede realizar gracias a la enseñanza, pero ésta, a su vez, puede llegar a modificar el nivel de desarrollo efectivo del alumno mediante los aprendizajes específicos que promueve. En COLL, encontramos que el aprendizaje escolar es un proceso activo, desde el punto de vista del alumno, en el cual éste construye, modifica, enriquece y diversifica sus esquemas de conocimiento con respecto a los distintos contenidos escolares, a partir del significado y el sentido que puede atribuir a esos contenidos y al propio hecho de aprenderlos. La enseñanza debe entenderse como una ayuda al proceso de aprendizaje; sólo ayuda, porque la enseñanza no puede sustituir la actividad mental constructiva del alumno ni ocupar su lugar.

NOVOA (2006)[89] define que los "contenidos" son las capacidades y competencias que se espera que el estudiante adquiera, y constituyen el cuerpo de conocimientos que llevarán al estudiante a desarrollar las capacidades y habilidades esperadas. Éstos pueden ser del orden: de aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales. Según el autor, lo conceptual se refiere a:

---

[87] COLL, C. y otros (2000). Opus cit., pág. 304.
[88] Coll, C. (1997) Aprendizaje escolar y construcción del conocimiento. 2da Ed. Paidós. Barcelona, España. pág. 98.
[89] NOVOA, L. (2006). *Red de maestros de maestros.* Chile. Noviembre 2006, On line: http://www.rmm.cl/index sub.php?id_contenido= 8693&id_seccion=2094 &id_portal=329). pág.3.

"hechos, conceptos y principios; y los conocimientos conceptuales se refieren al conjunto de objetos, hechos o símbolos que tienen ciertas características comunes. Los sistemas conceptuales hacen referencia a imágenes mentales y expresan hechos, datos, conceptos, principios, teorías que constituyen el saber de la ciencia. Ordinariamente consisten en conjuntos de datos que el alumno debe aprender de memoria sin necesidad de comprenderlos (representan el "Saber" de la educación)".

COLL (2000)[90] expresa que el aprendizaje de conocimientos conceptuales se fundamenta en que el estudiante:

"1. Presenta el mundo, tomando en cuenta su identificación con su cultura.
2. Identifica los aspectos relevantes de la realidad y manifiesta sobre el qué se debe hacer.
3. Debe tener una visión coherente, estructurada y organizada de la realidad.
4. Debe reducir la complejidad y la variabilidad de los acontecimientos a una estructura de entidades que posiblemente.
5. Debe predecir acontecimientos de manera que no sea necesario realizar aprendizajes continuos y garantizar la funcionalidad de los ya realizados.
6. Debe comprender y explicar con mayor profundidad todo lo que nos rodea y establecer relaciones cada vez más complejas."

1.6.2. APRENDIZAJE DE CONTENIDOS CONCEPTUALES

NOVOA (2006)[91] se refiere al aprendizaje de contenidos factuales y los define como "hechos, acontecimientos, situaciones, datos y fenómenos concretos". Información que debemos saber porque asociada a otro tipo de contenidos, más complejos, permitirán comprender los problemas de la vida cotidiana y profesional. NOVOA refiere que los contenidos conceptuales son:

[90] COLL, C. y otros (2000). Opus. cit. págs. 304.
[91] NOVOA, L. (2006). Opus. cit. pág. 5.

"ideas y conceptos, que los estudiantes deben alcanzar en una etapa determinada de su formación".

El aprendizaje supone la incorporación de todos los componentes del hecho, e implican un recuerdo con la mayor fidelidad posible. Aprender hechos supone en síntesis, repetición, memorización, las que a su vez requieren de estrategias que permitan una asociación significativa entre ellos y otros conceptos o situaciones. Para ello, se usan listas o agrupaciones significativas, cuadros o representaciones gráficas, visuales, o asociaciones con otros conceptos fuertemente asimilados.

COLL (2000)[92] refiere que los conceptos aluden a un conjunto de hechos, objetos o símbolos que tienen características comunes (mamífero, ciudad, potencia, concierto); y los principios, a los cambios en los hechos, objetos o situaciones en relación con otros (leyes de termodinámica, principio de Arquímedes, el tercio excluido, etc.). En ambos casos, su aprendizaje requiere comprender de qué se trata, qué significa. Por tanto no basta su aprendizaje literal, es necesario que el estudiante o aprendiz sepa utilizarlo para interpretar, comprender o exponer un fenómeno. Por ello, aprender conceptos y principios es toda una reforma de las estructuras mentales. Implica una construcción personal, una reestructuración de conocimientos previos, con el fin de construir nuevas estructuras conceptuales que permitan integrar tanto estos conocimientos como los anteriores, a través de procesos de reflexión y toma de conciencia conceptual.

1.6.3. APRENDIZAJE DE CONTENIDOS PROCEDIMENTALES

ZAVALA (1993)[93] define los contenidos procedimentales como: "... un conjunto de acciones ordenadas y finalizadas, es decir dirigidas a la consecución de un objetivo". ZAVALA precisa que el aprendizaje procedimental se refiere a la adquisición y/o mejora de nuestras habilidades, a través de la ejercitación reflexiva en diversas técnicas, destrezas y/o estrategias para hacer cosas concretas. Se trata de determinadas formas de actuar, cuya principal característica es que se realizan de forma ordenada: "Implican secuencias de habilidades o destrezas más complejas y encadenadas que un simple hábito de conducta". Los principales tipos de contenidos procedimentales son las técnicas y estrategias.

---

[92] COLL, C. y otros (2000). Opus cit., págs. 298-299.
[93] ZAVALA, A. (1993). *Cómo trabajar los contenidos procedimentales en el aula*. Barcelona, España: Graó /ICEU, pág. 81.

VALLS (1995)[94] define los procedimientos como "un conjunto de acciones ordenadas a la consecución de una meta". Asimismo, menciona que "no debe confundirse un procedimiento con una determinada metodología. El procedimiento es la destreza que queremos ayudar a que el alumno construya. Es, por tanto, un contenido escolar de la planificación e intervención educativa, y el aprendizaje de ese procedimiento puede trabajarse mediante distintos métodos".

Por otra parte, los contenidos procedimentales designan conjuntos de acciones, de formas de actuar en pos de metas. Se trata de unos conocimientos con los cuales nos referimos al saber hacer (con las cosas, o sobre las cosas, las personas, la información, las ideas, los números, la naturaleza, los símbolos, los objetos, etcétera) y su aprendizaje supone, en último término, que se sabrá usar y aplicar en otras situaciones de persecución de metas. En ellos agrupamos las habilidades y capacidades básicas para actuar de alguna manera, a las estrategias que uno aprende para solucionar problemas o a las técnicas y actividades sistematizadas relacionadas con aprendizajes concretos.

Es lógico pensar que los procedimientos forman parte del currículo porque con ellos, una vez aprendidos de manera significativa, los alumnos sabrán hacer cosas. Sabrán, por ejemplo, hacerlas funcionar, transformarlas o producirlas, medirlas, observarlas, representarlas, graficarlas, organizarlas, leerlas, elaborarlas, etc. Por esta razón se afirma que se aprende en definitiva cuando se adquieren los procedimientos, es una vía, un camino, un recurso para llegar a objetivos con la particularidad de que lo más interesante del aprendizaje es que se trata de adquirir una secuencia de pasos o componentes, una secuencia ordenada de obrar. Hablar de enseñar y aprender contenidos procedimentales quiere decir que insistimos en una determinada orden de actuar hacia una meta (VALLS, 1995)[95].

ZAVALA (1993)[96] define las técnicas como: "encadenamientos de acciones complejas que requieren un cierto entrenamiento explícito, basado en un aprendizaje asociativo, por repetición, que debe concluir en una automatización de la cadena de acciones, con el fin de que la ejecución sea más rápida y certera, al tiempo que menos costosa en recursos cognitivos. Las técnicas son muy eficaces cuando nos enfrentamos a

[94] VALLS, E. (1995). Los Procedimientos. Aprendizaje, enseñanza y evaluación. Barcelona. Ed. Morsori, págs. 19-20.
[95] VALLS, E. (1995). Opus cit., págs. 28-29.
[96] ZAVALA, A. (1993). Opus cit., págs. 18-19.

ejercicios, tareas rutinarias, siempre iguales a sí mismas, pero cuando la situación varía en algún elemento importante, no basta con dominar la técnica, hay que saber también modificarla sobre la marcha para adecuarla a las nuevas condiciones".

Para ZAVALA, el aprendizaje de estrategias permite planificar, tomar decisiones y controlar la aplicación de las técnicas para adaptarlas a las necesidades específicas de cada tarea. En la estrategia no se adquieren aprendizajes por procesos asociativos, es decir, procesos en los que se desarrolla la repetición, sino por procesos de reestructuración de la propia práctica, producto de una reflexión y toma de conciencia sobre lo que hacemos y cómo lo hacemos.

POZO (1999)[97] expresa que "aprendemos estrategias a medida que intentamos comprender o conocer nuestras propias técnicas y sus limitaciones, y ello requiere que hayamos aprendido a tomar conciencia y reflexionar sobre nuestra propia actividad y cómo hacerla más efectiva".

A diferencia de las técnicas, no es posible adquirir las estrategias por entrenamiento, porque su uso supone la aplicación organizada y controlada de técnicas y recursos disponibles. ¿Qué condiciones son fundamentales para el aprendizaje de contenidos procedimentales? La realización de las acciones que conforman los procedimientos es una condición fundamental para el aprendizaje: se aprende a hablar, hablando; a dibujar, dibujando; a observar, observando. La ejercitación múltiple es necesaria para el aprendizaje de una técnica; no basta con realizar alguna vez las acciones del contenido procedimental, hay que realizar tantas veces como sea necesario las diferentes acciones o pasos de dichos contenidos de aprendizaje. La reflexión sobre la misma actividad es un elemento imprescindible que permite tomar conciencia de la actuación. No basta con repetir el ejercicio, habrá que ser capaz de reflexionar sobre la manera de realizarlo y sobre las condiciones ideales de su uso. Esto implica realizar ejercitaciones, pero con el mejor soporte reflexivo que nos permita analizar nuestros actos, y por consiguiente, mejorarlos.

Según POZO, para la reflexión, hace falta tener un conocimiento significativo de contenidos conceptuales asociados al contenido procedimental que se ejercita o se

[97] POZO, J. I. (1999). *Aprendices y maestros.* La nueva cultura del aprendizaje. 3ra ed. Madrid, España: Alianza, pág.54.

aplica. Así, por ejemplo, yo puedo revisar una composición a partir de un conjunto de reglas morfosintácticas que me permitirán establecer errores y hacer modificaciones posteriores. La aplicación en contextos diferenciados se basa en el hecho de que aquello que hemos aprendido será más útil en la medida en que podamos utilizarlo en situaciones siempre imprevisibles. Las ejercitaciones han de realizarse en contextos diferentes para que los aprendizajes puedan ser utilizados en cualquier ocasión.

COLL (2000)[98] manifiesta que se acostumbra clasificar los procedimientos atendiendo sobre todo la diversidad de los elementos que los definen y podemos distinguir entre:

"1. Destrezas motoras o corporales y cognitivas.
2. Procedimientos simples (para tomar en pocos las decisiones para cumplir una secuencia) y complejos (con muchos pasos de decisiones).
3. Procedimientos de naturaleza algorítmica y de naturaleza heurística.
4. Procedimientos específicos de tareas escolares y procedimientos generales."

Una clasificación de las características de la estrategia de aprendizaje planteadas por COLL es:

"- El tipo de aprendizaje, estrategia de aprendizaje, finalidad u objetivo, técnica o habilidad:
- Por asociación, repaso, repaso simple: repetir
- Por asociación, repaso, apoyo al repaso: subrayar, destacar, copiar.
- Por asociación, elaboración, simple: palabra clave, Imagen, rimas y abreviaturas, códigos.
- Por asociación, elaboración, compleja: formas analógicas, lee textos
- Por reestructuración, organización, clasificar: formar categorías.
- Por reestructuración, organización, Jerarquizar: formar redes de conceptos, identificar estructuras, hacer mapas conceptuales."

[98] COLL, C. y otros (2000). Opus. cit. págs. 114-121.

Para explicar cómo se producen esos aprendizajes, muchos psicólogos de la corriente de enseñanza cognitiva parten comúnmente de distinguir entre conocimientos declarativos y conocimientos procedimentales. Esta forma de aprendizaje permite que el estudiante pueda aprender a aprender, a veces mediante la metodología de "prueba y error" (ver la clasificación adoptada en la Tabla Nº 1).

1.6.4. APRENDIZAJE DE CONTENIDOS ACTITUDINALES

POZO (1999)[99] define las actitudes como: "tendencias o disposiciones adquiridas y relativamente duraderas a evaluar de un modo determinado un objeto, persona, suceso o situación y a actuar en consonancia con dicha evaluación". Son disposiciones afectivas y racionales que se manifiestan en los comportamientos; por ello, tienen un componente conductual (forma determinada de comportarse), rasgos afectivos y una dimensión cognitiva no necesariamente consciente.

POZO (1999)[100] señala que "la consistencia de una actitud depende en buena medida de la congruencia entre distintos componentes. Una actitud será más firme y consistente, y con ello más estable y transferible, cuando lo que hacemos es congruente con lo que nos gusta y lo que creemos".

COLL (2000)[101] presenta la clasificación del aprendizaje de los conocimientos actitudinales, y manifiesta que las actitudes y valores trascienden las situaciones específicas y se manifiestan de manera personalizada, y por ende se refleja en la sociedad; los valores y actitudes que se encuentran en los objetivos de la etapa de enseñanza y aprendizaje se fundamentan en:

"1. La autonomía y la iniciativa.
2. La salud y la higiene.
3. La participación y la solidaridad.
4. El respeto a los valores de los otros.
5. La responsabilidad.
6. La convivencia y la paz.
7. La tradición histórica y cultural.
8. Conservación del medio ambiente físico y natural.
9. La identidad nacional y cultural."

[99] POZO, J. I. (1999). Opus cit., págs. 34-37.
[100] POZO, J. I. (1999). Opus cit., pág. 76.
[101] COLL, C. y otros (2000). Opus cit., págs. 422-329.

Por lo tanto, después de haber presentado los fundamentos del aprendizaje significativo, desde la perspectiva del aprendizaje de conocimientos conceptuales, aprendizaje de conocimientos procedimentales y aprendizaje de conocimientos actitudinales, según los fundamentos teóricos de Coll, Ausubel y Pozo, se utiliza la clasificación que se muestra en la Tabla N° 1, para poder evaluar el aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales del desarrollo de software sobre base de datos.

Tabla N° 1. Clasificación de los aprendizajes, según COLL (2000).

Conocimiento conceptual	"Consiste en comprender y ordenar el mundo de las ideas, en categorías y relaciones significativas."
Datos	Información de fácil enunciado.
Hechos	Sucesos o acontecimientos.
Conceptos	Definiciones acerca del objeto de estudio.
Principios o leyes	Relaciones entre dos o más conceptos o leyes.

Conocimiento procedimental	"Conjunto de acciones, orientadas a la consecuencia de una meta. Saber hacer, aplicación de los cocimientos."
Generales	Secuencia de actualización estable sin variaciones en la realización, observación, descripción, datos, manipulación.
Algoritmos	Secuencia de acciones y decisiones que se debe respetar para resolver determinados problemas.
Heurísticos	Orientar y ejecutar hacia un resultado óptimo, secuencia o un problema.
Destrezas y habilidades	Uso correcto de la información recibida.
Técnicas	Pasos precisos para resolver problemas.
Estrategias	Planteamiento de diferentes formas de resolver un problema.
Motriz cognitivo	Aprender a aprender.

Conocimiento actitudinal	"Conjunto de acciones, orientadas a la consecuencia de una meta. Saber hacer, aplicación de los cocimientos."
Valores	Principios éticos, las personas asumen compromisos.
Normas	Patrones de conducta compartidos por un grupo social.
Actitudes	Autonomismo.
Juicios valorativos	Juicio reflexivo.

Por los fundamentos expuestos, es muy difícil hablar de quién aprende sin referirse inmediatamente a qué contenidos aprende y a cómo se ayuda al estudiante en este proceso para que sea un éxito. Basándonos en esta apreciación, se ha analizado los aspectos de aprendizaje de conceptos, procedimientos y actitudes poniéndolas en relación con las oportunidades de enseñanza que el docente brinda en el proceso de enseñanza y aprendizaje al cuestionamiento: ¿Qué permite al alumno aprender

conceptos, procedimientos y actitudes en la escuela? (COLL, 2000). Referido al aprendizaje de los conocimientos conceptuales, lo que, entre otros requisitos, le permite al estudiante aprender de manera significativa conceptos en la escuela es: a) Poseer una serie de saberes personales, b) Tener un profesorado dispuesto a trabajar tomando al estudiante como el centro de su intervención. Referido al aprendizaje de los procedimientos es: a) Saberes personales del estudiante, b) Disposición del docente a enseñar en la construcción del propio conocimiento procedimental. Por último, referido al aprendizaje de los conocimientos actitudianales es: a) Saberes personales del estudiante, b) Intervención del docente en la construcción de actitudes por parte del estudiante.

A modo de resumen, el aprendizaje en el nivel universitario es un proceso complejo que implica en el estudiante a una dedicación integral a construir su conocimiento; es él quien aprende. Sin embargo, hacer posible esto es una aventura colectiva. En primer lugar, porque la sociedad es un ente continuamente exigente para con las capacidades de todos los que la componen y con ellos contribuye a concretar nuestras propias exigencias. En segundo lugar, porque la cultura (usos, costumbres, saberes de diferentes tipos, valores) nos hace saber, en cierto modo, quiénes somos, y poder apropiarnos de ella, revisar críticamente y contribuir a renovarla constantemente; a su vez asume, responsabilidades en la elaboración de nuestra identidad. Y en tercer lugar, porque sin la contribución del docente consciente de que el conocimiento es una construcción, el aprendizaje universitario sería un incierto viaje de dudosas consecuencias.

1.6. DEFINICIÓN DE TÉRMINOS BÁSICOS

Se presenta a continuación la definición de términos básicos utilizados en la presente investigación:

Aprendizaje significativo. Es la acumulación de conocimientos nuevos sobre experiencias previas que tiene el estudiante, y es de vital importancia y de utilidad concreta para su formación. Estos conocimientos son teóricos (conceptuales), prácticos (procedimentales), y valorativos (actitudinales) que se adquieren mediante actividades por descubrimiento.

Hardware. Es un conjunto de componentes físicos de una computadora.

Software. Conjunto de instrucciones escritas para la computadora. Las principales categorías de software son: Software de sistema (sistema operativo) y software de aplicación (software que realiza tareas concretas).

Módulo didáctico. Son unidades temáticas sistematizadas en forma ordenada y gradual, que se desarrolla de lo simple a lo complejo. En esta investigación, los módulos didácticos son elaborados en texto, de un caso práctico de estudio y desarrollados secuencialmente mediante la aplicación de la metodología GeneXus.

Dato. Registro de hechos o acontecimientos. Unidad mínima de información.

Información. Un conjunto de datos ordenados de alguna manera.

Acción. Una actividad deseada por el autor.

Actividad. Término de acción y comportamiento. Esta palabra se usa en el sistema de actividad humana para enfatizar que tales sistemas no son descripciones de acciones observadas en el mundo real.

Ambiente. En el modelo formal del sistema, lo que permanece fuera de los límites del sistema.

Comportamiento (de un sistema). Es el transcurso del tiempo entre las variables del estado de un sistema.

Metodología de sistemas duros. Es una metodología de desarrollo de sistemas, conocido también como "ingeniería de software" para enfrentar problemas del mundo real en los cuales los objetivos o fin a lograrse puede ser tomado como dado. Por tanto, el sistema es ingeniado para lograr el objetivo establecido.

Metodología de sistemas blandos. Es una metodología de desarrollo de sistemas, para enfrentar problemas del mundo real en la cual los fines conocidos como deseables no pueden ser tomados como dados. La metodología de sistemas blandos está basado en la "posición fenomenológica".

Campo. Es la unidad más pequeña a la cual uno puede referirse en un programa de computadora.

Registro. Es un conjunto de campos relacionados entre sí.

Archivo. Es un conjunto de registros del mismo tipo.

Ingeniería de software. Es una disciplina o área de la información o ciencias de la computación, que ofrece métodos o técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo.

Sistema. Una unión de partes conectadas de una manera organizada que ha sido identificado por alguien como un interés especial y que tiene una conducta singular (hace algo más que solamente existir) o, también, un conjunto estructurado de objetos y/o atributos, unidos o relacionados entre sí.

Modelo. Una construcción intelectual y descriptiva de una entidad en la cual por lo menos un observador tiene interés. El observador podría relacionar su modelo y, si es propicio, sus mecanismos, con lo que se aprecia en el mundo.

Modelo de proceso. Es determinar el orden de las fases involucradas en el desarrollo y evolución de software y establecer los criterios de transición para avanzar de una fase a la próxima.

Sistema abierto. Un sistema que está conectado e interactúa con su entorno.

Registro lógico. Es la representación de la percepción del analista de lo que es un registro de datos.

Registro físico. Es la unidad de transferencia de datos entre el dispositivo de almacenamiento de datos y la memoria principal.

Base de datos. Conjunto de datos almacenados entre los que existen relaciones lógicas.

Sistema de gestión de la base de datos (SGBD). Es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma.

Algorítmico. Conjunto de operaciones que siguen un proceso predefinido para la solución de un problema.

Requerimiento de sistema. Es la especificación de qué información debe brindar el software, sistema o aplicación informática.

Ampliación de requerimiento. Se refiere a cuando a partir de nuevos requerimientos es necesario agregar funciones o procesos al software que está en proceso de desarrollo o que está implementado. Se agregan funciones a las ya existentes, aumenta el número de funciones.

Software. Es un conjunto de tareas desarrolladas por un sistema informático. Se usan como sinónimos sistema o aplicación informática.

Clave primaria. Es un conjunto mínimo de atributos que sirven para identificar de manera única un determinado registro de la tabla, entidad u objeto.

Usuario. Persona que usa el sistema informático. Persona que contrata el desarrollo del sistema. Suele usarse cliente como sinónimo.

Comienzo del proyecto. Conjunto de actividades iniciales de un proyecto donde se establecen los objetivos, límites, alcances, información a brindar y alguna particularidad específica de la aplicación a desarrollar.

Comportamiento dinámico. Forma de procesamiento donde el computador recibe y/o procesa la información en forma inmediatamente (sin diferir).

Comportamiento interactivo. Forma de procesamiento en que el usuario dialoga con el interfaz de un computador.

Proceso de test de software. Los medios a través de los cuales se integran las personas, los métodos, las mediciones, las herramientas y los equipos con el objetivo de evaluar la funcionalidad de un producto de software y detectar los errores.

Modelo incremental. Consiste en afrontar las modificaciones del producto central a fin de cumplir mejor las necesidades del cliente y la entrega de funciones, y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

Modelo de desarrollo evolutivo. Consiste en expandir los incrementos de un producto de software operativo, siendo las direcciones de evolución determinadas por la experiencia operativa.

Metodología tradicional. Es una metodología de desarrollo de software que está compuesta por los siguientes pasos: una vez obtenido el modelo de datos, el siguiente es diseñar la base de datos que soporte el modelo de dato. Sin embargo, el modelo de dato no es suficiente para el desarrollo de software sobre base de datos, ya que él mismo describe los datos pero no los comportamientos. Entonces, es necesario recurrir a una tarea adicional llamada análisis funcional, mediante el cual se estudia la organización desde el punto de vista de las funciones existentes. El resultado de dicha tarea es una especificación funcional.

Metodología GeneXus de Gonda. Es una metodología de desarrollo de software mediante aproximaciones sucesivas y el análisis de las visiones de los usuarios que consiste en: primer paso es crear un nuevo proyecto o base de conocimiento. Una vez creada una nueva base de conocimiento, el siguiente paso es describir las visiones de los usuarios. Para ello, se deben identificar los objetos de la realidad y luego pasar a definirlos mediante los objetos GeneXus.

Entorno del sistema. Son los elementos externos del software o aplicación que se vinculan con ella recibiendo o enviando información.

Ciclo del modelo del prototipo. Es el bucle Diseño-Prototipo que se recorre repetidamente, construyendo y probando sucesivas implementaciones.

Modelo de transformación. Asume la existencia de una capacidad de convertir automáticamente una especificación formal de un producto de software en un programa que satisfaga la especificación.

Proceso de software. Una colección de actividades que toman uno o más tipos de entradas y crea una salida que es de valor para el cliente.

Factibilidad de desarrollo de software. Es la posibilidad de implementar los requerimientos del software y existen razones que justifican, condicionan su desarrollo o selección considerando los recursos disponibles y los aspectos políticos.

Estructura de una transacción. Define los atributos que integran la transacción y contiene las relaciones existentes entre ellas.

Problemas de Arquitectura. Incumplimiento de las restricciones que se deben tener en cuenta al diseñar y configurar los componentes de un software.

Proyecto. Conjunto integrado de planes, tareas, actividades y operaciones necesarias para planificar y desarrollar un sistema o una aplicación.

Redefinición. Se refiere a cuando se cambian los procedimientos de una o varias funciones principales ya definidas e implementadas en un software.

Requisitos definidos exhaustivamente. Que se han definido en detalle y sin omisiones, las restricciones que debe cumplir el sistema a desarrollar.

Requisitos definidos formalmente. Que se han definido las restricciones que debe cumplir el sistema a desarrollar en forma expresa en algún documento o informe y con precisión.

Sistema de Información. Conjunto integrado de las personas, procedimientos, medios materiales y otros recursos destinado a la captura, administración, proceso y distribución de información en el ámbito de una organización.

Software de base. Conjunto integrado de programas que se encargan de la gestión de algún servicio básico para el usuario del computador.

Subsistemas. Componentes o partes distinguibles de un sistema, software o aplicación.

Transacción. Es una colección de operaciones que se llevan a cabo como una única función lógica en un software sobre una base de datos.

CAPÍTULO II: PLANTEAMIENTO DEL PROBLEMA

2.1. DETERMINACIÓN DEL PROBLEMA

En el plan de estudios[102] de la especialidad de Informática en la Facultad de Ciencias de la Universidad Nacional de Educación Enrique Guzmán y Valle, se tiene programada la asignatura de *base de datos,* correspondiente al IV Ciclo Académico. La asignatura tiene 04 créditos (teoría: 02 horas y práctica 02 horas). Los estudiantes que se matriculan en esta asignatura, en las horas que corresponden a la teoría, estudian los fundamentos teóricos para el aprendizaje del desarrollo de software sobre base de datos, y en las horas correspondientes a la práctica, el profesor presenta a los estudiantes diferentes situaciones problemáticas de las instituciones educativas y otras áreas de negocios, para que los estudiantes mediante el uso de las metodologías y herramientas informáticas puedan construir software sobre base de datos, que solucionen el problema planteado.

El profesor describe los casos en forma literal y, a partir de estas situaciones problemáticas, los estudiantes deben desarrollar el software sobre base de datos que automatice el problema planteado. En el proceso de la construcción del software, los estudiantes deben hacer uso de los fundamentos teóricos del desarrollo de software, sobre base de datos (metodologías de desarrollo de software), utilizando las diversas herramientas de desarrollo de software existentes en el mercado tecnológico para este objetivo.

MÁRQUEZ (2006)[103] señala que una de las metodologías para el desarrollo de software sobre base de datos es el GeneXus. En esta metodología, es común referirse al analista que trabaja con GeneXus como "Analista GeneXus" en lugar de "Programador GeneXus" (64). La tarea básica del analista GeneXus es la descripción de la realidad y el resultado es parte de lo que se conoce como esquema conceptual.

---

[102] UNIVERSIDAD NACIONAL DE EDUCACIÓN Enrique Guzmán y Valle (2004). Currículo 2004. Resolución Nº 0017-2004-R-UNE. 1ra ed. Lima: CEMED. pág.167

[103] MÁRQUEZ, D, (2006). Opus. cit. pág. 6.

REINGRUBER (1994)[104] define el "esquema conceptual" como "una descripción abstracta y general de la parte o sector del universo real que el contenido de la base de datos va a representar, llamada algunas veces: "Universo del discurso".

CASTAÑO (2000)[105] refiere que el "esquema conceptual describe lo que deseamos almacenar y resulta del análisis de la documentación existente, junto con las entrevistas a los usuarios. Asimismo, SILBERSCHATZ y otros (2002)[106] plantean que el esquema de datos se irá refinando sucesivamente y aplicando las formas normales: la primera forma normal (1FN), segunda forma normal (2FN) y la tercera forma normal (3FN), hasta obtener un esquema de datos óptimo en el modelo Entidad-Relación (E-R)". Actualmente, sólo el ser humano puede desarrollar la tarea de describir la realidad, ya que solamente él puede entender el problema del usuario. Para esta tarea, los integrantes del área de negocios deben intervenir en las reuniones descriptivas en las que expone cada usuario su visión de la unidad de negocio, y éste es uno de los fundamentos en el que se basa la metodología GeneXus.

MÁRQUEZ (2006)[107] expresa que "Para comenzar el desarrollo de software mediante la metodología GeneXus, el primer paso del analista GeneXus consiste en crear una base de conocimiento y el segundo paso es describir las visiones de los usuarios, actividad que se realiza mediante la identificación de los objetos de la realidad y éstos se definen mediante los objetos GeneXus". El mismo MÁRQUEZ (2006)[108] señala que los "objetos GeneXus son las transacciones, reportes, procedimientos, work panels, web panels, data views y otros, los que constituyen elementos básicos para la construcción de aplicaciones sobre base de datos, y sin los objetos GeneXus la metodología no se podría implementar", que son los elementos básicos para la construcción del software sobre base de datos, mediante las aproximaciones sucesivas.

Las metodologías tradicionales del desarrollo de software sobre base de datos difieren entre sí en la forma de cómo abordar el problema, razón por la cual presentan dificultades en la construcción del software y éstas dificultan el aprendizaje de los estudiantes; y necesitamos conocer y experimentar la aplicación de otras metodologías

[104] REINGRUBER, M. y otros (1994). *The Data Modelling Handbook*. Pág. 64.
[105] CASTAÑO, A. y otros (2000). Opus cit., pág. 317.
[106] SILBERSCHATZ, A. y otros (2002). Opus cit., págs. 161-179.
[107] MÁRQUEZ, D. (2006). Opus cit., pág. 43.
[108] MÁRQUEZ, D. (2006). Opus cit., pág. 41.

para el desarrollo de software sobre base de datos, que sean eficientes, de aprendizaje fácil y que permitan reducir el tiempo de mantenimiento del software.

El problema de investigación se enmarca dentro de las interrogantes de la realidad educativa en la universidad. Atiende el caso de la enseñanza del desarrollo de software sobre base de datos mediante el uso de la metodología GeneXus de Gonda[109], comparada con la metodología tradicional. Existen muchas metodologías de desarrollo de software sobre base de datos, pero falta realizar evaluaciones, validaciones y experimentar su uso en la enseñanza de las asignaturas de la especialidad de informática.

Desde hace varios años, el rendimiento académico de los estudiantes de la Facultad de Ciencias amerita ser analizado. Se observa que los estudiantes que ingresan a la especialidad de Informática, no obstante haber obtenido mejores calificaciones en los exámenes de admisión a la Universidad, en relación con los estudiantes que ingresan a las otras especialidades de la UNE, en su mayoría presentan bajo rendimiento académico en las asignaturas de su especialidad, particularmente en la asignatura de base de datos, cuyo objetivo es aprender a desarrollar el software sobre base de datos.

Este problema académico de los estudiantes de la Facultad de Ciencias se muestra en la Tabla N° 2.

Tabla N° 2. Promedio del curso de *Base de datos.*

Periodos Académicos	Especialidad	Secciones	Promedio total del curso	
		C-6	C-8	
2006-B	Informática	11,99	14,90*	13,45
2005-C	Informática	12,64	12,10	12,37
2004-C	Informática	14,21	14,17	14,19
2003-C	Informática	13,32	11,68	12,55
	Promedio aritmético por secciones	13.10	12.91	13.02

Fuente: Oficina Central de Registro y Servicios Académicos (OCRyS) de la UNE.
*: Promedio de nota del semestre 2006-B, sección C-8, en el que se utilizó la metodología Genexus de Gonda en el desarrollo de la asignatura.

[109] Breogán Gonda y Nicolás Jodal son ingenieros de sistemas en Computación, egresados de la Facultad de Ingeniería de la Universidad de la República de Uruguay.Ellos han desarrollado la actividad docente y de consultoría, han recibido el Premio Nacional de Ingeniería 1995, otorgado por la Academia Nacional de Ingeniería de Uruguay por el Proyecto GeneXus; son socios fundadores y directores de ARTech Inc. y GeneXus Consulting en Uruguay. Actualmente el Ing. Breogán Gonda es Presidente de Artech Inc.

Como se puede apreciar en la Tabla N° 2, el promedio máximo alcanzado por los estudiantes en los cuatro periodos académicos de estudio de la asignatura de base de datos es de 14,9 puntos –en la escala vigesimal- en el periodo académico 2006-B y el promedio mínimo alcanzado es de 11,91 puntos que se observa en el periodo 2004-C. El promedio total de los ciclos académicos de las secciones que llevaron la asignatura de base de datos es 13,20 puntos en la escala vigesimal.

De acuerdo con lo observado, algunos elementos que ocasionan el bajo rendimiento en la asignatura de base de datos de los estudiantes de la Facultad de Ciencias, es el uso de metodologías inadecuadas del desarrollo de software sobre base de datos en la conducción de la asignatura por parte del docente. El docente acostumbra limitar el desarrollo de software sobre base de datos sólo al uso de las metodologías tradicionales conocidas por él.

Como resultado de estas deficiencias se tiene que un gran número de estudiantes no asimila los fundamentos básicos de base de datos y no puede aplicar lo aprendido en la construcción de software eficiente y eficaz.

En el semestre académico 2006-B, se hizo una prueba piloto aplicado a un grupo de treinta estudiantes de la sección C-8, en el que podemos observar que el promedio de rendimiento fue de 14,9. Esta sección llevó la asignatura mediante el uso de la metodología GeneXus de Gonda y la sección C-6 de 31 estudiantes obtuvo un promedio 11,90. Esta sección desarrolló la asignatura mediante el uso de la metodología tradicional.

Esta experiencia nos obliga a sospechar que el aprendizaje de los estudiantes de la asignatura de base de datos puede mejorar si el profesor conduce la asignatura mediante el uso de la metodología GeneXus de Gonda, como medio didáctico, en lugar de las metodologías tradicionales.

2.2. FORMULACIÓN DEL PROBLEMA

El problema de investigación es planteado mediante un problema general y tres específicos:

2.2.1. PROBLEMA GENERAL

¿Cuáles son los efectos de la aplicación de la metodología GeneXus de Gonda y la metodología tradicional en el aprendizaje del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle?

2.2.2. PROBLEMAS ESPECÍFICOS

2.2.2.1. ¿Cuáles son los efectos de la aplicación de la metodología GeneXus de Gonda y la metodología tradicional en el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle?

2.2.2.2. ¿Cuáles son los efectos de la aplicación de la metodología GeneXus de Gonda y la metodología tradicional en el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle?

2.2.2.3. ¿Cuáles son los efectos de la aplicación de la metodología GeneXus de Gonda y la metodología tradicional en el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle?

2.3. IMPORTANCIA Y ALCANCES DE LA INVESTIGACIÓN

2.3.1. IMPORTANCIA

El presente trabajo de investigación es importante porque trata de una propuesta de estrategia metodológica que incluye innovaciones metodológicas en el desarrollo de software, que permitirá superar uno de los problemas principales de la enseñanza y aprendizaje del desarrollo de software sobre base de datos. En ese sentido, será un aporte para la educación universitaria en el área de Informática de nuestra Universidad y el país, porque no es sólo una investigación explicativo-experimental sino más bien es una propuesta innovadora que consiste en la elaboración de un módulo de desarrollo de software sobre base de datos, mediante la aplicación de la metodología GeneXus. Se ha optado por esta propuesta toda vez que la enseñanza mediante la metodología tradicional del desarrollo de software sobre base de datos es eminentemente teórica y en la parte práctica siempre se pasa por enseñar un lenguaje de programación y un gestor de base de datos, que el docente conoce y tal vez el mercado laboral requiere profesionales analistas de sistemas, con resultados siempre insatisfactorios.

El material de enseñanza denominado "módulo de desarrollo de software sobre base de datos, mediante la aplicación de la metodología GeneXus" es una innovación metodológica en la construcción de software, el cual constituye una de las alternativas de solución para superar el problema de aprendizaje del desarrollo de software sobre base de datos. En una propuesta metodológica de aprendizaje del tema mencionado, que permite una mejor fijación en el sistema neurológico del estudiante.

La elaboración del módulo didáctico antes mencionado se ha realizado a partir del material proporcionado por la empresa Artech Inc de Uruguay, la que fue adaptada a nuestras necesidades con la participación del autor de esta tesis y las sugerencias del equipo profesional de la empresa antes mencionada.

El módulo didáctico de desarrollo de software sobre base de datos, mediante la aplicación de la metodología GeneXus, tiene doble posibilidad de uso: como estrategia de enseñanza el docente y como estrategia de aprendizaje para el estudiante. A su vez, es un material para enseñanza virtual y presencial de estudiantes interesados en aprender a desarrollar software sobre base de datos, mediante la metodología GeneXus de Gonda.

2.3.2. ALCANCES DE LA INVESTIGACIÓN

Los alcances de esta investigación van más allá de los estudiantes de la población de estudio, puesto que mediante la metodología GeneXus de Gonda, los estudiantes obtienen aprendizajes muy significativos. Además, no es sólo para los estudiantes de la especialidad de Informática de la Universidad, sino para los estudiantes de otras instituciones educativas del país. Asimismo, el módulo de desarrollo de software sobre base de datos será de gran utilidad para las personas autodidactas que deseen aprender a desarrollar software sobre base de datos, ya que el dominio del desarrollo de software sobre base de datos, mediante la metodología propuesta es una oportunidad competitiva de los estudiantes y futuros profesionales del país, y así poder obtener logros profesionales y laborales en el mercado exigente y competitivo de nuestro país.

2.4. LIMITACIONES DE LA INVESTIGACIÓN

Las limitaciones que se presentaron durante el desarrollo del presente trabajo de investigación fueron los siguientes:

- La cantidad de computadoras en los laboratorios de informática no guardaban relación con el número de estudiantes matriculados en la asignatura.
- No se contaban con licencias de software profesionales de la herramienta GeneXus y los estudiantes sólo trabajaron con la versión trial de GeneXus 8.0; la que se halla en la dirección electrónica siguiente: http://www.abab.com.pe.
- Carencia de bibliografía especializada y el escaso recurso de los estudiantes que en la mayoría de ellos no cuentan con computadora en el hogar para poder practicar sus tareas domiciliarias.
- La Facultad de Ciencias cuenta con pocos docentes con conocimiento de metodologías ágiles en el desarrollo de software sobre base de datos.

CAPÍTULO III: DE LA METODOLOGÍA

3.1. PROPUESTA DE OBJETIVOS

El cuadro de objetivos está constituido por un objetivo general y tres específicos, planteados en la forma siguiente:

3.1.1. OBJETIVO GENERAL

Comprobar que la aplicación de la metodología GeneXus de Gonda mejora el aprendizaje del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.1.2. OBJETIVOS ESPECÍFICOS

3.1.2.1. Objetivo específico 1:

Probar que la aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.1.2.2. Objetivo específico 2:

Probar que la aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.1.2.3. Objetivo específico 3:

Probar que la aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.2. SISTEMA DE HIPÓTESIS

El sistema de hipótesis está constituido por una hipótesis principal, tres específicas, que se plantea en la siguiente forma:

3.2.1. HIPÓTESIS GENERAL

La aplicación de la metodología GeneXus de Gonda mejora significativamente el aprendizaje del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.2.2. HIPÓTESIS ESPECÍFICAS

Hipótesis de investigación 1:

La aplicación de la metodología GeneXus de Gonda mejora significativamente el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

Hipótesis de investigación 2:

La aplicación de la metodología GeneXus de Gonda mejora significativamente el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

Hipótesis de investigación 3:

La aplicación de la metodología GeneXus de Gonda mejora significativamente el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.3. SISTEMA DE VARIABLES

Las variables consideradas en la investigación son:

3.3.1. INDEPENDIENTES:

X_1: Metodología GeneXus de Gonda.
X_2: Metodología tradicional.

3.3.2. DEPENDIENTE:

Y: Aprendizaje del desarrollo de software sobre base de datos.

3.3.3. DIMENSIONES E INDICADORES

Tabla Nº 3. Dimensión e indicadores de la metodología GeneXus de Gonda (elaborado por el autor).

VARIABLE	DIMENSIONES	INDICADORES
Metodología GeneXus de Gonda	Análisis de la realidad	Describe el área de negocios.
	Definir los objetos GeneXus	Identifica las transacciones.
		Identifica los atributos y reglas del negocio.
		Realiza consultas de tablas.
	Creación de prototipos	Construye la base de datos.
		Genera los programas.
	Mantenimiento del software	Modifica la base de datos y programas.

Tabla Nº 4. Dimensión e indicadores de la metodología tradicional (elaborado por el autor).

	DIMENSIONES	INDICADORES
Metodología Tradicional	Análisis de datos	Describe el área de negocios.
	Modelo de datos	Identifica las entidades.
		Identifica los atributos y claves.
		Define las relaciones entre entidades.
	Definir la base de datos	Construye la base de datos.
	Análisis funcional	Presenta la especificación funcional.
	Codificación	Codifica los programas.
	Integración	Integra los programas.
	Prueba del sistema integral	Prueba del programa final.
	Mantenimiento del software	Modifica la base de datos y programas.

Tabla Nº 5. Dimensión, categoría e indicadores de la variable dependiente: Aprendizaje del desarrollo de software sobre base de datos.

VARIABLE	DIMENSIÓN	CATEGORÍA	INDICADORES	ÍTEMS	ESCALA
Aprendizaje del desarrollo de software sobre base de datos	Aprendizaje de conocimientos conceptuales.	Conoce los fundamentos del desarrollo de software sobre base de datos.	1. Ciclo de vida del desarrollo de software. 2. Metodologías de desarrollo de software. 3. Modelos de desarrollo de software. 4. Estructuras de datos. 5. Estructuras de transacciones. 6. Análisis de impacto sobre una base de datos. 7. Entidades y atributos. 8. Tipos de relaciones o cardinalidad. 9. Funciones de la base de datos. 10. Herramientas de desarrollo de software sobre base de datos (lenguajes de programación y gestores de bases de datos).	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	2:Respuesta Correcta 0:Respuesta incorrecta
	Aprendizaje de conocimientos procedimentales	Reconoce los procesos del desarrollo del software	1. Tiempo requerido para el desarrollo del software. 2. Trabajo en equipo. 3. Proceso de software. 4. Participación del usuario. 5. Migración de software entorno Windows y Web. 6. Consultas a la base de datos. 7. Diseño de base de datos (estable). 8. Conocimiento de lenguajes de programación y gestores de bases de datos. 9. Normalización de tablas. 10. Mantenimiento de bases de datos y programas (construcción de prototipos).	11, 12, 13, 14, 15, 16, 17, 18, 19, 20	5: Siempre 4: Casi siempre 3. A veces 2: Casi nunca 1: Nunca
	Aprendizaje de Conocimientos actitudinales	Presenta actitud positiva durante el desarrollo del software	1. Desarrollo de la asignatura. 2. Metodología de desarrollo de software. 3. Trabajo en equipo. 4. Habilidades y destrezas. 5. Lenguajes de programación y gestores de bases de datos. 6. Autoaprendizaje. 7. Hábitos de trabajo. 8. Interrelación con situaciones problemáticas de la vida real. 9. interés en el desarrollo de software. 10. innovación de desarrollo de software sobre la base de datos.	21, 22, 23, 24, 25, 26, 27, 28, 29, 30	2: Muy bueno 1: Bueno 0. Ni bueno, ni malo -1: Malo -2: Muy malo

Como muestra la Tabla N° 5, la dimensión de la variable dependiente es estudiada desde la perspectiva del aprendizaje de los conocimientos conceptuales, aprendizaje de los conocimientos procedimentales y aprendizaje de los conocimientos actitudinales. Para su estudio se distribuye los ítems de cada instrumento tal como se presenta en la Tabla N° 6.

Tabla N° 6. Distribución de ítems por categorías y dimensiones.

CONOCIMIENTO CONCEPTUAL	ÍTEMS
Datos	4, 5
Hechos	6, 7, 9
Conceptos	1, 3
Principios o leyes	2, 8, 10
CONOCIMIENTO PROCEDIMENTAL	ÍTEMS
Generales	2, 7
Algoritmos	4, 8
Heurísticos	3
Destrezas y habilidades	5
Técnicas	6, 10
Estrategias	1
Motriz cognitivo	9
CONOCIMIENTO ACTITUDINAL	ÍTEMS
Valores	3, 4, 7
Normas	2,
Actitudes	6, 8, 9
Juicios valorativos	1, 5,10

Variables Intervinientes:

- Edad y sexo del estudiante.
- Número de horas de trabajo con el computador.
- Software que utiliza para el estudio.
- Software que utiliza para el trabajo.
- Estilo de enseñanza.
- Condiciones de los laboratorios de enseñanza.
- Conocimiento de los docentes.

3.4. TIPO Y MÉTODOS DE INVESTIGACIÓN

3.4.1. TIPO DE INVESTIGACIÓN

El presente trabajo es una investigación cuantitativa se trata de una investigación de tipo explicativo-experimental.

La investigación planteada también puede concebirse como tipo de tecnologías sociales, tal como plantea PISCOYA (1995)[110], con dos grupos: uno de control y otro experimental; medidos antes y después, en el proceso del desarrollo de la asignatura de *Base de datos*.

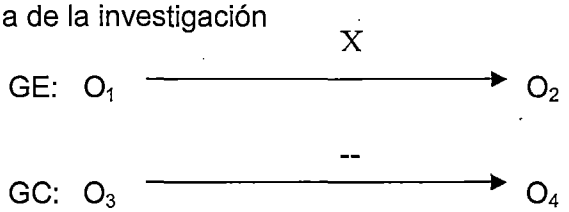
3.4.2. MÉTODO DE INVESTIGACIÓN

La investigación es de carácter cuasi-experimental, ya que se mide el aprendizaje del desarrollo de software sobre base de datos como resultado de aplicar la metodología GeneXus de Gonda y la metodología tradicional en los estudiantes de la especialidad de Informática del IV ciclo de la Universidad Nacional de Educación Enrique Guzmán y Valle. El experimento tuvo una duración de un semestre académico y para darle uniformidad se utilizó el lenguaje de programación Microsoft Visual Basic y el gestor de base de datos Microsoft Access.

3.5. DISEÑO DE LA INVESTIGACIÓN

En la presente investigación se trabajó con el diseño cuasi-experimental; para ello se tiene un grupo de control (GC) y otro grupo experimental (GE).

Esquema de la investigación



Donde:

GE: Grupo experimental
GC: Grupo de control

O_1, O_3: Pretest

O_2, O_4: Postest

X : Aplicación de la metodología GeneXus de Gonda.

-- : Aplicación de la metodología tradicional.

[110] PISCOYA, L. (1995). *Investigación Científica y Educacional.* 2da ed. Lima, Perú: Amauta Editores, pág.86.

A los grupos experimental y control se aplicó el pretest (O_1 y O_3) y el postest (O_2 y O_4) y luego se realizó:

- Un estudio estadístico comparativo de los puntajes obtenidos de los grupos relacionados O_1 con O_2 y O_3 con O_4.
- Un estudio estadístico comparativo de los puntajes obtenidos de los grupos independientes O_1 con O_3 y O_2 con O_4

El diseño de la investigación es cuasi-experimental, con dos grupos: uno de control y otro experimental. Los grupos se constituyeron en forma aleatoria de diecinueve estudiantes cada uno. Al grupo experimental se enseñó el desarrollo de software sobre base de datos, utilizando como medio didáctico la metodología GeneXus de Gonda, y al grupo control utilizando la metodología tradicional (ver la Tabla N° 7).

Tabla N° 7. Grupos según metodología.

	Metodología GeneXus de Gonda	Metodología tradicional
Grupo control		X
Grupo experimental	X	

La medición se llevó a cabo de la forma siguiente:

1. Al total de los estudiantes del grupo experimental y control, se aplicó al inicio del curso el pretest (prueba de entrada), que mide el aprendizaje de conocimientos conceptuales (diez preguntas) (ver Anexo N° 3), conocimientos procedimentales (diez preguntas) (ver Anexo N° 3) y conocimientos actitudinales (diez preguntas) (ver Anexo N° 3). Los estudiantes tuvieron quince minutos para contestar cada test.

2. Los grupos experimental y control se constituyeron mediante una "elección aleatoria simple sin reposición", y los estudiantes recibieron sus clases con profesores distintos y en distintos laboratorios[111]. El procedimiento para constituir los grupos experimental y control fueron: 1) elaborar una balota con 19 letras A y otras 19 balotas con letras B, 2) los 38 estudiantes hacen una cola en el pabellón de laboratorio de Ciencias, 3) los estudiantes van sacando al azar una balota para ubicarse en el salón donde recibirá su clase (Aulas C-6: Aula de grupo experimental y C-8: aula de grupo control).

[111] En la Facultad de Ciencias de la Universidad Nacional de Educación, para la enseñanza de esta asignatura, se cuenta con laboratorios con tecnologías apropiadas y en cada una de ellas están instaladas las licencias de GeneXus Trial Version 8.0.

3. Al finalizar el semestre académico a los estudiantes del grupo de control y grupo experimental se les aplicó el postest (prueba de salida).

4. El postest está constituido por ítems de conocimientos: conceptuales (ver Anexo Nº 4), procedimentales (ver Anexo Nº 4) y actitudinales (ver Anexo Nº 4). Los estudiantes tuvieron quince minutos para contestar cada test.

3.6. POBLACIÓN Y MUESTRA

3.6.1. POBLACIÓN

Todos los estudiantes matriculados en el IV Ciclo Académico 2006-II de la especialidad de Informática de la Facultad de Ciencias de la Universidad Nacional de Educación Enrique Guzmán y Valle.

3.6.2. MUESTRA

La muestra está constituida por todos los estudiantes de la especialidad de Informática, que se encuentran matriculados en la asignatura de *Base de datos* y el número de estudiantes matriculados en la asignatura son 38 (Ver Tabla Nº 8).

Tabla Nº 8. Muestra de estudio.

	Especialidad	Promoción	Secciones	Número de alumnos matriculados
Grupo experimental	Informática	2006	C-6	19
Grupo control	Informática	2006	C-8	19
			Total:	38

SEGUNDA PARTE: TRABAJO DE CAMPO

CAPÍTULO IV: INSTRUMENTOS DE INVESTIGACIÓN Y RESULTADOS

4.1. SELECCIÓN Y VALIDACIÓN DE INSTRUMENTOS

4.1.1. LOS INSTRUMENTOS DE INVESTIGACIÓN

Los ítems del test de conocimientos conceptuales fueron de múltiple opción de cuatro alternativas en que sólo una es verdadera; los ítems del test de conocimientos procedimentales fueron sometidos a opinión del "grado de acuerdo" mediante la escala de likert (1:Nunca, 2:Casi nunca, 3:A veces, 4:Casi siempre y 5:Siempre) y los ítems del test de conocimientos actitudinales fueron sometidos a opinión del "grado de aceptación" mediante la escala de likert (-2:Muy malo, -1:Malo, 0:Ni bueno, ni malo, 1:Bueno, 2:Muy bueno).

Los instrumentos utilizados fueron:

- Pretest y postest de conocimientos conceptuales (mide el aprendizaje de los fundamentos del desarrollo de software sobre base de datos) (ver Anexo 2-A y 2-B).
- Pretest y postest de conocimientos procedimentales (mide el aprendizaje de los procedimientos para el desarrollo de software sobre base de datos) (ver Anexo 2-A y 2-B).
- Pretest y postest de conocimientos actitudinales (mide el aprendizaje de los valores en el desarrollo del software sobre base de datos) (ver Anexo 2-A y 2-B).

4.1.1.1. PRETEST Y POSTEST DE CONOCIMIENTOS CONCEPTUALES

El test que evalúa los conocimientos conceptuales del "aprendizaje del desarrollo de software sobre base de datos" está constituido por un total de diez preguntas. Se formuló una pregunta por cada ítem. Los ítems miden el aprendizaje de los fundamentos teóricos del desarrollo de software sobre base de datos, que consiste en: ciclo de vida del desarrollo de software, metodologías de desarrollo de software, modelos de desarrollo de software, estructuras de datos, estructuras de transacciones, análisis de impacto sobre

una base de datos, entidades y atributos, tipos de relaciones, funciones de la base de datos y herramientas de desarrollo de software sobre base de datos (lenguajes de programación y gestores de bases de datos) (ver Anexos Nº 3 y Nº 4).

4.1.1.2. PRETEST Y POSTEST DE CONOCIMIENTOS PROCEDIMENTALES

El test que evalúa los conocimientos procedimentales del "aprendizaje del desarrollo de software sobre base de datos" está constituido por un total de diez preguntas. Se formuló una pregunta por cada ítem. Los ítems miden el aprendizaje de los procedimientos del desarrollo de software sobre base de datos, que consiste en: Tiempo requerido para el desarrollo del software, trabajo en equipo, proceso de software, participación del usuario, migración de software en torno Windows y Web, consultas a la base de datos, diseño de base de datos, lenguajes de programación y gestores de base de datos, normalización de tablas, mantenimiento de bases de datos y programas (construcción prototipos) (ver Anexos Nº 3 y Nº 4).

4.1.1.3. PRETEST Y POSTEST DE CONOCIMIENTOS ACTITUDINALES

El test que evalúa los conocimientos actitudinales del "aprendizaje del desarrollo de software sobre base de datos" está constituido por un total de diez preguntas. Se formuló una pregunta por cada ítem. Los ítems miden el aprendizaje de las actitudes en el desarrollo de software sobre base de datos, que considera la apreciación de: desarrollo de la asignatura, metodología de desarrollo de software, trabajo en equipo, desarrollo de habilidades y destrezas, conocimiento de los lenguajes de programación y gestores de base de datos, autoaprendizaje, hábitos de trabajo, interrelación con situaciones problemáticas de la vida real, motivación e interés en el desarrollo de software e innovación de desarrollo de software sobre la base de datos (ver Anexos Nº 3 y Nº 4).

4.2. VALIDACIÓN Y CONFIABILIDAD DEL INSTRUMENTO DE INVESTIGACIÓN.

Los instrumentos que miden específicamente el aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales del desarrollo de software sobre base de datos, fueron sometidos a la validación de contenidos a través del juicio de expertos. Para tal efecto, se utilizó el formato para la evaluación de los ítems (Anexo Nº 1: Instrumento de validación de expertos). La primera evaluación sugirió reajustar algunos ítems del Test de conocimientos conceptuales, y posteriormente se sometió a una segunda evaluación de los mismos expertos y se obtuvo una validez de 95,28% en el Test de conocimientos conceptuales, 95,94% en el Test de Conocimientos procedimentales y de 95,42% en el

de conocimientos actitudinales (ver Anexo N° 7). Los expertos a los que se recurrió para la validación de contenidos fueron los profesores: Dr. Raúl LOAYZA JAQUI (docente activo de la Universidad Ricardo Palma), Mg. José Fernando DAVELOUIS MENDÍA (docente activo de la Universidad Nacional Mayor de San Marcos), Mg. Jimmy ROSALES HUAMÁN (docente activo de Universidad Nacional Mayor de San Marcos), Mg. Bertila GARCÍA DÍAZ (docente activo de Universidad Nacional del Callao) y Mg. Hugo Froilán VEGA HUERTA (docente activo de Universidad Nacional Mayor de San Marcos). Todos ellos son ingenieros de Sistemas y/o Informática, y poseen amplia experiencia en la enseñanza del desarrollo de software sobre base de datos, desarrollo de sistemas corporativos e investigadores en el campo de la informática y afines.

Los test que miden el aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales, fueron validados mediante una prueba piloto que se administró a quince estudiantes del ciclo anterior que llevaron la asignatura de base de datos. El cálculo del coeficiente de Alfa de Cronbach del test de aprendizaje del desarrollo de software se muestra en la Tabla N° 9.

Tabla N° 9. Resultados de la prueba de confiabilidad y validación del instrumento, mediante el coeficiente de Alfa de Cronbach.

TEST DE :	ALFA DE CRONBACH
Aprendizaje de conocimientos conceptuales	0,782
Aprendizaje de conocimientos procedimentales	0,678
Aprendizaje de conocimientos actitudinales	0,637

4.3. TRATAMIENTO ESTADÍSTICO

Las técnicas estadísticas empleadas en la investigación son las siguientes:

4.3.1. ESTADÍGRAFO "U" DE MANN WHITNEY

Según GUILFORD y otros (1984)[112], la prueba estadística U de Mann Whitney es una técnica de inferencia que se utiliza en el estudio del comportamiento de grupos independientes. En este grupo, los sujetos se eligen al azar entre la población y luego se dividen de manera aleatoria en dos o más grupos. Todos los sujetos participan en los grupos de control o experimental. Al analizar los datos se establece una comparación entre los datos de cada grupo para determinar si el azar es una explicación razonable de las diferencias entre los puntajes de los grupos.

La prueba de U de Mann Whitney analiza el *grado de separación* entre los dos conjuntos de datos provenientes de las muestras. Este *grado de separación* se calcula mediante las siguientes fórmulas:

$$U_{obt} = n_1 n_{2+} \frac{n_1(n_1+1)}{2} - R_1$$

$$U'_{obt} = n_1 n_{2+} \frac{n_2(n_2+1)}{2} - R_2$$

donde n_1 es el número de datos en el grupo 1

n_2 es el número de datos en el grupo 2

R_1 es la suma de los rangos para los datos del grupo 1

R_2 es la suma de los rangos para los datos del grupo 2

Para un nivel de significancia o confianza de α_{2colas} = 0,01, y la que nos permite realizar la prueba de las hipótesis:

H_0: la hipótesis nula indica que la variable independiente no influye sobre la variable dependiente.

H_1: la hipótesis alterna indica que la variable independiente sí influye sobre la variable dependiente.

Según PAGANO (1999)[113], al utilizar la prueba de U de Mann-Whitney para evaluar H_0, se debe determinar lo siguiente:

---

[112] GUILFORD J. P. Y FRUCHTER, Benjamín (1984). *Estadística aplicada a la psicología y la educación.* Colombia. McGraw-Hill. 1ra ed. págs.191-199.

[113] PAGANO, R. (1999*). Estadísticas para las ciencias del comportamiento.* 5ta ed. México. Internacional Thomsan Editores. pág. 249.

- El grado de separación entre los datos del grupo control y el grupo experimental Uobt.
- La probabilidad de obtener un valor U igual o menor que Uobt, suponiendo que los datos provenientes de las muestras son muestras aleatorias de poblaciones con distribuciones idénticas.

4.3.2. PRUEBA NO PARAMÉTRICA DE RANGOS SEÑALADOS Y PARES IGUALADOS DE WILCOXON

Según PAGANO (1999)[114], la prueba de rangos señalados y pares igualados de Wilcoxon toma en cuenta la magnitud y la dirección de los datos de la diferencia, y debe satisfacer dos supuestos. El primero es que los datos dentro de cada pareja deben alcanzar al menos la escala ordinal. El segundo consiste en que los puntajes de diferencia también deben alcanzar al menos una escala ordinal.

Para el proceso del cálculo del parámetro W de Wilcoxon, se utiliza la Tabla Nº 10.

Tabla Nº 10. Cálculo del parámetro de W de Wilcoxon.

Sujeto (N)	antes	después	diferencia	Rango de la diferencia en valor absoluto|	Rangos positivos	Rangos negativos
1	A1	D1	D1-A1			
2	A1	D2	D2-A1			
...	...		...			
N	AN	DN	DN-AN			
				Suma de los rangos	Suma de los rangos positivos	Suma de los rangos negativos

W_{obt} es igual a la suma de los rangos negativos, la que debe ser comparada con los valores críticos de T (W_{crit}) para la prueba de rangos con signo de Wilcoxon (W_{crit}). Luego, se debe comparar si $W_{obt} < W_{crit}$, para un nivel de significancia de dos colas: $\alpha_{2colas} = 0,01$.

También, para el nivel de significancia $\alpha_{2colas} = 0,01$, se calcula el parámetro "W" de Wilcoxon, mediante:

$$W = \frac{T - \frac{N(N*1)}{4}}{\sqrt{\frac{N(N+1)(2N+1)}{24}}}$$

[114] PAGANO, R. (1999). Opus. cit. pág. 441.

Donde: N es el tamaño de la muestra y T es la suma de los rangos de signo menos frecuente.

4.3.3. PRUEBA NO PARAMÉTRICA DE DÓCIMA DE KOLMOGOROV-SMIRNOV PARA DOS GRUPOS.

Según GARCÍA (2002)[115], la prueba de Kolmogorov-Smirnov es una prueba que nos permite advertir si los datos obtenidos de dos grupos objeto de estudio se asemejan al comportamiento de la distribución Normal o los datos difieren de la curva Normal. La única premisa que se necesita es que las mediciones se encuentren al menos en una escala ordinal.

Características de la dócima de Kolmogorov-Smirnov:

Está construida, teniendo como base detectar las discrepancias existentes entre las frecuencias relativas acumuladas de las dos muestras objeto de estudio. Lo anterior propicia que esta dócima pueda advertir diferencias no tan solo entre las medias, sino que éstas sean debidas a la dispersión, o la simetría o la oblicuidad. Esta característica la hace distintiva de aquellas en que solamente se ocupan de analizar las diferencias entre las medias, también la dócima admite que los tamaños de las muestras no sean iguales.

Para el estadígrafo y la distribución muestral se designa por T_1 y T_2 , las tablas de distribución de frecuencias relativas acumuladas, particionadas en K categorías. Se analiza entonces en la columna de las diferencias de las frecuencias, en qué clase se obtiene el valor máximo. Se calcula mediante:

$D = \max|P1t - P2t|$; con t = 0,1,2,...,K

El estadígrafo de esta dócima se designa por χ^2 , y está distribuido según Chi-cuadrado con dos grados de libertad y se define mediante:

$$\chi^2 = \frac{4D^2(n_1 * n_2)}{n_1 + n_2}$$

donde n_1 es el número de datos en el grupo 1

n_2 es el número de datos en el grupo 2.

[115] GARCÍA, C. (2002). Métodos estadísticos en la evaluación educacional. Lima: CONCYTEC-OFOPCYTE. pág. 77.

4.4. RESULTADOS Y CONTRASTACIÓN DE HIPÓTESIS

En esta sección se desarrolla los procedimientos pertinentes, orientados a demostrar que si la aplicación de la metodología GeneXus de Gonda mejora el aprendizaje del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle. Asimismo, se evalúa el aprendizaje mencionado desde la perspectiva del aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales.

4.4.1. PROCEDIMIENTO DEL USO DE LAS METODOLOGÍAS

Se procedió a la aplicación de las dos metodologías del desarrollo de software sobre base de datos, mediante el módulo: Metodología GeneXus de Gonda (ver Anexo N° 5). La metodología GeneXus de Gonda y la tradicional, en el desarrollo de la asignatura base de datos. En la mencionada asignatura, se matricularon 38 estudiantes de la especialidad de Informática, los cuales fueron divididos en dos grupos: diecinueve estudiantes estudiaron el desarrollo de software sobre base de datos utilizando la metodología GeneXus de Gonda, y el otro grupo constituido por dicinueve estudiantes estudiaron utilizando la metodología tradicional.

La asignatura se desarrolló en doce semanas, tanto para el grupo control como para el experimental, cada uno de ellos conducido por docentes distintos. Cada semana tuvieron una sesión de clase de una duración de cinco horas académicas de cincuenta minutos. Al inicio y final de la asignatura se aplicó los instrumentos: pretest y postest, respectivamente para medir el aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales de ambos grupos.

Los resultados de la evaluación del grupo experimental y control del aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales se presentan en los Anexos No.2-A y 2-B.

En la Tabla del Anexo N° 2-A, se observa que el grupo experimental en el pretest que mide el aprendizaje de los conocimientos conceptuales ha obtenido un puntaje promedio de 5,05 y en la Tabla del Anexo N° 2-B se observa que el grupo control obtiene un puntaje de 5,16. Esto quiere decir que ambos grupos poseen un conocimiento conceptual equivalente u homogéneo de los fundamentos teóricos del desarrollo de aplicaciones sobre base de datos. Pero, si comparamos los resultados en las Tablas N°

2-A y 2-B del postest del aprendizaje de los conocimientos conceptuales, éstos nos muestran que el grupo experimental tiene un promedio de 13,89 puntos y el grupo control un promedio de 9,89 puntos. Esto quiere decir que el grupo experimental tiene una diferencia por encima de cuatro puntos sobre el grupo control.

En la Tabla del Anexo N° 2-A se observa que el grupo experimental en el pretest que evalúa el aprendizaje de los conocimientos procedimentales ha obtenido un puntaje promedio de 1,89 y en la Tabla del Anexo N° 2-B se observa que el grupo control obtiene un puntaje de 1,79. Esto quiere decir que ambos grupos poseen un conocimiento procedimental equivalente u homogéneo de los procedimientos del desarrollo de aplicaciones sobre base de datos. Pero, si comparamos los resultados en las Tablas del Anexo N° 2-A y 2-B del postest del aprendizaje de los conocimientos procedimentales, éstos nos muestran que el grupo experimental tiene un promedio de 34,00 puntos y el grupo control un promedio de 27,37 puntos; esto quiere decir que el grupo experimental presenta una diferencia por encima de 6,63 puntos sobre el grupo control.

En la Tabla del Anexo N° 2-A se observa que el grupo experimental en el pretest que evalúa el aprendizaje de los conocimientos actitudinales ha obtenido un puntaje promedio de -1,00 y en la Tabla del Anexo N° 2-B se observa que el grupo control obtiene un puntaje de -0,68. Esto quiere decir que ambos grupos no presentan variación significante en los conocimientos actitudinales del desarrollo de aplicaciones sobre base de datos. Pero, si comparamos los resultados en las Tablas del Anexo N° 2-A y 2-B del postest del aprendizaje de los conocimientos actitudinales, éstos nos muestran que el grupo experimental tiene un promedio de 10,74 puntos y el grupo control un promedio de 2,42 puntos; esto quiere decir que el grupo experimental presenta una diferencia por encima de 8,32 puntos sobre el grupo control.

4.4.2. APRENDIZAJE DE LOS CONOCIMIENTOS CONCEPTUALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS.

Hipótesis específica 1:

La metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

Comprobación de la hipótesis estadística 1:

Nula (H_o):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos son iguales.

$$\mu_{pretest} = \mu_{postest}.$$

Alterna (H_a):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos no son iguales.

$$\mu_{pretest} ? \mu_{postest}.$$

Utilizando los estadígrafos Kolmogorov-Smirnov para la prueba de Normalidad de datos del pretest y postest en el grupo control y experimental, W de Wilcoxon para la prueba rangos señalados y pares igualados, teniendo en cuenta la dirección y magnitud de la diferencia de datos del pretest y postest en el grupo control y experimental, y U de Mann Whitney para el estudio de comportamiento de grupos independientes, comparado entre el grupo experimental y control, para lo cual se ha utilizado el software SPSS v15.0.

Diseño de la prueba de hipótesis 1

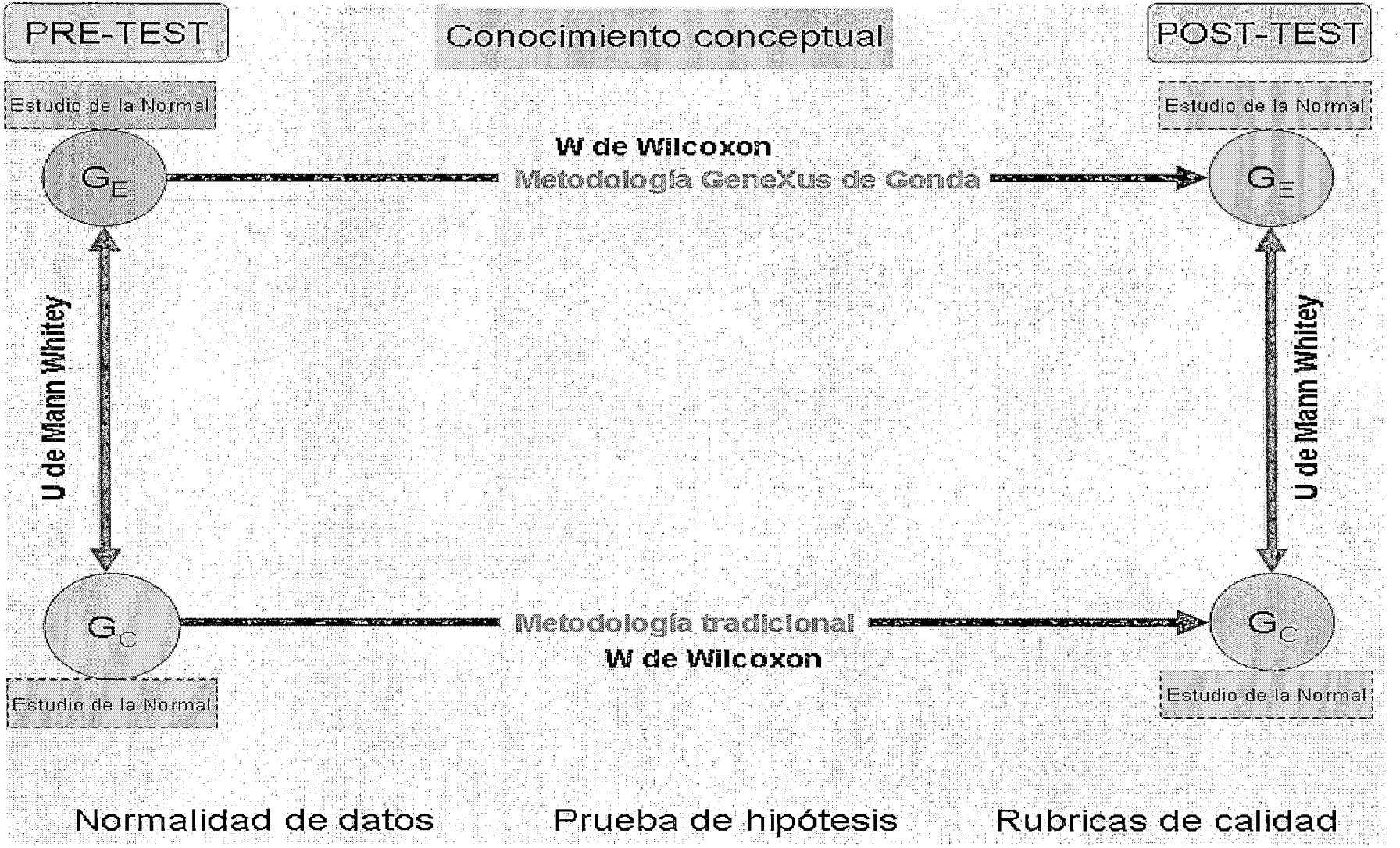


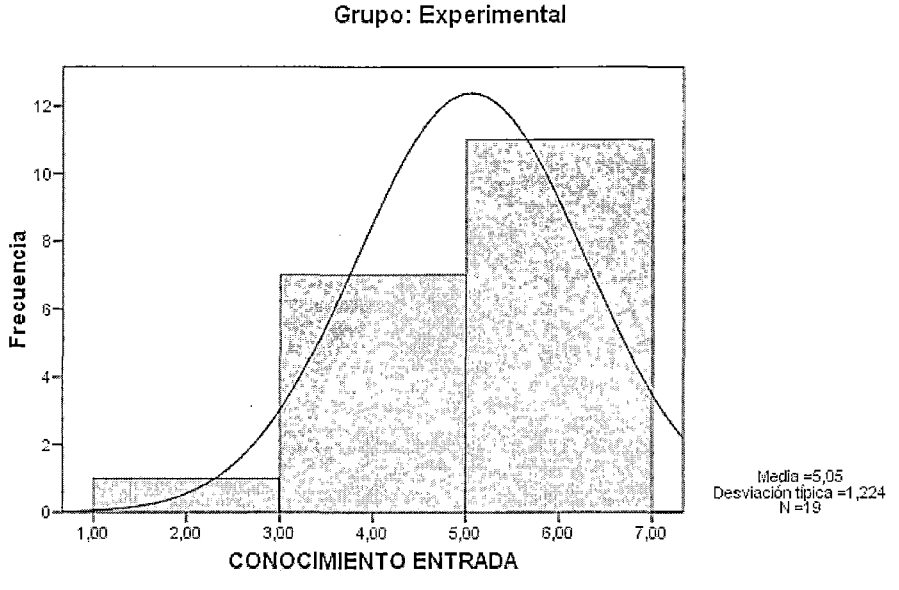
Tabla N° 11. Prueba de normalidad de la distribución de puntajes de conocimiento conceptual en el grupo experimental.

	PRETEST	POSTEST
N	19	19
Z de Kolmogorov-Smirnov	1,567	1,362
Nivel de significancia bilátera	0,015	0,049

p<0,05

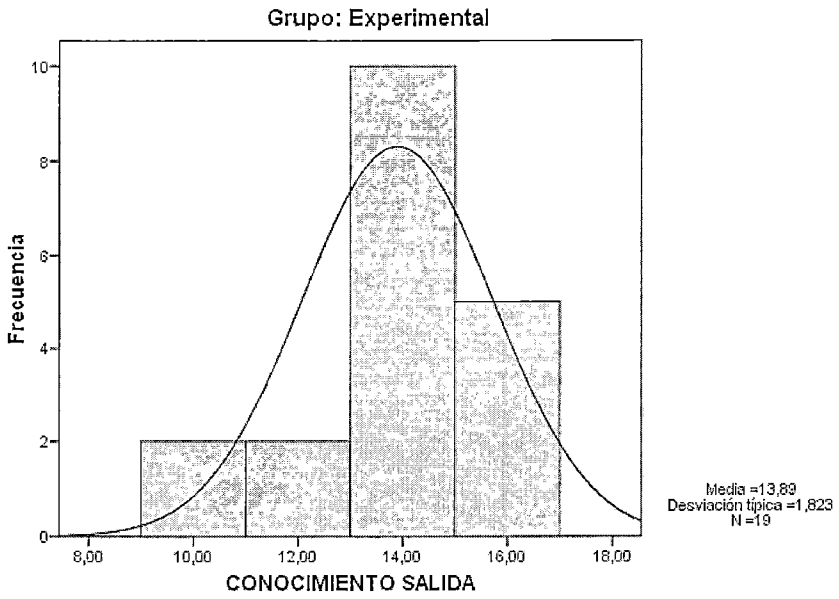
En la Tabla N° 11 puede observarse que el nivel de significancia para la Z de Kolmogorov-Smirnov es menor que 0,05 tanto en los puntajes del pretest y postest, por lo que se deduce que la distribución de estos puntajes en el grupo experimental difieren de la distribución normal.

Gráfico N° 13. Distribución de frecuencias de los puntajes en el pretest de conocimiento conceptual del grupo experimental.



En el Gráfico N° 13 puede observarse la distribución de frecuencias de los puntajes de conocimiento conceptual del pretest del grupo experimental. En el mencionado gráfico los puntajes se hallan sesgados hacia la derecha, teniendo una media de 5,05 y una desviación estándar de 1,22. Asimismo, el gráfico muestra que la curva de distribución difiere notoriamente de la curva normal.

Gráfico N° 14. Distribución de frecuencias de los puntajes en el postest del aprendizaje del conocimiento conceptual del grupo experimental.



En el Gráfico N° 14, puede observarse la distribución de frecuencias de los puntajes de conocimiento conceptual en el postest del grupo experimental. Dicho gráfico muestra que los puntajes se hallan sesgados hacia la derecha, teniendo una media de 13,89 y una desviación estándar de 1,82. Como puede observarse en el gráfico, la curva de distribución difiere notablemente de la curva normal.

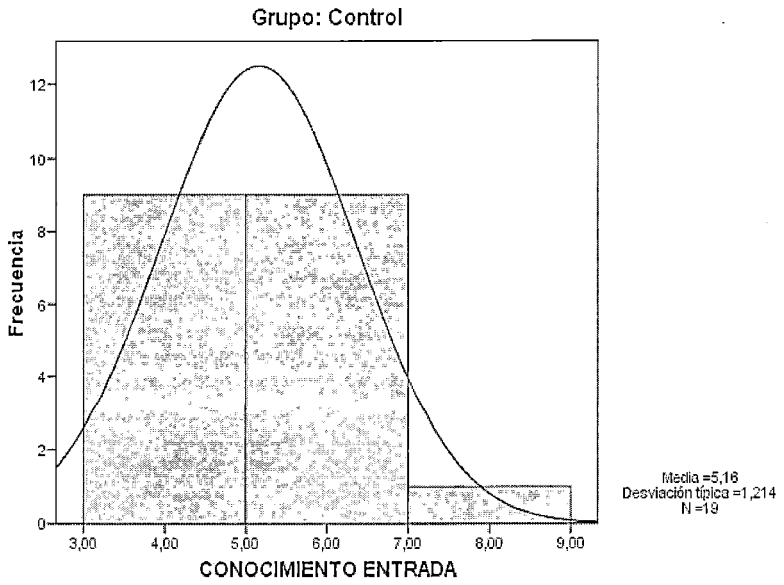
Tabla N° 12. Prueba de normalidad de la distribución de puntajes del pretest de conocimiento conceptual en el grupo control.

	PRETEST	POSTEST
N	19	19
Z deKolmogorov-Smirnov	1,323	1,096
Nivel de significancia (2 colas)	0,060	0,181

p>0,05

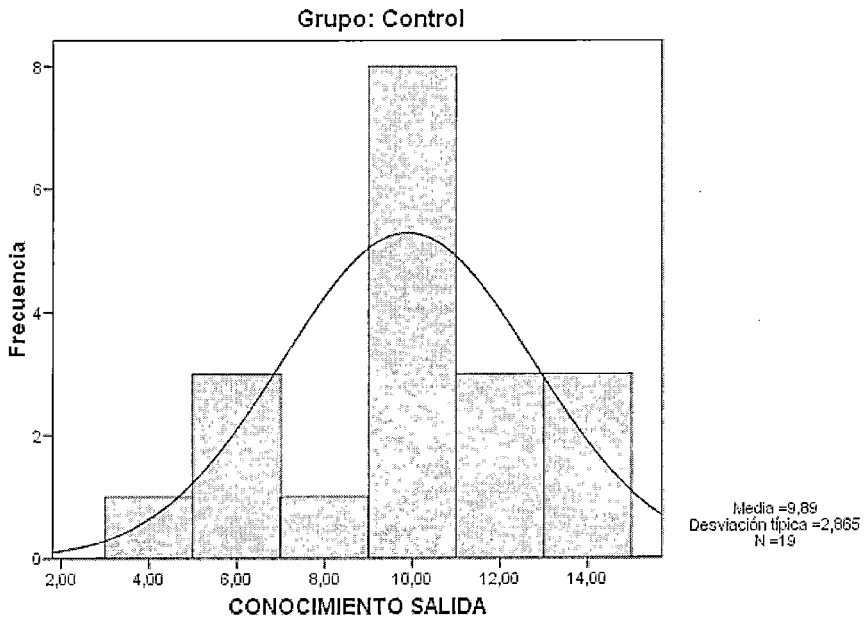
En la tabla N° 12, puede observarse que el nivel de significancia para la Z de Kolmogorov-Smirnov es mayor que 0,05 tanto en los puntajes del pretest como del postest, por lo que se deduce que la distribución de los puntajes del pretest y postest en el grupo control no difieren de la distribución normal.

Gráfico Nº 15. Distribución de frecuencias de los puntajes del pretest del aprendizaje del conocimiento conceptual del grupo control.



En el Gráfico Nº 15, puede observarse la distribución de frecuencias de los puntajes del conocimiento conceptual en el pretest del grupo control. En el mencionado gráfico, los puntajes se hallan sesgados hacia la derecha, teniendo una media de 5,16 y una desviación estándar de 1,214. Asimismo, el gráfico muestra que la curva de distribución difiere notoriamente de la curva normal.

Gráfico Nº 16. Distribución de frecuencias de los puntajes en el postest del conocimiento conceptual del grupo control.



En el Gráfico N° 16, puede observarse la distribución de frecuencias de los puntajes de conocimiento conceptual en el postest del grupo control. El gráfico muestra que los puntajes se hallan sesgados hacia la derecha, teniendo una media de 9,89 y una desviación estándar de 2,87. Como se observa en el gráfico, la curva de distribución difiere de la curva normal.

Tabla N° 13. Tipo de curva de distribución de los puntajes del conocimiento conceptual en los grupos experimental y control.

GRUPO	PRETEST	POSTEST
EXPERIMENTAL	DIFIERE DE LA DISTRIBUCIÓN NORMAL (p = 0,015)	DIFIERE DE LA DISTRIBUCIÓN NORMAL (p = 0,049)
CONTROL	NO DIFIERE DE LA DISTRIBUCIÓN NORMAL (p = 0,060)	NO DIFIERE DE LA DISTRIBUCIÓN NORMAL (p = 0,181)

Los resultados mostrados en la Tabla N° 13 nos permiten utilizar el estadístico U de Mann-Whitney para el análisis de los puntajes entre los grupos control y experimental (grupos no relacionados). Asimismo, se utiliza el estadístico W de Wilcoxon para analizar la diferencia de puntajes entre los grupos control y experimental (grupos relacionados) en pretest y postest.

Tabla N° 14. Prueba de diferencia de puntajes en el pretest y postest entre los grupos control y experimental en el área de conocimiento conceptual.

	Grupo experimental vs grupo control	Grupo experimental vs grupo control
	PRETEST	POSTEST
U de Mann-Whitney	180,000	44,000
Z	-0,017	-4,116
Nivel de significación bilateral	0,987	0,000**

** p<0,01

El estadístico U de Mann-Whitney que se presenta en la Tabla N° 13 muestra que no existen diferencias estadísticamente significativas entre los puntajes de pretest al comparar los grupos control y experimental (p>0,01); mientras que al comparar los puntajes de la postest entre los grupos control y experimental, existen diferencias estadísticamente muy significativas (p<0,01).

Tabla Nº 15. Resumen de las diferencias estadísticas significativas en el pretest y postest de conocimiento conceptual en los grupos experimental y control.

GRUPO	PRETEST	POSTEST
EXPERIMENTAL	No existen diferencias estadísticamente significativas (p > 0,01)	Existen diferencias estadísticamente muy significativas (p < 0,01)
CONTROL		

Tabla Nº 16. Prueba de la diferencia de puntajes de conocimiento conceptual en el pretest y postest del grupo experimental, mediante el estadístico W de Wilcoxon.

W de Wilcoxon	(CONCEPTUAL POSTEST) – (CONCEPTUAL PRETEST)
Z	-3,867(a)
Nivel de significación bilateral	0,000**
Diferencia de Media	8,842

** p<0.01a basado en los rangos negativos.

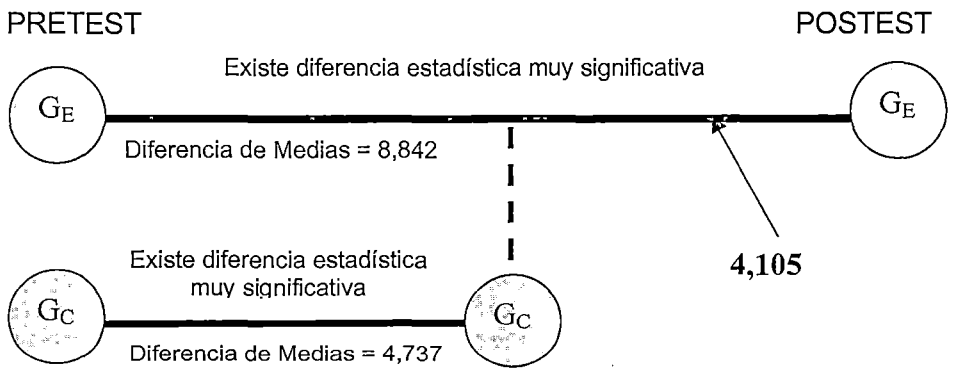
En la Tabla Nº 16 se observa, según el estadístico de contraste W de Wilcoxon, que existen diferencias estadísticas muy significativas entre los puntajes de conocimiento conceptual en la evaluación de pretest y postest en el grupo experimental.

Tabla Nº 17. Prueba de la diferencia de puntajes de conocimiento conceptual en el pretest y postest del grupo control, mediante el estadístico W de Wilcoxon.

W de Wilcoxon	(CONCEPTUAL POSTEST) – (CONCEPTUAL PRETEST)
Z	-3,652
Nivel de significación bilateral	0,000**
Diferencia de Medias	4,737

El estadístico W de Wilcoxon indicado en la Tabla Nº 17 muestra que existen diferencias estadísticas muy significativas entre los puntajes de conocimiento conceptual en la evaluación de pretest y postest en el grupo control.

Gráfico N° 17. Diferencias estadísticas significativas entre los grupos control y experimental en el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos.



El Gráfico N° 17 muestra que el grupo experimental tuvo un aprendizaje mucho más significativo, con diferencia de medias 8,842, superior al grupo control que tuvo una diferencia de medias 4,737. Esto quiere decir que existe una diferencia estadística muy significativa entre los puntajes del pretest y postest en el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos; por lo que se rechaza la hipótesis **Nula (H_o):**

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos son iguales."

Por lo tanto, se acepta la hipótesis **Alterna (H_A):**

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos no son iguales."

Como consecuencia de la afirmación anterior, queda contrastada la hipótesis: "*La metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle*".

4.4.3. APRENDIZAJE DE LOS CONOCIMIENTOS PROCEDIMENTALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS.

Hipótesis estadística 2:

La metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

Comprobación de la hipótesis estadística 2:

Nula (H_o):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos son iguales.

$$\mu_{pretest} = \mu_{postest}.$$

Alterna (H_a):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos no son iguales.

$$\mu_{pretest} ? \mu_{postest}.$$

Utilizando los estadígrafos Kolmogorov-Smirnov para la prueba de Normalidad de datos del pretest y postest en el grupo control y experimental, W de Wilcoxon para la prueba rangos señalados y pares igualados, teniendo en cuenta la dirección y magnitud de la diferencia de datos del pretest y postest en el grupo control y experimental, y U de Mann Whitney para el estudio de comportamiento de grupos independientes, comparado entre el grupo experimental y control; para el cuál se ha utilizado el software SPSS v15.0.

Diseño de la prueba de hipótesis 2

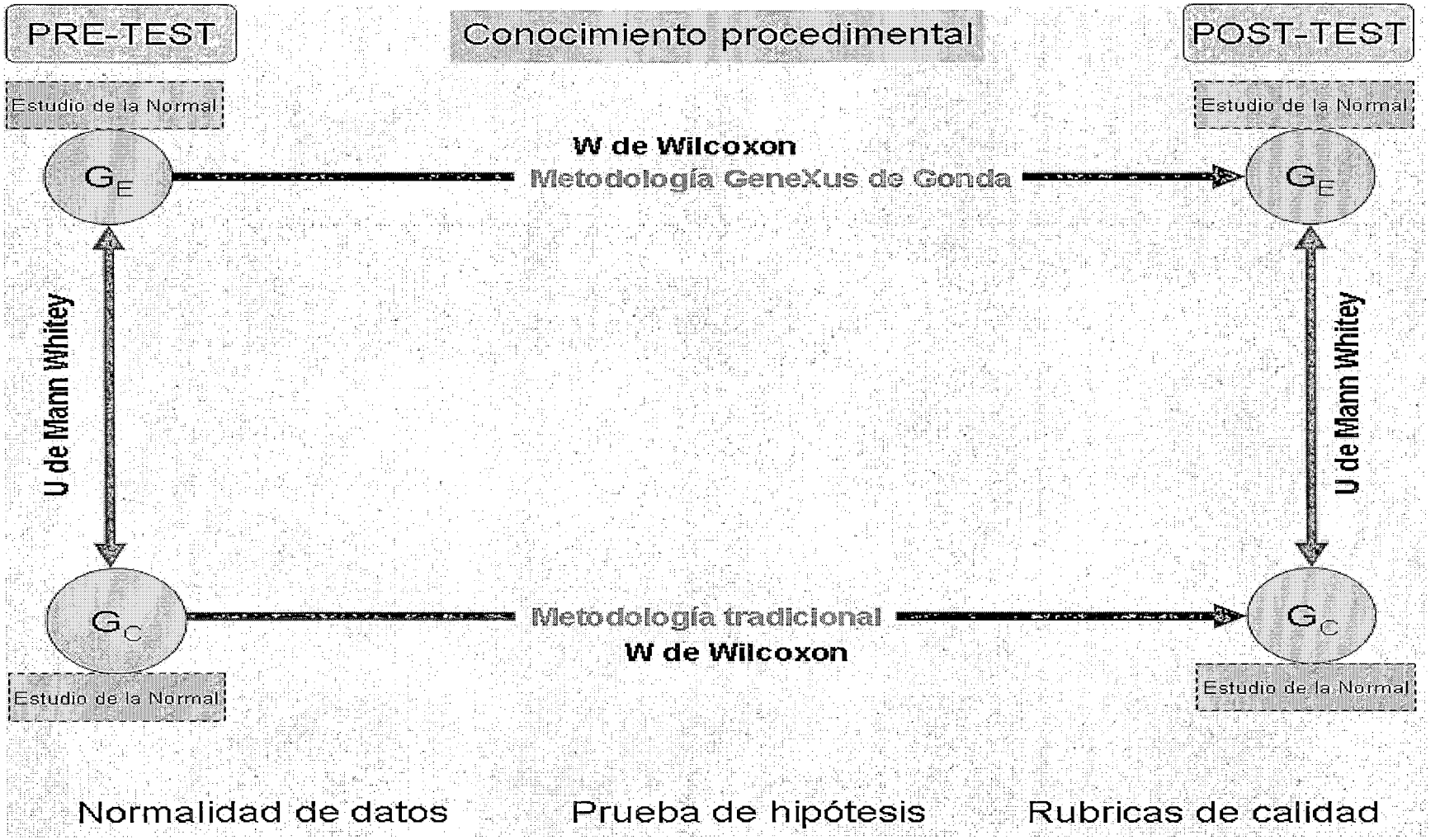
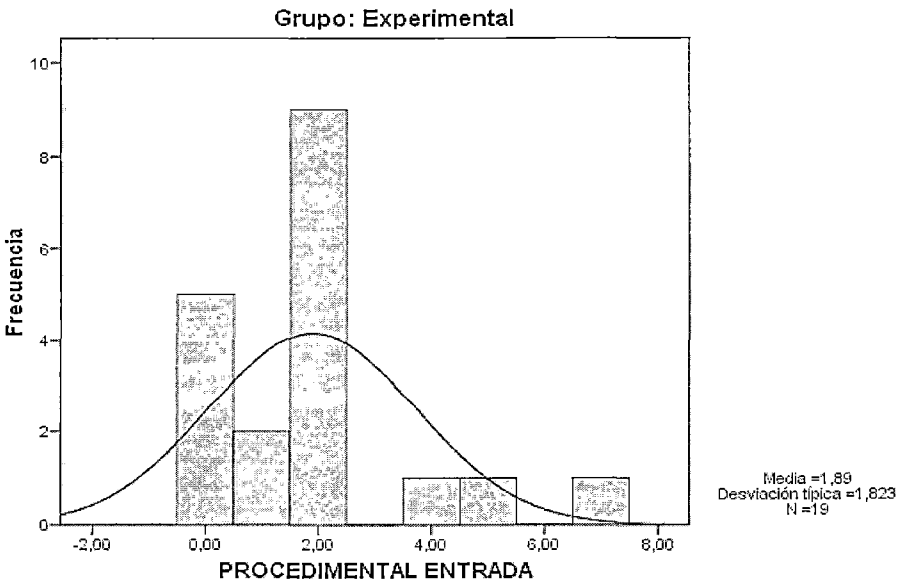
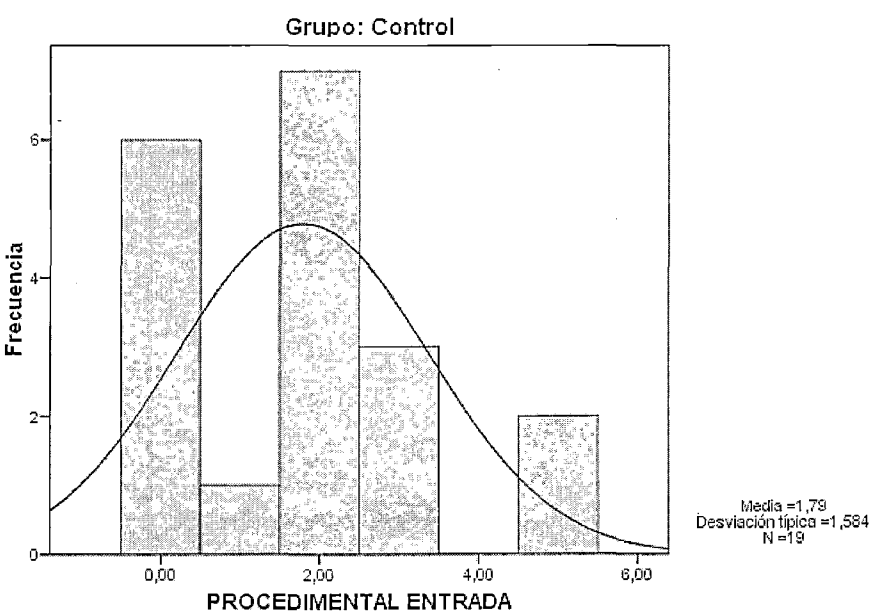


Gráfico Nº 18. Distribución de frecuencias de los puntajes en el pretest de conocimiento procedimental del grupo experimental.



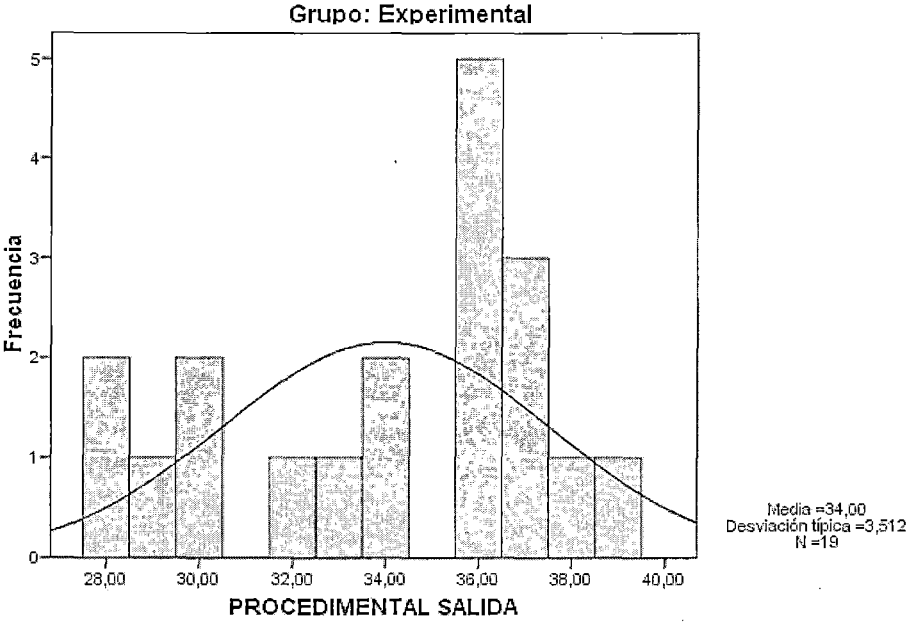
En el Gráfico Nº 18, puede observarse la distribución de frecuencias de los puntajes de conocimiento procedimental en el pretest del grupo experimental. En el mencionado gráfico, los puntajes se hallan sesgados hacia la izquierda, teniendo una media de 1,89 y una desviación estándar de 1,82. Como puede observarse en el gráfico, la curva de distribución difiere notablemente de la curva normal.

Gráfico Nº 19. Distribución de frecuencias de los puntajes del pretest de conocimiento procedimental del grupo control



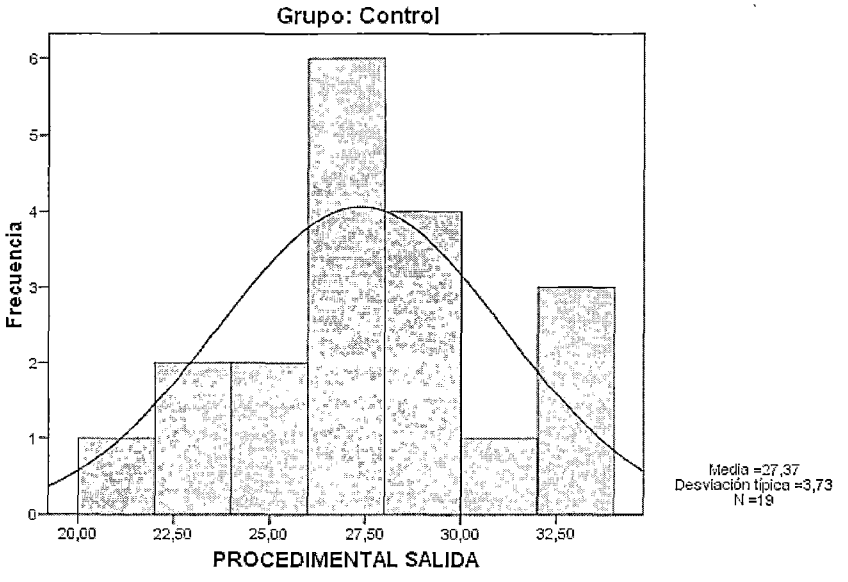
En el Gráfico N° 19 puede observarse la distribución de frecuencias de los puntajes de conocimiento procedimental en el pretest del grupo control. En dicho gráfico se observa que los puntajes se hallan sesgados hacia la izquierda, teniendo una media de 1,78 y una desviación estándar de 1,58. Asimismo, la curva de distribución difiere de la curva normal.

Gráfico N° 20. Distribución de frecuencias de los puntajes en el postest de conocimiento procedimental del grupo experimental



En el Gráfico N° 20, se observa la distribución de frecuencias de los puntajes de conocimiento procedimental en el postest del grupo experimental. Dicho gráfico muestra que los puntajes se hallan sesgados ligeramente hacia la derecha, teniendo una media de 34,00 y una desviación estándar de 3,51. En el gráfico, anterior, también se puede observar que la curva de distribución es casi rectangular y difiere de la curva normal.

Gráfico Nº 21. Distribución de frecuencias de los puntajes en el postest de conocimiento procedimental del grupo control.



En el Gráfico Nº 21 puede observarse la distribución de frecuencias de los puntajes de conocimiento procedimental en el postest del grupo control. Dicho gráfico muestra que los puntajes se hallan sesgados ligeramente hacia la derecha, teniendo una media de 27,37 y una desviación estándar de 3,73. Como puede observarse en el gráfico, la curva de distribución se asemeja a la curva normal.

Tabla Nº 18. Tipo de curva de distribución de los puntajes del conocimiento procedimental en el grupo experimental y control.

GRUPO	PRETEST	POSTEST
EXPERIMENTAL	DIFIERE DE LA DISTRIBUCIÓN NORMAL (ver Gráfico Nº 18)	DIFIERE DE LA DISTRIBUCIÓN NORMAL (ver Gráfico Nº 20)
CONTROL	DIFIERE DE LA DISTRIBUCIÓN NORMAL (ver Gráfico Nº 19)	SE ASEMEJA A LA CURVA NORMA (ver Gráfico Nº 21)

Los resultados mostrados en la Tabla Nº 18 hacen cumplir las condiciones para utilizar el estadístico U de Mann-Whitney para el análisis de los puntajes entre los grupos control y experimental (grupos no relacionados). Asimismo, se utiliza el estadístico W de Wilcoxon para analizar la diferencia de puntajes entre los grupos control y experimental (grupos relacionados) en el pretest y postest.

Tabla Nº 19. Prueba de la diferencia de puntajes en la prueba de conocimiento procedimental en las pruebas de pretest y postest entre los grupos control y experimental, mediante el estadístico W de Wilcoxon.

	Grupo experimental vs grupo control	Grupo experimental vs grupo control
	PRETEST	POSTEST
U de Mann-Whitney U	176,500	33,500
Z	-0,123	-4,308
Nivel de significancia bilateral	0,286	0,000**

** p<0,01

El estadístico U de Mann-Whitney que se presenta en la Tabla Nº 19 muestra que no existen diferencias estadísticamente significativas entre los puntajes de pretest al comparar los grupos control y experimental (p>0,01), mientras que al comparar los puntajes de postest entre los grupos control y experimental (p<0,01) se encuentran diferencias estadísticamente muy significativas (p<0.01).

Tabla Nº 20. Prueba de la diferencia de puntajes de conocimiento procedimental en el pretest y postest del grupo experimental, mediante el estadístico W de Wilcoxon.

Análisis de rangos:

		N
(PROCEDIMENTAL POSTEST) – (PROCEDIMENTAL PRETEST)	Diferencias Negativas	0
	Diferencias Positivas	19
	Empates	0
	Total	19

Prueba de rangos de signos de Wilcoxon:

	(PROCEDIMENTAL POSTEST) – (PROCEDIMENTAL PRETEST)
Z	-3,832(a)
Nivel de significancia bilateral	0,000**
Diferencia de medias	32,105

** p<0,01

a) Basado en rangos negativos.

En la Tabla Nº 20 puede observarse, según el estadístico de contraste de prueba de los signos de W de Wilcoxon, que existen diferencias estadísticas muy significativas en la dirección de los puntajes del conocimiento procedimental, entre el pretest y postest en el grupo experimental.

Tabla Nº 21. Prueba de la diferencia de puntajes de conocimiento procedimental en el pretest y postest del grupo control, mediante el estadístico W de Wilcoxon.

Análisis de rangos:

		N
(PROCEDIMENTAL POSTEST) – (PROCEDIMENTAL PRETEST)	Diferencias Negativas	0
	Diferencias Positivas	19
	Empates	0
	Total	19

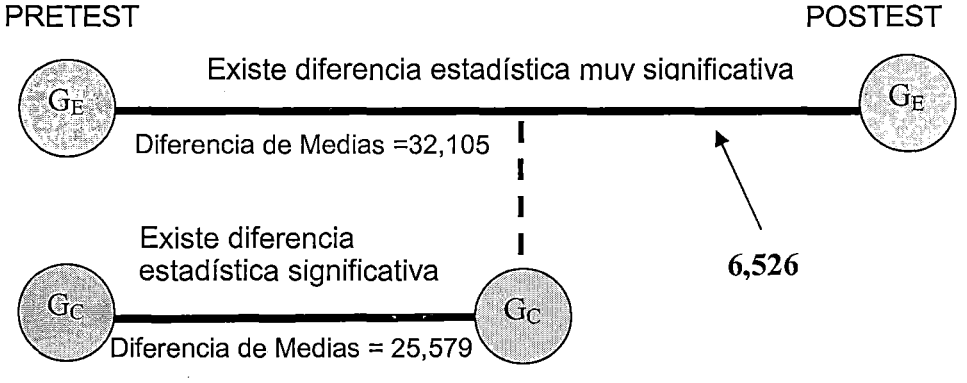
Prueba de rangos de signos de Wilcoxon:

	(PROCEDIMENTAL POSTEST) – (PROCEDIMENTAL PRETEST)
Z	-3,830(a)
Nivel de significancia bilateral	0,000**
Diferencia de medias	25,579

** p<0,01
a). basado en rangos negativos

En la Tabla Nº 21 puede observarse, según el estadístico de contraste de prueba de los signos de W de Wilcoxon, que existen diferencias estadísticas muy significativas en la dirección de los puntajes del conocimiento procedimental, entre el pretest y postest en el grupo control.

Gráfico Nº 22. Diferencia estadística significativa entre los grupos control y experimental en el aprendizaje de conocimientos procedimentales en el desarrollo de software sobre base de datos.



El Gráfico Nº 22 muestra que el grupo experimental tuvo un aprendizaje mucho más significativo, con diferencia de medias 32,105, superior al grupo control que tuvo una diferencia de medias 25,579. Esto quiere decir que existe una diferencia estadística muy

significativa entre los puntajes del pretest y postest en el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, por lo que se rechaza la hipótesis Nula (H_o):

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos son iguales."

Por lo tanto, se acepta la hipótesis Alterna (H_A):

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos no son iguales."

Como consecuencia de la afirmación anterior, queda contrastada la hipótesis: "*La metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle*".

4.4.4. APRENDIZAJE DE LOS CONOCIMIENTOS ACTITUDINALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Hipótesis específica 3:

La metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

Comprobación de la hipótesis estadística 3:

Nula (H_o):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos son significativamente iguales.

$$\mu_{pretest} = \mu_{postest}.$$

Alterna (H_a):

La diferencia de medias de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos no son significativamente iguales.

$$\mu_{pretest} ? \mu_{postest}.$$

Utilizando los estadígrafos Kolmogorov-Smirnov para la prueba de Normalidad de datos del pretest y postest en el grupo control y experimental, W de Wilcoxon para la prueba rangos señalados y pares igualados, teniendo en cuenta la dirección y magnitud de la diferencia de datos del pretest y postest en el grupo control y experimental, y U de Mann Whitney para el estudio de comportamiento de grupos independientes, comparado entre el grupo experimental y control; para el cuál se ha utilizado el software SPSS v15.0.

Diseño de la prueba de hipótesis 3

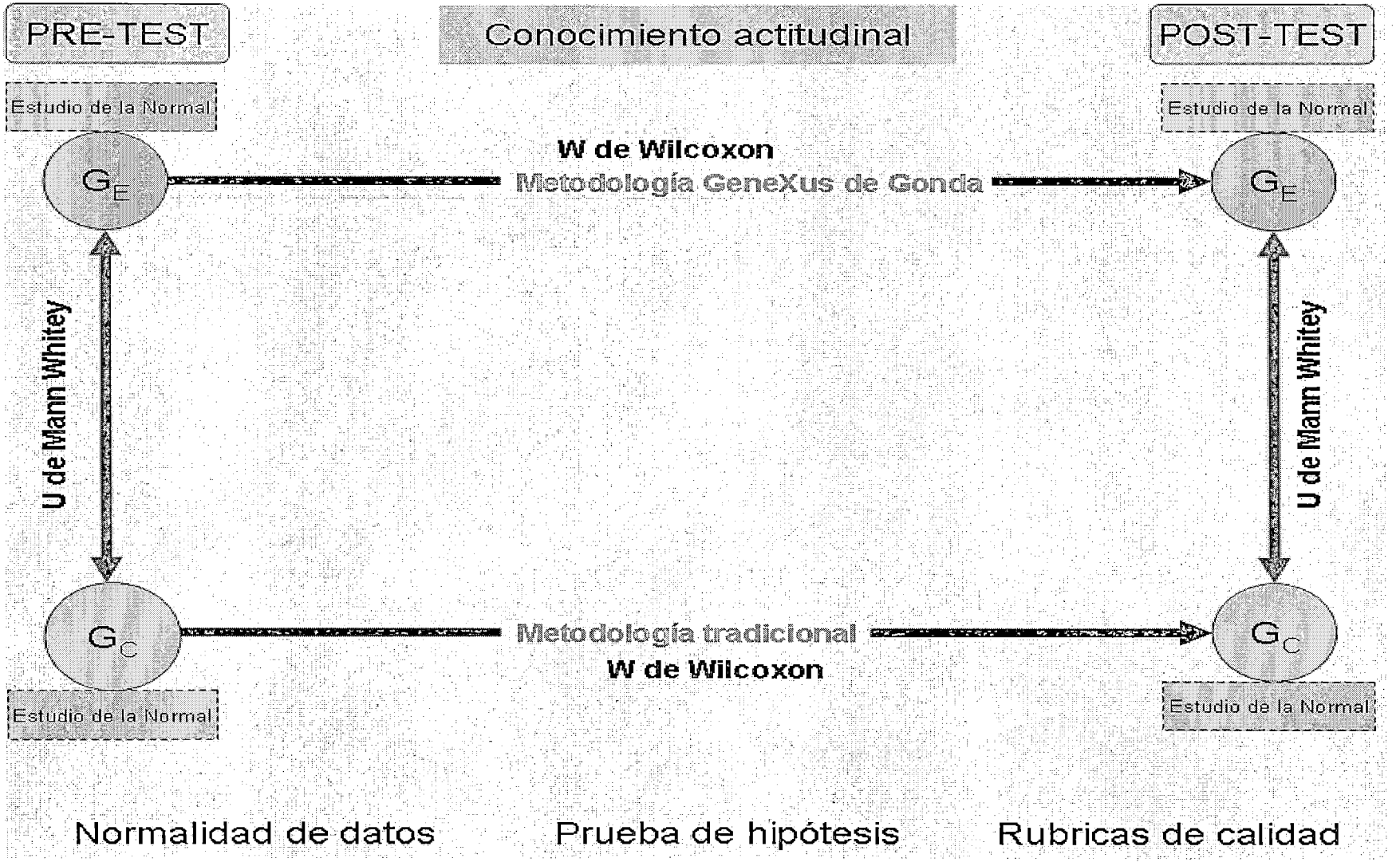
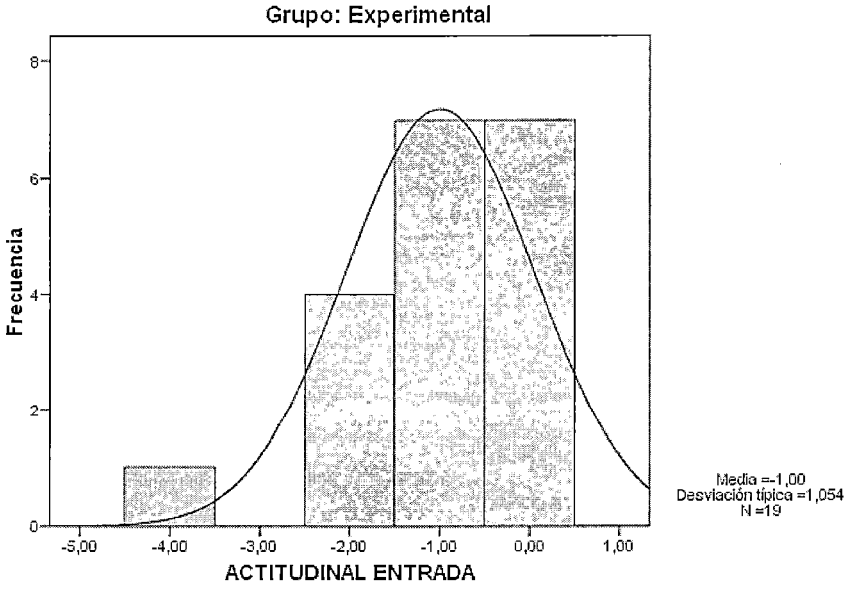
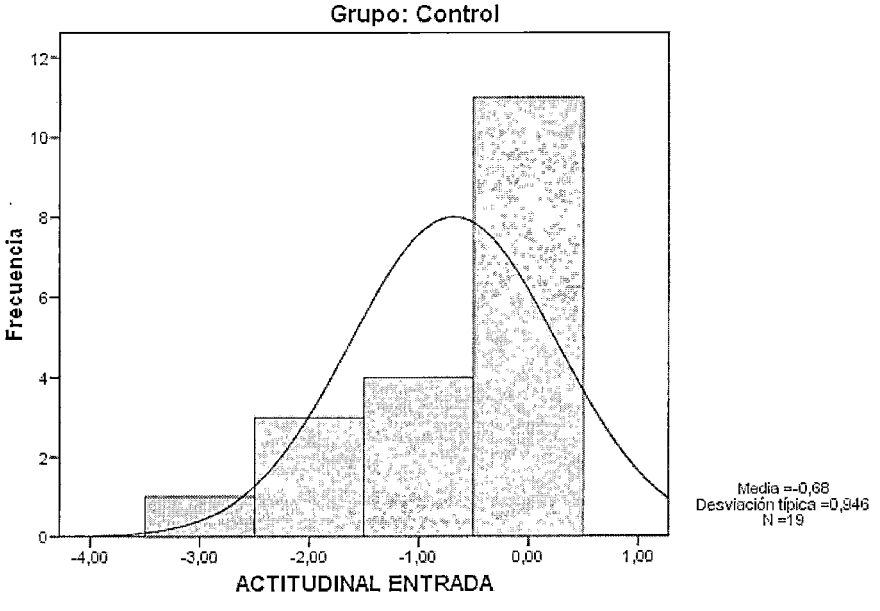


Gráfico Nº 23. Distribución de frecuencias de los puntajes en el área de conocimiento actitudinal, en el pretest del grupo experimental.



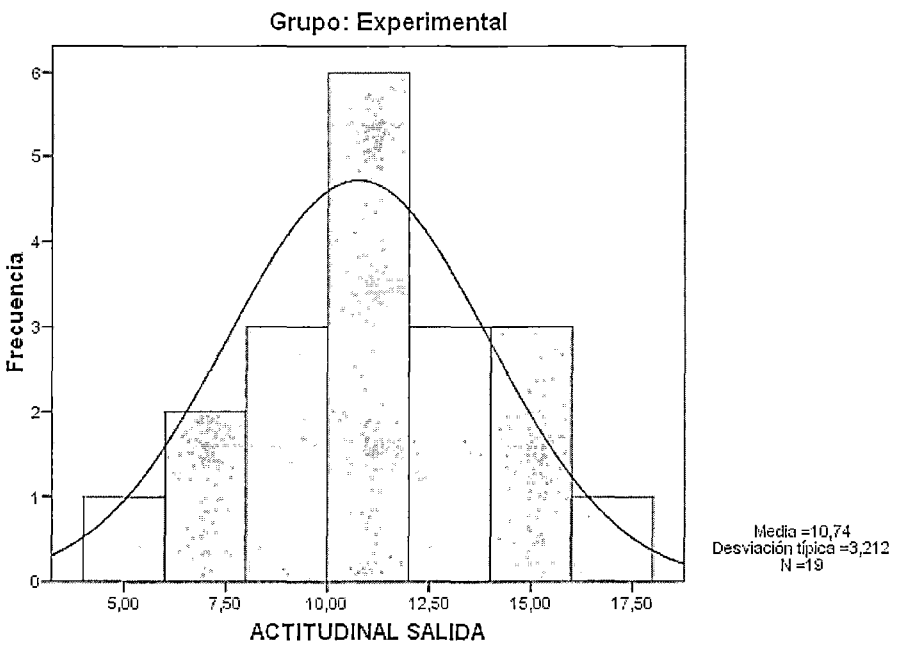
En el Gráfico Nº 23 puede observarse la distribución de frecuencias de los puntajes del área actitudinal en el pretest del grupo experimental. En dicho gráfico puede observarse que los puntajes se hallan sesgados hacia la derecha, teniendo una media de 1 y una desviación estándar de 1,05. Como puede observarse en el gráfico, la curva de distribución difiere de la curva normal.

Gráfico Nº 24. Distribución de frecuencias de los puntajes en el área actitudinal, en el pretest del grupo control.



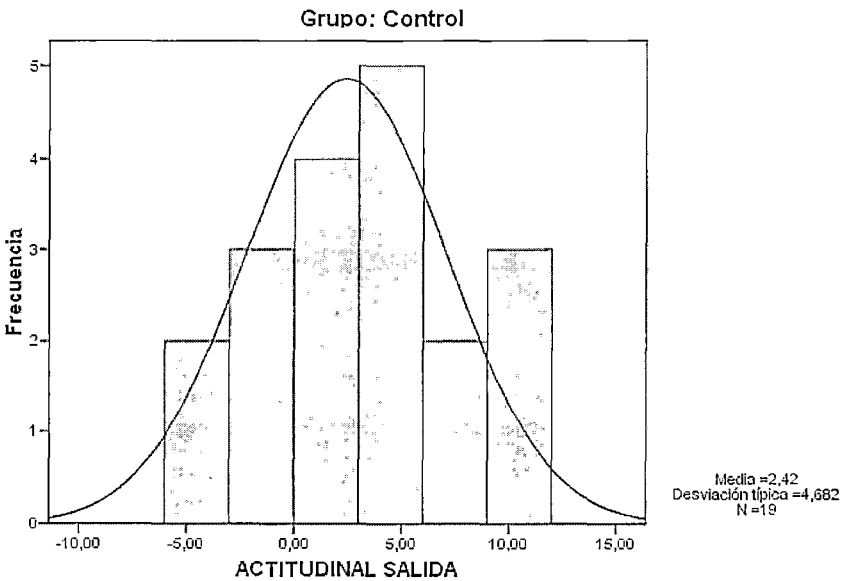
En el gráfico N° 24 puede observarse la distribución de frecuencias de los puntajes del área actitudinal en el pretest del grupo control. En dicho gráfico puede observarse que los puntajes se hallan sesgados hacia la derecha, teniendo una media de 0,68 y una desviación estándar de 0,946. Como puede observarse en el gráfico, la curva de distribución difiere notablemente de la curva normal.

Gráfico N° 25. Distribución de frecuencias de los puntajes en el área actitudinal en el postest del grupo experimental.



En el Gráfico N° 25 puede observarse la distribución de frecuencias de los puntajes del área actitudinal en el postest del grupo experimental, En dicho gráfico puede observarse que los puntajes se hallan ligeramente sesgados hacia la izquierda, teniendo una media de 10,74 y una desviación estándar de 3,21. Como puede observarse en el gráfico, la curva de distribución se asemeja mucho a la curva normal.

Gráfico N° 26. Distribución de frecuencias de los puntajes en el área actitudinal en el postest del grupo control.



En el Gráfico N° 26 puede observarse la distribución de frecuencias de los puntajes del área actitudinal en el postest del grupo control. En dicho gráfico puede observarse que los puntajes se hallan sesgados ligeramente hacia la izquierda, teniendo una media de 2,42 y una desviación estándar de 4,68. Como puede observarse en el gráfico, la curva de distribución se asemeja a la curva normal.

Tabla N° 22. Tipo de distribución de los puntajes del conocimiento actitudinal en el grupo experimental y control.

GRUPO	PRETEST	POSTEST
EXPERIMENTAL	DIFIERE DE LA DISTRIBUCIÓN NORMAL (ver gráfico N° 23)	SE ASEMEJA MUCHO A LA CURVA NORMA (ver gráfico N° 25)
CONTROL	DIFIERE DE LA DISTRIBUCIÓN NORMAL (ver gráfico N° 24)	SE ASEMEJA MUCHO A LA CURVA NORMAL (ver gráfico N° 26)

Los resultados mostrados en la Tabla N° 22 hacen cumplir las condiciones para utilizar el estadístico U de Mann-Whitney para el análisis de los puntajes entre los grupos control y experimental (grupos no relacionados). Asimismo, se utiliza el estadísticos "W" de Wilcoxon para analizar la diferencia de puntajes entre los grupos control y experimental (grupos relacionados) en el pretest y postest.

Tabla Nº 23. Prueba de diferencia de puntajes en el pretest y postest entre los grupos control y experimental en el área de conocimiento actitudinal.

	Grupo experimental vs grupo control	Grupo experimental vs grupo control
	ACTITUDINAL PRETEST	ACTITUDINAL POSTEST
U de Mann-Whitney	146,500	24,500
Z	-1,068	-4,564**
Nivel de significancia (2 colas)	0,286	0,000**

** p<0,01

La Tabla Nº 23 muestra que el nivel de significancia es de 0,000 lo que nos permite afirmar que sí existe diferencias estadísticamente significativas entre los grupos control y experimental en los puntajes del pretest y postest en el aprendizaje de los conocimientos actitudinales.

Tabla Nº 24. Prueba de significancia de los cambios en los puntajes de conocimiento actitudinal entre el pretest y postest en el grupo experimental.

Análisis de rangos:

		N
(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)	Diferencias Negativas	0
	Diferencias Positivas	19
	Empates	0
	Total	19

Prueba de rangos de signos de Wilcoxon:

	(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)
Z	-3,832(a)
Significancia (2 colas)	0,000**

** p<0,01

a) Basado en rangos negativos

En la Tabla Nº 24 puede observarse, según la prueba de los signos, que se ha producido un cambio estadísticamente muy significativo en la dirección de los puntajes del conocimiento actitudinal, entre el pretest y postest en el grupo experimental.

Tabla N° 25. Prueba de significancia de los cambios en los puntajes de conocimiento actitudinal entre el pretest y postest en el grupo control.

Análisis de rangos:

		N
(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)	Diferencias Negativas	3
	Diferencias Positivas	13
	Empates	3
	Total	19

Prueba de rangos de signos de Wilcoxon:

	(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)
Z	-2,436(a)
Significancia (2 colas)	0,015*

* p<0,01
a) basado en rangos negativos

Los resultados de la prueba de los signos indicados en la Tabla N° 25 muestran que se ha producido un cambio estadísticamente significativo en la dirección de los puntajes del conocimiento actitudinal, entre el pretest y postest en el grupo control.

Tabla N° 26. Prueba de diferencia de puntajes de conocimiento actitudinal en pretest y postest del grupo experimental.

	(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)
Z	-3,832**
Significancia (2 colas)	0,000
Diferencia de medias	11,737

**(p<0.01)

El estadístico "W" de Wilcoxon indicado en la Tabla N° 26 muestra que existen diferencias estadísticas muy significativas entre los puntajes de conocimiento actitudinal en el pretest y postest en el grupo experimental.

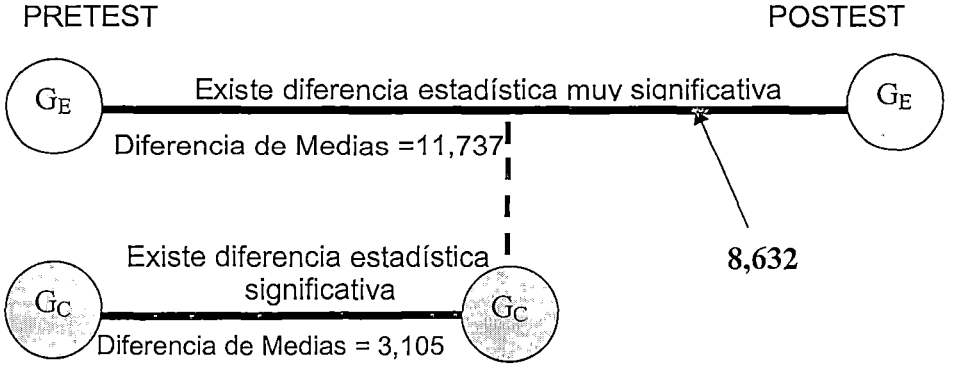
Tabla N° 27. Prueba de diferencia de puntajes de conocimiento actitudinal en el pretest y postest del grupo control.

	(ACTITUDINAL POSTEST) – (ACTITUDINAL PRETEST)
Z	-2,436*
Significancia (2 colas)	0,015
Diferencia de medias	3,105

**(p<0,01)

La Tabla N° 27 muestra, según el estadístico W de Wilcoxon, que existen diferencias estadísticas muy significativas entre los puntajes de conocimiento actitudinal en el pretest y postest en el grupo control.

Gráfico N° 27. Diferencia estadística significativa entre los grupos control y experimental en el aprendizaje de conocimientos actitudinales en el desarrollo de software sobre base de datos.



El Gráfico N° 27 muestra que el grupo experimental tuvo un aprendizaje mucho más significativo, con diferencia de medias 11,737, superior al grupo control que tuvo una diferencia de medias 3,105. Esto quiere decir que existe una diferencia estadística muy significativa entre los puntajes del pretest y postest en el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos; por lo que se rechaza la hipótesis Nula (H_o):

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos son iguales."

Por lo tanto, se acepta la hipótesis Alterna (H_A):

"La media de los puntajes obtenidos del pretest y postest del aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos no son iguales."

Como consecuencia de la afirmación anterior, queda contrastada la hipótesis: "*La aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, comparada con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle*".

4.4.5. ANÁLISIS DEL LOGRO DE APRENDIZAJE POR CLASIFICACIÓN DE COLL

4.4.5.1. CONOCIMIENTO CONCEPTUAL

La Tabla Nº 28 muestra los puntajes del logro de aprendizaje de los conocimientos conceptuales por clasificación de competencias. Los resultados del procesamiento de los mismos se presenta en el Gráfico Nº 28.

Tabla Nº 28. Prueba de diferencia de puntajes de conocimiento conceptual en el pretest y postest del grupo control.

CONOCIMIENTO CONCEPTUAL	EXPERIMENTAL		CONTROL	
Datos	14,50	76,3%	7,00	36,8%
Hechos	11,33	59,6%	9,67	50,9%
Conceptos	14,00	73,7%	11,50	60,5%
Principios o leyes	13,67	71,9%	9,33	49,1%

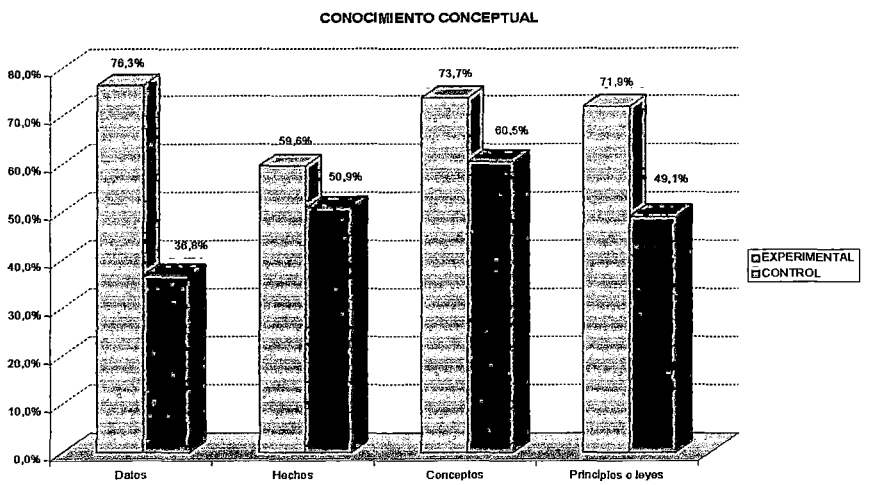


Gráfico Nº 28. Logros de aprendizaje de conocimiento conceptual del grupo experimental y control.

4.4.5.2. CONOCIMIENTO PROCEDIMENTAL

La Tabla Nº 29 muestra los puntajes del logro de aprendizaje de los conocimientos procedimentales por clasificación de competencias y los de resultados del procesamiento de los mismos se presentan en el Gráfico Nº 29.

Tabla Nº 29. Prueba de diferencia de puntajes de conocimiento procedimental en el pretest y postest del grupo control.

CONOCIMIENTO PROCEDIMENTAL	EXPERIMENTAL		CONTROL	
Generales	3,58	71,6%	2,82	56,3%
Algoritmos	3,24	64,7%	2,90	57,9%
Heurísticos	3,47	69,4%	3,11	62,2%
Destrezas y habilidades	3,42	68,4%	2,47	49,4%
Técnicas	3,53	70,5%	2,74	54,8%
Estrategias	2,89	57,8%	2,79	55,8%
Motriz cognitivo	3,53	70,6%	2,11	42,2%

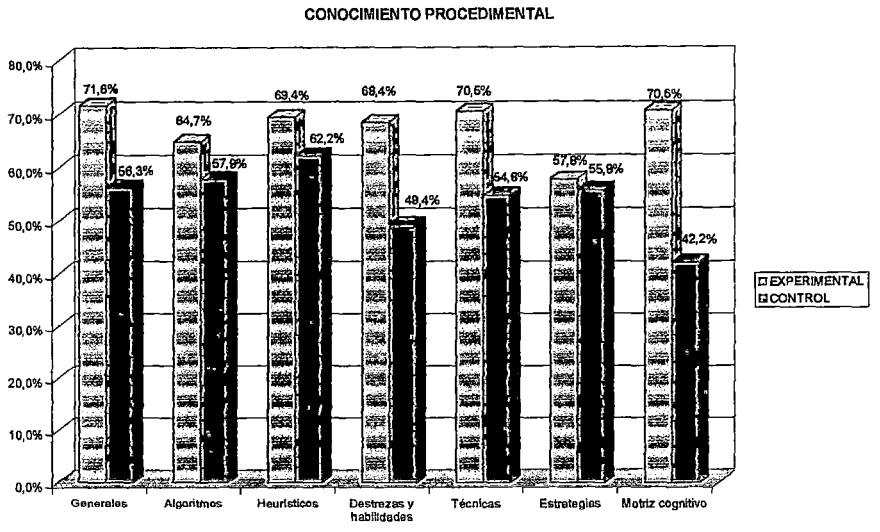


Gráfico Nº 29. Logros de aprendizaje de conocimiento procedimental de los grupos experimental y control.

4.4.5.3. CONOCIMIENTO ACTITUDINAL

La Tabla Nº 30 muestra los puntajes del logro de aprendizaje de los conocimientos actitudinales por clasificación de competencias y los de resultados del procesamiento de los mismos se presenta en el Gráfico Nº 30.

Tabla Nº 30. Prueba de diferencia de puntajes de conocimiento actitudinal en el pretest y postest del grupo control.

CONOCIMIENTO ACTITUDINAL	EXPERIMENTAL		CONTROL	
Valores	3,96	79,3%	3,58	71,5%
Normas	4,11	82,2%	3,53	70,6%
Actitudes	4,16	83,2%	3,36	67,1%
Juicios valorativos	4,09	81,7%	3,34	66,7%

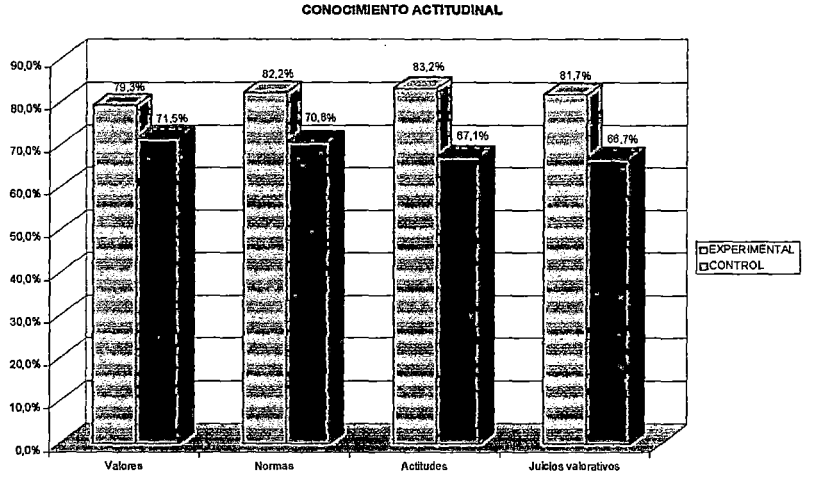


Gráfico Nº 30. Logros de aprendizaje de conocimiento actitudinal del grupo experimental y control.

4.4.6. VALORACIÓN DE CALIDAD DE APRENDIZAJE MEDIANTE LAS RÚBRICAS

4.4.5.1. APRENDIZAJE DE CONOCIMIENTO CONCEPTUAL

La rúbrica de la valoración de la calidad se determina haciendo uso de la ecuación siguiente: Valoración de escala de Calidad = Valoración de Ítems x 50

Valor de Calidad (%) = 50 (Media aritmética)

EXPERIMENTAL

Respuesta	Frecuencia de los Ítems										Valor total de calidad
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
Correcto (2)	15	12	13	13	16	10	13	12	11	17	
Incorrecto (0)	4	6	6	6	3	9	6	6	8	2	
Promedio:	1,58	1,26	1,37	1,37	1,68	1,05	1,37	1,26	1,16	1,79	1,39
Valoración de la calidad:	78,9	63,2	68,4	68,4	84,2	52,6	68,4	63,2	57,9	89,5	69,50 %

CONTROL

Respuesta	Frecuencia de los Ítems										Valor total de calidad
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
Correcto (2)	13	10	10	1	13	11	12	11	6	7	
Incorrecto (0)	3	5	9	18	5	7	7	7	13	12	
Promedio:	1,37	1,05	1,05	0,11	1,37	1,16	1,26	1,16	0,63	0,74	0,99
Valoración de la calidad:	68,4	52,6	52,6	5,26	68,4	57,9	63,2	57,9	31,6	36,8	49,50 %

Tabla Nº 31. Tabla de valoración de calidad del aprendizaje del conocimiento conceptual.

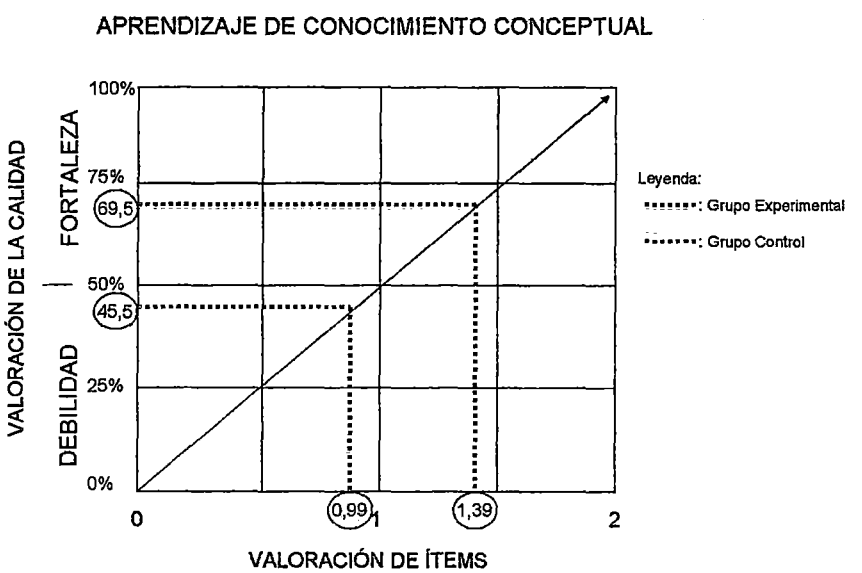


Gráfico Nº 31. Valoración porcentual de la escala de calidad de aprendizaje del conocimiento conceptual del grupo experimental y control.

4.4.5.2. APRENDIZAJE DE CONOCIMIENTO PROCEDIMENTAL

La rúbrica de la valoración de ítems de la calidad se determina haciendo uso de la ecuación siguiente: Valoración de escala de Calidad = (Valoración de Ítems x 25)-25

Valor de Calidad (%) = 25 (Media aritmética)-25

EXPERIMENTAL

Respuesta	Frecuencia de los Ítems										Valor total
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	
SIEMPRE (5)	0	5	3	5	3	4	0	2	4	3	
CASI SIEMPRE (4)	5	6	8	4	6	8	13	4	5	5	
A VECES (3)	7	6	3	5	6	4	3	7	7	7	
CASI NUNCA (2)	7	2	5	5	4	3	2	5	3	4	
NUNCA (1)	0	0	0	0	0	0	0	0	0	0	
Promedio:	2,89	3,737	3,47	3,474	3,421	3,684	3,42	3	3,53	3,37	3,40
Valoración de la Calidad:	47,37	68,42	61,8	61,84	60,53	67,11	60,5	50	63,2	59,2	60,00%

CONTROL

Respuesta	Frecuencia de los Ítems										Valor total
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	
SIEMPRE (5)	0	0	1	1	1	0	0	0	1	0	
CASI SIEMPRE (4)	5	9	5	4	3	3	5	6	1	7	
A VECES (3)	6	3	10	5	3	4	4	7	2	4	
CASI NUNCA (2)	7	5	1	9	9	12	8	5	10	8	
NUNCA (1)	1	2	2	0	3	0	2	1	5	0	
Promedio:	2,79	3,00	3,11	2,84	2,47	2,53	2,63	2,95	2,11	2,95	2,74
Valoración de la Calidad:	44,74	50	52,6	46,05	36,84	38,16	40,8	49	27,6	48,7	43,42%

Tabla Nº 32. Tabla de valoración de calidad del aprendizaje del conocimiento procedimental.

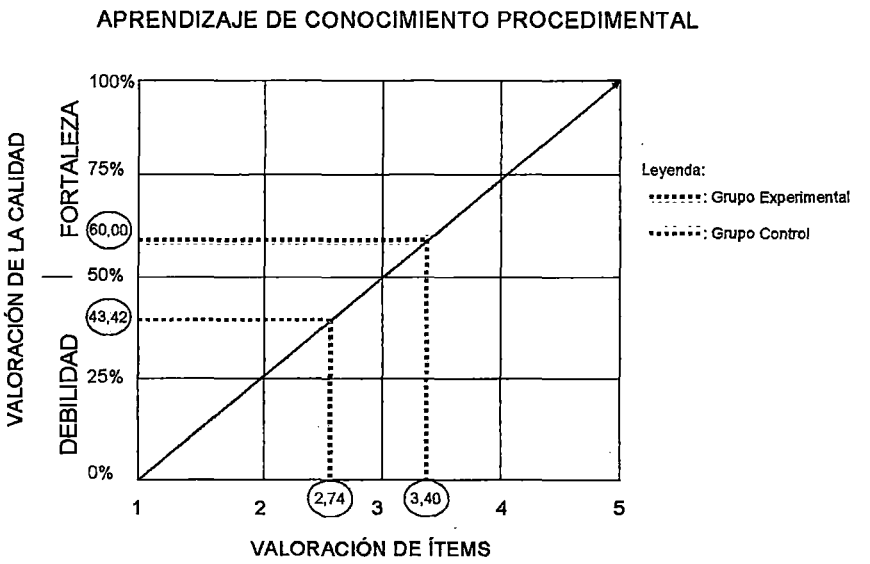


Gráfico Nº 32. Valoración porcentual de la escala de calidad de aprendizaje del conocimiento procedimental del grupo experimental y control.

4.4.5.3. APRENDIZAJE DE CONOCIMIENTO ACTITUDINAL

La rúbrica de la valoración de ítems de la calidad se determina haciendo uso de la ecuación siguiente: Valoración de escala de Calidad = (Valoración de Ítems x 25)-25

Valor de Calidad (%) = 25 (Media aritmética)-25

EXPERIMENTAL

Respuesta	Frecuencia de los Ítems										Valor total
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	
MUY BUENO (5)	4	6	1	3	5	6	2	5	5	6	
BUENO (4)	14	9	15	14	8	11	14	11	12	12	
NI BUENO NI MALO (3)	1	4	3	2	5	2	3	3	2	1	
MALO (2)	0	0	0	0	0	0	0	0	0	0	
MUY MALO (1)	0	0	0	0	1	0	0	0	0	0	
Promedio:	4,16	4,11	3,89	4,05	3,84	4,21	3,95	4,11	4,16	4,26	4,07
Valoración de la calidad:	78,95	77,63	72,37	76,32	71,05	80,26	73,68	77,63	78,95	81,58	76,84 %

CONTROL

Respuesta	Frecuencia de los Ítems										Valor total
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	
MUY BUENO (5)	2	1	1	3	0	0	1	0	0	0	
BUENO (4)	8	9	11	8	5	9	7	6	6	5	
NI BUENO NI MALO (3)	9	4	5	7	8	7	5	9	9	5	
MALO (2)	0	5	2	1	6	2	3	4	4	9	
MUY MALO (1)	0	0	0	0	0	1	3	0	0	0	
Promedio:	3,85	3,53	3,78	3,90	3,15	3,43	3,05	3,32	3,32	3,01	3,44
Valoración de la calidad:	71,26	63,30	69,60	72,44	53,74	60,80	51,25	58,03	58,10	50,35	60,89%

Tabla Nº 33. Tabla de valoración de calidad del aprendizaje del conocimiento actitudinal.

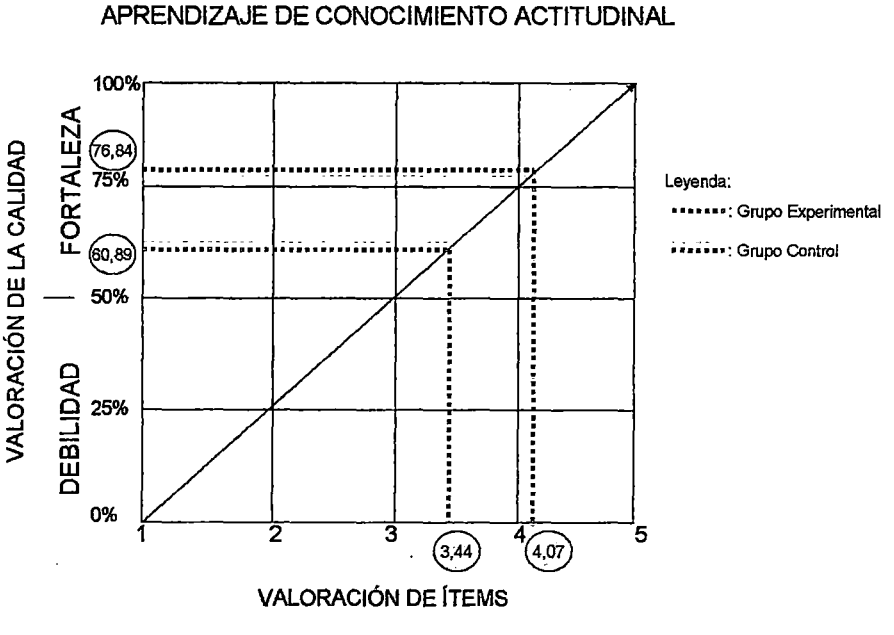


Gráfico Nº 33.Valoración porcentual de la escala de calidad de aprendizaje del conocimiento actitudinal del grupo experimental y control.

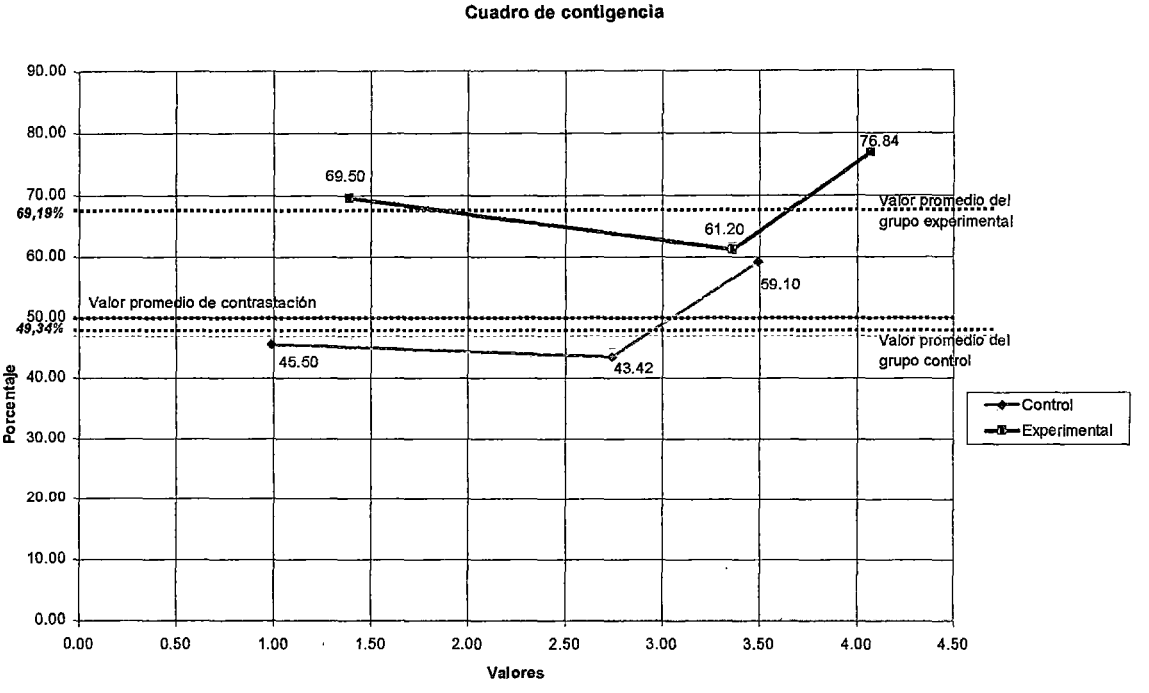


Gráfico Nº 34.Cuadro de contingencia del aprendizaje del conocimiento conceptual, procedimental y actitudinal del grupo experimental y control.

4.4.7. RESULTADOS DE ENCUESTA A LOS DOCENTES DEL ÁREA DE INFORMÁTICA

Como se puede observar, la encuesta consignada el Anexo N° 8, ha sido aplicada a los docentes del área, que trabajan en la Universidad Nacional de Educación Enrique Guzmán y Valle u otras universidades del país, y que conducen el dictado de las asignaturas base de datos o que desarrollan software sobre base de datos. Esta encuesta recoge la opinión de los dcoentes acerca de algunas características de las metodologías GeneXus de Gonda y la tradicional. Los resultados que se ha obtenido para cada ítem se presenta en las Tablas N° 34 y 35, y en el Gráfico N° 34.

Tabla N° 34. Opinión de docentes sobre la metodología GeneXus de Gonda.

METODOLOGÍA GENEXUS DE GONDA:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Media
Docente 1	5	4	5	4	5	5	4	5	4	4	4,5
Docente 2	5	5	4	4	5	5	4	4	4	4	4,4
Docente 3	4	4	5	5	4	4	5	4	4	4	4,3
Docente 4	5	4	4	4	5	5	5	4	4	5	4,5
Docente 5	4	5	5	5	4	5	4	4	5	4	4,5
Docente 6	5	4	5	4	4	5	5	5	4	4	4,5
Docente 7	5	5	5	4	5	5	5	5	5	5	4,9
Docente 8	4	5	5	4	4	4	4	5	5	4	4,4
Docente 9	4	5	5	4	4	4	4	5	5	4	4,4
Docente 10	4	5	5	5	5	5	5	5	5	5	4,9
Media	4,43	4,71	4,86	4,29	4,43	4,71	4,57	4,71	4,71	4,43	**4,59**

Tabla N° 35. Opinión de docentes sobre la metodología tradicional.

METODOLOGÍA TRADICIONAL:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Media
Docente 1	4	4	3	4	2	3	1	4	4	2	3,10
Docente 2	4	4	3		2	2	2	2	1	2	2,44
Docente 3	4	4	3	3	2	2	2	2	4	2	2,80
Docente 4	4	4	4	4	3	3	3	2	2	3	3,20
Docente 5	4	3	4	5	3	4	5	3	4	3	3,80
Docente 6	4	5	5	4	4	4	3	3	3	3	3,80
Docente 7	3	2	4	4	2	3	2	1	3	1	2,50
Media	3,75	3,50	4,25	4,25	3,00	3,50	3,25	2,25	3,00	2,50	**3,33**

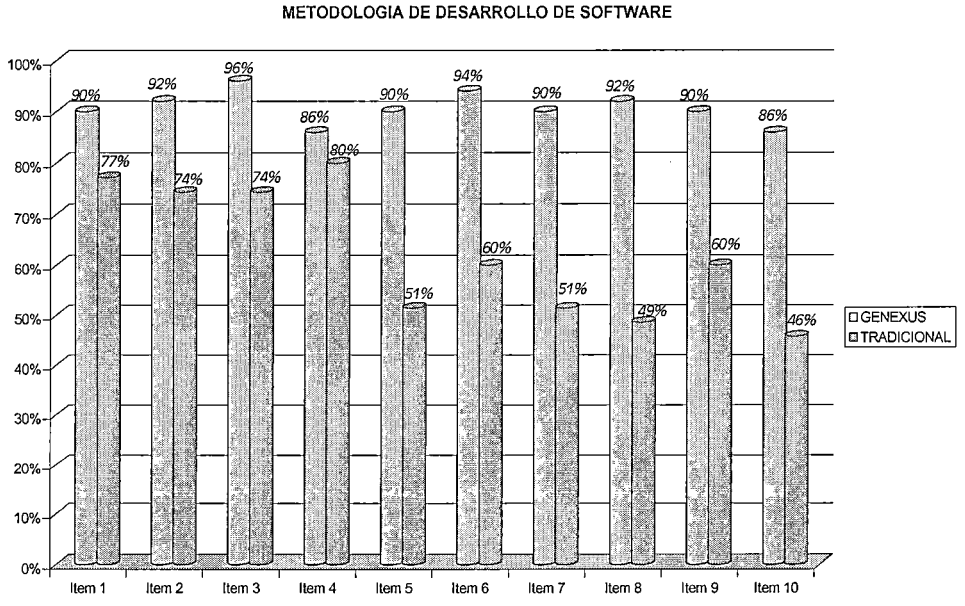


Gráfico N° 35. Resultados de opinión de docentes acerca de las metodologías de desarrollo de software sobre base de datos.

DISCUSIÓN

Según DIEZ (2003), quien experimenta el uso de la metodología IDEAL en la construcción de sistemas basados en conocimientos, "Este enfoque no pretende ser exclusivo y en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podrá ser su complemento, adaptándolo convenientemente". En tal sentido, en la presente investigación el Gráfico N° 17 muestra que en los conocimientos conceptuales el grupo experimental tuvo un logro de aprendizaje muy significativo, con diferencia de medias 8,842, en relación al grupo control que tuvo una diferencia de medias 4,737; el Gráfico N° 22 muestra que en los conocimientos procedimentales el grupo experimental tuvo un logro de aprendizaje muy significativo, con diferencia de medias 32,105, superior al del grupo control que tuvo diferencia de medias 25,579; y el Gráfico N° 27, en conocimientos actitudinales, el grupo experimental tuvo un logro de aprendizaje muy significativo, con diferencia de medias 11,737, en referencia al grupo control que tuvo una diferencia de medias 3,105. Probablemente, las mejoras en el logro de aprendizaje del desarrollo de software sobre base de datos en los estudiantes de la especialidad de Informática se deben a la aplicación de la metodología GeneXus de Gonda.

Según la conclusión de REYNOSO (2006), las metodologías no son fáciles de comparar entre sí conforme a un pequeño conjunto de criterios, y algunas metodologías, como XP, han definido claramente sus procesos, mientras que otros, como Scrum, son bastante difusos en ese sentido, lo que limita a un conjunto de principios y valores. En el trabajo de investigación se ha evaluado el logro de aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, donde el grupo experimental presentó un logro de aprendizaje de 69,5 % y el grupo control de 45,5%. En lo referente al logro de aprendizaje de los conocimientos procedimentales y actitudinales, el grupo experimental presentó un logro de aprendizaje de 60% y 76,84%, respectivamente; y el grupo control presentó un logro de aprendizaje de 45,5% y 43,42%, respectivamente (ver Gráficos N° 29 y 30). Estas mejoras probablemente se deban a la aplicación de la metodología de desarrollo de software GeneXus de Gonda, la que está

enfocada al estudio de las visiones del usuario, el desarrollo incremental y la presentación temprana del software, mediante la construcción de los prototipos.

Según el cuadro de contingencia del aprendizaje de los conocimientos conceptual, procedimental y actitudinal, tal como se presenta en el Gráfico Nº 34, la contrastación de la hipótesis general ha permitido dividir en cuartiles, y la suma de los resultados parciales hacen un total de 100%. Para ubicar los logros alcanzados en el grupo control y grupo experimental. Para la calificación de los logros alcanzados se considera los intervalos siguientes: El cuartel comprendido entre el [75% ,100%], tiene el valor de altamente significativo. El cuartel comprendido entre el [50% ,75%[, tiene el valor de muy significativo. El cuartel comprendido entre el [25% ,50%[, tiene el valor de no significativo y el cuartel comprendido entre el [0% ;25%[, tiene el valor de considerablemente no significativo. Teniendo en cuenta las consideraciones anteriores, se observa que el grupo experimental obtuvo un resultado promedio de 69,19%, el que está ubicado en el cuartil superior cuyo valor es muy significativo y el grupo control está ubicado por debajo del valor promedio de constrastación, con un resultado promedio de 49,34% ubicado en el cuartel de intervalo [75% ,100%], y cuyo valor es no significativo. En consecuencia, podemos concluir en que la aplicación de la metodología GeneXus de Gonda mejora significativamente el aprendizaje del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

En el Cuadro Nº 31, se observa los logros alcanzados por el grupo experimental y el grupo control en la dimensión del aprendizaje de los conocimientos conceptuales lo siguiente: En el proceso experimental se obtiene un promedio de 69,50%, la que está ubicado en el cuartil muy significativo y el resultado en promedio del grupo control es 45,50% el que se ubica en el cuartil no significativo.

En el Cuadro Nº 32, se observa los logros alcanzados por el grupo experimental y el grupo control en la dimensión del aprendizaje de los conocimientos procedimental, lo siguiente: En el proceso experimental se obtiene un promedio de 60,00%, el que está ubicado en el cuartil muy significativo y el resultado en promedio del grupo control es 43,42% la que se ubica en el cuartil no significativo.

En el Cuadro N° 33, se observa los logros alcanzados por el grupo experimental y el grupo control en la dimensión del aprendizaje de los conocimientos procedimental lo siguiente: En el proceso experimental se obtiene un promedio de 76,84%, el que está ubicado en el cuartil altamente significativo y el resultado en promedio del grupo control es 60,89% el que se ubica en el cuartil muy significativo.

Según la distribución de ítems por clasificación de COLL presentada en la Tabla N° 6, podemos afirmar lo siguiente:

- En los logros de aprendizaje de los conocimientos conceptuales, como lo muestran la Tabla N° 28 y el Gráfico N° 28, el aprendizaje de éste consiste en comprender y ordenar el mundo de las ideas en categorías y relaciones significativas, referidos a los datos: informaciones de fácil enunciado, el grupo experimental respondió correctamente el 76,3% y el grupo control el 36,8%. Referido a los hechos: sucesos o acontecimientos, el grupo experimental respondió correctamente el 59,6% y el grupo control el 50,9%; referido a los conceptos: definiciones acerca del objeto de estudio, el grupo experimental respondió correctamente el 73,7% y el grupo control, el 60,5%; y referidos a principios o leyes, el grupo experimental respondió correctamente el 71,9% y el grupo control el 49,1%. Presumiblemente estos resultados se deben a la aplicación de la metodología GeneXus, que permite al estudiante vivenciar el ciclo de vida del desarrollo de software sobre base de datos, considerando los fundamentos teóricos del desarrollo de software.

- En los logros de aprendizaje de los conocimientos procedimentales, como muestran la Tabla N° 28 y el Gráfico N° 28, el aprendizaje de éste consiste en realizar un conjunto ordenado de acciones ordenadas, orientado a la consecución de una meta, saber y aplicación de los conocimientos generales; el grupo experimental respondió correctamente el 71,6% y el grupo control el 56,3%, referidos a algoritmos: secuencia de acciones y decisiones a respetarse para resolver determinados problemas, el grupo experimental respondió correctamente el 64,7% y el grupo control el 57,9%, referidos a la heurística: orientar y ejecutar hacia un resultado óptimo, secuencia o un problema; el grupo experimental respondió correctamente el 69,4% y el grupo control el 62,2%, referidos a destrezas y habilidades;el grupo experimental respondió correctamente el 68,4% y el grupo control el 49,4% referidos a la técnica; el grupo experimental respondió

correctamente el 70,5% y el grupo control el 54,8%, referidos a estrategias; el grupo experimental respondió correctamente el 57,8% y el grupo control el 55,8%, y al aprender a aprender, el grupo experimental respondió correctamente el 70,6% y el grupo control el 42,2%. Es probable que los resultados obtenidos se deban a la aplicación de la metodología GeneXus, que permite realizar los procedimientos del desarrollo de software en forma incremental, mediante las aproximaciones sucesivas.

- El logro de aprendizaje de los conocimientos actitudinales, como muestran la Tabla Nº 29 y el Gráfico Nº 29, el aprendizaje de éste consiste en la valoración personal sobre determinados objetos, personas, sucesos o situaciones referidos a los valores; el grupo experimental respondió a su favor el 79,3% y el grupo control el 71,5%, referidos a las normas, patrones de conducta compartidos por el grupo social; el grupo experimental respondió a su favor el 82,2% y el grupo control el 70,6% referidos a las actitudes intuitivas: autonomismo, el grupo experimental respondió a favor el 83,2% y el grupo control el 67,1%; y referidos a juicios valorativos, el grupo experimental respondió a su favor el 81,7% y el grupo control el 66,7%: Es probable que los resultados obtenidos se deban a la aplicación de la metodología GeneXus, que obliga a trabajar en grupos pequeños de estudiantes y con tareas de colaboración mutua.

El grupo experimental obtuvo un valor de calidad de 69,50% (ver el Gráfico Nº 28) en el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "B: Bueno" y el grupo control obtuvo un valor de calidad de 49,50% en el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "C: Regular" (ver Anexo Nº 6).

El grupo experimental obtuvo un valor de calidad de 69,50% (ver el Gráfico Nº 29) en el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "B: Bueno" y el grupo control obtuvo un valor de calidad de 43,42% en el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "C: Regular" (ver Anexo Nº 6).

El grupo experimental obtuvo un valor de calidad de 76,84% (ver el Gráfico N° 30) en el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "A: Excelente" y el grupo control obtuvo un valor de calidad de 60,89% en el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, lo que se ubica en la escala de valoración "B: Bueno" (ver Anexo N° 6).

Según los resultados mostrados en las Tablas N°. 28, 29 y 30, el grupo experimental que estudió el desarrollo de software sobre base de datos mediante la aplicación de la metodología GeneXus de Gonda, presenta las siguientes actitudes positivas:

- Un incremento en el nivel de aprendizaje del desarrollo de software sobre base de datos, los estudiantes pueden ver el software construido en el menor tiempo, es decir, el proceso de construcción de software es más rápido; y a medida que ellos van adquiriendo más experiencias en esta tarea, retienen más los patrones y secuencia de acciones del desarrollo de software; en cambio, los estudiantes del grupo control requieren mayor tiempo para realizar el diseño integral del sistema, luego, tienen la necesidad de aprender un lenguaje de programación y manejar un gestor de base de datos, por lo que el software demora mucho en su construcción y sin la participación del usuario final.
- Motiva al incremento del tiempo destinado a la realización de sus tareas complementarias que el docente encarga a los estudiantes, y además destinan más tiempo al autoestudio; mientras que los estudiantes del grupo control destinan poco tiempo al autoestudio de los temas de la asignatura.
- Aumenta las habilidades de comunicación interpersonal entre los integrantes del equipo de estudiantes que desarrollan el software que el docente deja como tarea de la asignatura de base de datos; mientras que los estudiantes del grupo control realizan trabajos individuales.
- El equipo dedica la mayor parte del tiempo al análisis de los datos del área de negocios; y destinan muy poco tiempo a la etapa de codificación y/o programación del software, la que permite al estudiante comprender mejor los problemas del mundo real y buscar las formas diferentes de resolver el problema; en cambio los estudiantes del grupo control siempre están implementando las mismas soluciones al problema planteado.

- Crean en los estudiantes valores en la innovación de los procesos de desarrollo de software, como estrategia para la mejora de la calidad de software. Aprenden que no es indispensable conocer varios lenguajes de programación y gestores de bases de datos del mercado; en cambio, los estudiantes del grupo control destinan mucho tiempo al aprendizaje de los lenguajes de programación y el manejo de gestores de base de datos, y disponen de poco tiempo para su implementación.
- Aprenden a desarrollar software sobre base de datos para ambientes Windows y Web simultáneamente, en comparación con los estudiantes del grupo control que sólo crean para el ambiente Windows.

De acuerdo con el Gráfico Nº 35, los resultados de la encuesta a los docentes del área de Informática nos permiten afirmar lo siguiente:

- A la opinión acerca de que la aplicación de la metodología de desarrollo de software permite "asistir al analista y a los usuarios en todo el ciclo de vida del desarrollo de software", "dedicarse a entender los problemas del usuario" y "trabajar con las visiones de los datos del usuario"; los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor del primero; el 90%, del segundo, el 92% y del tercero, 90%, respectivamente, y los que trabajaron con la metodología tradicional opinaron a favor del primero el 77%; del segundo, el 74%, y el tercero, 51%, respectivamente. Estas diferencias pueden ser explicadas debido a que la metodología GeneXus de Gonda ofrece una mayor interacción con los usuarios finales y los desarrolladores, y a la construcción de prototipos en el menor tiempo y su respectiva prueba.

- A la opinión acerca de si la aplicación de la metodología de desarrollo de software permite "construir un modelo de datos estable de la empresa", los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor el 96% y el 74% a favor de la metodología tradicional. Esta diferencia es probable que se deba a que la metodología GeneXus de Gonda tiene por fundamento que en el área de negocios los datos son más estables que los procesos.

- A la opinión acerca de si la aplicación de la aplicación de la metodología de desarrollo de software permite "encarar el análisis de datos y diseñar la base de datos", los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor el 86% y el 80% a favor de la metodología tradicional. Probablemente, esta pequeña diferencia se deba a que la metodología GeneXus de Gonda y la tradicional permiten realizar el análisis y diseño de la base de datos obligatoriamente.

- A la opinión acerca de si la metodología de desarrollo de software utilizado permite "reducir los altos costos de mantenimiento de software" los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor el 90% y el 51% a favor de la metodología tradicional. Es probable que el uso de la GeneXus de Gonda tienda a minimizar el costo de mantenimiento del software y la metodología tradicional a mantener altos costos de mantenimiento.

- A la opinión acerca de si la aplicación de la metodología de desarrollo de software permite "Obtener prototipos de software automáticos" y "desarrollar software independiente de lenguajes y gestores de bases de datos", los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor del primero el 94% y 20 90% del segundo el 90%; y en la metodología tradicional a favor del primero el 60% y, del segundo, el 51%. Estas diferencias de opiniones pueden ser explicadas debido a que la metodología GeneXus de Gonda es independiente de los lenguajes de programación y gestores de bases de datos y la metodología tradicional obliga a conocer los anteriormente mencionados.

A la opinión acerca de si la aplicación de la metodología de desarrollo de software permite "Implementar un desarrollo incremental de software", los docentes que utilizaron la metodología GeneXus de Gonda en la enseñanza del desarrollo de software estuvieron a favor el 92% y el 60% estuvo a favor de la metodología tradicional. Probablemente esta diferencia de opiniones pueda ser explicada debido a que la metodología GeneXus de Gonda incorpora implícitamente la metodología incremental; en cambio, la metodología tradicional, no.

CONCLUSIONES

De las pruebas de hipótesis, la discusión realizada y de los resultados obtenidos se llega a las siguientes conclusiones:

1. La aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional.

2. La aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional.

3. La aplicación de la metodología GeneXus de Gonda mejora el aprendizaje de los conocimientos actitudinales del desarrollo de software sobre base de datos, en comparación con la metodología tradicional.

4. La aplicación de la metodología GeneXus de Gonda mejora el aprendizaje del desarrollo de software sobre base de datos, en comparación con la metodología tradicional, en los estudiantes del IV ciclo de la especialidad de Informática de la Universidad Nacional de Educación Enrique Guzmán y Valle.

SUGERENCIAS

1. Proponer que, de considerarlo pertinente, la Escuela de Postgrado de la Universidad Nacional de Educación Enrique Guzmán y Valle auspicie el desarrollo de una Conferencia Magistral, a fin de difundir los resultados de la presente investigación a nivel de instituciones de formación de docentes de la especialidad de Informática.

2. Que la Escuela de Postgrado de la Universidad Nacional de Educación Enrique Guzmán y Valle, promueva el desarrollo de Seminarios-Taller a nivel de docentes de la especialidad de Informática, Computación y afines, para su capacitación en la aplicación de la metodología GeneXus de Gonda en el aprendizaje de los conocimientos conceptuales, procedimentales y actitudinales del desarrollo de software sobre base de datos.

3. Que la Universidad Nacional de Educación Enrique Guzmán y Valle, a través de la editorial universitaria, factibilice la difusión de la presente investigación, a fin de lograr el efecto multiplicador en los docentes de la especialidad de Informática, Computación y afines.

REFERENCIAS BIBLIOGRÁFICAS*

AGUILAR SIERRA, Alejandro (2003). Tesis: "*Las metodologías ágiles en la enseñanza de la Ingeniería de Software*". Facultad de Ingeniería de Sistemas. Universidad Nacional Autónoma de México.

ALARCON, Jacinta (2003). Tesis: "*Los medios educativos computarizados y sus implicancias en el proceso de enseñanza y aprendizaje, en la Facultad de Ciencias de la Educación y Humanidades de la Universidad Nacional San Luis Gonzaga de Ica.*" ICA: UN "SLG".

AUSUBEL, David y otros (1983). *Psicología Educativa: Un punto de vista cognoscitivo.* 2da Ed. México: TRILLAS.

ARTech Inc (1993). *GeneXus General View: 1989-2003.* Montevideo, Uruguay: Ed. GeneXus Developer Library.

BACHMAN, Charles N. (1964). *The Integrated Data Store, a General Purpose Programming System for Random Access Memories.* 1ra ed. Pones, Arizona: ACM Press.

BARBERÀ, Elena (2004). *La enseñanza a distancia y los procesos de autonomía en el aprendizaje.* Uiversitat Oberta de Catalunya, Barcelona. España.

BLAT, Josep (2001). *Introducción a la Ingeniería del Software: diseño de interfaces.* 1ra ed. Barcelona, España: Universitat Pompeu Fabra.

BOOCH, Grady y otros. (1999). *El lenguaje unificado de modelado.* 2da ed. Madrid, España: Addison Wesley Iberoamericana.

BOEHM BARRY, W. (1988). *Applying process programming to the spiral Model.* Proc. Fourth software process. Work Shop. IEEE. UCLA.

BUSTOS PACHECO, Miguel (2002). Informe de Investigación: "*Elaboración de software educativo sobre modificación de conducta*". Lima, Instituto de Investigaciones de la UNE.

CANALES OPAZO, Tatiana. (2006). *Normas de la Asociación de Psicólogos Americanos (A.P.A.) para citar información bibliográfica* (http:// facultad. usfq.edu.ec/ cornellm /Academia%20Documents/ Apa_Edicion5. pdf).

CASTAÑO, Adoración de Miguel y otros (2000). *Fundamentos y modelos de Bases de Datos.* 2da ed. Madrid: Ra-Ma.

*CANALES OPAZO, Tatiana. (2006). Normas de la Asociación de Psicólogos Americanos (A.P.A.) para citar información bibliográfica (http://facultad.usfq.edu.ec/cornellm/Academia %20Documents/Apa_Edicion5.pdf).

CODD, Edgar F. (1971). *"Further Normalization of the Data Base Relational Model"*, IBM Research Report. San José, California.

(1976). *"Seminario Avanzado de Bases de Datos", base de datos y avances.* Pontificia Universidad Católica de Río de Janeiro. Río de Janeiro, Brasil.

COLL, César, MAURI, Teresa, ONRUBIA, Javier (2006). *Análisis y resolución de casos-problema mediante el aprendizaje colaborativo.* Revista de Universidad y Sociedad del Conocimiento (RUSC). Vol. 3, n.° 2.

COLL, César y otros (2000). *Psicología do ensino.* 2da Ed. Sãu Paulo, Brasil: Editora Artes Médicas Sull Ltda.

COLL, César. (1997). *Aprendizaje escolar y construcción del conocimiento.* 2da Ed. Paidós. Barcelona, España.

COROMINAS, Joan (1998). *Breve diccionario etimológico de la Lengua Castellana.* 4ta ed. Madrid, España: Gredos S.A.

DATE, Chris J. (1976). *An Introduction to Database Systems*, 1ra ed. New York, New York: Addisson-Wesley Publishing.

(1983). *"The Outer Join". Proc. 2nd International Conference on Databases (ICOD-2).* Cambridge, England.

DIEZ, Eduardo (2003). *Aseguramiento de la calidad en la construcción de sistema basados en el conocimiento.* Tesis de Maestría publicada electrónicamente. Instituto Tecnológico de Buenos Aires. Buenos Aires.

FERRÁNDEZ, Adalberto (1997). *El perfil profesional de los formadores.* (Documento mimeografiado). Universitat Autónoma de Barcelona, Departamento de Pedagogía Aplicada.

GARCÍA ORÉ, Celestino (2002). Métodos estadísticos en la evaluación educacional. Lima: CONCYTEC-OFOPCYTE.

GARCÍA, Teonila. (2003). Tesis: *Estimulación de la creatividad en la Facultad de Ingeniería Industrial para el desarrollo y producción de software.* Universidad Nacional Mayor de San Marcos. CIDESOFT.

GARRET, Henry E. (1990). *Estadística en psicología y educación.* 1ra ed. México: Editorial Paidós.

GONDA, Breogán, JODAL, Juan Nicolás (1995). *"Proyecto GeneXus: Resumen del Trabajo Ganador del Premio Nacional de Ingeniería 1995". Academia Nacional de Ingeniería.* Montevideo, Uruguay.

(2003). *Desarrollo Incremental.* 3ra ed. Montevideo, Uruguay: GeneXus Developer Library - Artech Inc.

(2006). *Desarrollo Incremental.* 4ra ed. Montevideo, Uruguay: GeneXus Developer Library - Artech Inc.

GUILFORD J. P. Y FRUCHTER, Benjamín (1984). *Estadística aplicada a la psicología y la educación.* 1ra ed. Colombia. McGraw-Hill.

HAMMER, Michael y CHAMPY, James (1993). *Reengineering de corporation: A manifesto for Business Revolutions.* 1ra ed. Harper Collins. New York.

HOSSIAN, Alejandro Armando. (2003). Tesis: *"Sistema de asistencia para la selección de estrategias y actividades instruccionales".* Instituto Tecnológico de Buenos Aires. Buenos Aires. Argentina.

JACKSON, Michael (1975). *Principles of Program Design.* 2da ed. New York, New York: Academy Press.

JEAN-DOMINIQUE, Warnier (1976). *The Logical Construction of Programs.* 1da ed. New York: Van Nostrand Reinhold.

(1975). *Les procedures de traitement et leurs donnes.* 1da ed. Paris: Les Editions D´Organization.

KIT, Edward (1995). *Software testing in the real world.* Boston, MA: Addison-Wesley Publishing Company.

KUVAJA, P., y otros (1994). *Software Process Assessment and Improvement: The BOOTSTRAP Approach.* Oxford. Blackwell Business Publishers.

KNUTH, Donald E. (1997). The art of computer programming. Fundamental algorithms. Vol. I. 3ra ed. USA: Addison-Wesley.

KRUG, Steven (2001). *No me hagas pensar, Una aproximación a la usabilidad en la Web.* 1ra ed. Traducción: KME Sistemas. Madrid: Pearson Educación.

KOZULIN, Alex (1996). *Instrumentos psicológicos. La educación desde una perspectiva sociocultural.* Barcelona, España: Paidós.

LI LOO KUNG, Carlos Antonio (2002). Tesis: *"Propuesta de un sistema de registro de evaluación de los aprendizajes utilizando un software para centros educativos secundarios, Iquitos, 2002".* Iquitos: Universidad Nacional de la Amazonía Peruana.

LUQUE RUIZ, Irene y otros (2002). *Bases de datos desde Chen hasta Codd con Oracle.* México. 1ra ed. Alfaomega Ra-Ma.

MAIER, David (1983). *The Theory of Relational Databases.* 1ra ed. *París:* Computer Science Press.

MÁRQUEZ LISBOA, Daniel (2006). *Guía práctica GeneXus. Desarrollo basado en conocimiento.* 1ra ed. Montevideo, Uruguay: Grupo Magró.

MAYHEW, J. Deborah (1999). *The usability engineering lifecyle.* 1ra ed. San Francisco, California: Morgan Kauffmann.

McDERMID, Gregory J. (1993). *Software Development Process Model.* 1ra ed. En software engineer's Reference Book. CRC Press.

NORMAN, A. Donald (1990). *La psicología de los objetos cotidianos.* 1ra ed. Madrid, España: Nerea editorial.

NOVOA MONTAÑO, Lorena del Carmen (2006). *Red de maestros de maestros.* Chile. Noviembre 2006, On line (http://www.rmm.cl/index_sub.php?id_contenido= 8693&id_seccion=2094 &id_portal=329).

ONRUBIA, Jorge (1997). *Enseñar: crear zonas de desarrollo próximo e intervenir en ellas.* Barcelona, España: Graó.

ORR, Ken (1977). *Structured Systems Development.* 1ra ed. New York, New York: Yourdon Press.

PAGANO, Robert (1999*). Estadísticas para las ciencias del comportamiento.* 5ta ed. México. Internacional Thomsan Editores.

PAULK, Mark y otros (1993). *Capability Maturity Model for Software: Version 1.1.* Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University. Pittsburg. Pennsylvania.

PIAGET, Jean. (1972). La equilibración de las estructuras cognitivas. Problema central de desarrollo. Madrid: Siglo XXI. (Publicação original em francés, no ano de 1975).

PISCOYA HERMOZA, Luís (1995). *Investigación Científica y Educacional.* 2da ed. Lima, Perú: Amauta Editores.

PISCOYA HERMOZA, Luis. (1995). *Investigación Científica y Educacional.* 2da ed. Lima, Perú: Amauta Editores.

POZO MUNICIO, Juan Ignacio (1999). *Aprendices y maestros.* 3ra ed. Madrid, España: Alianza.

PRESSMAN, Roger (2006). *Ingeniería del Software: Un enfoque práctico.* 6ta ed. México: McGraw-Hill.

REINGRUBER, Michael y otros (1994). *The Data Modelling Handbook.*

REYNOSO, Carlos (2006). *Métodos heterodoxos en desarrollo de software.* Tesis de Maestría en Ingeniería de Sistemas. Universidad de Buenos Aires. Buenos Aires.

RIZZI, Marcelo Francisco (2005). Tesis: *"Sistema experto asistente de requerimiento".* Instituto Tecnológico de Buenos Aires. Buenos Aires. Argentina.

ROSSI D., Bibiana (2001). Tesis: "*Sistema experto de ayuda para la selección del modelo de ciclo de vida".* Argentina. Universidad Privada Instituto Tecnológico de Buenos Aires.

ROYCE, Winston (1970). *Managing the Development Of Large Software System: Concepts and Techniques.* Proc. of IEEE. WESCON.

SCHULMEYER G.G y MacKENZIE, G. (2000). *Verification and Validation of Modern Software-Intensive Systems.* Boston. Prentice Hall.

RUBIN, Jeffrey (1994). *Handbook of Usability Testing. How to plan, test and conduct effective tests.* New York, New York: John Willey & Sons.

SAYAGO, Sergio (2002). *Técnicas de Ingeniería de Usabilidad y metodologías de diseño en algunas aplicaciones informáticas.* Barcelona, España: Universidad Pompeu Fabra.

SEFFAH, Ahmed. y ANDREEVSKAIA, Alina (2003). *Empowering Software Engineers in Human-Centered Design. in Proceedings of the ICSE.* III Conferencia en Portland, Oregon. IEEE Computer Society.

SHNEIDERMAN, Ben (1998). *Designing the User Interface, Strategies for Effective Human-Computer Interaction.* 1ra ed. USA: Addison Wesley.

SHEPARD, Terry y KELLY, Diane (2000). *Task-directed software inspection technique: an experiment and case study.* Of the 2000 conference of the Centre for Advanced Studies on Collaborative research. November. Mississauga, Ontario, Canadá.

SILBERSCHATZ, Abraham y otros (2002). *Fundamento de base de datos.* Madrid. MacGraw-Hill, ISBN: 84-481-3654-3.

SOMMERVILE, Lan (2001). Software Engineering, 6ta. ed. USA: Addison Wesley. Reading.

TAPIA MOLINA, Toribio. (2004). Tesis: *Metodología de sistemas blandos y cuadro de mando integral para la gestión de la Universidad Tecnológica de los Andes de Apurímac.* Universidad Nacional Federico Villarreal. Lima.

TEJADA FERNÁNDEZ, José (1999). *"Acerca de las competencias profesionales".* Documento publicado en la Revista Herramientas. Parte 1. España.

TÉLLEZ VALERO, Alberto (2005). En la Tesis: *Extracción de información con algoritmos de clasificación,* para optar la Maestría en Ciencias Computacionales en el Instituto Nacional de Astrofísica, Óptica y Electrónica. México.

TRINIDAD RIVERA, Lorgio Heraclio. (2006). Tesis: *Modelo de sistema de información gerencial en la facultad de una universidad pública para ventajas competitivas en el tema decisiones.* Universidad Nacional Federico Villarreal. Lima.

TRISTÁN VIDALÓN, Luis Alfredo (2003). Tesis: *Una metodología de análisis de factores que intervienen en el desarrollo de sistemas de información de apoyo para la toma de decisiones.* Universidad Nacional Mayor de San Marcos. Lima, Perú.

UNIVERSIDAD NACIONAL DE EDUCACIÓN Enrique Guzmán y Valle (2004). Currículo 2004. Resolución Nº 0017-2004-R-UNE. 1ra ed. Lima: CEMED.

VICHILO, César (2003). Tesis: *Bases para la instrumentación computacional del constructivismo, aplicado a las ciencias exactas en la enseñanza primaria.* Facultad de Ingeniería de Sistemas. Universidad de las Américas Puebla. Escuela de Ingeniería, Departamento de Ingeniería en Sistemas Computacionales. Chulula, Puebla, México.

VICTORIO ECHAVARRIA, Jorge (2007). Tesis Doctoral: *Los módulos didácticos de ortografía a través de la multimedia y su eficacia en el aprendizaje significativo.* Universidad Nacional de Educación Enrique Guzmán y Valle. Perú.

VEGA PORRAS, Pablo (2002). *Evaluación Educativa.* Lima: Editorial de la UNE.

ULLMAN, Jeffrey D. (1988). *Principles of Database and Knowledge-Base Systems.* 1ra ed. USA: Computer Science Press.

YOURDON, Edward, CONSTANTINE, Larry (1978). *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* 1ra ed. New York, New York: Yourdon Press.

(1975). A *Case Study in Structured Programming – Redesign of a Payroll System.* 2da ed. New York, New York: IEEE.

ZAHRAN, Sami (1998). *Software Process Improvement.* Practical Guidelines for Business Success. SEI Series in Software Engineering. Massachussets. 1ra ed. Addison-Wesley Longman.

ZANONI, A. (2002). *Las metodologías ágiles en la enseñanza de la Ingeniería de Software.* Facultad de Ciencia de la Computación de la Pontificia Universidad Católica de Río Grande del Sur, Porto Alegre, Brasil.

ZAVALA, Antonio (1993). *Cómo trabajar los contenidos procedimentales en el aula.* Barcelona, España: Graó /ICEU.

ANEXOS

ANEXO Nº 1:
Instrumento de validación de expertos.

ANEXO Nº 2-A
Consolidado de los resultados del pretest y postest de conocimientos conceptuales, procedimentales y actitudinales del grupo experimental.

ANEXO Nº 2-B
Consolidado de los resultados del pretest y postest de conocimientos conceptuales, procedimentales y actitudinales del grupo control.

ANEXO Nº 3
Pretest de conocimientos conceptuales, procedimentales y actitudinales del desarrollo de aplicaciones sobre base de datos.

ANEXO Nº 4
Postest de conocimientos conceptuales, procedimentales y actitudinales del desarrollo de software sobre base de datos.

ANEXO Nº 5
Manual de enseñaza de desarrollo de aplicaciones sobre base de datos mediante el uso de la metodología GeneXus de Gonda.

ANEXO Nº 6
Tabla de valoración porcentual de rúbricas del aprendizaje del grupo experimental y control.

ANEXO Nº 7
Tabla de evaluación de expertos de los instrumentos: Test de conocimientos conceptuales, procedimentales y actitudinales.

ANEXO Nº 8
Tabla de resultados de encuesta a docentes del área de informática.

ANEXO: Nº 1

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

INSTRUMENTO DE VALIDACIÓN DE EXPERTOS

I. DATOS GENERALES

Apellidos y Nombres del experto	Cargo o Institución donde labora	Nombre del Instrumento de Evaluación	Autor del instrumento
		Test de Conocimientos Conceptual	Florencio Flores Ccanto

Título: La metodología GeneXus de Gonda y la Tradicional en el aprendizaje del desarrollo de software sobre base de datos de los estudiantes del IV Ciclo de la especialidad de Informática de la Universidad Nacional de Educación - Enrique Guzmán y Valle.

II. ASPECTOS DE VALIDACIÓN

INDICADORES	CRITERIOS	Deficiente 0 - 20				Regular 21 - 40				Buena 41 - 60				Muy Buena 61 - 80				Excelente 81 - 100			
		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
1. CLARIDAD	Es formulado con lenguaje apropiado																				
2. OBJETIVIDAD	Está expresado en conductas observables.																				
3. ACTUALIDAD	Está acorde a los cambios de la tecnología educativa																				
4. ORGANIZA-CION	Existe una organización lógica.																				
5. SUFICIENCIA	Comprende los aspectos en cantidad y calidad.																				
6. INTENCIONA-LIDAD	Adecuado para valorar el servicio educativo.																				
7. CONSISTENCIA	Basado en aspectos teóricos científicos.																				
8. COHERENCIA	Entre los índices ,indicadores y las dimensiones.																				
9. METODOLOGIA	La estrategia responde al propósito del diagnóstico.																				

III. OPINIÓN DE APLICABILIDAD :

IV. PROMEDIO DE VALORACIÓN :

Lugar y Fecha	**DNI : N°**	**Firma del Experto Informante**	**Teléfono N°**

Lima, de de 2006.

ANEXO Nº 2-A

Consolidado de los resultados del Pre-test y Post-test de conocimientos conceptuales, procedimentales y actitudinales del grupo experimental:

Nº	APELLIDOS Y NOMBRES	Conocimiento Conceptual		Conocimiento procedimental		Conocimiento actitudinal	
		Pre-Test (Puntaje)	Post-Test (Puntaje)	Pre-Test (Suma)	Post-Test (Suma)	Pre-Test (Suma)	Post-Test (Suma)
01		**6**	**14**	**1**	**30**	0	**10**
02		**6**	**14**	**0**	**39**	-1	**8**
03		**6**	**16**	**0**	**36**	0	**10**
04		**4**	**10**	**2**	**34**	-2	**13**
05		**6**	**14**	**2**	**36**	-1	**12**
06		**6**	**14**	**1**	**33**	-1	**14**
07		**4**	**14**	**2**	**34**	-4	**15**
08		**4**	**12**	**0**	**28**	-1	**9**
09		**4**	**10**	**2**	**36**	-2	**11**
10		**4**	**14**	**0**	**29**	0	**7**
11		**6**	**16**	**2**	**30**	-2	**14**
12		**6**	**14**	**7**	**37**	-1	**9**
13		**4**	**16**	**5**	**38**	0	**6**
14		**6**	**16**	**2**	**36**	0	**5**
15		**2**	**14**	**2**	**37**	0	**18**
16		**6**	**14**	**2**	**28**	0	**10**
17		**6**	**12**	**0**	**32**	-2	**12**
18		**4**	**16**	**4**	**37**	-1	**10**
19		**6**	**14**	**2**	**36**	-1	**11**
	Promedio:	5,05	13,89	1,89	34,00	-1,00	10,74

ANEXO Nº 2-B

Consolidado de los resultados del Pre-test y Post-test de conocimientos conceptuales, procedimenatles y actitudinales del grupo control:

Nº	APELLIDOS Y NOMBRES	Conocimiento Conceptual		Conocimiento procedimental		Conocimiento actitudinal	
		Pre-Test (Puntaje)	Post-Test (Puntaje)	Pre-Test (Suma)	Post-Test (Suma)	Pre-Test (Suma)	Post-Test (Suma)
01		4	10	0	34	0	6
02		4	6	2	27	-3	6
03		6	10	2	25	0	3
04		6	12	5	27	0	9
05		4	10	2	22	-2	4
06		6	6	3	29	-1	-5
07		4	14	2	28	-1	-6
08		8	10	2	27	-1	-1
09		6	12	2	27	0	2
10		4	10	0	24	-1	0
11		6	10	3	32	0	3
12		6	12	3	27	0	0
13		4	8	0	28	0	-3
14		4	14	5	31	-2	3
15		4	4	0	26	0	11
16		4	6	1	23	0	2
17		6	14	2	34	0	5
18		6	10	0	29	-2	-2
19		6	10	0	20	0	9
	Promedio	5,16	9,89	1,79	27,37	-0,68	2,42

ANEXO Nº 3

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

PRE-TEST DE CONOCIMIENTOS CONCEPTUALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

Indicaciones: 1) La prueba tiene una duración de 15 minutos.
2) Cada pregunta tiene 4 alternativas y encierre mediante un círculo su respuesta correcta.

1. ¿Cuál de ellos no es un modelo de ciclo de vida del desarrollo de software sobre base de datos?
   a. Incremental.
   b. Programación estructurada.
   c. Espiral.
   d. Madurez de capacidad (Capability Madurity Model: CMM).

2. La metodología incremental del desarrollo de software se basa en:
   a. Iniciar el análisis de una parte del área de negocios.
   b. Desarrollar los software a partir de las visiones de los usuarios y con la participación de los usuarios.
   c. En el diseño y análisis de los modelos de datos corporativos.
   d. Crear modelos de datos para cada arquitectura del sistema.

3. En un modelo de Prototipo:
   a. No es posible generar programa alguno.
   b. Se generan los programas de la aplicación y se prueban su funcionalidad.
   c. Se generan los programas de la aplicación en el lenguaje elegido para el modelo, pero no se crea la base de datos.
   d. No se puede generar programas de aplicación y probarlos.

4. Dadas las siguientes estructuras de transacciones:

Transacción "Paises"
PaiCod *
PaiNom

Transacción "Alumnos"
AlnCod *
AlnNom
PaiCod
PaiNom

¿Cuál será la estructura de la tabla "Paises" y "Alumnos" en la base de datos?

a.

Tabla: Paises	Tabla: Alumnos
PaiCod *	AlnCod * PaiCod *(Clave foránea)

b.

Tabla: Paises	Tabla: Alumnos
PaiCod *	AlnCod * PaiCod *(Clave foránea)

c.

Tabla: Paises	Tabla: Alumnos
PaiCod * PaiNom	AlnCod * AlnNom

d.

Tabla: Paises	Tabla: Alumnos
PaiCod * PaiNom	AlnCod * AlnNom PaiCod (Clave foránea) PaiNom

5. Dada las tablas "Alumnos", "Cursos" y la estructura de la transacción "Matrícula":

Tabla: Alumnos	Tabla: Cursos	Transacción "Matricula"
AlnCod * AlnNom	CurCod * CurNom CurCre	MatCod * MatFch AlnCod AlnNom (CurCod* CurNom CurCre) MatTot

¿Cuáles son los atributos inferidos a partir de las tablas "Alumnos" y "Cursos" en la transacción "Matrícula"?

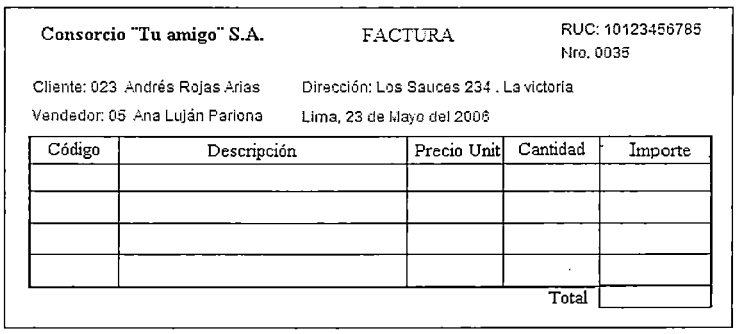
a. AlnCod, CurCod.
b. MatCod, MatFch, AlnCod, CurCod, CurCre, MatTot.
c. AlnNom, CurNom, CurCre.
d. MatCod, MatFch, CurCod, CurCre, MatTot.

6. El análisis de impacto de una aplicación sobre una base de datos, considera:
   a. La funcionalidad del producto final.
   b. Solamente los cambios estructurales que se deben realizar.
   c. Evaluar las modificaciones a realizarse.
   d. Cuantificar el esfuerzo ya realizado.

7. Los prototipos de una aplicación sobre base de datos sirven para:
   a. Entregar como producto final.
   b. Obtener los códigos de la aplicación.
   c. Crear la base de datos.
   d. Presentar el sistema que será implementado, y en ella se evalúa su funcionalidad.

8. Los atributos tipo fórmula contienen:
   a. Sólo variables.
   b. Variables y constantes.
   c. Sólo constantes.
   d. Sólo funciones.

9. La transacción:



Consorcio "Tu amigo" S.A. FACTURA RUC: 10123456785
Nro. 0035

Cliente: 023 Andrés Rojas Arias Dirección: Los Sauces 234 . La victoria
Vendedor: 05 Ana Luján Pariona Lima, 23 de Mayo del 2006

Código	Descripción	Precio Unit	Cantidad	Importe
			Total	

¿Cuántos niveles tiene la estructura de la transacción Factura?

a. 1 b. 2 c. 3 d. 4

10. Una base de datos:
    i: Representa algún aspecto del mundo real.
    ii: Es un conjunto de datos lógicamente coherente, con cierto significado inherente.
    iii: No es una colección aleatoria de datos.

    ¿Cuáles de las afirmaciones es verdadera?
    a. Sólo i.
    b. Sólo i y ii.
    c. Sólo iii.
    d. todas.

Gracias.

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

PRE-TEST DE CONOCIMIENTOS PROCEDIMENTALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

A continuación encontrarás una serie de preguntas de tu participación individual en el desarrollo de software sobre base de datos y contesta en forma sincera, marcando con una "X" en el casillero que mejor interprete tu opinión:

1. El desarrollo del software lo hacemos en poco tiempo:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

2. Prefiero trabajar en grupo que trabajar solo:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

3. Participo en el desarrollo de la aplicación junto con el usuario final:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

4. Puedo migrar software entre entornos Windows y Web automáticamente:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

5. Puedo elaborar reportes de diversos tipos:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

6. Puedo diseñar una base de datos estable:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

7. Estoy obligado a conocer varios lenguajes de programación y manejadores de base de datos:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

8. Soy capaz de normalizar cualquier tabla:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

9. Puedo fácilmente realizar el mantenimiento de la base de datos y programas, cuando existe cambios en la visión del usuario:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

10. Puedo construir prototipos rápidamente:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

Gracias.

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

PRE-TEST DE CONOCIMIENTOS ACTITUDINALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

A continuación encontrarás una serie de preguntas de tu participación individual en el Desarrollo de software sobre base de datos y contesta en forma sincera, marcando con una "X" en el casillero que mejor interprete tu opinión:

1. El desarrollo de la asignatura es:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

2. La metodología de desarrollo de software utilizada es:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

3. La metodología obliga a desarrollar software en pequeños grupos de estudiantes:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

4. La metodología de desarrollo de software utilizada en clase, ha permitido el desarrollo de mis habilidades y destrezas:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

5. El conocimiento de varios lenguajes de programación y manejadores de base de datos, facilita el uso de la metodología de desarrollo de software sobre base de datos:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

6. La metodología de desarrollo de software utilizada en clase, fomenta el autoaprendizaje convirtiendo a los alumnos en agentes de su propia educación.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

7. La metodología de desarrollo de software utilizada en clase, permite la adquisición de buenos hábitos de trabajo en equipo.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

8. La metodología de desarrollo de software utilizada en la asignatura, permite tener contacto directo con situaciones problemáticas de la vida real.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

9. La metodología de desarrollo de software utilizada en la asignatura, permite despertar el mayor interés y mayor actividad en el estudiante.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

10. La metodología de desarrollo de software utilizada en la asignatura, nos permite innovar en el desarrollo de software.
    ( ) muy bueno
    ( ) bueno
    ( ) ni bueno, ni malo
    ( ) malo
    ( ) muy malo

Gracias.

ANEXO Nº 4

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

POST-TEST DE CONOCIMIENTOS CONCEPTUALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

Indicaciones: 1) La prueba tiene una duración de 20 minutos.
2) Cada pregunta tiene 4 alternativas y encierre mediante un círculo su respuesta correcta.

1. En un modelo de prototipo:
   a. No es posible generar programa alguno.
   b. Se generan los programas de la aplicación y se prueban su funcionalidad.
   c. Se generan los programas de la aplicación en el lenguaje elegido para el modelo, pero no se crea la base de datos.
   d. No se puede generar programas de aplicación y probarlos.

2. La metodología incremental del desarrollo de software se basa en:
   a. Iniciar el análisis de una parte del área de negocios.
   b. Desarrollar los software a partir de las visiones de los usuarios y con la participación de los usuarios.
   c. En el diseño y análisis de los modelos de datos corporativos.
   d. Crear modelos de datos para cada arquitectura del sistema.

3. Una base de datos:
   i: Representa algún aspecto del mundo real.
   ii: Es un conjunto de datos lógicamente coherente, con cierto significado inherente.
   iii: No es una colección aleatoria de datos.
   ¿Cuáles de las afirmaciones es verdadera?
   a. Sólo i.
   b. Sólo i y ii.
   c. Sólo iii.
   d. todas.

4. Dada las tablas "Alumnos", "Cursos" y la estructura de la transacción "Matrícula":

Tabla: Alumnos
AlnCod *
AlnNom

Tabla: Cursos
CurCod *
CurNom
CurCre

Transacción "Matricula"
MatCod *
MatFch
AlnCod
AlnNom
(CurCod*
CurNom
CurCre)
MatTot

¿Cuáles son los atributos inferidos a partir de las tablas "Alumnos" y "Cursos" en la transacción "Matrícula"?

a. AlnCod, CurCod.
b. MatCod, MatFch, AlnCod, CurCod, CurCre, MatTot.
c. AlnNom, CurNom, CurCre.
d. MatCod, MatFch, CurCod, CurCre, MatTot.

5. Los prototipos de una aplicación sobre base de datos sirven para:

a. Entregar como producto final.
b. Obtener los códigos de la aplicación.
c. Crear la base de datos.
d. Presentar el sistema que será implementado, y en ella se evalúa su funcionalidad.

6. El análisis de impacto de una aplicación sobre una base de datos, considera:

a. La funcionalidad del producto final.
b. Solamente los cambios estructurales que se deben realizar.
c. Evaluar las modificaciones a realizarse.
d. Cuantificar el esfuerzo ya realizado.

7. La transacción:

Consorcio "Tu amigo" S.A. FACTURA RUC: 10123456785
Nro. 0035

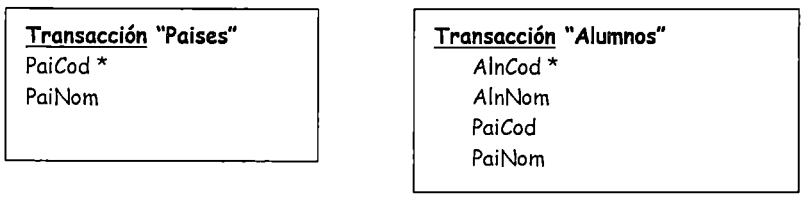
Cliente: 023 Andrés Rojas Arias Dirección: Los Sauces 234 , La victoria
Vendedor: 05 Ana Luján Pariona Lima, 23 de Mayo del 2006

Código	Descripción	Precio Unit	Cantidad	Importe
			Total	

¿Cuántos niveles tiene la estructura de la transacción Factura?

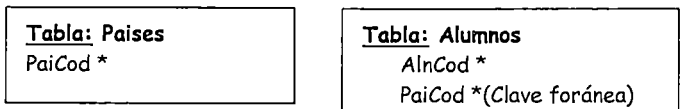
a. 1 b. 2 c. 3 d. 4

8. Dadas las siguientes estructuras de transacciones:

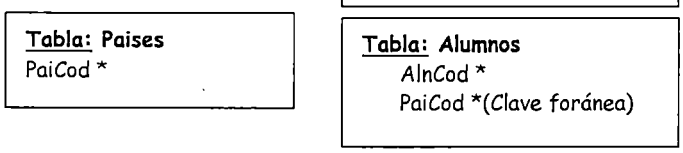


¿Cuál será la estructura de la tabla "Paises" y "Alumnos" en la base de datos?

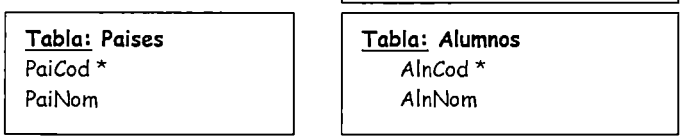
a.



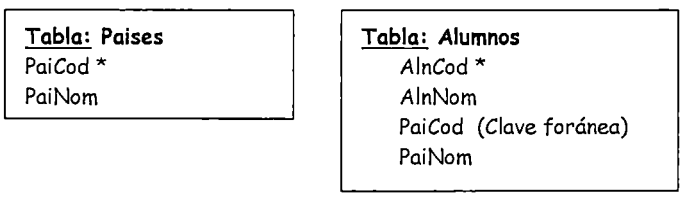
b.



c.



d.



9. ¿Cuál de ellos no es un modelo de ciclo de vida del desarrollo de software sobre base de datos?
   a. Incremental.
   b. Programación estructurada.
   c. Espiral.
   d. Madurez de capacidad (Capability Madurity Model: CMM).

10. Los atributos tipo fórmula contienen:
    a. Sólo variables.
    b. Variables y constantes.
    c. Sólo constantes.
    d. Sólo funciones.

Gracias.

UNIVERSIDAD NACIONAL DE EDUCACIÓN

Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

POST-TEST DE CONOCIMIENTOS PROCEDIMENTALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

A continuación encontrarás una serie de preguntas de tu participación individual en el desarrollo de software sobre base de datos y contesta en forma sincera, marcando con una "X" en el casillero que mejor interprete tu opinión:

1. El desarrollo del software lo hacemos en poco tiempo:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

2. Prefiero trabajar en grupo que trabajar solo:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

3. Participo en el desarrollo de la aplicación junto con el usuario final:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

4. Puedo migrar software entre entornos Windows y Web automáticamente:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

5. Puedo elaborar reportes de diversos tipos:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

6. Puedo diseñar una base de datos estable:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

7. Estoy obligado a conocer varios lenguajes de programación y manejadores de base de datos:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

8. Soy capaz de normalizar cualquier tabla:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

9. Puedo fácilmente realizar el mantenimiento de la base de datos y programas, cuando existe cambios en la visión del usuario:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

10. Puedo construir prototipos rápidamente:
   ( ) siempre
   ( ) casi siempre
   ( ) a veces
   ( ) casi nunca
   ( ) nunca

Gracias

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

POST-TEST DE CONOCIMIENTOS ACTITUDINALES DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS

Nombre del alumno: Fecha:

A continuación encontrarás una serie de preguntas de tu participación individual en el Desarrollo de software sobre base de datos y contesta en forma sincera, marcando con una "X" en el casillero que mejor interprete tu opinión:

1. El desarrollo de la asignatura es:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

2. La metodología de desarrollo de software utilizada es:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

3. La metodología obliga a desarrollar software en pequeños grupos de estudiantes:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

4. La metodología de desarrollo de software utilizada en clase, ha permitido el desarrollo de mis habilidades y destrezas:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

5. El conocimiento de varios lenguajes de programación y manejadores de base de datos, facilita el uso de la metodología de desarrollo de software sobre base de datos:
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

6. La metodología de desarrollo de software utilizada en clase, fomenta el autoaprendizaje convirtiendo a los alumnos en agentes de su propia educación.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

7. La metodología de desarrollo de software utilizada en clase, permite la adquisición de buenos hábitos de trabajo en equipo.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

8. La metodología de desarrollo de software utilizada en la asignatura, permite tener contacto directo con situaciones problemáticas de la vida real.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

9. La metodología de desarrollo de software utilizada en la asignatura, permite despertar el mayor interés y mayor actividad en el estudiante.
   ( ) muy bueno
   ( ) bueno
   ( ) ni bueno, ni malo
   ( ) malo
   ( ) muy malo

10. La metodología de desarrollo de software utilizada en la asignatura, nos permite innovar en el desarrollo de software.
    ( ) muy bueno
    ( ) bueno
    ( ) ni bueno, ni malo
    ( ) malo
    ( ) muy malo

Gracias

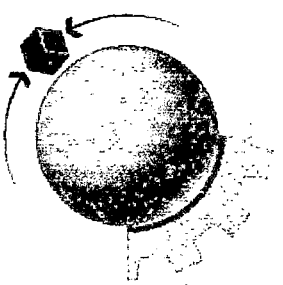
ANEXO Nº 5

UNIVERSIDAD NACIONAL DE EDUCACIÓN
ENRIQUE GUZMÁN Y VALLE
Alma Máter del Magisterio Nacional

FACULTAD DE CIENCIAS

Curso: Base de datos

Metodología GeneXus de Gonda
(desarrollo de software sobre base de datos)



GENEXUS™
THE FIRST INTELLIGENT TOOL

Prof. Florencio Flores Ccanto

La Cantuta
2006

ÍNDICE

Introducción

El presente módulo es una introducción al desarrollo de software sobre base de datos, mediante la aplicación de la metodología GeneXus de Gonda. Está dirigido a los alumnos de la asignatura Base de datos.

GeneXus es una herramienta para el desarrollo de software sobre bases de datos. Su objetivo es ayudar a los estudiantes a analizar las áreas del negocio e implementar software en el menor tiempo y con la mejor calidad posible.

A grandes rasgos, el desarrollo de software implica tareas de análisis, diseño e implementación. El camino de la metodología GeneXus de Gonda para alcanzar el objetivo anterior es liberar a las estudiantes de las tareas automatizables (por ejemplo, la implementación), permitiéndoles así concentrarse en las tareas realmente difíciles y no automatizables (comprender el problema del usuario).

La metodología GeneXus de Gonda desde el punto de vista teórico, tiene algunos puntos de contacto con las metodologías tradicionales, pero aporta enfoques bastante diferentes en otros.

Con la presente metodología el estudiante estará la mayor parte del tiempo realizando tareas de análisis y GeneXus se encargará de las tareas de diseño e implementación (por ejemplo, normalización de la base de datos, creación de la base de datos, generación de programas, etc.). El punto de partida de la metodología GeneXus es describir las visiones de los usuarios para modelar el sistema. A partir de este modelo, GeneXus construye el soporte computacional (base de datos y programas) en forma totalmente automática.

Para presentar estos nuevos conceptos, y a los efectos de no realizar una presentación demasiado abstracta del tema, se ha elegido un software sobre base de datos que se irá desarrollando a través de los distintos capítulos. El primer capítulo presenta la aplicación y los aspectos iniciales de un desarrollo con GeneXus y los siguientes capítulos presentan algunos de los objetos con los que GeneXus cuenta para describir el software, los que serán utilizados para desarrollar la aplicación de ejemplo. Así, el segundo capítulo trata sobre el objeto Transacción, el tercero sobre el objeto Reporte, el quinto sobre sobre el Work Panel.

CAPÍTULO I

1. CASO DE ESTUDIO: SISTEMA DE COMPRAS PARA UNA CADENA DE FARMACIAS.

El caso práctico de estudio considerado para el trabajo de los estudiantes en el laboratorio correspondiente al curso, en la que se plantea el requerimiento de una cadena de farmacias en la que se desea implementar un sistema de compras que permita a los analistas de compras realizar dicha tarea con la mayor cantidad de información posible. La función de un analista de compras es decidir los pedidos que se deben efectuar a los proveedores, de los distintos artículos.

1.1. Funciones del software

Se desea que el analista de compras utilice el computador para definir los pedidos de artículos a los distintos proveedores. Una vez que el pedido esté hecho y aprobado, se quiere que el computador emita las órdenes de compra. En el momento de hacer un pedido es necesario hacer varias consultas; por ejemplo cuánto hay en stock del artículo, cuál fue el precio de la última compra, etc. Los siguientes puntos describen los pasos que se siguen para el desarrollo del software sobre base de datos.

Definir el objetivo

No se debe olvidar que los computadores son meras herramientas. Los usuarios de los sistemas tienen objetivos específicos. Ellos esperan que la aplicación los ayude a alcanzarlos más rápidamente, más fácilmente, o a un menor costo. Es parte del trabajo de análisis del requerimiento el conocer esos propósitos y saber por medio de qué actividades los usuarios quieren alcanzarlos.

Estos objetivos deben ser expresados en pocas palabras y ser conocidos por todas las personas involucradas con el sistema.

En el ejemplo, algunos de los propósitos posibles son:

- "El sistema de compras debe disminuir la burocracia existente para la formulación de un pedido."
- El sistema de compras debe asistir a usuarios no entrenados en la formulación de pedidos de tal manera que su desempeño se asemeje al de un experto."
- "El sistema de compras debe permitir la disminución del stock existente en las farmacias."

De todos los objetivos posibles se debe elegir uno como el objetivo principal o prioritario. Esto es muy importante para el futuro diseño de la aplicación. Para nuestro caso de estudio elegiremos el primer objetivo, dado que en la situación real el analista de compras no contaba con toda la información que necesitaba, por lo cual debía consultar una serie de planillas manuales y llamar por teléfono a los empleados del depósito para que realizaran un conteo manual del stock. No se debe confundir el objetivo de la aplicación -el QUE- con la funcionalidad de la misma –COMO se alcanzará el objetivo.

1.2. Definir el equipo de trabajo

Se debe definir cuál será el equipo de alumnos encargado de la implementación del sistema. Dicho equipo debe tener como mínimo dos alumnos y como máximo tres alumnos. Como el software con la aplicación de GeneXus de Gonda se desarrolla de una manera incremental, es muy importante la participación de los usuarios en todas las etapas del desarrollo. Se recomienda tener un usuario disponible para la prueba de los prototipos.

1.3. Definir el alcance del software

Luego de un estudio primario se debe decidir cuál será el alcance de la aplicación para poder cumplir con el objetivo. Para ello se recomienda seguir el Principio de "esencialidad":

"Solo lo imprescindible, pero todo lo imprescindible"

En el caso de estudio, una vez que una orden de compra es enviada a un proveedor, se debe controlar cómo y cuándo se deben entregar efectivamente los productos. Sin embargo vemos que esto no es imprescindible para cumplir el objetivo del software y por lo tanto no será tratado.

Mantener el diseño simple

Se debe planificar pensando en un proceso de diseño y desarrollo incremental. Comenzando por pequeños pasos y verificando la evolución del modelo frecuentemente con el usuario.

1.4. Orientarse a los datos

En esencia, un software sobre base de datos es un conjunto de mecanismos para realizar ciertos procesos sobre ciertos datos. Por lo tanto, en el análisis de datos del software, se puede poner mayor énfasis en los procesos o en los datos.

Cuando el diseño se basa en los datos se obtienen las siguientes ventajas:

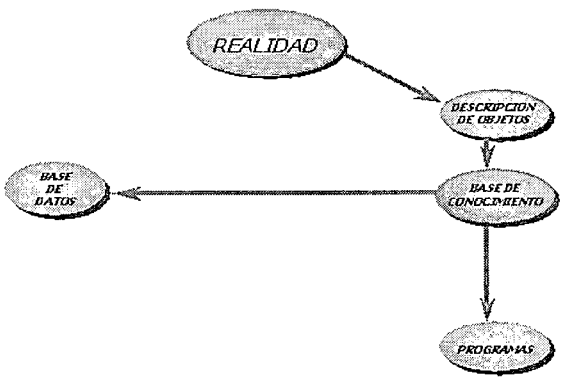
- Más estabilidad. Los datos tienden a ser más estables que los procesos y en consecuencia la aplicación será más fácil de mantener.
- Facilidad de integración con otras software. Difícilmente el software sea totalmente independiente de las otras dentro de una organización. La mejor forma de integrarlas

es tener en cuenta los datos que comparten. Por lo tanto, orientarse a los datos parecería ser más beneficioso. La pregunta natural que surge es, entonces, ¿cómo analizar los datos?. La respuesta de la metodología GeneXus de Gonda es: analizar directamente los datos que el usuario conoce, sin importar como se implementarán en el computador. De esta manera se alcanzan dos objetivos importantes: el análisis se concentra en hechos objetivos, y éste puede ser evaluado directamente por los usuarios, utilizando la facilidad de la prototipación de GeneXus.

1.5. Metodología GeneXus de Gonda

Para comenzar el desarrollo de un software con la metodolgía GeneXus, el primer paso consiste en crear un nuevo proyecto o base de conocimiento. El siguiente paso es describir las visiones de los usuarios, a partir de las cuáles GeneXus captura y sistematiza el conocimiento. Para ello, se deben identificar los objetos de la realidad y pasar a definirlos mediante objetos GeneXus.

Con la definición de estos objetos, GeneXus puede extraer el conocimiento, y diseñar la base de datos y los programas del software en forma totalmente automática.



Si un objeto de la realidad cambia, se identifican nuevas o diferentes características acerca del mismo, o si se encuentran objetos aún no modelados, el analista GeneXus debe reflejar dichos cambios en los objetos GeneXus que corresponda, y la herramienta se encargará automáticamente de realizar las modificaciones necesarias tanto en la base de datos como en los programas asociados.

La metodología GeneXus es una metodología incremental, pues parte de la base de que la construcción de un sistema se realiza mediante aproximaciones sucesivas. En cada momento el analista GeneXus define el conocimiento que tiene y luego cuando pasa a tener más -o simplemente diferente- conocimiento, GeneXus se ocupará de hacer automáticamente todas las adaptaciones en la base de datos y programas. Si GeneXus no fuera capaz de realizar automáticamente estas modificaciones conforme se realicen

cambios que así lo requieran, el desarrollo incremental sería inviable.

GeneXus puede generar la aplicación en diversas plataformas. Así, por ejemplo, se puede implementar la aplicación para un DBMS como SQL Server, MySQL, Oracle, Informix, etc., en un lenguaje como .NET, C/SQL, Visual Basic, Visual FoxPro, J A V A , etc. Las tablas, índices y programas serán generados por GeneXus en la plataforma elegida, y el analista no tendrá que contar con conocimientos profundos del lenguaje ni del DBMS.

1.6. Objetos GeneXus

A continuación se describen los objetos GeneXus más importantes (no siendo los únicos):

1.6.1. Transacciones

Permiten definir objetos de la realidad –reales o imaginarios- que el usuario manipula (ej: analistas de compras, artículos, proveedores, pedidos, etc.). Son los primeros objetos en definirse, ya que a través de las transacciones, GeneXus infiere el diseño de la base de datos.

Cada transacción tiene asociada una pantalla para ambiente windows y otra para ambiente web, que permiten al usuario dar altas, bajas y modificaciones en forma interactiva a la base de datos. El analista GeneXus decidirá si trabajar en ambiente windows, web, o ambos.

1.6.2. Reportes

Permiten recuperar información de la base de datos, y desplegarla ya sea en la pantalla, en un archivo o impresa en papel. Son los típicos listados o informes. No permiten actualizar la información de la base de datos.

1.6.3. Work Panels

Permiten al usuario realizar interactivamente consultas a la base de datos, a través de una pantalla, en ambiente windows. Ejemplo: un work panel permite al usuario ingresar un rango de caracteres, y muestra a continuación todos los clientes cuyos nombres se encuentran dentro del rango. Son objetos muy flexibles que se prestan para múltiples usos. No permiten la actualización de la base de datos, sino solo consultarla.

1.6.4. Otros objetos.

Existen otros objetos como los procedimientos y los Web Panel.

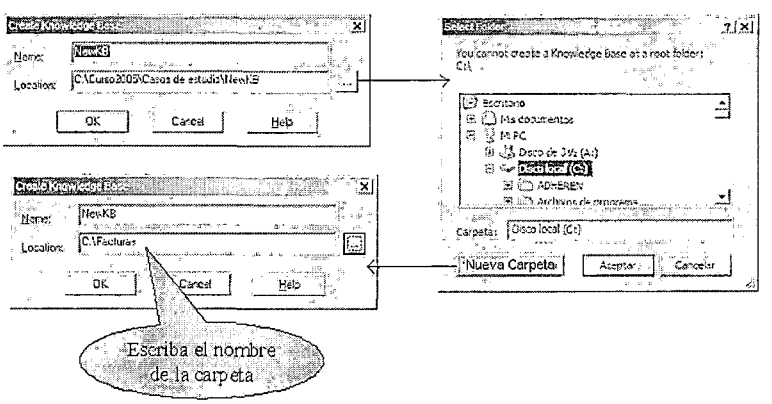
Universidad Nacional de Educación
Enrique Guzmán y Valle
Alma Máter del magisterio Nacional

Facultad de Ciencias

Práctica

Tema: Creacion de la base de conocimientos

1. Creacion de una base de conocimientos de GeneXus (Facturas).
   Creación de nuevo folder:
   - File/New Knowledge Base.../ ... /
   - Seleccionar la unidad C:/
   - Crear nueva Carpeta (Colocar el nombre)



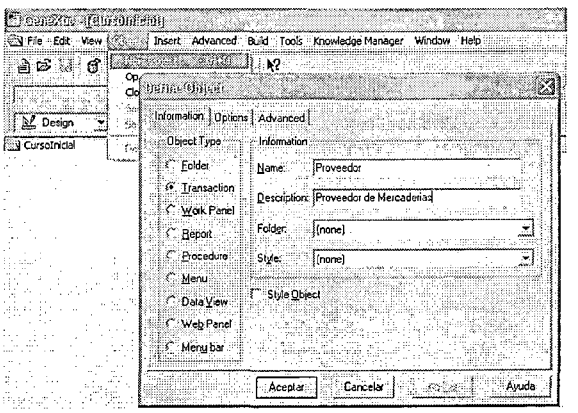
   - Seleccionar Ok y Aceptar.

Esto permite crear la nueva base de conocimiento de GeneXus, con el nombre que se ingresa.

2. Diseño de transacciones. Las transacciones tienen distintos elementos que conforman las en distintas vistas (Tabs).

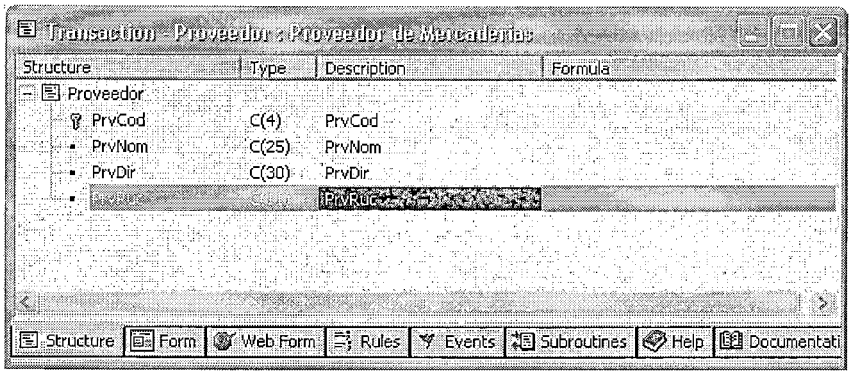
Es la sección donde se ingresa la estructura de la transacción. Se debe detallar los atributos pertenecientes a la transacción, niveles que la componen y clave para cada uno de los niveles especificados.

a. Crear una nueva transacción: Object/ NewObject...

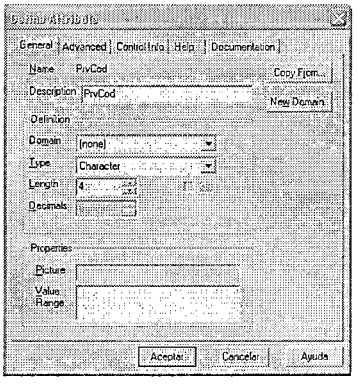


En esta ventana se debe ingresar el nombre de la transacción y la descripción de la misma.

b. Crear la estructura de la transacción Proveedor (digitar los nombres de los atributos). Luego, guardar mediante la herramienta (💾):



Atributos. Para cada atributo se deba ingresar la información de sus tipos de datos y sus tamaños, tal como muestra la ventana siguiente:



Similarmente, ingresar todas las propiedades de los demás atributos.

CAPÍTULO II

2. OBJETO TRANSACCIÓN

El análisis del software comienza con el diseño de las transacciones. Las transacciones permiten definir los objetos de la realidad. Para identificar cuáles transacciones deben crearse, se recomienda prestar atención a los sustantivos que el usuario menciona cuando describe la realidad.

Como el objetivos es el análisis de la realidad y la consecuente creación de la base de datos normalizada, por cada transacción se generarán programas para ambiente windows y para ambiente web, para permitir dar altas, bajas y modificaciones en forma interactiva a la base de datos. Por lo tanto, se determina una estructura para cada tabla, al igual que el resto de los objetos GeneXus.

Es importante tener en cuenta que todo el proceso de desarrollo es incremental y por lo tanto no es necesario definir en esta etapa todas las transacciones y cada una de ellas en su mayor detalle. Por el contrario, lo importante aquí es reconocer las más relevantes y para cada una de ellas identificar y definir su estructura.

Para poder definir cuáles son las transacciones se recomienda estudiar cuáles son los objetos -reales o imaginarios- que el usuario manipula. Es posible encontrar la mayor parte de las transacciones a partir de:

La descripción del software. Cuando un usuario describe su área de negocios se pueden determinar muchas transacciones si se presta atención a los sustantivos que utiliza. En el caso de estudio:

Analistas de compras
Pedidos
Proveedores
Artículos
Órdenes de Compra

Formularios existentes. Por cada formulario que se utilice en el área de negocios es casi seguro que existirá una transacción para el ingreso del mismo.

2.1. Elementos de una transacción.

Para cada transacción se definen, entre otros, los siguientes elementos:

Estructura

Permite definir los atributos (campos) que componen la transacción y la relación entre ellos. A

partir de la estructura de las transacciones, GeneXus inferirá el diseño de la base de datos: tablas, claves, índices, etc.

Form GUI-Windows

Cada transacción contiene un form (pantalla) GUI-Windows (Graphical User Interface Windows) mediante el cuál se realizarán las altas, bajas y modificaciones en ambiente Windows.

Form Web

Cada transacción contiene un form (pantalla) Web mediante el cuál se realizarán las altas, bajas y modificaciones en ambiente Web.

Reglas

Las reglas permiten definir el comportamiento particular de las transacciones. Por ejemplo, permiten definir valores por defecto para los atributos, definir chequeos sobre los datos, etc.

Eventos

Las transacciones soportan la programación orientada a eventos. Este tipo de programación permite definir código ocioso, que se activa en respuesta a ciertas acciones provocadas por el usuario o por el sistema.

Subrutinas

Permiten definir rutinas locales a la transacción, que se ejecutan al ser invocadas desde la propia transacción.

Propiedades

Son características a ser configuradas para definir ciertos detalles referentes al comportamiento general de la transacción.

Ayuda

Permite la inclusión de texto de ayuda, para ser consultado por los usuarios en tiempo de ejecución de la transacción.

Documentación

Permite la inclusión de texto técnico, para ser utilizado como documentación del sistema.

Estructura (Structure)

La estructura define qué atributos (campos) integran la transacción y cómo están relacionados.

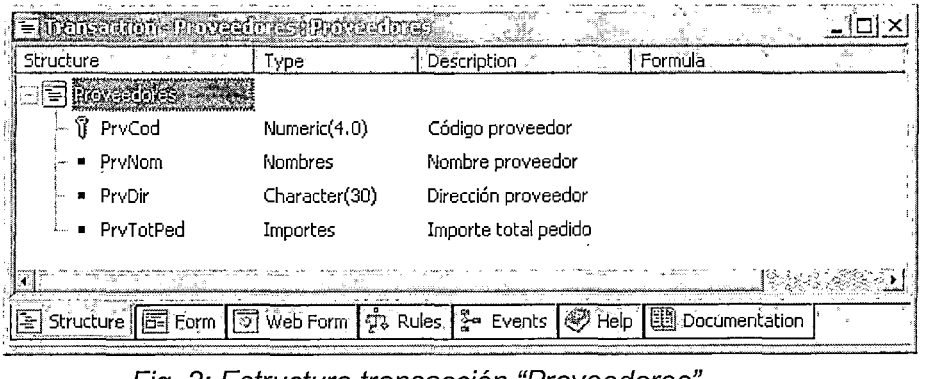


Fig. 2: Estructura transacción "Proveedores"

En el caso de estudio, la transacción "Proveedores" posee los siguientes atributos, que componen la información que se quiere tener de un proveedor:

PrvCod	Código de proveedor
PrvNom	Nombre del proveedor
PrvDir	Dirección del proveedor
PrvTotPed	Importe total pedido al proveedor

A partir de esta estructura GeneXus creará una tabla:

PROVEEDORES	*PrvCod*	*PrvNom*	*PrvDir*	*PrvTotPed*

y creará para la transacción un formulario (form GUI) por defecto:

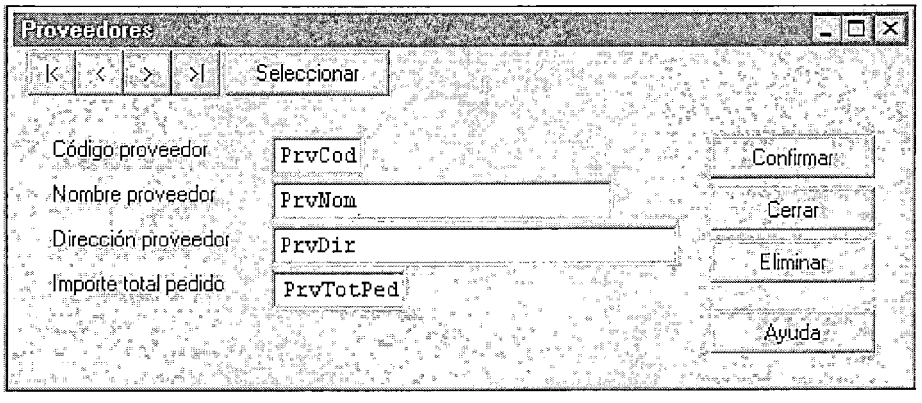


Fig. 3: Form GUI de la transacción "Proveedores"

así como uno Web:

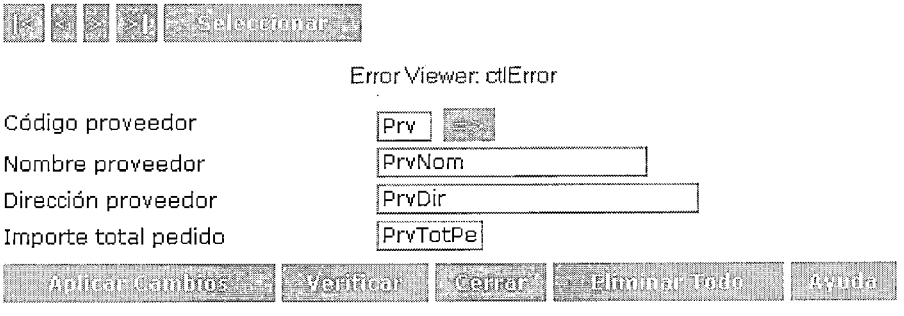


Fig. 4: Form Web de la transacción "Proveedores"

El analista del software podrá luego modificados.

A través de este form (ya sea del GUI o del Web, esto dependerá del ambiente para el que se genere la aplicación), el usuario podrá ingresar nuevos proveedores, que serán dados de alta como nuevos registros en la tabla asociada. También podrá modificar los datos de un proveedor dado. Para ello, la transacción "Proveedores" tiene encapsulada la lógica que le permite acceder a la tabla asociada, recuperar el registro solicitado, y mostrar sus datos en el form, de manera tal que el usuario pueda verlos y modificarlos. Al realizar una modificación sobre los datos que aparecen en el form, por ejemplo cambiar el nombre del

proveedor, la transacción en el momento adecuado, deberá realizar ese cambio en el registro físico correspondiente. Algo similar ocurre con respecto a las eliminaciones de registros.

El analista GeneXus no deberá programar nada de esto, pues ya está implícito en la lógica de toda transacción. La codificación la realiza GeneXus, y es absolutamente transparente para el analista, quien solo debe encargarse de aspectos de alto nivel que hacen al objeto.

Al representar transacciones en GeneXus estamos:

- **Explícitamente:** definiendo la interfaz con el usuario en la presentación y captura de los datos
- **Implícitamente:** definiendo la relación entre los datos (tablas, índices, etc.)

Atributos

Cada atributo tiene propiedades asociadas, que pueden editarse para ser modificadas. La siguiente pantalla corresponde a las propiedades del atributo PrvCod:

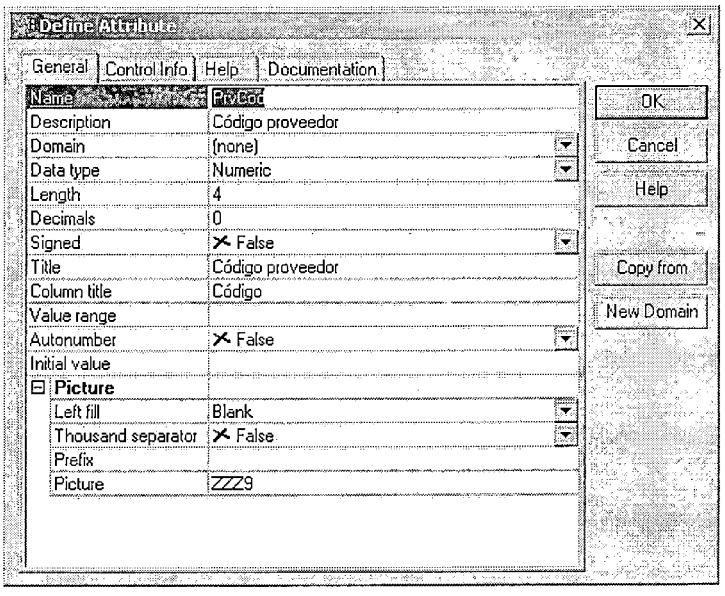


Fig. 5: Diálogo "Define Attribute"

1. **General.** Permite configurar características generales del atributo, como:

   Name: Es el nombre del atributo. Se utiliza para identificarlo.

   Description: Es un "nombre largo" o descripción ampliada del atributo. Debe identificarlo bien, con independencia del contexto, y principalmente debe ser entendible por el usuario final. Debe ser un identificador global del atributo, es decir, no se debe asignar a dos atributos de la aplicación la misma descripción.

   Title: La descripción que se ingrese aquí, será colocada por defecto al lado del atributo cada vez que éste se utilice en salidas "planas".

   Column Title: La descripción que se ingrese aquí será colocada por defecto como título del atributo, cada vez que se lo incluya como columna de un grid (grilla).

Domain: Permite asociar un dominio al atributo. En caso que el atributo no pertenezca a un dominio, se dejará el valor "(none)" en esta propiedad, y de esta forma las propiedades correspondientes al tipo de datos del atributo aparecerán habilitadas, para que el analista especifique sus valores.

Data Type: Permite indicar el tipo de datos del atributo. Aquí se podrá elegir uno de los tipos de datos soportados por GeneXus. Entre ellos se encuentran los tipos Numeric, Character, VarChar, Long VarChar, Date, DateTime, Blob.

Length: Permite indicar el largo del atributo. Si en la propiedad Data Type se indica que el atributo es numérico, entonces, se deberá tener en cuenta que el largo incluya las posiciones decimales, el punto decimal y el signo

Decimals: Si en la propiedad Data Type se indica que el atributo es numérico, se ofrecerá esta propiedad, para que se especifique la cantidad de decimales. El valor 0 en este campo, indicará que se trata de un entero.

Signed: Si en la propiedad Data Type se indica que el atributo es numérico, se ofrecerá esta propiedad, para que se indique si manejará signo, o no. El valor por defecto es "False", lo que indica que no se representarán valores negativos.

Value Range: Permite indicar un rango de valores válidos para el atributo.

1:20 30: - significa que los valores válidos son aquellos entre 1 y 20; y 30 o mayor.

1 2 3 4 - significa que los valores válidos son 1, 2, 3 y 4.

'S' 'N' - significa que los valores válidos son 'S' y 'N'.

Autonumber: Permite indicar que el atributo debe ser numerado automáticamente por el sistema. Solo aplica si el atributo es de tipo numérico y clave primaria de la tabla.

Initial Value: Cuando el atributo se está agregando a una tabla que ya existe y tiene registros cargados, esta propiedad permite especificar qué valor "inicial" se le dará a este atributo para los registros pre-existentes.

Picture: Permite indicar el formato de edición para la entrada y salida del atributo.

2. Control Info

Permite indicar el tipo de control que se desea asociar al atributo. Esta información es utilizada por GeneXus cuando crea los forms por defecto.

3. Help

A todo atributo puede asociársele una descripción larga para ayudar al usuario final en tiempo de ejecución. Si el usuario final solicita ayuda presionando F1 sobre el atributo, entonces esta descripción será desplegada.

4. Documentation

Permite definir documentación técnica del atributo, útil para otros desarrolladores.

2.2. Dominios

Es común cuando estamos definiendo la base de datos, tener atributos que comparten definiciones similares, y para los que no se puede establecer ninguna relación directa

El uso de dominios permite utilizar definiciones de atributos genéricos. Por ejemplo, en la transacción de proveedores tenemos el nombre, PrvNom, y más adelante vamos a definir el nombre del analista, entonces podemos definir un dominio Nombres de tipo character con largo 25 y asociarlo a estos atributos.

2.3. Atributos Clave

Es necesario definir cuál es el atributo o conjunto de atributos que identifican a la transacción, es decir, aquel o aquellos atributos cuyos valores son únicos. Si en la realidad que estamos modelando un proveedor se identifica por su código, entonces el atributo PrvCod será el identificador de la transacción "Proveedores". Esto queda expresado en GeneXus mediante el símbolo de llave a la izquierda del atributo en la estructura de la transacción, como puede observarse claramente en la fig.2.

Es común encontrar como notación para representar en papel la estructura de una transacción (tanto en libros, manuales de GeneXus, help, etc.) una lista de atributos, donde aquellos que constituyen el identificador aparecen sucedidos del símbolo asterisco (*). Así, por ejemplo, denotaremos a la estructura de la transacción "Proveedores" de la siguiente manera:

PrvCod*
PrvNom
PrvDir
PrvTotPed

Aquí estamos representando que el identificador de la transacción "Proveedores" es el atributo PrdCod, es decir, no podrán haber dos proveedores con el mismo código. En tiempo de ejecución, se realizará este chequeo de unicidad automáticamente.

Con respecto a los identificadores:

- Toda transacción debe tener un identificador.
- Los identificadores tienen valor desde un principio (por ejemplo cuando se crea un proveedor nuevo se debe saber cuál será su PrvCod)
- No cambian de valor. Por ejemplo al proveedor con código 123 no se lo puede cambiar para que tenga código 234.

2.4. Relación entre la estructura y el modelo de datos

GeneXus utiliza la estructura de la transacción para capturar el conocimiento necesario para definir cuál es el modelo de datos correspondiente.

En particular de la estructura anterior GeneXus infiere que:

- No existen dos proveedores con el mismo PrvCod.
- Para cada PrvCod existe solo un valor de PrvNom, PrvDir y PrvTotPed.

y con esta información va construyendo el modelo de datos.

El nombre de la tabla y el del índice son asignados automáticamente por GeneXus utilizando el nombre de la transacción, pero pueden ser modificados por el analista cuando así lo desee.

2.5. Otras transacciones simples del caso de estudio

Además de la transacción de proveedores, tendremos transacciones para representar y manejar la información relevante de los analistas de compras, de los artículos y de los pedidos.

Las estructuras de las dos primeras serán:

Analistas:

*AnlNro**	Número de analista
AnlNom	Nombre de analista

Artículos:

*ArtCod**	Código de artículo
ArtDsc	Descripción del articulo
ArtCnt	Cantidad en stock
ArtFchUltCmp	Fecha última compra
ArtPrcUltCmp	Precio última compra
ArtUnd	Unidad del artículo
ArtTam	Tamaño
ArtFlgDsp	Disponibilidad

que tendrán asociadas las tablas ANALISTAS y ARTICULOS respectivamente:

ANALISTAS	AnlNro	AnlNom

ARTICULOS	ArtCod	ArtDsc	ArtCnt	ArtFchUltCmp
	ArtPrcUltCmp	ArtUnd	ArtTam	ArtFlgDsp

Como podemos apreciar en estas transacciones, todos los atributos de la estructura están presentes en la tabla asociada. Sin embargo esto no es lo más frecuente, como veremos en breve, cuando definamos la transacción correspondiente a los pedidos.

2.6. Definición de la transacción de pedidos

Consideremos ahora la transacción para manejar la información relativa a los pedidos. El formulario preexistente de pedidos es:

Pedido :	1		Fecha :	02/02/01
Analista :	21	Juan Gómez		
Proveedor :	125	ABC Inc.		

Código	Descripción	Cantidad	Precio	Importe
321	Aspirinas	100	30	3000
567	Flogene	120	50	6000
			Total	9000

Los atributos que integran el cabezal de la transacción son (dejando momentáneamente de lado la información de los artículos del pedido):

*PedNro**	Número del pedido
PedFch	Fecha del pedido
AnlNro	
AnlNom	
PrvCod	
PrvNom	
PedTot	Total del pedido

donde PedNro es el identificador de un pedido.

2.7. Reglas para nombres de atributos

Se debe poner el mismo nombre al mismo atributo, en todas las transacciones en las que éste se encuentre. De esta forma, al nombre del proveedor se lo denomina PrvNom tanto en la transacción de proveedores como en la de pedidos.

Se deben poner nombres distintos a atributos conceptualmente distintos, aunque tengan dominios iguales. Por ejemplo, el nombre del proveedor y el nombre del analista tienen el mismo dominio (son del tipo Character de largo 25), pero se refieren a datos diferentes, por lo tanto se deben llamar diferente: PrvNom y AnlNom.

2.8. Integridad referencial en las transacciones

Cuando se define la estructura de una transacción no se está describiendo exactamente la estructura de una tabla, sino los datos que se necesitan en la pantalla (form), en las reglas o en los eventos de la transacción.

En el caso de estudio, si bien AnlNom aparece en la estructura de la transacción "Pedidos",

este atributo no pertenecerá a la tabla correspondiente a la transacción. Lo incluimos en la estructura pues queremos que esté presente en la pantalla (form) de "Pedidos".

La tabla asociada al cabezal del "Pedido" será:

PEDIDOS	PedNro	PedFch	AnlNro	PrvCod	PedTot

Esta tabla será creada automáticamente por GeneXus, junto con los siguientes índices:

- IPEDIDOS (PedNro) Índice por clave Primaria
- IPEDIDOS1 (AnlNro) Índice por clave Foránea
- IPEDIDOS2 (PrvCod) Índice por clave Foránea

Observar que PrvNom no se encuentra en la tabla PEDIDOS, así como tampoco AnlNom. Diremos que GeneXus normalizó y que existe una relación entre la tabla PROVEEDORES, que tiene a PrvCod como clave primaria, y la tabla PEDIDOS que lo tiene como clave foránea. La relación, que llamaremos de integridad referencial, determina que:

- Para insertar o actualizar un registro en la tabla PEDIDOS debe existir el PrvCod correspondiente en la tabla PROVEEDORES.
- Cuando se elimina un registro de la tabla PROVEEDORES no deben haber registros en la tabla PEDIDOS con el valor del PrvCod a eliminar.

2.9. Niveles de una transacción

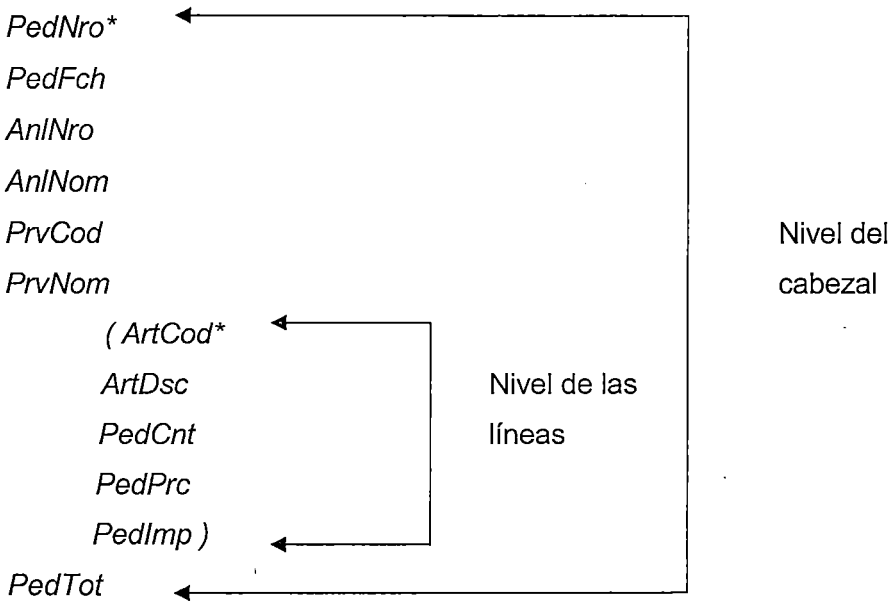
Volviendo al ejemplo, para terminar de diseñar la transacción "Pedidos" se debe incorporar en la estructura la información relativa a los artículos del pedido (las líneas). Si definimos la estructura de la siguiente manera:

*PedNro**
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
ArtCod
ArtDsc
PedCnt Cantidad pedida
PedPrc Precio pedido
PedImp Importe por el artículo
PedTot

no estaremos modelando correctamente la realidad, ya que aquí se está representando que para cada pedido existe solo UN artículo. Sin embargo, si observamos el formulario preexistente,

vemos claramente que un pedido está compuesto de muchos artículos.

La estructura correcta es, pues:



donde el identificador es PedNro y para cada número de pedido existe solo una fecha, un número y nombre de analista, un código y nombre de proveedor, y un solo total, pero existen muchos artículos, con su descripción, cantidad pedida, precio e importe.

Así, la transacción "Pedidos" tiene dos niveles:

- PedNro, PedFch, AnlNro, AnlNom, PrvCod, PrvNom y PedTot pertenecen al primer nivel, y
- ArtCod, ArtDsc, PedCnt, PedPrc y PedImp pertenecen al segundo nivel.

En GeneXus el segundo nivel queda representado como se muestra en la siguiente figura:

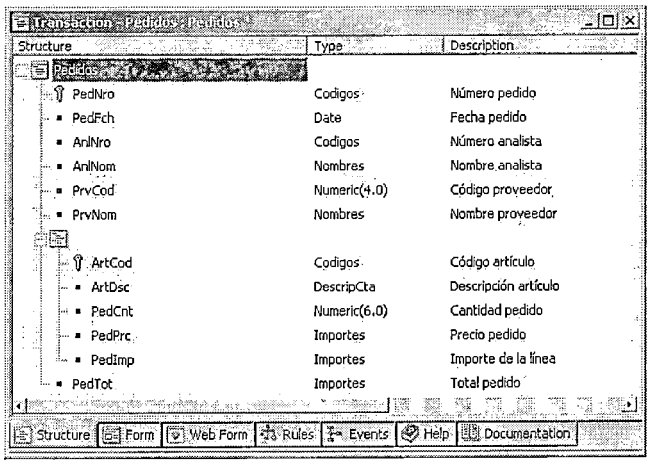


Fig. 6: Estructura de "Pedidos" en GeneXus

Cada nivel de la transacción tiene un identificador, es decir, un conjunto de atributos que determinan la unicidad de los datos del nivel.

Así, en el pedido se definió que ArtCod es el identificador del segundo nivel, lo cual significa que dentro de cada pedido no pueden existir dos líneas con el mismo valor de ArtCod.

En consecuencia, el siguiente pedido no se podría ingresar:

Pedido : 1			Fecha : 02/02/01	
Analista : 21 Juan Gómez				
Proveedor : 125 ABC Inc.				
Código	Descripción	Cantidad	Precio	Importe
321	Aspirinas	100	30	3000
321	Aspirinas	120	50	6000
			Total	9000

porque en él existen dos líneas del pedido con el mismo ArtCod.

Cuantas más reglas de la realidad se puedan reflejar en la estructura, menor será la cantidad de procesos que se deben implementar, ganando así en simplicidad y flexibilidad.

2.10. Tipos de relación entre los objetos

Cuando los usuarios están describiendo el software, es común preguntarles qué tipo de relación existe entre los distintos objetos. Por ejemplo, cuál es la relación entre los pedidos y los proveedores:

- Un proveedor tiene muchos pedidos
- Un pedido tiene un solo proveedor

A una relación que cumple con los requerimientos anteriores le llamamos N-1 (y leemos: "Pedidos y Proveedores tienen una relación N a 1").

De la misma forma, si recurrimos a la realidad, obtenemos que la relación entre los pedidos y los artículos es la siguiente:

- Un pedido tiene muchos artículos.
- Un artículo puede estar en muchos pedidos.

En este caso decimos que la relación entre pedidos y artículos es N-M.

2.11. Forms

Para cada transacción, GeneXus crea un form GUI-Windows y un form Web, los cuales serán la interfaz con el usuario, en ambiente windows y web respectivamente. Ambos forms son creados por defecto por GeneXus al momento de salvar la estructura de la transacción, y contienen todos los atributos incluidos en la misma, con sus respectivas descripciones, además de algunos botones.

Si bien son creados por defecto, es posible modificarlos para dejarlos más vistosos, cambiar por ejemplo controles de tipo edit a otros tipos de controles, agregar y/o quitar botones, etc.

El Form GUI Windows por defecto para la transacción "Proveedores" es:

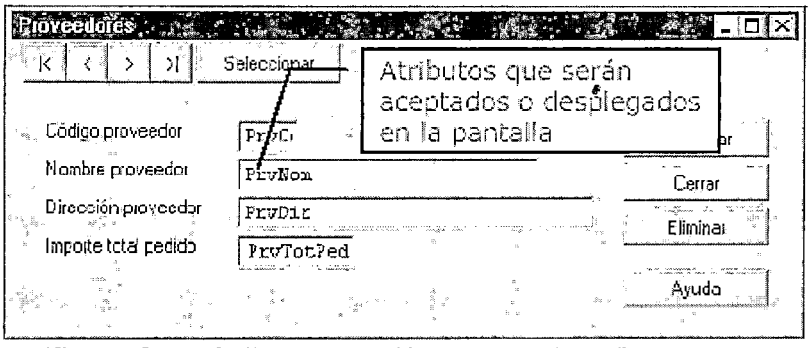


Fig. 7: Form GUI - transacción "Proveedores"

Utilizando esta pantalla el usuario final puede ingresar, modificar y eliminar proveedores. Para ello la pantalla tiene un modo, que indica cuál es la operación que se está realizando (Insert, Update, Delete o Display) y el usuario puede pasarse de un modo a otro sin tener que abandonar la pantalla.

GeneXus genera varios tipos de diálogo para las transacciones. Entre ellos: diálogo campo a campo y full-screen. El tipo de diálogo utilizado dependerá tanto del ambiente (Windows o Web), como de la plataforma de implementación elegida.

Diálogo campo a campo

En este diálogo, cada vez que se digita un valor en un campo, se controla inmediatamente su validez. Por ejemplo, si se trata de un atributo de tipo Date, una vez que el usuario ingresa un valor en el mismo, y abandona el campo, inmediatamente se controlará que el valor corresponda a una fecha válida.

Diálogo full-screen

En este diálogo a pantalla completa, primero se aceptan todos los campos de la pantalla y luego se realizan todas las validaciones. En ambiente Web no hay otra opción que trabajar de esta forma, ya que como los datos viajan del navegador a un servidor Web, y viceversa, es inviable trabajar con el diálogo campo a campo, por los factores tiempo y tráfico.

Botones de posicionamiento

Tanto los forms GUI-Win como Web son inicializados por defecto con botones de posicionamiento.

Si observamos el form GUI de la transacción "Proveedores" (fig.7), vemos en el borde superior izquierdo botones que permiten seleccionar el primer proveedor: [>] , el siguiente a partir del que se está mostrando en pant[<] , elegir uno en particu[Seleccionar] , etc. También en

el form Web de la figura 4 vemos estos controles, aunque con otro aspecto.

2.12. Atributos de entrada y atributos de salida

Consideremos nuevamente la transacción "Pedidos". El form GUI por defecto es:

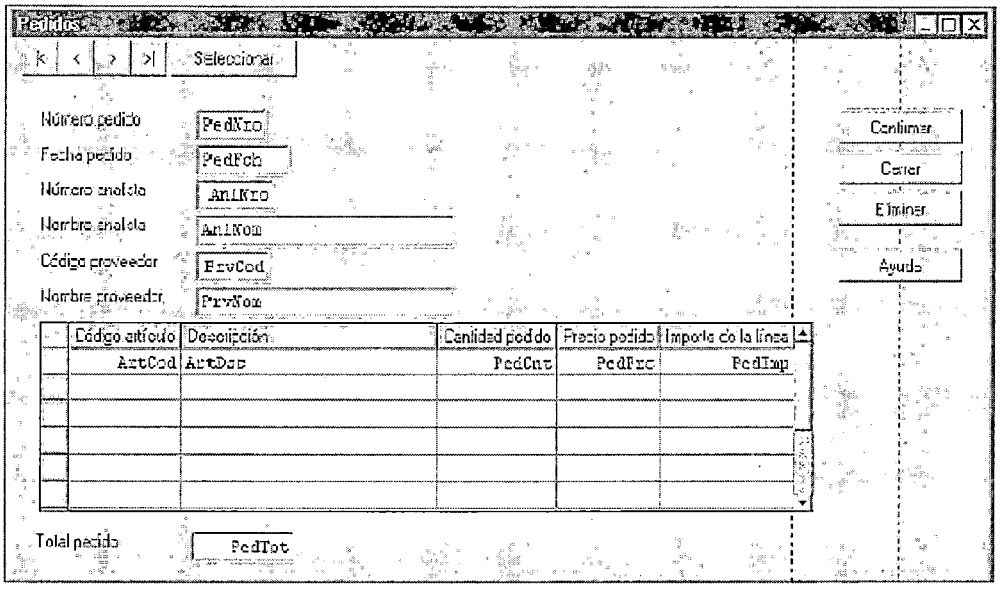


Fig. 8: Form GUI por defecto de la transacción "Pedidos"

Observemos que aparece una grilla que corresponde a los datos del segundo nivel, dado que por cada pedido, se deben poder ingresar varias líneas. Aquí hay algunos atributos que deben ser digitados (son atributos de entrada), por ejemplo PedNro y PrvCod y otros que no, como PrvNom (son atributos de salida, es decir, solo se muestra su valor).

2.13. Facilidad de Prompt de Selección

Para los atributos que están involucrados en la integridad referencial de la tabla asociada (atributos que son claves foráneas) se genera la facilidad de Prompt, que permite visualizar el conjunto de valores válidos para esos atributos, de forma tal de poder seleccionar uno sin tener que recordarlo de memoria.

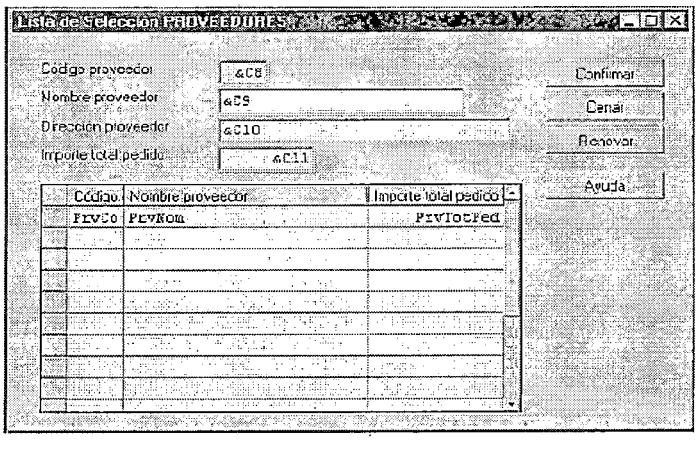


Fig. 9: "Lista de Selección PROVEEDORES"

2.14. Editor de forms

Los objetos GeneXus que implementan una interacción con el usuario, tanto para ingresar como para desplegar datos, lo hacen a través de una pantalla. Las transacciones y los work panels –en ambiente Win- y las transacciones y los web panels –en ambiente Web- son objetos de este tipo, y por tanto tienen forms asociados.

Controles

Un form está compuesto de controles. Podemos definir a un control como un área de la interfaz con el usuario, que tiene una forma y un comportamiento determinado.

Existen distintos controles, entre ellos:

- Form: Un form puede verse en sí mismo como un control.
- Texto: Permite colocar texto fijo (por ejemplo las descripciones de los atributos: Nombre proveedor", "Código de artículo", o cualquier otro texto).
- Atributo/Variable: Permite colocar atributos o variables.
- Línea: Con este control se dibujan líneas horizontales o verticales.
- Recuadro: Permite definir recuadros de distintos tamaños y formas.
- Grid: Permite definir grillas de datos.
- Botón: Permite incluir botones en los forms.
- Bitmap: Permite definir bitmaps estáticos.

Los controles anteriores están disponibles tanto para ser utilizados en interfaz GUI-Windows como Web. La edición del form GUI-Windows se realiza seleccionando la solapa "Form" del objeto. En la figura siguiente mostramos el form GUI-Windows de la transacción "Articulos" que hemos personalizado, cambiando el tipo de control de algunos de los controles atributo:

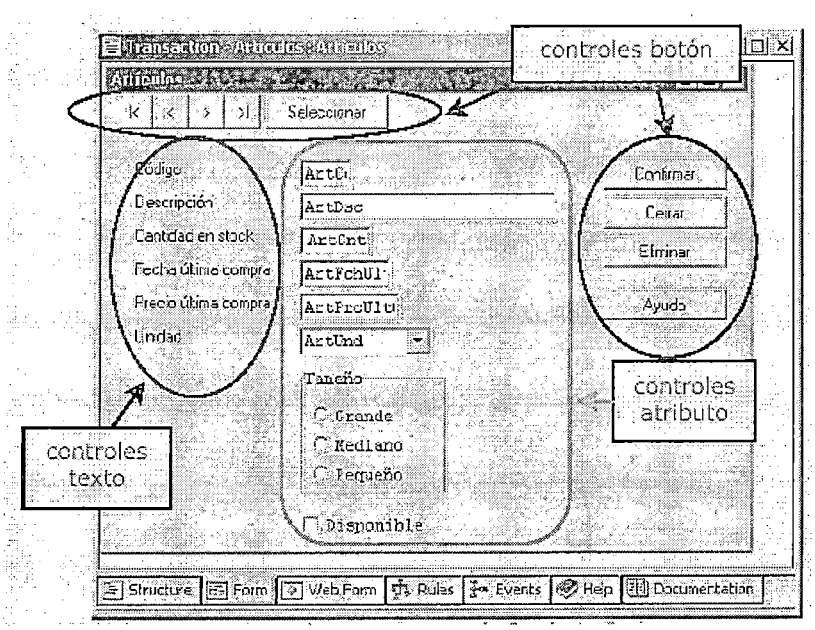


Fig. 10: Form transacción "Artículos"

Los atributos ArtUnd, ArtTam y ArtFlgDsp no aparecen en el form como los atributos convencionales (de tipo edit). ArtUnd lo definimos como un Combo Box, ArtTam como un Radio Button y ArtFlgDsp como un Check Box.

2.15. Tipos de controles atributo/variable

Edit

Normalmente los atributos tienen un rango de valores muy grande, por ejemplo: un nombre, un precio, etc. En estos casos se le permite al usuario entrar el valor del atributo y el sistema se encarga de validarlo. A estos tipos de controles se los llama "Edit Box", o más simplemente "Edit". ArtCod, ArtDsc, etc. en la figura anterior son ejemplos de controles Edit.

Check Box

Es usado para aquellos atributos que tienen solo dos valores posibles. En nuestro ejemplo, para señalar si existe disponibilidad del artículo, el atributo tomará uno de los valores 'S' o 'N'. Existe una única descripción (Disponible) y en caso que este campo esté seleccionado, el valor será el especificado por el analista para este caso ('S' en nuestro ejemplo) y en caso contrario será el otro valor especificado ('N' en el ejemplo).

Radio Button

Los 'Radio Button', en cambio, pueden tener más de dos valores. Todos los valores se despliegan en el form (en realidad se despliegan sus descripciones, el valor que se almacena es manejado internamente) y solo se puede seleccionar uno. En el ejemplo, al "tamaño del artículo", ArtTam, lo definimos como un Radio Button.

Combo Box

Es generalmente usado para aquellos atributos que tienen un rango grande de valores, que hace poco práctico utilizar un Radio Button para su manejo. Se despliega un campo de tipo Edit y presionando un botón que se encuentra a la derecha del campo se despliega una lista con todos los valores válidos. No es recomendable utilizar este tipo de control como lista de selección de atributos que puedan ser leídos de una tabla. Para estos casos se usan los Dynamic Combobox.

Dynamic Combobox

Un Dynamic Combobox es un tipo de control similar al Combo Box. La forma de operación es similar, salvo que los valores desplegados no son estáticos (ingresados por el analista como valores fijos) sino que son descripciones leídas de una determinada tabla de la base de datos.

List Box

Este tipo de control tiene asociada una colección de ítems. Cada ítem tiene asociado un par <valor, descripción>. El control da la posibilidad de seleccionar un solo ítem a la vez. El atributo o variable toma el valor en el momento que se selecciona el ítem. La selección se realiza dando clic con el mouse en un ítem o con las flechas del teclado.

Dynamic List Box

Este tipo de control tiene asociada una colección de ítems. Cada ítem tiene asociado un par <valor, descripción>. La colección de ítems se carga desde una tabla de la base de datos en tiempo de ejecución. En tiempo de diseño se asocian dos atributos al Dynamic List Box, uno al valor que tendrá el ítem y el otro a la descripción que éste tomará. Ambos atributos deben pertenecer a la misma tabla.

2.16. Personalizando el form de "Pedidos"

En la fig.8 vimos el form que crea por defecto GeneXus para la transacción "Pedidos". En dicho form podemos ver que por cada atributo del primer nivel de la estructura, se inserta un control texto y un control atributo. Para los atributos del segundo nivel se diseña una grilla (grid), donde aparece una columna por cada atributo de este segundo nivel, siendo el título de la columna el valor de la propiedad "Column title" del atributo correspondiente. Trabajando con el editor de forms, podemos personalizar la pantalla anterior, de manera tal que nos quede:

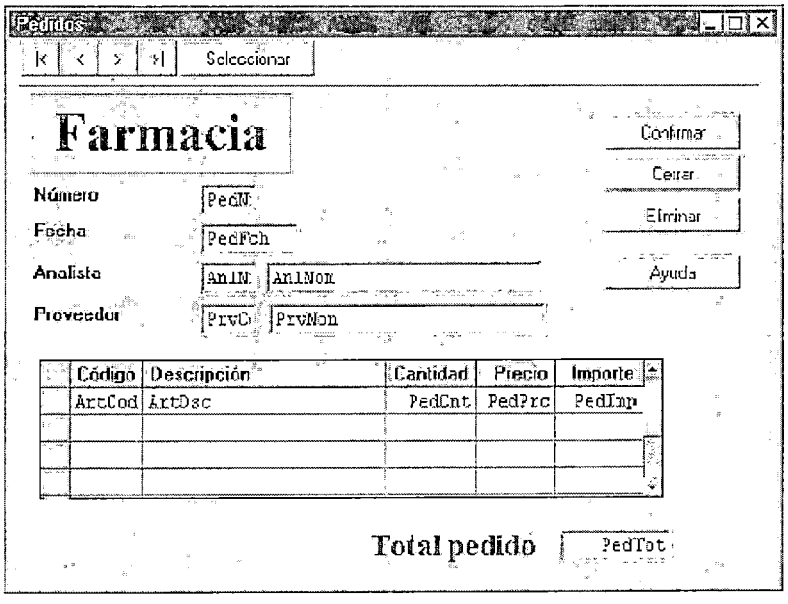


Fig. 11: Form personalizado de transacción "Pedidos"

El form está delimitado por un marco de ventana (frame) cuyo título es la descripción de la transacción. Puede verse como el control Form. Dentro figurarán los distintos tipos de controles de edición del form, que acabamos de mencionar.

Cada control tiene una serie de propiedades (ancho de línea, color, color de fondo, font, etc.), que dependen del tipo de control, y cuyos valores pueden fijarse en forma independiente. Unas páginas más adelante estudiaremos las propiedades de algunos de los controles mencionados.

2.17. Paletas de herramientas

Mientras se está editando un form, están disponibles varias paletas de herramientas, en forma de ventanas adicionales.

Tenemos una paleta que permite insertar controles:

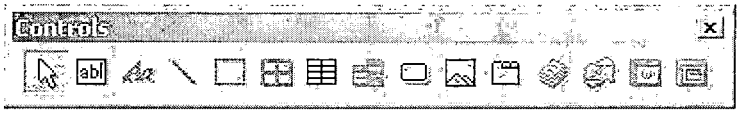


Fig. 12: Paleta de herramientas "Controls"

y otra que permite alinearlos, copiar el tamaño de uno en otro, etc:

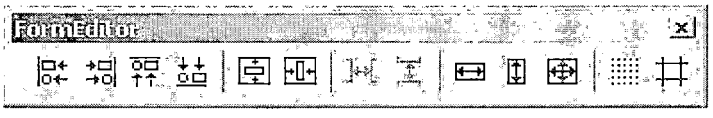


Fig. 13: Paleta de herramientas "Form Editor"

Uso de las Herramientas

Sobre los objetos seleccionados con el puntero vamos a poder realizar varias operaciones:

- Cambiar el tamaño y forma de un control.
- Edición de propiedades.
- Move Forward, Move Back, Move to Front y Move to Back. Estas operaciones nos van a permitir cambiar la capa en que se encuentra un control.
- Move, Copy y Delete.

Propiedades de los controles

Principales propiedades de algunos de los controles que podemos insertar en un form.

Atributo/Variable

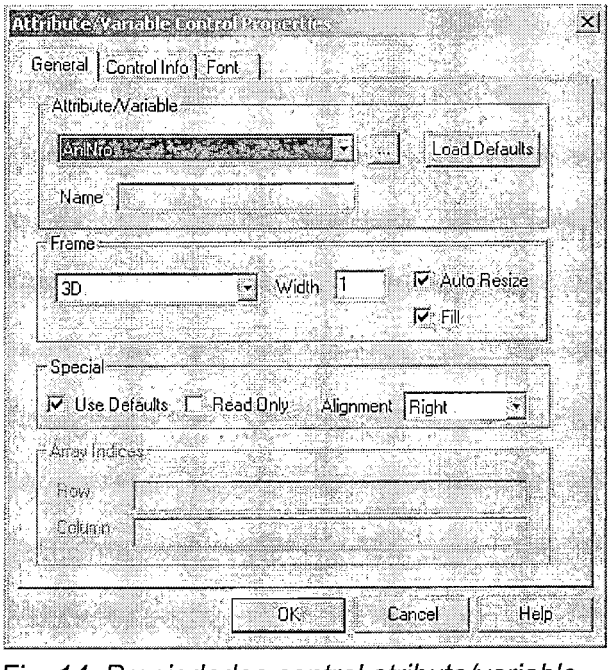


Fig. 14: Propiedades control atributo/variable

Solapa General:

Atribute/Variable: Permite indicar el atributo o variable asociado al control.

Name: En esta opción se permite asignar un nombre al control que se está editando. Este nombre será usado luego para asociarle propiedades, eventos o métodos al control.

Frame: Se puede indicar que el atributo o variable esté rodeado por un marco. Es posible indicar el tipo: None, Single o 3D; el ancho de la/s línea/s (Width), si se pinta dentro de él o no (Fill) y si el tamaño y forma deben determinarse a partir del atributo (Auto Resize).

Array e índices: En caso de utilizar variables no escalares, se habilitan estas propiedades para indicar la fila y la columna de la variable no escalar, que determinan el elemento cuyo contenido se quiere mostrar.

Solapa Control Info:

Nos da la posibilidad de seleccionar el tipo de control, y de acuerdo a él, nos pide que seleccionemos otras características del mismo, así como nos permite seleccionar el Fore Color y el Back Color del control.

Control Type: En los forms generados un atributo podrá asociarse a distintos tipos de controles. Se puede seleccionar uno de los siguientes: Combo Box, Dynamic Combobox, Radio Button, Edit, List Box, Dynamic List Box y Check Box. El botón de Setup que aparece luego de seleccionar el control type permite definir características propias del tipo de control elegido.

Fore Color: Este botón es para seleccionar el color con el que se desplegará el valor del atributo y la/s línea/s del frame, si tuviera.

Back Color: Con este botón se puede seleccionar el color con el que se pinta el área asignada al atributo en la pantalla. Sólo tiene efecto si está presente la propiedad Fill.

Solapa Font

Permite seleccionar el font que se utilizará para el atributo o variable en la pantalla.

Texto

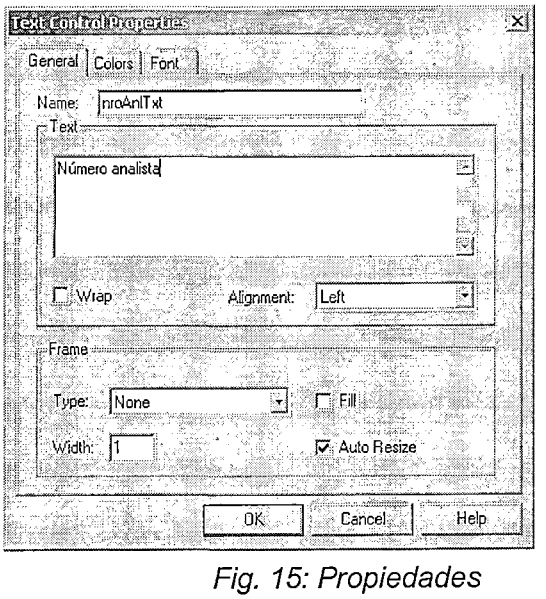


Fig. 15: Propiedades

Text: Indica el texto que se quiere insertar en la pantalla. Puede consistir de una o más líneas.

Wrap: Indica si queremos que se ajuste el texto al tamaño del control, de forma tal que cuando se llegue al tope derecho, lo que siga se coloque en el siguiente renglón.

Alignment: El texto puede estar justificado a la izquierda (Left), derecha (Right) o centrado (Center).

Recuadro. Del combo box se selecciona el tipo del borde del recuadro: single, none (sin borde) o 3D.

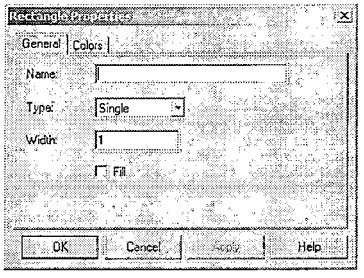


Fig. 16: Propiedades control rectángulo

Línea

Utiliza el mismo diálogo de edición de propiedades con la salvedad de que la opción Fill aparece deshabilitada.

2.18. Editor de Propiedades, Métodos y Eventos

De todos los controles utilizados en un form, a aquellos que tengan nombre asignado se les pueden asociar propiedades, métodos o eventos dinámicamente, por programación dentro del objeto. Por ejemplo: cambiarle el font a un texto, hacer un texto visible o invisible, deshabilitar un botón, etc. El tipo de propiedades/eventos/métodos que se pueden asociar a un control dependen del tipo de control.

Para acceder al diálogo donde poder asociar propiedades a los controles se usa la opción de menú Insert/Property (Insert/Events e Insert/Methods para los eventos y métodos respectivamente) cuando se están editando los eventos, reglas o subrutinas de la transacción. El diálogo que aparece es el siguiente:

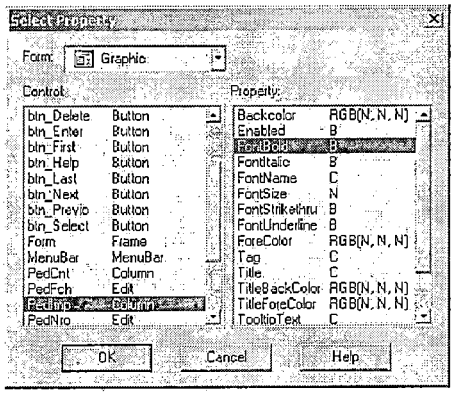


Fig. 17: Diálogo de selección de propiedades de controles

Así, por ejemplo, si en la transacción "Pedidos" queremos que el importe de una línea aparezca en negrita si supera los $1000, podemos escribir las siguientes reglas de la transacción:

PedImp.FontBold = 1 if *PedImp* > 1000;

PedImp.FontBold = 0 if *PedImp* <= 1000;

2.19. Fórmulas

Se dice que un atributo es una fórmula si su valor se puede calcular a partir del valor de otros atributos. En el ejemplo, el atributo PedImp de la transacción "Pedidos" se puede calcular a partir de la cantidad pedida, PedCnt y del precio del producto, PedPrc:

*PedImp = PedCnt * PedPrc*

En cada transacción se puede definir qué atributos son fórmulas, pero esta definición es global, es decir, toda vez que se necesite el atributo se calcula la fórmula, tanto en transacciones como en los otros objetos GeneXus (reportes, work panels, etc.).

Existen distintos tipos de fórmulas:

- Horizontales
- Verticales
- Aggregate/Select

Fórmulas Horizontales

Una fórmula horizontal se define como una o varias expresiones aritméticas condicionales. Por ejemplo, si agregamos un atributo en el primer nivel de la transacción "Pedidos", *PedDto*, que represente el descuento que se aplica sobre el total del pedido y sabemos que este descuento se calcula de la siguiente manera:

- Si el total del pedido es menor a 100 no se aplica descuento
- Si el total del pedido está en el rango 100 - 1000 entonces se aplica sobre éste un 10% de descuento
- Si el total del pedido es mayor a 1000 entonces se aplica un 20% de descuento

El atributo *PedDto* será definido como la fórmula horizontal:

PedDto =	PedTot * 0.10	if PedTot >= 100 and PedTot <= 1000;
	PedTot * 0.20	if PedTot > 1000;
	0	OTHERWISE;

En las expresiones se pueden utilizar los operadores aritméticos (+, -, *, /, ^), funciones del sistema (por ejemplo "today()" que devuelve la fecha del día), atributos, variables y constantes, y para los casos en donde el cálculo es muy complicado se puede llamar a una rutina que lo realice. Por ejemplo: *PedDto = udp('Pdesc', PedTot)*

En donde udp es una función que implementa la comunicación entre objetos GeneXus, 'Pdesc' es el nombre del procedimiento invocado, PedTot es un parámetro que se envía y PedDto es el atributo que recibe el resultado que calcula el procedimiento.

El importe del pedido también es una fórmula horizontal: *PedImp = PedCnt * PedPrc*

Ahora bien, ¿cuáles son los atributos que pueden estar involucrados?. En una fórmula horizontal los atributos de los cuales depende la fórmula deben estar en la tabla extendida del atributo que se está definiendo como fórmula (ver el concepto de tabla extendida en el anexo sobre modelos de datos relacionales).

Como con cualquier otro atributo, cuando se define uno de los atributos de la estructura de una transacción como fórmula, GeneXus, siguiendo los criterios de normalización, puede determinar directamente la tabla a la que está "asociado" ese atributo. A partir de ello encuentra su tabla extendida y si todos los atributos utilizados en la definición de la fórmula se encuentran en esa tabla extendida, entonces la fórmula estará correctamente definida, y será del tipo horizontal.

Fórmulas Verticales

Consideremos ahora el total del pedido, PedTot. Este se debe calcular sumando los importes, *PedImp*, de cada una de las líneas del pedido. Esta fórmula se expresa como:

PedTot = SUM(*PedImp*)

Es un caso de fórmula vertical. En este caso los atributos involucrados no se encuentran dentro de la tabla extendida de la fórmula. En el caso de PedTot, si observamos el modelo de datos usando el diagrama de tablas de GeneXus, conocido como diagrama de Bachman:

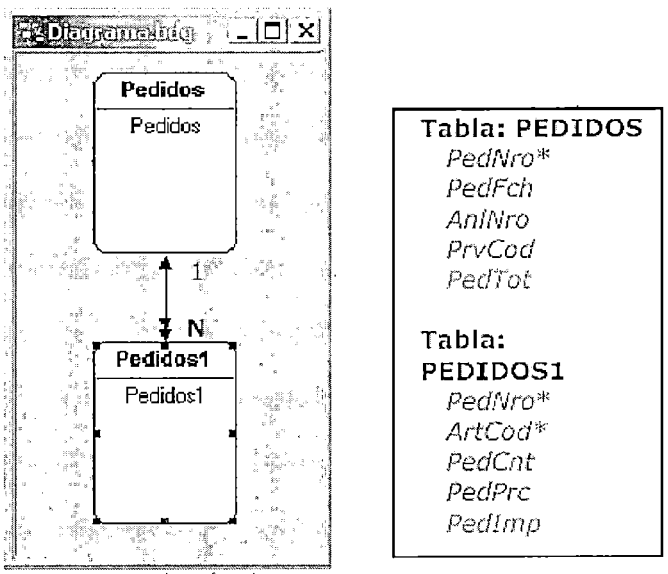


Fig. 18: Diagrama de Bachman de tablas PEDIDOS y PEDIDOS1

tenemos que PedTot está asociada a la tabla PEDIDOS y PedImp a la tabla PEDIDOS1, que es una tabla directamente subordinada a la tabla PEDIDOS. Podemos utilizar también el navegador (browser) de GeneXus para determinar qué tablas están subordinadas y superordinadas a PEDIDOS (eligiendo la opción Subordinated Tables o Superordinated Tables del diálogo del browser, respectivamente):

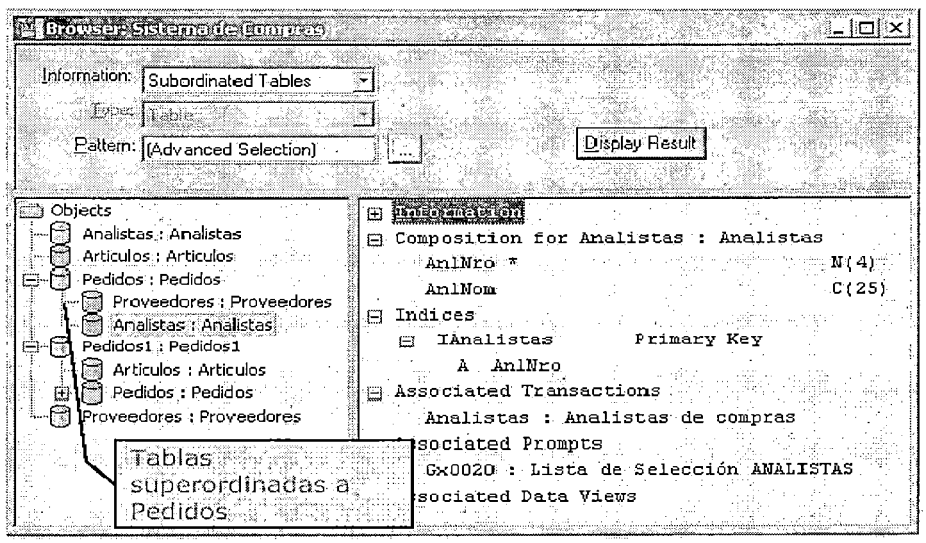


Fig. 19: Browser: Sistema de Compras

Además de ver qué tablas tiene subordinadas y superordinadas, podemos consultar la estructura de una tabla, que índices tiene (composición de los índices), fórmulas, etc.

Existen dos operadores para fórmulas verticales: SUM que suma todos los datos y COUNT que cuenta cuántos datos hay.

Fórmulas y Redundancia

En base a los criterios de normalización y dado que por definición una fórmula siempre puede ser calculada, no es necesario que la misma esté almacenada y basta con recalcularla cada vez que sea necesario.

Sin embargo el hecho de que la fórmula no esté almacenada puede ocasionar problemas de performance, debido al tiempo que puede demorar el recálculo.

2.20. Fórmulas de Fórmulas

Una fórmula se calcula a partir del valor de otros atributos. Dichos atributos pueden estar almacenados o ser otras fórmulas. Por ejemplo, si en la transacción de proveedores definimos:

PrvTotPed = SUM(*PedTot*) Total pedido del proveedor

tenemos que para calcularlo se necesita:

PedTot = SUM(*PedImp*)

que a su vez necesita:

PedImp = *PedCnt* * *PedPrc*

Para fórmulas de fórmulas GeneXus determina cuál es el orden de evaluación correspondiente:

PedImp = *PedCnt* * *PedPrc*

PedTot = SUM(*PedImp*)

PrvTotPed = SUM(*PedTot*)

Las fórmulas aparecen en la estructura de la transacción, como se muestra en la siguiente figura:

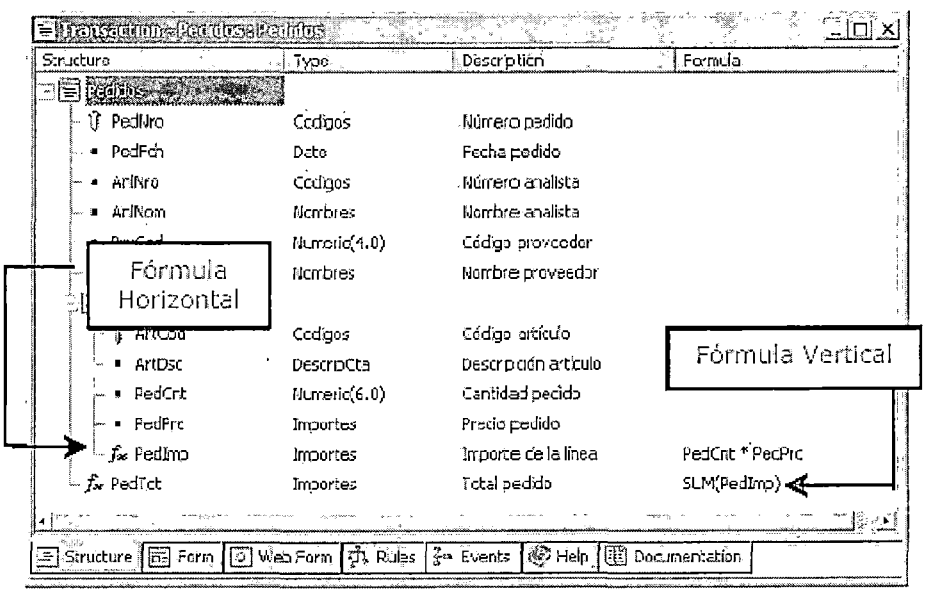


Fig. 20: Fórmulas en la estructura de transacción "Pedidos"

2.21. Reglas

Para definir el comportamiento de las transacciones se utilizan las reglas. Usando reglas se definen valores por defecto, controles, etc. Veremos algunas de las reglas en lo que sigue.

Default

Se utiliza para definir los valores por defecto de algunos atributos, por ejemplo:

default(PedFch, today());

El funcionamiento de la regla default varía según el tipo de diálogo (full-screen o campo a campo). En el diálogo full-screen se asigna el valor por defecto si el usuario no digitó nada en el campo. En el diálogo campo a campo primero se asigna el valor por defecto y luego se permite modificarlo.

Error

Es la regla para definir los controles que deben cumplir los datos. Por ejemplo, si queremos

impedir que quede en un pedido un artículo con cantidad pedida cero, escribimos la regla:

error('La cantidad pedida debe ser mayor que 0') if PedCnt <= 0;

Cuando se cumple la condición (*PedCnt* <= 0) se despliega el mensaje (' La cantidad pedida debe ser mayor que 0') en pantalla y no se permite continuar hasta que el usuario ingrese un valor correcto o abandone la transacción.

Una utilización relativamente común es incluir una regla de error para prohibir alguna de las modalidades de la transacción. Por ejemplo:

error('No se permite eliminar Pedidos') if delete;

Con esta regla se prohibe que el usuario entre en el modo eliminación y así se prohibe la eliminación de Pedidos.

Asignación

Dentro de las reglas de la transacción se permiten definir asignaciones a atributos y variables. La asignación a atributos implica una actualización en la tabla base del nivel o en alguna de las tablas que pertenecen a la tabla extendida del nivel. Veamos un ejemplo. En la transacción de Artículos:

Artículos:
*ArtCod**
ArtDsc
ArtCnt
ArtFchUltCmp
ArtPrcUltCmp
ArtUnd
ArtTam
ArtFlgDsp

En el atributo *ArtPrcUltCmp* se desea almacenar cuál fue el precio del último pedido realizado para ese artículo, y en *ArtFchUltCmp* en qué fecha fue realizado. Esto se logra definiendo en la transacción "Pedidos" las siguientes reglas:

ArtPrcUltCmp = *PedPrc* if insert;

ArtFchUltCmp = *PedFch* if insert;

Así cada vez que se ingrese una línea del pedido se actualiza la tabla de artículos con la fecha y el precio correspondiente. Existe una similitud entre fórmulas y asignaciones, incluso la sintaxis de definición es similar. La diferencia entre ambas es que una fórmula es GLOBAL, es decir, vale para todos los objetos GeneXus que la utilicen, mientras que una asignación es LOCAL, vale solo para la transacción en la cuál esté definida.

2.22. Add y Subtract

Las asignaciones que vimos en la sección anterior eran relativamente fáciles, pero existen casos más sofisticados. Por ejemplo, en la misma transacción de artículos tenemos el atributo *ArtCnt* en donde se quiere mantener cuál es el stock que tenemos de cada artículo.

Sin duda la transacción "Pedidos" debe modificar ese valor, porque cada pedido nuevo aumenta el stock. También existirá alguna otra transacción (ej: "Facturas") que hace disminuir el stock cada vez que se vende el artículo. Como esta transacción está fuera del alcance del proyecto solo estudiaremos la actualización del stock relacionada con la transacción "Pedidos".

En dicha transacción se debe actualizar rtCnt. Esto se podría hacer con la regla de asignación:

ArtCnt = ArtCnt + PedCnt;

Pero no debemos olvidar que en la misma transacción se permite crear, modificar o eliminar un pedido, y la asignación anterior solo es correcta si se está creando uno nuevo, ya que si por ejemplo se está eliminando, la asignación correcta es:

ArtCnt = ArtCnt - PedCnt;

Entonces, para actualizar *ArtCnt* correctamente se necesitaría también considerar los casos de modificación y eliminación, pero para evitar todo esto GeneXus provee la regla add, que lo hace automáticamente:

add(*PedCnt*, *ArtCnt*);

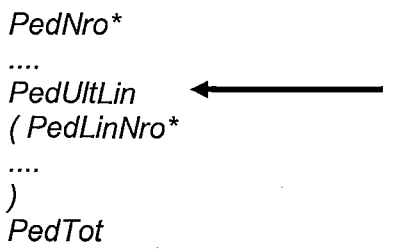
Con esta regla si se está insertando una línea del pedido se suma *PedCnt* a *ArtCnt*; si se está eliminando se resta y si se está modificando se resta el valor anterior de PedCnt (que se define como old(*PedCnt*)) y se suma el nuevo valor.

2.23. Serial

Esta regla es útil cuando se quiere asignar automáticamente valores a algún atributo de la transacción. Por ejemplo, en la transacción "Pedidos", si consideramos la segunda opción vista, en la que un artículo puede repetirse para un mismo pedido, teníamos como identificador del segundo nivel a *PedLinNro* y si queremos que este número sea asignado automáticamente por el sistema se puede usar la regla:

serial(*PedLinNro*, *PedUltLin*, 1);

en donde el primer parámetro define cuál es el atributo que se está numerando, el segundo indica cuál es el atributo que tiene el último valor asignado (aquí se trata del último número de línea asignado) y el tercer parámetro el incremento (en este caso se incrementa de a uno). El atributo *PedUltLin* (última línea asignada) debe ser incluido en la estructura de la transacción en el nivel de PedNro (un pedido solo tiene UNA última línea asignada):



De esta manera para cada nueva línea del pedido el programa asigna automáticamente su número. En el diálogo campo a campo (donde la modalidad era inferida automáticamente), se debe digitar un valor inexistente en *PedLinNro* (usualmente 0) y el programa asigna el valor correspondiente. En el diálogo full-screen el valor se asigna cuando el usuario presiona Enter en modo Insert.

Orden de evaluación

La definición de reglas es una forma DECLARATIVA de definir el comportamiento de la transacción. El orden en el cual fueron definidas no necesariamente coincide con el orden en que se encuentran en el programa generado, y por tanto en el que son ejecutadas.
GeneXus se encarga de determinar el orden correcto según las dependencias existentes entre atributos.

Supongamos que definimos en la transacción "Artículos" un nuevo atributo: *ArtCntMax*, del mismo tipo que *ArtCnt*. Este atributo indicará el stock máximo que se desea tener de ese artículo.
Cuando se ingresa un pedido debemos emitir un mensaje si se sobrepasa el stock máximo de un artículo. Para ello definimos las siguientes reglas en la transacción de pedidos:

msg('Se sobrepasa el stock máximo del artículo') if *ArtCnt* > *ArtCntMax*;

add(*PedCnt*, *ArtCnt*);

El orden de evaluación será:

add(*PedCnt*, *ArtCnt*);

msg('Se sobrepasa el stock máximo del artículo') if *ArtCnt* > *ArtCntMax*;

ya que la regla add modifica ArtCnt y la regla msg consulta ese atributo.

Esto implica que en la regla msg se debe preguntar por ArtCnt mayor que ArtCntMax y no por *ArtCnt* + *PedCnt* mayor que *ArtCntMax*.

Call

Call es una regla -y a la vez un comando- con la cual se permite llamar a otro programa. Este último puede ser cualquier otro objeto GeneXus (por ejemplo, de una transacción se puede llamar a un reporte o a otra transacción, etc.), o un programa externo.

En el caso de la regla call es necesario precisar en qué momento se debe disparar el llamado. Por ejemplo, si en la transacción "Pedidos" se quiere llamar a un reporte que imprima el pedido que se acaba de ingresar, se utiliza la regla:

call(RImpRec, *PedNro*) On AfterComplete ;

donde 'RImpRec' es el programa llamado y PedNro es el parámetro que se le pasa. Al tener el evento de disparo AfterComplete, el call se realizará después de haber completado todo el Pedido.

El uso del evento AfterComplete no es privativo de la regla call y puede ser utilizado en otras reglas.

2.24. Eventos y condiciones de disparo de reglas

Existen eventos del sistema en una transacción, que están relacionados con los distintos momentos que se van sucediendo en el ingreso de los datos. Por ejemplo, existe un evento que ocurre inmediatamente después de que se inserta el registro correspondiente al cabezal en la tabla asociada. Existe otro que ocurre un instante antes. Otro evento ocurre inmediatamente luego de insertados todos los registros (cabezal y líneas), etc.

A estos eventos se les conoce como eventos de disparo ya que se utilizan para condicionar el momento de ejecución –"disparo"- de una regla.

Con lo visto hasta el momento, la sintaxis de las reglas nos queda:

regla [if <cond>] [On <evento>] [level <att>];

Los eventos de disparo tienen que ver con los momentos en los que ocurren las grabaciones/actualizaciones/elminaciones físicas de los registros de la base de datos.

AfterValidate

Ocurre después de que se confirman los datos del nivel de la transacción en el que se está trabajando, pero antes de realizarse la actualización física correspondiente. Se usa por ejemplo, en algunos casos de numeración automática cuando no se quiere utilizar la regla serial para evitar problemas de control de concurrencia.

AfterInsert | AfterUpdate | AfterDelete

Una regla condicionada a alguno de estos eventos de disparo, se ejecutará inmediatamente después de haber insertado, actualizado o eliminado el registro de la tabla base del nivel. Se usa fundamentalmente para llamar a procesos que realizan actualizaciones.

AfterLevel

Se dispara después de haber entrado todos los datos de un nivel. Se usa muchas veces para controles de totales. Por ejemplo, si cuando se entra el cabezal del pedido se digita un total (llamémosle *PedTotDig*) y se quiere verificar que éste sea igual que el total calculado

escribiremos las reglas:

error('El total digitado no cierra con el calculado') if (*PedTotDig* <> *PedTot*) On AfterLevel Level *ArtCod*;

en donde el control recién se realizará cuando se terminen de entrar todas las líneas del pedido.

AfterComplete

Se dispara después de haber entrado todos los datos de una transacción. Se usa fundamentalmente para llamar a programas que imprimen los datos, o a otras transacciones. En ambientes con integridad transaccional este evento ocurre luego de que se realiza el commit de la transacción.

2.25. Propiedades

En el editor de propiedades de las transacciones se pueden seleccionar configuraciones específicas para definir el comportamiento general del objeto. A continuación vemos la pantalla para la edición de las mismas:

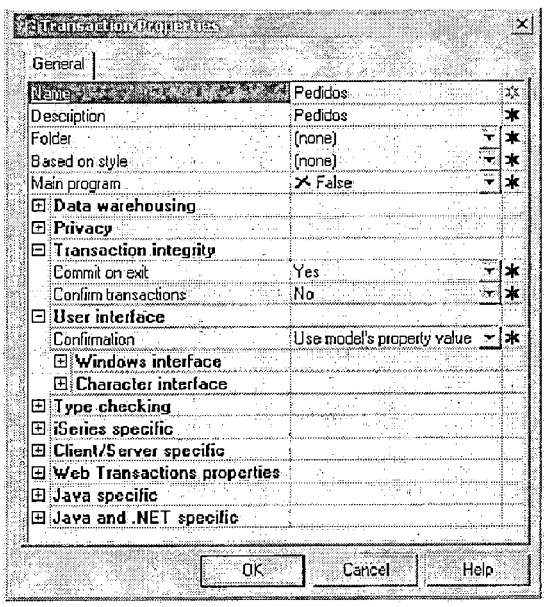


Fig. 21: Diálogo de propiedades de una transacción

CAPÍTULO III

3. OBJETO REPORTE

Con reportes se definen procesos no interactivos de extracción de datos. Los reportes son listados cuya salida puede: emitirse por impresora, visualizarse por pantalla, o enviarse a un archivo. Es un proceso no interactivo, porque primero se extraen los datos y luego se muestran, no habiendo interacción con el usuario durante el proceso de extracción, y es solo de extracción ya que no se pueden actualizar los datos leídos.

Elementos

Para cada reporte se definen, entre otros, los siguientes elementos:

Layout

Así como las transacciones tienen una pantalla (form), los reportes tienen un "layout" de la salida. En esta sección se define la presentación del reporte: los datos que se quieren listar y el formato de la salida.

Source

Aquí se escribe el código correspondiente a la lógica del reporte. También pueden definirse al final del código subrutinas que podrán ser invocadas desde el propio código mediante el comando adecuado.

Reglas - Propiedades

Definen aspectos generales del reporte, como su nombre, descripción, tipo de salida (impresora, archivo, pantalla), parámetros que recibe el objeto, etc.

Condiciones

Condiciones que deben cumplir los datos para ser recuperados (filtros).

Ayuda

Texto para la ayuda a los usuarios en el uso del reporte.

Documentación

Texto técnico para ser utilizado en la documentación del sistema.

3.1. Solapas de acceso a los elementos

Como para todo objeto GeneXus, se puede acceder a varios de los elementos que lo componen utilizando las solapas que aparecen bajo la ventana de edición del objeto:

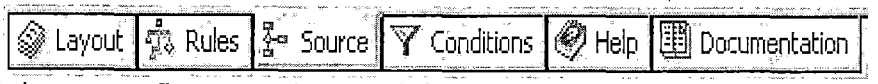
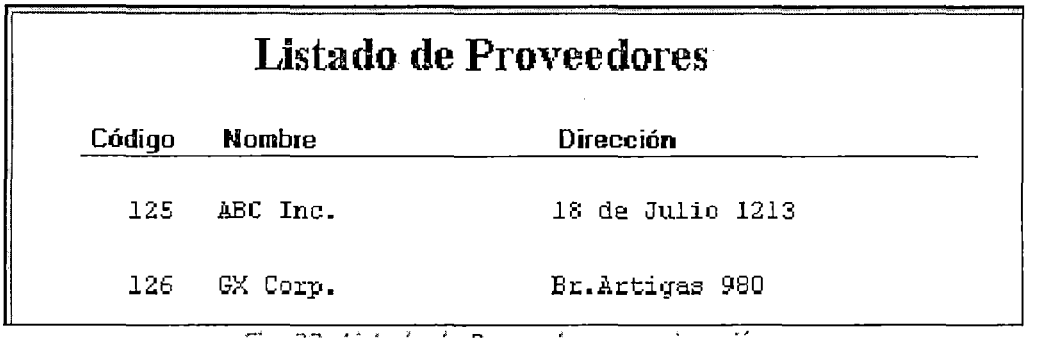


Fig. 22: Solapas que figuran en la parte inferior de la ventana de edición de un reporte

Layout

En esta sección se definen los datos que se quieren listar y el formato de la salida. El Layout es una sucesión de bloques conocidos como print blocks (bloques de impresión). Estos bloques son áreas que serán desplegadas en el listado resultante y que contendrán controles que indicarán qué es lo que se quiere imprimir, dando forma a la salida.

Por ejemplo, supongamos que queremos implementar un reporte para imprimir el código, nombre y dirección de todos nuestros proveedores y queremos que el listado luzca como sigue:



Listado de Proveedores

Código	Nombre	Dirección
125	ABC Inc.	18 de Julio 1213
126	GX Corp.	Br.Artigas 980

Fig. 23: Listado de Proveedores en ejecución

Para implementar este listado creamos un objeto reporte, y definimos en su sección Layout todos los print blocks necesarios. Para este ejemplo, podríamos dividir la salida en 2 áreas, una con datos fijos y otra con datos variables, cada una implementada con un print block. El Layout correspondiente se muestra a continuación:

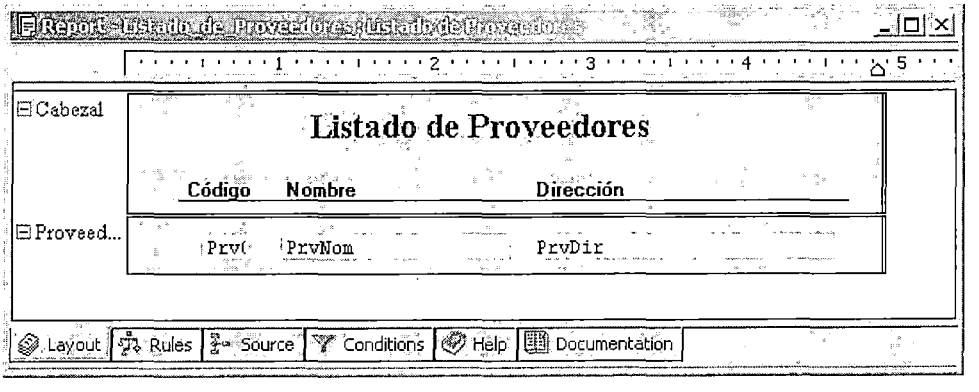


Fig. 24: Layout del "Listado de Proveedores"

El orden de los print blocks dentro del Layout es irrelevante: en esta sección simplemente se declaran. El orden en el que serán desplegados en la salida queda determinado en la sección Source, que es la que contiene la lógica del reporte. Desde allí serán invocados para ser impresos, mediante un comando específico para tal fin (el comando print).
Por esta razón cada print block deberá tener un nombre, de forma de poder ser referenciado luego desde el Source. Este nombre es el que aparece a la izquierda de cada uno de ellos en el Layout de la figura 24.

El print block es un nuevo tipo de control, solo válido en reportes y procedimientos, que indica que debe mostrarse en la salida la información especificada dentro de él. Esta información es la representada por otros controles, de los ya estudiados (atributos, textos, recuadros, líneas, etc.).

Para listar el código, nombre y dirección de todos los proveedores, el print block de nombre "Proveedor" deberá ser invocado dentro de una estructura repetitiva en el Source, de forma tal que se recorran todos los proveedores y se impriman cada vez los datos del proveedor en el que se está posicionado en ese momento. Esta estructura repetitiva es el comando for each que introduciremos luego.

Como cualquier otro control, el print block tiene asociadas ciertas propiedades, que pueden ser configuradas desde un diálogo que se abre al hacer doble-clic sobre el control. Algunas de estas propiedades son el tipo de papel, el tipo de letra, etc. El diálogo es el siguiente:

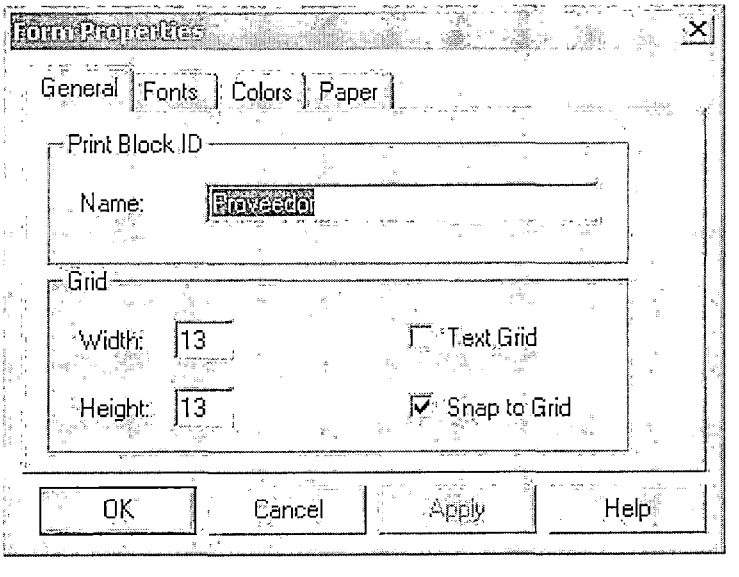


Fig. 25: Propiedades del control Print Block

Para el diseño de los print blocks tenemos disponible la misma paleta de herramientas "Controls" de las transacciones, teniendo habilitadas las opciones que aplican a este tipo de objeto:

Fig. 26: Paleta de herramienta "Controls" en reportes

Esta paleta permite insertar atributos, textos, líneas, recuadros, bitmaps y nuevos print blocks. Dentro del reporte, los controles son insertados y configurados de la misma forma que en el editor del form de las transacciones.

Source

En esta sección se programa la lógica del reporte. El lenguaje que se utiliza es muy simple, y consta de algunos comandos.

El estilo de programación es procedural (imperativo), por lo que el Source será una sucesión de comandos para los que importa el orden, dado que éste será el orden de ejecución.
Existen, como en todo lenguaje imperativo, comandos de control para la ejecución condicional (if, do case), la repetitiva (do while, for), para invocar a otro objeto (call), para cortar las iteraciones dentro de un bucle (exit) o abandonar el programa (return), así como también comandos específicos de este lenguaje: para imprimir un print block del Layout (print), para acceder a la base de datos (for each), para invocar a una subrutina (do), etc.

Al final del source pueden definirse subrutinas (mediante el comando sub) que serán locales al objeto, y que podrán ser invocadas más arriba, dentro del mismo source, en el código de ejecución, mediante el comando do. El siguiente es el Source que implementa la lógica del reporte del ejemplo:

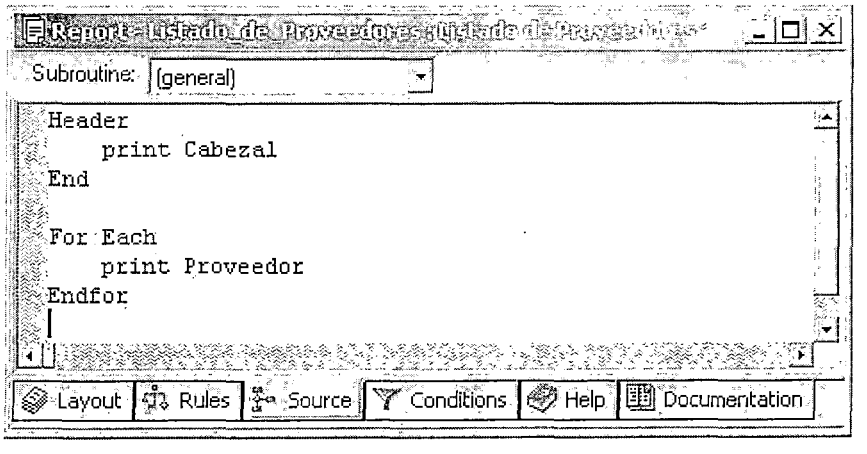


Fig. 27: Source del "Listado de Proveedores"

donde tenemos los comandos Header, Print y For each.

Como resulta evidente, con el comando print mandamos a imprimir un print block en la salida. Todos los print blocks que se manden a imprimir entre los comandos Header y end son las líneas que se imprimen en el cabezal de la página. También existe el comando Footer que permite imprimir un pie de página en cada hoja.

El comando For each indica que se consultarán cada uno de los registros de cierta tabla, mostrando la información que se indique en el/los print block/s que se mande/n a imprimir dentro del comando.

Comando FOR EACH

Este comando es fudamental, ya que es el único comando que permite recuperar información de la base de datos. Usando el for each se define la información que se va a leer. La forma de hacerlo se basa en nombrar los atributos a utilizar y NUNCA se especifica de qué tablas se deben obtener, ni qué índices se deben utilizar.

Con este comando se define QUE atributos se necesitan y en qué ORDEN se van a recuperar, y GeneXus se encarga de encontrar COMO hacerlo. Obviamente esto no siempre es posible, y en esos casos GeneXus da una serie de mensajes de error indicando por qué no se pueden relacionar los atributos involucrados.

El acceso a la base de datos queda especificado por los atributos que son utilizados dentro de un grupo FOR EACH - ENDFOR. Para ese conjunto de atributos GeneXus buscará la tabla extendida que los contenga (el concepto de tabla extendida es muy importante y sugerimos repasar su definición en el Anexo sobre Modelos de Datos Relacionales).

Por ejemplo, en el reporte anterior, dentro del FOR EACH - ENDFOR se utilizan los atributos *PrvCod*, *PrvNom* y *PrvDir*. GeneXus "sabe" que estos atributos se encuentran en la tabla PROVEEDORES y por lo tanto el comando:

```
For each
   Print Proveedores
endfor
```

es interpretado por GeneXus como:

```
For each record in table PROVEEDORES
   Imprimir PrvCod, PrvNom y PrvDir
endfor
```

Para clarificar el concepto hagamos un reporte que involucre datos de varias tablas. Por ejemplo, listemos el cabezal de todos los pedidos:

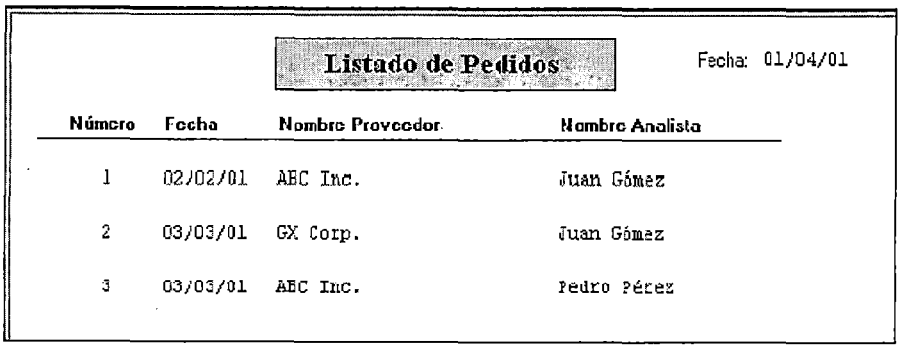


Fig. 28: "Listado de Pedidos" en ejecución

El Layout para este listado es:

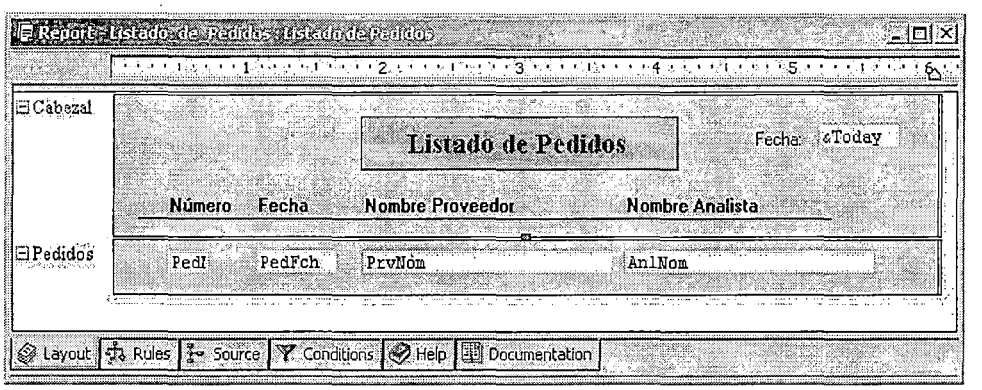


Fig. 29: Layout "Listado de Pedidos"

El diagrama de Bachman de las tablas del modelo es:

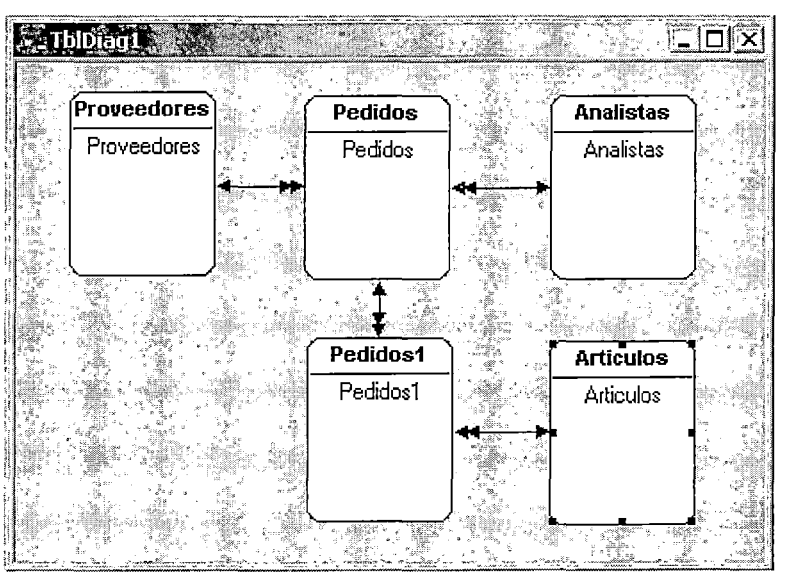


Fig. 30: Diagrama de Bachman del modelo

En el listado anterior se requieren datos de las tablas PROVEEDORES (*PrvNom*), ANALISTAS (*AnlNom*) y PEDIDOS (*PedNro* y *PedFch*). La tabla extendida de PEDIDOS contiene a todos estos atributos y por lo tanto el comando FOR EACH se interpretará como:

For each record in tabla PEDIDOS

Find the corresponding PrvNom in table PROVEEDORES

Find the corresponding AnlNom in table ANALISTAS

Imprimir PedNro, PedFch, PrvNom y AnlNom

endfor

Cuando se especifica un reporte, GeneXus brinda un listado, que llamamos Listado de Navegación, donde se indica cuáles son las tablas que se están accediendo:

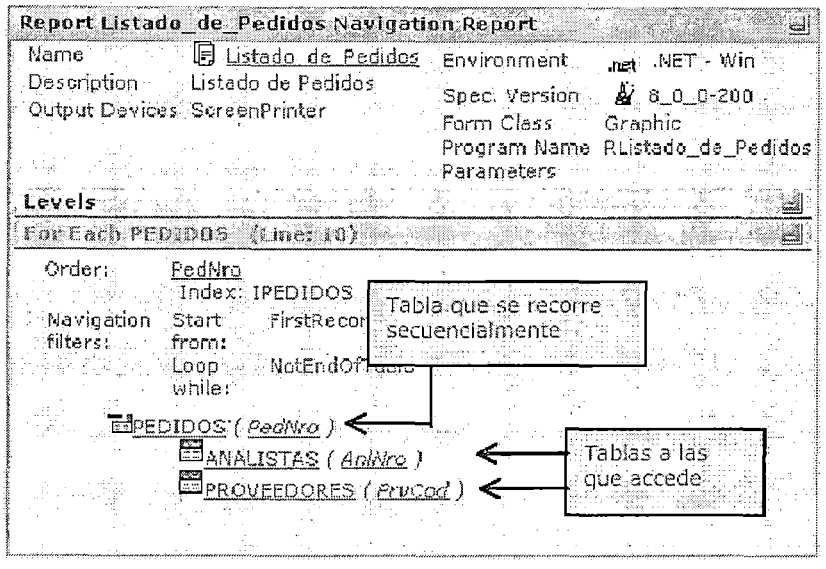


Fig. 31: Listado de Navegación del reporte "Listado de Pedidos"

Donde la primer tabla representada indica cuál es la tabla base del for each y las tablas que figuran debajo de ésta, e indentadas, indican las tablas que deben accederse a partir de cada registro de la tabla base en el que se esté posicionado en cada momento. Una interpretación muy práctica de este diagrama es: para la tabla del primer nivel de indentación se leen muchos registros y para cada tabla del siguiente nivel se lee un solo registro: el asociado al del nivel anterior.

En este diagramita de tablas que aparece en el listado, la relación entre la tabla de un nivel de indentación, con cualquier tabla "anidada" es N-1.

Orden

En los reportes anteriores no se tuvo en cuenta en qué orden se deben listar los datos. Cuando no se indica el orden, GeneXus utiliza como orden el de la clave primaria de la tabla base del For Each (existen algunas excepciones a esta regla).

En el listado de navegación, para cada for each se indica cuál es su tabla base, e inmediatamente el orden elegido para recuperar la información. Como podemos ver en la fig.31, GeneXus indica que va a ordenar por PedNro, es decir, por el atributo que es clave

primaria de la tabla base del for each.

Para los casos en donde se quiere definir el orden de los datos de forma explícita, se utiliza la cláusula ORDER en el comando For each. Por ejemplo, para listar los pedidos ordenados por código de proveedor, *PrvCod*:

```
For each ORDER PrvCod
    print Pedidos
endfor
```

En este caso el índice a utilizar será el de nombre IPEDIDOS2 (*PrvCod*) de la tabla PEDIDOS, que fue definido automáticamente por GeneXus, ya que *PrvCod* es una clave foránea. La navegación para este reporte es la siguiente:

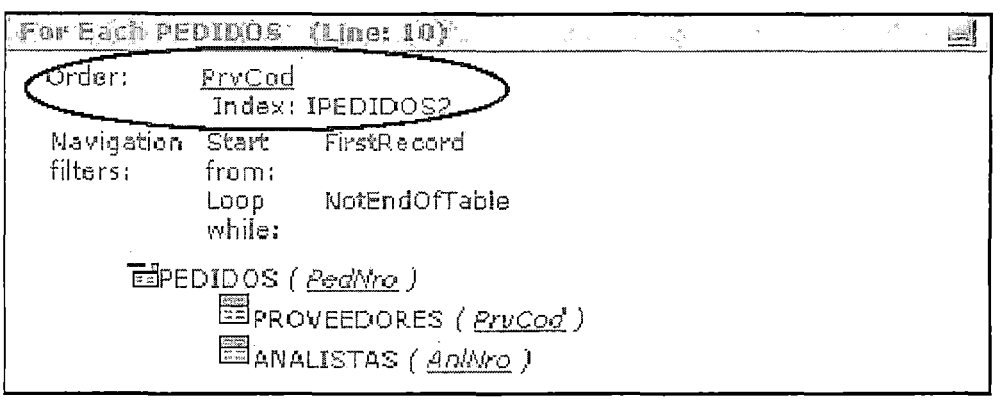


Fig. 32: Fragmento del listado de navegación reporte "Listado de Pedidos"

Si queremos que el listado de pedidos salga ordenado por fecha haremos:

```
For each ORDER PedFch
    print Pedidos
endfor
```

En este caso no hay ningún índice definido en PEDIDOS para satisfacer el orden, dado que PedFch no es clave en ningún lado. Como no existe un índice definido, GeneXus indicará en el listado de navegación mediante una advertencia ("warning") que no existe un índice para satisfacer el orden, lo que podría ocasionar problemas de performance, dependiendo de la plataforma de implementación, de la cantidad de registros que deben ser leídos de la tabla, etc.

En algunas plataformas, como VFP o iSeries (AS/400), si no existe índice para satisfacer el orden especificado se crea uno temporal cada vez que se ejecute el reporte y se elimina al finalizar la ejecución.

Este tipo de índice es el que llamamos índice temporal.

En otras plataformas, como VB con Access o en ambientes cliente/servidor, si no existe índice para satisfacer el orden, el For Each se traduce en una consulta sql ("select") que es resuelta por el motor del DBMS de la plataforma.

El listado de navegación mostrará en este caso:

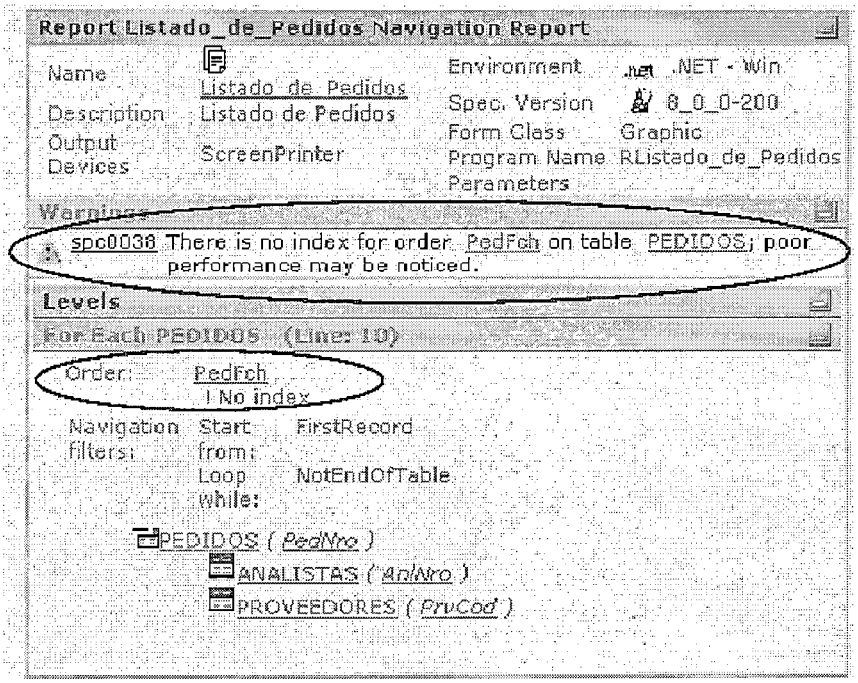


Fig. 33: Listado de navegación del reporte "Listado de Pedidos" ordenado por fecha

Evidentemente este proceso es más lento si lo comparamos con el listado anterior en donde había un índice definido. Dada esta situación el analista debe tomar una decisión:

- Crear un índice de usuario. En este caso el reporte será bastante más eficiente, pero se debe mantener un índice más (lo que implica mayor almacenamiento y mayor procesamiento en las transacciones para mantener actualizado el índice).
- No crear el índice de usuario, y dejar que la navegación sea resuelta por el DBMS o mediante la creación de un índice temporal.

No es posible recomendar, a priori, cuál de las dos soluciones es la mejor, por lo tanto se debe estudiar, caso por caso, la solución particular, siendo fundamental la frecuencia con que se ejecutará el reporte. De cualquier manera si al comienzo no se definió el índice y posteriormente se decide definirlo, alcanza con regenerar el reporte (sin modificar nada del mismo) y éste pasará a utilizarlo.

Los criterios de ordenamiento utilizados por GeneXus son:

Cada grupo FOR EACH puede estar ordenado por CUALQUIER conjunto de atributos pertenecientes a la tabla base del grupo, pero no se puede ordenar por atributos que NO se encuentren en la misma. Por ejemplo el listado anterior no se puede ordenar por *PrvNom* porque la tabla base del grupo es PEDIDOS en la que no está almacenado el atributo *PrvNom*. Sin embargo, si la plataforma de implementación es cliente/servidor, el criterio es más amplio y podrá también ordenarse por atributos de la tabla extendida (por lo tanto en este caso sí se podrá ordenar por *PrvNom*).

Condiciones en el FOR EACH

Para restringir los datos que se quieren listar se utiliza la cláusula WHERE. Por ejemplo, para listar sólo los pedidos de hoy:

```
For each
WHERE PedFch = Today()
        print Pedidos
endfor
```

Se pueden definir varios WHERE dentro del FOR EACH:

```
For each
WHERE PedFch = Today()
WHERE PedNro > 100
        print Pedidos
endfor
```

Lo que significa que se deben cumplir AMBAS condiciones, es decir, se interpreta como que ambas cláusulas están relacionadas por un AND. Si se quiere utilizar un OR (es decir, se desea que se cumpla alguna de las dos condiciones) se debe utilizar un solo WHERE:

```
For each
WHERE PedFch = Today() OR PedNro > 100
        print Pedidos
endfor
```

GeneXus posee una serie de mecanismos para optimizar el reporte en función del orden del FOR EACH y de las condiciones del mismo.

Cuando la condición ha sido optimizada, en el Listado de Navegación aparece como 'Navigation Filter' y cuando no, como 'Constraint'.

Por ejemplo, la parte del listado de navegación que corresponde al siguiente for each de un reporte:

For each order *PedFch*

WHERE *PedFch* = *&today*

print Pedidos

endfor

siendo *&today* una variable del sistema que contiene la fecha de hoy, es:

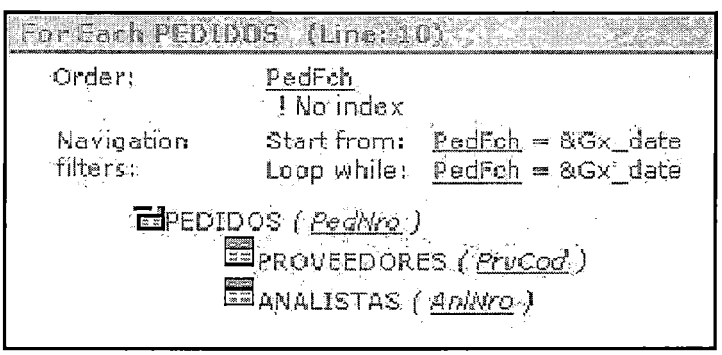


Fig. 34: Fragmento listado de navegación de for each optimizado

Observar que los "Navigation filters" especifican el rango de la tabla que se va a recorrer para luego seleccionar los registros que cumplan con los filtros que aparecen como "Constraints". En el listado anterior no hay "Constraints", lo que significa que se van a imprimir todos los registros del rango especificado. En este caso el rango a recorrer no es toda la tabla. Como se está ordenando por *PedFch* y filtrando por *PedFch*, no tiene que recorrerse toda la tabla. Decimos que se eligió un orden compatible con los filtros, y por ello GeneXus puede optimizar el acceso a las tablas.

For each

WHERE *PedFch* = *&today*

print Pedidos

endfor

muestra en el listado de navegación:

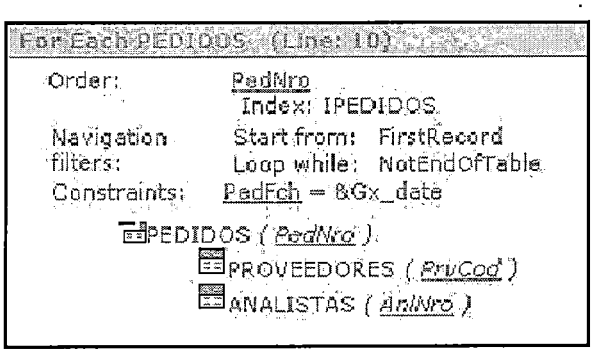


Fig. 35: Fragmento listado de navegación de for each NO optimizado

Este reporte no optimiza el acceso a las tablas, pues no hemos ordenado de forma compatible con los filtros.

Cláusula WHEN NONE de un For Each

En muchos casos es necesario ejecutar determinado código cuando en un For Each no se encuentra ningún registro. Para esto se usa la cláusula WHEN NONE.

```
For each
where PedFch = Today()
                    print Pedidos
WHEN NONE
                    Msg('No se realizó ningún pedido en el día de hoy')
Endfor
```

Es importante aclarar que si se incluyen For Eachs dentro del When None no se infieren Joins ni filtros de ningún tipo con respecto al For Each que contiene el When None.

Determinación de la tabla extendida del grupo FOR EACH

Resulta importante saber cómo determina GeneXus la tabla extendida que se debe utilizar. El criterio básico es:

Dado el conjunto de atributos que se encuentran dentro de un grupo FOR EACH - ENDFOR, GeneXus determina la MINIMA tabla extendida que los contiene.

Supongamos que queremos listar el nombre del analista y proveedor de cada pedido. Para ello escribiremos en el Source:

```
For each
                    print NombreProveedorAnalista
endfor
```

donde "NombreProveedorAnalista" será el nombre de un print block del Layout que contendrá los atributos *PrvNom* y *AnlNom*. Observemos que la única diferencia con el listado de pedidos realizado antes es que aquí no nos interesa listar ni el nro. de pedido ni la fecha del mismo, sino solo analista y proveedor.

Ahora los atributos que participan en la determinación de la tabla base del For Each son *PrvNom* y *AnlNom*. La mínima tabla extendida que contiene a los atributos del For Each sigue siendo PEDIDOS.

Para ver un ejemplo de ello, supongamos que agregamos a nuestra aplicación una transacción "Contactos" para representar los contactos efectuados por un analista a un

proveedor pidiendo cotizaciones. Nos interesa mantener un histórico de tales contactos.

Para ello, definimos la siguiente estructura:

*CtoNro**	Nro de contacto
PrvCod	
PrvNom	
AnlCod	
AnlNom	
CtoFch	Fecha del contacto

Si hacemos un diagrama de Bachman que represente las tablas PEDIDOS, ANALISTAS, PROVEEDORES y CONTACTOS, tenemos:

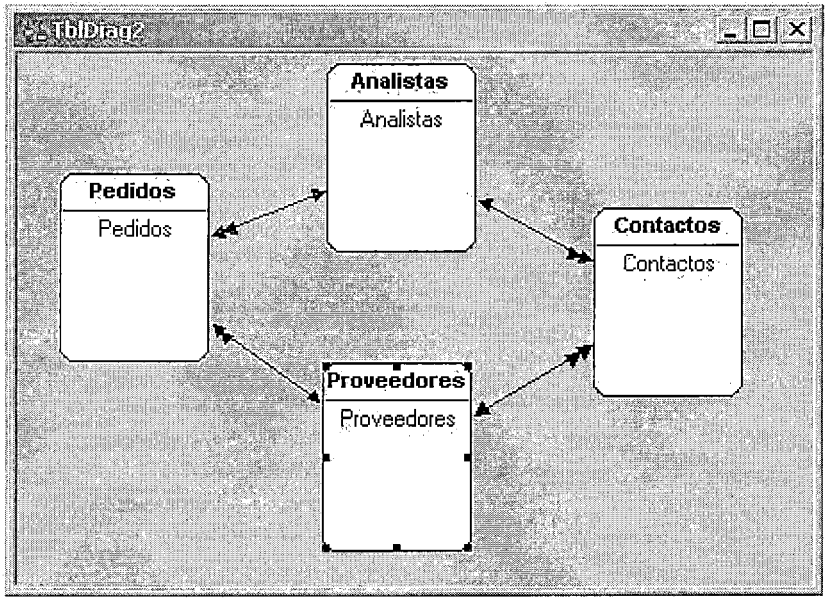


Fig. 36: Diagrama de Bachman

Ahora existen dos tablas extendidas mínimas que contienen a los atributos *PrvNom* y *AnlNom* del For Each: la que tiene como tabla base PEDIDOS y la que tiene como tabla base CONTACTOS.

Pero nosotros queremos listar los nombres de proveedor y analista de cada pedido, y no de cada contacto. Para resolver la ambigüedad que aparece, se utiliza la cláusula DEFINED BY dentro del comando FOR EACH. Así, en el ejemplo agregaremos:

```
For each
DEFINED BY PedFch
        print NombreProveedorAnalista
Endfor
```

En el DEFINED BY se debe hacer referencia a por lo menos un atributo de la tabla base deseada. Aún en los casos en los que no se da el problema anterior, se recomienda el uso del DEFINED BY para reportes más o menos complejos porque mejora bastante el tiempo de especificación del reporte.

3.2. Cortes de Control

Un caso particular de FOR EACHs anidados son los llamados cortes de control. Supongamos que queremos listar los pedidos ordenados por fecha:

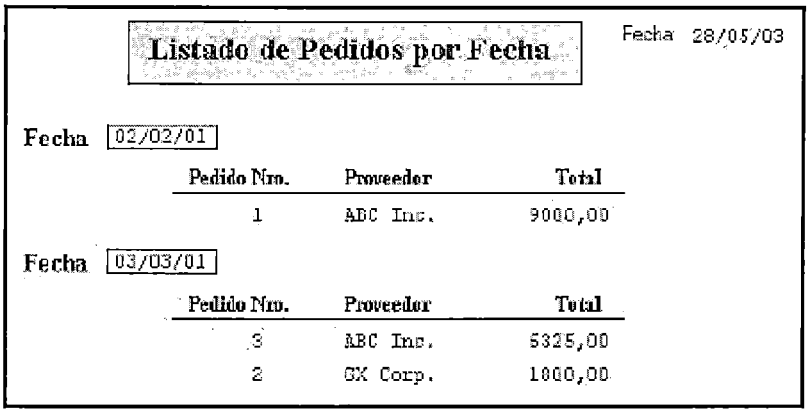


Fig. 40: Listado de Pedidos por Fecha

El Layout nos queda:

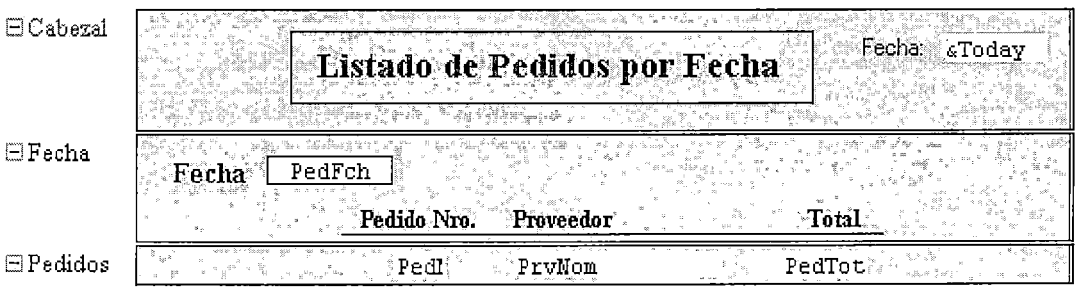


Fig. 41: Layout del "Listado de Pedidos por Fecha"

el Source es:

```
header
        print Cabezal
end
for each order PedFch
        print Fecha
        for each
                print Pedidos
        endfor
endfor
```

La tabla base del primer FOR EACH es PEDIDOS (pues el único atributo que aparece es *PedFch*) y la del segundo FOR EACH también (pues contiene *PedNro, PrvNom y PedTot*). Este es un caso de corte de control sobre la tabla PEDIDOS y GeneXus lo indica en el Listado de Navegación con la palabra BREAK -en lugar de FOR EACH- para el For Each anidado. Los cortes de control pueden ser de varios niveles. Por ejemplo, si quisiéramos listar los pedidos por fecha y dentro de cada fecha por proveedor:

```
For each order PedFch
        ....
        For each order PrvCod
            ...
            For each
                ....
            endfor

        endfor

endfor
```

Cuando se definen cortes de control es MUY IMPORTANTE indicar el ORDEN en los FOR EACH ya que es la forma de indicar a GeneXus por qué atributos se quiere realizar el corte de control.

For Eachs paralelos

También se pueden definir grupos FOR EACH paralelos, por ejemplo, si se desean listar los proveedores y luego los analistas en el mismo listado, implementamos el siguiente Source:

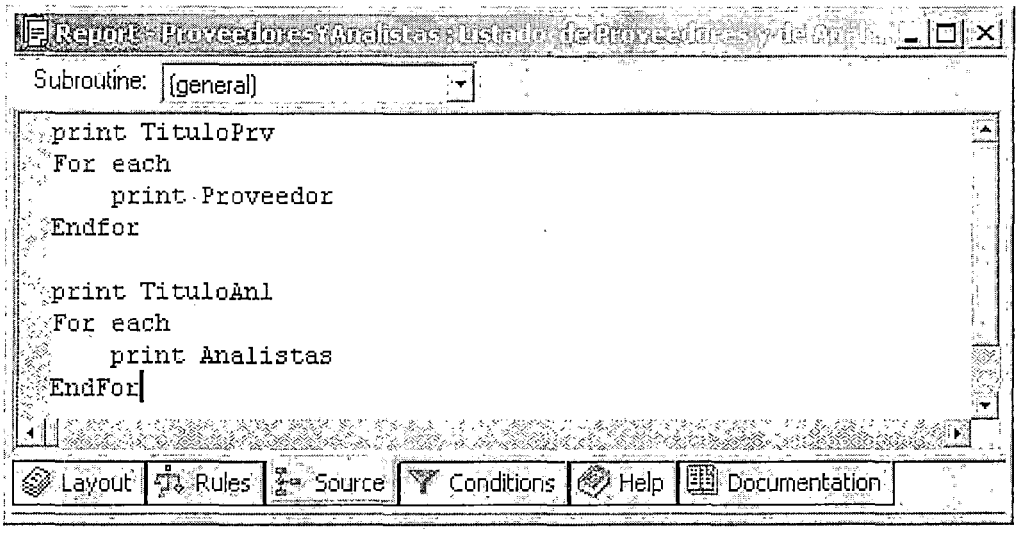


Fig. 42: Source del "Listado de Proveedores y de Analistas"

Siendo el Layout:

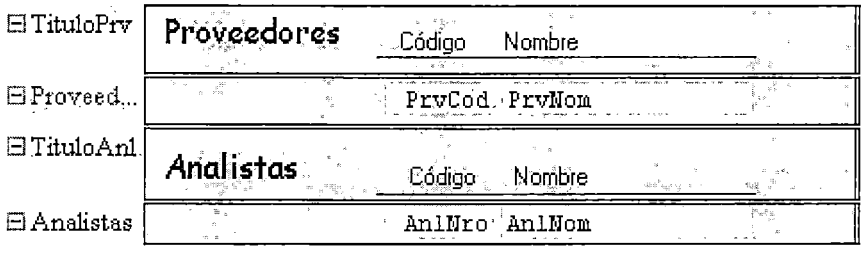


Fig. 43: Layout del "Listado de Proveedores y de Analistas"

Los listados con FOR EACHs paralelos son muy prácticos cuando se quiere hacer listados del tipo "Listar toda la Información que se tenga sobre un Proveedor", en donde el Source será del tipo:

```
For each
    ...     //Información del Proveedor
    For each
        ....    // Información sobre Pedidos
    Endfor
    For each
        ....    // Información sobre Precios
    Endfor
Endfor
```

3.3. Definición de Variables

Una variable es reconocida por GeneXus por ser un nombre que comienza con &, por ejemplo *&Today* o *&Aux*.

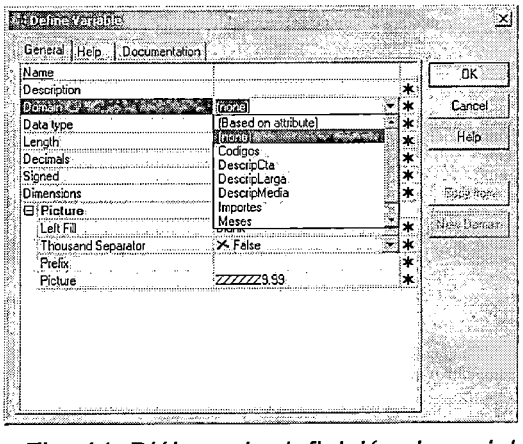


Fig. 44: Diálogo de definición de variable

Las variables se pueden definir también en función de otros atributos, usando la opción "Based on attribute" que aparece en el combo asociado al dominio (Domain). Se recomienda utilizar esta opción siempre que sea posible. De esta manera, cada vez que cambie el tipo de datos del atributo en el cuál una variable está basada, automáticamente el cambio será propagado también a la variable.

Asignación

Existen varios comandos (=, +=, -=, *=, /=) que permiten asignar valores a variables (en los procedimientos también se permite la asignación de valores a atributos, como veremos oportunamente). Por ejemplo si queremos mostrar totales en un listado:

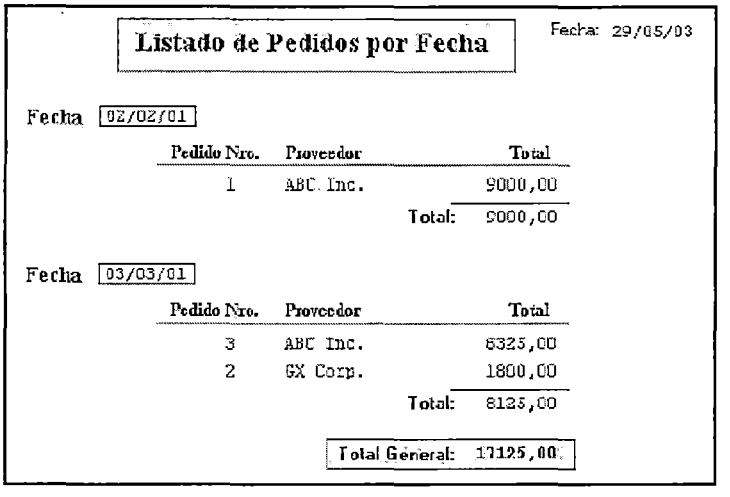


Fig. 45: Listado de Pedidos por Fecha en ejecución

El Layout es:

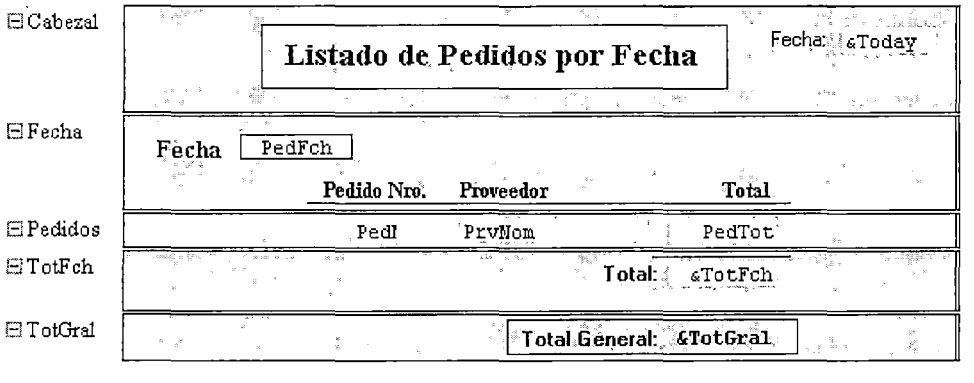


Fig. 46: Layout del reporte "Listado de Pedidos por Fecha"

y en el Source tendremos:

```
header
        print Cabezal
end
```

```
&TotGral = 0
for each order PedFch
        &TotFch = 0
        print Fecha
        for each
                &TotFch += PedTot
                print Pedidos
        endfor
        print TotFch
        &TotGral += &TotFch
endfor
print TotGral
```

3.4. Llamadas a otros Programas y a Subrutinas

Con el comando Call se puede llamar a otro programa. Este puede ser otro programa GeneXus o un programa externo. El formato del comando es:

Call(Program_name, Parameter1, ..., Parametern)

Donde *Program_name* es el nombre del programa llamado y *Parameter1* a *Parameter n* son los parámetros que se envían. Estos parámetros pueden ser atributos, variables y constantes. Los parámetros pueden ser actualizados en el programa llamado.

Cuando desde un reporte se llama a otro reporte que también imprime, es importante pasarle como parámetro la variable predefinida *&Line* (que contiene el número de línea que se está imprimiendo) para que el reporte llamado no comience en una página nueva.

Con el comando Do se llama desde el código principal del reporte a una subrutina que debe estar definida mediante el comando Sub, al final del Source. En este caso no son necesarios los parámetros porque están disponibles para la subrutina los mismos datos (variables) que están disponibles en el lugar donde se hace el Do.

Así, en el Source tendremos:

```
....
DO 'Nombre-Subrutina'
....

Sub 'Nombre-Subrutina'
        ....
EndSub
```

Condiciones

En esta sección de un reporte se definen las condiciones globales (filtros globales) que se quieren aplicar sobre los datos a recuperar. Es equivalente a la cláusula WHERE del comando FOR EACH (incluso tienen la misma sintaxis), con la diferencia que el WHERE se aplica al FOR EACH en el que se encuentra y las condiciones definidas aquí se aplicarán a TODOS los FOR EACHs que correspondan. En el Listado de Navegación se indica a qué FOR EACH se están aplicando.

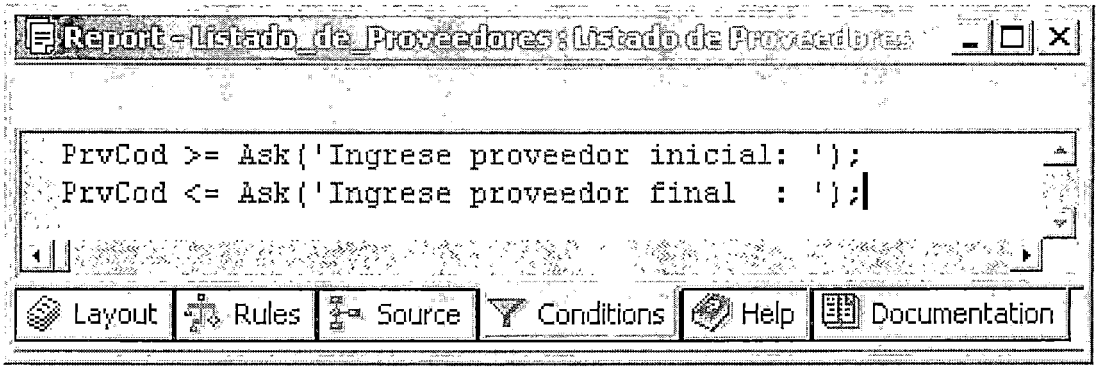


Fig. 47: Sección "Conditions" del reporte "Listado de Proveedores"

Cuando se ejecute el reporte aparecerá la siguiente pantalla:

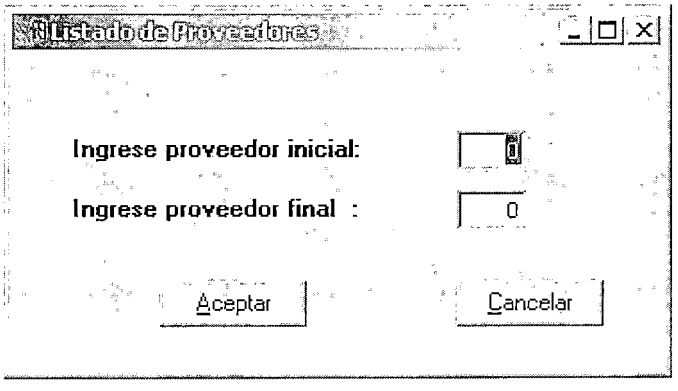


Fig. 48: Pantalla de ingreso de rango de proveedores para el listado

Donde se acepta el rango de proveedores a listar. El valor digitado por el usuario se almacena en las variables &ask1, &ask2, etc., según el orden de aparición de los ask en las condiciones. Estas variables son útiles para imprimir cuál fue el valor digitado.

CAPÍTULO IV

4. OBJETO WORK PANEL

Con Work Panels (paneles de trabajo) se definen las consultas interactivas a la base de datos, en ambiente Windows. Si bien este es un objeto muy flexible que se presta para múltiples usos, es especialmente útil para aquellos usuarios que utilizan el computador para pensar o para tomar decisiones. La forma de programar los work panels está inspirada en la programación de los ambientes gráficos, especialmente la programación dirigida por eventos.

Elementos

Para cada work panel se puede definir:

Form. Al igual que las transacciones, se debe definir el form, que será la interfaz con el usuario. Pero a diferencia de las transacciones, los work panels son objetos que solo se utilizan en ambiente Windows, por lo que el form de un work panel es análogo al form GUI-Windows de una transacción. Para programar una pantalla Web para consulta de datos, se utilizará el objeto web panel, que se mencionará más adelante.

Eventos. Aquí se define la respuesta a los eventos que pueden ocurrir durante la ejecución del work panel. Por ejemplo, qué respuesta dar cuando el usuario presiona Enter o un botón, etc.

Condiciones. Define qué condiciones deben cumplir los datos para ser presentados en la pantalla. Son equivalentes a las Condiciones de los reportes y procedimientos.

Reglas/Propiedades. Definen aspectos generales del work panel, como los parámetros que recibe, etc.

Ayuda. Texto para la ayuda a los usuarios en el uso del work panel.

Documentación. Texto técnico para ser utilizado en la documentación del sistema.

Solapas de acceso a los elementos

Como para todo objeto GeneXus, se puede acceder a varios de los elementos que lo componen utilizando las solapas que aparecen bajo la ventana de edición del objeto:

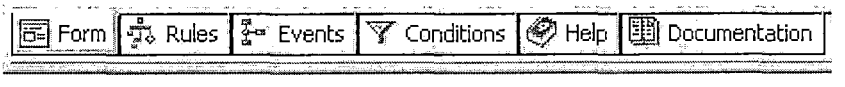


Fig. 49: Solapas que figuran en la parte inferior de la ventana de edición de un work panel

En este capítulo solo se presentarán las características y usos más salientes de los work panels. Un estudio más profundo será realizado en el curso GeneXus.

Ejemplos de desarrollo

Para ilustrar el uso de este tipo de objetos, vamos a realizar una serie de consultas encadenadas. Estas consultas comienzan con un work panel de proveedores, donde se despliegan todos los proveedores y el usuario selecciona uno para luego realizar alguna acción con él. En este caso para cada proveedor seleccionado se podrá:

- Visualizar los datos del proveedor
- Visualizar los pedidos del proveedor

Al work panel lo llamaremos "Trabajar con Proveedores" dado que es precisamente lo que buscamos: trabajar -efectuar alguna acción- con los proveedores.

El form en ejecución se verá:

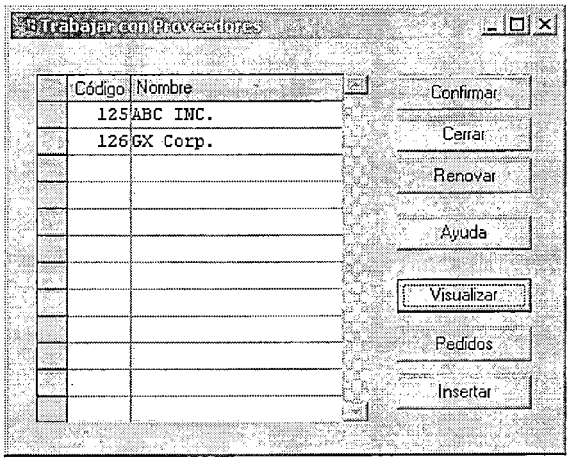


Fig.50: Form de "Trabajar con Proveedores" en ejecución

Aquí se muestra el código y nombre de todos los proveedores del sistema, en la grilla. Luego el usuario puede posicionarse sobre una línea y hacer algo con ese proveedor.

Por ejemplo, si el usuario presiona el botón de "Visualizar" en el proveedor 125, se abre una ventana donde se muestran los datos de ese proveedor:

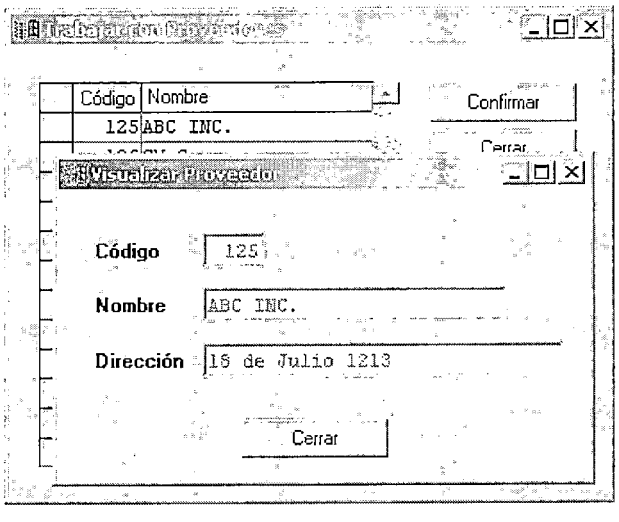


Fig. 51: Visualizar datos de un proveedor

Análogamente, si se presiona el botón "Pedidos" se abrirá la siguiente pantalla que muestra los pedidos de ese proveedor:

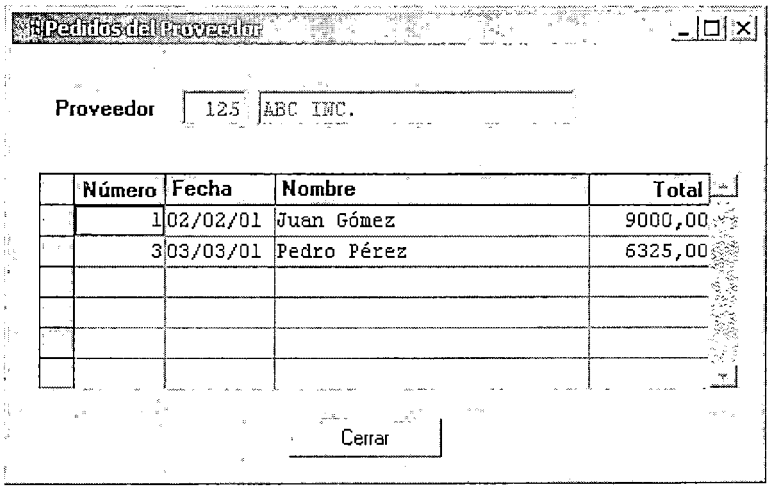


Fig. 52: Visualizar pedidos de un proveedor

A su vez este work panel podría extenderse, permitiendo seleccionar algún pedido para visualizar qué artículos lo componen y así sucesivamente.

Las acciones que vimos en el primer work panel se aplican a una línea, pero existen otras acciones que no se aplican a ninguna línea en particular, por ejemplo, si queremos insertar un nuevo proveedor. Si se elige esta acción (presionando el botón "Insertar" que figura en el form) se pasa el control a la transacción de proveedores para permitir el ingreso de un nuevo proveedor:

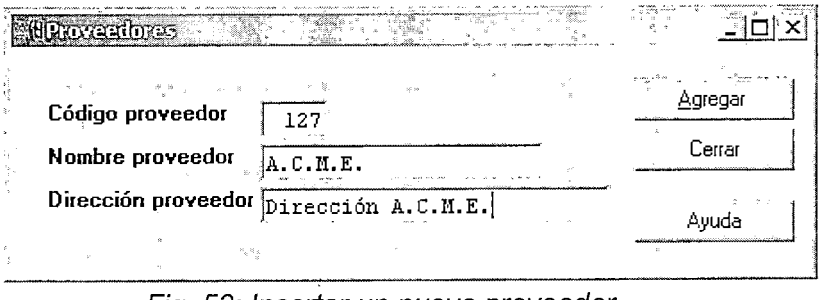


Fig. 53: Insertar un nuevo proveedor

Botón "Renovar"

Es incluido automáticamente por GeneXus en el form de todo work panel y la consecuencia de presionarlo es que se vuelven a cargar todas las líneas del grid. Como veremos más adelante existen varios casos en los que puede resultar necesario "renovar" o "refrescar" los datos cargados. Por ejemplo cuando se agrega un nuevo proveedor al sistema. Más adelante nos detendremos en esta acción.

Veamos ahora cómo se define el primer work panel del ejemplo.

Form

El form de un work panel se define de forma similar al form GUI-Windows de una transacción.

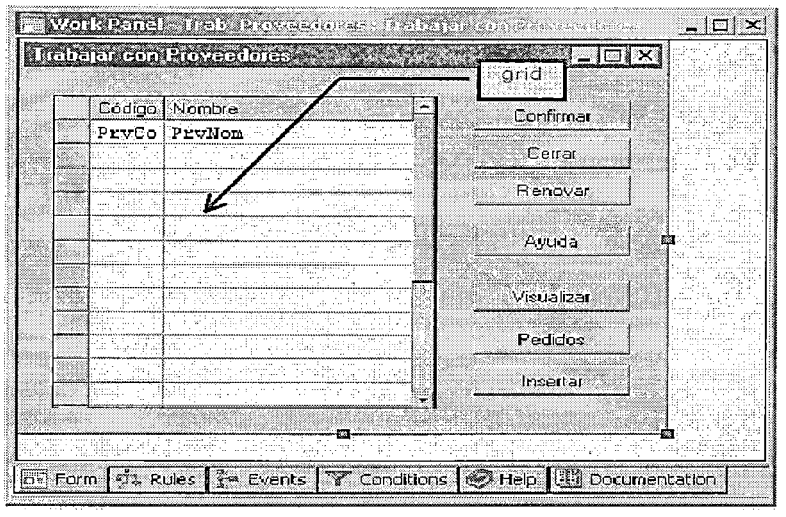


Fig. 54: Form del work panel "Trabajar con Proveedores"

Cuando se define un grid en el form (se inserta como cualquier otro control, utilizando la paleta de herramientas "Controls") se está indicando que se va a mostrar una cantidad indefinida de datos (en este caso, los proveedores). Si bien sólo se pueden ver simultáneamente las líneas presentes en la pantalla, GeneXus provee todos los mecanismos para que el usuario se pueda "mover" (llamaremos "paginar") dentro de la lista.

El work panel anterior contiene un grid cuyas columnas corresponden a los atributos *PrvCod* y *PrvNom*. Los grids pueden contener atributos, variables y elementos de arrays de una y dos dimensiones. A diferencia de las transacciones, aquí los atributos serán de salida, es decir, se utilizan para mostrar información de la base de datos, y las variables de entrada (se utilizan fundamentalmente para solicitar información al usuario).

Grid

El ícono de la paleta de herramientas que se muestra a continuación, nos va a permitir insertar un grid en el form del work panel y definir las caracterísitcas del mismo, como los atributos y variables que conformarán las columnas del grid, los títulos de tales columnas, etc.

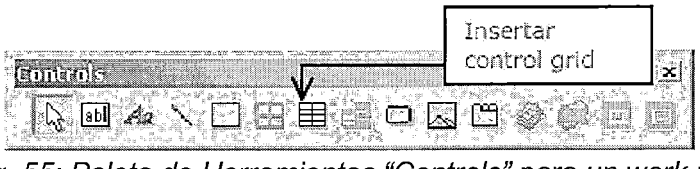


Fig. 55: Paleta de Herramientas "Controls" para un work panel

Una vez insertado el control en el form, se abre automáticamente el siguiente diálogo, donde se especifican las columnas:

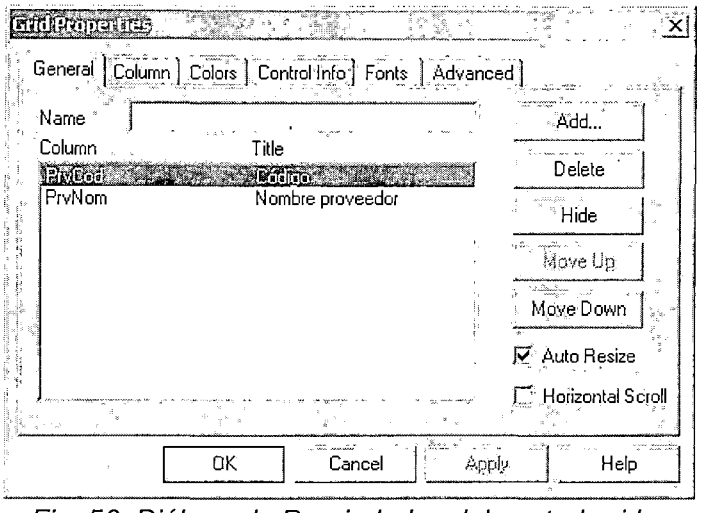


Fig. 56: Diálogo de Propiedades del control grid

En "Name" se permite dar un nombre al control, lo que será necesario si luego se le quieren asociar propiedades, eventos o métodos al mismo. Si se van a incluir varios grids dentro del form, entonces será indispensable identificarlos, por lo que se vuelve fundamental asignar valor a estar propiedad para cada uno de los grids insertados.

Para seleccionar los atributos y variables que van a ser incluidos en el grid se debe presionar el botón "Add".

El botón "Hide" nos permite ocultar la columna seleccionada. Esto resultará necesario cuando no se quieran mostrar en pantalla determinados datos, pero sea indispensable tenerlos cargados, para poder luego utilizar sus valores cuando se seleccione una línea y se quiera efectuar alguna acción sobre la misma.

Los botones "MoveUp" y "Move Down" nos van a permitir cambiar el orden en que se van a desplegar los atributos en el grid.

Con la solapa "Column" vamos a poder cambiar los títulos de las columnas (por defecto, para un atributo toma el valor de la opción "Title Column" del atributo, ver fig.5 del capítulo 2) y en caso de tratarse de una variable no escalar, podremos seleccionar el elemento de la variable que queremos utilizar en esa columna, indicando valores de fila y columna.

Con las solapas "Colors" y "Fonts" podemos cambiar los colores y el tipo de letra de las columnas y de las filas.

Con la solapa "Control Info" se selecciona el tipo de control a utilizar para cada atributo o variable del grid, y dado el tipo, indicar la información necesaria para ese tipo. Las opciones disponibles son: Edit, Check box, Combo y Dynamic Combo (o el default del atributo o variable de la columna).

La solapa "Advanced" nos va a permitir configurar ciertas propiedades que tienen que ver con la carga del grid. Si no se configuran, entonces valen para ese grid las opciones a nivel del objeto, que son las que figuran bajo la categoría "Loading" en el diálogo de propiedades del work panel, que se ilustra más adelante, en la fig.77.

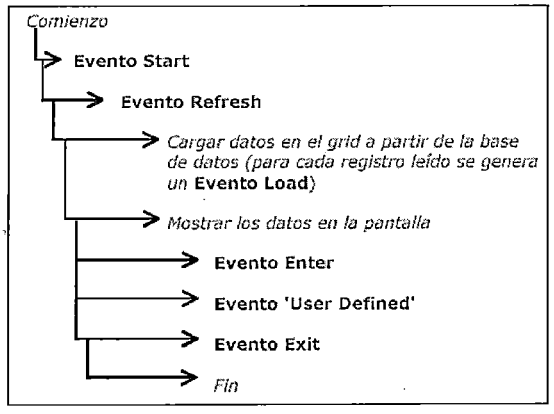
Podemos también ajustar manualmente el ancho de cada columna desmarcando la opción de "Auto Resize". Si esta opción queda marcada, el ancho de la columna es determinado por GeneXus en función del largo del atributo o variable y del título de la columna.

El check box "Horizontal Scroll" permite que en tiempo de ejecución aparezca una barra de scroll horizontal (además de la de scroll vertical que aparece automáticamente).

4.1. Eventos

La forma tradicional de programar la interfaz entre el usuario y el computador (incluidos los lenguajes de uarta generación) es subordinar la pantalla al programa. Es decir, desde el programa se hacen "calls" a la pantalla. En work panels es lo opuesto: el programa está subordinado a la pantalla, ya que una pantalla realiza "calls" al programa para dar respuesta a un evento.

Existen algunas diferencias entre work panels con un solo grid y work panels con múltiples grids, fundamentalmente con respecto a los eventos y a la determinación de las tablas que deben navegarse para recuperar los datos. En este libro trataremos el primer caso, es decir, aquel en el que el work panel cuenta a lo sumo con un grid, y dejamos el estudio de múltiples grids para el curso. La discusión que sigue toma en cuenta, pues, el caso de un solo grid.
La estructura de eventos es, en forma resumida, la siguiente:



Para programar cada uno de los eventos se utiliza el mismo lenguaje que vimos en reportes y procedimientos, aunque no están permitidos los comandos relacionados con la impresión (Header, Footer, etc.) ni los de actualización de la base de datos (New, Delete, etc.). También existen comandos específicos para work panels, como lo son el comando Refresh y el

comando Load.

Evento Start

El evento Start ocurre cuando se abre el work panel.

```
Event Start
        &Fecha = &Today
Endevent
```

En este caso, se utiliza el evento Start para mover la fecha actual a la variable &Fecha, que por ejemplo, se podrá mostrar en la pantalla. Esta técnica es muy útil para inicializar valores por defecto, sobre todo para aquellos que son complejos (por ejemplo, *&Valor* = udf('PX', ...)).

Evento Enter

Este evento se dispara cuando el usuario presiona ENTER (o el botón "Confirmar"). Supongamos que en el work panel "Trabajar con proveedores", deseamos agregar la posibilidad de eliminar un conjunto de proveedores. Podemos implementar esto definiendo una nueva columna en el grid, compuesta por una variable, &op, para que el usuario ingrese allí el tipo de operación que quiere efectuar sobre cada línea. Supongamos que un valor "4" en la variable &op significa que se quiere eliminar ese proveedor. Programaremos el evento "Enter" de manera tal de recorrer el grid y para cada línea que tenga el código de operación "4" llamar a un procedimiento que borre ese proveedor. El código asociado será:

```
Event Enter
        For each line
            If &op = '4'
                Call(PDltPrv, PrvCod)
            else
                If &op <> ' '
                    Msg(concat(&op, ' no es una opción válida') )
                Endif
            Endif
        EndFor
endevent
```

El evento Enter es un poco más extenso y vale la pena detenernos en él. En primer lugar se utiliza el comando FOR EACH LINE que realiza un FOR EACH, pero no sobre los datos de la base de datos, sino sobre los datos cargados en el grid.

Para cada una de las líneas del grid, se pregunta por el valor de &op. Si es '4' se llama al procedimiento DltPrv, y en otro caso, se da un mensaje indicando que la opción no es válida. Observar que se podrían haber definido más opciones y preguntar por ellas en este evento.

Eventos definidos por el usuario (User Defined Events)

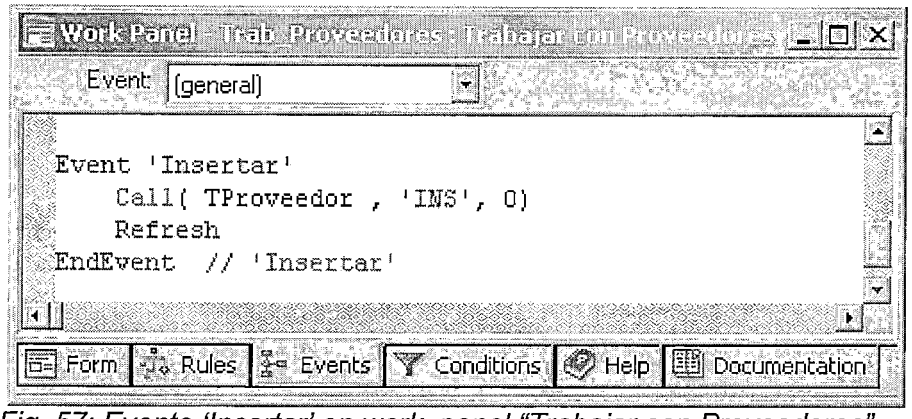


Fig. 57: Evento 'Insertar' en work panel "Trabajar con Proveedores"

Este es un evento definido por el usuario, al que se le debe asociar un nombre, en este caso, 'Insertar'. En este evento se llama a la transacción de proveedores. Luego de llamar a la transacción, se ejecuta el comando Refresh, que indica que se debe cargar nuevamente el grid, ya que se ha adicionado un nuevo proveedor y queremos que éste aparezca en la pantalla. También se podría haber usado el comando con el parámetro keep (Refresh Keep) que hace que una vez que el comando refresh se ejecute, el cursor quede posicionado en la misma línea del grid en la que estaba antes de la ejecución.

Evento Refresh

Los datos presentados en pantalla, son cargados en el grid desde la base de datos cuando se abre el work panel. La interacción con la base de datos se da en el momento de la carga, y luego se trabajará sobre los datos cargados.

En un ambiente multi-usuario, los datos de una pantalla pueden haber quedado desactualizados si desde otra terminal fueron modificados. Cuando el usuario desea actualizar los datos, debe dar clic sobre el botón Refresh ("Renovar" en español), o presionar la tecla F5. El efecto es que se vuelve a cargar el grid, accediendo nuevamente a la base de datos.

El evento Refresh (evento del sistema) ocurre en el instante de tiempo que antecede a la carga del grid. Inmediatamente después de producido, se accede a la base de datos y se carga el grid. Por lo tanto, como consecuencia del disparo del evento Refresh, los datos son cargados y mostrados en pantalla.

Tenemos, por lo tanto, que el grid se carga (es decir, ocurre el evento Refresh) cuando:

- Se carga la pantalla - por lo tanto después del evento Start hay un evento Refresh.
- El usuario hizo clic sobre el botón Refresh (o presionó la tecla F5).
- Se ejecutó el comando Refresh en otro evento.
- El usuario modificó el valor de alguna de las variables de la parte fija del form, que son

utilizadas como condiciones sobre los datos a cargar (esto lo veremos unas páginas más adelante)

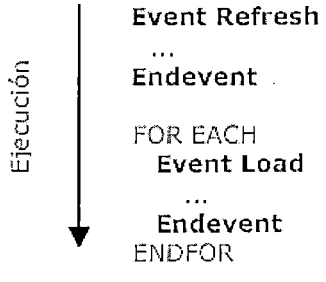
4.2. Carga de datos en el work panel

Como vimos en el diagrama de eventos, después del evento Refresh, se carga el grid con los datos obtenidos de la base de datos. Para determinar de qué tablas se deben obtener los datos, GeneXus utiliza el mismo mecanismo descripto para los grupos FOR EACH en el capítulo de reportes. ¡Pero en este caso no hay ningún FOR EACH!. Sin embargo, generalmente se considera que cada work panel con un grid tiene un FOR EACH implícito7, que contiene los siguientes atributos:

- Todos los atributos que se encuentran en el form (incluso los que se encuentran en la parte fija del mismo).
- Todos los atributos que se encuentren en los Eventos, con la excepción de aquellos que se encuentren dentro de un FOR EACH explícito.
- Todos los atributos del grid que estén ocultos, es decir, aquellos para los que se presionó el botón "Hide" en la solapa "General" del diálogo de propiedades del grid.
- Todos los atributos del grid que se hayan insertado en la pantalla "Grid Order" a la que se accede mediante la opción "Order" del menú pop up que se despliega al hacer botón derecho sobre el grid, como veremos luego

Con estos atributos, GeneXus determina cuál es la tabla extendida correspondiente, de la misma manera en que lo hacía con los atributos que extraía de un For Each.

Luego recorre la tabla base y accede a las tablas de la extendida, para recuperar los valores de los atributos que aparecen en el grid, tanto visibles como ocultos, de la misma manera en que lo hacía con un print block dentro de un For Each en un reporte.
Para cada uno de los registros encontrados, se genera el evento del sistema Load. Así, la relación entre el evento Refresh, el FOR EACH implícito y el evento Load es la siguiente:



Los datos que se leen en este FOR EACH implícito se cargan en el grid, que es el que se presenta en pantalla. Esta carga se realiza "bajo pedido", es decir que al comienzo sólo se carga la primera página y a medida que el usuario requiere más información, se van cargando

las siguientes páginas. Existe una propiedad para indicar que se quiere cargar el grid de una sola vez si fuera necesario (la propiedad "Load Records"). Igual que en los reportes y procedimientos, en el Listado de Navegación del work panel se indica qué tablas se están utilizando, índices, etc.

Puede darse el caso de que habiendo un grid en el form, no exista el FOR EACH implícito. Esto ocurre cuando no hay atributos ni en el form, ni en los eventos fuera de For Each, ni a nivel del grid, en las opciones "Hide" y "Order". Es decir, solo aparecen variables en esos lugares.

En este caso se genera sólo UN evento Load (en lugar de uno por cada línea a cargar en el grid) y es en éste donde se deben cargar los datos en el grid (utilizando el comando Load).

Por ejemplo, supongamos que queremos implementar un work panel donde se muestren todos los proveedores y por cada uno el total de los pedidos efectuados al mismo. Una forma de hacerlo es utilizando variables para cargar los datos en el grid:

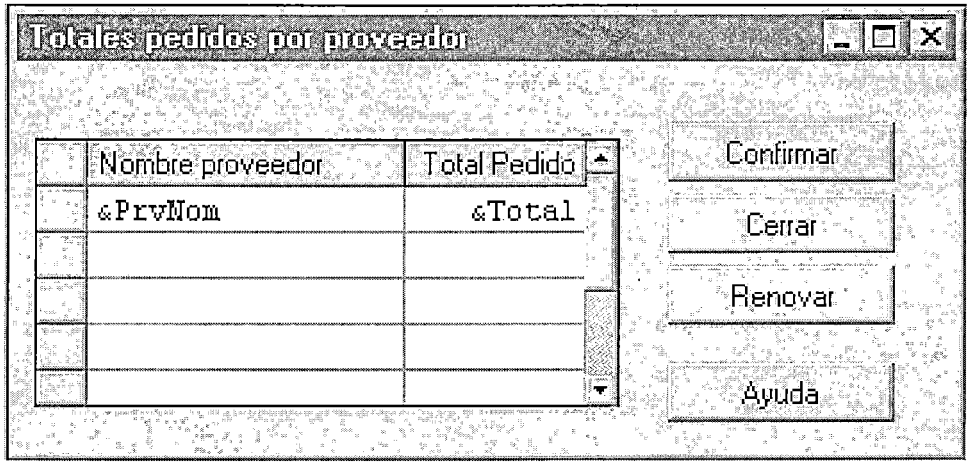


Fig. 58: Form de work panel "Totales por proveedor"

y programamos el evento Load:

```
Event Load
    For each
        &PrvNom = PrvNom
        &Total = 0
        for each
            &total += PedTot
        endfor
        Load
    EndFor
endevent
```

Ocurrirá por tanto el evento Refresh, e inmediatamente después el evento Load, una sola vez. La carga de cada línea en el grid se realiza con el comando Load y la iteración se realiza con el For Each programado por el analista.

Dependiendo de la plataforma de implementación, se mantiene o no la facilidad de carga "bajo pedido" para este caso en el que no hay tabla base. En .NET y Java se mantiene. Sin embargo en Visual Basic y Visual FoxPro, antes de mostrar los datos, se carga toda la información del grid.

Condiciones

En esta sección se definen las restricciones sobre los datos a recuperar de la misma forma que en los reportes. Lo interesante de las condiciones en los work panels, es que en las mismas se pueden utilizar variables que se encuentran en la pantalla, de tal manera que el usuario, cambiando los valores de estas variables cambia los datos que se muestran en el grid sin tener que salir de la pantalla.

Por ejemplo, si queremos visualizar sólo un rango de proveedores, agregamos en el form las variables, &*PrvIni* y &*PrvFin*:

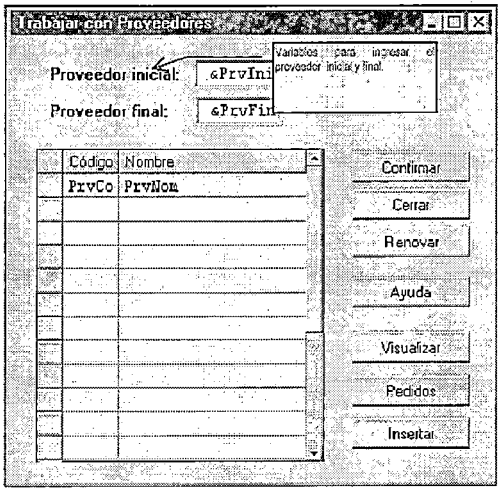


Fig. 59: Form de work panel para trabajar con los proveedores del rango seleccionado por el usuario

Y definiendo las condiciones:

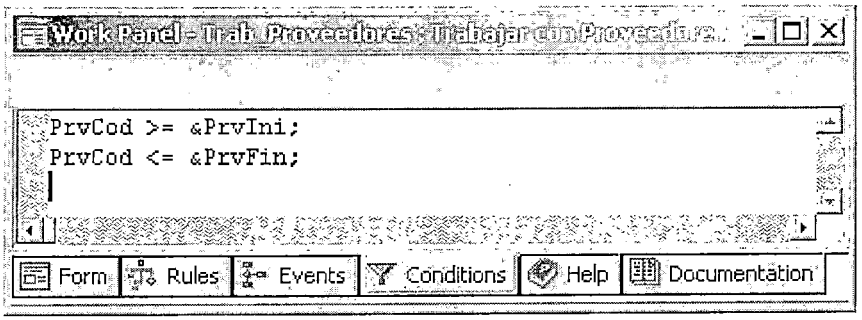


Fig. 60: Condiciones para filtrar los proveedores que se muestran en el grid

se obtiene el efecto deseado.

Observemos que cada vez que el usuario modifique una de estas dos variables que aparecen en las condiciones, deberá dispararse el evento Refresh para volver a cargar el grid. Esto es realizado automáticamente.

4.3. Orden de los datos en el grid

Cuando en un reporte queríamos ordenar un For Each por determinado conjunto de atributos, utilizábamos la cláusula Order. En el caso de un work panel en el que tenemos un For Each implícito, ¿dónde podemos decir que queremos recuperar los datos en el grid con determinado orden? Por ejemplo, si queremos ver los proveedores por orden alfabético, ¿cómo especificamos este orden?

Esto se logra haciendo botón derecho sobre el grid y seleccionando la opción "Order" que aparece en el menú que se abre:

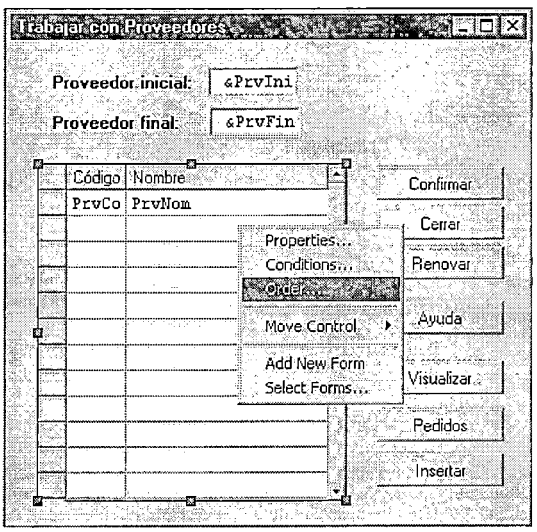


Fig. 61: Orden de los datos en el grid

Se abrirá una pantalla que permite insertar los atributos por los que se quiere ordenar el grid (en nuestro caso elegiremos *PrvNom*)

Es equivalente a la cláusula ORDER del FOR EACH y se aplica todo lo dicho sobre el tema en el capítulo sobre el objeto reporte.

Reglas

Valen la mayoría de las reglas de los reportes, más algunas particulares:

Noaccept

A diferencia de las transacciones, en los work panels ningún atributo es aceptado (siempre se

muestra su valor pero no se permite modificarlo). Para variables se sigue el siguiente criterio:

- todas las variables presentes en la pantalla son aceptadas, a no ser que tengan la regla NOACCEPT.

Por ejemplo, en un work panel en el que deseemos mostrar la fecha en el form, teniendo la fecha cargada en la variable &Fecha, escribimos la siguiente regla, para que el valor del campo no se permita modificar:

noaccept(*&Fecha*);

Search

Cuando el grid tiene muchas líneas cargadas, a menudo se quiere brindar la posibilidad de que el usuario pueda "posicionarse" en alguna línea determinada del mismo en forma directa, sin tener que avanzar página por página haciendo scroll. Por ejemplo, si en nuestro work panel de proveedores queremos que exista la posibilidad de buscar un proveedor según su nombre, se debe definir la regla: search(*PrvNom* >= *&Nom*);

y en la pantalla se debe agregar la variable *&Nom*:

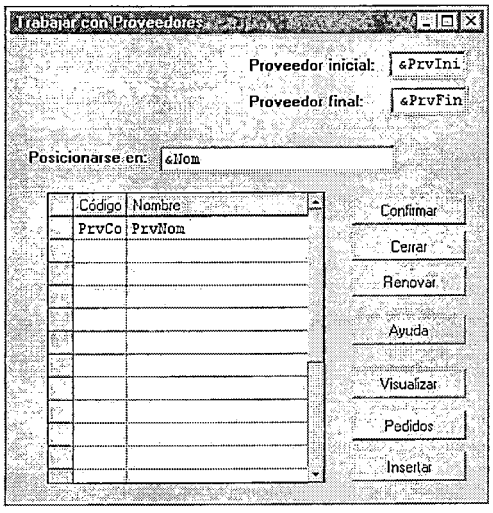


Fig. 62: Form de "Trabajar con Proveedores" con posibilidad de posicionarse en una línea dada

De esta manera, cada vez que el usuario digite algo en &Nom, luego de dar Enter el cursor quedará posicionado en el primer proveedor con *PrvNom* mayor o igual al digitado. En los casos de nombres, es muy usual utilizar el operador like:

search(*PrvNom* LIKE *&Nom*);

en donde el cursor se posiciona en el primer proveedor cuyo PrvNom contenga a &Nom (en el ejemplo si en &Nom se digitó 'X' se posicionará en 'GX Corp.').

4.4. Bitmaps

Los bitmaps son utilizados usualmente para mejorar la presentación de las aplicaciones. Se pueden incorporar en los Forms de las transacciones y work panels, así como en el Layout de

reportes y procedimientos. GeneXus provee dos métodos diferentes para agregar un bitmap:

Fixed Bitmaps

Se puede agregar un control bitmap seleccionando el ítem correspondiente de la paleta de herramientas "Controls". Al insertar un control bitmap se abre un diálogo para configurar sus propiedades. La más importante es la ubicación, FileName, del bitmap.

Agreguemos un logo en el work panel que permite visualizar los datos de un proveedor. El nuevo form se verá de la siguiente forma:

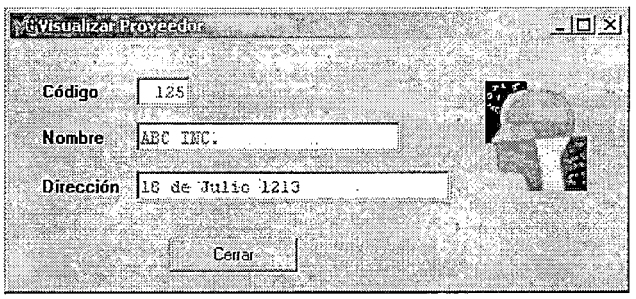


Fig. 63: Work panel "Visualizar Proveedor" con bitmap fijo

En este caso siempre aparecerá el mismo bitmap en el Form, independientemente del proveedor.

4.5. Dynamic Bitmaps

Este método tiene como diferencia que el bitmap lo asignamos a una variable y usando la función Loadbitmap podemos cargar el bitmap que deseemos. Por ejemplo, se puede tener un bitmap que despliegue la foto de cada proveedor.

Definición de una variable de tipo bitmap:

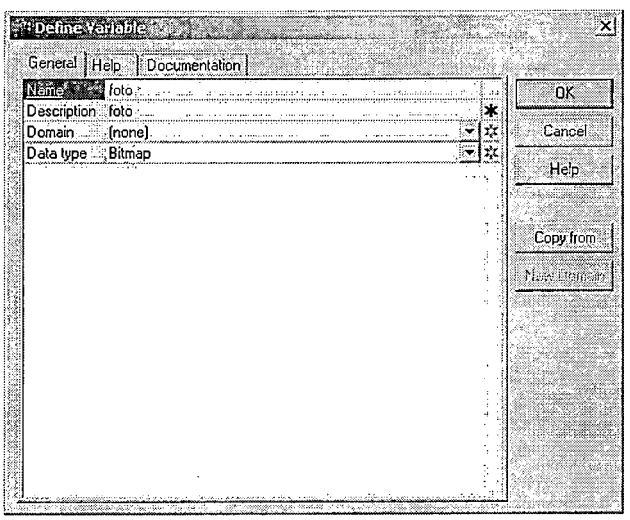


Fig. 64: Definición de variable de tipo Bitmap

En el evento load hay que cargar el bitmap para el proveedor pasado por parámetro. Para poder hacer esto, debemos agregar un atributo, *PrvFoto*, en la transacción de proveedores, que contendrá la ubicación del archivo (bmp, gif, jpg, etc.) con la foto a cargar.

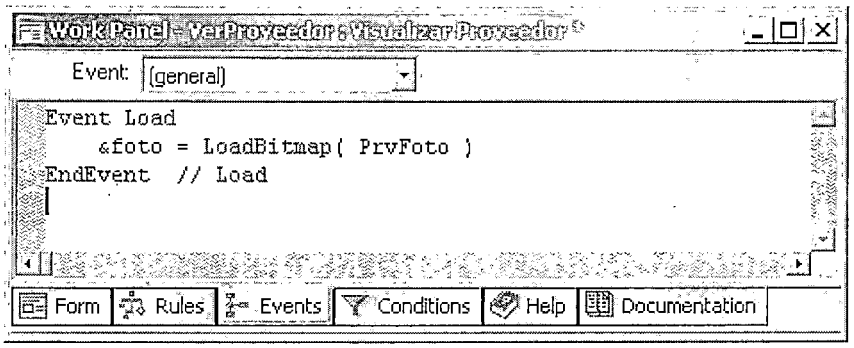


Fig. 65: Evento Load para cargar bitmap dinámico

En tiempo de ejecución se verá de la siguiente forma:

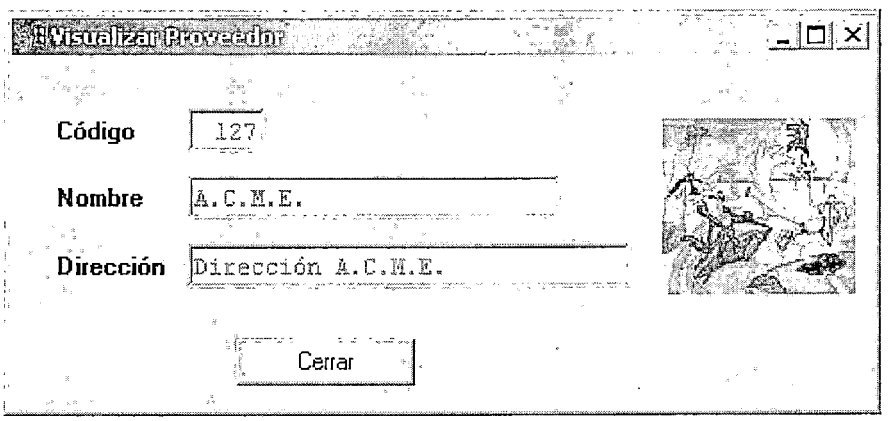


Fig. 66: Visualizar proveedor en ejecución con bitmap dinámico

Propiedades

Como para cualquier otro objeto, se pueden configurar las propiedades de un work panel, accediendo a la siguiente pantalla:

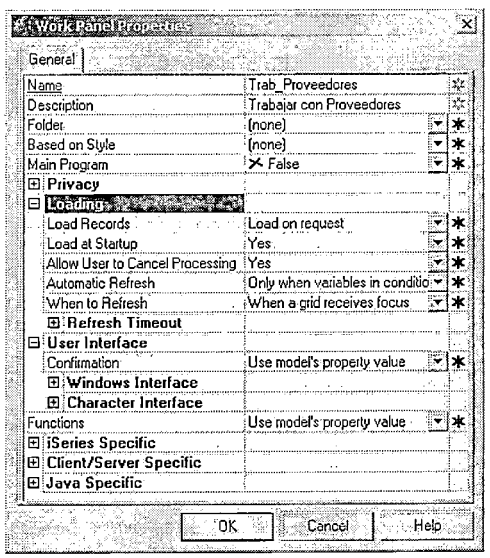


Fig. 67: Diálogo de Propiedades del work panel "Trabajar con Proveedores"

Las propiedades aparecen agrupadas por categorías. Por ejemplo, bajo la categoría "Loading" aparecen las propiedades que tienen que ver con la carga de los datos del work panel. Entre ellas aparece la propiedad "Load Records" que permite especificar si se quieren cargar los datos en el grid a medida que se va haciendo scroll (paginando), o si se quieren cargar todos al comienzo. Si la cantidad de datos a cargar es muy grande, entonces el valor de esta propiedad producirá una diferencia significativa en el tiempo de respuesta.
Si bien cada grid tiene la posibilidad de definir estas mismas propiedades de carga, si un work panel tiene múltiples grids, y en todos queremos especificar las mismas propiedades, la forma más simple es hacerlo a nivel del objeto, en lugar de tener que hacerlo grid por grid.

Existen otras propiedades que tienen que ver con la interfaz y son las que se presentan bajo el grupo "User Interface".

5. ANEXOS

Anexo A. Modelos de datos relacionales

El propósito de este anexo es explicar una serie de conceptos sobre bases de datos relacionales relevantes para el uso de GeneXus.

Tablas

En una base de datos relacional la única forma de almacenar información es en tablas. Una tabla es una matriz (tiene filas y columnas) con un nombre asociado. A las columnas les llamamos atributos, y a las filas registros o tuplas. Por ejemplo:

PROVEEDORES	*PrvCod*	*PrvNom*	*PrvDir*
	125	ABC Inc.	18 de julio 1213
	126	GX Corp.	Br. Artigas 980

Una tabla se diferencia de un archivo convencional porque debe cumplir las siguientes reglas:

- Todos los atributos representan la misma información para cada una de las filas (o registros). Es decir en el atributo PrvNom se almacenan los nombres de los proveedores, y no puede ocurrir que para algunos proveedores en ese atributo se almacene la dirección del mismo. En la práctica esta regla implica que no existen tablas con diferentes tipos de registros.
- No existen dos registros con exactamente la misma información.
- El orden de los registros no contiene información. Es decir se pueden reordenar los registros sin perder información.

Clave primaria y Claves candidatas

Dado que en una tabla no pueden haber dos registros con la misma información, siempre se puede encontrar el atributo o conjunto de atributos cuyos valores no se duplican. Se llama a ese atributo o conjunto de atributos clave primaria *(PK: Primary Key)* de la tabla.

En realidad pueden existir varios de esos conjuntos; a cada uno de ellos lo llamamos clave candidata.

Por ejemplo, si sabemos que además de no haber dos proveedores con el mismo código, tampoco puede haber dos proveedores con el mismo nombre, entonces en la tabla PROVEEDORES tanto PrvCod como PrvNom serán claves candidatas.

En una base de datos relacional para toda tabla debe definirse una de las claves candidatas como clave primaria. También es importante respetar la siguiente regla: no deben existir dos tablas con la misma clave primaria.

Por ejemplo consideremos el siguiente diseño:

PROVEEDORES	*PrvCod*	*PrvNom*

PROV_DIR	*PrvCod*	*PrvDir*

Tanto la tabla PROVEEDORES como PROV_DIR tienen la misma clave primaria8: el atributo *PrvCod*. Esto era relativamente común cuando el criterio de diseño era separar las actualizaciones. Por ejemplo dado que *PrvDir* cambia más que *PrvNom*, entonces era mejor definir un archivo diferente en donde el registro que se modificaba era menor, y por lo tanto la performance mejoraba.

Sin embargo en una Base de Datos Relacional el criterio de diseño es diferente (ver sección de Normalización) y es preferible una sola tabla con los tres atributos:

PROVEEDORES	*PrvCod*	*PrvNom*	*PrvDir*

Integridad Referencial

Son las reglas de consistencia entre los datos de las distintas tablas.
Supongamos que nuestro modelo de datos corresponde a la representación del Sistema de Compras para una cadena de Farmacias. En este sistema queremos representar pedidos de artículos a proveedores, efectuados por los analistas de compras. Para ello necesitamos una tabla PEDIDOS, que contendrá información de los pedidos efectuados por los analistas de compras a los proveedores. Esta tabla podría contener los siguientes atributos: *PedNro, PedFch, PrvCod, AnlNro* y *PedTot* para representar el número de pedido, su fecha, el proveedor al cuál se le realiza el pedido, el analista de compras que efectúa el pedido y el importe total por concepto del pedido, respectivamente:

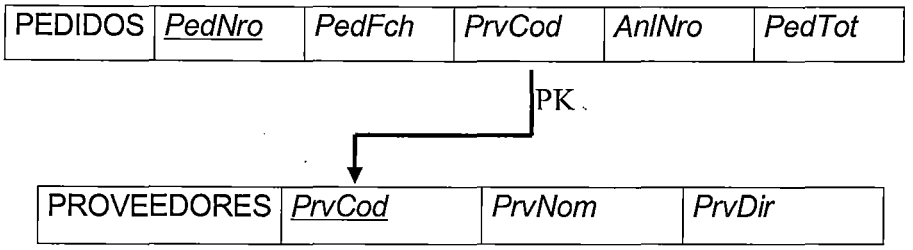
PEDIDOS	*PedNro*	*PedFch*	*PrvCod*	*AnlNro*	*PedTot*

Si observamos las dos tablas: PROVEEDORES definida antes y PEDIDOS, vemos que tienen un atributo en común: *PrvCod*. Queremos indicar con ello, que la información de ambas tablas está relacionada, ya que se trata del mismo *PrvCod* en ambas tablas.

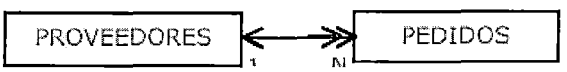
Utilizamos, pues, el concepto de clave foránea *(FK: Foreing Key)*. Decimos que *PrvCod* es una clave foránea (FK) en PEDIDOS a la tabla PROVEEDORES (que tiene a *PrvCod* como clave primaria), con lo que queremos significar que el valor que se ingrese para un registro cualquiera de PEDIDOS en el atributo *PrvCod* (PEDIDOS.*PrvCod*) no puede ser cualquiera: tiene que cumplir que exista un registro en la tabla PROVEEDORES para el que:

PROVEEDORES.*PrvCod* = PEDIDOS.*PrvCod*

o dicho de otra manera: tiene que existir el proveedor.



La relación entre PROVEEDORES y PEDIDOS también puede representarse mediante el siguiente diagrama, que está inspirado en los conocidos Diagramas de Bachman:



En general, en los modelos de datos relacionales deben establecerse explícitamente las claves foráneas de cada tabla. En GeneXus se hace implícitamente: la relación entre estas dos tablas se determina analizando los atributos que tienen en común. Aquí tenemos que el atributo común es *PrvCod*, que es la clave primaria de la tabla PROVEEDORES y se encuentra en la tabla PEDIDOS y, por lo tanto, ambas tablas están relacionadas y la relación es 1-N.
Con relación 1-N se indica que un Proveedor puede tener varios Pedidos asociados y que un Pedido sólo puede tener un Proveedor. También es usual decir que la tabla PROVEEDORES está superordinada a la tabla PEDIDOS, y PEDIDOS está subordinada a PROVEEDORES.

Esta relación implica que los datos de ambas tablas no son independientes y cada vez que se realicen modificaciones en una de las tablas, se deben tener en cuenta los datos de la otra tabla. A estos controles se les llama de integridad referencial y son:

- Cuando se elimina un registro en la tabla superordinada (PROVEEDORES), no deben existir registros asociados (que lo "referencien") en la tabla subordinada (PEDIDOS).
- Cuando se crean/modifican registros en la tabla subordinada (PEDIDOS), debe existir el registro correspondiente ("referenciado") en la tabla superordinada (PROVEEDORES).

Los Diagramas de Bachman tienen la virtud de explicitar las relaciones de integridad referencial del modelo de datos.

Índices

Son vías de acceso eficientes a las tablas. En GeneXus existen cuatro tipos de índices: Primarios, Foráneos, de Usuario y Temporales. A su vez los de Usuario se dividen en "Unique" o "Duplicate".

El índice primario de una tabla es el que se define para la clave primaria y se utiliza para hacer eficiente el control de unicidad de los registros.

Por otro lado, la existencia del índice primario hace posible que se realice eficientemente el segundo control de integridad referencial enunciado más arriba. En el ejemplo visto, la existencia de un índice por *PrvCod* en la tabla PROVEEDORES, hace eficiente el chequeo de que cuando se ingresa/modifica un registro en la tabla PEDIDOS, exista uno en la tabla PROVEEDORES tal que el valor del atributo *PrvCod* del primer registro coincida con el valor del atributo de igual nombre en el segundo registro.

Con GeneXus todos los índices primarios son definidos automáticamente a partir de los identificadores de las transacciones.

Los índices foráneos son utilizados para hacer eficientes los controles de integridad inter-tablas. Se definen para las claves foráneas, por lo que son los que hacen posible que se realice eficientemente el primer control de integridad referencial mencionado anteriormente.

En el ejemplo, la existencia del índice foráneo por *PrvCod* en la tabla PEDIDOS, posibilita que eficientemente se chequee y, en base al chequeo se prohiba, eliminar un registro de la tabla PROVEEDORES si existe uno en la tabla PEDIDOS, para el cual PEDIDOS.*PrvCod* = PROVEEDORES.*PrvCod*. De esta manera se evita dejar referencias colgadas.
Con GeneXus todos los índices foráneos son definidos automáticamente a partir de las claves foráneas, que como mencionamos son identificadas por GeneXus basándose en el nombre de los atributos.

De no tener creados estos dos tipos de índices, todo el sistema de control de la integridad referencial, fundamental para asegurar la consistencia de los datos, sería terriblemente ineficiente. Esto es lo que justifica que GeneXus cree y mantenga en forma automática estos indices.

Los índices de usuario, en cambio, no son creados automáticamente por GeneXus. Son creados a pedido del analista y se definen, fundamentalmente, para recuperar eficientemente los datos con determinado orden. Por ejemplo en la tabla PROVEEDORES se puede definir un índice por PrvNom, que es muy útil para los listados de proveedores ordenados por nombre.

La forma de definir claves candidatas para una tabla (distintas de la clave primaria) en GeneXus es a través de la creación de índices de usuario. Cuando se define una clave candidata en GeneXus, éste controla la unicidad de los valores de los atributos que la

componen, así como lo hace para la clave primaria. Para poder realizar este control eficientemente, GeneXus deberá crear un índice, por lo que directamente, la forma de definir una clave candidata, será crear un índice de usuario, y especificar que los valores deben ser únicos. Por lo tanto, los índices de usuario permiten especificar que los valores serán "Unique" o "Duplicate".

Por último, los índices temporales son creados en algunas plataformas de implementación, cuando quiere recorrerse una tabla por un determinado orden de forma eficiente, pero no quiere crearse un índice de usuario para ello. En este caso, se crea un índice en forma temporal, se utiliza y luego se elimina. Los índices temporales también son creados automáticamente por GeneXus.

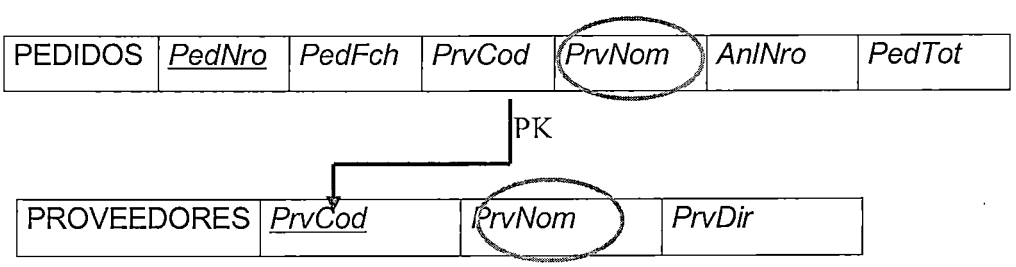
En una base de datos relacional los índices se utilizan sólo por problemas de performance, pero siempre es posible acceder a los datos de una tabla por cualquiera de los atributos de la misma.

Normalización

El proceso de normalización determina en qué tablas debe residir cada uno de los atributos de la base de datos.

El criterio que se sigue es básicamente determinar una estructura tal de la base de datos, que la posibilidad de inconsistencia en los datos sea mínima.

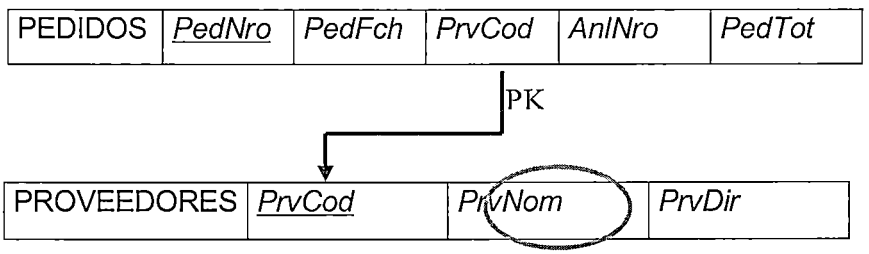
Por ejemplo, si tenemos que almacenar información sobre proveedores y pedidos, se podría tener la siguiente estructura:



en donde vemos que ambas tablas tienen dos atributos en común (*PrvCod* y *PrvNom*). En el caso de *PrvCod* es necesario que se encuentre en ambas tablas dado que es el identificador de los proveedores y por lo tanto debe estar tanto en la tabla PROVEEDORES como clave primaria, como en la tabla PEDIDOS como clave foránea, para indicar a qué proveedor corresponde el pedido.

La situación es diferente para *PrvNom*. No es necesario que esté en la tabla PEDIDOS, porque si conocemos el código de proveedor, *PrvCod*, podemos determinar cuál es su nombre, *PrvNom*. Basta con buscar en la tabla PROVEEDORES por la clave primaria. Si de cualquier manera se almacena *PrvNom* en la tabla PEDIDOS tenemos que esta estructura tiene más posibilidades de inconsistencia que si no estuviera allí.
Por ejemplo si un proveedor cambia su *PrvNom*, el programador debe encargarse de tener un programa que lea todos los pedidos de ese proveedor y le asigne el nuevo *PrvNom*.

Entonces la estructura correcta (diremos "normalizada") será:



El proceso de normalización, que se acaba de introducir de una manera muy informal, se basa en el concepto de dependencia funcional, cuya definición es:

> Se dice que un atributo depende funcionalmente de otro si para cada valor del segundo sólo existe UN valor del

Por ejemplo *PrvNom* depende funcionalmente de *PrvCod* porque para cada valor de *PrvCod* sólo existe UN valor de *PrvNom*.

La definición es algo más amplia, ya que se puede definir que un conjunto de atributos dependa funcionalmente de otro conjunto de atributos.

GeneXus diseña la base de datos llevándola a tercera forma normal (admitiendo algunas excepciones).

Para un estudio más detallado de normalización, dirigirse a cualquier libro que trate sobre el modelo de datos relacional.

Anexo B: Tabla extendida

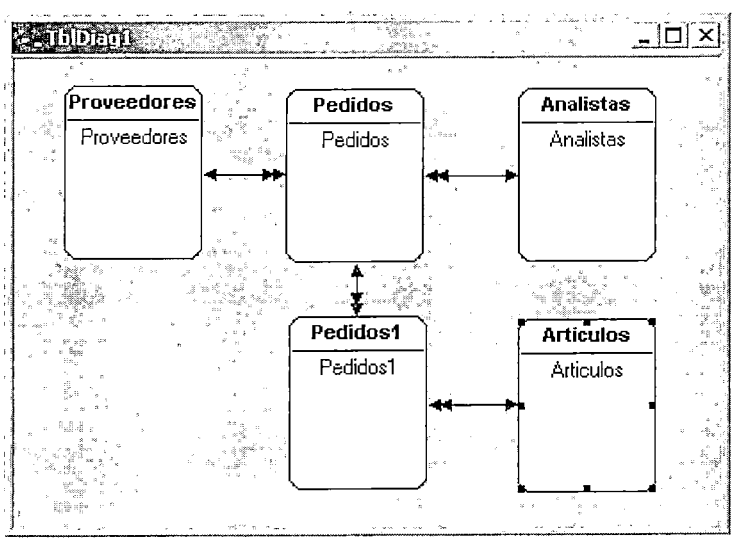
Como vimos en la sección anterior, el criterio de diseño de la base de datos se basa en minimizar la posibilidad de inconsistencia en los datos.

Una base de datos diseñada de esta manera tiene una serie de ventajas importantes (tal es así que actualmente la normalización de datos es un estándar de diseño), pero se deben tener en cuenta también algunos inconvenientes.

El inconveniente más notorio es que los datos se encuentran dispersos en muchas tablas, y por lo tanto cuando se quieren hacer consultas más o menos complejas a la base de datos se debe consultar una cantidad importante de tablas. Así, por ejemplo, para listar los pedidos por proveedor es necesario consultar las tablas PROVEEDORES y PEDIDOS. Para simplificar esta tarea GeneXus utiliza el concepto de tabla extendida, cuya definición es:

> Dada una tabla de la base de datos, que llamaremos tabla base, se denomina **tabla extendida** de la misma al conjunto conformado por los atributos que:
>
> - pertenecen a la tabla.
> - pertenecen a toda tabla Y, tal que la relación entre la tabla extendida determinada hasta el

Si hacemos un diagrama de Bachman del modelo de datos correspondiente al sistema de compras para la cadena de farmacias, obtenemos:



donde además de las tablas de proveedores y pedidos, tenemos tres tablas más: ARTICULOS, para almacenar la información de los artículos, ANALISTA que almacena la información de los analistas de compras, y PEDIDOS1, que registra las líneas de pedido, es decir, los artículos que componen cada pedido, junto al precio por artículo y la cantidad pedida del mismo.

La tabla extendida de PEDIDOS comprende a todos los atributos de dicha tabla, más todos los

atributos de PROVEEDORES y todos los de ANALISTAS, pero no los atributos de las líneas del pedido, PEDIDOS1, o de ARTICULOS (porque si bien están relacionados, su relación no es N-1).

En el siguiente diagrama se muestra, para cada una de las tablas del modelo, cuál es su tabla extendida:

Tabla base	Tabla extendida
PROVEEDORES	PROVEEDORES
ANALISTA	ANALISTAS
PEDIDOS	PEDIDOS, PROVEEDORES, ANALISTAS
PEDIDOS1	PEDIDOS1, PEDIDOS, PROVEEDORES, ANALISTAS, ARTICULOS
ARTICULOS	ARTICULOS

El concepto de tabla extendida es muy utilizado en GeneXus, donde generalmente se trabaja con tablas extendidas y no con tablas físicas.

Por ejemplo, el comando FOR EACH que estudiaremos determina una tabla extendida y no una tabla física. Lo mismo ocurre en las transacciones, donde se pueden actualizar directamente mediante las reglas cualesquier atributos que pertenezcan a la tabla extendida correspondiente al nivel de la transacción en el que se esté trabajando.

BIBLIOGRAFIA

ARTech Inc (1993). *GeneXus General View: 1989-2003*. Montevideo, Uruguay: Ed. GeneXus Developer Library.

BACHMAN, Charles N. (1964). *The Integrated Data Store, a General Purpose Programming System for Random Access Memories*. 1ra ed. Pones, Arizona: ACM Press.

BOEHM BARRY, W. (1988). *Applying process programming to the spiral Model*. Proc. Fourth software process. Work Shop. IEEE. UCLA.

CASTAÑO, Adoración de Miguel y otros (2000). *Fundamentos y modelos de Bases de Datos*. 2da ed. Madrid: Ra-Ma.

CODD, Edgar F. (1971). *"Further Normalization of the Data Base Relational Model"*, IBM Research Report. San José, California.

(1976). *"Seminario Avanzado de Bases de Datos", base de datos y avances*. Pontificia Universidad Católica de Río de Janeiro. Rió de Janeiro, Brasil.

DATE, Chris J. (1976). *An Introduction to Database Systems*, 1ra ed. New York, New York: Addisson-Wesley Publishing.

GONDA, Breogán, JODAL, Juan Nicolás (1995). *"Proyecto GeneXus: Resumen del Trabajo Ganador del Premio Nacional de Ingeniería 1995". Academia Nacional de Ingeniería*. Montevideo, Uruguay.

(2003). *Desarrollo Incremental*. 3ra ed. Montevideo, Uruguay: GeneXus Developer Library - Artech Inc.

(2006). *Desarrollo Incremental*. 4ra ed. Montevideo, Uruguay: GeneXus Developer Library - Artech Inc.

GUILFORD J. P. Y FRUCHTER, Benjamín (1984). *Estadística aplicada a la psicología y la educación*. 1ra ed. Colombia. McGraw-Hill.

KIT, Edward (1995). *Software testing in the real world*. Boston, MA: Addison-Wesley Publishing Company.

LUQUE RUIZ, Irene y otros (2002). *Bases de datos desde Chen hasta Codd con Oracle*. México. 1ra ed. Alfaomega Ra-Ma.

MÁRQUEZ LISBOA, Daniel (2006). *Guía práctica GeneXus. desarrollo basado en conocimiento*. 1ra ed. Montevideo, Uruguay: Grupo Magró.

PRESSMAN, Roger (2006). *Ingeniería del Software: Un enfoque práctico*. 6ta ed. México: McGraw-Hill.

SILBERSCHATZ, Abraham y otros (2002). *Fundamento de base de datos*. Madrid. MacGraw-Hill, ISBN: 84-481-3654-3.

ANEXO: Nº 6

MATRIZ DE VALORACIÓN PARA LA EVALUACIÓN DEL APRENDIZAJE DEL DESARROLLO DE SOFTWARE SOBRE BASE DE DATOS, MEDIANTE EL USO DE LAS RÚBRICAS[1].

Fundamento:

Si partimos de la premisa de que la evaluación tiene como propósito fundamental proporcionar información sobre los distintos momentos del aprendizaje del estudiante, el uso de las rúbricas ofrece ventajas claras como son:

- Es ventajoso para el maestro y para evaluar.
- Promueve expectativas sanas de aprendizaje pues aclara cuáles son los objetivos del maestro y de qué manera pueden alcanzarlos los estudiantes.
- Enfoca al maestro para que determine de manera específica los criterios con los cuales va a medir y documentar el progreso del estudiante.
- Permite al maestro describir cualitativamente los distintos niveles de logro que el estudiante debe alcanzar.
- Permite que los estudiantes conozcan los criterios de calificación con que serán evaluados.
- Aclara al estudiante cuales son los criterios que debe utilizar al evaluar su trabajo y el de sus compañeros.
- Permite que el estudiante evalúe y haga una revisión final a sus trabajo, antes de entregarlo al profesor.
- Indica con claridad al estudiante las áreas en las que tiene falencias y con éste conocimiento planear con el maestro los correctivos a aplicar.
- Provee al maestro información de retorno sobre la efectividad del proceso de enseñanza que está utilizando.
- Proporciona a los estudiantes retro alimentación sobre sus fortalezas y debilidades en las áreas que deben mejorar.
- Reduce la subjetividad en la evaluación.
- Promueve la responsabilidad.
- Ayuda a mantener el o los logros del objetivo de aprendizaje centrado en los estándares de desempeño establecidos y en el trabajo del estudiante.
- Proporciona criterios específicos para medir y documentar el progreso del estudiante.
- Es fácil de utilizar y de explicar.

[1] Bingham S, Norat T, Moskal A, Ferrari P, Slimani N, Clavel-Chapelon F, et al. Is the association with fiber from foods in colorectal cancer confounded by folate intake? Cancer Epidemiol Biomarkers Prev 2005;14(6):1552-6.[FI=4,46]

Puede considerarse dos tipos de matrices de valoración, la Comprehensiva (total) y la Analítica. En la Comprehensiva el profesor evalúa la totalidad del proceso o producto sin juzgar por separado las partes que lo componen. En contraposición, con la Matriz de valoración analítica el profesor evalúa inicialmente, por separado, las diferentes partes del producto o desempeño y luego suma el puntaje de estas para obtener una calificación total.

Con fines metodológicos y siguiendo las tendencias internacionales, el estándar de evaluación que se construye, considera la clasificación de Coll (ver Tabla Nº A-1).

CONCEPTUAL		Instrumento	ÍTEMS
Clasificación	Datos	Postest	4, 5
	Hechos		6, 7, 9
	Conceptos		1, 3
	Principios o leyes		2, 8, 10
PROCEDIMENTAL		Instrumento	ÍTEMS
Clasificación	Generales	Postest	2, 7
	Algoritmos		4, 8
	Heurísticos		3
	Destrezas y habilidades		5
	Técnicas		6, 10
	Estrategias		1
	Motriz cognitivo		9
ACTITUDINAL		Instrumento	ÍTEMS
Clasificación	Valores	Postest	3, 4, 7
	Normas		2,
	Actitudes		6, 8, 9
	Juicios valorativos		1, 5,10

Tabla Nº A-1. Distribución de ítem, considerando la clasificación de Coll.

La plantilla para la matriz de valoración del aprendizaje de los conocimientos conceptuales del desarrollo de software sobre base de datos se utiliza los valores de la Tabla Nº A-2:

Escala de Valoración	Valor asignado		Detalle
A	**75 ≤ % < 100**	**Fortaleza: Altamente significativo**	**Excelente**: Demuestra que conoce los fundamentos teóricos y prácticos del desarrollo de software sobre base de datos. Define los términos o principios involucrados en el desarrollo de la asignatura de manera completa y clara; o demuestra a través del uso que le da a los conceptos un entendimiento sofisticado de ellos.
B	**50 ≤ % < 75**	**Fortaleza: Muy significativo**	**Bueno**: Demuestra que conoce algunos fundamentos teóricos y prácticos del desarrollo de software sobre base de datos. Define de forma adecuada los términos o principios involucrados en el desarrollo de software; o demuestra a través del uso que le da a los conceptos un entendimiento correcto de ellos.
C	**25 ≤ % < 50**	**Debilidad: No significativo**	**Regular**: Demuestra que conoce algunos de los fundamentos teóricos y prácticos del desarrollo de software sobre base de datos. Define de forma rudimentaria o incompleta los términos o principios involucrados en el desarrollo de la asignatura; o demuestra a través del uso de los conceptos un entendimiento parcialmente correcto de ellos.
D	**0 ≤ % < 25**	**Debilidad: Considerablemente no significativo**	**Deficiente**: Demuestra poco conocimiento de los fundamentos teóricos y prácticos del desarrollo de software sobre base de datos. Define de forma incorrecta los términos o principios centrales involucrados en la asignatura; o demuestra a través del uso que le da a los conceptos un entendimiento incorrecto o muy parcial de ellos.

Tabla Nº A-2. Matriz de valoración de aprendizaje del conocimiento conceptual.

La plantilla para la matriz de valoración del aprendizaje de los conocimientos procedimentales del desarrollo de software sobre base de datos se utiliza los valores de la Tabla Nº A-3:

Escala de Valoración	Valor asignado		Detalle
A	**75 ≤ % < 100**	**Fortaleza: Altamente significativo**	**Excelente**: Demuestra que conoce los procedimientos del desarrollo de software sobre base de datos. Emplea los términos o principios involucrados en el desarrollo de la asignatura de manera completa y clara; o demuestra a través del uso que le da a los conceptos un entendimiento sofisticado de ellos.
B	**50 ≤ % < 75**	**Fortaleza: Muy significativo**	**Bueno**: Demuestra que conoce algunos procedimientos del desarrollo de software sobre base de datos. Emplea de forma adecuada los términos o principios involucrados en el desarrollo de software; o demuestra a través del uso que le da a los conceptos un entendimiento correcto de ellos.
C	**25 ≤ % < 50**	**Debilidad: No significativo**	**Regular**: Demuestra que conoce algunos de los procedimientos del desarrollo de software sobre base de datos. Emplea de forma rudimentaria o incompleta los términos o principios involucrados en el desarrollo de la asignatura; o demuestra a través del uso de los conceptos un entendimiento parcialmente correcto de ellos.
D	**0 ≤ % < 25**	**Debilidad: Considerablemente no significativo**	**Deficiente**: Demuestra poco conocimiento de los procedimientos del desarrollo de software sobre base de datos. Emplea de forma incorrecta los términos o principios centrales involucrados en la asignatura; o demuestra a través del uso que le da a los conceptos un entendimiento incorrecto o muy parcial de ellos.

Tabla Nº A-3. Matriz de valoración de aprendizaje del conocimiento procedimental.

ANEXO Nº 7

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

VALIDACIÓN DEL INSTRUMENTO:

- Test de Conocimientos Actitudinales

INFORME DE EXPERTOS

Tabla de evaluación de instrumentos por expertos

Nº	Criterio:	Puntaje asignado de 5 a 100				
		Experto 1	Experto 2	Experto 3	Experto 4	Promedio
1. CLARIDAD	Es formulado con lenguaje apropiado	90	90	90	90	90,00
2. OBJETIVIDAD	Está expresado en conductas observables	95	95	95	95	95,00
3. ACTUALIZAD	Está acorde a los cambios de la tecnología educativa.	100	95	95	95	96,25
4. ORGANIZACIÓN	Existe una organización lógica.	90	95	95	95	93,75
5. SUFICIENCIA	Comprende los aspectos de cantidad y calidad.	95	100	100	100	98,75
6. INTENCIONALIDAD	Adecuado para valorar los servicios educativos	100	95	95	95	96,25
7. CONSISTENCIA	Basado en aspectos teóricos científicos.	95	95	95	95	95,00
8. COHERENCIA	Entre los índices, indicadores y las dimensiones.	95	95	95	95	95,00
9. METODOLOGÍA	La estrategia responde al propósito del diagnóstico.	95	100	100	100	98,75
	Promedio	95,00	95,56	95,56	95,56	**95,42**

Experto Nº 1.- Dr. Raúl LOAYZA YAQUI (Docente de la Universidad Ricardo Palma)
Experto Nº 2.- Mg. Hugo VEJA HUERTA (Docente de la Universidad Mayor de San Marcos)
Experto Nº 3.- Mg. Jimmy ROSALES HUAMANI (Docente de la Universidad Mayor de San Marcos)
Experto Nº 4.- Mg. Bertila GARCÍA DIAZ (Docente de la Universidad Nacional del Callao)

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

VALIDACIÓN DEL INSTRUMENTO:

- Test de Conocimientos Conceptuales

INFORME DE EXPERTOS

Tabla de evaluación de instrumentos por expertos

Nº	Criterio:	Puntaje asignado de 5 a 100				
		Experto 1	Experto 2	Experto 3	Experto 4	Promedio
1. CLARIDAD	Es formulado con lenguaje apropiado	95	90	95	85	91,25
2. OBJETIVIDAD	Está expresado en conductas observables	100	95	90	95	95,00
3. ACTUALIZAD	Está acorde a los cambios de la tecnología educativa.	95	100	95	95	96,25
4. ORGANIZACIÓN	Existe una organización lógica.	95	95	95	95	95,00
5. SUFICIENCIA	Comprende los aspectos de cantidad y calidad.	95	95	95	95	95,00
6. INTENCIONALIDAD	Adecuado para valorar los servicios educativos	95	100	95	100	97,50
7. CONSISTENCIA	Basado en aspectos teóricos científicos.	95	100	95	95	96,25
8. COHERENCIA	Entre los índices, indicadores y las dimensiones.	100	95	90	95	95,00
9. METODOLOGÍA	La estrategia responde al propósito del diagnóstico.	95	95	100	95	96,25
	Promedio	96,11	96,11	94,44	94,44	**95,28**

Experto Nº 1.- Dr. Raúl LOAYZA YAQUI (Docente de la Universidad Ricardo Palma)
Experto Nº 2.- Mg. Hugo VEJA HUERTA (Docente de la Universidad Mayor de San Marcos)
Experto Nº 3.- Mg. Jimmy ROSALES HUAMANI (Docente de la Universidad Mayor de San Marcos)
Experto Nº 4.- Mg. Bertila GARCÍA DIAZ (Docente de la Universidad Nacional del Callao)

UNIVERSIDAD NACIONAL DE EDUCACIÓN
Enrique Guzmán y Valle
Alma Máter del Magisterio Nacional

Facultad de Ciencias

VALIDACIÓN DEL INSTRUMENTO:

- Test de Conocimientos Procedimentales

INFORME DE EXPERTOS

Tabla de evaluación de instrumentos por expertos

N°	Criterio:	Puntaje asignado de 5 a 100				
		Experto 1	Experto 2	Experto 3	Experto 4	Promedio
1. CLARIDAD	Es formulado con lenguaje apropiado	95	95	95	95	95,00
2. OBJETIVIDAD	Está expresado en conductas observables	95	95	100	95	96,25
3. ACTUALIZAD	Está acorde a los cambios de la tecnología educativa.	100	95	100	100	98,75
4. ORGANIZACIÓN	Existe una organización lógica.	95	95	95	100	96,25
5. SUFICIENCIA	Comprende los aspectos de cantidad y calidad.	100	95	100	100	98,75
6. INTENCIONALIDAD	Adecuado para valorar los servicios educativos	95	95	95	95	95,00
7. CONSISTENCIA	Basado en aspectos teóricos científicos.	100	100	100	100	100,00
8. COHERENCIA	Entre los índices, indicadores y las dimensiones.	100	95	95	100	97,50
9. METODOLOGÍA	La estrategia responde al propósito del diagnóstico.	95	95	95	95	95,00
	Promedio	97,22	95,56	97,22	97,78	**96,94**

Experto N° 1.- Dr. Raúl LOAYZA YAQUI (Docente de la Universidad Ricardo Palma)
Experto N° 2.- Mg. Hugo VEJA HUERTA (Docente de la Universidad Mayor de San Marcos)
Experto N° 3.- Mg. Jimmy ROSALES HUAMANI (Docente de la Universidad Mayor de San Marcos)
Experto N° 4.- Mg. Bertila GARCÍA DIAZ (Docente de la Universidad Nacional del Callao)

ANEXO Nº 8

RESULTADOS DE ENCUESTA A LOS DOCENTES DEL ÁREA DE INFORMÁTICA

METODOLOGIA GENEXUS:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Promedio
Docente 1	5	4	5	4	5	5	4	5	4	4	4,5
Docente 2	5	5	4	4	5	5	4	4	4	4	4,4
Docente 3	4	4	5	5	4	4	5	4	4	4	4,3
Docente 4	5	4	4	4	5	5	5	4	4	5	4,5
Docente 5	4	5	5	5	4	5	4	4	5	4	4,5
Docente 6	5	4	5	4	4	5	5	5	4	4	4,5
Docente 7	5	5	5	4	5	5	5	5	5	5	4,9
Docente 8	4	5	5	4	4	4	4	5	5	4	4,4
Docente 9	4	5	5	4	4	4	4	5	5	4	4,4
Docente 10	4	5	5	5	5	5	5	5	5	5	4,9
Promedio	4,43	4,71	4,86	4,29	4,43	4,71	4,57	4,71	4,71	4,43	**4,59**

METODOLOGIA TRADICIONAL:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10	Promedio
Docente 1	4	4	3	4	2	3	1	4	4	2	3,10
Docente 2	4	4	3		2	2	2	2	1	2	2,44
Docente 3	4	4	3	3	2	2	2	2	4	2	2,80
Docente 4	4	4	4	4	3	3	3	2	2	3	3,20
Docente 5	4	3	4	5	3	4	5	3	4	3	3,80
Docente 6	4	5	5	4	4	4	3	3	3	3	3,80
Docente 7	3	2	4	4	2	3	2	1	3	1	2,50
Promedio	3,75	3,50	4,25	4,25	3,00	3,50	3,25	2,25	3,00	2,50	**3,33**