ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

USO DEL PSP (PERSONAL SOFTWARE PROCESS) EN EL DESARROLLO DE SOFTWARE

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

ENRIQUE ANDRES LARCO AMPUDIA

DIRECTOR: ING. CARLOS MONTENEGRO

Quito, Marzo 2007

DECLARACIÓN

Yo, Enrique Andrés Larco Ampudia, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Enrique Andrés Larco Ampudia

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Enrique Andrés Larco Ampudia, bajo mi supervisión.

Ing. Carlos Montenegro
DIRECTOR DE PROYECTO

DEDICATORIA

Dedico este trabajo a todas las niñas y niños especiales que con sus rostros me hacen recordar que los ángeles si existen y a veces están entre nosotros y no nos damos cuenta.

A mi madre por darme todo lo que tiene y lo que no tiene.

A mi hermana Gaby por ser mi principal inspiración para este trabajo.

AGRADECIMIENTOS

Agradezco a ese ser supremo llamado Dios por guiar mi camino y darme este don llamado vida.

A mi madre por ser mi amiga y mi mayor apoyo.

A mi padre por dejarme elegir libremente mi carrera y toda mi familia por siempre estar apoyándome con sus gestos de cariño y amor.

A mi enamorada, amiga, confidente y dueña de mi corazón.

A mis amigos y amigas por su tiempo, ayuda y buen humor.

Al Dr. Clímaco Vinueza por su gran ayuda y guía en el desarrollo del test para los niños con parálisis cerebral.

A mi director de tesis por creer en mi y lo que soy como persona.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En el presente trabajo se combina en forma sistemática el PSP (Personal Software Process) con la metodología RUP para el desarrollo de Software.

Inicialmente se describe el PSP y el RUP, así como su combinación tomando como referencia los flujos de trabajo, cuyo resultado sirve de referencia para su aplicación práctica en el caso de estudio

En la parte del desarrollo del sistema para niños con parálisis cerebral, se emplean los flujos de trabajo del RUP y las mejores prácticas del PSP, demostrándose la validez del modelo teórico definido.

PRESENTACIÓN

El presente trabajo esta dividido en tres capítulos. En el primer capitulo se realiza la descripción del PSP, en el mismo que se explica su estructura y forma de utilización a través de los scripts que guían al ingeniero paso a paso para la realización de su trabajo. Además, se describe a la metodología RUP, haciendo énfasis en sus flujos de trabajo. Finalmente, se estructura y describe la combinación de RUP con PSP

En el segundo capitulo se describe la parálisis cerebral su concepto, tipos y características. Con toda la información del primer capitulo se describe el desarrollo del sistema, que empieza por los requerimientos y termina en las pruebas. Y por último en este capitulo se realiza la evaluación de la combinación del PSP y el RUP.

En el tercer capitulo están las conclusiones y recomendaciones a las que se ha llegado con el presente trabajo.

CAPITULO I

1 MARCO TEÓRICO

1.1 ESPECIFICACIÓN DE PSP

1.1.1 Calidad de Software y PSP (PERSONAL SOFTWARE PROCESS)

Hasta poco después de la Segunda Guerra Mundial, la estrategia de calidad en la mayoría de las organizaciones industriales se basaba en la realización de pruebas. Típicamente, se establecieron departamentos especiales de calidad, que se dedicaron a localizar defectos en los productos de software finales.

La estrategia tradicional de prueba y depuración es costosa, consumista de tiempo e inefectiva para la ingeniería y el trabajo manufacturado (diseñar y fabricar). Aunque la mayoría de las organizaciones industriales han adoptado principios de calidad modernos, la comunidad del software ha continuado confiando en las pruebas como el principal método de administración de calidad.

Calidad es un conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer unas necesidades explícitas o implícitas (ISO 8402).

Las organizaciones han usado las inspecciones como método para mejorar la calidad del software. Otro paso encaminado a la mejora de calidad de software se produjo con la introducción del Modelo de Madurez de Suficiencia para software (CMM) en 1987. El enfoque principal del CMM se establece en el sistema de manejo, soporte y asistencia provisto para los ingenieros de desarrollo.

Otro paso significativo para la mejora de la calidad de software se realizó con el Personal Software Process (PSP) [1], el cual extiende el proceso de mejora a

las personas que hacen el trabajo: los ingenieros novatos. El PSP se concentra en las prácticas individuales de trabajo de los ingenieros.

Los principios del PSP sirven para producir software de calidad, cada ingeniero tiene que hacer un trabajo de calidad al producir software. El PSP ha sido diseñado para ayudar de forma consistente a los profesionales de software, usando prácticas de ingeniería probadas, además indica cómo planear y rastrear su trabajo, utilizar procesos definidos, establecer metas definidas, y desarrollar rastreos para sus metas.

El PSP muestra a los ingenieros cómo manejar la calidad del trabajo desde principio, cómo analizar los resultados de cada trabajo, y cómo usar los resultados para mejorar el proceso para el próximo proyecto.

1.1.1.1 Manejo de la Calidad del PSP

La calidad en el desarrollo de software es importante, y los defectos inclusive los pequeños, pueden causar serios problemas. Cuando los sistemas crecen en complejidad y son automáticos, existe mayor probabilidad de fallas catastróficas en los mismos.

El problema de la calidad de programas grandes, depende de la calidad de las partes más pequeñas de las cuales son construidas. Así, para producir grandes programas de calidad, cada ingeniero de software, desarrolla una o más de las partes del sistema, realizando un trabajo de calidad. Esto significa que todos los ingenieros deben manejar la calidad de su trabajo personal. Para ayudarles a hacer esto, el PSP guía a los ingenieros a encaminar y manejar cada defecto.

1.1.1.1.1 Defectos y Calidad.

Para los usuarios las necesidades funcionales deben ser de calidad, con un producto de software confiable y consistente, Para mejorar la funcionalidad en el software, los ingenieros deben remover casi todos los defectos.

Los datos del PSP muestran que aun los programadores experimentados inyectan un defecto en cada 7 a 10 líneas de código. Los ingenieros regularmente encuentran y arreglan la mayoría de estos defectos al compilar y en la prueba de unidad del programa, pero los métodos de software tradicionales dejan muchos defectos en el producto terminado.

Recopilando errores se pueden producir defectos destructivos o muy difíciles de encontrar. Recíprocamente, muchos defectos sofisticados del plan son a menudo fáciles encontrar. El error y sus consecuencias son considerablemente independientes. Incluso los errores triviales de implementación pueden causar problemas serios en el sistema. Es particularmente importante el origen de la mayoría los defectos de software, que el programador obvia por error. Mientras los problemas del plan siempre son importantes, nuevos programas desarrollados tienen pocos defectos en su plan, comparados con el gran número de errores sencillos.

Para mejorar calidad del programa, los ingenieros entrenados en el PSP muestran el seguimiento y el manejo de todos los defectos, que ellos encuentran en sus programas.

1.1.1.1.2 La Responsabilidad del Ingeniero

El primer principio de calidad del PSP, indica, que los ingenieros son personalmente responsables de la calidad de los programas que ellos producen. Porque el ingeniero de software que escribe un programa está muy familiarizado con él, ese ingeniero puede eficazmente y efectivamente encontrar, arreglar, y prevenir sus defectos.

El PSP proporciona una serie de prácticas y medidas, para ayudar a los ingenieros a evaluar la calidad de los programas que ellos producen y para guiarlos a encontrar y arreglar rápidamente todos los defectos del programa.

Además de la medida de calidad y del rastreo, el método del PSP hace que los defectos sean removidos y prevenidos tempranamente.

1.1.1.1.3 Eliminación temprana de Defectos

El principal objetivo de calidad del PSP es encontrar y arreglar defectos antes de la primera compilación o prueba de unidad. El proceso del PSP incluye un plan y pasos para la revisión del código, el cual los ingenieros revisan personalmente, antes de ser inspeccionado, compilado, o probado.

El PSP revisa los procesos que las personas realizan, en los cuales tienden a cometer repetitivamente los mismos errores. Por consiguiente, analizando los datos de los defectos, ellos han construido una lista de chequeo de defectos, para que los ingenieros puedan encontrar y arreglar esos defectos más efectivamente.

Los ingenieros entrenados en el PSP encuentran un promedio de 6.52 defectos por hora en revisiones personales del código y 2.96 defectos por hora en revisiones personales del diseño. Esto comparado con 2.21 defectos encontrados por hora en la prueba unidad. Usando estos datos del PSP, los ingenieros pueden al mismo tiempo, ahorrar tiempo y mejorar la calidad del producto.

Por ejemplo, en un promedio de datos del PSP, el tiempo para remover 100 defectos en la prueba de unidad es 45 horas, mientras el tiempo para encontrar ese número de defectos en revisiones del código sólo es 15 horas.

1.1.1.1.4 Prevención del defecto

La manera más eficaz de manejar defectos es prevenir su introducción inicial. En el PSP, hay tres maneras diferentes, pero mutuamente encaminadas a favorecer la prevención de defectos.

El primer método indica que los ingenieros deben registrar por escrito los datos de cada defecto que ellos encuentran y arreglan. Entonces ellos revisan estos datos para determinar la causa que provocaron los defectos y para hacer los cambios necesarios en el proceso para eliminar dichas causas. Con la medición de los defectos, los ingenieros son más conscientes de sus errores, más sensibles a sus consecuencias y obtienen los datos necesarios para evitar cometer los mismos errores en el futuro.

El segundo método de prevención del defecto, es usar un método efectivo de diseño y registrarlo por escrito, de esta manera se producen diseños completos. Para completar el registro del diseño los ingenieros tienen que entenderlo completamente. Esto no sólo produce mejores diseños; esto produce menos errores en el diseño.

El tercer método de prevención del defecto es una consecuencia directa del segundo: con un diseño más completo y codificado, se reduce el tiempo y consecuentemente se reduce la inyección del defecto.

1.1.2 Descripción de PSP

1.1.2.1 Introducción

La investigación realizada por Donna L. Jonson y Judith G. Brodman [12] alrededor de 1992 y 1995 señala que existe frustración por parte de las organizaciones pequeñas, negocios pequeños y organizaciones con proyectos pequeños al tratar de mejorar sus procesos de software basados en el CMM, puesto que este modelo aplica prácticas que son inaplicables al contexto de dichas organizaciones, negocios y proyectos.

La investigación realizada se divide en dos fases. En la primera fase se identifica los problemas específicos a través de encuestas y entrevistas. En esta extensa investigación, se determina que el problema del proyecto pequeño también se extiende a negocios grandes que han estado en pérdida al tratar de mejorar el proceso en sus pequeños proyectos al más alto nivel de madurez.

La segunda fase proporciona soluciones a las necesidades de las organizaciones pequeñas. Una de las soluciones que se identifico fue moldear el CMM, para lograr este trabajo a medida, se trabajo con más de 50 organizaciones de software que estaban experimentando los problemas identificados en la primera fase de la investigación.

Como resultado de seis meses de estudio, se concluye que el recoger grandes cantidades de información para detallar las áreas y prácticas utilizadas en el CMM son la razón y causa de los problemas.

El PSP nace cuando Watts Humphrey decide aplicar los principios de CMM para escribir pequeños programas. Muchas personas se han estado preguntando cómo aplicar el CMM a organizaciones pequeñas o equipos de trabajo pequeños de software. Mientras los principios del CMM se aplican a dichos grupos, más se sentía la necesidad de aplicar una guía precisa para saber que hacer.

Humphrey utiliza los principios del CMM para desarrollar un programa de tamaño modular, para aproximarse y ver el trabajo y la configuración, para la conveniencia de los ingenieros que fueran a adoptar dicha práctica. Poco después de que él empezó este proyecto en abril de 1989, el Instituto de Ingeniería del Software (SEI) y Humphrey se hicieron aliados, de esa forma Humphrey se dedica por completo a la investigación del PSP.

Durante los próximos tres años, él desarrolló un total de 62 programas y definió aproximadamente 15 versiones del PSP. Él utilizo como lenguajes de programación: Pascal, Object Pascal, y C++ para desarrollar aproximadamente un código de 25,000 líneas. De esta experiencia, él concluyó que los principios

de administración de procesos de Deming y Juran eran tan aplicables al trabajo individual del ingeniero de software como a los otros campos de la tecnología.

Inmediatamente después Humphrey escribió un documento, que provee la forma de enseñar cursos de PSP a varias sociedades. Basado en las experiencias y datos de los cursos iniciales, Humphrey revisó el documento del PSP y publicó la versión final del libro a finales de 1994. Watts Humphrey y el SEI han continuado trabajando en el PSP, desarrollando e introduciendo los mismos principios al trabajo de equipos de ingeniería. Este trabajo es llamado el Proceso de Software de Equipo TSP.

1.1.2.2 Definición de PSP

El PSP es un mejoramiento en si de los procesos y fue diseñado para ayudar a los ingenieros a controlar, administrar y mejorar su manera de trabajar. Es una estructura con un marco referencial, directrices y procedimientos para el desarrollo de software. El PSP proporciona datos históricos para realizar mejor su trabajo de acuerdo a los compromisos establecidos, y hace que los elementos rutinarios de su trabajo sean más previsibles y más eficaces. Además proporciona métodos detallados de planificación y estimación, muestra a los ingenieros cómo controlar su rendimiento frente a esos planes y explica cómo los procesos definidos guían su trabajo.

1.1.2.3 Los Principios del PSP

El diseño del PSP esta basado en los siguientes principios de planificación y calidad:

1. Cada ingeniero es diferente, para ser efectivo el ingeniero debe planear su trabajo y basar sus planes de acuerdo a su información personal.
2. Para Mejorar la consistencia del desarrollo los ingenieros tienen que definir bien el uso y la medida del proceso.
3. Para producir productos de calidad los ingenieros deben sentirse personalmente responsables de la calidad del producto. Productos

superiores no son producidos por error, el ingeniero tiene que agotar todos los esfuerzos para llegar a trabajos de calidad.

4. Un defecto cuesta menos detectarlo que arreglarlo más tarde durante el proceso.
5. Es más eficiente prevenir los defectos que detectarlos y arreglarlos.
6. La manera correcta siempre es la manera más rápida y más barata de hacer un trabajo.

El trabajo de la ingeniería es hacer software correctamente, indica que los ingenieros tienen que planear el trabajo antes comprometerse a realizarlo. Para entender su desempeño personal, ellos tienen que medir el tiempo que se van a demorar en cada etapa del trabajo, los defectos que van a inyectar y remover, y el tamaño del producto que van a producir.

Para producir productos de calidad de forma consistente, los ingenieros deben planear, medir, y rastrear la calidad del producto, y ellos tienen que enfocarse en la calidad desde el inicio del trabajo. Finalmente, ellos deben analizar los resultados de cada trabajo y usar estos resultados para mejorar sus procesos personales.

1.1.2.4 La Estructura del Proceso PSP

La estructura del proceso PSP inicia con los requerimientos, los cuales son el primer paso en el proceso de planeación del PSP. Hay una guía escrita de planificación para este trabajo y un resumen para registrar los datos de la planificación. Mientras los ingenieros van siguiendo la guía escrita para hacer el trabajo, registran el tiempo que tardaron, los datos del tiempo mal aprovechado y los caracteres mal tipiados.

Al final del trabajo, durante la fase postmortem se resume el tiempo utilizado, los caracteres mal tipiados y se mide el tamaño del programa. Se entrega el producto terminado junto con el plan completo y el formato de resumen.

La estructura del proceso PSP se muestra conceptualmente en la Figura 1.1

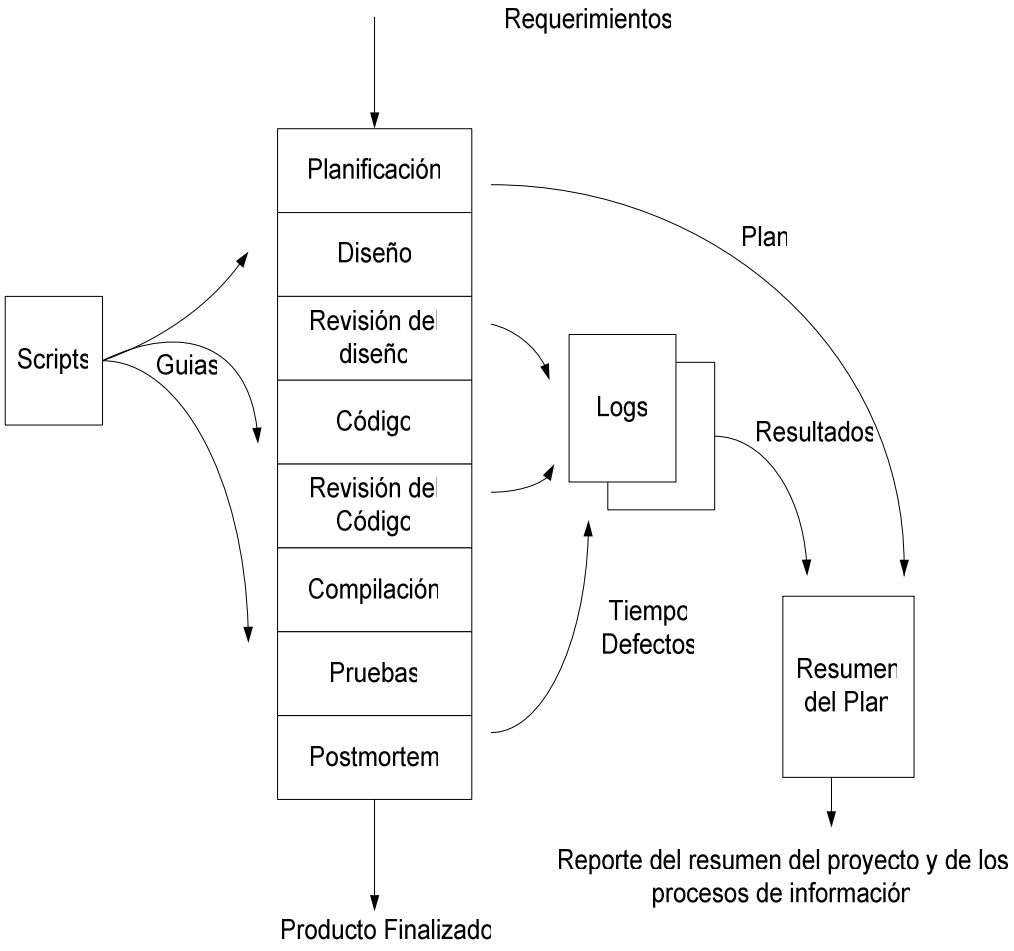


Figura 1.1 Flujo del proceso del PSP

El proceso PSP tiene métodos numéricos que generalmente no han sido utilizados por los ingenieros, desde que los métodos han sido introducidos en una serie de 7 versiones. Estas versiones se han etiquetadas desde PSP0 a PSP3, y cada versión tiene un juego similar de logs, formas, scripts, y estándares; se muestra en la Figura 1.2.

Los scripts del proceso definen los pasos para cada parte del proceso, los logs y las formas, las mismas que tienen plantillas para registrar y archivar datos, y la guía de estándares que los ingenieros utilizan cuando desarrollan su trabajo.

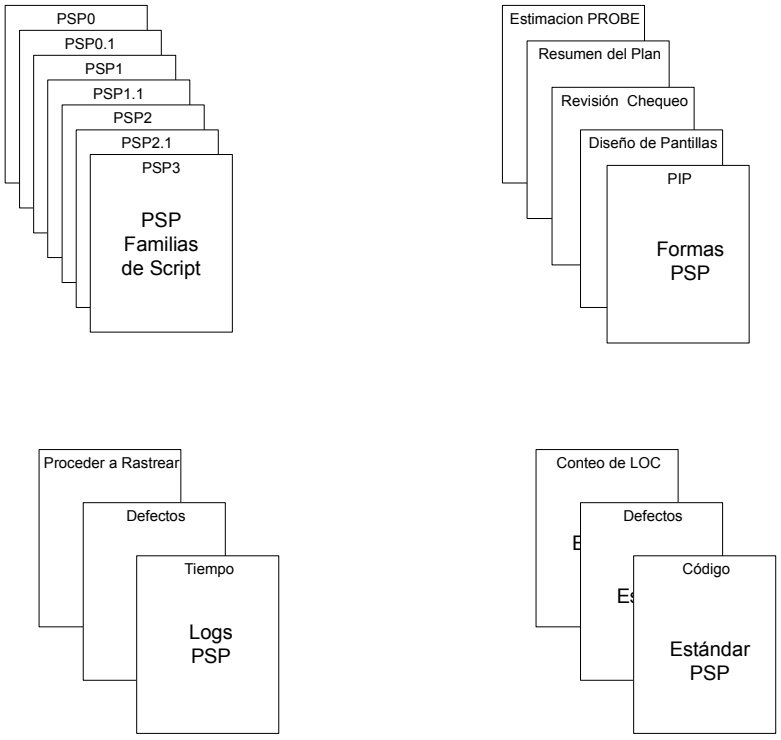


Figura 1.2 Los Elementos del Proceso PSP

Un script de PSP es lo que Deming llamó un proceso operacional. En otras palabras, es un proceso que se diseña para ser usado. Es construido en un formato simple de usar, corto y con instrucciones precisas. Mientras los script describen que hay que hacer, ellos son más como listas de chequeo que guías didácticas.

El PSP consta de varios scripts, los cuales sirven para guiar a los ingenieros a entender el proceso utilizado, los mismos que se muestran en las siguientes tablas:

El Script del Proceso PSP3 se describe en la Tabla1.1

Número de fase	**Propósito**	**Guía para ayudar a desarrollar programas a nivel de componentes**
	Criterio de entrada	• Descripción del problema o especificaciones de los componentes.

		• Estándares y formas del proceso. • Historial del tamaño actual y estimado de los datos del tiempo. • Cronometro (opcional).
1	Requerimientos y Planificación	• Producir el plan de los requerimientos y desarrollo. o Documentos de requerimientos. o Diseño conceptual. o Tamaño, calidad, recursos, horarios de los planes. • Producir un Log principal del rastreo del problema.
2	Diseño de Alto Nivel	Producir la estrategia de diseño e implementación: • Especificaciones funcionales. • Especificaciones de estado. • Escenarios operacionales. • Especificaciones de Reuso. • Estrategia de desarrollo. • Estrategia de pruebas.
3	Revisión del Diseño de Alto Nivel.	• Revisión del diseño de alto nivel. • Revisión de la estrategia de desarrollo y pruebas. • Arreglar y anotar todos los defectos establecidos. • Anotar los problemas destacados en el Log de rastreo de los problemas. • Registrar todos los defectos establecidos en el Log de registro de defectos.

4	Desarrollo	• Diseñar el programa. • Revisar el diseño, reparar y anotar todos los defectos establecidos. • Implementar el diseño, usando las plantillas apropiadamente. • Revisar el código, reparar y anotar todos los defectos establecidos. • Compilar el programa, reparar y anotar todos los defectos establecidos. • Probar el programa, reparar y anotar todos los defectos establecidos. • Completar el tiempo de registro de Log. • Reponga y recicle como necesite.
5	Postmortem	• Completar la forma del plan del proyecto con el tiempo actual, defectos, tamaño de los datos actualizados. • Completar la forma del resumen del ciclo con datos actuales del ciclo.
	Criterios de salida	• Una completa prueba del programa. • Completar el resumen del plan del proyecto con datos estimados y actuales. • Completar las plantillas de diseño. • Completar las lista de chequeo del diseño y la lista de chequeo del código. • Completar la plantilla del reporte de pruebas. • Completar el Log de rastreo de problemas. • Completar los Logs de registro de defectos y tiempo.

Tabla 1.1 Script del Proceso PSP3

El Script de la Planificación PSP3 se describe en la Tabla1.2

Número de fase	**Propósito**	**Guía para la planificación del proceso PSP3**
	Criterio de entrada	• Descripción del problema. • Resumen del plan del proyecto PSP3 y el resumen del ciclo. • Estimación del tamaño, planificación de tareas, plantillas de planificación de horarios. • Historial del tamaño estimado y actual de los datos del tiempo. • Log del tiempo de registro.
1	Requerimientos del Programa	• Producir u obtener un informe de los requerimientos del programa. • Asegurar que el informe escrito de requerimientos sea claro y sin ambigüedad. • Resolver cualquier pregunta.
2	Estimación de Tamaño	• Producir el diseño conceptual de un programa. • Usar el método PROBE para estimar el nuevo y cambiado LOC requerido para el desarrollo del programa. • Estimar el LOC base, añadido, borrado, modificado y reusado. • Completar la plantilla de estimación del tamaño y el resumen del plan del proyecto. • Calcular el 70 por ciento del tamaño predecible del intervalo del proyecto

		total (usted puede usar una hoja de cálculo).
3	Estrategia del ciclo de desarrollo	Subdividir el desarrollo del programa en módulos de 100 y ninguno más de 250 nuevo y cambiado LOC. • Asigne el LOC para ser desarrollado entre los ciclos • Registre el resumen del ciclo en el plan.
4	Estimación de Recursos	• Use el método PROBE para estimar el tiempo requerido en el desarrollo del nuevo programa. • Calcular el 70 por ciento del tamaño predecible del intervalo del proyecto total (usted puede usar una hoja de cálculo). • Subdividir el total del desarrollo entre varios ciclos de desarrollo. • Usar el % a la fecha del más reciente programa desarrollado como una guía, distribuir por encima del tiempo de desarrollo las fases planeadas del proyecto de cada uno de los ciclos de desarrollo. • Registrar los datos del tiempo en el resumen del plan del ciclo.
5	Planificación de tareas y Horarios	Para los proyectos se requiere varios días de trabajo o más, complete las plantillas de planificación de tareas y planificación de horarios.

6	Estimación de Defectos	• Basado en datos de los defectos de nuevos y cambiados LOC, de la fecha actual, estimar el total de defectos establecidos en el programa. • Basado en el % de datos a la fecha, estimar el número de defectos inyectados y removidos por ciclo y por fase.
	Criterios de salida	• Un documento escrito de los requerimientos. • Diseño conceptual del programa. • Una completa plantilla de estimación del tamaño. • Para proyectos de varios días de duración, completar las plantillas de planificación de tareas y horarios. • Resumen del plan del proyecto y del resumen del plan del ciclo, formas con la estimación del tamaño del programa y tiempo de desarrollo. • Predicción del 70% de intervalos del total del proyecto. • Completar el registro del tiempo en el Log.

Tabla 1.2 Script de la Planificación PSP3

El Script del diseño de alto nivel PSP3 se describe en la Tabla 1.3

Número de fase	**Propósito**	**Guía para el diseño de alto nivel PSP3**
	Criterio de	Chequear lo que se tiene más cerca:

	entrada	• El documento escrito de requerimientos. • El diseño conceptual. • Estimación de tamaño y de tiempo. • El Log de rastreo de problemas.
1	Especificaciones Externas	• Completar la plantilla del escenario operacional para todos los usos normales y anormales de las funciones externas. • Completar la plantilla de especificación de estado para el programa completo. • Completar la plantilla de especificación de funciones para todos los programas.
2	Modulo de Diseño	• Subdividir el programa en varias unidades modulares. • Especificar las características externas de cada una de las plantillas de escenario Operacional y especificaciones Funcionales. • Completar las plantillas de especificaciones de Lógica y de estado para el más alto nivel o rutina de control. • Registrar los principales problemas o preocupaciones en el Log de rastreo de problemas.
3	Prototipos	• Identificar, para el plan las necesidades a ejecutar del experimento de prototipo.

4	Estrategia de Desarrollo	Utilizar una estrategia para el desarrollo y pruebas en los ciclos del programa. Posiblemente donde: • Se guarde cada uno de los ciclos sobre los 100 LOC de nuevo y cambiado código. • Se complete el desarrollo de módulos en un ciclo. • Se minimice la cantidad de pruebas entablados en los requerimientos. • Se exponga lo más pronto posible el principal riesgo en el desarrollo.
5	Documentación de la Estrategia de Desarrollo	Producir la estrategia de desarrollo: • Definir las funciones del programa y/o módulos a producir en cada ciclo de desarrollo. • Especificar las pruebas aprobadas para la integración progresiva de los módulos en cada uno de los ciclos.
6	Log de Registro de Problemas	Revisar el Log de registro de problemas durante el diseño y hacer las adiciones y cambios apropiados.
	Criterios de salida	• El diseño global del programa, incluidas las plantillas de especificaciones de control principal y de rutina. • Especificación del diseño para todo el plan de componente de los módulos del programa.

		• Plantillas de especificación de funciones. • Plantillas del escenario operacional. • Desarrollo del ciclo y estrategias de pruebas. También: • Un completo resumen del ciclo con los datos del plan. • Una actualización del Log de registro de problemas.

Tabla 1.3 Script de diseño de alto nivel PSP3

El Script de revisión del diseño de alto nivel PSP3 se describe en la Tabla 1.4

Número de fase	**Propósito**	**Guía para la revisión del diseño de alto nivel PSP3**
	Criterio de entrada	Chequear lo que se tiene más cerca: • El documento escrito de requerimientos. • Especificaciones del diseño para todos los componentes de los módulos, incluyendo las especificaciones de las funciones y las plantillas del escenario operacional. • Las plantillas de especificaciones de todo el diseño para todo el programa y su rutina principal. • La estrategia de desarrollo del ciclo. • El Log de rastreo de problemas.
1	Alcance del Diseño	Verificar los requerimientos según el alcance del diseño.

		• Todas las especificaciones y funciones requeridas. • Los temas del más alto nivel de diseño han sido bien diseccionados al Log de rastreo de problemas. • Los materiales requeridos para el diseño deben estar producidos.
2	Verificación de la Máquina de Estados	Verificar la máquina de estados del diseño. • El estado es completo y ortogonal. • Las condiciones de transición de cada uno de los estados es completo y ortogonal.
3	Verificación de la Lógica	Verificación de la lógica del más alto nivel de diseño. Utiliza un método definido que sea fluido: • Tablas de ejecución. • Tablas de rastreo. • Pruebas de verificación.
4	Verificación de la Consistencia del Diseño	Verificar que todos los nombres especiales y tipos sean claros y consistentes de acuerdo a los estándares establecidos.
5	Verificación de Reuso	Verificar todas las funciones reusadas que están disponibles han sido usadas apropiadamente según lo planeado.
6	Verificación de la Estrategia de	Revisar la estrategia de desarrollo para asegurar: • Todas las funciones que son requeridas. • Todas las funciones que son necesarias

	Desarrollo	y están disponibles en la estrategia de pruebas. • La estrategia de pruebas tiene todas las funciones requeridas y los principales estados del programa.
7	Reparar Defectos	• Reparar todos los defectos establecidos. • Registrar los defectos en el Log de grabación de defectos.
	Criterios de salida	Al haber realizado se debe tener lo siguiente: • Un completo diseño del más alto nivel. • Actualizado el Log de Rastreo de Problemas. • Un completo Log del registro del tiempo. • Un completo Log del registro de defectos.

Tabla 1.4 Script de revisión del diseño de alto nivel PSP3

El Script del Desarrollo PSP3 se describe en la tabla 1.5

Número de fase	**Propósito**	**Guía para el desarrollo de pequeños programas a nivel de componentes**
	Criterio de entrada	• Modulo funcional y especificación operacional. • Estrategia de desarrollo y pruebas. • Revisión completa del más alto nivel de diseño. • Actualización del Log de rastreo de problemas. • Log del registro del tiempo y Log del

		registro de defectos. • Estándar de los tipos de defectos y estándar de codificación.
1	Modulo de Diseño	• Revisar el modulo de los requerimientos y producir una especificación externa de ellos. • Complete las especificaciones de las funciones y de las plantillas del escenario operacional y registre esa especificación. • Producir un diseño que reúna esas especificaciones. • Registre en las plantillas del diseño la especificación de las funciones, especificaciones operacionales, especificaciones de estado y especificaciones de lógica requeridas. • Completar el diseño del modulo con los materiales y facilidades de las pruebas.
2	Revisión del Diseño	• Seguir revisando el diseño con listas de chequeo y revisar el modulo con los materiales de prueba diseñados. • Reparar todos los defectos establecidos. • Registrar los defectos en el Log de registro de defectos. • Registrar el tiempo en el Log de registro del tiempo.
3	Codificación	• Seguir el estándar de codificación y de

		implementación del diseño. • Registrar en el Log de registro de defectos todos los defectos establecidos. • Registrar el tiempo en el Log de registro del tiempo.
4	Revisión de la Codificación	• Seguir la revisión del código con listas de chequeo y revisar el modulo con el código de prueba. • Reparar todos los defectos establecidos. • Registrar los defectos en el Log de registro de defectos. • Registrar el tiempo en el Log de registro del tiempo.
5	Compilación	• Compilar el programa hasta que este libre de errores. • Reparar todos los defectos establecidos. • Registrar los defectos en el Log de registro de defectos. • Registrar el tiempo en el Log de registro del tiempo.
6	Prueba	• Probar los módulos hasta que la prueba se ejecute sin errores. • Reparar todos los defectos establecidos. • Registrar los defectos en el Log de registro de defectos. • Registrar el tiempo en el Log de

		registro del tiempo. • Completar la plantilla de reporte de pruebas en las pruebas dirigidas y en los resultados obtenidos.
7	Reasegurar y reciclar	• Registrar los datos del ciclo de desarrollo. • Reasegurar el estado del plan y decidir si continua lo planeado o se realizan cambios.
	Criterios de salida	• Los módulos completamente probados de acuerdo al estándar de codificación. • Completar las plantillas de diseño. • Completar la revisión del diseño y la revisión de la compilación con las listas de chequeo. • Actualizar la forma del resumen del ciclo con datos actuales. • Completar la plantilla del reporte de pruebas. • Completar el Log de registro de defectos. • Completar el Log de registro de tiempo. • Actualizar el Log de rastreo de problemas.

Tabla 1.5 Script del Desarrollo PSP3

El Script de Postmortem PSP3 se describe en la tabla 1.6

Número de fase	**Propósito**	**Guía para el proceso postmortem de el PSP.**
	Criterio de entrada	• Descripción del problema e informe escrito de requerimientos. • Resumen del plan del proyecto con el tamaño del programa, tiempo de desarrollo y datos de los defectos. • Completar el resumen del ciclo. • Para los proyectos de varios días de duración, completar las tareas planificadas y las plantillas de planificación de horarios. • Completar la plantilla del reporte de pruebas. • Completar las plantillas de diseño. • Completar las plantillas de revisión de diseño y de codificación con las listas de chequeo. • Completar el Log de registro de tiempo. • Completar el Log de registro de defectos. • Probar y ejecutar el programa de acuerdo al los estándares de codificación. • Actualizar el Log de rastreo de problemas.
1	Defectos	• Determinar el número de defectos

	inyectados	inyectados en el registro de defectos del Log de la fase PSP3. • Registre el número de defectos inyectados actualmente en el resumen del plan del proyecto.
2	Defectos removidos	• Determine el número de defectos removidos en el registro de defectos del Log de la fase PSP3. • Registre el número de defectos removidos actualmente en el resumen del plan del proyecto. • Calcular el rendimiento global actual del proceso en el resumen del plan del proyecto.
3	Tamaño	• Contar el LOC en el programa completo. • Determine la base, reuso, borrado, modificado, añadido, total nuevo y cambiado, nuevo y reusado LOC. • Registre los datos el resumen del plan del proyecto.
3	Tiempo	• Revisar completamente el Log de registro de tiempo. • Registrar el tiempo total gastado en la actual fase PSP3 en el resumen del plan del proyecto.
	Criterios de salida	• Un programa totalmente probado de acuerdo a los estándares de

		codificación. • Completar la plantilla de diseño. • Completar la revisión del diseño y de codificación con listas de chequeo. • Completar la plantilla del reporte de pruebas. • Completar las formas del plan del proyecto y del resumen del ciclo. • Completar los Logs de registro de defectos y tiempo. • Actualizar el Log de rastreo de problemas.

Tabla 1.6 Script de Postmortem PSP3

1.1.2.5 Planificación de PSP

El proceso planificado en el PSP se muestra conceptualmente en la Figura 1.3

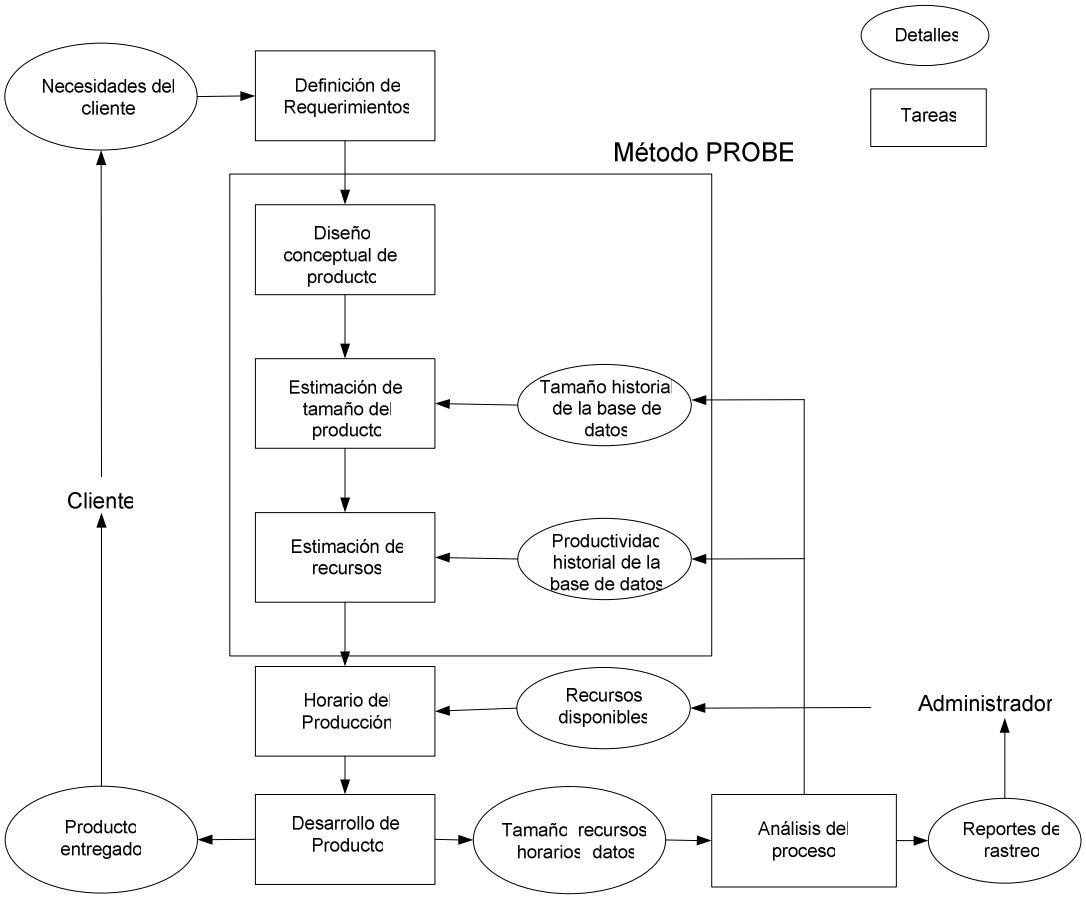


Figura 1.3 Procesos en el planeamiento del proyecto

- **Requerimientos:** Los Ingenieros comienzan a planificar el trabajo que se va a realizar con todos los detalles posibles. Si lo que tienen es una simple oración como informe, esta información debe ser la base del plan. Por supuesto, la exactitud de la estimación y plan, esta relacionada directamente con el conocimiento que tenga el ingeniero acerca del trabajo que se va a realizar.

- **Diseño conceptual:** Para hacer una estimación y un plan, primero el ingeniero debe definir cómo va a diseñar el producto y construirlo. Sin embargo, en la fase de planificación se hace demasiado temprano para producir un diseño completo del producto, los ingenieros producen lo

que se llama un diseño conceptual. Durante la fase del diseño, los ingenieros examinan alternativas del diseño y producen un diseño completo del producto.

- **Estimación del Tamaño del Producto y los Recursos:** La correlación del tamaño del programa con tiempo de desarrollo, es moderado por buenos equipos de ingenieros y organizaciones. Sin embargo, individualmente para los ingenieros la correlación es generalmente alta. Por consiguiente, el PSP inicia con la estimación del tamaño del producto que los ingenieros desarrollaran personalmente. Entonces, basados en sus datos personales y en los datos de productividad, los ingenieros estiman el tiempo para hacer el trabajo. En el PSP, éste tamaño y recurso estimativo son realizados con el método PROBE.

- **El Horario de Producción:** Una vez que los ingenieros saben el tiempo requerido para cada etapa del proceso, ellos, estiman el tiempo que ellos pasarán en el trabajo cada día o semana. Con esa información, ellos distribuyen el tiempo para realizar la tarea durante las horas establecidas, para producir en el tiempo planeado y completar la tarea.

- **Desarrollo del Producto.** Los ingenieros realizan el trabajo programado. Mientras este trabajo normalmente no es considerado parte del proceso de la planificación, los ingenieros deben utilizar estos datos del proceso para hacer planes futuros.

- **Análisis del Proceso.** Después de completar un trabajo, los ingenieros hacen un análisis postmortem del trabajo. En el postmortem, ellos ponen al día el resumen del plan de proyecto con datos actuales, calculan cualquier cantidad requerida o desempeño de los datos, y la revisión qué ellos realizaron contra el plan.

1.1.2.5.1 Estimación del tamaño con PROBE

EL PROBE[1] se utiliza como la base de la estimación del tamaño del producto [1]. Con PROBE, los ingenieros primero determinan los objetos requeridos para construir el producto descrito por el diseño conceptual. Entonces ellos determinan el probable tipo y número de métodos para cada objeto. Ellos se refieren a los datos históricos de los tamaños de objetos similares, que ellos han desarrollado previamente y usando la regresión lineal se determina el supuesto tamaño global del producto terminado.

Desde que el tamaño del objeto es una función, el método PROBE les muestra a los ingenieros cómo usar los datos en los programas que ellos han desarrollado personalmente, para generar rangos del tamaño para su uso personal.

Una vez que ellos han estimado los tamaños de los objetos, ellos usan regresión lineal para estimar la cantidad total de código que ellos planean desarrollar. Para usar regresión lineal, los ingenieros deben tener datos históricos mínimo de tres programas anteriores.

El ejemplo del tamaño del objeto de datos en la Tabla 1.7, muestra los 5 rangos que el PSP usa para objetos.

Tamaño de C++ Object en LOC por Método					
Categoría	**Muy Pequeño**	**Pequeño**	**Mediano**	**Grande**	**Muy Grande**
Cálculo	2.34	5.13	11.25	24.66	54.04
Datos	2.60	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
Lógica	7.55	10.98	15.98	23.25	33.83
Establecer	3.88	5.04	6.56	8.53	11.09
Texto	3.75	8.00	17.07	36.41	77.66

Tabla 1.7 Tamaño de los Datos de C++ Object

[1] Proxy Based Estimating

1.1.2.5.2 Cálculo

En la figura 4 se describe el método PROBE de estimación del tamaño.

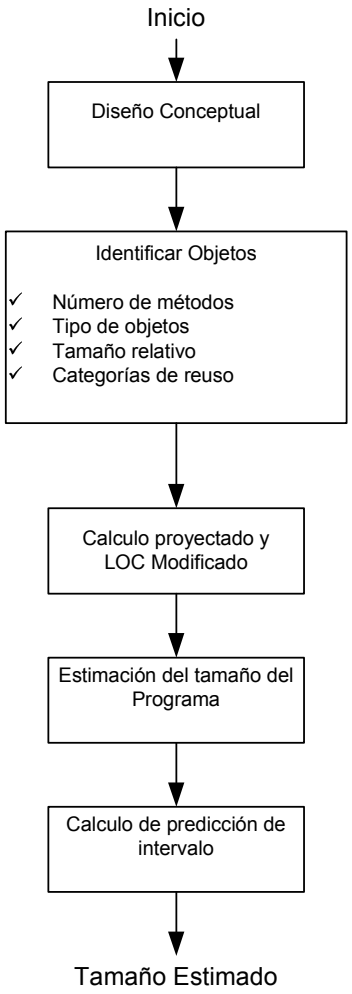


Figura 1.4 Método PROBE PARA Estimación del Tamaño

1.1.2.5.3 Recursos estimados con PROBE

El método PROBE usa regresión lineal para estimar recursos de desarrollo. De nuevo, esta estimación esta basada en tamaño estimado contra los datos de esfuerzo reales de por lo menos tres proyectos anteriores. Los datos deben demostrar una correlación razonable entre el tamaño del programa y el tiempo de desarrollo. El PSP requiere que los r^2 para la correlación sean por lo menos 0,5.

Una vez que ellos han estimado un tiempo total para el trabajo, los ingenieros usan sus datos históricos para empatar el tiempo necesario para cada fase del trabajo. Esto se usa en la columna % A la fecha en el resumen de plan de proyecto.

Usando éstos porcentajes como una guía, los ingenieros asignan su tiempo de desarrollo total estimado para la planificación, diseño, revisión del plan, codificación, revisión del código, compilación, prueba de unidad, y de postmortem. Cuando finaliza, ellos tienen una estimación para el tamaño del programa, el tiempo de desarrollo total, y el tiempo requerido para desarrollar cada fase.

1.1.2.6 Recolectando Datos en el PSP

En el PSP, los ingenieros utilizan su información personal para monitorear su trabajo y ayudarse a planificar mejor. Para hacer esto, ellos recogen datos del tiempo que ellos gastan en cada fase del proceso, los tamaños de los productos que ellos producen, y la calidad de estos productos.

1.1.2.6.1 Medida del Tiempo

En el PSP, los ingenieros usan el tiempo en el registro del Log para medir el tiempo gastado en cada fase del proceso. En este Log, se registra la hora en que inicio la tarea, el tiempo en que se detiene la tarea, y cualquier tiempo de interrupción. Por ejemplo, una interrupción sería una llamada telefónica, un descanso breve, o alguien interrumpiendo para hacer una pregunta.

Los ingenieros rastrean el esfuerzo gastado realmente en las tareas del proyecto. El tiempo de interrupción es al azar y esto reduce la estimación exacta.

1.1.2.6.2 Medida del tamaño

Desde el momento en que se establece el desarrollo de un producto, este esta determinado por el tamaño de ese producto, al usar el PSP, los ingenieros primero estiman el tamaño del producto que se planifica desarrollar.

Entonces, cuando han terminado, miden los tamaños de los productos que produjeron. Esto proporciona a los ingenieros los datos del tamaño que necesitan para hacer las estimaciones del tamaño estimativo. Sin embargo, para que estos datos sean útiles, la medida del tamaño debe poner en correlación con el tiempo de desarrollo del producto.

Mientras las líneas de código (LOC) son las principales medidas de tamaño del PSP, cualquier tamaño, puede ser usado para proveer una correlación razonable entre tiempo de desarrollo y tamaño del producto.

1.1.2.6.3 Líneas de Código (LOC)

El PSP usa el término “LOC lógico” para referirse a una construcción lógica del lenguaje de programación utilizado. Hay muchas maneras de definir LOC lógico subsecuentemente, los ingenieros deben definir precisamente cómo ellos piensan medir el LOC. En un equipo de trabajo o una organización grande de software, se deben usar las normas de LOC establecidas.

Si no hay una norma, el PSP guía a los ingenieros a definir su propia norma. El PSP requiere que los ingenieros midan el tamaño de los programas que producen, y que midan manualmente el tamaño del programa, el tiempo consumido y el tiempo inexacto, el PSP también guía a los ingenieros que pongan por escrito una contabilidad del LOC para el uso del curso de PSP.

1.1.2.6.4 Categorías del tamaño

Para rastrear cómo el tamaño de un programa cambia durante desarrollo, es Importante considerar varias categorías de producto LOC.

Estas categorías son:

- **Base.-** El código LOC base es el tamaño de la versión original del producto antes de que ninguna modificación sea hecha.

- **Sumado.-** El código sumado es un código escrito del nuevo programa o sumado al programa base existente.

- **Modificación.-** El LOC modificado es el código base del producto existente que ha sido cambiado.

- **Eliminado.-** El LOC eliminado, es el código eliminado del programa base.

- **Nuevo y Cambiado.-** Cuando los ingenieros desarrollan un software les toma mucho tiempo añadir o modificar un LOC que no elimine o reuse uno ya existente.

- **Reusado.-** En el PSP, el código LOC reusado se toma de una biblioteca reusable y usable, sin modificar la versión del nuevo programa. Reusar no se cuenta como un código base de una versión de un programa anterior y no se cuenta como un código de modificación reusada.

- **Nuevo reuso.-** El nuevo reuso mide y cuenta el LOC que él ingeniero desarrollo y contribuyó a la biblioteca de reuso.

- **Total.-** El LOC total es el tamaño total del programa, sin tener en cuenta el recurso del código.

1.1.2.6.5 Contabilidad del tamaño

Cuando se modifican los programas, es necesario rastrear los cambios hechos al programa original. Estos datos se usan para determinar el volumen de producto desarrollado, la productividad de los ingenieros, y calidad del producto. Para proporcionar estos datos, el PSP usa el método de contabilidad del tamaño para rastrear todas las sumas, eliminaciones, y cambios que se han realizado a un programa.

Para utilizar la contabilidad del tamaño, los ingenieros necesitan la cantidad de los datos del tamaño del código categorízado. Por ejemplo, si un producto de 100,000 LOC fuese utilizado para desarrollar una nueva versión, y había 12,000 LOC de código anulado, 23,000 LOC de código agregado, 5,000 LOC de código modificado, y 3,000 LOC de código reusado, el Nuevo y cambiado LOC es:

N&C LOC = Añadir + Modificar
28,000 = 23,000 + 5,000

Al medir el tamaño total de un producto, los cálculos son los siguientes:

LOC Total = Base – Suprimir + Añadir + Reusado

Ni la modificación ni el “nuevo reuso” del LOC son incluidos en el total.

Esto es porque un LOC modificado puede representarse anulando y un LOC añadido, y el LOC “nuevo reuso” ya están contabilizados para el LOC agregado. Usando esta fórmula, el total del LOC para el anterior ejemplo es:

Total = 100,000–12,000 + 23,000 + 3,000 = 114,000 LOC

Esto es 114 KLOC, donde KLOC simboliza 1,000 LOC.

1.1.2.6.6 Medidas de calidad

El enfoque principal de calidad del PSP está en los defectos. Para Manejar defectos los ingenieros necesitan datos de los defectos que ellos inyectan, las fases en las que ellos los inyectaron, las fases en que ellos los encuentran y los arreglan, y cuánto tiempo les toma arreglarlos. Con el PSP, los ingenieros recolectan datos de los defectos encontrados en cada fase, incluyendo las revisiones, las inspecciones, compilaciones y pruebas. Estos datos se registran en el Log de defectos.

El término defecto se refiere a algo que está equivocado en un programa. Podría ser una falta de ortografía, un error de puntuación, o una declaración del programa incorrecta. Los defectos pueden estar en programas, en planes, en los requisitos, característica técnicas, o en otra documentación. Los defectos pueden ser encontrados en cualquier fase del proceso, y ellos pueden ser declaraciones redundantes o extras, declaraciones incorrectas, u omisiones del programa.

Un defecto es cualquier cosa que restringe la funcionalidad del programa para completar y efectuar la necesidad del usuario. Así un defecto es una cosa objetiva. Es algo que los ingenieros pueden identificar, describir, y explicar.

Con tamaño, tiempo, y datos del defecto, hay muchas maneras de medir, evaluar, y manejar la calidad de un programa. El PSP proporciona un juego de medidas de calidad que ayudan a los ingenieros a examinar la calidad de sus programas desde varias perspectivas. Mientras ninguna medida unitaria puede adecuadamente indicar la calidad global de un programa, el cuadro provisto por las medidas del conjunto PSP, es generalmente un indicador de calidad fiable.

Las medidas de calidad principales del PSP son:

- Densidad del defecto.
- Razón de revisión.
- Razón del tiempo de desarrollo.

- Razón del defecto (MEDIDA DE ALGUNA CANTIDAD).
- Productividad.
- Defectos por hora.
- Defectos renovables viables.
- Nivelación de la razón de falla (A/FR).

Cada una de estas medidas se describe en los siguientes párrafos.

Densidad del defecto. La densidad del defecto se refiere a los defectos por nuevo y cambió KLOC encontrados en un programa. Así, si un programa de 150 LOC tuviese 18 defectos, La densidad del defecto sería

1000*18/150 = 120 defectos/KLOC

La densidad del defecto es medida para el proceso de desarrollo completo y para el proceso específico por fases. Sólo probar una fracción de los defectos en un producto, cuando hay más defectos hace que permanezcan allí hasta cuando la fase de prueba este completa.

En el PSP, se considera que un programa con cinco o menos defectos/KLOC en prueba de unidad es de buena calidad. Para los ingenieros que no han sido entrenados con el PSP el defecto en la prueba de unidad, tiene un rango de 20 a 40 o más defectos/KLOC.

Razón de Revisión. En el PSP la revisión del diseño y del código, la realizan los ingenieros. Los datos del PSP muestran que cuando los ingenieros han revisado el diseño o el código a una razón mayor a 150 a 200 LOC por hora, no localizan muchos defectos. Con el PSP, los ingenieros recogen datos de sus revisiones y determinan que tan rápido deben revisar sus programas para encontrar todos o la mayoría de los defectos.

La razón del tiempo de desarrollo. Se refiere a la cantidad de tiempo utilizado por cada ingeniero en cualquiera de las fases de desarrollo.

En el PSP, las tres razones de tiempo de desarrollo usadas en el proceso de evaluación son:

- Tiempo de diseño / tiempo de codificación
- Tiempo de revisión del diseño / tiempo de diseño.
- Tiempo de revisión del código / tiempo de codificación.

La primera relación es la más útil en determinar el diseño y calidad del trabajo del ingeniero. Cuando los ingenieros gastan menos tiempo diseñando que codificando, ellos están produciendo la mayoría del diseño mientras codifican. Esto significa que el diseño no se documenta, el diseño no puede ser revisado o inspeccionado, y la calidad del diseño es probablemente pobre. El indicador del PSP es que los ingenieros deben utilizar el mismo tiempo produciendo un diseño detallado, como produciendo un código de su diseño.

El indicador del PSP es que los ingenieros deben pasarse por lo menos el 50% del tiempo revisando el diseño como produciendo el diseño. El uso del PSP da un indicador aproximado del 50% tanto para la revisión como para la codificación.

Razón del defecto. La tasa de defectos del PSP se compara entre los defectos encontrados en una fase y aquéllos encontrados en otra. Las principales tasas de defecto son encontradas en la revisión del código, dividida por defectos encontrados al compilar, los defectos encontrados en revisión del diseño y por los por defectos encontrados en prueba de unidad. Una regla razonable es la de un pulgar, porque los ingenieros deben encontrar por lo menos dos defectos cuando revisan el código que compilan. El número de defectos encontrados mientras compilamos es la medida objetiva de calidad del código.

Cuando los ingenieros encuentran más de dos defectos en la revisión del código compilado, esto generalmente quiere decir que ellos han hecho una revisión del código competente o no grabaron todos los defectos compilados. Los datos del PSP también sugieren que la unidad del diseño revisada es de una tasa de 2 o más. Si los ingenieros encuentran dos defectos durante la

prueba de unidad del diseño, ellos probablemente han hecho una revisión del diseño aceptable.

El PSP usa estas dos medidas en combinación con proporciones de tiempo y medidas de densidad de defecto para indicar si un programa está listo para la integración y comprobación del sistema.

Si cualquier medida es pobre, el criterio de calidad del PSP sugiere que es probable que el programa tenga problemas en las pruebas o posteriormente.

Rendimiento. En el PSP, el producto es medido de dos maneras. La medida del producto en una fase, es un porcentaje del total del defecto que se ha encontrado y ha sido removido en la fase. Por ejemplo, si un programa entra en una prueba de unidad con 20 defectos y se encuentran 9, la prueba de unidad del producto en la fase sería 45%.

Similarmente, si un programa entra en la revisión del código con 50 defectos y en la revisión se encuentra 28, el código de producción de la fase sería 56%. El rendimiento del producto se refiere al porcentaje de los defectos removidos antes de la primera compilación y prueba de unidad. Desde que el objetivo del PSP es producir programas de alta calidad, el indicador sugerido para el rendimiento del proceso es mayor al 70%.

La medida del producto precisamente no puede calcularse hasta el extremo de la vida útil de un programa. Por que entonces, probablemente, todos los defectos se habrían encontrado y se habrían informado. Cuando los programas son de alta calidad, sin embargo, se estima que pueda ser un producto razonablemente bueno.

Por ejemplo, si un programa tuviese los datos del defecto mostrados en tabla 1.8, el indicador del PSP por producto estimado, es asumir que el número de defectos que permanecen en el programa es igual al número encontrado en la última fase de comprobación. Así, en Tabla 1.8, es probable que siete defectos permanezcan después de la comprobación de la unidad.

Fase	Defectos removidos
Revisión del diseño	11
Revisión del código	28
Compilar	12
Prueba de unidad	7
Total	58

Tabla 1.8 Ejemplo de defectos removidos

Para Calcular el rendimiento, los ingenieros usan los datos de las fases en las que los defectos fueron inyectados. Ellos también necesitan asumir que los defectos inyectados que quedan después de la prueba de unidad sean los mismos que en la fase encontrados durante la prueba de unidad. Un ejemplo de defectos inyectados y removidos se encuentra en la Tabla 1.9.

Fase	Defectos Inyectados	Defectos Removidos
Diseño detallados	26	0
Revisión de diseño	0	11
Código	39	0
Revisión de código	0	28
Compilar	0	12
Prueba de unidad	0	7
Después de la prueba de unidad	0	7
TOTAL	65	65

Tabla 1.9 Ejemplos de defectos Inyectados y Removidos

Como señala la tabla 1.10, los defectos totales inyectados, incluyen aquéllos estimados, que son 65. En la revisión del diseño de entrada, los defectos presentados fueron 26, para que el rendimiento de la revisión del diseño fuera 100*11/26 = 42.3%. En la entrada de la revisión del código, los defectos

presentados tuvieron un total de 65 menos aquéllos removidos en la revisión del diseño, o 65–11 = 54, por tanto que el rendimiento de la revisión del código fue 100*28/54 = 51.9%.

Similarmente, el rendimiento de compilación fue 100*12/(65-11-28) = 46.2%. El rendimiento del proceso es la mejor medida global de la calidad del proceso. Es el porcentaje de defectos inyectado antes de compilar. Para los datos en la tabla 1.10, el rendimiento del proceso es 100*39/65 = 60.0%.

Fase	Defectos Inyectados	Defectos Removidos	Defectos en la fase de entrada	Fase de Cálculo
Diseño detallados	26	0	0	
Revisión de diseño	0	11	26	42.3 %
Código	39	0	15	
Revisión de código	0	28	54	51.9%
Compilar	0	12	26	46.2%
Prueba de unidad	0	7	14	50.0%
Después de la prueba de unidad	0	7	7	
TOTAL	65	65		

Tabla 1.10 Cálculos de Rendimiento

Defectos por Hora. Con los datos del PSP, los ingenieros pueden calcular los números de defectos que ellos han inyectado y removido por hora. Ellos pueden utilizar esta medida para guiarse en su planificación personal. Por ejemplo, si un ingeniero inyectó cuatro defectos por hora codificando y arregló ocho defectos por hora en revisiones del código, él o ella necesitaría gastar aproximadamente 30 minutos en revisiones del código durante todas las horas gastadas codificando. Tomaría un largo tiempo en encontrar y arreglar los defectos que eran probablemente inyectados. Los defectos por hora pueden ser calculados del plan del proyecto en una forma condensada.

Influencia de la eliminación del defecto (DRL). La influencia de la eliminación del defecto mide la efectividad relativa de dos fases de levantamiento de defecto. Por mencionar, de los ejemplos anteriores, el levantamiento del defecto del diseño revisado sobre la prueba de unidad es 3.06/1.71 = 1.79. Esto significa que el ingeniero será 1.79 veces más eficaz para encontrar defectos en revisiones del diseño como en comprobación de la unidad. Los DRL ayudan a medir el diseño para que los ingenieros tengan un plan más eficaz de remover.

A/FR. La apreciación a la proporción de la falla (A/FR) es la medida de la calidad del proceso de la ingeniería, usando parámetros de costo de la calidad.

La **A** representa levantamiento del costo de calidad o el porcentaje del desarrollo del tiempo gastado en calidad y en actividades superiores.

La **F** en A/FR representa la falla de la calidad del costo, que es el tiempo gastado en recuperar la falla y repararla. El costo de la falla es el tiempo gastado en compilar y realizar la prueba de unidad, el tiempo gastado, encontrando, arreglando, recompilando, y reprobando los defectos encontrados.

La medida de A/FR proporciona una manera útil de evaluar la calidad, para comparar la calidad de los procesos de desarrollo usada para los programas individuales o para varios programas. También indica el grado en el cual el ingeniero intentó encontrar y arreglar los defectos desarrollados en el proceso.

1.1.2.7 Diseño del PSP

Mientras el PSP puede usarse con cualquier método de desarrollo, este requiere que el diseño este completo. El PSP considera que un diseño está completo cuando define las cuatro dimensiones mostradas en la Tabla 1.11.

Especificación del Objeto	Interna	Externa
Estática	Atributos Comprimidos	Herencia de la clase Estructura
Dinámica	Máquina de Estado	Servicios Mensajes

Tabla 1.11 Especificación de la Estructura del Objeto

El método que estas cuatro plantillas corresponden al objeto de la estructura de especificación que se muestra en la Tabla 1.12. El PSP tiene cuatro plantillas diseñadas que se dirigen estas dimensiones.

Especificación de la Plantilla del Objeto	Interna	Externa
Estática	Plantillas de especificación Lógica	Especificación de la función de las plantillas (Herencia Tipo Estructura)
Dinámica	Plantillas de la Máquina de Estado	Especificación de la función de las plantillas (Interacción del usuario) Escenario Operacional de la Plantilla

Tabla 1.12 Las estructuras de las plantillas del PSP

- El Escenario Operacional de la Plantilla define como debe interactuar el usuario con el sistema.
- La Especificación de la función en las plantillas especifica el comportamiento de llamadas de retorno de los objetos y las clases de herencia de la clase estructura.
- Los estados específicos de las plantillas definidas en el programa definen el comportamiento del estado de la máquina.

- Las Plantillas de especificación Lógica, especifican la lógica interna del programa, normalmente en un segundo código como lenguaje.

1.1.2.8 Disciplina del PSP

En cualquier campo de la ingeniería el problema principal es conseguir que ingenieros usen de forma consistente los métodos que ellos han aprendido. En el PSP, estos métodos están siguiendo un proceso definido, planificando el trabajo, recogiendo datos, y usando estos datos para analizar y mejorar el proceso. Mientras esto parece simple en concepto, no es fácil en la práctica.

1.1.2.9 Introduciendo el PSP

La Introducción del PSP en las universidades comenzó con cursos de PSP en los cuales los estudiantes escribían 10 programas y completaban 5 reportes de datos de análisis. El curso universitario típicamente dura un semestre completo, durante el cual los estudiantes siguen el proceso del PSP mostrado en la tabla 5 para completar 10 ejercicios del programa.

Los estudiantes empiezan con el proceso del PSP0, donde ellos usan su las prácticas mas utilizadas en la actualidad, registrando el tiempo de cada fase del proceso, y anotando todos los defectos que ellos encuentran.

El proceso del PSP se refuerza a través de siete versiones del proceso, con estudiantes escribiendo uno o dos programas con cada versión del PSP. Para cada programa, ellos usan métodos del proceso que acaban de introducir, así como todos los métodos introducidos previamente en la versión anterior del proceso. A finales del curso, los estudiantes han escrito 10 programas y han aprendido a usar todos los métodos del proceso PSP. Este curso fue diseñado para graduados y los que están por graduarse. Es mostrado en la figura 1.5.

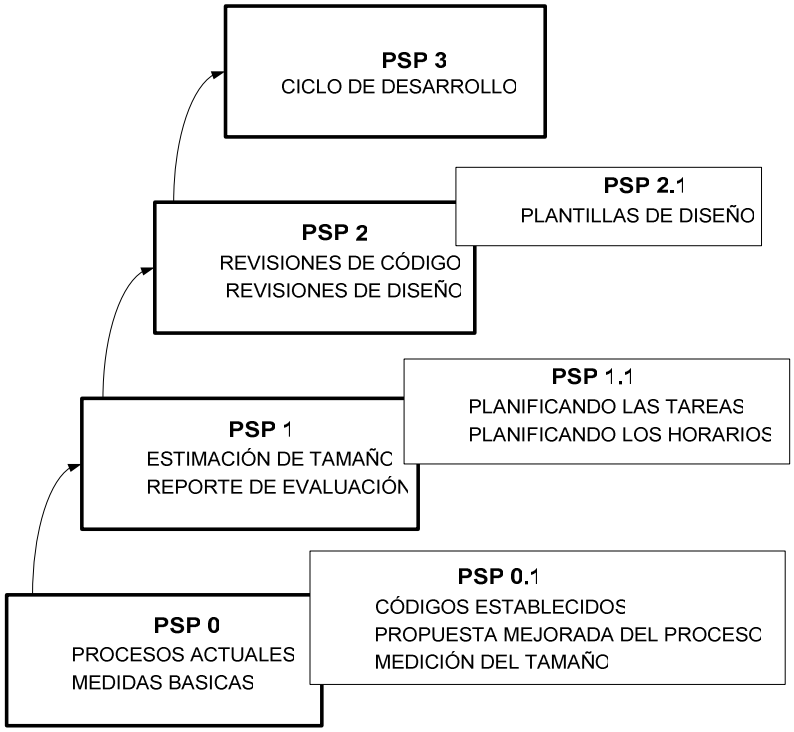


Figura 1.5 PSP/TSP Estructura del Curso

Hay también un curso de PSP para principiantes universitarios. Mientras este curso introductorio enseña los mismos principios básicos del PSP como el curso avanzado, este no es tan riguroso y los estudiantes generalmente no aprenden el método global de práctica del PSP. Sin embargo, ellos comienzan usando prácticas de ingeniería auditiva en sus primeras prácticas en las asignaturas de programación. Entonces lo más probable es que ellos desarrollen una práctica personal auditiva, que les facilita la enseñanza de la ingeniería rigurosa cuando más tarde ellos tomen el curso completo del PSP.

Al introducir el PSP en industria, los ingenieros completan el curso del PSP y escriben todos 10 ejercicios de la serie A del programa en el texto del PSP. Este curso tiene una duración aproximada 120 a 150 horas de ingeniería, y generalmente dura más de 14 días laborales. Después de que los ingenieros son entrenados en el PSP, la experiencia ha mostrado que ellos tienen manejarse apropiadamente y tener la consistencia para respaldarse con el uso del PSP en su trabajo. Para ayudar en el trabajo los ingenieros, equipos, y

gerentes usan los métodos del PSP de forma consistente, el SEI ha desarrollado el Proceso de Software de equipo (TSP).

1.1.2.10 El PSP y Mejoras del Proceso

El PSP es uno de las tres series del proceso de mejoras complementario. Estas Son:

- El modelo de la capacidad de madurez (CMM), quien gerencia la dirección de los sitios de software.
- El Personal Software Process (PSP) es un mejoramiento en si de los procesos diseñados para ayudar a controlar, administrar y mejorar su manera de trabajar.
- El Proceso de Software en equipo (TSP), que guía al ingeniero entrenado en PSP, equipos y gerentes en el desarrollo intensivo de los productos del software.

Mientras estos métodos son todos relacionados, ellos se dirigen hacia diferentes aspectos de la capacidad organizacional.

Primero, para producir productos superiores, la dirección de la organización tiene que tener una estrategia legítima del negocio y una programación que coloque el producto según las necesidades del mercado.

Segundo, para producir productos superiores, la dirección debe obtener un trabajo superior de sus ingenieros. Esto requiere que ellos contraten personal capacitado y les proporcione una conveniente orientación, procesos, y apoyo.

Tercero, los ingenieros deben poder trabajar juntos eficazmente en un ambiente de equipo y sepa cómo producir productos de calidad de forma consistente.

Cuarto, los equipos de ingeniería deben entrenarse propiamente y capacitarse en hacer disciplina en su trabajo de ingeniería.

Sin cualquiera de estas condiciones, las organizaciones no realizan un trabajo superior.

En la anterior lista, el CMM se diseño para la segunda capacitación, el TSP para la tercera capacitación, y el PSP para la cuarta capacitación. A menos que los ingenieros tengan el entrenamiento provisto por el PSP ellos no pueden respaldar apropiadamente a su equipo o consistencia y producir productos de calidad confiable. A menos que propiamente se formen equipos y líderes, ellos no pueden manejar y rastrear su trabajo o encontrar horarios.

Finalmente, sin dirección apropiada y liderazgo ejecutivo, las organizaciones no pueden tener éxito.

1.1.2.11 Los Estatutos del PSP y las Tendencias del Futuro

En el futuro, los grupos de ingeniería de software, requerirán que se incremente la entrega de productos de calidad, a tiempo, y con costos planeados.

La Fiabilidad garantizada y nivel de servicio, seguridad de garantía y penalidades en el contrato serán normales y el grupo de ingenieros que no puedan cumplir estas obligaciones no sobrevivirán.

Mientras el trabajo técnico continué siendo requerido, la actuación individual de cada ingeniero será reconocida como importante. Los sistemas de calidad requieren partes de calidad, y a menos que cada ingeniero se esfuerce por producir trabajo de calidad, el equipo no puede funcionar. Dirección de calidad será una parte integral del entrenamiento del ingeniero de software. Los ingenieros tendrán que aprender cómo medir la calidad de su trabajo y cómo usar estas medidas para producir trabajos libres de defecto.

El PSP fue diseñado para proporcionar prácticas disciplinadas profesionales del software que se necesitarán en el futuro. Mientras algunas organizaciones industriales están introduciendo estos métodos, una introducción más amplia, de métodos disciplinados debe empezar en las universidades.

La introducción académica del PSP es actualmente es apoyada en dos formas; cursos introductorios y de nivel avanzado. Varias universidades en el U.S., Europa, y Australia ahora ofrecen el PSP, y varias instituciones en Asia están considerando su introducción. El SEI, junto con varias universidades, también apoya un taller de verano de duración de una semana por facultad a quien enseña o desea enseñar el PSP.

La introducción del PSP en industrial también es apoyada por el SEI con un curso de PSP para ingenieros e instructores. Este curso se ofrece varias veces al año en un formato condensado estructurado en un formato de necesidades industriales. El SEI asiste o respalda a organizaciones de instructores del PSP calificados para introducir el PSP y mantiene un registro de instructores de PSP calificados.

Mientras el PSP es relativamente nuevo, los resultados son prometedores. Ambos Usos industriales y académicos lo adoptan en aumento.

1.2 DESARROLLO DE SOFTWARE USANDO PSP Y RUP

1.2.1 Descripción del RUP.

Es un proceso de ingeniería de software, el cual provee una disciplina que enfoca la asignación de tareas y responsabilidades dentro de la organización, tiene por objetivo garantizar la producción de software de alta calidad conociendo las necesidades de los usuarios finales, dentro de un horario y presupuesto previsto.

El RUP garantiza la productividad del equipo, proporcionando a cada miembro del equipo el acceso fácil a una misma base de conocimiento, a plantillas y herramientas, es por ello que comparten un idioma común. Además es una guía de cómo usar el UML de manera eficaz. El UML es un lenguaje estándar de la industria que nos permite comunicar claramente los requisitos, las arquitecturas y los planes.

El RUP[2] es el producto final de tres décadas de desarrollo y uso práctico, como se muestra en la figura 1.6.

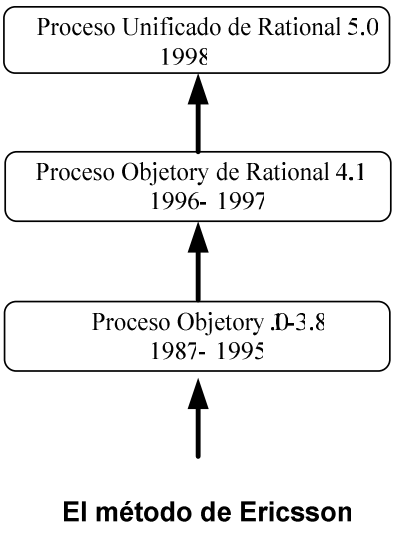


Figura 1.6 Evolución del RUP.

[2] Rational Unified Process

1.2.1.1 Las Seis Mejores Prácticas

El RUP propone el uso de las mejores prácticas desarrollo del software que son usadas por las empresas más exitosas.

Estas mejores prácticas son las siguientes:

1. Desarrollo iterativo del software.
2. Administración de requerimientos.
3. Uso de arquitectura basada en componentes.
4. Modelamiento visual del software.
5. Verificación de la calidad del software.
6. Control de cambios del software.

1.2.1.1.1 Desarrollo iterativo del Software

El software moderno es complejo y sofisticado lo cual hace que no sea posible seguir una secuencia lineal, como lo hace el modelo en cascada que se muestra en la Figura 1.7.

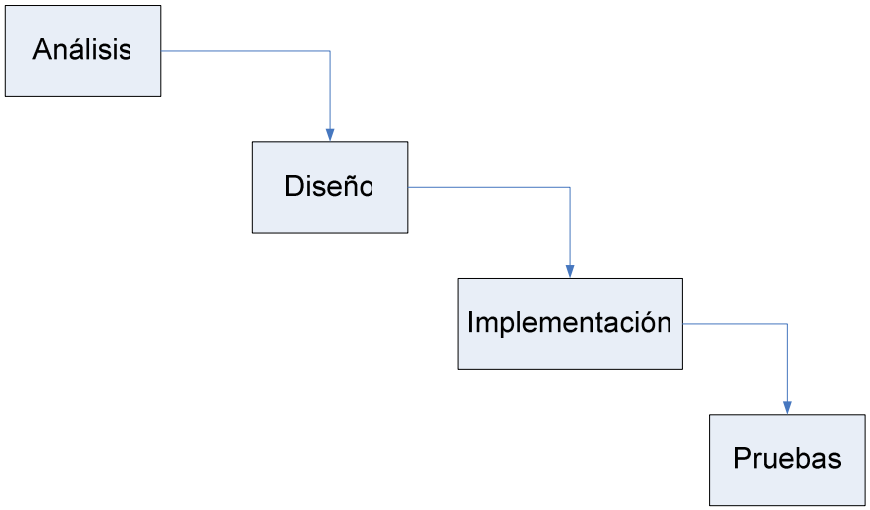


Figura 1.7 Modelo Cascada

Iterar.- Es cumplir y volver sobre lo mismo hasta que las prioridades sean completadas satisfactoriamente, como se muestra en la Figura 1.8.

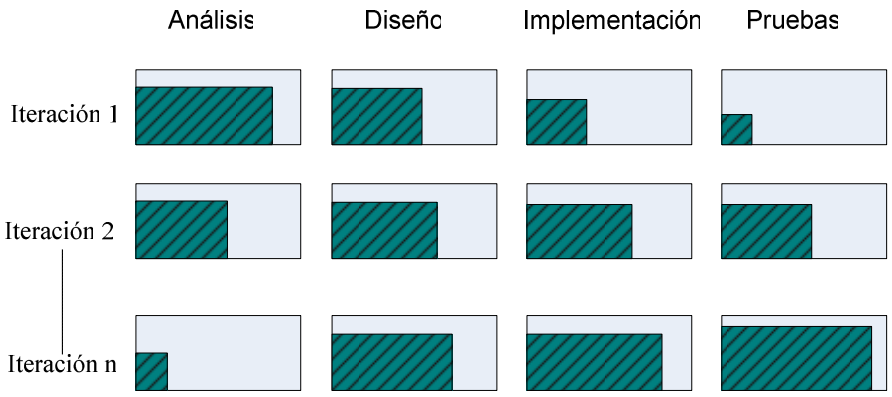


Figura 1.8 Modelo Iterativo

Iterativo.- El RUP permite trabajar con un ciclo iterativo porque permite definir ciclos en los cuales se realiza todas las actividades del proceso de desarrollo.

Este proceso iterativo ayuda a reducir los riesgos y permiten retroalimentación ya que cada iteración termina como un ejecutable, el equipo de desarrollo se mantiene en constante producción de resultados y con seguridad, mantiene al proyecto bajo un horario.

Incremental.- Para que al cumplir cada ciclo se añada más información al proyecto, cada ciclo aporta algo más al desarrollo del proyecto.

Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo

1.2.1.1.2 Administración de requerimientos

El RUP describe:

- Como obtener los requerimientos.
- Como organizarlos.
- La forma de documentar requerimientos de funcionalidad y restricciones.

- Rastrear y documentar decisiones.
- Captar y comunicar requerimientos de negocio.

De tal manera los requerimientos guían el diseño, la implementación y las pruebas y lo hacen más similar al sistema final, cumpliendo con todas las necesidades del usuario.

Para Administrar los requerimientos se debe tener centralizada la información de requerimientos y poder realizar cambios y tener un histórico de ellos, el mismo que tiene la siguiente información:

- Quien realizo los cambios.
- Cuando se realizaron.
- Que cambios realizo y
- Porque.

1.2.1.1.3 Uso de arquitectura basado en componentes

El proceso enfoca un desarrollo temprano, bajo una arquitectura flexible, entendible y cambiante. Los componentes son módulos no triviales. El RUP apoya el desarrollo basado en componentes, tanto nuevos como preexistentes.

Se usan arquitecturas que separen los servicios de las aplicaciones, por lo menos de deben usar tres capas, el uso de estas arquitecturas proveen transaccionalidad a los procesos. El uso de componentes permite rehusar el software.

Entre otras las arquitecturas que permiten eso son:

- DNA – COM
- .Net – COM, Web Services
- J2EE – EJB

1.2.1.1.4 Modelamiento Visual del Software

El proceso indica como visualizar el modelo de software, para capturar la estructura y el comportamiento de los componentes. Los bloques de construcción permiten ocultar detalles, permite la comunicación en el equipo de desarrollo, permite analizar la consistencia entre los componentes, el diseño e implementación. Permite ver como los elementos del sistema trabajan juntos, asegura que la construcción de los bloques sea consistente con el código.

Dejar a un lado los extensos documentos que a la larga no permiten tener una administración de proyectos manejables. Se apoya en el modelo gráfico y poco en la documentación. El UML es la base del modelamiento visual del RUP.

1.2.1.1.5 Verificación de la calidad del software

El RUP ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifican la calidad del software. El aseguramiento de la calidad es parte del proceso de desarrollo y no solo la funcionalidad es esencial, también el rendimiento y la confiabilidad.

1.2.1.1.6 Control de cambios en el software

La habilidad de manejar los cambios, asegura que cada cambio sea aceptado, el proceso describe como controlar los cambios para que el proceso iterativo sea exitoso. Hace que el grupo de desarrollo se vea como una sola unidad. Los cambios son inevitables, pero es necesario evaluar si éstos son necesarios y rastrear su impacto.

1.2.1.2 Estructura del RUP

El RUP se fundamenta en el desarrollo iterativo e incremental, se desarrolla en base a iteraciones. El RUP describe dos dimensiones identificables en dos ejes, que son:

- El eje horizontal representa el tiempo y los aspectos dinámicos (FASES).
- El eje vertical representa las actividades y los aspectos estáticos (FLUJOS DE TRABAJO).

Un proceso describe quien está haciendo, que, como y cuando. El RUP está representado por los siguientes elementos:

- Trabajador, “quien”
- Actividades, “como”
- Artefactos, “que”
- Flujos de Trabajo, “cuando”

Trabajador

Un trabajador define el comportamiento y las responsabilidades de un individuo. Es el rol que desempeña en un momento dado. Las responsabilidades pueden ser: Hacer una serie de actividades, Ser el responsable de una serie de artefactos.

Actividades

Una actividad es una unidad de trabajo que se asigna a un trabajador. Una actividad lleva entre un par de horas y un par de días, involucra un solo trabajador y un número pequeño de artefactos. Una actividad puede ser usada como un elemento parar planear y progresar.

Artefactos

Un artefacto es la pieza de información que es producida, modificada o usada por un proceso. Los artefactos son usados como una entrada de los

trabajadores, para una actividad y son el resultado de salidas para cada actividad. Los artefactos pueden tomar varias formas:

- Un modelo.
- Un documento.
- Código.
- Ejecutable.

Flujos de trabajo

Los flujos de trabajo son la enumeración de los trabajadores, artefactos y actividades que no constituyen el proceso. Un flujo de trabajo es una secuencia que produce un resultado de un valor observado. En UML un flujo de trabajo representa un diagrama de secuencia, de colaboración o de actividad. Es importante considerar que no siempre se puede representar las dependencias entre actividades.

El RUP propone nueve Flujos de Trabajo Fundamentales, que se dividen en "ingeniería" y "de soporte" que forman el eje vertical.

Los flujos de trabajo de ingeniería o de desarrollo de software son:

- Modelado de negocios.
- Requerimientos.
- Análisis y diseño.
- Implementación.
- Pruebas.
- Implantación.

Los flujos de trabajo de soporte son:

- Administración del proyecto.
- Configuración y Administración de cambios.
- Entorno.

El RUP propone cuatro Fases en el desarrollo de software que forman el eje horizontal:

- Incepción (Inicio - Concepción).
- Elaboración (Diseño).
- Construcción.
- Transición.

Cada fase tiene asociado un conjunto de actividades, procesos y entregables. De acuerdo a la fase en la que se encuentre el proceso de desarrollo se ejecutaran más ó menos actividades relacionadas con esa fase. Cuando se han completado las 4 primeras fases del desarrollo se dice que se ha liberado el primer Release.

Lo dicho anteriormente se resume en la siguiente figura 1.9:

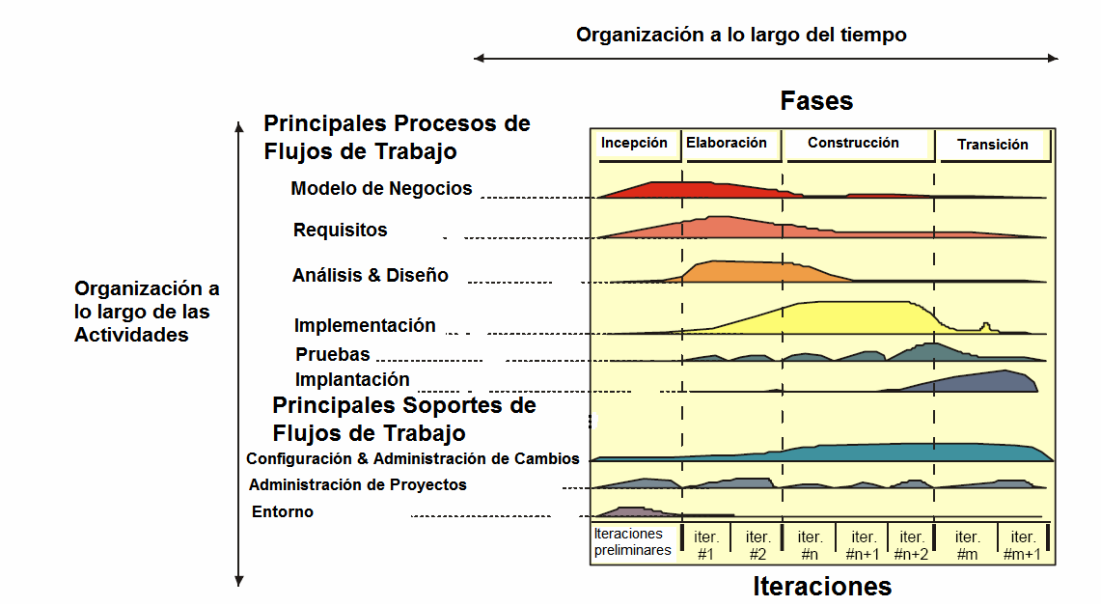


Figura 1.9 Dos Dimensiones del RUP

Al revisar la bibliografía en inglés y en español se encuentran los siguientes detalles que son:

- En español se tienen 5 flujos de trabajo y que el análisis y diseño van separados.
- En inglés se tiene 6 flujos de trabajo y que el análisis y diseño van como un solo flujo.

En el presente trabajo se va a considerar los 6 flujos de trabajo como se muestra en la figura 8.

1.2.1.2.1 Fases del RUP

1.2.1.2.1.1 Concepción (Inicio)

Establece el inicio del proyecto. Se establecen los objetivos y el alcance del sistema, se identifican los riesgos potenciales que pueden afectar el desarrollo del proyecto.

En esta fase se realizan las siguientes actividades:

- Realizar el modelo de negocios.
- Recopilar los requerimientos.

Como resultado de esta fase se obtienen los siguientes entregables:

- Requerimientos generales del proyecto.
- Visión general del proyecto.
- Plan inicial del proyecto.
- Características generales.
- Restricciones.
- Arquitectura inicial.
- Entorno de desarrollo.
- Glosario del proyecto.
- Diagrama del caso de uso del Negocio.
- Diagramas de casos de uso con su respectiva descripción.

1.2.1.2.1.2 Elaboración (Diseño)

Es la encargada de definir la arquitectura en que se implementara. Se encarga de definir el alcance real del proyecto basándose en los modelamientos de:

- Requerimientos.
- Arquitectura.
- Plataforma.

En esta fase se realiza la captura de la mayor parte de los requerimientos funcionales, manejando los riesgos que interfieran con los objetivos del

sistema, acumulando la información necesaria para el plan de construcción y obteniendo suficiente información para hacer realizable el caso del negocio. Se realizan los manuales preliminares del sistema y se construye un prototipo arquitectónico, se construyen los modelos del sistema.

En esta fase se realizan las siguientes actividades:

- Afinamiento en la recopilación de los requerimientos.
- Análisis y diseño del sistema.
- Definición de la arquitectura.

Como resultado de esta fase se obtienen los siguientes entregables:

- Modelos de casos de uso al 80%.
- Descripción de la arquitectura de software final.
- Modelos:
    - Diagrama de Clases.
    - Diagrama de Secuencia.
    - Diagrama de Colaboración.
    - Diagrama de Actividad.
    - Diagrama de Estado.
    - Preliminar del diagrama de deployment.
- Diseño preliminar de la arquitectura del repositorio de datos.
- Prototipo funcional de la arquitectura.
- Manual de usuario preliminar.

La fase de elaboración es la fase más crítica de las cuatro fases. Al finalizar esta fase, la ingeniería dura está completa. Las actividades de la fase de elaboración aseguran que la arquitectura, requerimientos y el plan son estables.

1.2.1.2.1.3 Construcción

En esta fase se alcanza la capacidad operacional del producto, a través de sucesivas iteraciones e incrementos se desarrolla un producto software, listo para operar, éste es frecuentemente llamado versión beta. Es esta fase se

pone mucho énfasis en el manejo de recursos y control de operaciones para optimizar los costos.

En esta fase se realizan las siguientes actividades:

- Desarrollo de todos los componentes la arquitectura.
- Llegar a una versión estable del sistema.
- Realización de las pruebas:
    - Pruebas unitarias.
    - Pruebas de integración.
- Registrar los resultados.

Como resultado de esta fase se obtienen los siguientes entregables:

- Código fuente del proyecto con todas sus etapas.
    - Presentación.
    - Lógica del negocio.
    - Datos.
- Plan de transición inicial:
    - Planos de capacitación.
    - Plan de distribución y ventas.
- Manuales finales:
    - Usuario – Manejo del sistema.
    - Técnico – Como se instala.

Al finalizar esta fase se está en capacidad de definir si el software y los usuarios son operacionales.

1.2.1.2.1.4 Transición

En esta fase se realiza la entrega del producto operando, una vez realizadas las pruebas de aceptación por un grupo especial de usuarios y habiendo efectuado los ajustes y correcciones que sean requeridos.

Esta incluye:

- Prueba beta para validar el nuevo sistema
- Entrenamiento de usuarios y mantenimiento

- Conversión de las base de datos operacional

Esta fase se enfoca en las actividades requeridas para colocar el software en manos de los usuarios. Al finalizar esta fase, usted decide si los objetivos fueron alcanzados.

1.2.1.2.2 Flujos de Trabajo Fundamentales

1.2.1.2.2.1 Modelo del Negocio

El modelo de negocio es una técnica para comprender los procesos de negocios de la organización. El modelado de negocios está soportado por dos tipos de modelos de UML: modelos de casos de usos y modelos de objetos. El modelo de casos de usos se describe mediante diagramas de casos de uso. El modelo de objetos es un modelo interno del negocio.

Un modelo del negocio se desarrolla en dos pasos y estos son:

1. Los modeladores del negocio deben confeccionar un modelo de casos de uso del negocio que identifique los actores del negocio y los casos de uso del negocio que utilicen los actores. De esta manera los moderadores comprenden mejor el valor qué proporciona el negocio a sus actores.
2. Los modeladores deben desarrollar un modelo de objetos del negocio compuesto por trabajadores, entidades del negocio, y unidades de trabajo que junto realizan los casos de uso del negocio. Se asocian a estos diferentes objetos las reglas del negocio y otras normas impuestas por el negocio. El objetivo es crear trabajadores, entidades del negocio, y unidades de trabajo que realicen los casos de uso del negocio de la manera más eficaz posible.

1.2.1.2.2.2 Requisitos

La captura de requisitos es complicada debido a que los desarrolladores de software crean código para otros y no para sí mismos, lo hacen para usuarios finales de software que con frecuencia no saben cuáles son los requisitos ni tampoco como especificarlos de una manera correcta. Sin embargo lo más importante es que los sistemas den soporte a la misión para la cual se construyen.

El propósito fundamental de la captura de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una buena descripción de los requisitos del sistema, para que se pueda llegar a un acuerdo entre el cliente y el desarrollador sobre qué debe y qué no debe hacer el sistema.

Cada proyecto de desarrollo de software es diferente, esta singularidad proviene de las diferencias en el tipo de sistema, en el cliente, en la organización de desarrollo, en la tecnología, etc. Esto hace que se tengan diferentes puntos de partida para la captura de los requisitos. Los diferentes puntos de partida plantean diferentes tipos de riesgos, por lo que los analistas deben elegir las técnicas que reduzcan de mejor forma esos riesgos.

Este flujo de trabajo tiene los siguientes pasos:

- Enumerar los requisitos candidatos.
- Comprender el contexto del sistema.
- Capturar los requisitos funcionales.
- Capturar los requisitos no funcionales.

Enumerar los requisitos candidatos. Es crear una lista de características que crece a medida que aparecen nuevos elementos y disminuye cuando algunas características se convierten en requisitos o en otros artefactos.
La lista de características se utiliza sólo para la planificación del trabajo y tiene un conjunto valores como son:

- Estado.
- Coste estimado de implementación.

- Prioridad.
- Nivel del riesgo asociado a la implementación de la característica.

Comprender el contexto del sistema. Además de ser especialistas en temas relacionados al desarrollo de software los ingenieros deben tener un firme conocimiento del contexto del en el cual se emplaza el sistema. Para expresar el contexto de un sistema en forma utilizable para los desarrolladores de software existe dos técnicas y estas son: Modelado del dominio y modelado del negocio.

- **Modelado del dominio:** Describe los conceptos importantes del contexto como objetos del dominio y enlaza estos objetos unos con otros.
- **Modelado del negocio:** Describe los procesos con el objetivo de comprenderlos.

La técnica para identificar los requisitos se basa en casos de uso, los cuales capturan tanto los requisitos funcionales como los no funcionales.

Capturar los requisitos funcionales. La captura de los casos de uso que realmente se quieren para el sistema, como aquellos que soportarán el negocio son los requisitos funcionales.

Capturar los requisitos no funcionales. La captura de los casos de uso que especifican las propiedades del sistema, restricciones del entorno o de la implementación, rendimiento, facilidad de mantenimiento, extensibilidad y fiabilidad.

1.2.1.2.2.3 Análisis

Este flujo de trabajo se encarga de analizar los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una

descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema completo.

El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptual, al cual se le denomina *modelo de análisis.* El modelo de análisis ayuda a refinar los requisitos y permite razonar los aspectos internos del sistema, además proporciona una estructura centrada en el mantenimiento, en la flexibilidad ante los cambios y la reutilización.

Analizar los requisitos en la forma de un modelo de análisis es importante por varios motivos:

- Ofrece una especificación más precisa de los requisitos que la que tenemos como resultado de la captura de requisitos.
- Se describe utilizando el lenguaje de los desarrolladores.
- Estructura los requisitos de modo que facilita su comprensión y mantenimiento.
- Se considera una primera aproximación al modelo de diseño.

Artefactos

- **Modelo de Análisis**

  Se representa mediante un sistema de análisis que denota el paquete de más alto nivel del modelo.

- **Clase del Análisis**

  Representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema.

- **Realización de caso de uso-análisis**

  Es una colaboración dentro del modelo de análisis que describe como se lleva acabo y se ejecuta un caso de uso determinado en términos de las clases del análisis y de sus objetos del análisis en su interrelación.

- **Paquete del análisis**

  Proporciona un medio para organizar los artefactos de modelo de análisis en piezas manejables.

- **Descripción de la arquitectura (Vista del modelo de análisis)**

Contiene una vista de la arquitectura del modelo de análisis que muestra los artefactos significativos para la arquitectura.

Trabajadores

- **Arquitecto**

Es el responsable de la integridad del modelo de análisis, garantiza que éste sea correcto, consistente y legible como un todo.

- **Ingeniero de casos de uso**

Es el responsable de la integridad de una o más realizaciones de casos de uso, garantiza que se cumplan los requisitos que recaen sobre ellos.

- **Ingeniero de componentes**

Es el que define y mantiene las responsabilidades, atributos, relaciones y requisitos especiales de una o varias clases del análisis, y también mantiene la integridad de uno o varios paquetes del análisis.

Flujo de trabajo

El flujo de trabajo inicia con el arquitecto el cual se encarga del análisis de la arquitectura, luego el ingeniero de casos de uso se encarga de analizar un caso de uso y por último el ingeniero de componentes se encarga de analizar una clase y un paquete, a continuación cada una de las actividades:

- **Análisis de la arquitectura**

Tiene como propósito esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes del análisis, clases del análisis evidente, y requisitos especiales comunes.

- **Analizar un caso de uso**

Analizar un caso de uso para:

  - Identificar las clases del análisis cuyos objetos son necesarios para el flujo de sucesos del caso de uso.
  - Distribuir el comportamiento del caso de uso entre los objetos del análisis que interactúan.

- Capturar los requisitos especiales sobre la realización del caso de uso.

- **Analizar una clase**

Los objetivos de analizar una clase son:

- Identificar y mantener las responsabilidades de una clase del análisis, basadas en su papel en las realizaciones de casos de uso.
- Identificar y mantener los atributos y relaciones de la clase del análisis.
- Capturar requisitos especiales sobre la realización de la clase del análisis.

- **Analizar un paquete**

Los objetivos de analizar un paquete son:

- Garantizar que el paquete del análisis sea independiente de los otos paquetes.
- Garantizar que el paquete del análisis cumpla con realizar algunas clases del caso de uso.
- Describir las dependencias de forma que pueda estimarse el efecto de los cambios futuros.

1.2.1.2.2.4 Diseño

En el diseño se modela el sistema y se define su forma para que soporte todos los requisitos. Los propósitos del diseño son los siguientes:

- Adquirir una compresión profunda de los aspectos relacionados con los requisitos no funcionales y las restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, etc.
- Crear una entrada apropiada y un punto de partida para actividades de implementación.
- Capturar las interfaces entre los subsistemas antes del ciclo de vida del software.
- Crear una abstracción sin costuras de la implementación del sistema.

Artefactos

- **Modelo de Diseño**

  El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

- **Clase del Diseño**

  Una clase de diseño es una abstracción sin costuras de una clase en el siguiente sentido:

  - El lenguaje utilizado para especificar una clase de diseño es lo mismo que el lenguaje de programación.
  - La visibilidad de los atributos y operaciones de una clase de diseño se especifica frecuentemente.
  - Las relaciones de aquellas clases del diseño implicadas con otras clases, a menudo tienen un significado directo cuando la clase es implementada.
  - Los métodos de la clase de diseño tienen correspondencia directa con el correspondiente método en la implementación.

- **Realización de caso de uso – diseño**

  Una realización de caso de uso – diseño es una colaboración en el modelo de diseño que describe como realizar un caso de uso específico y como se ejecuta, en términos de clases de diseño y sus objetos.

- **Subsistema del diseño**

  Los subsistemas de diseño son una forma de organizar los artefactos del modelo de diseño en piezas más manejables. Un subsistema debería ser cohesivo; es decir, sus contenidos deberían encontrarse fuertemente asociados.

- **Interfaz**

  Las interfaces se utilizan para especificar las operaciones que proporcionan las clases y los subsistemas del diseño. Las interfaces constituyen una forma de separar la especificación d la funcionalidad en términos de operaciones de sus implementaciones en términos de métodos.

- **Descripción de la arquitectura (Vista del modelo de Diseño)**

  La descripción de la arquitectura contiene una vista de la arquitectura de diseño. Para la arquitectura son relevantes los siguientes artefactos del modelo de diseño:

  - La descomposición del modelo de diseño en subsistemas, sus interfaces, y la dependencia entre ellos.
  - Clases del diseño fundamentales.
  - Realizaciones de caso de uso – diseño que describan alguna funcionalidad importante y crítica que debe desarrollarse pronto dentro del ciclo de vida del software.

- **Modelo de despliegue**

  El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

- **Descripción de la arquitectura (Vista del modelo de despliegue)**

  La descripción de la arquitectura contiene una vista de la arquitectura del modelo de despliegue, que muestra sus artefactos relevantes para la arquitectura.

Trabajadores

- **Arquitecto**

El arquitecto es responsable de la integridad de los modelos de diseño y de despliegue, garantizando que los modelos son correctos, consistentes y legibles en su totalidad.

- **Ingeniero de casos de uso**

  El ingeniero de casos de uso es el responsable de la integridad de una o más realizaciones de casos de uso – diseño, y debe garantizar que cumplen los requisitos que se esperan de ellos.
- **Ingeniero de componentes**

  El ingeniero de componentes define y mantiene las operaciones, métodos, atributos, relaciones y requisitos de implementación de una o más clase de diseño, garantizando que cada clase del diseño cumple con los requisitos que se espera de ella según las realizaciones de casos de uso en las que participa.

Flujo de trabajo

Los arquitectos inician la creación de los modelos de diseño y de despliegue. Después, los ingenieros de caso de uso realizan cada caso de uso en términos de clases y/o subsistemas del diseño participantes y sus interfaces.

- **Diseño de la arquitectura**

  El objetivo del diseño de la arquitectura es esbozar los modelos de diseño y despliegue y su arquitectura mediante la identificación de los siguientes elementos:
  - Nodos y sus configuraciones de red.
  - Subsistemas y sus interfaces.
  - Clases del diseño.
  - Mecanismos de diseño genérico.

- **Diseñar un caso de uso**

  Los objetivos del diseño de un caso de uso son:
  - Identificar las clases del diseño y/o los subsistemas cuyas instancias son necesarias para llevar a cabo el flujo de sucesos del caso de uso.

- Distribuir el comportamiento del caso de uso entre los objetos del diseño que interactúan y/o entre los subsistemas participantes.
- Definir los requisitos sobre las operaciones de las clases de diseño.
- Capturar los requisitos de implementación del caso de uso.

- **Diseñar una clase**

  El propósito de diseñar una clase es crear una clase del diseño que cumpla su papel en las realizaciones de los casos de uso y los requisitos no funcionales que se aplican a estos.

- **Diseñar un subsistema**

  Los objetivos del diseño de un subsistema son:
  - Garantizar que el subsistema es tan independiente como sea posible.
  - Garantiza que el subsistema proporciona las interfaces correctas.
  - Garantizar que el subsistema cumple su propósito de ofrecer una realización correcta de las operaciones.

1.2.1.2.2.5 Implementación

Los objetivos de la implementación son:

- Definir el código de organización, en términos de implementación de subsistemas
- Implementar clases y objetos en términos de componentes
- Probar el desarrollo de los componentes como unidades
- Integrar los resultados producidos de la implementaciones individuales

RUP describe como reutilizar los componentes existentes o implementar nuevos componentes con responsabilidades bien definidas.

Artefactos

- **Modelo de implementación**

  El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc.

- **Componente**

  Un componente es un empaquetamiento físico de los modelos de un modelo, como son las clases en el modelo de diseño.

- **Subsistema de la implementación**

  Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Un subsistema puede estar formado por componentes, interfaces y otros subsistemas.

- **Interfaz**

  Una interfaz sirve para especificar las operaciones implementadas por componentes y subsistemas de implementación.

- **Descripción de la arquitectura (Vista del modelo de implementación)**

  Vista de la arquitectura de un sistema, abarcando los componentes usados para el ensamblado y lanzamiento del sistema físico; vista que aborda la gestión de la configuración de las versiones del sistema, constituida por componentes independientes que pueden ser ensamblados de varias formas para producir un sistema ejecutable.

- **Plan de integración de constructores**

  Es importante construir el software incrementalmente en pasos manejables, de forma que cada paso dé lugar a pequeños problemas de integración o prueba.

Trabajadores

- **Arquitecto**

  El arquitecto es responsable de la integridad del modelo de implementación y asegura que el modelo como un todo es correcto, completo y legible.

- **Ingeniero de componentes**

  El ingeniero de componentes define y mantiene el código fuente de un o varios componentes, garantizado que cada componente implementa la funcionalidad correcta.

- **Integrador de sistemas**

  El integrador de sistemas es responsable de planificar la secuencia de construcciones necesarias en cada iteración y la integración de cada construcción cuando sus partes han sido implementadas.

Flujo de trabajo

- **Implementación de la arquitectura**

  El propósito de la de implementación de la arquitectura es esbozar el modelo de implementación y su arquitectura mediante:

  - La identificación de componentes.

  La asignación de componentes a los nodos en las configuraciones de red.

- **Integrar el sistema**

  Los objetivos de integración del sistema son:

  - Crear un plan de integración de construcciones que describa las construcciones necesarias en una iteración y los requisitos de cada construcción.
  - Integrar la construcción antes de que sea sometida a pruebas de integración.

- **Implementar un subsistema**

  El propósito de implementar un subsistema es el de asegurar que un subsistema cumpla su papel en cada construcción, tal como se especifica en el plan de integración de la construcción.

- **Implementar una clase**

  El propósito de la implementación de una clase es implementar una clase de diseño en un componente fichero.

- **Realizar prueba de unidad**

  El propósito de realizar una prueba de unidad es probar los componentes implementados como unidades individuales.

1.2.1.2.2.6 Pruebas

En la prueba verificamos el resultado de la implementación probando cada construcción. El resultado principal es el modelo de prueba, el cual describe cómo ha sido probado el sistema. El modelo de prueba incluye:

- Casos de prueba, que especifican qué probar en el sistema.
- Procedimientos de prueba, que especifican cómo realizar los casos de prueba.
- Componentes de prueba, que automatizan los procedimientos de prueba.

Artefactos

- **Modelo de pruebas**
  El modelo de pruebas describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema.
- **Caso de prueba**
  Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las cuales ha de aprobarse.
- **Procedimiento de prueba**
  Un procedimiento de prueba especifica cómo realizar uno o varios casos de prueba o partes de éstos.
- **Componente de prueba**
  Un componente automatiza uno o varios procedimientos de prueba o partes de ellos.
- **Plan de prueba**
  El plan de pruebas describe las estrategias, recursos y planificación de la prueba.
- **Defecto**
  Un defecto es un anomalía del sistema, como por ejemplo un síntoma de un fallo software o un problema descubierto en una revisión.

- **Evaluación de prueba**

  Una evaluación de prueba es una evaluación de los resultados de los esfuerzos de prueba, tales como la cobertura del caso de prueba, cobertura de código y el estado de los defectos.

Trabajadores

- **Diseñador de pruebas.**

  Un diseñador de pruebas es responsable de la integridad del modelo de pruebas, asegurando que el modelo cumple son el propósito.
- **Ingeniero de componentes**

  El ingeniero de componentes es responsable de los componentes de prueba que automatizan algunos de los procedimientos de prueba.
- **Ingeniero de pruebas de integración**

  El ingeniero de pruebas de integración es responsable de realizar las pruebas de integración que se necesitan para cada construcción producida en el flujo de trabajo de la implementación.
- **Ingeniero de pruebas del sistema**

  Un ingeniero de pruebas del sistema es responsable de realizar las pruebas necesarias sobre una construcción que muestra el resultado de una iteración completa.

Flujo de trabajo

- **Planificar prueba**

  El propósito es planificar los esfuerzos de prueba en una iteración llevando a cabo las siguientes tareas es:
  - Describiendo una estrategia de prueba.
  - Estimando los requisitos para el esfuerzo de la prueba.
  - Planificando el esfuerzo de la prueba.
- **Diseñar prueba**

  Los propósitos de diseñar pruebas son:
  - Identificar y describir los casos de prueba para cada construcción.

  - Identificar y estructurar los procedimientos de la prueba especificando cómo realizar los casos de prueba.

- **Implementar prueba**

  El propósito de la implementación de pruebas es automatizar los procedimientos de prueba creando componentes de prueba si esto posible, pues no todos los procedimientos de prueba pueden ser automatizados.

- **Realizar pruebas de integración**

  En esta actividad se realizan las pruebas de integración necesarias para cada una de las construcciones creadas en una iteración. Las pruebas de integración se llevan a cabo en los siguientes pasos:

  - Realizar las prueba de integración relevante a la construcción.
  - Comparar los resultados de las pruebas con los resultados esperados.
  - Informar de los defectos a los ingenieros de componentes.
  - Informar de los defectos a los diseñadores de pruebas.

- **Realizar pruebas de sistema**

  El propósito es el realizar las pruebas de sistema necesarias en cada iteración y el recopilar los datos de las pruebas.

- **Evaluar prueba**

  El propósito es el de evaluar los esfuerzos de prueba en una iteración.

1.2.2 Combinación de PSP y RUP.

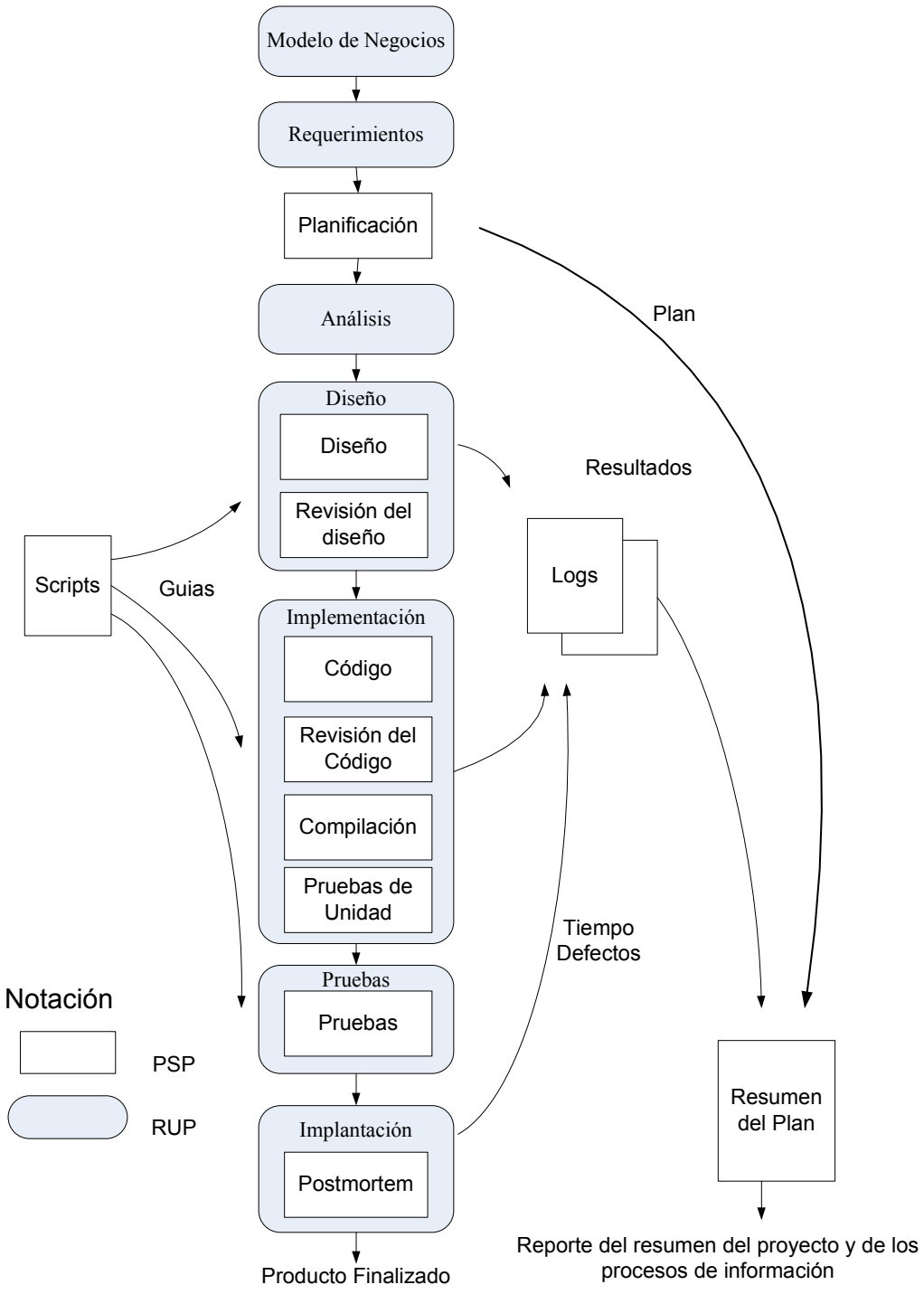


Figura 1.10 combinación de RUP y PSP

Para la realización de este trabajo se considero los flujos fundamentales de trabajo del RUP sobre los cuales se colocaron las mejores prácticas del PSP los cuales se describen a continuación.

1.2.2.1 Modelo del Negocio

El modelo de negocio es una técnica para comprender los procesos de negocios de la organización.

1.2.2.2 Requisitos

La captura de requisitos o requerimientos sirve para guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una buena descripción de los requisitos del sistema, para que se pueda llegar a un acuerdo entre el cliente y el desarrollador sobre qué debe y qué no debe hacer el sistema.

El PSP para este flujo de trabajo recomienda:

- Producir un informe de los requerimientos del programa.
- Asegurar que el informe escrito de requerimientos sea claro y sin ambigüedad.
- Resolver cualquier pregunta.

1.2.2.3 Planificación

El RUP en su estructura no tiene este ítem o flujo de trabajo, por lo que el PSP aporta con las siguientes prácticas:

- Diseño conceptual del sistema.
- Estimación de Tamaño.
- Estrategia del ciclo de desarrollo.
- Estimación de Recursos.
- Planificación de tareas y Horarios.
- Estimación de Defectos.

1.2.2.4 Análisis

Este flujo de trabajo se encarga de analizar los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. Además se va a considerar la planificación del PSP. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema completo.

1.2.2.5 Diseño

En el diseño se modela el sistema y se define su forma para que soporte todos los requisitos. El PSP recomienda en primer lugar obtener la siguiente información:

- Especificaciones externas.
- Modulo de Diseño.
- Prototipos.
- Estrategia de desarrollo.
- Log de rastreo de problemas.
- El diseño global del programa, incluidas las plantillas de especificaciones de control principal y de rutina.

Luego el PSP recomienda una la revisión del diseño que contiene la siguiente información:

- Alcance del diseño.
- Verificación lógica.
- Verificación de reuso.
- Verificación de la estrategia de Desarrollo.
- Reparar Defectos.

1.2.2.6 Implementación

Los objetivos de la implementación son:

- Definir el código de organización, en términos de implementación de subsistemas
- Implementar clases y objetos en términos de componentes
- Probar el desarrollo de los componentes como unidades
- Integrar los resultados producidos de la implementaciones individuales

Por su parte el PSP requiere de la siguiente información:

- Modulo de diseño.
- Revisión del diseño.
- Codificación.
- Revisión de la Codificación.
- Compilación.
- Reasegurar y reciclar.

1.2.2.7 Pruebas

En la prueba verificamos el resultado de la implementación probando cada construcción. El resultado principal es el modelo de prueba, el cual describe cómo ha sido probado el sistema.

Por su parte el PSP requiere de la siguiente información:

- Probar los módulos hasta que la prueba se ejecute sin errores.
- Reparar todos los defectos establecidos.
- Registrar los defectos en el Log de registro de defectos.
- Registrar el tiempo en el Log de registro del tiempo.
- Completar la plantilla de reporte de pruebas en las pruebas dirigidas y en los resultados obtenidos.

1.2.2.8 Implantación

El propósito principal de este flujo de trabajo es producir un releases del producto y entregar el software al usuario final, esto incluye las siguientes actividades:

- Producir un releases externo del software.
- Empaquetar el software.
- Distribuir el software.
- Instalación del software.
- Proveer de ayuda y asistencia a los usuarios.

En este flujo de trabajo el PSP propone:

- Defectos inyectados.
- Defectos removidos.
- Tamaño.
- Tiempo.
- Completar la forma del plan del proyecto con el tiempo actual, defectos, tamaño de los datos actualizados.
- Completar la forma del resumen del ciclo con datos actuales del ciclo.

CAPITULO II

2 CASO DE ESTUDIO

2.1 Caso de Estudio: Sistema de ayuda para niños con parálisis cerebral

2.1.1 El niño con deficiencia motora

Concepto.- Niño deficiente motórico es todo aquel que presenta de manera transitoria o permanente alguna alteración en su aparato motor, debido un deficiente funcionamiento en el sistema óseo – articular, muscular y nervioso, y que en grados variables limita algunas de las actividades que pueden realizar el resto de niños de su misma edad.

2.1.2 Clasificación de los tipos deficiencia motora

Las deficiencias motoras comprenden una extrema diversidad de situaciones. Así, se la puede clasificar atendiendo a:

2.1.2.1 Fecha de aparición

Desde el nacimiento

- Malformaciones congénitas.
- Espina bífida.
- Luxación congénita de caderas.
- Artrogriposis.

Después del nacimiento

- Parálisis cerebral.
- Miopatía de Duchenne.

En la adolescencia

- Miopatías facio-escápulo-humerales (Landouzy-Dejerine).

A lo largo de toda la vida

- Traumatismos craneoencefálicos.
- Traumatismos vertebrales.
- Tumores.

2.1.2.2 Etiopatología

Transmisión genética

- Madre portadora: miopatía de Duchenne de Boulogne.
- Uno de los progenitores tiene el mismo hándicap: miopatía de Landouzy-Dejerine.
- Los dos padres son portadores recesivos de una misma aberración genética: enfermedad de Werdning-Hoffmann.

Infecciones microbianas

- Tuberculosis ósea.
- Poliomielitis anterior aguda.

Accidentes

- En el embarazo o parto: + parálisis cerebral.
- A lo largo de la vida:
    - Coma por traumatismo craneal.
    - Paraplejia post-traumática.
    - Amputaciones.
    - Quemaduras.

Origen desconocido

- Espina bífida.
- Escoliosis ideopática.
- Tumores.

2.1.2.3 Localización topográfica

Parálisis

- Monoplejia: Parálisis de un solo miembro, ya sea brazo o pierna.
- Hemiplejía: Parálisis de un lado del cuerpo (derecho o izquierdo).
- Paraplejia: Parálisis de las dos piernas.
- Diaplejía: Parálisis que afecta a partes iguales a cada lado del cuerpo (parálisis bilateral) (*)
- Tetraplejia: Parálisis de los cuatro miembros.

Paresia

- Monoparesia: Parálisis ligera o incompleta de un solo miembro.
- Hemiparesia: Parálisis ligera o incompleta de un lado del cuerpo (derecho o izquierdo).
- Parapesia: Parálisis ligera o incompleta de las dos piernas.
- Tetraparesia: Parálisis ligera o incompleta de los cuatro miembros.

2.1.2.4 Origen de la deficiencia

Origen cerebral

- Parálisis cerebral.
- Traumatismos craneoencefálicos.
- Tumores.

Origen espinal

- Poliomielitis anterior aguda.
- Espina bífida.
- Lesiones medulares degenerativas:
    - Enfermedad de Werding-Hoffmann.
    - Síndrome de Wohlfart-Kugelberg.
    - Enfermedad de Charcot-Marie-Tcoth.
    - Ataxia de Friedreich.
- Traumatismos medulares.

Origen muscular

- Miopatías:
  - Distrofia muscular progresiva de Duchenne de Boulogne.
  - Distrofia escapular de Landouzy-Dejerine.

Origen óseo-articular

- Malformaciones congénitas:
  - Amputaciones congénitas.
  - Luxación congénita de caderas.
  - Artrogriposis.
- Distróficas:
  - Condodistrofia.
  - Osteogénesis imperfecta.
- Microbianas:
  - Osteomielitis aguda.
  - Tuberculosis óseo-articular.
- Reumatismos de la infancia:
  - Reumatismo articular agudo.
  - Reumatismo crónico.
- Lesiones osteoarticulares por desviaciones del raquis:
  - Cifosis.
  - Lordosis.
  - Escoliosis.

2.1.3 Parálisis cerebral

Definición.- "Desorden permanente y no inmutable de la postura y el movimiento debido a una lesión del cerebro antes que su desarrollo y crecimiento sean completes" (International Society of Cerebral Palsy-Edimburgo, 1969).

Etiología, es decir algunas características de la parálisis cerebral:

Factores prenatales:

- Retardo en el crecimiento intrauterino.
- Infecciones virales de la madre: rubéola, toxoplasmosis.
- Medicación inadecuada durante el embarazo.
- Prematuridad.

Factores perinatales (Son los más frecuentes):

Dificultades en el parto (distocias).

- Problemas respiratorios (anoxias).
- Ictericia.
- Traumatismos: parto con ventosas, fórceps.

Factores postnatales (Se consideran hasta los tres años)

- Deshidratación aguda.
- Anomalías metabólicas.
- Traumatismos.
- Alimentación deficiente.

2.1.3.1 Tipos de parálisis cerebral.

A continuación se describe los tipos de parálisis cerebral.

2.1.3.1.1 Espasticidad

Se cree que es consecuencia de una lesión en el sistema piramidal (rige los movimientos voluntarios). Incremento del tono muscular en el momento de realizar movimientos voluntarios. Carece de juego muscular flexión/extensión, dando lugar a una acción refleja de hipertonía permanente, provocando contractura y espasmos en un grupo muscular e hipotonía del grupo muscular antagonista. Si uno de estos niños intenta la flexión de cualquier parte del cuerpo (columna, brazos o piernas) no podrá realizarla sin flexionar la totalidad del cuerpo; si pretende la extensión de cualquier parte del cuerpo, este intento

hace que todo el cuerpo se extienda. Este estado se agrava por los estiramientos pasivos, por los actos voluntarios y en estados emocionales.

El lenguaje tiende a ser explosivo, interrumpido por largas pausas. En casos muy graves, puede quedar bloqueado al no poder mover sus mecanismos de fonación.

2.1.3.1.2 Atetosis

Parece tratarse de una lesión en el sistema extrapiramidal, por lo general en los ganglios básales del tronco cerebral (controla los movimientos involuntarios, el tono muscular y los estados de vigilia).

Se caracteriza por contracciones involuntarias de las extremidades dístales (miembros y cara). Estos movimientos pueden mantenerse en reposo o, por el contrario, dispararse completamente durante la relajación o el sueno.

El tono muscular es fluctuante y va desde la hipertonía paroxística hasta la hipotonía (desde la tensión muscular extrema hasta la laxicitud extrema). Alterna el movimiento abrupto y mal orientado con las posturas rígidas del espástico; pero, a diferencia de este, solo mantiene tales posturas fugazmente.
El lenguaje es muy variable. Los casos leves pueden presentar pequeños fallos en la articulación, mientras que los graves no hablan en absoluto. El atetósico produce, en general, un habla incordinada y carente de ritmo.

2.1.3.1.3 Ataxia

Se debe a una lesión en el cerebelo (regulador de la adaptación postural y de la coordinación de los movimientos voluntarios).

Se manifiesta por problemas de equilibrio en la marcha y por una mala coordinación espacial y temporal de los gestos. En estado puro es muy rara. Muchas veces se clasifican como atáxicos los que son una mezcla de atetosis y ataxia.

2.1.3.1.4 Estados mixtos

Frecuentemente nos encontramos con un problema motor en el que se asocian estos tres tipos.

2.1.4 Medida de la Inteligencia

Esta escala fue publicada en 1937, luego de 10 años de investigación y tipificación. Sus actores son Terman y Merrill los cuales diseñaron el Test Forma L. Durante los años transcurridos desde la primera publicación, al trabajo de la segunda revisión de 1.937 se había recopilado y observado el, funcionamiento de varios tests que miden la inteligencia y es así, que estos dos autores publican su obra titulada; "MEASURING INTELLIGENGE" (medida de la Inteligencia) en el año de 1944.

Esta Escala y su revisión de 1.937, guardan las mismas características que distinguen los tests Binet, pues utilizan los mismos supuestos y los mismos principios. Así por ejemplo conservan el supuesto de que la inteligencia general es un rasgo que se desarrolla con la edad. De igual manera a través de esta Escala se mide la inteligencia general y no una capacidad especifica.

Las pruebas que demostraron ser útiles en 1916 siguieron demostrando su utilidad en 1.937. Siguen manteniendo la misma actividad mental que Binet había considerado útiles para distinguir las diferencias de inteligencia.

El material a emplearse fue escogido y bien seleccionado para que ofrecieran situaciones modelos y pueda de esta manera observase una muestra de 3.184 casos.

Fiabilidad y Validez

Se calculó coeficientes de fiabilidad para cada edad separadamente puesto que la fiabilidad está en función directa de la edad, de los cuales podemos indicar las siguientes:

Desde 0.83 para los C.I de 140 – 149

Desde 0.91 para los C.I de 60 - 69
Desde 0.91 para los C.I de 140 - 149
Desde 0.97 para los C.I de 60 - 69
Desde 0.95 para los C.I de 140 - 149
Desde 0.98 para los C.I de 60- 69

La variabilidad de las valoraciones de edad mental aumenta a la par de la edad cronológica puesto que el C.I. debe tener un significado constante de un nivel de edad a otra.

Por lo mismo, significa que la valoración de la edad mental en un sujeto fuera proporcional no a la variabilidad de un grupo de edad cronológica, sino a la variabilidad de su grupo de edad mental. Dicho otra en tras palabras el sujeto de la ciudad como al de una tribu de la amazonia el valor del C.I. estarla o Pondría validez si su aplicación se lo hace en cada tramo social y no en la variabilidad existente entre la edad cronológica del niño de la ciudad y/o la del niño de una tribu de la amazonia.

Se deba indicar que si un test no tiene la medida apropiada para capacidades diferenciales de inteligencia no quiere decir que no ofrezca oportunidad a la observación, es decir, un psicólogo experimentado así no lo haga apreciaciones cuantitativas sobre un test, lo podrá servir para explorar aspectos cualitativos, de la actuación del sujeto examinado podrá descubrir métodos todos de trabajo, forma de abordar los problemas, manera do dar solución a los mismos y otras observaciones clínicas de importancia

Sujetos de Aplicación

Desde 2 años y medio hasta Adultos superiores III

Duración del Examen

Las tareas que sugiere el test están bien distribuidas, por eso su aplicación en Condiciones normales no produce fatiga, puesto que la fatiga la única señal

para detener la prueba. Hay ocasiones en que se prolonga hasta una hora y media por eso su aplicación en niños muy pequeños, sin que se advierta pérdida del interés o alguna señal de cansancio. En definitiva no es posible calcular el tiempo que loa sujetos van a emplear en el test.

También es necesario considerar que inclusive el examinador -experimentado necesita menos tiempo que las personas iniciantes e inexpertos.

¿Dónde empezar el Examen?

El examen debe iniciarse con la actitud que probablemente el sujeto examinado responda con acierto sin hacer ningún esfuerzo, puesto que si son difíciles desde el comienzo la persona examinada puede inhibirse y por lo mismo negarse a continuar. Debe tenerse en cuenta la edad cronológica, la conducta general o cualquier tipo de información que se haya obtenido previamente Ej. Que no sabe leer ni escribir.

Si no contesta con aciertos en todos los ítems de una edad determinada como base del examen será necesario progresar a edades anteriores hasta cuando cumpla las respuestas todas con acierto (**edad base**).

Ahora es necesario indicar cuando se debe dar por terminado la prueba y es cuando ha dejado de responder con aciertos todos los ítems de una edad menos uno.

Cálculo de la Edad Mental

Se procede de la siguiente manera:

- Se anota la edad base en años y meses.
- Las respuestas registradas como aciertos dentro de una edad determinada tienen el siguiente equivalente: 2 años y medio hasta 4 años dos semestres cada acierto equivale a un (1) mes de edad mental.
- Desde los 5 años cada acierto equivale 2 meses de edad mental
- Se suman los meses de edad mental y se saca los años de edad mental

y más la edad base esa cifra constituye la EDAD MENTAL el C.I se calcula de acuerdo a la formula:

$$CI = \frac{EM \times 100}{EC}$$

EM: Edad Mental.

EC: Edad Cronológica.

CI: Coeficiente Intelectual.

2.2 Desarrollo del Sistema usando RUP y PSP.

2.2.1 Misión del Sistema

El presente sistema tiene como misión determinar el coeficiente intelectual de niños con parálisis cerebral. Para lo cual el sistema debe de ser capaz de manejar la siguiente información:

- Registro de los datos personales de los niños.
- Resultados del test que es evaluado el niño.
- Historial de los test que el niño ha realizado en el sistema.
- Administración de los usuarios según los perfiles.
- Información referente al test.
- Estadísticas de las evaluaciones.

2.2.2 Justificación

Se ha observado que en el mercado no existe un software que realice una evaluación semiautomática del coeficiente intelectual de los niños con parálisis cerebral en si de cualquier tipo de niño, al momento lo que se tiene son test escritos que son llevados acabo de manera manual.

Este sistema apunta a una de las áreas desatendidas de nuestra sociedad con son los niños con necesidades especiales. En vista de que muchas veces los sistemas se inclinan a áreas financieras, científicas, resulta interesante tomar una área como la social, específicamente el campo de los niños especiales.

2.2.3 Alcance del Proyecto:

- El sistema tendrá la capacidad de funcionar como un sistema distribuido.
- Puede ser usados por cualquier tipo de usuario acorde a las restricciones de los perfiles.
- Almacenar la información de cada uno de los niños.
- Llevar un registro de las evaluaciones de cada uno de los niños.
- Guiar y ayudar al instructor para que evalué al niño.
- Tener estadísticas de los test, para reportes de cada uno de los niños.
- Tener información clasificada y ordenada de manera rápida y segura, acerca de los niños.

2.2.4 Requisitos

La toma de requisitos esta centrada en:

- Enumeración de las características que debe incorporar el sistema.
- Comprender el contexto del sistema
- Requisitos funcionales del sistema.

2.2.4.1 Lista de Características

- Creación de usuarios,

Definición.- Proceso que permite generar los usuarios, es decir los instructores que van a utilizar el sistema.

Valores

Estado: Propuesto.

Tiempo estimado:

Prioridad: Importante.

- Registro del niño a ser evaluado,

Definición.- Proceso que permite guardar los datos del niño a ser evaluados por el instructor.

Valores

Estado: Propuesto.

Tiempo estimado:

Prioridad: Importante.

- Emisión de Test,

Definición.- Proceso que permite generar el resultado del test para determinar la edad mental del niño evaluado.

Valores

Estado: Propuesto.

Tiempo estimado:

Prioridad: Critica.

- Emisión de reportes de Test evaluados,

Definición.- Se presentarán los siguientes reportes de los Test evaluados:

- Detalle de los test por niño.
- Detalle de los test por instructor.
- Reporte de comparación entre test de un mismo niño en diferentes periodos.

Valores

Estado: Propuesto.

Tiempo estimado:

Prioridad: Importante.

2.2.4.2 Modelo de casos de uso del negocio

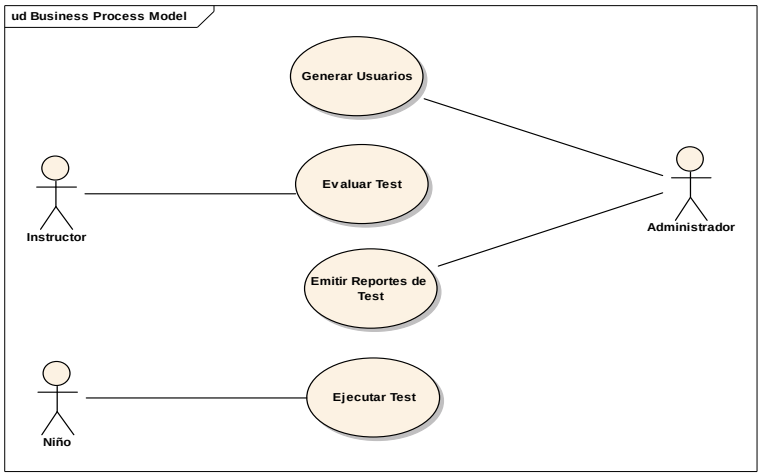


Figura 2.1 Modelo de Casos de Uso del Negocio

Actores

INSTRUCTOR	El instructor es el responsable de evaluar al niño con parálisis cerebral, seleccionando al niño a ser evaluado y ayudándolo durante todo el proceso de evaluación.
ADMINISTRADOR	El Administrador es la persona responsable de crear usuarios de acuerdo a sus perfiles, de registrar a los niños que van a ser evaluados con el sistema. También es responsable de revisar las estadísticas sobre los test de determinado niño.
NIÑO	Es la persona a ser evaluada, de la cual se determinara su coeficiente intelectual.

Caso de Uso Generar Usuarios
El proceso de crear usuarios lo realiza el administrador, el cual de acuerdo al perfil crea un nuevo usuario, registrando todos sus datos, actualiza sus datos o lo elimina del sistema.

Caso de Uso Evaluar Test
El Instructor luego de seleccionar el niño a ser evaluado genera el test para evaluarlo en un tiempo indeterminado, ya que el test no tiempo limite de tiempo.

Caso de Uso Emitir Reportes de Test
El administrador del sistema con los resultados del test obtiene datos estadísticos de los niños que fueron evaluados con el sistema.

Caso de Uso Ejecutar Test
El niño ejecuta el test, respondiendo una a una las preguntas del test, dependiendo de los parámetros de inicio del Test.

2.2.4.3 Casos de uso del Sistema

A continuación se detallan los casos de uso del sistema, que están representados en las siguientes figuras:

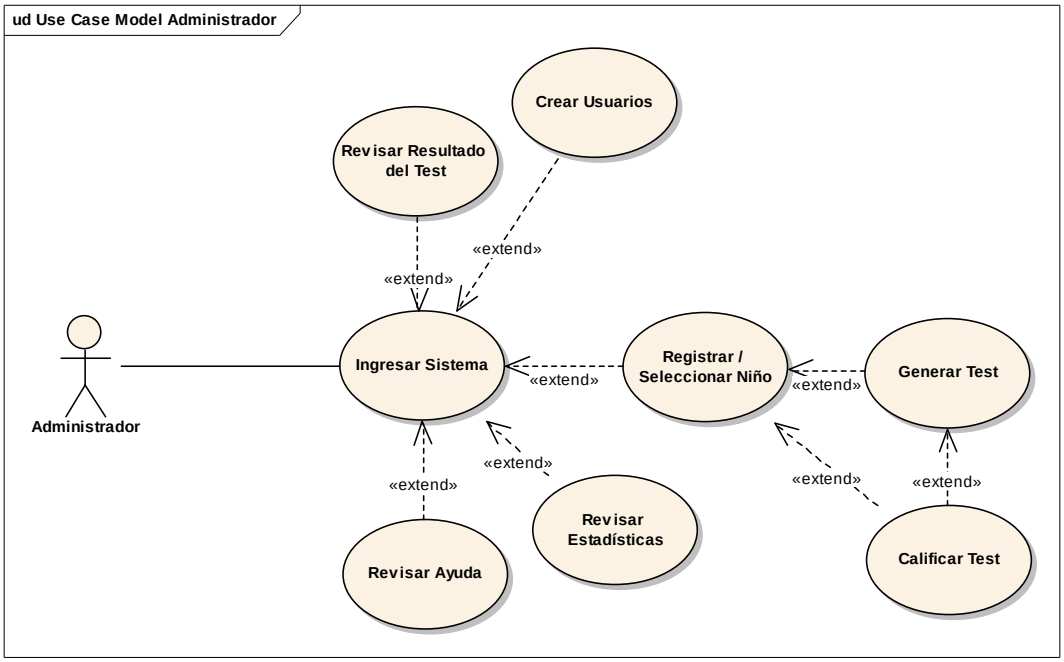


Figura 2.2 Modelo de Casos de Uso del Administrador

DESCRIPCIÓN DEL CASO DE USO INGRESAR SISTEMA	
Caso de Uso	Ingresar Sistema
Descripción	Realiza la validación con el sistema e ingresa al sistema dependiendo de su respectivo perfil.
Actores	Administrador e Instructor.
Precondiciones	Ninguna
Poscondiciones	Ninguna.
Pasos	1. Ingresar el nombre de usuario. 2. Ingresar la contraseña. 3. Verificar si el usuario existe. 4. Verificar si la contraseña es correcta. 5. Activar el perfil. 6. Bienvenida.
Variaciones	1. No hay
Observaciones	El sistema luego de tres intentos de acceso envía un aviso que se ponga

	en contacto con el administrador.

DESCRIPCIÓN DEL CASO DE USO CREAR USUARIOS	
Caso de Uso	Crear Usuarios
Descripción	Permite crear usuarios que en este caso son los instructores que realizan la evaluación del test
Actores	Administrador.
Precondiciones	Solicitud del usuario.
Poscondiciones	Ninguna.
Pasos	1. Solicitud del cliente. 2. Ingresar al sistema. 3. Llenar el formulario de administración de usuarios. 4. Preguntar si la información es correcta. 5. Guardar los datos del nuevo usuario.
Variaciones	1. En el paso 3, si el usuario existe en el sistema, verificar su información y actualizarla.
Observaciones	Este caso de uso no tiene ni incluyes ni extends.

DESCRIPCIÓN DEL CASO DE USO REVISAR ESTADÍSTICAS	
Caso de Uso	Revisar Estadísticas
Descripción	Permite al administrador generar reportes de los test evaluados a un determinado niño.
Actores	Administrador.
Precondiciones	Solicitud del director del instituto.
Poscondiciones	Ninguna.
Pasos	1. Solicitud del director del instituto. 2. Ingresar al sistema. 3. Seleccionar estadísticas. 4. Seleccionar el niño del cual se desea la información. 5. Guardar dicha consulta. 6. Imprimir la consulta.
Variaciones	1. En el paso 5, si no se desea guardar la consulta, proceder a

	imprimir.
Observaciones	Este caso de uso no tiene ni incluyes ni extends.

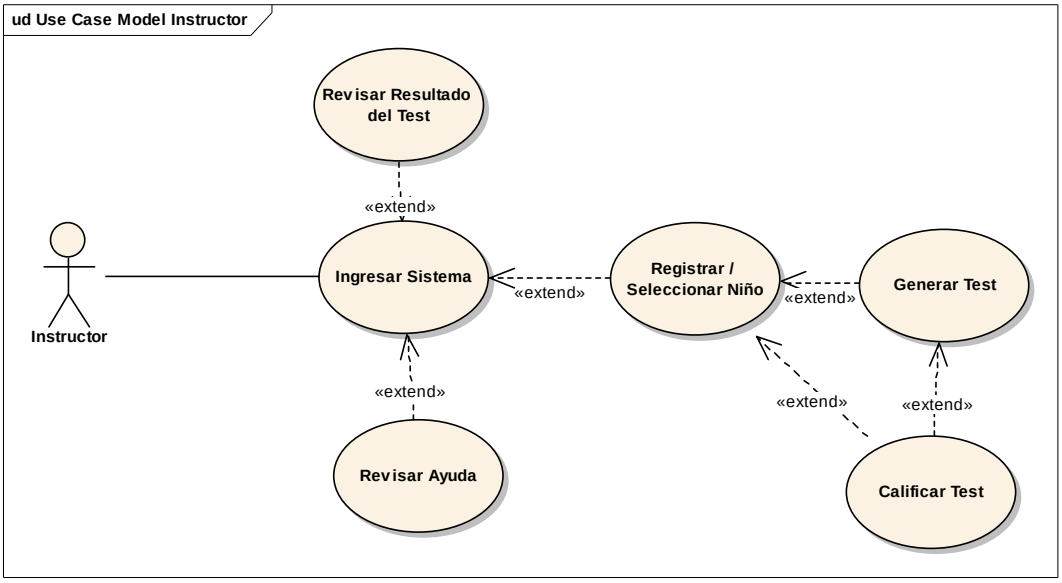


Figura 2.3 Modelo de Casos de Uso del Instructor

DESCRIPCIÓN DEL CASO DE USO REVISAR AYUDA	
Caso de Uso	Revisar Ayuda
Descripción	Realiza la acción de mostrar al actor Instructor la información pertinente a como evaluar al niño y todo el test escrito.
Actores	Administrador e Instructor.
Precondiciones	Ninguna
Poscondiciones	Ninguna.
Pasos	1. Ingresar al sistema. 2. Seleccionar ayuda. 3. Mostrar condiciones para iniciar y terminar evaluación. 4. Mostrar instrucciones del test.
Variaciones	1. No hay
Observaciones	El sistema luego de tres intentos de acceso envía un aviso que se ponga en contacto con el administrador.

DESCRIPCIÓN DEL CASO DE USO REVISAR RESULTADO DEL TEST	
Caso de Uso	Revisar Resultado del Test
Descripción	Una vez generado y calificado el test se procede a revisar el resultado del mismo.
Actores	Administrador e Instructor.
Precondiciones	Ninguna
Poscondiciones	Ninguna.
Pasos	1. Ingresar al sistema. 2. Seleccionar resultados. 3. Seleccionar el nombre del niño. 4. Generar lista de test. 5. Consultar resultados del Test. 6. Imprimir resultados.
Variaciones	1. En el paso 4 si existe mas de un test verificar que sea el test que deseamos saber los resultados.
Observaciones	El sistema no necesita guardar los datos en ese momento, ya que una vez finalizada la calificación todos los resultados están guardados en la base de datos.

DESCRIPCIÓN DEL CASO DE USO REGISTRAR / SELECCIONAR NIÑO	
Caso de Uso	Registrar / Seleccionar Niño
Descripción	Registra al niño en el sistema y si el niño ya s encuentra en el sistema solo lo selecciona.
Actores	Administrador e Instructor.
Precondiciones	Solicitud del padre de familia.
Poscondiciones	Ninguna.
Pasos	1. Ingresar al sistema. 2. Llenar el formulario con los datos personales del niño. 3. Verificar que los datos sean correctos. 4. Guardar los datos del niño. 5. Generar Test. 6. Calificar test.
Variaciones	1. Si el niño ya existe en el sistema, proceder a generar un nuevo test.
Observaciones	Una vez que el niño es registrado en el sistema, se puede generar el test o

	simplemente dejarlo para otra ocasión.

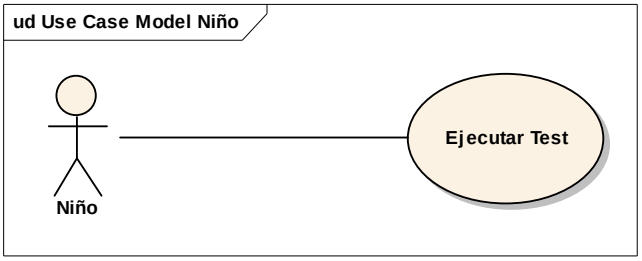


Figura 2.4 Modelo de Casos de Uso del Niño

DESCRIPCIÓN DEL CASO DE USO EJECUTAR TEST	
Caso de Uso	Ejecutar Test
Descripción	El actor niño contesta a cada una de las preguntas del test.
Actores	Niño
Precondiciones	Niño registrado en el sistema.
Poscondiciones	Niño evaluado.
Pasos	1. Niño registrado. 2. Generar el test por parte del instructor. 3. El test inicia cuando contesta un nivel completo. 4. Contestar a las preguntas del test. 5. El test termina cuando contesta máximo dos preguntas de un nivel.
Variaciones	1. Si el niño no contesta todo el nivel seleccionado debe regresar a un nivel anterior.
Observaciones	No hay límite de tiempo para que el niño realice el test.

2.2.4.4 Requisitos Adicionales

Para el sistema se debe tomar las siguientes restricciones de plataforma, hardware y software adicional.

Servidor:
Hardware:

- Pentium IV en adelante.
- 80 GB disco duro.

- 512 MB de ram
- Tarjeta de red

Software:

- Plataforma Windows. Sistema Operativo Windows XP, 2000 Server, 2003 Server.
- PHP 5.0.4
- Internet Explorer 5.0 o superiores.
- Base de Datos: MySQL 4.1.13-nt

Cliente:

Hardware:

- Pentium II en adelante.
- 40 GB disco duro.
- 128 MB de ram
- Tarjeta de red

Software:

- Plataforma Windows. Sistema Operativo Windows 2000 o superiores.
- Internet Explorer 5.0 en adelante para que se visualice de forma adecuada la aplicación.

Impresora instalada en la red, la cual puede ser a inyeccion o laser.

Software Adicional.

El computador donde se instale el SISTEMA debe de tener:

- xampp-win32-1.4.15, este software contiene al apache, mysql.

2.2.4.5 Representación de los requisitos como casos de uso

A continuación se detallaran específicamente dos casos de uso en los cuales radica la mayor importancia en el sistema:

- Registrar / Seleccionar Niño.
- Revisar Resultado del Test.

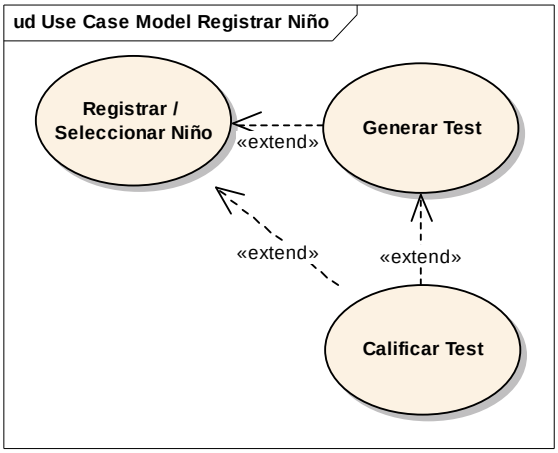


Figura 2.5 Modelo de Casos de Uso Registrar Niño

DESCRIPCIÓN DEL CASO DE USO GENERAR TEST	
Caso de Uso	Generar Test
Descripción	Genera las preguntas del test, de un niño seleccionado.
Actores	Administrador e Instructor.
Precondiciones	Niño registrado en el sistema.
Poscondiciones	Ninguna.
Pasos	1. Seleccionar un niño. 2. Crear test. 3. Generar preguntas del test.
Variaciones	1. No existen variaciones
Observaciones	Se puede generar el test y no evaluarlo.

DESCRIPCIÓN DEL CASO DE USO CALIFICAR TEST	
Caso de Uso	Calificar Test
Descripción	Califica cada una de las preguntas del test.
Actores	Administrador e Instructor.
Precondiciones	Contestar las preguntas del Test.
Poscondiciones	Ninguna.
Pasos	1. Seleccionar la pregunta. 2. El niño ejecuta el test, respondiendo a las preguntas. 3. El instructor califica la pregunta. 4. El sistema muestra el resultado.
Variaciones	1. Si el niño no ejecuta el test no hay calificación

Observaciones	La calificación es semiautomática, ya que el instructor en función de su observación calificara el test.

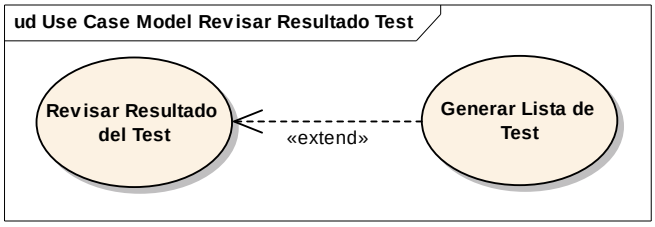


Figura 2.6 Modelo de Casos de Revisar Resultado del test

DESCRIPCIÓN DEL CASO DE USO GENERAR LISTA DE TEST	
Caso de Uso	Generar Lista de Test
Descripción	Genera la lista de test que el niño ha realizado utilizando el sistema.
Actores	Administrador e Instructor.
Precondiciones	Test calificado por el instructor.
Poscondiciones	Ninguna.
Pasos	1. El test debe estar calificado. 2. Seleccionar el niño evaluado. 3. El sistema Genera la lista de test evaluados al niño.
Variaciones	1. En el paso 3, en el caso de no haber una lista de test, el sistema no presenta datos.
Observaciones	Abra una lista de test en función de cuantas veces sea evaluado el niño.

El informe de requerimientos del sistema se realiza a través de:

- Misión del sistema.
- Alcance del Proyecto.
- Lista de características.

De esa forma se elimina cualquier ambigüedad en el informe, la cual es una práctica recomendada por el PSP.

2.2.5 Planificación

En el PSP se produce el diseño conceptual del sistema, el cual esta detallado en el modelo de casos de uso del negocio que utiliza el RUP.

Existen algunos métodos para determinar el tamaño del sistema, por el tipo de lenguaje de programación utilizado, se va utilizar el método Fuzzy-Logic[3] el cual en una de sus tablas indica un rango de tamaño de sistemas que va desde muy pequeño hasta muy grande, con los datos estimados que se tiene de 3000 LOC aproximadamente, se ha determinado que el sistema esta dentro del rango de pequeños sistemas. Al tener el sistema aproximadamente 3000 LOC, el PSP recomienda dividirlo en módulos para su desarrollo, los mismos que deben estar entre 100 y 250 nuevos y cambiados LOC.

Para el presente trabajo se ha dividido al sistema en ocho módulos. A continuación se muestra parte de la plantilla de resumen del plan del proyecto PSP, del modulo Resultados, el cual muestra la información sobre la estimación del tamaño, estimación el tiempo y estimación de defectos:

PSP3 Project Plan Summary - Program 8				
Student	1-Andrés Larco		Date	########
Program	Resultados		Program#	8
Instructor	Carlos Montenegro		Language	PHP
	Plan	**Actual**	**To Date**	**To Date%**
Program Size (LOC)				
Base(B)	**600**	**514**		
Deleted(D)	**10**	**5**		
Modified(M)	**10**	**5**		
Added(A)	40	16		
Reused(R)	**150**	**75**	75	
Total N&C (N)	**50**	21	21	
Total LOC(T)	780	**600**	600	
Total New Reused	**0**	**0**	0	
Total Object LOC(E)	**50**	**45**	45	

[3] Watts S. Humphrey, A Discipline for Software Engineering

UPI (70%)							
LPI (70%)							
Time in Phase (min.)							
Planning	0		25		25		6,3
Design	0		100		100		25,0
Design Review	0		25		25		6,3
Code	0		150		150		37,5
Code Review	0		25		25		6,3
Compile	0		30		30		7,5
Test	0		25		25		6,3
Postmortem	0		20		20		5,0
Total	**400**		400		400		100,0
UPI (70%)							
LPI (70%)							

Tabla 2.1 Resumen del Plan del Proyecto Parte 1

Continuación de la tabla anterior:

	Plan		**Actual**		**To Date**		**To Date%**
Defects Injected							
Planning	0,0		1		1		11,1
Design	0,0		1		1		11,1
Design Review	0,0		0		0		0,0
Code	0,0		3		3		33,3
Code Review	0,0		3		3		33,3
Compile	0,0		1		1		11,1
Test	0,0		0		0		0,0
Total Development	0,0		9		9		100,0
Defects Removed							
Planning	0,0		1		1		11,1
Design	0,0		1		1		11,1
Design Review	0,0		0		0		0,0
Code	0,0		3		3		33,3
Code Review	0,0		3		3		33,3
Compile	0,0		1		1		11,1
Test	0,0		0		0		0,0
Total Development	0,0		9		9		100,0
After Development			**10**		**10**		

Tabla 2.2 Resumen del Plan del Proyecto Parte 2

2.2.6 Análisis

2.2.6.1 Modelo de Análisis

Para desarrollar el modelo de análisis se utiliza los casos de uso que influyen directamente en la aplicación. Estos casos de uso son los identificados en la especificación de requerimientos. De estos se obtienen las siguientes relaciones de caso de uso- análisis:

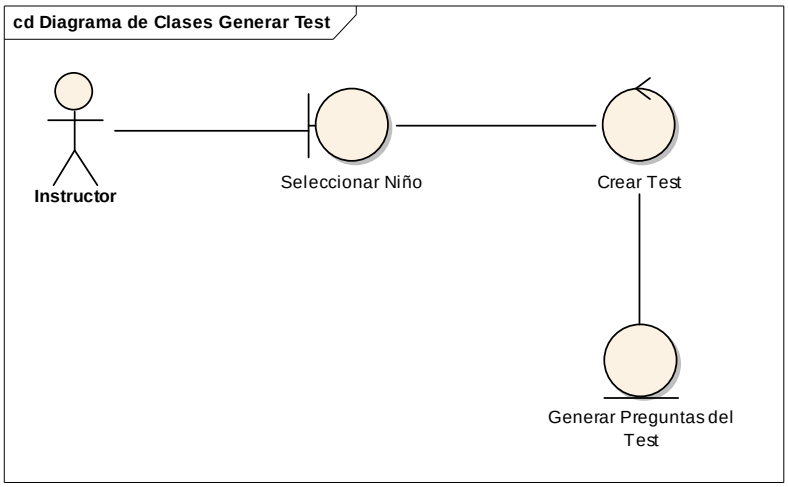


Figura 2.7 Diagrama de Clases de una realización del caso de uso Generar Test

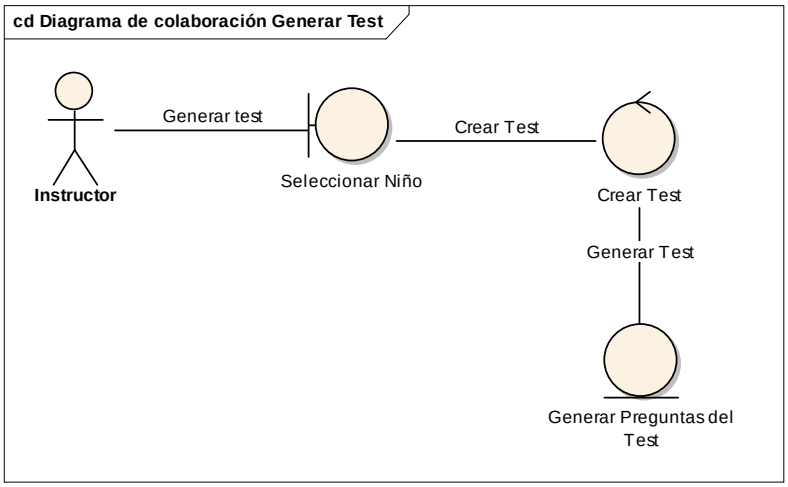


Figura 2.8 Diagrama de Colaboración de una realización del caso de uso Generar Test

FLUJO DE SUCESOS-DIAGRAMA DE COLABORACIÓN DE LA REALIZACIÓN DEL CASO DE USO GENERAR TEST
El instructor genera el test, a través de seleccionar el niño de una lista ya existente. Una vez seleccionado el niño se pide al objeto crear test que genere el test el objeto Generar Preguntas del Test.

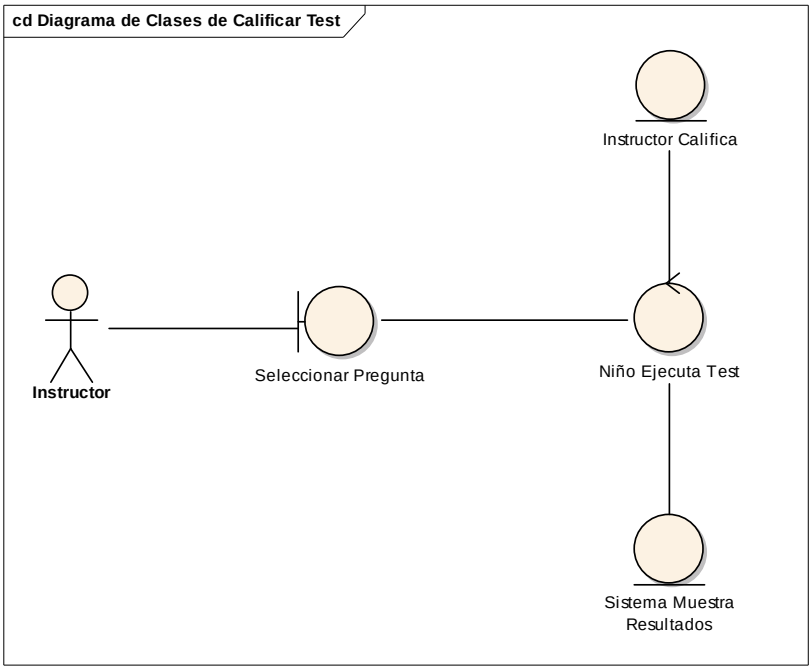


Figura 2.9 Diagrama de Clases de una realización del caso de uso Calificar Test

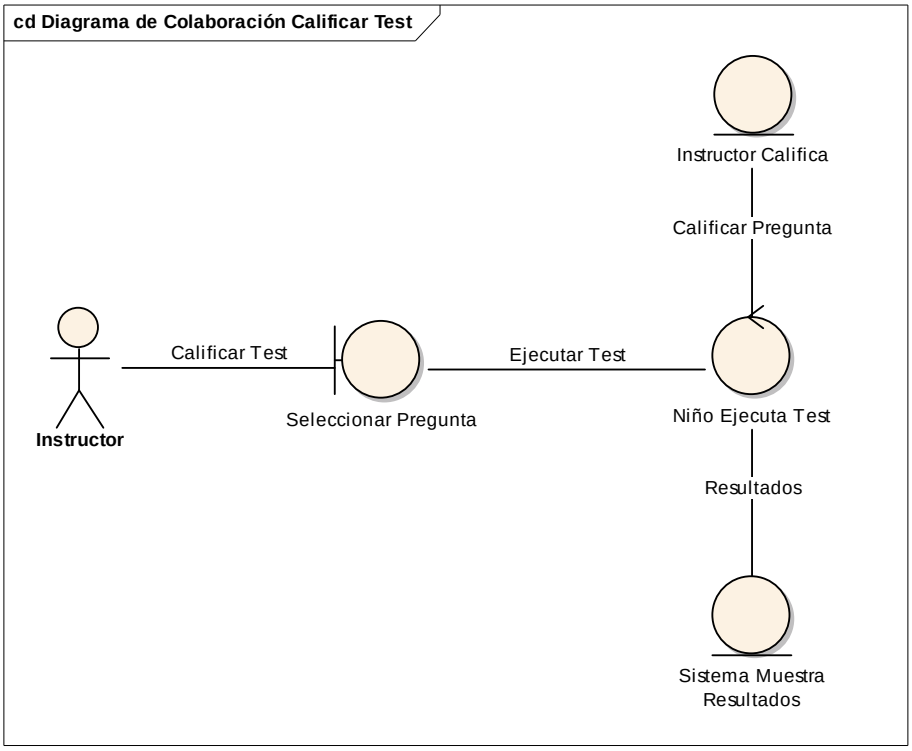


Figura 2.10 Diagrama de Colaboración de una realización del caso de uso Calificar Test

FLUJO DE SUCESOS-DIAGRAMA DE COLABORACIÓN DE LA REALIZACIÓN DEL CASO DE USO CALIFICAR TEST
El instructor califica el test, a través de seleccionar el objeto pregunta, el niño ejecuta el test, cada pregunta del test es calificada por el instructor. Y finalmente el sistema muestra el resultado.

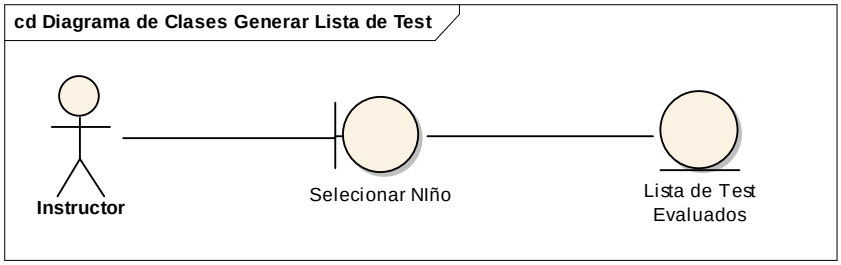


Figura 2.11 Diagrama de Clases de una realización del caso de uso Generar Lista de Test

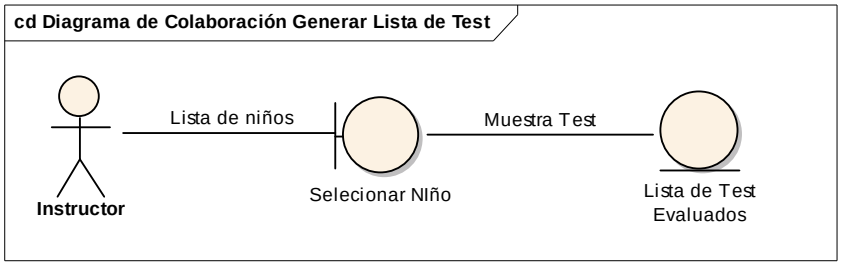


Figura 2.12 Diagrama de Colaboración de una realización del caso de uso Generar Lista de Test

FLUJO DE SUCESOS-DIAGRAMA DE COLABORACIÓN DE LA REALIZACIÓN DEL CASO DE USO GENERAR LISTA DE TEST
El instructor selecciona a un niño, a través de seleccionar el objeto niño, luego el objeto niño pide se le muestre el resultado del test al objeto Lista de Test Evaluados.

2.2.6.2 Paquete de Análisis

Con los casos de usos identificados en el Modelo de Casos de uso, y los diagramas de Colaboración en la sección anterior se identificaron los siguientes paquetes de análisis.

Paquete: Generar Test.

Este paquete representa el proceso de generar un test para que el niño lo ejecute. Agrupa a los casos de Uso:

Seleccionar niño.

Calificar test.

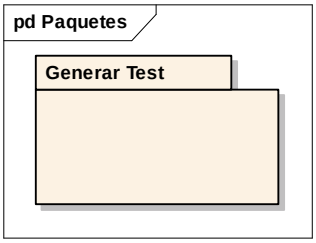


Figura 2.13 Paquete de análisis Generar Test.

Paquete: Administración de Usuarios

Este paquete permite dar un mantenimiento al perfil de los usuarios del sistema. Utiliza las clases que representan a los usuarios del sistema.

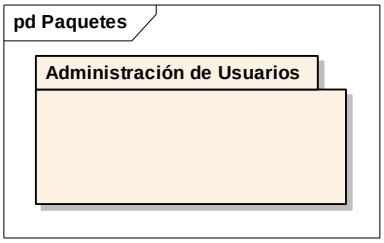


Figura 2.14 Paquete de análisis Administración de Usuarios.

2.2.7 Diseño

2.2.7.1 1dentificación de nodos y configuraciones de Red

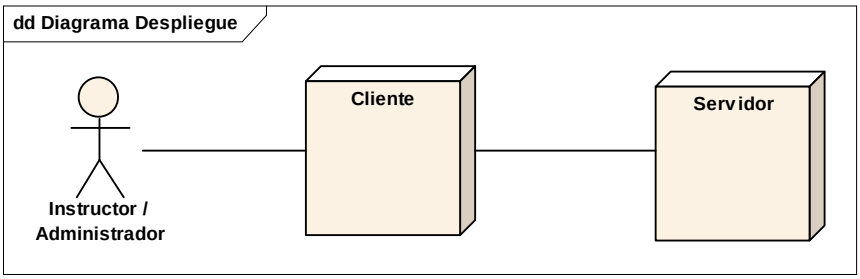


Figura 2.15 Diagrama de despliegue

Los usuarios del sistema podrán ingresar por medio del nodo Cliente, debe tomarse en cuenta que un usuario del sistema puede ser:

El Instructor.
El Administrador del Sistema.
En este nodo se valida el ingreso de cada uno de estos de acuerdo al perfil que haya sido asignado.

Los diagramas de secuencia que se muestran a continuación nos muestran mayor información del sistema:

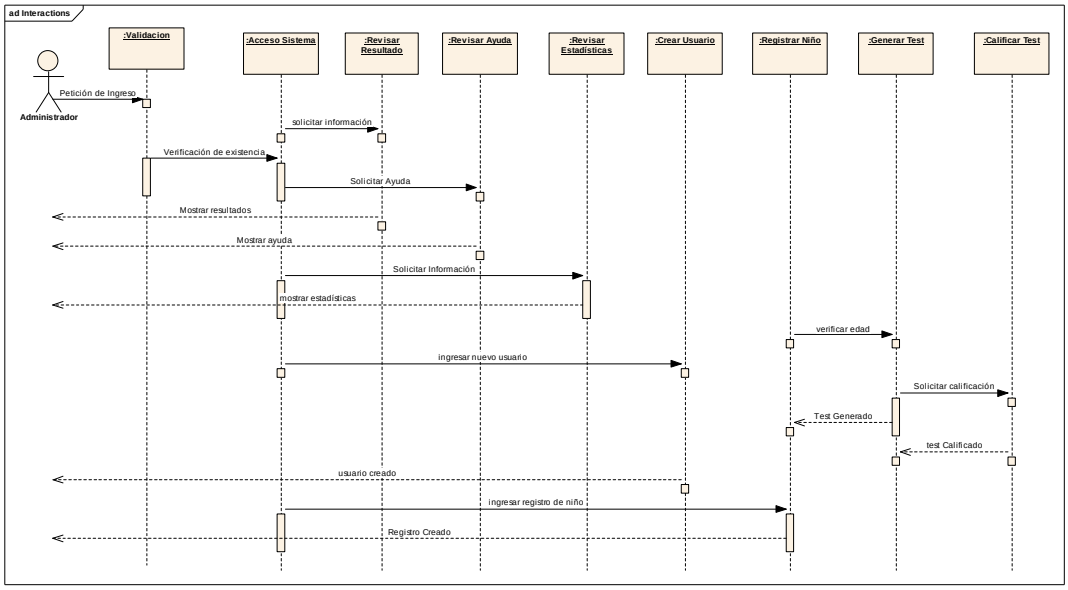


Figura 2.16 Diagrama de secuencia del Administrador

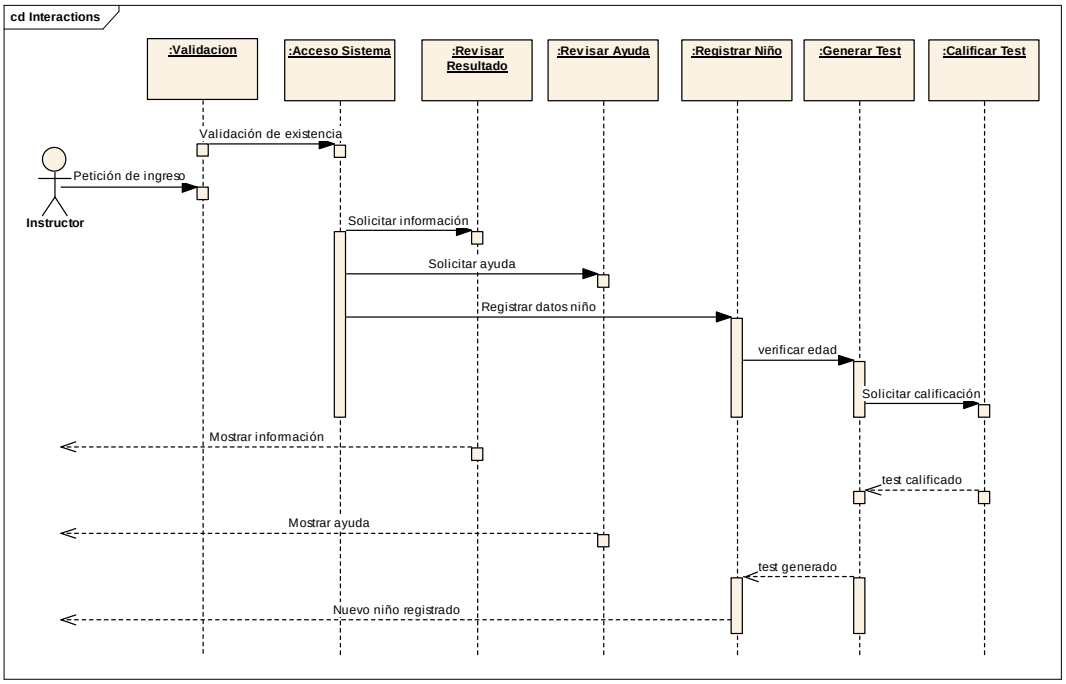


Figura 2.17 Diagrama de secuencia del Instructor

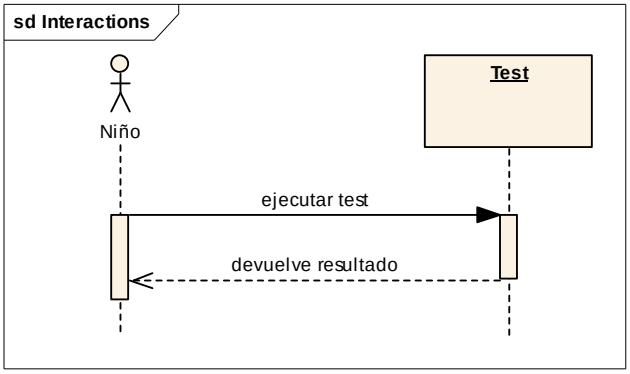


Figura 2.18 Diagrama de secuencia del Niño

2.2.7.2 Diagrama de Clases

A continuación se presenta el diagrama de las clases de diseño del sistema; en donde se encuentran las clases de entidad, clases de interfaz con sus relaciones y asociaciones.

Estas relaciones son las que probablemente nos conduzca a que ciertas clases de entidad formen parte de la base de datos como tablas.

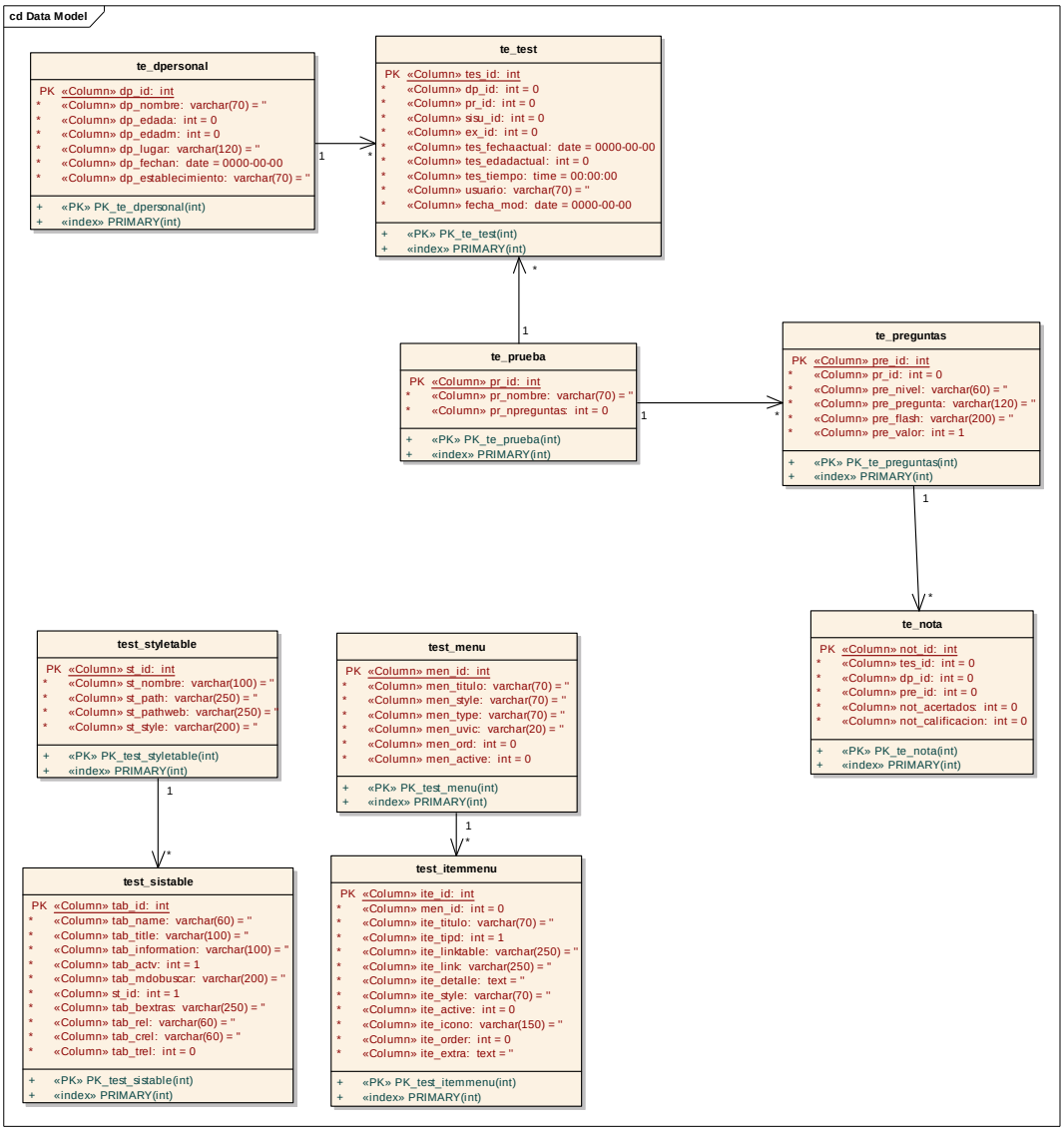


Figura 2.19 Diagrama de Clases

El PSP en la parte del diseño solicita las especificaciones de estado lo cual esta cubierto por el diagrama de estados del RUP, el cual se muestra en la figura 2.20.

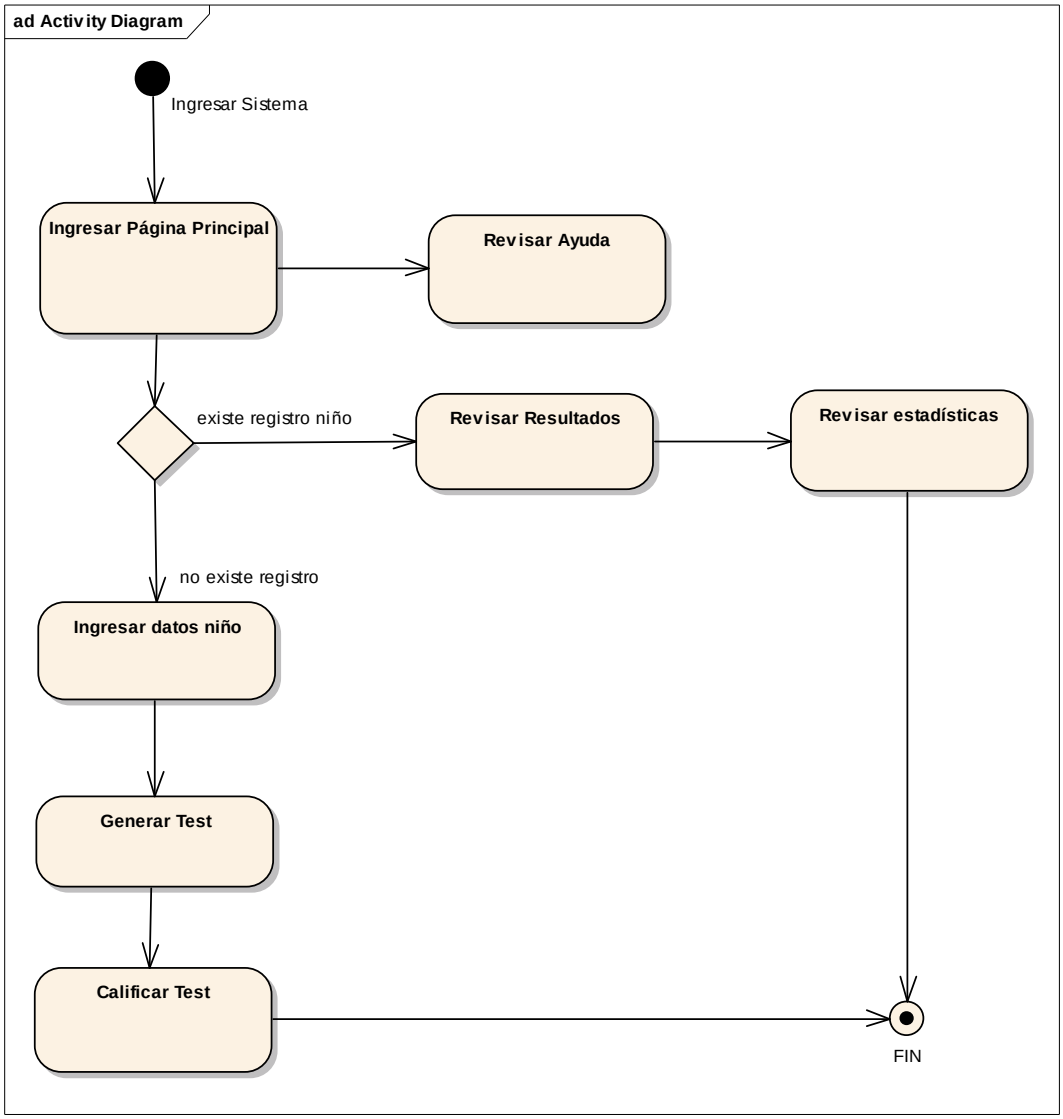


Figura 2.20 Diagrama de Estados

El PSP en la parte del diseño solicita el rastreo de problemas lo cual se muestra en la siguiente tabla:

Estudiante	Andrés Larco		Fecha	23/01/2007	
Programa	Resultados		Programa #	8	
Instructor	Carlos Montenegro		Lenguaje	Php	
Problema #:	Documentación	Fecha:	23/01/2007	Fase:	Planificación
Descripción:	Un mensaje erroneo				
Resolución:	Inserción de mensaje adecuado				
Fecha:	23/01/2007				
Problema #:	Sintaxis	Fecha:	23/01/2007	Fase:	Codificación
Descripción:	Error en puntuación				
Resolución:	Colocación de un ;				
Fecha:	23/01/2007				
Problema #:	Revisión	Fecha:	23/01/2007	Fase:	Compilación
Descripción:	Mensaje de alerta				
Resolución:	Corregir la formula				
Fecha:	23/01/2007				

Tabla 2.3 Log de Rastreo de Problemas

El siguiente paso que propone el PSP es revisar todo lo realizado en el flujo de trabajo de Diseño:

2.2.8 Implementación

2.2.8.1 Modelo de Datos

Seguidamente se presenta el modelo conceptual de la base de datos para el Sistema:

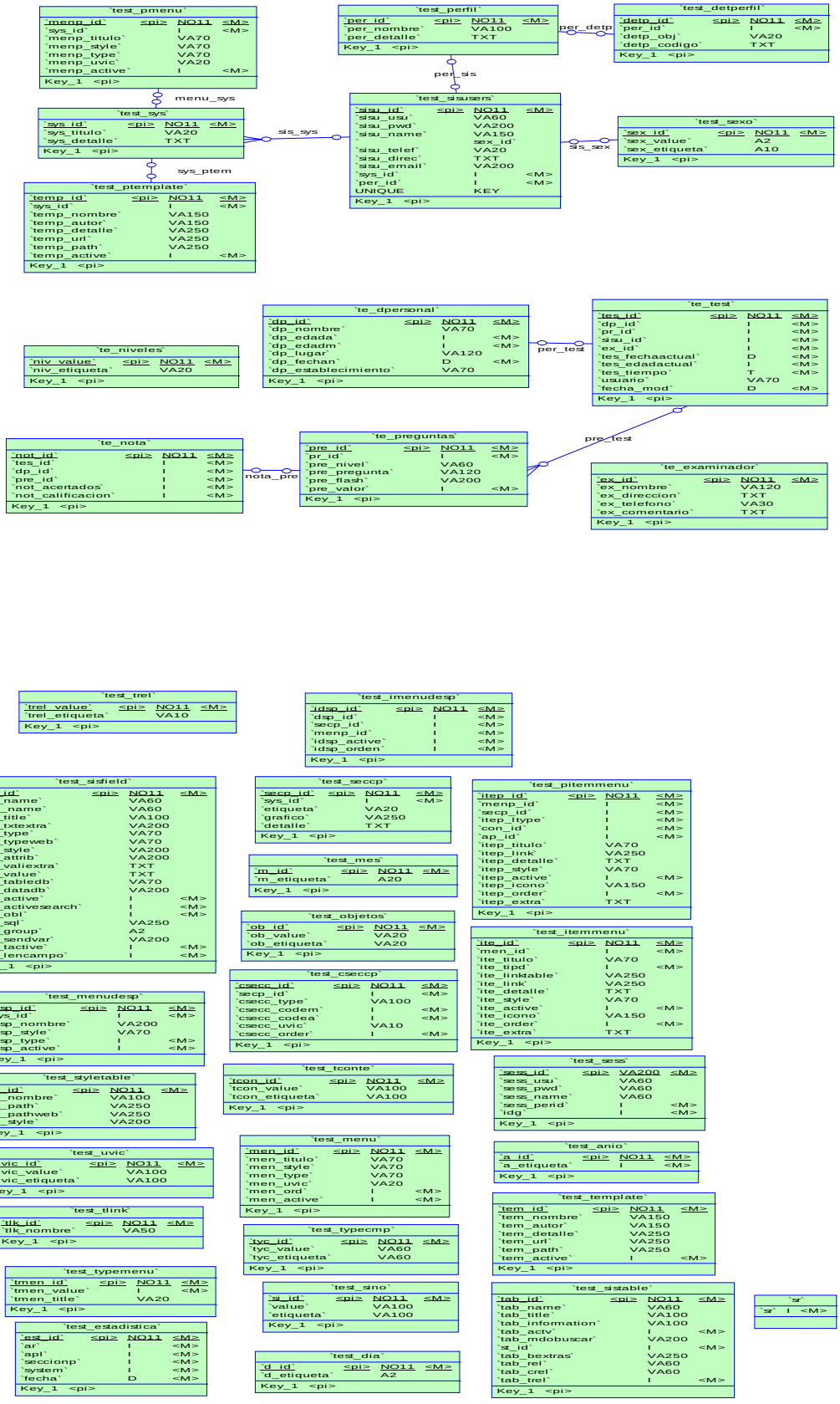


Figura 2.21 Modelo Conceptual de la Base de Datos

Seguidamente se presenta el modelo físico de la base de datos para el Sistema:

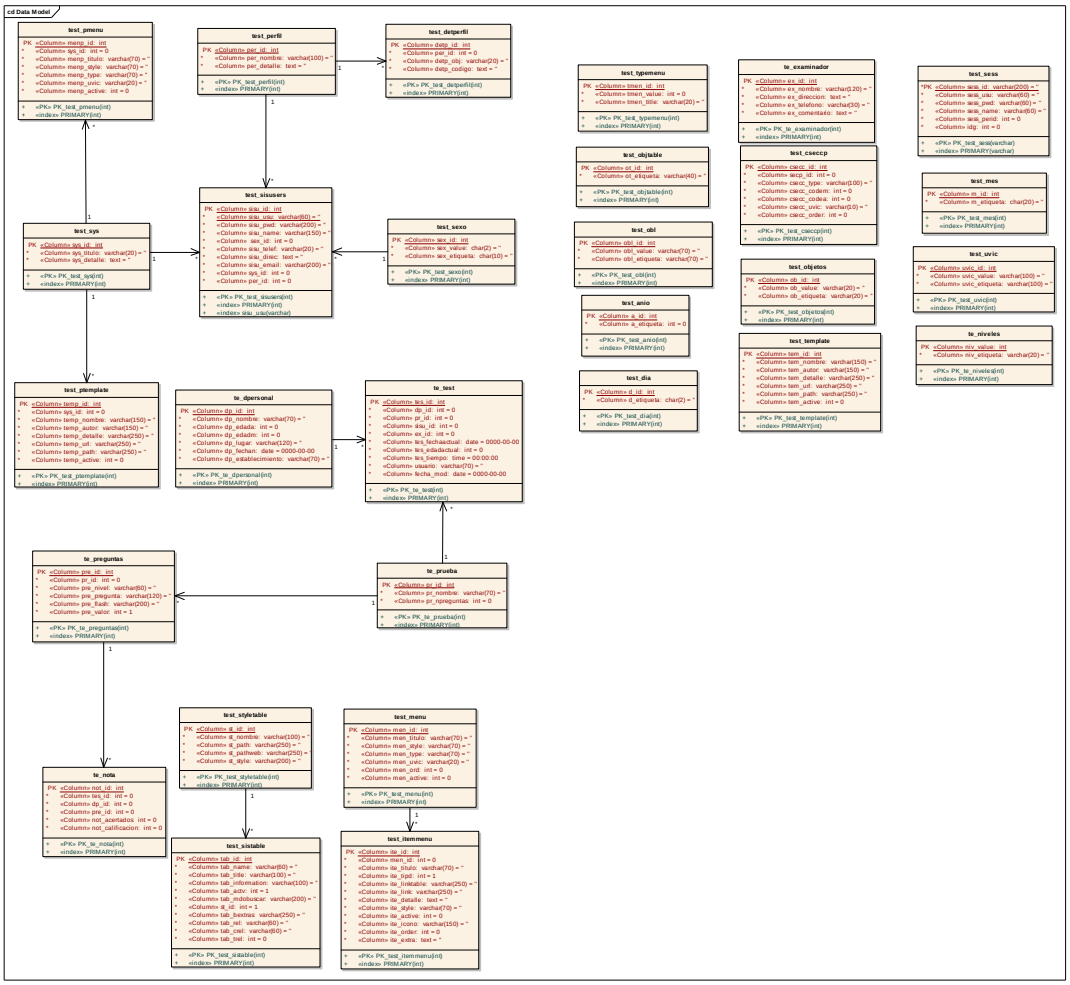


Figura 2.22 Modelo Físico de la Base de Datos

2.2.8.2 Generación de Código

La aplicación se desarrollará tomando en cuenta lo siguiente:

- El sistema esta diseñado para un ambiente Web.
- El sistema se adapta exclusivamente a niños con parálisis cerebral y analfabetos.

El código fuente.

Para la generación del código fuente se tomó en cuenta:

- Los nombres estarán relacionados directamente a la función central que realizan.
- Las partes referentes a operaciones en la base de datos se nombrarán según al tipo de acceso que realicen a la base de datos y a la información que presentarán o modificaran sea el caso.

A continuación se tiene muestra una parte del código:

```
<SCRIPT LANGUAGE=javascript>
<!--
function btest(){

                        //alert (testf.nino.options.value);
                        testf.action="index.php";
                        testf.submit();

}
//-->
</SCRIPT>

<form name="testf" method="post" action="">
  <div align="center" class="Estilo6">
    <input name="opcp" type="hidden" id="opcp" value="<?php echo $opcp; ?>">
    <input name="sessid" type="hidden" id="sessid" value="<?php  echo $sessid; ?>">
    Ni&ntilde;o:
    <select name="nino" onChange="btest();">
            <option value="0" selected>---Seleccione niño---</option>
            <?php
            $objformulario->fill_cmb("te_dpersonal","dp_id,dp_nombre",$nino);
            ?>
    </select>
  Test ID:
  <select name="test_id">
    <option value="0" selected>---Seleccione test---</option>
            <?php
            if ($nino)
            {

            $sqlrf12="select * from te_dpersonal,te_test where te_dpersonal.dp_id = te_test.dp_id  and
te_test.dp_id=".$nino;
            $resultadotrf12 = mysql_query($sqlrf12);
            if ($resultadotrf12)
                        {
                        while($rowtrf12 = mysql_fetch_array($resultadotrf12))
                                    {
                                                echo "<option value='".$rowtrf12["tes_id"]."'
selected>".$rowtrf12["tes_id"]."</option>";
                                    }
                        }
            }
            ?>
    </select>
  <input type="submit" name="Submit" value="Ver Resultados">
  </div>
</form>

<?php
$sqlrf1="select * from te_dpersonal,te_test,te_nota where te_dpersonal.dp_id = te_test.dp_id and
te_test.tes_id = te_nota.tes_id and te_nota.tes_id=".$test_id." and te_dpersonal.dp_id=".$nino;

$resultadotrf1 = mysql_query($sqlrf1);
```

```
if ($resultadotrf1)
            {
              $rowtrf1 = mysql_fetch_array($resultadotrf1)

?>

<table width="616" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="74"><span class="Estilo5">Nombre:</span></td>
    <td width="134" class="Estilo7"><?php echo $rowtrf1["dp_nombre"]; ?></td>
    <td width="49"><span class="Estilo5">Edad:</span></td>
    <td width="359" class="Estilo7"><?php echo $rowtrf1["dp_edada"]."años ".$rowtrf1["dp_edadm"]."
meses"; ?></td>
  </tr>
</table>
<table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
    <td width="36%" class="Estilo5">Lugar y fecha de naciemiento:</td>
    <td colspan="2"><span class="Estilo7"><?php echo $rowtrf1["dp_lugar"]; ?>, <?php echo
$rowtrf1["dp_fechan"]; ?></span></td>
  </tr>
  <tr>
    <td class="Estilo5">Escolaridad:</td>
    <td colspan="2" class="Estilo7"> </td>
  </tr>
  <tr>
    <td class="Estilo5">Establecimiento del que procede: </td>
    <td colspan="2" class="Estilo7"><?php echo $rowtrf1["dp_establecimiento"]; ?></td>
  </tr>
  <tr>
    <td class="Estilo5">Fecha de examen: </td>
    <td width="26%" class="Estilo7"><?php echo $rowtrf1["tes_fechaactual"]; ?></td>
    <td width="38%"><strong>Examinador:<span class="Estilo7"><?php echo $rowtrf1["usuario"];
?></span></strong></td>
  </tr>
</table>
<?php
}
?>

<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr>
    <th scope="col"><table width="383" border="1" cellpadding="0" cellspacing="1">
      <tr>
        <th width="125" class="Estilo5" scope="col">Edad / Preguntas </th>
        <th width="20" class="Estilo12" scope="col">1</th>
        <th width="20" class="Estilo12" scope="col">2</th>
        <th width="20" class="Estilo12" scope="col">3</th>
        <th width="20" class="Estilo12" scope="col">4</th>
        <th width="20" class="Estilo12" scope="col">5</th>
        <th width="20" class="Estilo12" scope="col">6</th>
        <th width="40" class="Estilo5" scope="col">A&ntilde;os en Meses </th>
        <th width="40" class="Estilo5" scope="col">Meses</th>
      </tr>
      <tr>
        <th scope='row'><span class="Estilo12">2 A&ntilde;os </span></th>
        <?php
$sqlrf1="select * from te_dpersonal,te_test,te_nota, te_preguntas where te_preguntas.pre_id =
te_nota.pre_id and te_dpersonal.dp_id = te_test.dp_id and te_test.tes_id = te_nota.tes_id and
te_nota.tes_id=".$test_id." and te_dpersonal.dp_id=".$nino." and pre_nivel=1" ;
                                $resultadotrf1 = mysql_query($sqlrf1);
                                if ($resultadotrf1)
                                        {
                                          while($rowtrf1 = mysql_fetch_array($resultadotrf1))
                                                                {
                                                                echo "<td>".$rowtrf1["not_calificacion"]."</td>";
                                                                $suma[1]=$suma[1]+$rowtrf1["not_calificacion"];
```

```
                                                }
                                                if ($suma[1]==6)
                                                        {
                                                         $sumanios[1]=1;

                                                        }
                                        }

                                        $anios[1]=24;
                                        echo "<td>".$anios[1]."</td>";
                                        echo "<td>".$suma[1]."</td>";
                        ?>
        </tr>


        <tr>
          <th scope="row"><span class="Estilo12">3 A&ntilde;os 6 meses </span></th>
          <?php

$sqlrf1="select * from te_dpersonal,te_test,te_nota, te_preguntas where te_preguntas.pre_id =
te_nota.pre_id and te_dpersonal.dp_id = te_test.dp_id and te_test.tes_id = te_nota.tes_id and
te_nota.tes_id=".$test_id." and te_dpersonal.dp_id=".$nino." and pre_nivel=12";
                                $resultadotrf1 = mysql_query($sqlrf1);
                                if ($resultadotrf1)
                                        {
                                          while($rowtrf1 = mysql_fetch_array($resultadotrf1))
                                                {
                                                echo "<td>".$rowtrf1["not_calificacion"]."</td>";

                                                $suma[12]=$suma[12]+$rowtrf1["not_calificacion"];
                                                }
                                        $suma[12]=$suma[12]*2;
                                        if ($suma[12]==12)
                                                        {
                                                         $sumanios[12]=1;

                                                        }


                                        }
                                        $anios[12]=72;
                                        echo "<td>".$anios[12]."</td>";
                                        echo "<td>".$suma[12]."</td>";
                        ?>
        </tr>
        <tr>
          <th scope="row"><span class="Estilo12">11 A&ntilde;os</span></th>
          <?php
        </tr>

    </table></th>
    <th scope="col">   <?php



 for ($i=1; $i<=16; $i++)
                {
                        //echo $sumanios[$i]."<br>";
                        if ($sumanios[$i])
                        {
                          $uvic=$i;
                          $i=16;
                        }
                }
//echo "Ubicacion de la respuesta completa:".           $uvic."<br>";
```

```
 for ($i=$uvic; $i<=16; $i++)
            {
                        $totalsuma=$totalsuma+$suma[$i+1];
            }
echo "Total suma de respuestas".$totalsuma."<br>";


$totalfinal=$totalsuma +  $anios[$uvic]  ;
echo "<br>Resultado:".$totalfinal."<br>";
            ?></th>
  </tr>
</table>
```

En este flujo de trabajo el PSP recomienda utilizar la plantilla de registro de defectos:

Estudiante	Andrés Larco				Fecha	23/01/2007
Instructor	Carlos Montenegro				Lenguaje	Php
Fecha	**Número**	**Tipo**	**Inyectado**	**Removido**	**Tiempo reparado**	**Defectos Reparados**
23/01/2007	2	10	Planificación	Planificación	20	2
Fecha	**Número**	**Tipo**	**Inyectado**	**Removido**	**Tiempo reparado**	**Defectos Reparados**
23/01/2007	1	20	Codificación	Codificación	30	1
Fecha	**Número**	**Tipo**	**Inyectado**	**Removido**	**Tiempo reparado**	**Defectos Reparados**
23/01/2007	2	60	Compilación	Compilación	30	1

Tabla 2.4 Log de registro de Defectos

2.2.9 Pruebas

Las pruebas que se realizaran al Sistema están dirigidas fundamentalmente al cumplimiento eficaz de la solución y verificar el fácil manejo para los usuarios que son el instructor y los niños.

2.2.9.1 Caso de Prueba para el caso de uso Registrar / Seleccionar Niño.

Caso de Prueba:	Registrar la niña Arboleda
Entrada:	Llenar el formulario de los datos personales de la niña y guardar.
Resultado:	Luego de la prueba se deben tener los datos de la niña almacenados en la

	base de datos.
Condiciones:	Debe de existir la solicitud para que la niña se evaluada con el sistema.

2.2.9.2 Procedimiento de Prueba para el caso de uso Registrar / Seleccionar Niño.

1. Seleccionar en la pantalla niño.
2. Llenar el formulario con los datos personales.
3. Dar un clic en el botón guardar.

2.2.9.3 Caso de Prueba para el caso de uso Revisar Resultado del Test.

Caso de Prueba:	Revisar resultado del Test de la niña Arboleda.
Entrada:	Debe existir un test calificado de la niña Arboleda.
Resultado:	El resumen del test evaluado a la niña Arboleda contados sus datos.
Condiciones:	Debe cumplirse con las condiciones de inicio y fin de la evaluación el test.

2.2.9.4 Procedimiento de Prueba para el caso de uso Revisar Resultado del Test.

1. Seleccionar en la pantalla resultados.
2. seleccionar niño.
3. Seleccionar el del test calificado.
4. Dar un clic en el botón ver resultados.

El PSP propone la siguiente plantilla para registra las pruebas realizadas al sistema:

Estudiante	Andrés Larco	Fecha	23-01-07
Programa	Resultados	Programa #	8
Instructor	Carlos Montenegro	Lenguaje	php
Nombre y Número de Prueba	Registrar Niña Arboleda		

Objetivo de la Prueba	Verificar que los datos se registraron correctamente en la Base de Datos.
Descripción de la Prueba	Registrar los datos personales de la niña en el respectivo formulario
Condiciones de la Prueba	Datos completos
Resultados Esperados	Que todos los datos estén bien en la base de datos
Resultados Actuales	Datos registrados correctamente.

Tabla 2.5 Plantilla de Reporte de Pruebas

2.2.10 Implantación

En este flujo de trabajo indica la instalación del sistema, el software adicional requerido que debe tener la máquina donde se va a ejecutar el sistema.

El PSP aporta con el resumen del plan proyecto y el resumen del ciclo. La cual se muestra en el anexo Probe.

2.3 Evaluación del Sistema.

La evaluación del sistema fue llevada acabo por el Dr. Clímaco Vinueza, Director del Instituto Especial del Norte, dedicado a terapia de niños con necesidades especiales. El citado profesional verifico pregunta a pregunta si el test estaba de acuerdo a los requerimientos que el mismo sugirió. Uno de los aspectos evaluados fue que las imágenes utilizadas estén en blanco y negro y

máximo en una escala de grises. El motivo de esta consideración es que si al Test se le añade color cambia su fundamento y con mayor razón su interpretación.

Luego de una extensa revisión el Dr. Vinueza dio la aceptación del sistema encontrándose satisfecho en cuanto a la interfase, la funcionalidad y a los resultados del Test.

2.4 Evaluación de la combinación PSP y RUP.

Como se ha establecido anteriormente, la combinación del PSP y RUP se ha realizado de la siguiente manera: la metodología RUP se ha puesto como base, y sobre esta se han incorporado las prácticas que propone el PSP.

El PSP exige la descripción del trabajo a realizarse, la misma que se presenta completa en la misión y alcance del sistema, componente de RUP. En este punto la combinación de RUP y PSP es natural y no requiere consideraciones especiales, puesto que la descripción del trabajo en RUP inclusive es de carácter formal.

En la metodología RUP no existe dentro de sus flujos de trabajo la planificación, por lo cual es insertada entre los requerimientos y análisis. Para ello se utiliza el script de planificación del proceso PSP, que a su vez utiliza una plantilla mostrada en la Tabla 2.2. Esta inserción tiene sentido, puesto se realiza la Planificación luego de conocidos los requerimientos, lo cual permite definir las actividades posteriores y la estimación de sus tiempos de ejecución. Es decir, en este rubro se complementan RUP y PSP.

En el flujo de trabajo análisis no se inserta ninguna práctica del PSP, dado que no existen definidas para el caso.

En el flujo de trabajo diseño se encuentra el diseño global del sistema y la especificación, los mismos que son representados por los diagramas de:

Secuencia y Clases. Los diagramas cubren totalmente con los requerimientos de diseño del PSP, razón por la cual la combinación en este punto es natural.

En el flujo de trabajo de implementación, el PSP al igual que el RUP establecen que se debe escribir código, al mismo que se lo revisa, repara y se anotan todos los defectos encontrados; para el efecto se usa la plantilla de Resumen del Ciclo. En esta etapa, el PSP aporta a completar la especificación del RUP, dotándolo de detalles para el tratamiento del código

En el flujo de trabajo pruebas, además de lo especificado en RUP, se utiliza la plantilla de reporte de pruebas de PSP, donde se indica el objetivo de la prueba, los resultados esperados y los resultados actuales. En esta etapa el PSP aporta a completar la especificación del RUP, dotándolo de detalles para la realización y reporte de pruebas.

En la implantación el PSP propone que se debe tener todas las plantillas solicitadas en los pasos anteriores, lo cual complementa lo especificado en RUP.

Desde el punto de vista práctico, la clave para la combinación de la metodología RUP y las prácticas del PSP, es tener un buen conocimiento de los dos, para irlos usando de una manera efectiva.

Teniendo en cuenta los principios de planificación y calidad del PSP, el desarrollo de software es más efectivo, ya que es más eficiente prevenir defectos que repararlos. Y la manera correcta es siempre la más rápida y la más barata.

Por otro lado, el caso de estudio ha permitido ir poniendo en práctica la combinación que se esta sugiriendo, cuyos resultados se documentan debidamente.

Experiencia Personal con el uso del PSP

Partiendo de la premisa que cada ingeniero es diferente, y por consiguiente su trabajo va ha ser distinto, el PSP aplicado al caso de estudio de los niños con parálisis cerebral, se ha llevado acabo de la siguiente manera. Los niños tienen necesidades y características específicas, las cuales son atendidas en ese contexto.

El test de Terman y Merrill permite medir la edad mental y el coeficiente intelectual, el mismo que tiene una vigencia desde los años 40 cuando dichos autores publicaron su trabajo, desde esa fecha hasta la actualidad el test se lo ha llevado de manera manual, los materiales son: tarjetas con ilustraciones en blanco y negro, figuras de madera, tarjetas con listas de palabras, con todo el material descrito anteriormente, se procede a evaluar a los niños.

Han pasado 66 años para que los sistemas pongan al servicio del ser humano su estructura de software y hardware, y colaboren en desarrollo de un test virtual.

El primer paso fue leer todo el Test de Terman y Merrill, lo cual tomo una semana de trabajo, al finalizar la lectura tenía más preguntas que certezas, las cuales fueron despejadas por el psicólogo.

A continuación el psicólogo me envió la tarea traer las imágenes que había asociado a cada una de las preguntas, con entusiasmo me dedique a esa tarea, regrese con el material que aparentemente para mi compresión estaban bien, lo cual no fue cierto.

La primera impresión del psicólogo fue de admiración he hizo el siguiente comentario “Están muy bonitas”, cuando uno escucha una frase de esas piensa lo logre!, pero la verdad era distinta, las imágenes escaneadas que había llevado eran a colores, y cuando las íbamos revisando el psicólogo me dio la siguiente explicación: Las tarjetas con imágenes que le preste estaban en blanco y negro, y además no tienen todas las imágenes que necesitamos, por

ese motivo le fue enviado de tarea conseguir el resto, el motivo de que todo en material no tenga color, es porque si le añade color, el test cambia su finalidad y su validez. Su finalidad es medir la edad mental y su validez se comprobó en 3.184 casos que fue la muestra para su estudio.

La experiencia citada anteriormente sirvió para darle un giro a mi trabajo, organizando un cronograma de trabajo con el psicólogo, se determino avanzar por niveles, los cuales consisten en elaborar las imágenes y sus respectivas animaciones en flash, de cada una de las preguntas, cada nivel consta de 6 preguntas y son 14 niveles que van desde los 2 años hasta los 12 años. Además se llego a establecer que las imágenes sean en blanco y negro, y en escala de grises por lo argumentado anteriormente.

A continuación se describe tres preguntas en las cuales se detalla, su grado de dificultad de ser llevadas de lo físico y material al intangible que es el software:

Nivel: 2 años
Pregunta # 1: Tablero de escavado
Grado de dificultad: bajo.
Detalle:
Es un tablero que tiene tres figuras: cuadrado, triangulo y circulo; a un rectángulo se le realizaron tres recortes con las figuras mencionadas, a continuación cada uno de esos recortes se los transformo en botones, los cuales al dar un clic sobre ellos se colocan en su respectivo lugar del rectángulo que simula un tablero.

Nivel: 10 años
Pregunta # 3: Lectura y relato
Grado de dificultad: Mediano
Detalle:
Es una lectura de un incendio en la ciudad, la dificultad radico en que todo el test tiene imágenes y esta es la única pregunta que no pudo ser llevada a imágenes, lo cual se soluciono de la siguiente manera, cada grupo de palabras

fue transformado en botones, las cuales deben ser bien pronunciadas por el niño, de esa manera se demuestra que es valida la lectura.

Nivel: 8 años
Pregunta # 2: Memoria de un relato
Grado de dificultad: Alto
Detalle:
Es un cuento de un caballito travieso, el cual fue pasado a imágenes, cada diapositiva del flash tiene una escena del cuento, de esta forma se hacen las preguntas al niño para saber que recuerda del relato.

People CMM

El modelo de la madurez de la capacidad de la gente (People CMM) es un modelo de organización del cambio, diseñado en función de la premisa que no sobrevivirán las prácticas mejoradas de la mano de obra, a menos que el comportamiento de una organización cambie para apoyarlas.

Fue desarrollado para dirigir sistemas y organizaciones de software, basado en la motivación, y en la retención del personal técnico talentoso.

El People CMM proporciona la dirección que mejora la capacidad de la organización, de satisfacer sus objetivos a través de una mano de obra competente, capaz.

1) Inicial.
En el nivel 1, una organización no tiene ninguna manera constante de realizar prácticas de la mano de obra.

2) Repetible.
En el nivel 2, las organizaciones establecen una fundación, en la cual desarrollen prácticas comunes de la mano de obra a través de la organización. La meta del nivel 2 es tener responsabilidad de la toma de los encargados de manejar y de desarrollar a su gente.

3) Definido.

En el nivel 3, la organización identifica y desarrolla capacidades de la mano de obra y alinea capacidades de la mano de obra y del grupo de trabajo con estrategias y objetivos de negocio.

4) Manejado.

En el nivel 4, la organización autoriza e integra capacidades de la mano de obra y maneja funcionamiento cuantitativo.

5) Optimización.

En el nivel 5, la organización mejora y alinea continuamente personal, grupo de trabajo, y capacidad de organización.

Algunas de las tecnologías complementarias de la People CMM son: CMMI, SW-CMM, IPPD, IPM, Integrated Teaming, Organizational Environment for Integration, y Total Quality Management.

El People CMM permitir que las organizaciones de software integren la mejora de la mano de obra con los programas de mejora de proceso del software dirigidos por el SW-CMM

El People CMM y el PSP consideran la mano de obra o persona como el principal actor del proceso dentro de un sistema u organización. Auque no sean tecnologías complementarias.

CAPITULO III

3 CONCLUSIONES y RECOMENDACIONES

3.1 Conclusiones.

1. Los principios del PSP ayudan a producir productos de software de calidad, puesto que constituye un conjunto de prácticas que enseña a los ingenieros a;
   - Administrar su tiempo y compromisos.
   - Realizar mejores estimaciones y planes de trabajo.
   - Medir y prevenir defectos.
   - Comprometerse con la calidad del producto.

2. La inserción de las prácticas del PSP se las ha realizado sobre los flujos de trabajo del RUP. En el trabajo se realizó una sola iteración.

3. La planificación hace que los resultados sean mejores, conociendo detalles del trabajo personal de quien desarrolla el software, de esa manera se reducen las diferencias entre el tiempo planeado y el tiempo gastado efectivamente.

4. Al saber cuanto y en que tiempo se escribe el código del sistema, es un aporte valioso del PSP para que la implementación sea efectiva.

5. El llevar un control sobre las pruebas realizadas ayuda a saber exactamente que se probó y cual fue su resultado.

6. El caso de estudio muestra la forma práctica de utilizar las plantillas del PSP dentro de la metodología de software RUP

7. Se encontró alguna dificultad al momento de aplicar la combinación, debido a que al registrar los datos solicitados en las plantillas del PSP se debe tener en consideración sus cálculos.

8. El sistema desarrollado es de fácil uso, para que los niños con parálisis cerebral ejecuten el Test adecuadamente, tal como lo certificó el especialista.

3.2 Recomendaciones.

1. Se recomienda poner en práctica los 6 principios de planificación y calidad en los que se fundamenta el PSP, los cuales sirven para que el trabajo de la ingeniería, produzca software de manera correcta y efectiva.

2. El momento de realizar el marco teórico del RUP, se encontró que la versión en inglés y la versión en español del mismo texto contiene una diferente estructura de los flujos de trabajo, así que sería adecuado para nuestra carrera sugerir un marco referencial estándar.

3. Se recomienda profundizar el estudio del PSP, lo cual ayudaría hacer más efectivos a los ingenieros que desarrollan software. Eso ayudaría, por ejemplo, a que los estudiantes a planificar su trabajo y medir el tiempo que se tardan en el mismo.

4. Es conveniente hacer énfasis en el uso de métodos matemáticos para realizar todos cálculos necesarios, para estimar de mejor manera los datos que orienten en el trabajo utilizando los principios del PSP.

5. Se recomienda que los casos de estudio para la aplicación de la combinación del PSP y RUP, no sean complejos y extensos, por que este trabajo esta diseñado para proyectos pequeños.

Bibliografía

[1] Watts S. Humphrey, A Discipline for Software Engineering, Ed. Addison Wesley Longman, 1999.

[2] Personal Software Process (PSP), Carnegie Mellon University, http://www.sei.cmu.edu/tsp/psp.html, Acceso último: Marzo 2005.

[3] PSP and TSP Publications, Carnegie Mellon University, http://www.sei.cmu.edu/tsp/publications.html#fitin, Acceso último: Mayo 2005.

[4] Watts S. Humphrey, The Software Quality Profile, Software Engineering Institute Carnegie Mellon University, Pittsburgh, http://www.sei.cmu.edu/publications/articles/quality-profile/index.html, Acceso último: Junio 2005.

[5] Watts S. Humphrey, Three Dimensions of Process Improvement Part II: The Personal Process, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, http://www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1998/03/dimensions.asp, Acceso último: Junio 2005.

[6] Watts S. Humphrey, The Personal Software Process^{SM} (PSP^{SM}), TECHNICAL REPORT CMU/SEI-2000-TR-022 ESC-TR-2000-022, http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr022.pdf, Acceso último: Septiembre 2005.

[7] Will Hayes James W, The Personal Software Process^{SM}(PSP^{SM}): An Empirical Study of the Impact of PSP on Individual Engineers, Technical Report CMU/SEI-97-TR-001 ESC-TR-97-001, http://www.sei.cmu.edu/publications/documents/97.reports/97tr001/97tr001abstract.html. Acceso último: Enero 2006.

[8] Ivar Jacobson, Grady Booch, James Rumbaugh, El Proceso Unificado de Desarrollo de Software, Ed. Pearson Education, 2000, 1ra ed.

[9] Philippe Kruchten, The Rational Unified Process an Introduction, Ed. Addison Wesley, 1999, 2da ed.

[10] Roger Pressman, Ingeniería de Software un enfoque Práctico, Ed. MC GRAW HILL, 1997, 4ta ed.

[11] EMAM, K. Elements of Software Process, Assesment & Improvement. IEEE Computer Society. Washington. 1999.

Anexos

Anexo 1:

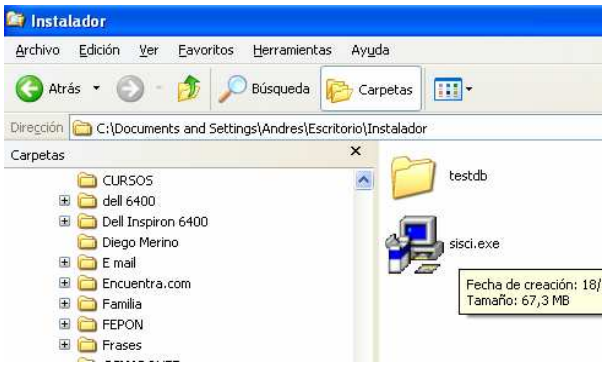
Manual de instalación del Sistema SISCI

Prerrequisitos:

1. Instalar xampp-win32 para Windows, en el cual se instalara el servidor apache y mysql.
2. En la siguiente dirección C:\Archivos de programa\xampp\mysql\data la carpeta que contiene la base de datos del sistema cuyo nombre es testdb.

Pasos para la instalación del Sistema Sisci

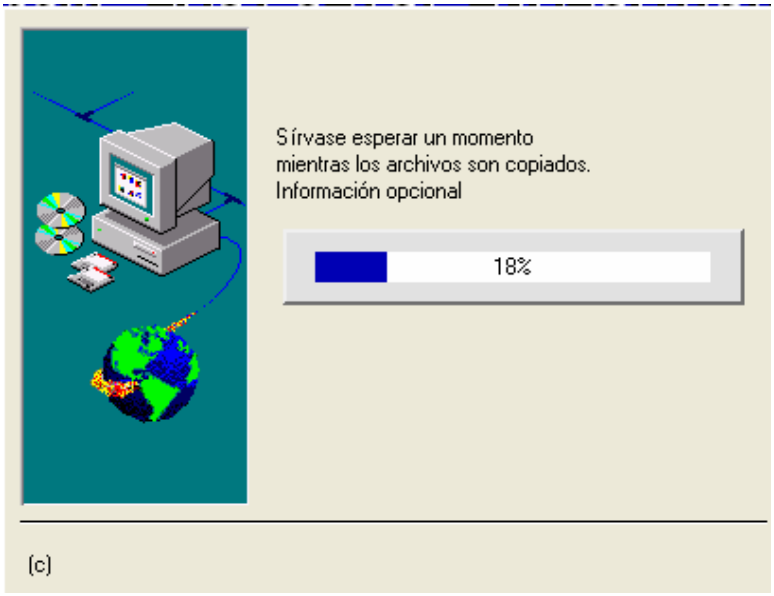
1. Seleccionar sisci.exe y dar un enter.



2. En esta pantalla se indica que se va ha instalar el sistema y que se recomienda se cierren otros programas.

3. El esta pantalla se indica la carpeta donde se deben copiar los archivos de instalación

4. La instalación en proceso



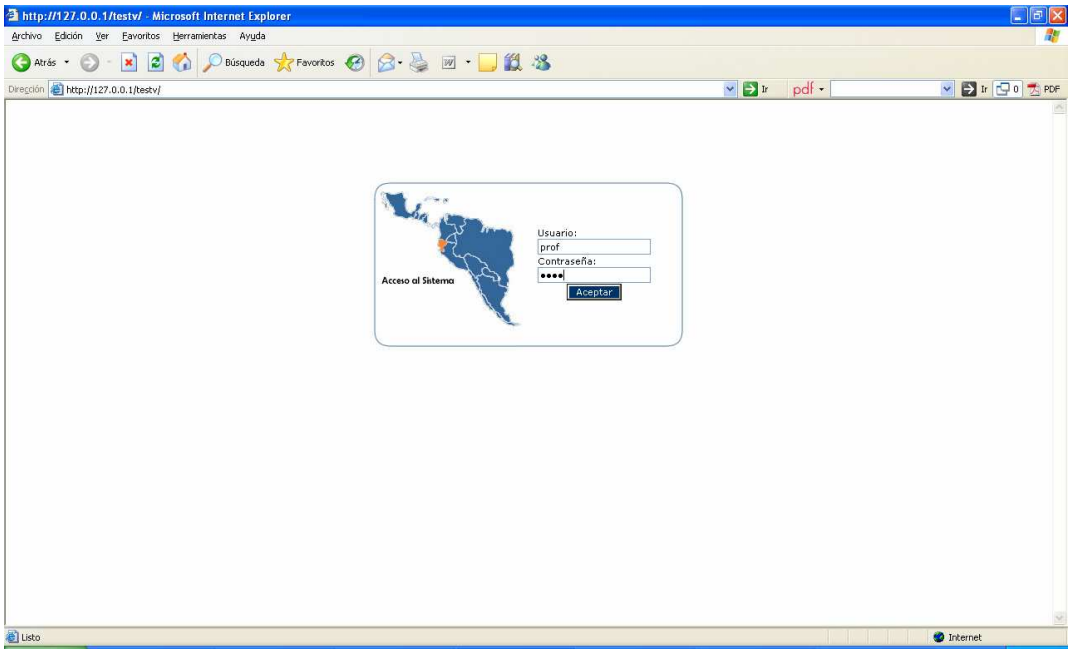
5. En esta pantalla termina el proceso de instalación del sistema sisci.

Anexo 2

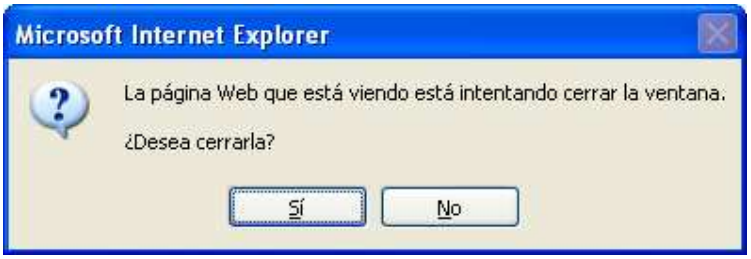
Manual del Instructor

Pantalla de ingreso al sistema:

En esta pantalla se digita el nombre del usuario y su respectiva clave.

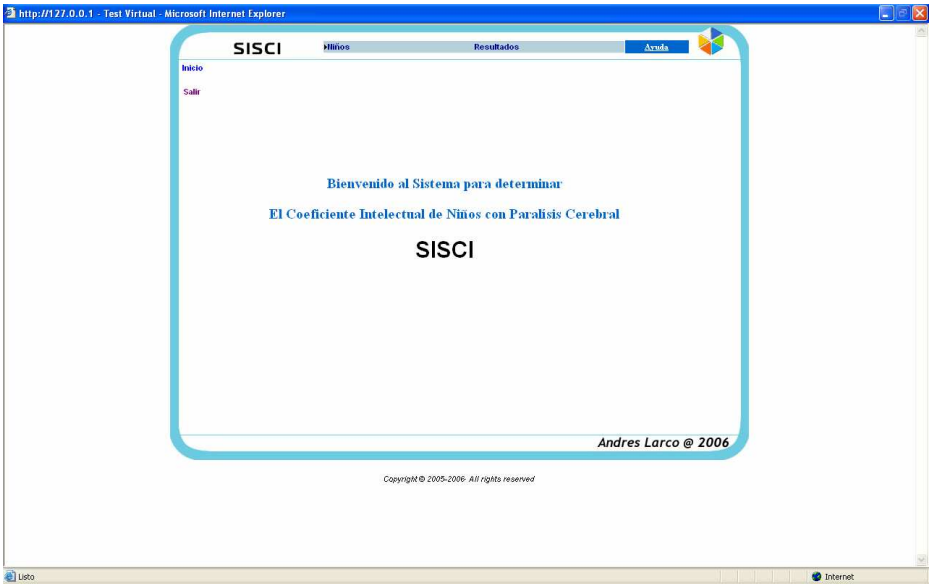


Este mensaje indica que las barras del Internet Explorer van a ocultarse y se va a ver el sistema SISCI



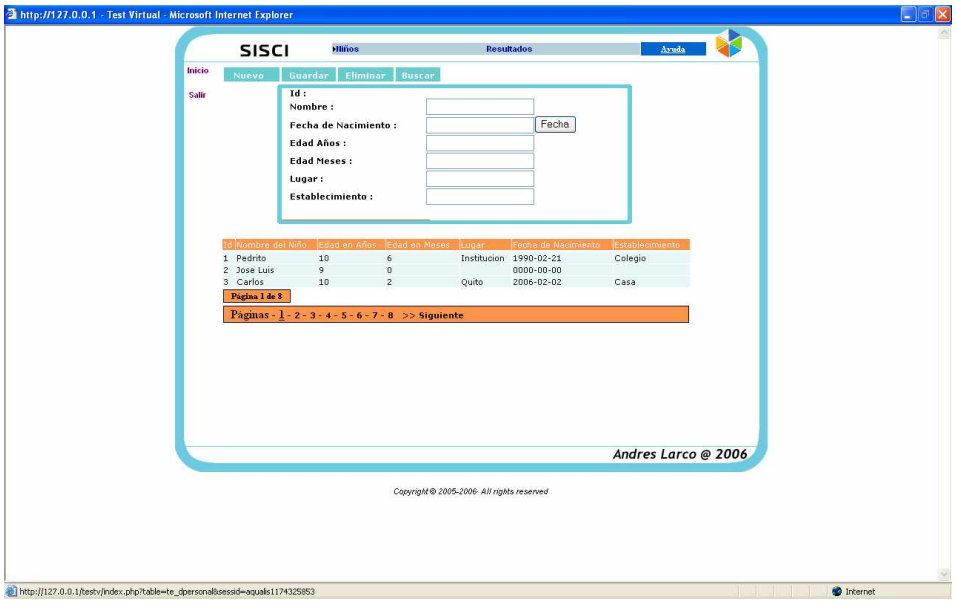
Pantalla de inicio:

Una vez que se ingresa al sistema se tiene la siguiente pantalla de inicio.



Pantalla de niños:

En esta pantalla se registra los datos personales del niño, se tiene cuatro opciones, nuevo, guardar, eliminar y buscar.

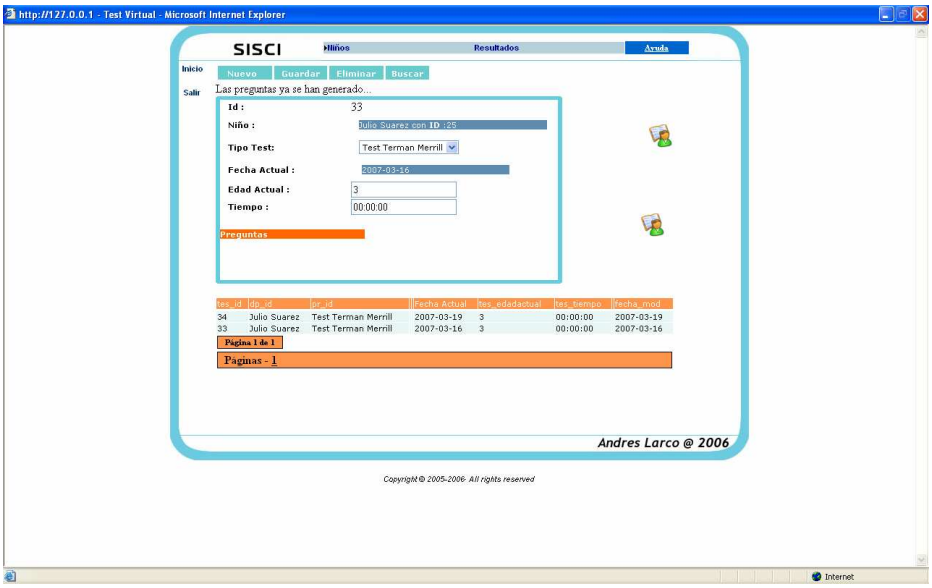


Pantalla con un niño registrado:

En esta pantalla se hace un nuevo ingreso

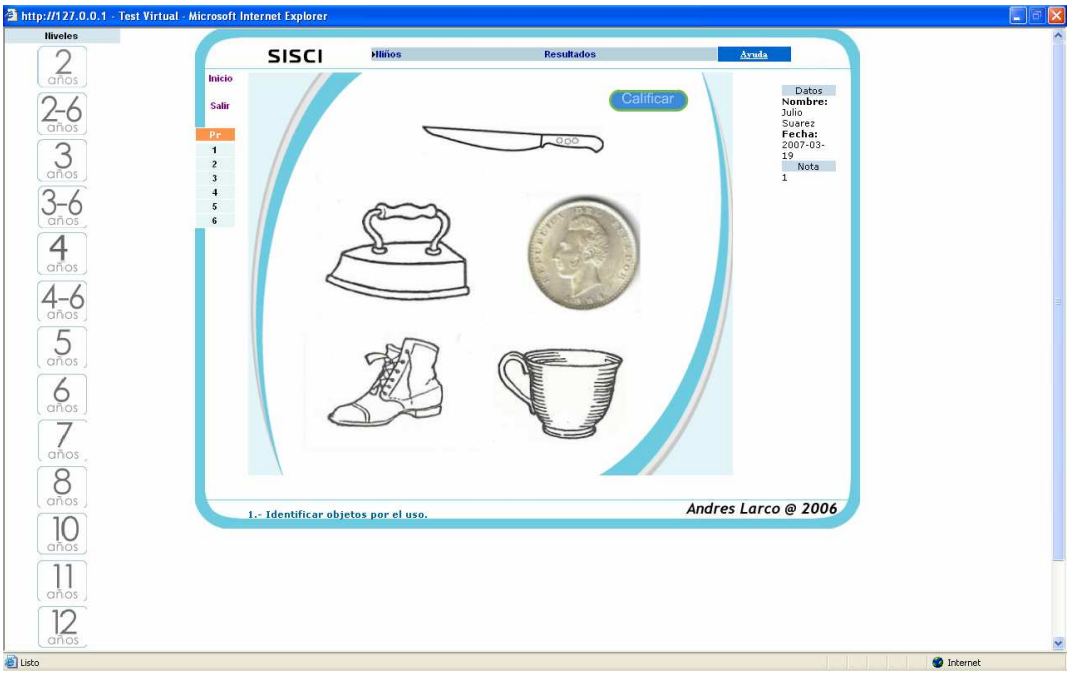
1. Nombre del niño.
2. Fecha de nacimiento que se selecciona del botón fecha.
3. La edad en meses y la edad en años se calcula automáticamente.
4. Se registra el lugar donde se esta evaluando el test.
5. La institución a la que pertenece.

En esta pantalla se selecciona al niño al cual se le va a crear un test y se generan las preguntas:



Al hacer clic en Preguntas se da inicio al test, y aparece la siguiente pantalla:

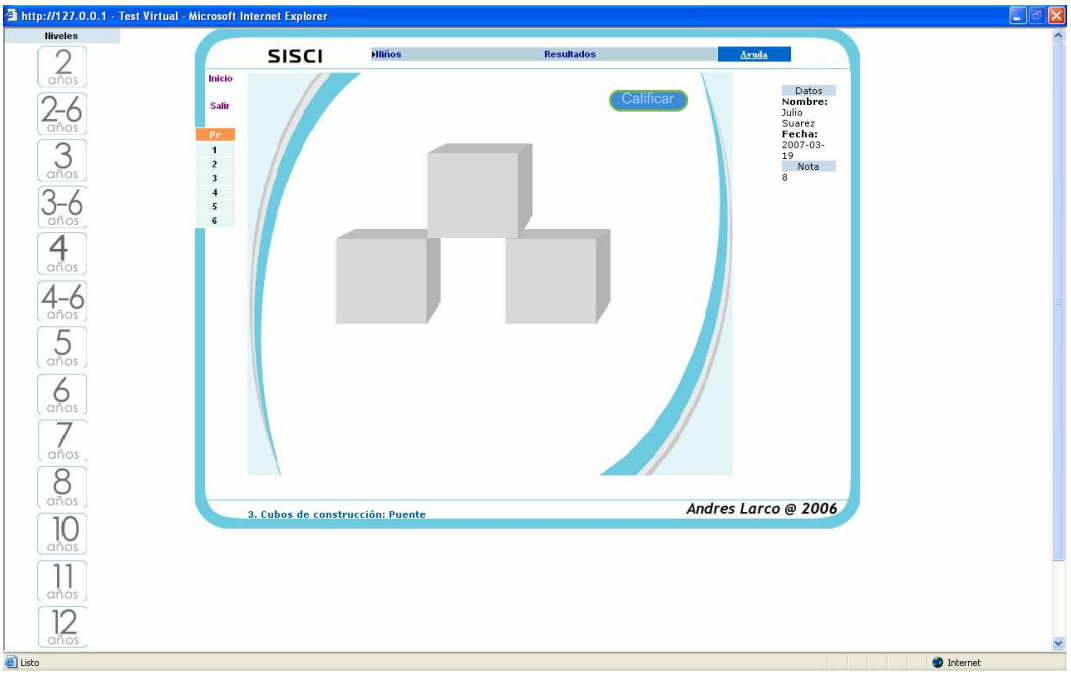
Siguiendo la norma para iniciar el examen, se procede a iniciar en las preguntas de 2-6 años:



Pregunta 2 Identificar partes del cuerpo:

Última pregunta contestada por el niño:

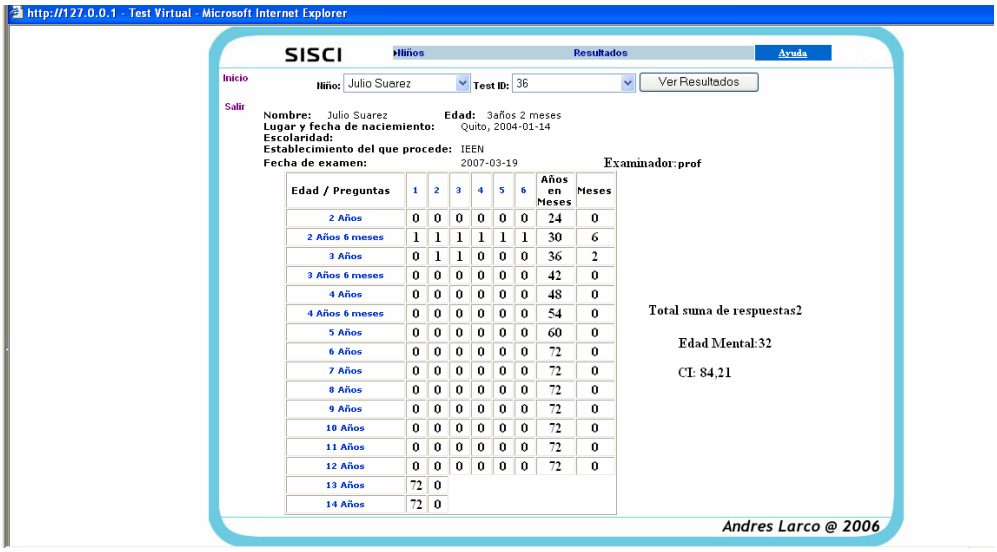
Que dice que el test termina cual el niño contesta máximo 2 preguntas d determinado nivel:



Pantallas con los resultados del test:

Para ver los resultados seguimos el siguiente procedimiento:

1. Se selecciona el niño.
2. Se selecciona el Id del Test.
3. Se da un clic sobre el botón ver resultados.



Edad / Preguntas	1	2	3	4	5	6	Años en Meses	Meses
2 Años	0	0	0	0	0	0	24	0
2 Años 6 meses	1	1	1	1	1	1	30	6
3 Años	0	1	1	0	0	0	36	2
3 Años 6 meses	0	0	0	0	0	0	42	0
4 Años	0	0	0	0	0	0	48	0
4 Años 6 meses	0	0	0	0	0	0	54	0
5 Años	0	0	0	0	0	0	60	0
6 Años	0	0	0	0	0	0	72	0
7 Años	0	0	0	0	0	0	72	0
8 Años	0	0	0	0	0	0	72	0
9 Años	0	0	0	0	0	0	72	0
10 Años	0	0	0	0	0	0	72	0
11 Años	0	0	0	0	0	0	72	0
12 Años	0	0	0	0	0	0	72	0
13 Años	72	0						
14 Años	72	0						

Anexo 3

Probe

Para el trabajo de PSP se tiene un archivo en Excel el cual es el libro de trabajo proporcionado por el SEI y realizado por Watts S. Humphrey, este libro de trabajo se ahorra el trabajo de realizar todos los cálculos descritos anteriormente.

A continuación se encuentran las instrucciones del libro del trabajo:

Student Workbook Instructions
The student workbook provides automated support for the Personal Software Process course. The workbook includes worksheets for the calculation-intensive forms and analysis charts and tables required for reports R3, R4, and R5.
The workbook includes:
Student - a worksheet for the student background survey
Summary - a worksheet containing PSP Project Plan Summary forms for each assignment, 1A to 8A
SizeEstimate - a worksheet contining a PSP Size Estimating Template for each assignment 3A to 8A
PROBE - a worksheet that automates the PROBE method computations
TimeLog - a worksheet for recording all time log data
DefectLog - a worksheet for recording all defect data
R3 - a worksheet that generates the defect analysis tables for the R3 report
R4 - a worksheet that generates the defect analysis tables for the R4 and R5 report
Pareto - a worksheet that generates pareto distributions of defects by type and by fix time
Defect, Plan, Process, and **Quality** analysis worksheets that include analyses required for the R4 and R5 reports
General Instructions
1. During the planning phase of an assignment, select the assignment number using the project selector on the Summary worksheet. Calculate and enter data in all the highlighted cells containing ????? in the Plan column of the Summary worksheet for that assignment.

For assignments 3A through 8A the SizeEstimate worksheet and the PROBE worksheet are used to generate size and time estimates. The SizeEstimate worksheet replaces the PSP Size Estimating worksheet. The planned and actual LOC are entered on this form (e.g. Base LOC, Deleted LOC, Modified LOC, Base Additions, New Object LOC, and Reused LOC).
The PROBE worksheet is used to select the estimating method, regression parameters, prediction intervals, and size and time estimates. The estimate is then transferred to the SizeEstimate worksheet.
2. During the development phases, use the time and defect logs to capture actual time and defect data for the assignment, or record this data on paper forms and enter during postmortem.
3. During the postmortem phase of an assignment, calculate and enter data in all the highlighted cells that contain ????? in the Actual, ToDate, or ToDate% columns for the assignment. For assignments 3A to 8A, measure and enter actual LOC on the SizeEstimate worksheet.
4. To print a Summary report, select the assignment with the project selector and print. For assignments 1A to 6A, an extra blank page will be printed. Print page 1 only to avoid printing the blank page.
Application Notes:
A. Only enter data in highlighted fields containing ?????.
B. All other fields contain formulas that are calculated by Excel and are locked to prevent accidental entry.
C. If you need to modify a protected cell you may turn cell protection off.
D. After your modifications are complete turn cell protection back.
E. Excel will request a password when you turn protection on. DO NOT ENTER A PASSWORD.
F.The Summary worksheet automates many of the PSP calculations, but as new PSP measures appear on the worksheet, they must be calculated and entered manually.

Antes de empezar con el trabajo se revisa información sobre la experiencia del estudiante que va ha realizar el trabajo.

PSP Student Background Survey				
		Student Name		Student #
		Andrés Larco		1
		Instructor Name(s)		
		Carlos Montenegro		
1,0		Which of the following best describes your employment		

		status in the software industry (mark one with an X)		
		Executive		
		Management		
		Senior technical		
	x	Technical		
	x	Full-time support		
		Other		
2,0		What is your software experience in: (Please specify for each category, rounded to the nearest year)		
	YRS.			
	2	In your present organization?		
	1	In your present position?		
	2	Your overall software experience?		
	2	Software requirements?		
	1	Software design?		
	1	Code and unit test?		
		Test and integration?		
		Software quality assurance?		
		Configuration management?		
	1	Software process improvement/quality management?		
3,0		About how much of your work-related time have you spent over the past year on each of the following? Please specify an approximate percentage for each, rounded up to the nearest whole number (with no % sign). The categories overlap, so your estimates need not total 100%		
	20	Software requirements?		
	20	Software design?		
	10	Code and unit test?		
	10	Test and integration?		
	30	Software quality assurance?		
		Configuration management?		
	10	Software process improvement/quality management?		
		Other		
4,0		What programming languages have you used? (Please mark as many as apply with an X)		
		Ada		
		Basic		
	x	C		
		C++		

		Java		
		PASCAL		
		OBJECT PASCAL		
	x	Visual Basic		
		Visual C++		
	x	OTHER		
4,1		How many high level languages do you know well? (Please specify)		
	4	Number of languages		
5,0		What language will you use in this course?		
		Ada		
		Basic		
		C		
		C++		
		Java		
		PASCAL		
		OBJECT PASCAL		
		Visual Basic		
		Visual C++		
	x	OTHER		
5,1		Approximately how much code have you written in this language? (please specify, rounded to the nearest 1,000 lines of code, e.g. 10K)		
	2	Lines of Code		
5,2		Approximately how much code have you written in ***all*** languages? (please specify, rounded to the nearest 1,000 lines of code, e.g. 10K)		
	5	Lines of Code		
5,3		Approximately how much code have you written in the last year? (please specify, rounded to the nearest 1,000 lines of code, e.g. 10K)		
	2	Lines of Code - In the language you will be using		
	5	Lines of Code - In all languages		
6,0		What is the highest degree that you have attained? (please specify)		

	3	Highest degree		
6,1		What was your major field of study? (please specify)		
	Ingeniería en Sistemas			
6,2		Have you had courses in any of the following subjects? (Please mark all that apply with an X)		
	x	Statistics		
	x	Physical Sciences		
	x	Software Project Management		
	x	Formal Software Methods		

A continuación se muestran las pantallas de los resúmenes:

PSP0 Project Plan Summary - Program 1

Student	1-Andrés Larco				Date		########	
Program	Access				Program#		1	
Instructor	Carlos Montenegro				Language		**PHP**	
		Plan		**Actual**	**To Date**		**To Date%**	
Time in Phase (min.)								
Planning				30	30		16,7	
Design				40	40		22,2	
Code				50	50		27,8	
Compile				20	20		11,1	
Test				20	20		11,1	
Postmortem				20	20		11,1	
Total		**180,0**		180	180		100,0	
Defects Injected								
Planning				0	0		0,0	
Design				0	0		0,0	
Code				1	1		50,0	
Compile				1	1		50,0	
Test				0	0		0,0	
Total Development				2	2		100,0	
Defects Removed								
Planning				0	0		0,0	

Design				0		0		0,0		
Code				1		1		50,0		
Compile				1		1		50,0		
Test				0		0		0,0		
Total Development				2		2		100,0		
After Development				**3**		**5**				

Esta pantalla muestra el selector del método probe el cual nos ayuda con los cálculos que realizamos para el proyecto:

PROBE Method Selector	
Project	1
Estimate (E)	0
Size Method Selector	3
Size Estimate	0
Range	0
B0	0
B1	0,75
R^2	0
Time Method Selector	3
Time Estimate	0
Range	0
B0	0
B1	9
R^2	0

En la siguiente pantalla se muestra el Log de tiempo para este primer proyecto:

Project	Phase	Date	Start	Int.	Stop	Delta
1	PLAN	01/15/07	12:03:52		12:33:52	30,0
1	DLD	01/15/07	13:00:00		13:40:00	40,0
1	CODE	01/15/07	13:40:00		14:30:00	50,0
1	COMPILE	01/15/07	14:30:00		14:50:00	20,0
1	PM	01/15/07	15:00:00		15:30:00	30,0

En la siguiente pantalla se muestra el Log de defectos para este primer proyecto:

Project	Date	num.	Type	Injected	Removed	FixTime	Fix Ref.
1	########	1	DOC	CODE	CODE	1,0	1
1	########	2	DOC	COMPILE	COMPILE	1,0	1

Para el ejercicio 8 se tiene el siguiente tabla de resumen:

PSP3 Project Plan Summary - Program 8								
Student	1-Andrés Larco				Date		########	
Program	Resultados				Program#		8	
Instructor	Carlos Montenegro				Language		PHP	
		Plan		**Actual**	**To Date**		**To Date%**	
Summary								
LOC/Hour		7,5		3,2	3,2			
Planned Time		400			600			
Actual Time				400	580			
CPI (Planned/Actual Time)					1,03			
% Reused		19,2		12,5	12,5			
% New Reused		0,0		0,0	0,0			
Test Defects/KLOC		0,00		0,00	0,00			
Total Defects/KLOC		0,00		428,57	428,57			
Yield %		100,0		100,0	100,0			
% Appraisal COQ		0,0		12,5	12,5			
% Failure COQ		0,0		13,8	13,8			
COQ A/F Ratio		0,00		0,91	0,91			
Program Size (LOC)								
Base(B)		**600**		**514**				
Deleted(D)		**10**		**5**				
Modified(M)		**10**		**5**				

Added(A)	40	16		
Reused(R)	**150**	**75**	75	
Total N&C (N)	**50**	21	21	
Total LOC(T)	780	**600**	600	
Total New Reused	**0**	**0**	0	
Total Object LOC(E)	**50**	**45**	45	
UPI (70%)				
LPI (70%)				
Time in Phase (min.)				
Planning	0	25	25	6,3
Design	0	100	100	25,0
Design Review	0	25	25	6,3
Code	0	150	150	37,5
Code Review	0	25	25	6,3
Compile	0	30	30	7,5
Test	0	25	25	6,3
Postmortem	0	20	20	5,0
Total	**400**	400	400	100,0
UPI (70%)				
LPI (70%)				
	Plan	**Actual**	**To Date**	**To Date%**
Defects Injected				
Planning	0,0	1	1	11,1
Design	0,0	1	1	11,1
Design Review	0,0	0	0	0,0
Code	0,0	3	3	33,3
Code Review	0,0	3	3	33,3
Compile	0,0	1	1	11,1
Test	0,0	0	0	0,0
Total Development	0,0	9	9	100,0
Defects Removed				
Planning	0,0	1	1	11,1
Design	0,0	1	1	11,1
Design Review	0,0	0	0	0,0
Code	0,0	3	3	33,3
Code Review	0,0	3	3	33,3
Compile	0,0	1	1	11,1
Test	0,0	0	0	0,0
Total Development	0,0	9	9	100,0
After Development		**10**	**10**	
Defect Removal Efficiency				

Def/Hr - DLDR	0,00		0,00		0,00					
Def/Hr - CDR	0,00		7,20		7,20					
Def/Hr - Compile	0,00		2,00		2,00					
Def/Hr - Test	0,00		0,00		0,00					
DRL(DLDR/UT)	0,00		0,00		0,00					
DRL(CDR/UT)	0,00		0,00		0,00					
DRL(Compile/UT)	0,00		0,00		0,00					

En la siguiente pantalla se muestra la estimación de tamaño:

Size Estimating Template										
Student		Andrés Larco					Date			23/01/2007
Instructor		Carlos Montenegro					Program#			8
BASE PROGRAM LOC							ESTIMATE			ACTUAL
BASE SIZE (B) =>							**600**			**514**
LOC DELETED (D) =>							**10**			**5**
LOC MODIFIED (M) =>							**10**			**5**
OBJECT LOC										
BASE ADDITIONS:		TYPE		METHODS		REL. SIZE	LOC			LOC
control		**logic**		**5**		**medium**	**10,0**			**10**
(BA) subtotal from page 2							10,0			10
TOTAL BASE ADDITIONS (BA)							20,0			20
NEW OBJECTS:		TYPE		METHODS		REL. SIZE	LOC		*	LOC
control		**logic**		**4**		**medium**	**10,0**			**10**
(NO) subtotal from page 2							10,0			10
TOTAL NEW OBJECTS							20,0			20

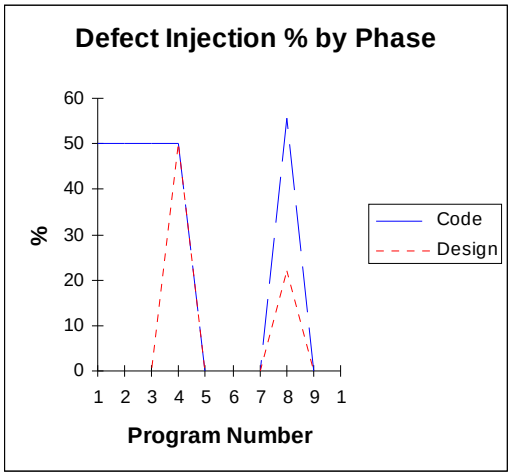
(NO)		
REUSED OBJECTS	LOC	LOC
clases	**100,0**	**50**
(R) subtotal from page 2	50,0	25
REUSED TOTAL (R)	150,0	75
	Size	Time
Estimated Object LOC:	E = BA+NO+M	50	
Regression Parameter:	B_0	0,00	0,00
Regression Parameter:	B_1	1,00	8,00
Estimated New and Changed LOC:	N = B_0 + B_1` * E	50,0	
Estimated Total LOC:	T = N + B - D - M + R	780,0	
Estimated Total New Reuse (sum of * LOC):		0	
Estimated Total Developent Time:	Time = B_0 + B_1` * E		400,0
Prediction Range:	Range	0,0	0,0
Upper Prediction Interval:	UPI = N + Range		
Lower Prediction Interval:	LPI = N - Range		
Prediction Interval Percent		70%	70%
Method Selected		D	D
R^2		0,00	0,00

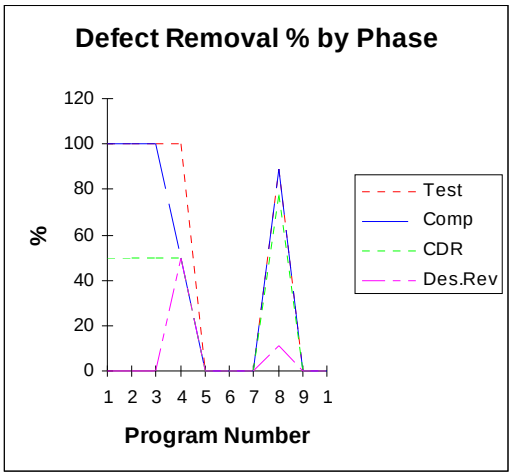
En la siguiente pantalla se muestra el método probe:

PROBE Method Selector	
Project	8
Estimate (E)	50
Size Method Selector	2
Size Estimate	35,25
Range	10,479708
B0	1,5
B1	0,675
R^2	0,84375

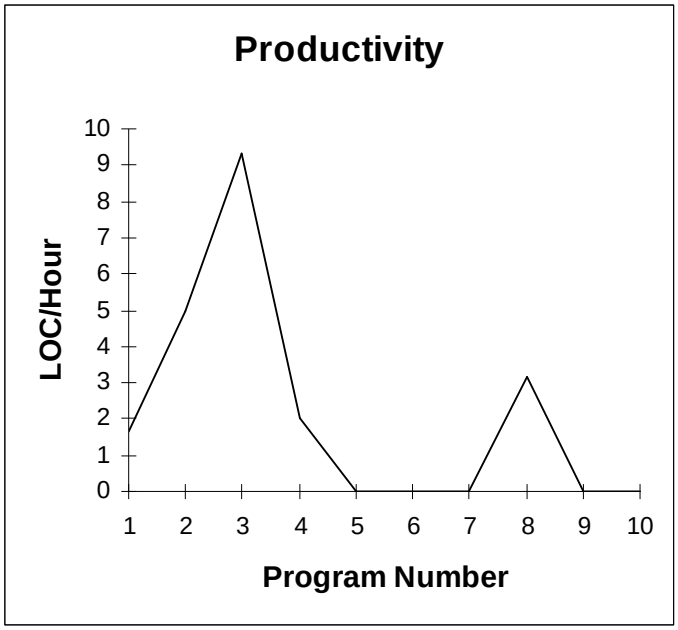
Time Method Selector	C 3
Time Estimate	1625
Range	0
B0	0
B1	32,5
R^2	0

A continuación se muestra gráficos de los defectos que han sido inyectados y removidos de acuerdo a cada fase:

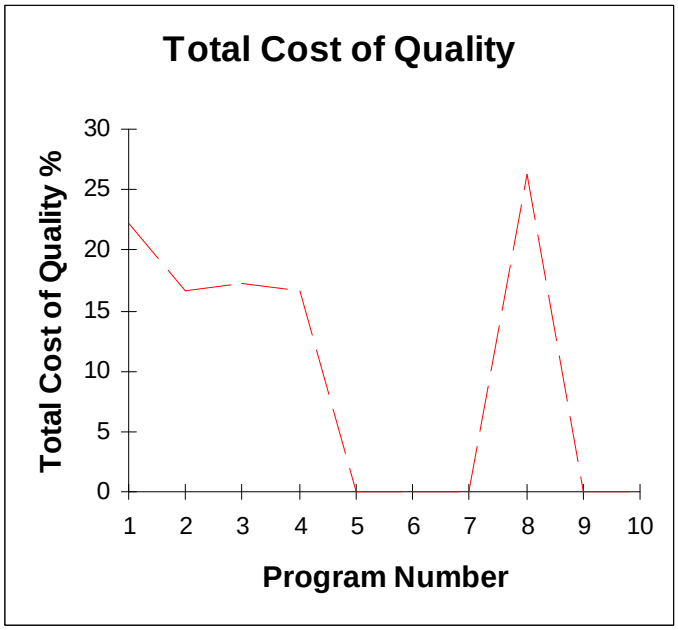




En esta gráfico muestra la productividad:



En el siguiente se muestra el costo de la calidad:



Anexo 4

Métodos estadísticos utilizados en el PSP

Uso de la media y desviación estándar en el PSP

La media y la desviación estándar son usadas para dividir el tamaño histórico de los datos con respecto a categorías y rangos de tamaño.

Calculando de la media y desviación estándar

La formula para calcular la media es:

$$x_{avg} = \frac{\sum_{i=1}^{n} x_i}{n}$$

La formula para calcular la desviación estándar es σ

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left(x_i - x_{avg}\right)^2}{n-1}}$$

Donde

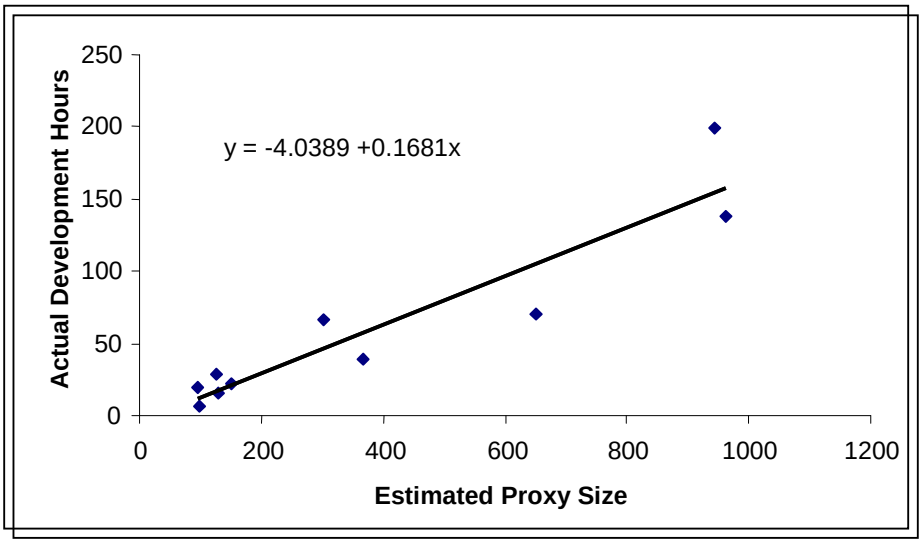
- Σ es el símbolo de sumatoria.
- *i* es el índice
- *x* es el dato de un grupo de datos
- *n* es el número de datos

Vista rápida

La regresión lineal sirve para calcular tamaños estimados. La regresión lineal esta representada por la siguiente ecuación de una recta:

$$y = \beta_0 + \beta_1 x$$

En la figura 1 se muestra la mejor regresión utilizando los siguientes parámetros β_0 = -4.0389 y β_1 = 0.1681.



Usando regresión en el PSP

El método PROBE del PSP usa los parámetros de la regresión para realizar las predicciones de tamaño y tiempo basados en los datos históricos.

El método PROBE tiene dos métodos el A y el B:

PROBE A, tamaño **estimado proxy** usando valores de x.

PROBE B, tamaño añadido y modificado usando valores de x.

Datos Históricos Usados		***x* valores**	***y* valores**
Estimación del Tamaño	PROBE A	Tamaño Estimado Proxy	Tamaño Actual Añadido y Modificado
	PROBE B	Tamaño Planeado Añadido y Modificado	Tamaño Actual Añadido y Modificado
Estimación del Tiempo	PROBE A	Tiempo Estimado Proxy	Tiempo Actual de Desarrollo
	PROBE B	Tiempo Planeado Añadido y Modificado	Tiempo Actual de Desarrollo

Calculando la regresión

Las formulas para calcular los parámetros de la regresión β_0 y β_1 son

$$\beta_1 = \frac{\left(\sum_{i=1}^{n} x_i y_i\right) - \left(n x_{avg} y_{avg}\right)}{\left(\sum_{i=1}^{n} x_i^2\right) - \left(n x_{avg}^2\right)}$$

$$\beta_0 = y_{avg} - \beta_1 x_{avg}$$

donde

- Σ es el símbolo de sumatoria
- *i* es el índice de n números
- *x* y *y* son dos pares de datos de un grupo de datos.
- *n* es el número de datos de *x* y *y*
- x_{avg} es el promedio de los valores de *x*
- y_{avg} es el promedio de los valores de *y*

Vista rápida

El cálculo de la correlación determina la relación entre dos grupos de datos numéricos.

La correlación $r_{x,y}$ tiene rangos de +1 a -1.

- Un resultado de +1 implica que la relación es positiva; donde *x* y *y* se incrementan.
- Un resultado -1 implica que la relación es negativa; donde *x* incrementa, *y* decrece.
- Un resultado de 0 implica que no hay relación.

Usando la correlación en el PSP

La correlación se usa en el PSP para juzgar la calidad de la relación lineal en varios datos del proceso históricos que se usan para planear. Por ejemplo, las relaciones entre el tamaño estimado Proxy y tiempo real o el tamaño planeado agregado y modificado y el tiempo real.

Para este propósito, nosotros examinamos el valor del r_{xy} de la relación cuadrado r^2.

si r^2 es	La relación es
$.9 \le r^2$	predictivo; úselo con confianza alta
$.7 \le r^2 < .9$	muy bien y puede usarse por planear
$.5 \le r^2 < .7$	adecuado para planear pero usar con cautela
$r^2 < .5$	no fiable para planear propósitos

Calculando la correlación

La formula para calcular el coeficiente de correlación $r_{x,y}$ es

$$r_{x,y} = \frac{n\left(\sum_{i=1}^{n} x_i y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{\left[n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\left(\sum_{i=1}^{n} y_i^2\right) - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$

donde

- Σ es el símbolo de sumatoria
- *i* es el índice de n números
- *x* y *y* son dos pares de datos de un grupo de datos.
- *n* es el número de datos de *x* y *y*