



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

**IMPLEMENTACIÓN DEL ESTÁNDAR ISO/IEC 29110 PERFIL BÁSICO
PARA EL DESARROLLO DE SOFTWARE DEL TIPO CIENTÍFICO**

T E S I S
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:
E R N E S T O R O D R Í G U E Z A L B A R R Á N

Director de Tesis:
M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ
Facultad de Ciencias, UNAM

Ciudad Universitaria, CD. MX. Febrero, 2017

Dedicatoria

A Sara una madre incansable, el combustible de mi esfuerzo.

A mis hermanos.

A mis amigos.

Agradecimientos

Agradezco a mi tutora la M. en C. María Guadalupe Elena Ibargüengoitia González por su tiempo, sus consejos, su paciencia, su confianza, sus cursos, pero sobre todo por transmitirme una pequeña parte de su saber, su experiencia y dedicación son dignas de admiración.

Agradezco a mis sinodales, Luis Miguel de la Cruz Salas, Gustavo Arturo Márquez Flores, Hanna Jadwiga Oktaba, Miguel Ehécatl Morales Trujillo, por el tiempo que dedicaron en revisar mi trabajo y por todos sus consejos, y conocimientos compartidos.

A mi familia, especialmente a mis padres Sara y Ernesto que han confiado siempre en mis decisiones y me han brindado siempre su apoyo, a mis hermanos Alberto y Yaret que la vida nos lleve siempre juntos.

Agradezco a Ingrid Mireya Negrete el estar conmigo siempre en las buenas y en las malas, en considerarme en todos sus proyectos y siempre hacer espacio en sus actividades para ayudarme en la revisión de mi trabajo, gracias por escucharme en esos momentos de mayor necesidad.

A Mario Arturo un amigo que me brindo su amistad en el Posgrado y siempre me apoyo en esos momentos de mayor necesidad, gracias por tu apoyo, sin tu ayuda este trabajo no hubiera sido posible. A mis amigos y compañeros Alex, Aurelio, Octavio, Heriberto, Juan Carlos, Karen, Marco y Paulo mi admiración para ustedes y mis mejores deseos.

A mis amigos del Instituto Politécnico Nacional Gerardo, Christian, Elder, Claudia, Heira y Mara, una amistad construida y solidificada por las experiencias de la vida, no importa que experiencia mala o buena nos espere se que siempre estaremos juntos.

A Lulú, Amalia y Cecilia por realizar un excelente trabajo y apoyar a todo alumno en el Posgrado. Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme otorgado una beca para realizar mis estudios.

Finalmente al Posgrado en Ciencia e Ingeniería de la Computación y a la Universidad Nacional Autónoma de México por creer en mi y darme una educación gratuita y de calidad.

¡Gracias a todos!

Índice general

Índice de tablas	9
Índice de abreviaturas	11
Introducción	13
1. Cómputo Científico	19
1.1. Desarrollo de Software Científico	20
1.2. Modelado	22
1.2.1. Modelo Conceptual	23
1.2.2. Modelo Matemático	24
1.2.3. Modelo Numérico	24
1.3. Validación	26
1.4. Buenos Programas	26
1.5. Comunidad científica y la Ingeniería de Software	27
1.6. ISO/IEC 29110 Perfil Básico	29
2. Guía de uso de la Norma ISO/IEC 29110 Perfil Básico para el desarrollo de Software Científico	37
2.1. Proceso de Administración del Proyecto	37
2.1.1. Actividades del Proceso de Administración del Proyecto	39
2.2. Proceso de Implementación de Software	48
2.2.1. Actividades del Proceso de Implementación de Software	51
3. Caso de estudio	71
3.1. Introducción	71
3.2. Objetivo.	71
3.3. Metodología	71
3.4. Proyectos de investigación	72
3.5. Proceso de desarrollo de software	73
3.6. Proceso de Administración del Proyecto según la Guía.	74
3.7. Problemas de la Administración de Software según la Guía.	77
3.8. Implementación de Software según la Guía.	77

3.9. Prueba de Laplace en dos dimensiones	82
3.9.1. Descripción de la prueba	82
3.9.2. Descripción de Resultados	82
3.9.3. Observaciones	83
3.10. Prueba de Laplace en un semicírculo	88
3.10.1. Descripción	88
3.10.2. Resultados	89
3.10.3. Observaciones	89
3.11. Problemas durante la implementación de Software según la Guía.	91
Conclusiones	93
Apéndice	96
A. Lista de Plantillas del Proceso de Administración del Proyecto	97
A.1. Plantilla para el Plan del Proyecto	97
A.2. Plantilla para el Reporte de Seguimiento	101
A.3. Plantilla para la Solicitud de Cambio	104
A.4. Plantilla para el Cierre	105
B. Lista de Plantillas Implementación de Software	107
B.1. Plantilla para la Especificación de Requerimientos	107
B.2. Plantilla para la Arquitectura y Diseño Detallado de Software	109
B.3. Plantilla para las Pruebas de Integración	110
B.4. Plantilla para las Pruebas Benchmark	111
C. Plantillas del caso de estudio.	113
C.1. Plan del Proyecto iteración inicial	113
C.2. Plan del Proyecto	118
C.3. Reporte de Seguimiento	120
C.4. Cierre del Plan del Proyecto	124
C.5. Especificación de Requerimientos	126
C.6. Arquitectura y Diseño Detallado de Software	130
C.7. Pruebas	138

Índice de tablas

1.1. Roles Involucrados de AP.	31
1.2. Roles Involucrados de IS.	33
1.3. Roles definidos para la realización de actividades y tareas propuestas por la norma ISO/IEC 29110 Perfil Básico	34
2.1. Elementos excluidos de las tareas a realizar en la actividad de planeación del proyecto.	39
2.2. Lista de tareas de la actividad Planeación del Proyecto.	40
2.3. Lista de tareas de la actividad de Ejecución del Plan del Proyecto.	43
2.4. Lista de tareas de la actividad de Evaluación y Control del Proyecto.	45
2.5. Lista de tareas de la actividad de Cierre del Proyecto	47
2.6. Tabla de Objetivo de cada una de las actividades propuestas del Proceso de Implementación de Software.	49
2.7. Lista de tareas de la actividad Antecedentes de Software	51
2.8. Lista de tareas de la actividad de Análisis de Modelos	52
2.9. Lista de tareas de la actividad de Arquitectura y Diseño Detallado de Software	56
2.10. Lista de tareas de la actividad Construcción del Modelo Computacional	59
2.11. Lista de Tareas de la Actividad Pruebas de Integración	62
2.12. Lista de Tareas de la Actividad Validación	67
2.13. Lista de Tareas de la Actividad Entrega del Producto.	69
3.1. Riesgos importantes durante la realización del proyecto.	75
A.1. Lista de objetivos a completar.	98
A.2. Lista de roles y responsabilidades de los miembros del equipo de trabajo.	98
A.3. Tabla de riesgos.	99
A.4. Tabla de la estrategia de control de versiones.	99
A.5. Tabla de las características y ubicación del repositorio.	100
A.6. Tabla de los Objetivos y actividades a realizar.	101
A.7. Tabla de tareas finalizadas.	101
A.8. Tabla de Productos.	102
A.9. Tabla que lleva el registro de cambios a realizar.	102
A.10. Tabla de riesgos nuevos y que se han manifestado.	102
A.11. Resumen de actividades.	103

A.12. Resumen de actividades Cierre.	105
A.13. Tabla de los Objetivos y actividades a realizar.	105
C.1. Plan del proyecto	114
C.2. Lista de roles y responsabilidades de los miembros del equipo de trabajo.	115
C.3. Objetivos de la iteración.	118
C.4. Tareas a ser realizadas en la iteración.	118
C.5. Cambios a realizar durante la iteración.	118
C.6. Riesgos encontrados.	119
C.7. Estado actual del proyecto.	120
C.8. Avance de tareas.	121
C.9. Productos.	121
C.10. Cambios.	122
C.11. Riesgos encontrados.	123
C.12. Resumen.	123
C.13. Resumen	124
C.14. Reporte de Productos.	124
C.15. Ejecuciones para distintos tamaños de una matriz.	140

Índice de Acrónimos

PO: Pequeñas Organizaciones (Grupo de 25 o menos personas).

AP: Administración del Proyecto (Proceso establecido por la ISO/IEC 29110 Perfil Básico).

IS: Implementación de Software (Proceso establecido por la Norma ISO/IEC 29110 Perfil Básico)

AN: Analista (Rol establecido por la Norma ISO/IEC 29110 Perfil Básico).

CL: Cliente (Rol establecido por la Norma ISO/IEC 29110 Perfil Básico).

DIS: Diseñador.

PR: Programador (También hace referencia a desarrollador).

AP: Administrador del Proyecto.

LT: Lider Técnico.

ET: Equipo de Trabajo.

CASE: Ingeniería de Software Asistida por Computadora (Traducido del ingles).

CVFEM: Método del Elemento Finito del Volumen de Control (Traducido del ingles).

Introducción

Un Poco de Historia

En 1936, Alan Turing describe el funcionamiento de una computadora en una forma teórica con el objetivo de crear una base sólida para una definición del concepto de números computables. Las dificultades de naturaleza ingenieril que le siguieron a las construcciones de máquinas de cálculo fueron muchas y los diseñadores de equipo de cómputo deseaban que sus máquinas funcionaran en primer lugar.

Mucho antes de que la palabra software entrara en uso general, era necesario describir largas secuencias de cálculos mediante la escritura de resultados intermedios en papel, los cuales indicaban el curso del cálculo; un ejemplo es el bien conocido método de Horner para la división de polinomios de manera eficiente. Pero el software tal como se desarrolló a finales de los 40's e inicios de los 50's, viene a compensar las insuficiencias del hardware mediante el uso de funciones programadas que contenían características que no se habían considerado.

Heinz Zemanek se ha expresado así:

“El Software comenzó con la traducción de fórmulas algebraicas en código de máquina.”

En 1936 el ingeniero alemán Konrad Zuse construye la Z1, la primer computadora del mundo programable, a pesar de ciertos problemas de ingeniería mecánica, ésta tenía los fundamentos básicos de las máquinas actuales, la cual era una calculadora mecánica binaria operada con electricidad. Para el año 1945 describe el primer lenguaje de programación de alto nivel el “*Plankalkuel*”.

Pronto siguieron el Math-Matic y Fortran de Remington-Rand e IBM respectivamente. Todos estos avances fueron hechos antes de que la palabra “**Software**” entrara en amplio uso en los 60's [11].

Antecedentes

La historia muestra como ha evolucionado el desarrollo de software conforme la tecnología presenta sus avances, tenemos que hoy en día el software tiene muchas aplicaciones diferentes y estas

diferencias tienen sus propios problemas en el desarrollo, debido a que sus aplicaciones pueden diferir de un software a otro, tanto en criticidad como en complejidad. De todos los problemas surgen sus soluciones, para los problemas de software se desarrolló la disciplina de Ingeniería del Software, la cual trata este tipo de cuestiones. El tema que nos interesa retomar en este trabajo es sobre las prácticas que la Ingeniería de Software presenta, y su aplicación en particular para el desarrollo de software científico.

El software del tipo científico tiene diferencias del software del tipo comercial. Para entender estas diferencias se debe entrar completamente en el desarrollo de dichas aplicaciones. La revista de investigación “*Developing scientific software*” [22] presenta un ejemplo con algunas características del desarrollo de software científico, hace mención a las dificultades que puede tener la comunidad científica ante la necesidad de desarrollar herramientas de software en apoyo a las investigaciones realizadas, mientras que las herramientas deben basarse en diversas áreas resulta complicado contratar empresas de software externas para el desarrollo de dichas herramientas.

Resulta engorroso para los desarrolladores de software ingresar a un área científica para la cual no están especializados, debido a que las diferencias entre desarrollar un sistema de recursos humanos a uno que deba resolver simultáneamente un conjunto de ecuaciones diferenciales parciales tienen complejidades que van desde los usuarios finales hasta los algoritmos utilizados. Las dificultades que se pueden presentar a los desarrolladores que no cuentan con una formación en este tipo de áreas, son diversas y varían de un área científica a otra, no por eso queremos dar a entender que los proyectos que no son de carácter científico no sean complejos.

Por otro lado, el solicitar que el científico, como experto en el área de comprensión y análisis de las ciencias exactas, aporte los requerimientos del sistema o herramienta a desarrollar es otra de las complicaciones a las que se enfrentan ambas partes. Debido a esto, el científico termina siendo el propio desarrollador del software.

Un científico trata de llevar la teoría a sus límites, y para poder validar su trabajo muchas veces es necesario tener el apoyo que una computadora ofrece, si no existe una herramienta de software que cumpla con sus necesidades inmediatas, ésta deberá ser desarrollada; no encontramos en la literatura una metodología de desarrollo de software seguido en las aplicaciones del tipo científico, pero si encontramos un proceso de desarrollo de software científico, el cual no está bien definido como tal por la Ingeniería de Software, debido a que un proceso es un conjunto de actividades vinculadas y tareas bien definidas, con entradas de recursos y salidas de productos, además de establecer roles responsables de ejecutar dichas actividades y tareas [19].

La revista de IEEE titulada “A Software Chasm: Software Engineering and Scientific Computing” [14], menciona ciertos aspectos de como las prácticas de Ingeniería de Software y el cómputo científico están divididos por una brecha, la construcción de puentes que conecten las prácticas de Ingeniería de Software y el desarrollo de software científico resultan no ser siempre los más adecuados. Los científicos que desarrollan software hoy en día son de tres tipos:

- Científicos que se encuentran en la industria realizando aplicaciones de un dominio específico

o aplicaciones de su dominio de interés.

- Investigadores de instituciones académicas.
- Estudiantes, ingenieros o científicos que eventualmente se unirán a una de las dos anteriores.

Las características que tiene el desarrollo de software científico son en general las siguientes [3]:

- El dominio del problema es de alta complejidad.
- Los equipos son pequeños.
- Las interfaces de las aplicaciones son simples.
- Generalmente no siguen estándares ni de diseño o documentación alguna.
- Los cambios en los requerimientos pueden ser muy bruscos.
- El código generado no contiene la documentación necesaria, por lo cual resulta complejo su entendimiento y funcionamiento.

Debido a éstas y otras características resulta difícil planificar los proyectos, no se tiene un hábito de medición de tiempos dedicado a la codificación y definición de requerimientos y la complejidad del proyecto depende de muchos factores que no pueden ser controlados en su totalidad.

Existen muchas investigaciones y herramientas desarrolladas en la actualidad, algunas de éstas fueron rechazadas por la IEEE debido a que se está interesado en el proceso de desarrollo de estas herramientas o sistemas y no en la parte funcional [22], muchas de estas aplicaciones no cuentan con la documentación necesaria y el código fuente queda sujeto al desarrollador, resultando imposible para otros científicos el entendimiento de estos sistemas.

Derivado de lo anterior, se observa la oportunidad de aplicar y seguir lo estipulado por la norma ISO/IEC 29110 Perfil Básico. Este estándar fue generado para el uso de pequeñas organizaciones.

Objetivo

El objetivo de este trabajo es adaptar las tareas contenidas en la norma ISO/IEC 29110 Perfil Básico, para dirigirlas a la comunidad científica que desarrolla software y desean implementar un modelo de desarrollo de software en sus proyectos. Se busca que con ésto se pueda estandarizar un proceso especializado para el desarrollo de software científico.

Metodología

Para lograr la propuesta de dicha guía, fue necesario comprender el contexto de desarrollo de software científico, visualizar los problemas a los que se enfrentan los científicos durante el desarrollo, realizar una revisión de las investigaciones sobre las prácticas de Ingeniería de Software en el cómputo científico y poder conocer el estado del arte sobre las propuestas de procesos y metodologías enfocadas al desarrollo. Esta revisión tiene por objetivo visualizar los problemas que se tienen en este tipo de proyectos. Por lo cual resulta necesario para nuestros objetivos trabajar en proyectos de Computo Científico que será la base para adaptar las tareas de la norma ISO/IEC 29110 Perfil Básico, y validar la propuesta.

Para el desarrollo de la Guía, que tienen por objeto: La planificación, gestión y ejecución de proyectos de tipo científico, se consideraron los siguientes puntos:

1. Revisión de los procesos, actividades y tareas, marcadas por la norma ISO/IEC 29110 Perfil Básico.
2. Revisión del estado del arte de la aplicación de metodologías y procesos de la Ingeniería de Software utilizados por la comunidad científica.
3. Búsqueda de un proyecto de cómputo científico, con el fin de adaptar las tareas establecidas por la Norma ISO/IEC 29110 Perfil Básico.
4. Modificación de las tareas de las actividades de la Norma ISO/IEC 29110 Perfil Básico, para la aplicación en proyectos de cómputo científico generando plantillas.
5. Validación de los documentos propuestos en las actividades, modificación de las tareas y actividades de la Norma ISO/IEC 29110 Perfil Básico, con el tutor del trabajo presentado y desarrolladores de software científico.

Estructura de la tesis

El trabajo de investigación realizado se encuentra estructurado de la siguiente manera:

Capítulo 1: **Cómputo Científico.** Introduce las características de los elementos del cómputo científico para la construcción de sistemas, abarcando los modelos analíticos, matemáticos y numéricos, utilizados para la construcción de software. Se hace un breve resumen de la norma ISO/IEC 29110 Perfil Básico, donde los procesos más importantes de esta norma son: Administración del Proyecto e Implementación de Software, y basaremos la propuesta en las actividades y tareas establecidas en estos procesos.

Capítulo 2: **Guía de uso de la Norma ISO/IEC 29110 Perfil Básico para el desarrollo de Software Científico.** Presenta la Guía propuesta, la cual es una modificación de la norma ISO/IEC 29110 Perfil Básico, proponiendo una serie de tareas por cada actividad propuesta en los Procesos de

Administración de Software e Implementación de Software, así como los diagramas de flujo de cada proceso definido. Los documentos a ser generados durante la aplicación de esta propuesta es una modificación a la documentación realizada del trabajo de Miguel E. [19], los documentos se encuentran en los Apéndices A y B, estas plantillas están dirigidas a la comunidad científica que desarrolle software.

Capítulo 3: Caso de estudio. Se presenta un proyecto de software científico, el cual fue la validación de la Guía introducida en el capítulo 2, la documentación más importante generada durante el desarrollo de dicho proyecto es anexada tanto en este capítulo como en el Apéndice C.

Se finaliza con las conclusiones obtenidas del trabajo realizado y el trabajo a futuro que se ha visualizado para continuar con su posible desarrollo.

Capítulo 1

Cómputo Científico

Existe una cantidad desconcertante de equipos de cómputo en el mundo, los cuales controlan ciertas operaciones importantes e imprescindibles para la continua vida económica y social de la humanidad. Tales tareas ya sean gubernamentales, privadas, industriales o sistemas administrativos que registran el comportamiento de la economía de un país o que controlan la contabilidad de una empresa, son ejemplos de sistemas de software que en la actualidad manejan una o muchas actividades.

Los modelos matemáticos no pueden prescindir de los cálculos para obtener la solución de alguna situación física, debido a que la capacidad humana para hacer operaciones es limitada, la computadora es una de las herramientas imprescindibles para la realización de estas tareas.

Tenemos que las técnicas utilizadas para obtener las soluciones de un modelo matemático específico son parte de la zona general llamada *computación científica*, y el uso de estas técnicas para obtener una idea de los problemas científicos o de Ingeniería se denomina *Ciencia Computacional* o *Ingeniería Computacional*.

Hoy en día es difícil determinar que área Científica o de Ingeniería no utiliza computadoras para el modelado. Actualmente se calculan de forma rutinaria las trayectorias terrestres de los satélites para que sigan misiones planetarias. Para verificar si la estructura de una aeronave que saldrá de la atmósfera de la tierra es la óptima, es necesario simular antes el flujo de aire alrededor de ella. Este tipo de estudios han sido en extremo importantes para la industria aeroespacial en el diseño de naves espaciales, el modelado en la computadora ha ahorrado miles de millones de dolares en comparación con la construcción de prototipos. Para el diseño de automóviles y otros productos aplica la misma situación.

Los meteorólogos utilizan grandes cantidades de tiempo de cómputo para predecir el tiempo del día siguiente, además hacer las predicciones de los comportamientos de tormentas se necesitan de cálculos más largos y hacer aproximaciones, en su caso, afirmaciones del cambio del clima global.

La astronomía es otra área que hace un uso importante de la computación, la modelación reali-

zada nos ha permitido tener un conocimiento básico sobre los fenómenos del universo, la evolución de las gigantes rojas o enanas blancas han sido obtenidas de tales cálculos.

Ecologistas y biólogos utilizan la computación para calcular la dinámica de la población o modelar la relación depredador–presa, el flujo de la sangre en el cuerpo humano, y la dispersión de contaminantes en el océano y la atmósfera.

Los modelos matemáticos de todos estos problemas son sistemas de ecuaciones diferenciales parciales u ordinarias. Las ecuaciones diferenciales vienen en todos los tamaños y formas, aún en las actividades actuales las computadoras más potentes están lejos de ser capaces de resolver muchos de los problemas planteados por científicos e ingenieros, pero la computación científica y su alcance están cambiando rápidamente. Existen muchos modelos matemáticos y cada uno de ellos tiene sus propios retos. Las herramientas para cada uno de estos modelos resultan imprescindibles y han aumentado nuestra capacidad de hacer mediciones más rápido que nuestra capacidad para asimilarlos. En las zonas donde la Ingeniería encontraba dificultades para resolver problemas complejos aún en un equipo de cómputo, en la actualidad son problemas rutinarios que están siendo resueltos una y otra vez con los cambios en los parámetros de diseño. Esto ha dado lugar a un número cada vez mayor de sistemas de diseño asistido por computadora.

No tenemos los elementos necesarios para poder definir los límites que el *Cómputo Científico* tiene y la mayoría de las herramientas, técnicas y teorías desarrolladas tienen origen en las matemáticas, un gran porcentaje de éstos aparecieron antes que las computadoras electrónicas fueran desarrolladas. Este conjunto de teorías y técnicas matemáticas se llama *Análisis numérico* y constituye una parte importante de la *Computación Científica*. Con el nacimiento de la computación muchos métodos numéricos tuvieron que ser revisados y en su momento abandonados.

Por tanto, la computación científica se basa en las matemáticas y la computación para desarrollar soluciones idóneas a problemas de ciencia e ingeniería. La relación que guardan el cómputo científico y otras áreas puede ser observado en la figura 1.1, en el cual se muestra la relación que tiene el cómputo científico con otras áreas, además del desarrollo de formas de usar los sistemas de computo para la solución de problemas específicos [9].

1.1. Desarrollo de Software Científico

Existen problemas y soluciones para el desarrollo de *Software Científico*, en este apartado hablaremos de las características que se siguen al desarrollar este tipo de software, en algunos casos la *Ingeniería de Software* ha jugado un papel secundario en las aplicaciones que se construyen, ciertas actividades son realizadas de manera natural y obligatoria.

De igual forma hablaremos del desarrollo de software que se sigue en la construcción de sistemas de este tipo, seguiremos lo redactado en [25].

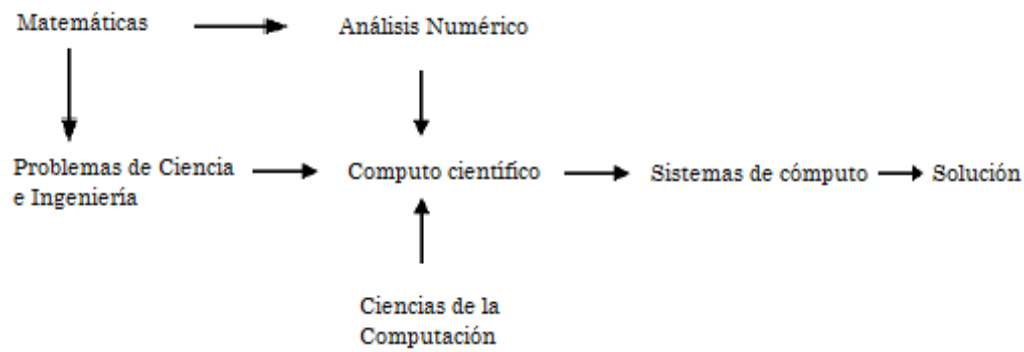


Figura 1.1: Cómputo científico. Tomada de *Gene. G and James M. Scientific Computing An introduction with Parallel Computing*,.

Cuando un problema de ciencia se resuelve con la ayuda de cálculos numéricos, el procedimiento de solución implica varios pasos:

1. Entender el problema y formular un modelo matemático.
2. Utilizar métodos numéricos para resolver problemas matemáticos.
3. Implementar los métodos numéricos en un programa de computadora.
4. Verificar que los resultados del programa son matemáticamente correctos.
5. Aplicar el programa al problema científico e interpretar los resultados.

Este proceso puede necesitar de ajustes en el modelo matemático planteado, los métodos numéricos y el programa de computadora, dependiendo de los resultados obtenidos. Este ciclo contiene una parte teórica y una parte práctica, en algunos casos se hace énfasis en la parte teórica, la cual deriva del análisis de los modelos matemáticos y los métodos numéricos. Por otra parte, la práctica consiste en la implementación de los modelos matemáticos y los métodos numéricos en un programa de computación ejecutable, el cual produce resultados numéricos que serán verificados.

El software que realiza cálculos científicos debe de ser:

1. Matemáticamente correcto.
2. Eficiente (rapidez de ejecución y mínimo uso de memoria).
3. Fácil de mantener y extender.

Si existe un error en el programa, los cálculos podrían ser incorrectos y el programa sería inútil. Por otro lado el método numérico implementado podría demandar días o semanas de tiempo de cálculo combinado con el uso de memoria. Por desgracia el realizar un cambio o una mejora de eficiencia fácilmente agrega errores al código. La complejidad del software científico ha alcanzado

el límite en su mantenimiento y las futuras mejoras se han vuelto complejas.

Por otra parte el desarrollador debería de contar con las siguientes habilidades:

1. Entender el problema matemático a ser resuelto.
2. Entender el método numérico a ser utilizado.
3. Diseñar un algoritmo y estructura de datos.
4. Seleccionar el lenguaje de programación ideal y sus herramientas.
5. Manejar el uso de bibliotecas.
6. Verificar que los resultados sean correctos.

En cualquier proyecto de software científico se tiene que los primeros dos puntos son la parte fundamental del desarrollo del sistema o la herramienta a generar, mientras que las siguientes son a veces olvidadas.

1.2. Modelado

La computación científica propone *modelos matemáticos* para estudiar los factores físicos que están relacionados con el problema, la formulación de un modelo matemático adecuado del problema en cuestión es uno de los primeros pasos.

En muchos problemas físicos, los factores se refieren a la relación de fuerzas y otras leyes de conservación de la física. Por ejemplo, la formulación del modelo de un problema de trayectoria está basado en la segunda Ley de Newton, la cual requiere que las fuerzas que actúan en un cuerpo sea igual a la tasa de cambio de la fuerza del cuerpo. Esta ley se debe especializar al problema en particular, por ejemplo, la fuerza gravitatoria de Júpiter ejerce una fuerza sobre un cohete en la atmósfera de la tierra pero esta fuerza es pequeña y puede ser despreciada. Otras fuerzas también pueden ser pequeñas en comparación con las dominantes pero sus efectos no pueden desecharse tan fácilmente.

La mayoría de los problemas físicos están modelados por ecuaciones diferenciales, en los cuales serán necesarias condiciones de frontera o iniciales. Por ejemplo, al estudiar el flujo de la sangre, una condición es que el flujo no pueda atravesar las paredes del recipiente que la contiene. En otros casos el contorno no puede ser tan evidente físicamente, pero lo que interesa es saber que el problema matemático tenga solución única, otras veces sucede que el modelo formulado tiene muchas soluciones, en este caso se tienen restricciones en la solución, un ejemplo es el pedir que la solución sea con energía mínima, es el caso en que los problemas seleccionados tengan solución, en otros casos la solución debe ser aproximada por algún método.

Uno de los procesos que hemos obtenido de la misma literatura y los cuales tienen características similares con algunos proyectos que serán mencionados posteriormente es el mostrado en la figura 1.2.

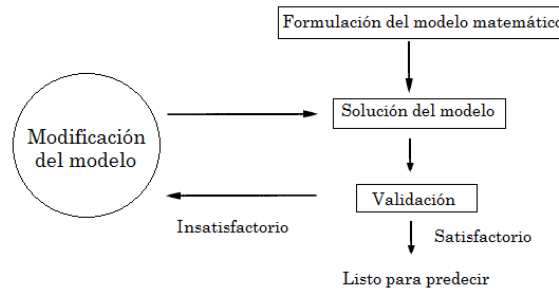


Figura 1.2: Proceso seguido para la construcción de programas de cómputo científico. Traducida de Gene. G and James M. *Scientific Computing An introduction with Parallel Computing*.

1.2.1. Modelo Conceptual

Una vez identificado el problema se puede recurrir a la generación de un *Modelo Conceptual*, el cual sirve de base para identificar las posibles soluciones que las restricciones permitan considerar factibles, la idea de proponer modelos responde a preguntas como el *¿Por qué?*, *¿Cómo?*, *¿Qué relación hay?*. Los modelos conceptuales buscan la descripción del problema en términos cualitativos del sistema, pero un modelo no puede incluir todos los aspectos de un sistema real, solamente se incluyen los más importantes, es decir, los objetivos y procesos físicos, símbolos y relaciones que constituyen el modelo, esto permite definir las leyes que las gobiernan permitiendo implantar un modelo matemático y modelos de solución (numérico).

Pero este modelo es la representación científica que se hace de un sistema bien definido, puede ser físico, químico, biológico, etc., cada uno de éstos con sus limitaciones y ventajas. Estos modelos, en general se desarrollan matemáticamente, aunque no se basan en las matemáticas. Las relaciones cuantitativas entre las magnitudes físicas son medidas perfectamente observadas, las cuales fueron o son determinadas por métodos experimentales. La naturaleza de este tipo de modelos es que son solo aproximados, y su interpretación difiere de las aplicaciones que se desean [2, 17].

Por ejemplo podemos comentar algunos de estos modelos que han sido la base de muchas aplicaciones en la actualidad [4]:

“Se tiene un gas en un contenedor”, el análisis cualitativo es que se tiene un conjunto de moléculas, las cuales tienen propiedades macroscópicas de los gases, estas partículas están en constante movimiento el cual produce energía, además se tiene que el gas ejerce una presión hacia afuera de las paredes del contenedor y el empuje total produce una presión. Tenemos que si se añade calor a este recipiente, entonces la energía cinética media de las moléculas aumentará así como la temperatura [20].

De este ejemplo hemos obtenido una descripción cualitativa de la ley de Boyle sin involucrar ningún cálculo, esto nos permite entender los fenómenos y hacer predicciones sobre el comportamiento de un problema de la teoría de gases.

“¿Cómo circula la sangre en el cuerpo humano?”, este es un problema gobernado por la ecuación de Bernoulli, al igual que una cañería, la sangre es bombeada por el corazón con un gradiente de presión que se establece a lo largo de la longitud del vaso sanguíneo. Si una arteria se estrecha, la velocidad de la sangre que circula por el estrechamiento aumenta, la sangre que circula por arterias estrechas puede ocasionar problemas, si la velocidad es lo bastante rápida puede hacerse turbulento y producir remolinos. Una turbulencia cerca del corazón produce soplos cardíacos con un sonido característico que los médicos reconocen [4].

Debemos de recordar que todo modelo generado es a base de asunciones y que este modelo no es en su totalidad exacto, habrá que hacer suposiciones que en la realidad no pueden ser aplicables pero estas suposiciones nos acercan a las situaciones reales.

1.2.2. Modelo Matemático

Debido a que todas las ecuaciones que gobiernan cada uno de los modelos conceptuales, deben de pasar de dichas ecuaciones a ecuaciones que se adapten al contexto de aplicación, no nos enfocaremos al trabajo matemático necesario para esta transformación. Pero identificado el fenómeno se pueden establecer las ecuaciones que describen matemáticamente al fenómeno, las ecuaciones de frontera y la variabilidad de la solución, así como las variables que intervienen y las hipótesis y restricciones que el problema contiene o debe respetar.

Cada una de las características y supuestos deben de ser considerados ya que afectan a las ecuaciones que describen al fenómeno, y esto puede determinar una ventaja o desventaja en la solución obtenida. También debe tenerse en cuenta la evolución del modelo planteado pues puede acumular errores de medición con lo cual se afecta a la solución [2].

1.2.3. Modelo Numérico

Los modelos que nos interesan son los numéricos, tenemos los modelos analíticos los cuales resuelven una ecuación en un punto específico y en un tiempo determinado, generalmente se desea saber el valor de la solución en la mayor cantidad posible de puntos del dominio establecido. Es en esta cuestión donde los modelos numéricos, que están representados por algoritmos o métodos numéricos y que son aplicados en muchos problemas, juegan un papel importante [5, 13, 27].

Los métodos numéricos que están enfocados en la solución de ecuaciones diferenciales deben discretizar el problema; estos métodos están diseñados para ser utilizados en conjunto con una computadora, ya que los cálculos a realizar pueden medirse en millones.

El uso de métodos numéricos implica no solo la discretización del problema, también podemos observar que se debe de tener un concepto del manejo de errores y la aritmética de computadoras.

Existen muchos métodos numéricos, están los métodos para aproximación de raíces los cuales son:

- Método de la bisección.
- Método de las aproximaciones sucesivas.
- Método de Newton.
- Método de la secante.
- Método de Steffensen.
- Método de la falsa posición.

También están los métodos diseñados para la solución de sistemas de ecuaciones, que se describen en:

- **Métodos exactos:** Son algoritmos finitos que permiten obtener la solución del sistema de manera directa.
- **Métodos aproximados:** Son algoritmos iterativos e infinitos para calcular las soluciones del sistema por aproximaciones sucesivas.

Entre los métodos directos están:

- Método de Gauss.
- Método de Gauss-Jordan

Entre los métodos iterativos se encuentran:

- Método de Richardson.
- Método de Jacobi.
- Método de Gauss-Seidel.

Los métodos numéricos son muy diversos, cada uno con sus aplicaciones y problemas, para un amplio conocimiento de ellos se puede consultar gran variedad de literatura especializada [4, 7, 18, 24].

1.3. Validación

Uno de los pasos ha seguir es la validación del modelo, es decir, someter a revisión los cálculos obtenidos para comprobar que cumplen con los requisitos de precisión, para esto es necesario tener una consistencia del manejo de errores en la solución numérica aunque también existe un error en el modelado.

Se debe de tener una verificación sobre los factores que puedan afectar de manera mínima a la solución, aunque también habrá que verificar si los factores desechados están bien justificados, hay que eliminar los errores de limitaciones del modelo, por ejemplo si se ha modelado la trayectoria de un cohete y este modelo se hizo bajo el supuesto de una altura máxima de 100 Km y la solución computarizada muestra alturas de 200 Km, existe un error obvio que puede ser corregido. Posteriormente que estos errores se han eliminado se puede comparar los resultados obtenidos con datos experimentales u observaciones disponibles, pero se debe tener cuidado de estos resultados, pues pueden diferir del modelo matemático propuesto, habrá que analizar las condiciones en que los resultados fueron obtenidos. Muchas veces la validación proviene de la experiencia e intuición del investigador y se obliga a tener un juicio humano en cuanto si los resultados obtenidos son o no aceptables y corresponden con los datos de observación.

Habrá que hacer una observación en este punto sobre la validación en términos del modelo mostrado previamente en la figura 1.2, las modificaciones pueden resultar en un replanteo completo del trabajo realizado, una vez que el modelo se modifica el ciclo inicia de nuevo. En cuanto el proceso de validación ha terminado, se puede utilizar para la predicción y suponemos que podemos responder a las preguntas que dieron nacimiento a nuestro modelo, el mundo físico es demasiado complicado y el conocimiento que se tiene de él puede ser limitado como para poder predecir el futuro a la perfección. No obstante se espera que las soluciones obtenidas aporten conocimientos importantes al problema de estudio.

1.4. Buenos Programas

El hablar de buenos programas puede depender de la perspectiva del diseñador pero en este caso hablaremos de buenos programas si cumplen ciertos criterios, los cuales son mencionados en [9].

Fiabilidad – El código no contiene errores y se puede confiar para realizar cálculos de lo que se supone debe calcular.

Robustez –Esta estrechamente relacionado con la fiabilidad. El código tiene una amplia gama de aplicaciones, así como la capacidad de detectar datos erróneos, “Singulares” u otros problemas que no se puede esperar manejar, así como situaciones anormales y que traten de manera satisfactoria para el usuario.

Portabilidad –El código puede ser transferido de una computadora a otra con un mínimo de esfuerzo sin perder fiabilidad. Por lo general, esto significa que el código ha sido escrito en un

lenguaje de alto nivel como FORTRAN y no utiliza “trucos” que dependen de las características de un equipo determinado. Cualquier característica de la máquina que deba ser utilizado, debe estar bien delineado.

Mantenibilidad –Cualquier código que necesite cambios de vez en cuando, ya sea para hacer correcciones o añadir mejoras, debería ser posible y con el mínimo esfuerzo.

El código debe ser escrito de una manera clara y sencilla para que tales cambios se puedan hacer fácilmente y con una probabilidad mínima de la inyección de nuevo errores. Una parte importante del mantenimiento es que haya una buena documentación del programa, de modo que se pueda modificar de manera eficiente por las personas que no escribieron originalmente el código. Una buena documentación también es importante para que el usuario del programa no solamente comprenda como usar el código, sino también sus limitaciones.

1.5. Comunidad científica y la Ingeniería de Software

La comunidad científica está inmersa en el desarrollo o uso de software hoy en día, podríamos asegurar que se ha formado una relación estrecha entre estas dos áreas de conocimiento y aplicación, gracias a la ciencia la humanidad se ha beneficiado con la tecnología y viceversa.

La comunidad científica necesita del uso del software para algunas de sus investigaciones, como ya hemos mencionado pueden utilizar software de terceros o generando ellos mismos sus propias herramientas, pero estamos interesados en como la comunidad científica visualiza las prácticas de Ingeniería de Software, para eso nos hemos basado en la encuesta realizada en [10], en la cual abarcaban un cierto número de preguntas;

- RQ1.** ¿Cómo aprendieron los científicos lo que saben sobre el desarrollo de software científico?
- RQ2.** ¿Cuándo los científicos aprendieron lo que saben sobre el desarrollo utilizando software científico?
- RQ3.** ¿Qué tan importante es el desarrollo / uso de software científico para los científicos?
- RQ4.** ¿Cuánto de su tiempo de trabajo los científicos dedican al desarrollo / uso de software científico?
- RQ5.** ¿Los científicos pasan más tiempo desarrollando / utilizando software científico que en el pasado?
- RQ6.** ¿En qué escala de hardware los científicos desarrollan / utilizan software científico?
- RQ7.** ¿Cuáles son los tamaños de las comunidades de usuarios de software científico?
- RQ8.** ¿Qué tan familiarizados están los científicos con conceptos de Ingeniería de Software?

RQ9. ¿El tamaño del programa, el tiempo dedicado a la programación o el tamaño del equipo influyen en las opiniones de los científicos sobre la importancia de las buenas prácticas de desarrollo de software?

En esta encuesta cada una de las preguntas abarcaba ciertas hipótesis a responder, de las cuales algunas dieron lugar a nuevos cuestionarios. Los participantes fueron investigadores de 40 países con edades entre los 18 a los 60 años, el 50 % de las respuestas fueron recibidas de Estados Unidos, Canadá, Reino Unido, Alemania y Noruega.

Los encuestados fueron académicos, estudiantes de posgrado, programadores, científicos gubernamentales, administradores de sistemas, técnicos de laboratorio y médicos.

Lo relevante para nosotros fueron las conclusiones obtenidas del trabajo realizado, sobre:

- *Prácticas de software.*

“El nivel de importancia que los científicos asignan a los conceptos de Ingeniería de Software es en su mayoría consistente con su comprensión de este concepto. No obstante, se encontró que, en particular para los conceptos de *pruebas de software* y *verificación de software*, los científicos asignan un nivel de importancia a estos conceptos superior al de su nivel de comprensión de estos conceptos.”

- *Pruebas de Software.*

“Las pruebas de software son particularmente difíciles para el software científico porque se sabe que las respuestas contienen errores matemáticos de aproximación de tamaño desconocido. Más específicamente, el desafío consiste en separar errores de software de errores de modelo y errores de aproximación. Además, la salida correcta al ejecutar un código de simulación es raramente conocida. Esto hace que los procedimientos de prueba en Ingeniería de Software (por ejemplo, pruebas de regresión y pruebas en caja negra) sean menos apropiados para software científico en muchas situaciones. Estos hechos pueden ser la razón por la cual los científicos se han centrado en las técnicas de prueba que se basan en la comprensión matemática del problema científico que se está resolviendo. Es muy común, tan pronto como la evidencia para un código correcto se proporciona, utilizar el resultado verificado en las pruebas de regresión. Sin embargo, postulamos que los científicos han reinventado en su mayoría este concepto en lugar de importar la técnica de la Ingeniería de Software . Por lo tanto, aunque debe ser beneficioso para enseñar a los científicos sobre las técnicas de pruebas de Ingeniería de Software , uno debe ser consciente de que las pruebas de software científico plantean problemas que aún no han sido lo suficientemente abordados por la comunidad de ingeniería de software. ”

Si bien la encuesta muestra que la comunidad científica tiene cierto interés en las prácticas de Ingeniería de Software, menciona que los equipos de desarrollo en grandes empresas podrían aumentar la importancia percibida de varias prácticas de software.

1.6. ISO/IEC 29110 Perfil Básico

Introducción

Las normas ISO internacionales aportan una contribución positiva al mundo en que vivimos, ya que facilitan el comercio, la difusión del conocimiento y diseminan los avances innovadores en tecnología [6].

“La capacidad de las organizaciones para competir, adaptarse y sobrevivir depende cada vez más del software”[15].

La siguiente sección esta dedicada a la norma ISO/IEC 29110 Perfil Básico, la información presentada fue obtenida de la Norma técnica peruana [1] la cual es una adopción de la versión en inglés de la Technical report ISO/IEC TR 29110-5-1-2 [12].

La norma ISO/IEC 29110 Perfil Básico esta dirigida a pequeñas organizaciones (PO) donde una PO es una entidad (empresa, organización, departamento o proyecto) conformada por hasta 25 personas, que realicen proyectos de desarrollo de software, esta norma pretende brindar a las (PO) en todo el mundo la posibilidad de estar mejor equipadas para el desarrollo de productos de software que satisfagan las expectativas de los clientes, en términos de funcionalidad, costo y cronogramas.

La mayoría de las normas ISO/IEC no se ajustan a las necesidades de las PO. Se ha identificado que las PO encuentran difícil relacionar los estándares internacionales con las necesidades de su negocio. La mayoría de las PO no pueden ampliar sus recursos en términos de tiempo, presupuesto y número de empleados, tampoco ven un beneficio neto en establecer procesos de ciclo de vida de software extensos.

El Perfil básico describe el desarrollo de software de una sola aplicación por un solo equipo de proyecto sin ningún riesgo especial o factores situacionales. La guía proporciona los procesos de *Administración del Proyecto* e *Implementación de Software*, estos procesos están intrínsecamente relacionados durante todo el ciclo de vida del proyecto, la figura 1.3 muestra un diagrama de las condiciones de entrada para la *Administración del Proyecto* y la salida esperada al final de la aplicación de la *Implementación de Software*.

Para el uso de la guía las PO necesitan cumplir con ciertas condiciones de entrada:

- El enunciado de trabajo del proyecto está documentado.
- La viabilidad del proyecto fue realizada antes de su inicio.
- Los bienes, servicios, infraestructura y el equipo del proyecto están asignados, entrenados y disponibles.

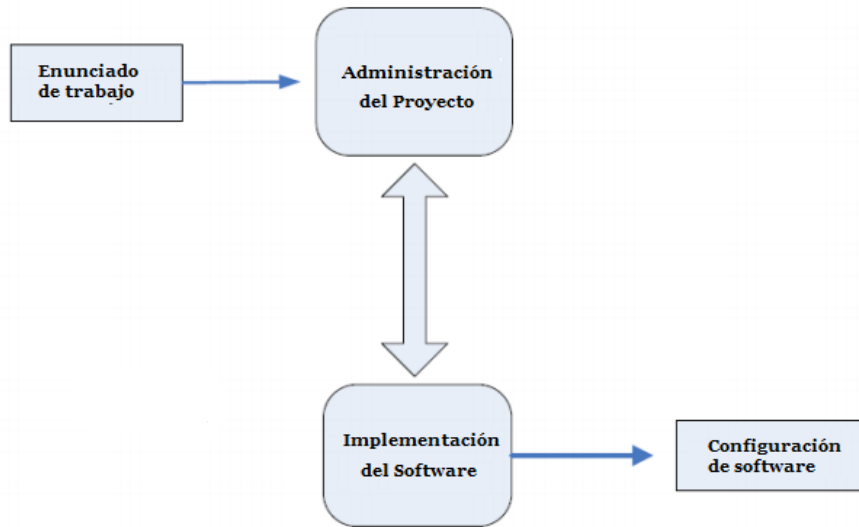


Figura 1.3: Procesos del Perfil Básico (modificado de *Abraham D. and Marcelo P. Factors driving the adoption of ISO/IEC 29110: a case study of small software enterprise, adaptada de [1, 12]*).

Proceso de Administración del Proyecto (AP)

El propósito del Proceso de Administración del Proyecto es establecer y llevar acabo de manera sistemática las tareas del Proceso de Implementación de Software, las cuales permiten cumplir con los objetivos del proyecto en calidad, tiempo y costo esperados.

El flujo de trabajo del Proceso de Administración del Proyecto está compuesto por cuatro actividades, las cuales proponen un total de 26 tareas a realizar, el flujo de trabajo se muestra en la figura 1.4.

Las cuatro actividades mencionadas anteriormente son las siguientes:

- AP.1 Planeación del Proyecto.
- AP.2 Ejecución del Plan del Proyecto.
- AP.3 Evaluación y Control del Proyecto.
- AP.4 Cierre del Proyecto.

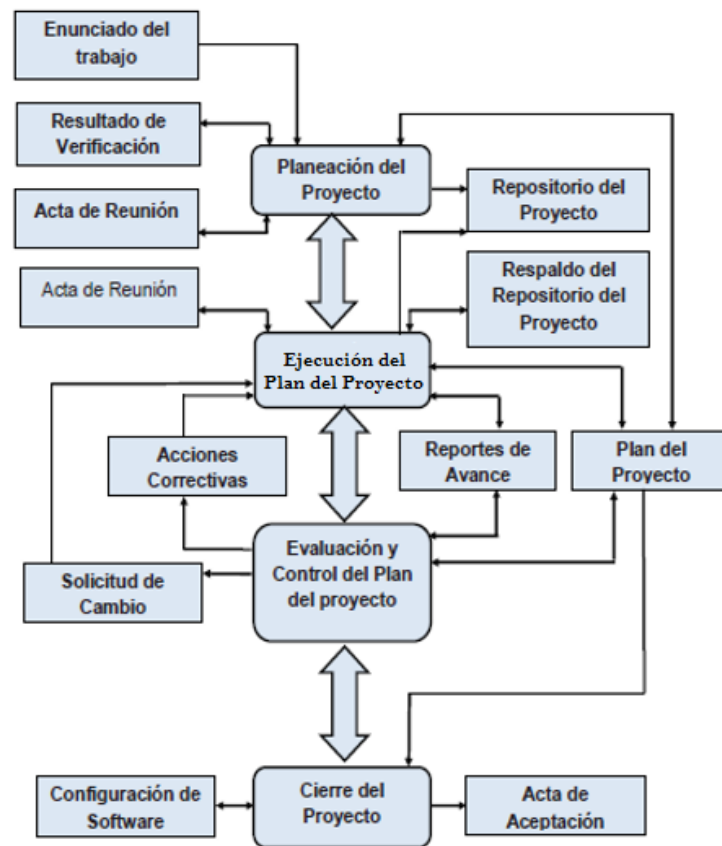


Figura 1.4: Proceso de la Administración de Proyectos. Tomada de *Abraham D. and Marcelo P. Factors driving the adoption of ISO/IEC 29110: a case study of small software enterprise, adaptada de [1, 12]*.

Durante el proceso AP se espera que los siguientes roles estén involucrados en las actividades y tareas establecidas en la figura 1.1.

Tabla 1.1: Roles Involucrados de AP.

Rol	Abreviatura
Cliente	CL
Administrador de Proyecto	AP
Líder Técnico	LT
Equipo de Trabajo	ET

Proceso de Implementación de Software (IS)

El propósito del proceso de Implementación de Software es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas para los productos de Software, nuevos o modificados, de acuerdo a los requisitos especificados. El Proceso de la Implementación de Software consiste en la realización de 43 tareas y 6 actividades, las actividades del proceso de IS son las siguientes:

- IS.1 Inicio de la Implementación de Software.
- IS.2 Análisis de Requisitos del Software.
- IS.3 Arquitectura y Diseño Detallado del Software.
- IS.4 Construcción del Software.
- IS.5 Integración y Pruebas del Software.
- IS.6 Entrega del Producto.

La figura 1.5 muestra el flujo de información entre las actividades del proceso Implementación de Software incluyendo los productos de trabajo más relevantes y la relación entre ellos.

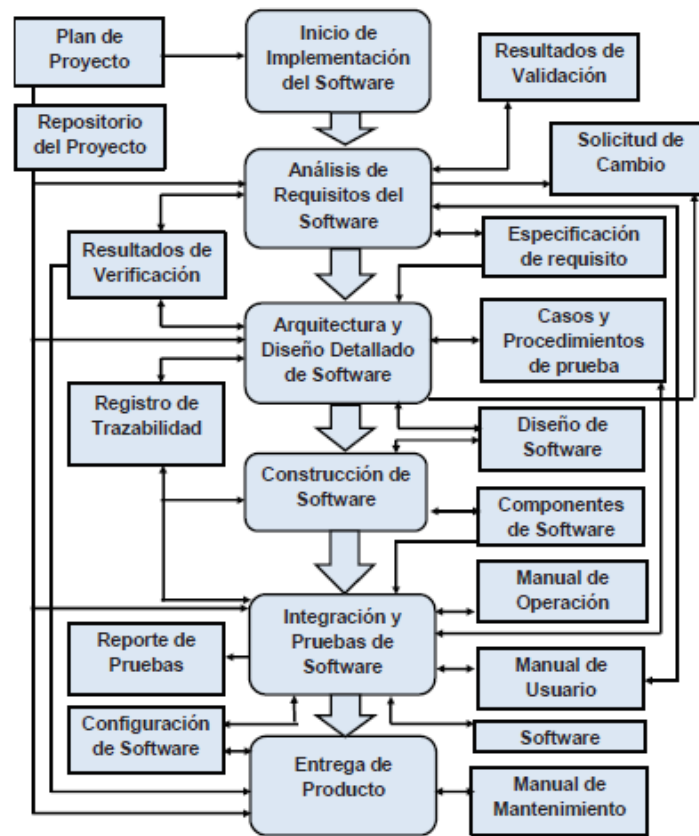


Figura 1.5: Diagrama del proceso de Implementación de Software. Tomada de *Abraham D. and Marcelo P. Factors driving the adoption of ISO/IEC 29110: a case study of small software enterprise, adaptada de [1, 12]*.

Los roles siguientes son las personas que estarán involucradas en las actividades y tareas del proceso IS, las abreviaturas correspondientes se encuentran en la tabla 1.2.

Tabla 1.2: Roles Involucrados de IS.

Rol	Abreviatura
Cliente	CL
Analista	AN
Diseñador	DI
Programador	PR
Administrador de Proyecto	AP
Líder Técnico	LT
Equipo de Trabajo	ET

Roles

Esta es una lista de los roles en orden alfabético, con sus abreviaciones y descripción de las competencias sugeridas.

Tabla 1.3: Roles definidos para la realización de actividades y tareas propuestas por la norma ISO/IEC 29110 Perfil Básico

#	Rol	Abreviatura	Competencias
1	Analista	AN	<ul style="list-style-type: none"> ■ Conocimiento y experiencia que permita obtener, especificar y analizar los requisitos. ■ Conocimiento en diseño de interfaces de usuario y criterios ergonómicos. ■ Conocimiento de técnicas de revisión. ■ Conocimiento de técnicas de edición. ■ Experiencia en desarrollo y mantenimiento de Software.
2	Cliente	CL	<ul style="list-style-type: none"> ■ Conocimiento de los procesos del Cliente y habilidad para explicar los requisitos del Cliente. ■ El Cliente (representante del Cliente) debe tener la autoridad para aprobar los requisitos y sus cambios. ■ El Cliente incluye usuarios representativos con la finalidad de asegurar que el entorno operacional sea dirigido de forma correcta. ■ Conocimiento y experiencia en el dominio de la aplicación.

#	Rol	Abreviatura	Competencias
3	Diseñador	DI	<ul style="list-style-type: none"> ■ Conocimiento y experiencia en Componente de Software y diseño de arquitectura. ■ Conocimiento de técnicas de revisión. ■ Conocimiento y experiencia en la planificación y ejecución de pruebas de integración. ■ Conocimiento de técnicas de edición. ■ Experiencia en desarrollo y mantenimiento de Software.
4	Programador	PR	<ul style="list-style-type: none"> ■ Conocimiento y/o experiencia en programación, integración y pruebas unitarias. ■ Conocimiento de técnicas de revisión. ■ Conocimiento de técnicas de edición. ■ Experiencia en desarrollo y mantenimiento de Software.
5	Administrador del Proyecto	AP	<ul style="list-style-type: none"> ■ Capacidad de liderazgo con experiencia para toma de decisiones, planificación, administrador de personal, delegación y supervisión, conocimiento de finanzas y desarrollo de Software.
6	Líder Técnico	LT	<ul style="list-style-type: none"> ■ Conocimiento y experiencia en el dominio del proceso de Software.

#	Rol	Abreviatura	Competencias
7	Equipo de Trabajo	ET	<ul style="list-style-type: none">■ Conocimiento y experiencia de acuerdo a sus roles dentro del proyecto: LT, AN, DIS y/o PR.■ Conocimiento de los estándares usados por el Cliente y/o por la PO.

Capítulo 2

Guía de uso de la Norma ISO/IEC 29110 Perfil Básico para el desarrollo de Software Científico

El objetivo de esta tesis es adaptar las tareas contenidas en la Norma ISO/IEC 29110 Perfil Básico, para dirigirlas a la comunidad científica que desarrolla software y desean implementar un modelo de desarrollo de software en sus proyectos. Se busca que con esto se pueda estandarizar un proceso de desarrollo especializado para el desarrollo de software científico.

La Guía propuesta está dirigida a PO, departamentos, o proyectos, dedicados a desarrollar software del tipo científico menores a 25 personas.

Esta Guía está basada en lo propuesto por la Norma ISO/IEC 29110 Perfil Básico [1], la cual es una adaptación de la versión en inglés [12]. Las *Plantillas* propuestas están basadas en [19], los documentos propuestos están disponibles para su uso y modificación en los Apéndices A y B.

Con el uso de esta Guía se espera obtener:

- Un proceso de administración que proporcione visibilidad y acciones correctivas sobre los problemas y desviaciones al plan, conforme se lleva a cabo el proyecto.
- Un proceso de Implementación de Software que satisfaga las necesidades de los interesados y que asegure la calidad de los productos de software realizados.

2.1. Proceso de Administración del Proyecto

El proceso de Administración del Proyecto permite establecer y llevar a cabo de manera sistemática las tareas del Proceso de Implementación de Software 2.2, el proceso AP propone realizar 22 tareas distribuidas en 4 actividades.

Las actividades gobernantes del proceso de Administración del Proyecto son las mostradas en la figura 2.1:

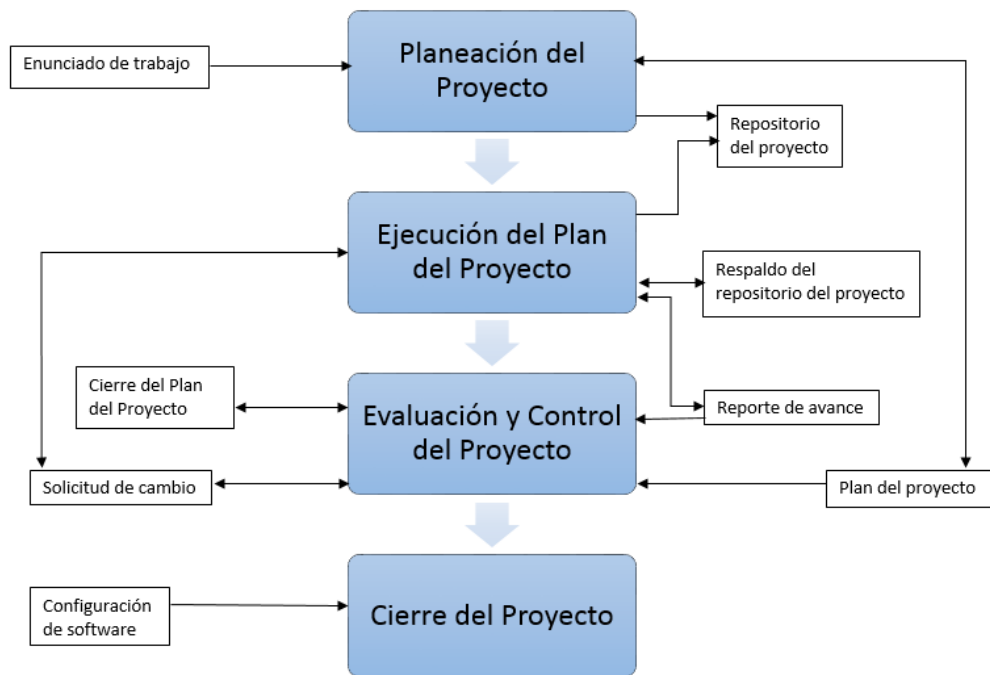


Figura 2.1: Actividades del proceso de Administración del Proyecto.

En cada una de las actividades se muestran ciertos elementos que alimentan y son producto de cada una de las actividades, estos elementos permitirán que se gestionen todas las actividades y tareas necesarias del proyecto en cuestión.

El proyecto será dirigido por el administrador del proyecto, que será un experto en el área de investigación que asignará el trabajo a los miembros del equipo del proyecto y les asignará tiempos de entrega a sus productos.

Consideraciones

Debido a las características de este tipo de proyectos se han excluido ciertas tareas:

Tabla 2.1: Elementos excluidos de las tareas a realizar en la actividad de planeación del proyecto.

Tarea	Causas
Composición del ET	Las características de este tipo de proyectos nos hace suponer que los ET son reducidos y conformado en su mayoría por miembros de la comunidad científica, no se excluyen los roles específicos de cada integrante del ET.
Alcance	El ET visualiza el alcance del proyecto debido a la experiencia o el trabajo de investigación realizado.
Entregables	Los elementos construidos son propensos a cambios bruscos, la definición de los productos depende de diferentes factores.

No hemos hecho mención de las actas de reunión, debido a que pueden ser propuestas por el ET, en nuestro caso no vimos la necesidad de llevar un documento como tal debido a que no se desea agregar formalidad a las reuniones realizadas, esto no implica que su uso no sea necesario, de esta manera lo dejamos a consideración del Administrador del Proyecto.

2.1.1. Actividades del Proceso de Administración del Proyecto

AP.1 Planeación del Proyecto

La actividad de planeación del proyecto documenta las tareas y actividades necesarias para la Administración del Proyecto en el *Plan del Proyecto*.

Dependiendo de la cantidad de integrantes que conforman el ET¹ un integrante puede realizar más de un rol, para promover la documentación de los productos y tareas realizados, los documentos presentados en los Apéndices A y B, contienen elementos clave para el desarrollo. El documento gobernante de todo el ciclo de vida del proyecto es el *Plan del Proyecto*. Debido a que este tipo de proyectos tienden a tener cambios drásticos se ha propuesto se genere un Plan del Proyecto para un tiempo específico de trabajo, cada Plan del Proyecto es similar, exceptuando el plan inicial el cual puede ser revisado en el Apéndice A en la plantilla A.1, este plan tiene la siguiente estructura.

- Enunciado de trabajo.
- Objetivos.
- Tareas.

¹El ET esta considerado a ser conformado por: **Diseñador, Programador, Analista, Líder Técnico.**

- Roles.
- Riesgos.
- Control de versiones.

Los objetivos documentados mostrarán un estado de: Iniciado, Pendiente, Retrasado, Finalizado, los cuales serán registrados en el reporte de seguimiento. Con el fin de llevar un control del estado de cada objetivo, los objetivos pueden ser enfocados al estado teórico del problema a resolver o a la aplicación de alguna actividad del Proceso IS.

Se incluye la tarea de anexar elementos complementarios a este Plan del Proyecto para que se adapte a las necesidades del cliente y no solamente de los investigadores. En esta tarea se pueden anexar instrucciones de entrega del producto, recursos, etc, marcados por la ISO/IEC 29110 Perfil Básico [12], en el proceso AP.

La tabla de tareas de la actividad AP.1 Planeación del Proyecto, contiene las tareas propuestas a realizar y algunas de las tareas originales de la norma ISO/IEC 29110 Perfil Básico [1] se muestran de color azul. También aludimos que existen tareas que son una copia parcial, que han sido complementadas con elementos importantes para este tipo de proyectos o tareas no consideradas por la norma ISO/IEC 29110 Perfil Básico , las cuales junto con las nuevas aportaciones se encuentran de color negro. La plantilla del Plan del Proyecto se encuentran en el Apéndice A.

Tabla 2.2: Lista de tareas de la actividad Planeación del Proyecto.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP ET	AP.1.1 Revisión del enunciado de trabajo, acotar y definir el área de estudio.	Enunciado de trabajo	Enunciado de trabajo revisado y documentado.
AP ET	AP 1.2 Identificar Objetivos y tareas específicas que permitan el logro de los elementos necesarios para producir entendimiento del problema, así como la producción de componentes de <i>Software</i> identificados, incluir las tareas del proceso IS. Documentar objetivos y tareas.	Enunciado del problema. [revisado, acotado y definido]	Objetivos y tareas.
AP LT ET	AP.1.3 Establecer la duración estimada para la realización de las tareas.	Plan del proyecto ■ Tareas.	Plan del Proyecto ■ Tareas.
AP LT ET	AP.1.4 Asignar responsabilidades al ET, establecer roles y responsabilidades.		Roles[Definidos]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT ET	AP.1.5 Identificar y documentar los riesgos que puedan afectar al proyecto.	Plan del Proyecto <ul style="list-style-type: none"> ■ Enunciado de trabajo. ■ Objetivos y tareas. ■ Roles. 	Plan del Proyecto <ul style="list-style-type: none"> ■ Identificación de Riesgos del Proyecto.
AP LT ET	AP.1.6 Documentar la estrategia de control de versiones en el Plan del Proyecto.		Plan del Proyecto <ul style="list-style-type: none"> ■ Estrategia de control de versiones.
AP LT ET	AP.1.7 Identificar y documentar los elementos que el ET crea complementarios e integrarlos al Plan del Proyecto.		Plan del Proyecto <ul style="list-style-type: none"> ■ Elementos complementarios.
AP LT ET	AP.1.8 Generar el Plan del Proyecto integrando los elementos Previamente identificados y documentados.	Todos los elementos previamente definidos.	Plan del Proyecto <ul style="list-style-type: none"> ■ Enunciado de trabajo². ■ Objetivos y tareas. ■ Roles. ■ Identificación de Riesgos del Proyecto. ■ Estrategia de control de versiones*. ■ Elementos complementarios.

²*Elementos que pertenece al Plan del Proyecto Inicial.

Capítulo 2. Guía de uso de la Norma ISO/IEC 29110 Perfil Básico para el desarrollo de Software Científico

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT	AP.1.9 Verificar y aprobar el Plan del Proyecto. Verificar que todos los elementos del Plan del Proyecto son viables y consistentes. Las correcciones necesarias son realizadas hasta que el documento este completo sea aprobado por el AP.	Plan del Proyecto.	Plan del Proyecto. [Verificado]
AP LT	AP.1.10 Establecer el repositorio del proyecto usando la Estrategia de Control de Versiones. Verificar que el ET tenga acceso al repositorio y comprenda el uso y manejo de éste.	Estrategia de control de versiones.	Repositorio del proyecto establecido y disponible.

AP.2 Ejecución del Plan del Proyecto

Esta actividad verifica la ejecución de las tareas y actividades documentadas en el *Plan del Proyecto*, verifica su estado y el tiempo dedicado a su realización. Se proponen realizar reuniones que permitan compartir los conocimientos de cada integrante y registrar el estado de las tareas y su avance, los integrantes del ET presentan dudas y observaciones con respecto a las tareas documentadas por el Plan del Proyecto definido en la tabla 2.2. La tabla 2.3 describe las tareas a realizar en la actividad de ejecución del proyecto, comparece con las presentada en la Tabla 2.7.

Tabla 2.3: Lista de tareas de la actividad de Ejecución del Plan del Proyecto.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP ET	AP.2.1 Monitorear la ejecución del Plan del Proyecto, y registrar la información actual en el reporte de seguimiento.	Plan del Proyecto.	Reporte de seguimiento.
AP LT	AP 2.2 Analizar y evaluar los cambios por su impacto en tiempo e impacto técnico. Los cambios que no afecten los modelos conceptual, matemático o numérico, deben ser realizados. En caso contrario verificar la necesidad de una solicitud de cambio.	Plan del Proyecto. Solicitud de cambio. [Iniciada]	Solicitud de cambio. [Evaluada]
AP ET	AP.2.3 Conducir reuniones de revisión con el ET, revisar el estado de las tareas, riesgos y cambios. Documentar riesgos nuevos y el estado de los riesgos que se hayan presentado, dar seguimiento hasta la conclusión de los mismos.	Reporte de Seguimiento.	Reporte de Seguimiento. [Actualizado]
AP ET	AP.2.4 Conducir reuniones que actualicen el conocimiento obtenido, externar dudas surgidas de la teoría y realizar propuestas de solución, darle seguimiento a las dudas presentadas.		
AP LT ET	AP.2.5 Realizar el Respaldo del Proyecto de acuerdo a la estrategia de control de versiones.	Estrategia de Control de Versiones.	Respaldo del Repositorio del Proyecto.
AP LT ET	AP.2.6 Realizar la recuperación del Repositorio del Proyecto, utilizando el respaldo del Repositorio del Proyecto en caso de ser necesario.	Respaldo del repositorio del Proyecto.	Repositorio [Recuperado].

Los documentos para esta actividad son el Reporte de Seguimiento y la Solicitud de Cambio, que se encuentran en el Apéndice A. Para la ejecución del Plan del Proyecto se propuso la plantilla A.2, en el cual se encuentran los elementos establecidos en la tabla 2.3, en este documento se registra el estado de los objetivos, los cuales pueden ser: Iniciado, Pendiente, Retrasado, Finalizado. También se anexan las observaciones que el ET pueda encontrar durante la realización de dicho Objetivo.

Por otra parte el registro de tareas tiene la finalidad de ser más específico en el trabajo realizado, se registra la fecha propuesta de entrega y la fecha real a cada una de las tareas a realizar, con el fin de poder dar un seguimiento al desempeño de las tareas realizadas.

Los productos de trabajo tienen tres estados: Iniciado, Terminado y Postergado. Los cambios y los riesgos son elementos que se anexan al reporte de avance, finalizando con un resumen del trabajo realizado hasta el momento de la revisión, todos y cada uno de estos elementos pueden ser revisados en el Apéndice A.

Para iniciar una solicitud de cambios, se debe de considerar el impacto que estos tienen en el proyecto, dicha solicitud debe de contener una descripción de la propuesta de las modificaciones, además de contener una justificación clara de la necesidad de realizar el cambio. La solicitud de cambio puede ser consultada en el Apéndice A, plantilla A.3.

AP.3 Evaluación y Control del Proyecto

Esta actividad se encarga del análisis de las tareas y actividades realizadas, con la finalidad de la toma de decisiones y acciones correctivas ante retrasos y problemas que afecten al proyecto en su realización y conclusión. La opinión del experto (AP) es parte fundamental en la toma de decisiones, se debe realizar un análisis del estado de las tareas, objetivos, riesgos y problemas encontrados en las actividades del proceso de IS. Las tareas para esta actividad están descritas en la tabla 2.4.

Tabla 2.4: Lista de tareas de la actividad de Evaluación y Control del Proyecto.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP ET	<p>AP.3.1 Evaluar el progreso del proyecto realizando una evaluación de:</p> <ul style="list-style-type: none"> ■ Objetivos planeados contra los realizados. ■ Tareas realizadas contra las planeadas. ■ Tiempo dedicado a las tareas y actividades. ■ Riesgos reales contra los identificados previamente. 	<ul style="list-style-type: none"> ■ Plan del Proyecto. ■ Reporte de seguimiento 	Reporte de seguimiento [Evaluado].
AP ET	<p>AP 3.2 Establecer acciones para corregir problemas que amenacen el cumplimiento de las tareas y objetivos, en caso de ser necesario documentarlos en el cierre del Plan del Proyecto y darles seguimiento hasta su conclusión.</p>	Reporte de seguimiento [evaluado]	Cierre del Plan del Proyecto.
AP ET	<p>AP.3.3 Realizar el cierre del Plan del Proyecto, realizando un resumen del trabajo realizado y productos generados. Proponer mejoras de trabajo y documentarlos.</p>	<ul style="list-style-type: none"> ■ Plan del Proyecto. ■ Reporte de seguimiento [Analizado]. 	<p>Cierre del Plan del Proyecto.</p> <ul style="list-style-type: none"> ■ Resumen de trabajo. ■ Retroalimentación.
AP ET	<p>AP.3.4 Identificar cambios en los modelos conceptuales, matemáticos y numéricos, en caso de necesitar cambios bruscos a los requerimientos, analizar el uso de una solicitud de cambio, dar seguimiento hasta su conclusión.</p>	Reporte de seguimiento [Analizado].	Solicitud de cambio. [Iniciado]

El documento de Cierre del Plan del Proyecto y solicitud de cambio, se encuentran en el Apéndice A. La evaluación es una tarea que se encuentra ligada al seguimiento, es por eso que se debe tener un mecanismo para realizar dicha evaluación con el objetivo de poder tomar las decisiones necesarias para que el proyecto se desarrolle de la mejor manera.

La evaluación está destinada a los responsables de la administración del proyecto, sin embargo, también se busca que esto sea comprensible para el ET.

Para la realización de esta actividad es necesario realizar una evaluación inicial, habrá que hacer la revisión de los reportes de seguimiento y hacer el análisis de:

- Objetivos.
- Tareas.
- Productos.
- Riesgos.

Se analiza el estado de cada uno de los elementos antes mencionados, se propone hacer una identificación inicial de los problemas que podrían afectar al proyecto, esta identificación se hace con respecto a los problemas encontrados en la realización de las tareas y cumplimiento de objetivos.

Se hace una revisión o evaluación a mitad de periodo del Plan del Proyecto verificando el estado de las tareas asignadas y dificultades encontradas, verificando que exista un cambio en el avance de cada uno de ellos, se debe prestar especial atención en las fechas de entrega de cada una de las tareas asignadas, si es posible definir colores para las tareas iniciadas, pendientes y finalizadas. Se espera tener una visión clara del estado del proyecto, teniendo una perspectiva del tiempo restante del proyecto y el avance de la construcción del sistema solicitado.

Habrà que hacer una reflexión y aprendizaje periódico, en la cual se recomienda la participación de todos los integrantes del ET, sobre las experiencias, enseñanzas obtenidas y propuestas de mejora³, con el objetivo de ir mejorando la forma de trabajo. También se debe establecer un resumen de las tareas realizadas y las tareas pendientes, revisar el estado de los productos a realizar. Para todo lo descrito en este último párrafo se ha propuesto la plantilla de cierre del Proyecto A.4, en nuestro caso de estudio se realizó una retroalimentación al final de cada Plan del Proyecto.

AP.4 Cierre del proyecto

El cierre del proyecto varia de un proyecto a otro, puede constar únicamente de la entrega de resultados obtenidos del software⁴, estos resultados se encuentran anexados al sistema. En caso de que exista un contrato, el cierre del proyecto proporciona documentación y productos del proyecto

³Sin llegar a establecer inconformidades entre los integrantes del ET.

⁴Tenemos por entendido por software a la amalgama que genera el código fuente del sistema creado y la documentación del mismo.

de acuerdo con los requisitos establecidos en dicho contrato, las tareas a realizar en esta actividad están descritas en la tabla 2.5.

Tabla 2.5: Lista de tareas de la actividad de Cierre del Proyecto

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP ET CL	AP.4.1 Realizar el cierre del proyecto en conformidad al contrato establecido o a las necesidades de los involucrados, realizando la entrega del software generado y la investigación realizada que respalden el funcionamiento del sistema.	Configuración de Software. [Entregada]	Configuración de Software [Aceptada].
AP ET CL	AP 4.2 Actualizar el repositorio del proyecto.	Configuración de Software. [Aceptada]	Repositorio del proyecto. [Actualizado]

Se debe de hacer un resumen de todos los productos generados, cambios realizados, riesgos encontrados, Actividades u objetivos, que se realizaron durante el proyecto, también hay que verificar que todos los defectos encontrados en los productos hayan sido corregidos. Finalmente habrá que hacer una retroalimentación general de todo el proyecto, el cual permita observar todas las experiencias generadas y poder obtener propuestas de mejora para los proyectos nuevos.

El cierre del proyecto se debe realizar conforme a los establecido al Plan del Proyecto, es decir, se entregará la configuración de Software la cual debe de contener el sistema y la documentación correspondiente (Especificación de requerimientos, Arquitectura y Diseño Detallado de Software, Pruebas, especificaciones de uso), los elementos que sean para consumo interno e histórico para la PO deberán de ser anexados al repositorio, junto con los elementos antes mencionados.

2.2. Proceso de Implementación de Software

El propósito del Proceso de *Implementación de Software* es la realización sistemática de 54 tareas distribuidas en 7 actividades, las actividades del proceso de IS son: Antecedentes de Software, Análisis de Modelos, Arquitectura y Diseño Detallado de Software, Construcción del Modelo Computacional, Pruebas de Integración, Validación y la Entrega de Productos.

La figura 2.3 muestra el flujo de información entre las actividades propuestas para este proceso, se ha definido una retroalimentación entre las actividades de Construcción del Modelo Computacional y Pruebas de Integración debido a que se observó una conexión directa entre estas dos actividades, pero puede haber una conexión directa con otras actividades, la cual esta definida por los cambios a realizar.

Hemos tenido la oportunidad de trabajar con el Dr. Luis Miguel de la Cruz Salas, investigador del Instituto de Geofísica de la UNAM. Algunos de los proyectos que ha dirigido tienen características importantes en el proceso utilizado en el desarrollo de software, las actividades principales en estos procesos se muestran en la figura 2.2. Se realizó la comparación de este proceso con el propuesto por la ISO/IEC 29110 Perfil Básico, en el diagrama de la figura 2.3, diagrama conocido como MCM (*mathematical and computational modeling*) propuesto en [7].

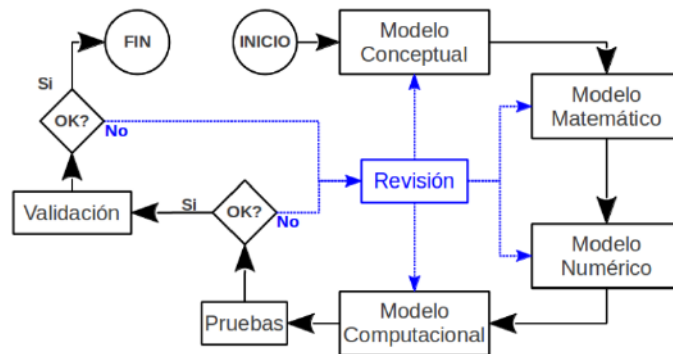


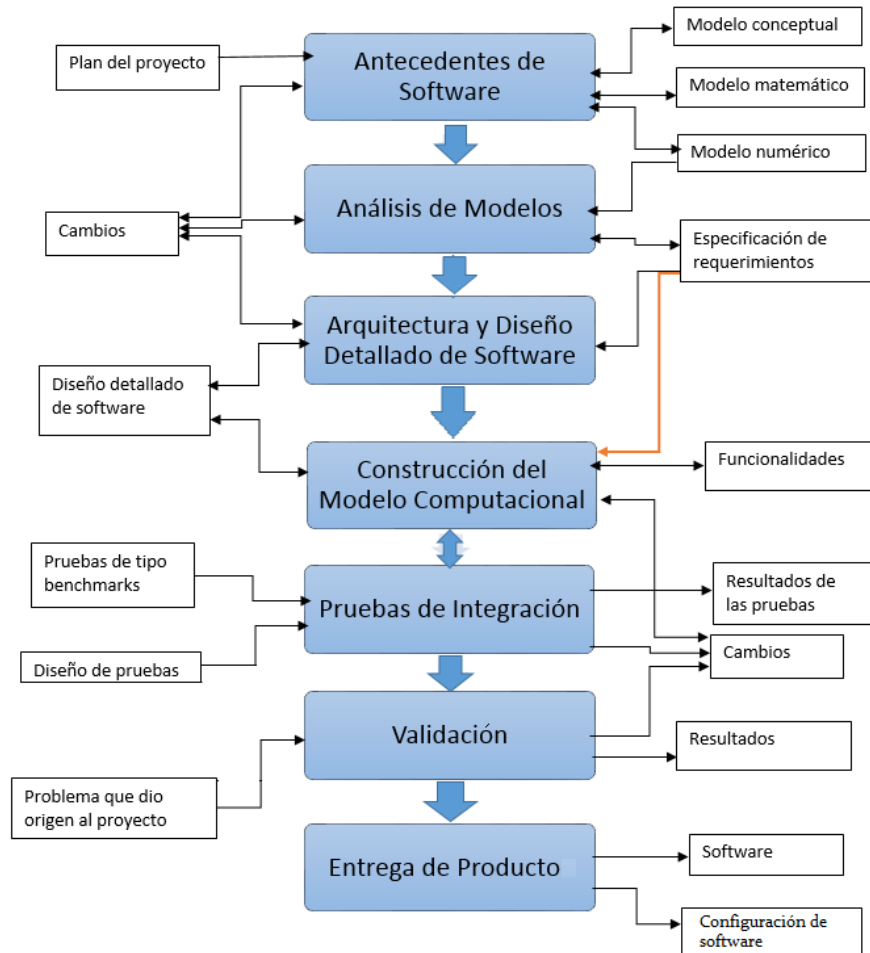
Figura 2.2: Proceso MCM seguido en la construcción de software científico, Tomada de Luis M. C and Eduardo R. *General template units for the finite volume method in box-shaped domains*.

El proceso MCM es seguido por algunos alumnos e investigadores de cómputo científico, este proceso está descrito en algunos trabajos realizados en la UNAM [18, 24], el proceso presentado fue la base para definir el diagrama de las actividades del proceso IS figura 2.3.

La figura 2.3 muestra una línea naranja en el documento de Especificación de Requerimientos, debido a que suponemos que podría existir la necesidad de revisar ciertos elementos de este

documento, los cuales complementen lo estipulado con el documento de *Arquitectura y Diseño Detallado de Software*, aunque esto es un elemento opcional.

Figura 2.3: Actividades del Proceso de Implementación de Software.



Las actividades definidas en la Norma ISO/IEC 29110 Perfil Básico se han modificado conforme al proceso MCM mostrado en [7], en la table 2.6 definimos los objetivos asociados a cada actividad.

Tabla 2.6: Tabla de Objetivo de cada una de las actividades propuestas del Proceso de Implementación de Software.

Actividad	Objetivos
-----------	-----------

Actividad	Objetivos
Antecedentes de Software	Realización de tareas de investigación para el establecimiento de los modelos conceptual, matemático y numérico, además de las capacitaciones necesarias para la realización de los roles definidos y las practicas de software a implementar.
Análisis de Modelos	Realización sistemática de tareas que permitan la comprensión de los modelos, trabajo matemático necesario para la definición de los requerimientos de software y las características del entorno de programación.
Arquitectura y Diseño Detallado de Software	Definición de los componentes, funcionalidades del software y Arquitectura de Software.
Construcción del Modelo Computacional	La realización sistemática de lo establecido en los documentos de Especificación de Requerimientos y Arquitectura y Diseño Detallado de Software, pruebas de las funcionalidades de cada componete en construcción.
Pruebas de Integración	Diseño y aplicación de pruebas de errores del sistema construido y establecimiento de las pruebas del tipo Benchmark. ⁵
Validación	Una vez que se ha pasado la etapa de pruebas de integración, se realiza la evaluación del sistema, haciendo uso del sistema en problemas que dieron origen al proyecto.
Entrega de Producto	Es la entrega de los productos de software terminados y funcionales.

Las actividades aquí propuestas están destinadas para ser utilizadas por la comunidad científica para establecer procesos e implementar cualquier enfoque o metodología de desarrollo, incluyendo: ágil, evolutivo, incremental, etc., basadas en las necesidades de la PO. Las plantillas de los documentos propuestos a ser generados durante el proyecto se encuentran anexados en el Apéndice B.

⁵Las pruebas del tipo Benchmark son problemas establecidos para los cuales se conoce la solución analítica, se realiza la evaluación necesaria entre la solución obtenida, con respecto a la conocida.

2.2.1. Actividades del Proceso de Implementación de Software

IS.1 Antecedentes de Software

La actividad de Antecedentes de Software asegura que el Plan del Proyecto es llevado a cabo por el ET. La actividad provee:

1. Revisión del Plan del Proyecto por parte del ET.
2. Compromiso con los integrantes del ET y el Administrador del proyecto.
3. Establecimiento de un ambiente para la investigación e implementación.
4. Consulta de fuentes de información.
5. Propuestas de modelos conceptual, matemático y numérico.

Las tareas para esta actividad están descritas en tabla 2.7.

Tabla 2.7: Lista de tareas de la actividad Antecedentes de Software

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP ET	IS.1.1 Revisar el Plan del Proyecto actual con los miembros del ET, con la finalidad de lograr un entendimiento común y tener un compromiso con el proyecto.	Plan del Proyecto.	Plan del Proyecto . [Revisado]
AP ET	IS.1.2 Establecer o actualizar el ambiente de investigación e implementación.	Plan del Proyecto. [Revisado]	
AP LT ET	IS.1.3 Identificar y proponer fuentes de información relacionados con el área de investigación y el problema en particular que se desea resolver, realizar revisiones en revistas u artículos y literatura especializada. Proponer capacitaciones o literatura especializada en practicas de Ingeniería de Software enfocadas a cada rol.	Plan del Proyecto . ■ Enunciado de Trabajo. ■ Roles.	bibliografía y capacitación propuesta.
LT ET	IS.1.4 Asignación de revisiones sistemáticas conjuntas para la propuesta y entendimiento de las bases teóricas del modelo conceptual, matemático y numérico, además de las practicas de software y herramientas (CASE) a utilizar.	■ Bibliografía.	■ Bibliografía [revisada]. ■ Propuesta de modelos.

IS.2 Análisis de Modelos

La actividad de análisis de modelos analiza los modelos propuestos, se realiza el análisis de viabilidad de los modelos propuestos para dar solución al problema, establece el trabajo matemático necesario para establecer los requerimientos del proyecto. La actividad de Análisis de Modelos provee:

1. Revisión del Plan del Proyecto por parte del ET para la asignación de tareas.
2. Análisis de la viabilidad de la solución propuesta.
3. Realización del trabajo matemático necesario.
4. Obtención de la Especificación de Requerimientos de Software provenientes de los modelos propuestos, el trabajo matemático y el cliente (Si aplica).
5. Acuerdo sobre los requerimientos.

Las tareas para esta actividad están descritas en la tabla 2.8, la plantilla B.1, es el documento a ser establecido en esta actividad.

Tabla 2.8: Lista de tareas de la actividad de Análisis de Modelos

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT	IS.2.1 Analizar el método de solución propuesto verificando que sea consistente y viable con el problema propuesto, documentar y sustentar los motivos del análisis realizado. Proponer un método de solución alternativo, si es necesario iniciar una solicitud de cambio.	<ul style="list-style-type: none"> ■ Plan del Proyecto*. ■ Modelos. 	Solicitud de cambio [Iniciada]
AP LT ET	IS.2.2 Asignar tareas a los miembros del ET de acuerdo a cada rol basado en el Plan del Proyecto Actual. Identificar y priorizar las tareas de trabajo matemático de los modelos matemático y numérico. Proponer nuevas fuentes de información especializadas que apoyen el trabajo matemático a realizar (sistemas previos, investigación, experto en el área de estudio correspondiente).	<ul style="list-style-type: none"> ■ Plan del Proyecto . [revisado] ■ Tareas. 	

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT ET	IS.2.3 Verificar el planteamiento del modelo matemático (Condiciones iniciales, de frontera, definición de ecuaciones matemáticas, etc), identificar los requerimientos identificados de los modelos propuestos (si es posible dividirlos en funcionalidades). Identificar los requerimientos provenientes del trabajo matemático realizado, documentarlos en la especificación de requerimientos.	Modelos matemático y numérico.	Especificación de requerimientos: <ul style="list-style-type: none"> ■ Descripción del producto. ■ Requerimientos Funcionales.
AN LT	IS.2.4 Recabar información necesaria proveniente del cliente, analizar requerimientos faltantes que complementen los Requerimientos del Software identificados, con el fin de recabar nuevo requerimientos.	Especificación de requerimientos: <ul style="list-style-type: none"> ■ Descripción del producto [Revisado]. ■ Requerimientos Funcionales. 	Especificación de requerimientos: <ul style="list-style-type: none"> ■ Descripción del Producto. [Actualizados] ■ Requerimientos Funcionales. [Actualizados]
AP ET	IS.2.5 Identificar limitaciones y restricciones que afecten a la construcción de software, se debe tener especial consideración de eficiencia en el uso de memoria, manejo de errores, entre otros.	Especificación de requerimientos	Especificación de requerimientos [Actualizado]. <ol style="list-style-type: none"> 1. Restricciones y limitaciones de construcción. 2. Requerimientos no Funcionales.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AN LT ET	IS.2.6 Analizar las funcionalidades propuestas para la Construcción de Software, actualizar el documento de especificación de requerimientos. Verificar que la especificación de requerimientos sea correcta, revisar que la descripción del producto y los requerimientos sean consistentes con los modelos propuestos. Revisar que no existen inconsistencias o ambigüedades. Realizar las correcciones pertinentes en el documento hasta tener el visto bueno del AN.	Especificación de requerimientos [Analizado] 1. Descripción del producto. 2. Requerimientos Funcionales. 3. Requerimientos no Funcionales. 4. Restricciones de construcción.	Especificación de requerimientos. [Actualizado] 1. Descripción del producto. 2. Requerimientos Funcionales. 3. Requerimientos no Funcionales. 4. Restricciones de construcción.
AP LT ET	IS.2.7 Verificar y obtener validación del análisis y discretización de los modelos propuestos.	<ul style="list-style-type: none"> ■ Especificación de requerimientos. [revisado] ■ Discretización y modelos. [revisados] 	<ul style="list-style-type: none"> ■ Especificación de requerimientos. [Validado] ■ Discretización y modelos. [Validado]
LT ET	IS.2.8 Verificar que los requerimientos estén completos, sin ambigüedades ni contradicciones, realizar las correcciones necesarias para que el documento tenga el visto bueno por el Analista, en caso de modificaciones significativas a los modelos, se propone una solicitud de cambio.	<ul style="list-style-type: none"> ■ Modelo propuestos. ■ Especificación de requerimientos. 	<ul style="list-style-type: none"> ■ Especificación de requerimientos. [verificada] ■ Solicitud de cambio. [Iniciada]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP CL AN	IS.2.9 Validar y obtener la aprobación de la especificación de requerimientos. Validar que la especificación de requerimientos satisfaga las necesidades y expectativas de los involucrados, incluyendo las interfaces (si aplica). Las correcciones necesarias se realizan hasta que el documento tenga el visto bueno por el AN; Establecer en línea base (opcional).	Especificación de Requerimientos. [Verificada]	Especificación de requerimientos. [validada, en línea base]

La plantilla para la especificación de requerimientos B.1, esta distribuida de la siguiente manera: Hay un apartado para la descripción del producto, esta descripción debe ayudar al ET a comprender las funcionalidades esperadas del sistema, basadas en el conocimiento obtenido de los modelos establecidos, contiene un glosario de términos que permite entender la notación y elementos específicos de la teoría utilizada. Posteriormente se hace una descripción de los requerimientos funcionales y no funcionales del sistema, utilizamos una sección para especificar las restricciones de construcción del sistema.

IS.3 Arquitectura y Diseño Detallado de Software.

Transforma los Requerimientos de Software en la arquitectura del sistema y en el Diseño Detallado de Software. Esta actividad provee:

1. Revisión por parte del ET al Plan del Proyecto para determinar la asignación de tareas.
2. Revisión de la especificación de requerimientos por parte del ET.
3. Normatividad interna para la construcción de los componentes de software.
4. El Diseño de la Arquitectura de Software, componentes (funcionalidades) y las interfaces (si aplica).
5. Diseño Detallado de Software.
6. El Diseño Detallado de Software verificado y los defectos corregidos.
7. Definición de entradas y salidas de los componentes de software para la integración.
8. Productos y documentos de Arquitectura y Diseño Detallado de Software bajo el control de versiones.

Las tareas para esta actividad están descritas en tabla 2.9.

Tabla 2.9: Lista de tareas de la actividad de Arquitectura y Diseño Detallado de Software

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT AN DIS	IS.3.1 Revisión del Plan del Proyecto actual para la asignación de tareas y responsabilidades de acuerdo a cada rol.	Plan del Proyecto: <ul style="list-style-type: none"> ■ Objetivos. ■ Tareas. 	
AN DIS	IS.3.2 Comprender la especificación de Requerimientos.	Especificación de Requerimientos [Validado, en línea base.]	
AN DIS	IS.3.3 Documentar y plantear la normatividad interna de programación, considerar el paradigma de programación utilizado, con el fin de poder generar el pseudocódigo más comprensible para el ET.	Especificación de Requerimientos [Validado, en línea base.]	Arquitectura y Diseño Detallado de Software: <ul style="list-style-type: none"> ■ Normatividad interna.
AN DIS	IS.3.4 Documentar o actualizar el documento de Arquitectura y Diseño Detallado de Software. Analizar la especificación de requerimientos para generar el diseño arquitectónico adecuado, la conformación en subsistemas y componentes de software de la arquitectura de software establecido. Basar la conformación de los componentes con la especificación de requerimientos, de tal manera que aproximen la matemática planteada. Proporcionar el detalle de los componentes de la arquitectura, de tal manera que pueda permitir la construcción en forma clara. Generar o actualizar la trazabilidad entre componentes (funcionalidades). Si es necesario revisar arquitecturas de sistemas anteriores o arquitecturas existentes con el fin de poder generar la arquitectura del sistema. Describir a detalle la apariencia y el comportamiento de las interfaces del sistema (si aplica).	Especificación de Requerimientos [Validado, en línea base.]	Arquitectura y Diseño Detallado de Software. <ul style="list-style-type: none"> ■ Arquitectura de Software.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT ET	IS.3.5 Revisar y obtener la aprobación del ET sobre la normatividad interna propuesta, obtener la comprensión, entendimiento y compromiso del ET, actualizar la normatividad interna con los cambios y las propuestas aceptadas por el ET.	Arquitectura y Diseño Detallado de Software. ■ Normatividad interna.	Arquitectura y Diseño Detallado de Software ■ Normatividad interna. [revisada, aceptada]
LT ET	IS.3.6 Verificar y obtener la aprobación del documento de Arquitectura y Diseño Detallado de Software, verificar que la arquitectura cumpla con los requerimientos de software, verificar que la trazabilidad entre componentes sea clara para todos y contenga las relaciones adecuadas. Verificar que no existan inconsistencias entre el Diseño Detallado de Software y los requerimientos de software; si aplica, establecer en línea base. Los cambios que no afecten los modelos planteados deber de ser realizados, en caso de necesitar cambios significativos proponer una solicitud de cambio.	■ Especificación de requerimientos. [Validados, en línea base] ■ Arquitectura y Diseño detallado de Software.	■ Arquitectura y Diseño Detallado de Software. [Verificado, en línea base] ■ Solicitud de cambio [Iniciada].
AN DIS ET	IS.3.7 Analizar la interacción del usuario con el sistema, establecer o actualizar los casos de prueba y procedimientos de prueba para la integración. Comprometer al ET a realizar las pruebas necesarias de cada uno de los componentes de software, definir los casos de prueba a los que se someterá el sistema ya integrado (pruebas Benchmark).	■ Especificación de Requerimientos. [Validado, en línea base.] ■ Arquitectura y Diseño Detallado de Software. [Validado, en línea base.]	■ Casos de prueba. [Preliminares] ■ Procedimientos de prueba. [Preliminares]

La plantilla B.2 de Arquitectura y Diseño Detallado de Software, contiene la información para los desarrolladores, con el objetivo de poder realizar la construcción de los componentes de software necesarios, para poder obtener las funcionalidades del sistema especificado en el documento de la actividad IS.2, el documento cuenta con tres puntos a ser establecidos, el primero es una *Normatividad Interna*, este elemento establece las reglas a las cuales el equipo de desarrollo debe de basarse y seguir, con el objetivo de que sea entendible el código desarrollado, los comentarios estén

establecidos, nombre de variables, clases, métodos, etc.

La arquitectura es un elemento que tiene su propio proceso de construcción, la propuesta debe de encapsular las funcionalidades del sistema, permitir que el mantenimiento del sistema sea posible y que las modificaciones no afecten a todo el sistema, se busca que haya un alto acoplamiento y una baja cohesión, la propuesta de la arquitectura ayuda a los desarrolladores a seguir una construcción del sistema más ordenada.

IS.4 Construcción del Modelo Computacional

La solución ha pasado por la transición suficiente que permite la construcción de la solución planteada del problema original, en este punto se cuenta con el conocimiento del comportamiento que el sistema debe de tener, así como las operaciones que éste debe de realizar. El documento de Arquitectura y Diseño Detallado de Software proporciona el conocimiento necesario para su construcción, el programador debe de tener el conocimiento suficiente del respaldo teórico del problema, así como el método de solución propuesto.

La actividad de Construcción del Modelo Computacional transforma la Arquitectura y Diseño Detallado de Software en el código y los componentes de software. La actividad provee:

1. La revisión por parte del ET del Plan del Proyecto para la asignación de tareas.
2. La revisión del Diseño Detallado de Software por parte del ET para el establecimiento de reglas de programación basadas en la normatividad interna.
3. La revisión del Diseño Detallado de Software por parte del ET para determinar la secuencia de construcción del software.
4. Los componentes de software codificados y probados a un nivel aceptable, con respecto al manejo y transformación de datos⁶.

Las tareas para esta actividad están descritas en tabla 2.10.

⁶Definimos esta transformación de datos en las funcionalidades del sistema y ecuaciones importantes de los modelos propuestos, cada componente puede definir una funcionalidad o un conjunto de operaciones.

Tabla 2.10: Lista de tareas de la actividad Construcción del Modelo Computacional

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT PR	IS.4.1 Asignar Tareas a los miembros del Equipo de Trabajo en relación a su rol, de acuerdo al Plan del Proyecto actual.	Plan del Proyecto <ul style="list-style-type: none"> ■ Objetivos. ■ Tareas. 	
AP LT ET	IS.4.2 Revisar el documento de Arquitectura y Diseño Detallado de Software, comprender la normativa de programación, los componentes a construir y sus relaciones.	Arquitectura y Diseño Detallado de Software [Verificado, en línea base].	
PR	IS.4.3 Construir o actualizar los Componentes de Software descritos en el documento de Arquitectura y Diseño Detallado de Software.	Arquitectura y Diseño Detallado de Software. [Verificado, en línea base]	Componentes de software.
LT ET	IS.4.4 Verificar que cada componente de software construido tengan las pruebas necesarias de las funcionalidades, estas pruebas son para verificar que los cálculos se estén realizando correctamente, debido a que pueden ser extensas estas se documentan a necesidad del ET.	Componentes de Software.	Componentes de software [componentes probados].
PR	IS.4.5 Identificar y documentar o actualizar pruebas esenciales a ecuaciones básicas de cada componente, estas pruebas deben definir funcionalidades principales que aseguren el funcionamiento del componente construido.	Componentes de software.	Funcionalidades. [Probadas]
PR	IS.4.6 Identificar o actualizar los casos de prueba de integración, identificar datos de entrada y salida de cada componente y verificar que se pueda realizar la integración entre componentes.	Arquitectura y Diseño Detallado de Software [Validado, en línea base]	Pruebas de integración.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
PR	IS.4.7 Corregir defectos encontrados hasta lograr la funcionalidad definida en la especificación de requerimientos, verificar que cada componente pueda aceptar los tipos de datos definidos y obtenga las salidas con las características adecuadas para la integración de componentes.	<ul style="list-style-type: none"> ■ Especificación de requerimientos. [validada, en línea base] ■ Arquitectura y Diseño Detallado de Software. [validado, en línea base] ■ Componentes de software. [Funcionalidades probadas] 	Componentes de software. [Corregidos]
LT DIS PR	IS.4.8 Actualizar el documento de Arquitectura y Diseño Detallado de Software, actualizar la trazabilidad entre componentes de software construidos o modificados.	Arquitectura y Diseño Detallado de Software <ul style="list-style-type: none"> ■ Arquitectura. ■ Componentes de software. [Corregidos] 	Arquitectura y Diseño Detallado de Software <ul style="list-style-type: none"> ■ Arquitectura. [Actualizado]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT PR	<p>IS.4.9 Incorporar los componentes de software, bajo el control de versiones al repositorio del proyecto, verificar que se tengan los permisos y accesos necesarios. Los elementos a ser almacenados en el repositorio son:</p> <ul style="list-style-type: none"> ■ Software. ■ Especificación de Requerimientos. ■ Arquitectura y Diseño Detallado de Software. ■ Pruebas de Integración. ■ Pruebas Benchmark. <p>Estos elementos deben de asegurar el buen funcionamiento del sistema solicitado.</p>	<ul style="list-style-type: none"> ■ Control de versiones. [validado] ■ Componentes de software. [probados] 	<p>Repositorio. [Actualizado]</p>

IS.5 Pruebas de Integración.

Esta actividad documenta e integra los componentes de software contruidos, si el usuario tiene una interacción alta con el sistema, se recomienda realizar casos de prueba más elaborados, en nuestro caso hemos supuesto que el usuario tiene una interacción mínima con el sistema, no es nuestro objetivo establecer pruebas que solo se enfoquen a nuestros supuestos, es por ésto que se debe prestar especial atención a que tipo de pruebas establecer.

La actividad de pruebas de integración asegura que los componentes de software integrados satisfacen los requerimientos de los modelos, así como los requerimientos de software, cumpliendo con las funcionalidades necesarias para la construcción del método de solución planteado. Esta actividad provee:

- Revisión por parte del ET al Plan del Proyecto para la asignación de tareas.
- Comprensión de los casos de prueba de integración.
- Comprensión de las pruebas de tipo (Benchmark).
- Componentes de software integrados, los defectos corregidos y documentados.
- Soluciones calculadas de las pruebas Benchmark con valores y resultados aceptables, los defectos corregidos y documentados.

Las tareas para esta actividad están descritas en tabla 2.11.

Tabla 2.11: Lista de Tareas de la Actividad Pruebas de Integración

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT PR	IS.5.1 Asignar tareas a los miembros del Equipo de Trabajo en relación a su rol de acuerdo al Plan del Proyecto actual.	Plan del Proyecto ■ Tareas.	
AP ET	IS.5.2 Entender las pruebas de integración, en base a la trazabilidad de los componentes de software y a la definición de la matemática establecida y realizada, establecer o actualizar el ambiente de prueba.	Componentes de software [probados].	■ Pruebas de integración. [revisadas] ■ Trabajo matemático. [revisado]
PR	IS.5.3 Integrar el Software usando los Componentes de Software generados, actualizar los Procedimientos de Prueba para las pruebas de integración conforme sea necesario.	Pruebas de integración	Componentes [integrados].
LT ET	IS.5.4 Realizar los procedimientos de integración documentar los resultados encontrados, los cambios no significativos deben ser realizados y comentados, documentar los resultados encontrados.	■ Componentes. [Integrados] ■ Pruebas de integración.	■ Resultados de pruebas. ■ Pequeñas modificaciones.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
PR	IS.5.5 Verificar que las correcciones a realizar no afecten las funcionalidades de los componentes ya construidos, verificar que las entradas y salidas de datos tengan las características exactas a las especificadas en los documentos de Especificación de Requerimientos y Arquitectura y Diseño Detallado de Software.	<ul style="list-style-type: none"> ■ Resultados de pruebas. ■ Cambios ■ Especificación de requerimientos. [verificado] ■ Arquitectura y Diseño Detallado de Software. [verificado]. 	Cambios. [verificados]
PR	IS.5.6 Corregir los defectos encontrados y realizar pruebas para verificar que los cambio realizados no afectan las características de las soluciones obtenidas, verificar la integración hasta que el sistema cumpla con la funcionalidad requerida. (modelen el método de solución requerido)	Componentes de software. [integrados]	Componentes de software. [corregido]
LT PR	IS.5.7 Actualizar y validar el documento de Arquitectura y Diseño Detallado de Software (trazabilidad de los componentes) en caso de ser necesario.	<ul style="list-style-type: none"> ■ Componentes de Software. [corregido] ■ Arquitectura y Diseño Detallado de Software. [Actualizado] 	Arquitectura y Diseño Detallado de Software. [Actualizado, validado]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT PR	IS.5.8 Documentar o Actualizar las pruebas de tipo Benchmark. Entender las pruebas a realizar y los criterios de satisfacción de las pruebas, en caso de ser necesario pedir al experto del área o al cliente que defina las pruebas a realizar. Verificar que las pruebas de integración hayan sido completadas y el sistema pueda realizar las pruebas Benchmark, verificar que se comprendan las limitaciones así como el dominio de implementación de la solución y el problema a resolver.	Componentes de software [integrado, funcional]	Pruebas Benchmark. [definidas]
LT PR	IS.5.9 Analizar y verificar que las pruebas Benchmark sean las adecuadas para el sistema realizado y sean comprendidas por los integrantes del ET, entender los requerimientos y la aplicación de estas al sistema integrado. Verificar con el cliente o experto del área que los datos estén disponibles y tengan las características definidas en el documento de especificación de requerimientos.	<ul style="list-style-type: none"> ■ Pruebas Benchmark. ■ Especificación de requerimientos. [validado] 	Pruebas Benchmark. [verificadas]
PR	IS.5.10 Verificar que los resultados a obtener estén bien identificados, en caso de necesitar herramientas de graficación y modelación no consideradas en la construcción del sistema, documentar su uso en el documento de pruebas Benchmark.	<ul style="list-style-type: none"> ■ Componentes de Software. [integrado.] ■ Pruebas Benchmark. [verificadas] 	Pruebas Benchmark. [verificadas]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
PR	IS.5.11 Realizar las pruebas Benchmark realizando las modificaciones necesarias al sistema para que se puedan realizar estas pruebas, se espera que las modificaciones no afecten la funcionalidad, y sean basadas en las entradas de datos. Documentar los resultados en cada prueba realizada.	<ul style="list-style-type: none">■ Componentes de software. [integrado]■ Pruebas Benchmark. [verificadas]■ Datos.	Resultados de pruebas Benchmark.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT PR	<p>IS.5.12 En caso de no pasar las pruebas Benchmark, revisar y analizar:</p> <ul style="list-style-type: none"> ■ Modelos: Verificar que los modelos propuestos sean correctos, que se han considerado todas las variables y condiciones necesarias para descartar que exista un error en estos. ■ Componentes: Verificar que los componentes de software se adapten a la especificación de requerimientos y la Arquitectura y Diseño Detallado de Software, para descartar que no exista un error en los componentes. ■ Otros: Verificar otros elementos que no hayan sido considerados en la programación de los componentes (funciones predefinidas por el lenguaje de programación, métodos de cálculos de estimación de errores), verificar que se están comparando los resultados obtenidos correctamente. <p>Realizar los cambios necesarios, dependiendo de los cambios se deben realizar todas las pruebas anteriores, hasta cumplir exitosamente los criterios de aceptación. En caso de necesitar cambios significativos proponer una solicitud de cambio.</p>	<ul style="list-style-type: none"> ■ Pruebas Benchmark. ■ Modelos propuestos. ■ Especificación de Requerimientos. [validado] ■ Diseño Detallado de Software. [validado] ■ Especificación de funciones utilizadas. ■ Componentes de software [integrados]. 	<ul style="list-style-type: none"> ■ Solicitud de cambio. [iniciada]. ■ Componentes de Software. [probado e integrado].

Las plantillas para las pruebas se dividen en dos, tal cual están especificadas en la tabla *Pruebas de Integración* y *Pruebas Benchmark*, sin perder la importancia de las pruebas realizadas durante la construcción de los componentes. El documento de *Pruebas de Integración* que se encuentra en la plantilla B.3, son pruebas de caja negra en las cuales se conoce el valor de entrada y el valor esperado, contiene los cambios propuestos y los defectos encontrados, esto con el objetivo de realizar las modificaciones necesarias y que la integración pueda ser realizada sin problemas.

Las *Pruebas de Benchmark*, están en la plantilla B.4, este documento esta compuesto por 3 elementos:

- Descripción de la prueba.
- Descripción de los resultados.
- Observaciones.

La descripción de la prueba contiene todos los elementos teóricos para la realización de la prueba, mientras que la descripción de los resultados establece lo obtenido al final de dicha prueba, el cálculo de errores es importante en esta parte, es por eso que se debe de considerar el método adecuado. Finalmente las observaciones contienen los elementos no considerados anteriormente, modificaciones realizadas para la realización de las pruebas, posibles modificaciones, etc.

IS.6 Validación

Al referirnos al software [Validado], se hace referencia a que el software ya ha pasado por las actividades de pruebas de integración y que ha sido aplicado al problema original del proyecto de manera exitosa, este termino sólo aplica para software.

El sistema debe de haber pasado exitosamente las pruebas de integración y las pruebas Benchmark, el sistema en este punto se encuentra listo para enfrentarse a problemas similares a los que dieron origen al proyecto. Esta actividad provee:

- La revisión por parte del ET al Plan del Proyecto para la asignación de tareas.
- Comprensión del procedimiento de la validación del sistema.
- La documentación y verificación de la validación.
- Software validado e incorporado a línea base.

Las tareas para esta actividad están descritas en tabla 2.12, esta actividad fue anexada a las actividades mostradas en la ISO/IEC 29110 Perfil Básico, puede verificarse en las actividades mostradas en la figura 2.3, con respecto a la figura 3.2.

Tabla 2.12: Lista de Tareas de la Actividad Validación

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT PR	IS.6.1 Asignar tareas a los miembros del ET en relación a su rol, de acuerdo al Plan del Proyecto actual.	Plan del Proyecto. <ul style="list-style-type: none"> ■ Objetivos. ■ Tareas. 	

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
AP LT ET	IS.6.2 Comprender la aplicación del sistema a la resolución del problema original, establecer los criterios de aceptación de las soluciones obtenidas del sistema, verificar que el sistema este configurado para la realización de la validación.	<ul style="list-style-type: none"> ■ Problema original. ■ Componentes de software. [Probado, integrado] 	
PR	IS.6.3 Realizar la validación usando el problema original, en caso de ser necesario realizar la validación en presencia de los interesados.	<ul style="list-style-type: none"> ■ Problema original. ■ Componentes de software. [Probado, integrado] 	Resultados de la validación.
LT ET	IS.6.4 Documentar la validación y los resultados obtenidos. Analizar los resultados obtenidos con el fin de verificar que la solución obtenida es la adecuada y es satisfactoria para los interesados.	Problema original, resultados	Resultados [analizados].
PR	IS.6.5 Corregir los defectos encontrados y realizar una prueba de regresión hasta satisfacer el criterio de finalización.	<ul style="list-style-type: none"> ■ Componentes de Software. [Probados, integrados] ■ Validación. ■ procedimiento de prueba y resultados de validación. [actualizado] 	<ul style="list-style-type: none"> ■ Software. [corregido y validado] ■ Resultados de validación. [corregidos]

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
PR	IS.6.10. Incorporar las pruebas de integración, la validación, resultados de las pruebas realizadas, manual de mantenimiento y manual de usuario a la configuración de software.	<ul style="list-style-type: none"> ■ Pruebas de integración. ■ Pruebas Benchmark. ■ Validación. ■ Software. [validado] 	Configuración de software. <ul style="list-style-type: none"> ■ Pruebas de integración y resultados. ■ Pruebas Benchmark y resultados. ■ Validación y resultados. ■ Software [Validado].

IS.7 Entrega del Producto

En esta actividad la configuración de software esta integrada por el software [validado] y la documentación necesaria para su uso y despliegue. La actividad de Entrega del Producto consiste en la entrega del software, integrado, probado y validado al *Cliente* o a los interesados, con respecto a las instrucciones de entrega. La actividad provee:

- La revisión por parte del ET al Plan del Proyecto para determinar la asignación de tareas.
- La entrega del producto de *Software* y la documentación aplicable al software (Investigación teórica, especificación de requerimientos, arquitectura y Diseño Detallado de Software, pruebas de integración), de acuerdo a las necesidades del cliente o de los interesados.

Tabla 2.13: Lista de Tareas de la Actividad Entrega del Producto.

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
LT ET	IS.7.1 Asignar Tareas a los miembros del ET relacionadas con su rol, de acuerdo al Plan del Proyecto actual.	Plan del Proyecto <ul style="list-style-type: none"> ■ Objetivos. ■ Tareas. 	

Capítulo 2. Guía de uso de la Norma ISO/IEC 29110 Perfil Básico para el desarrollo de Software Científico

Rol	Lista de Tareas	Productos de Entrada	Productos de Salida
DIS	IS.7.2 Verificar que la configuración de software cuente con la documentación completa.	<ul style="list-style-type: none"> ■ Software. [validado] ■ Pruebas. ■ Especificación de requerimientos. [validado] ■ Diseño Detallado de Software. [validado] 	Configuración de software. [verificada]
LT	IS.7.3 Llevar a cabo la entrega del producto de acuerdo a las necesidades de los interesados (instalación del sistema generado, capacitaciones necesarias, CD de instalación, etc.)	Configuración de software. [verificado]	Configuración de Software. [Liberado]

Capítulo 3

Caso de estudio

3.1. Introducción

Para validar la Guía propuesta se presenta un caso de estudio utilizado para la adaptación de la norma ISO/IEC 29110 Perfil Básico presentado en el capítulo anterior, también se presentan las prácticas de software utilizadas en el proyecto seleccionado, además de la forma de trabajo y documentación generada del proyecto.

Se presentó la oportunidad de trabajar en conjunto con el grupo de cómputo científico del posgrado en Ciencia e Ingeniería de la Computación impartido por el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) de la Universidad Nacional Autónoma Nacional (UNAM), los cuales desarrollan y aplican modelos con el objetivo de predecir el comportamiento de los sistemas de la Ciencia y la Ingeniería [26].

3.2. Objetivo.

El objetivo es aplicar la Guía propuesta en un proyecto de Cómputo Científico para administrar el proyecto y generar un producto de software.

3.3. Metodología

Fue necesario trabajar en conjunto con un proyecto de cómputo científico, con el objetivo de entender las necesidades y problemas de este tipo de proyectos. Se realizó una retroalimentación del trabajo y se propusieron las modificaciones necesarias a las tareas de Norma ISO/IEC 29110 Perfil Básico, para adaptarse al caso de estudio.

3.4. Proyectos de investigación

Los proyectos de investigación realizados en el Posgrado en Computación en el área de Cómputo Científico, están realizados bajo la tutoría de investigadores expertos en áreas de investigación como:

- Análisis Numérico.
- Métodos de Optimización.
- Métodos Numéricos.
- Cómputo Científico.
- Programación Orientada a Objetos.
- Combinatoria.
- Métodos Numéricos y Computacionales para Ecuaciones Diferenciales Parciales.
- Matemáticas.
- Álgebra.

Las investigaciones realizadas se enfocan al estudio de los comportamientos de sistemas físicos, para la descripción de estos sistemas son utilizados modelos matemáticos, estos modelos en su mayoría están definidos por ecuaciones diferenciales ordinarias o parciales [16, 18, 28].

Doxygen

Para la documentación del código y las funciones generadas se utilizó la herramienta Doxygen. el cual es un generador de documentación para diversos programas. Es un programa que analiza el código en busca de directivas en forma de comentarios y las transforma en documentación, ya sea HTML, \LaTeX incluso se puede crear una página de manual para Linux [8].

Doxygen acepta multitud de lenguajes como C/C++ o Java. La principal ventaja de Doxygen es que las directivas no son más que comentarios de forma que no se tiene que crear una documentación aparte, basta con comentar el código, figura. 3.1, toda esta información se encuentra en [8].

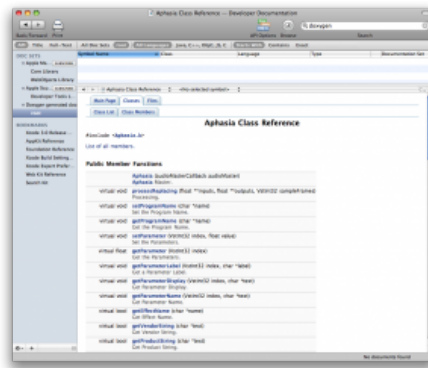


Figura 3.1: Herramienta Doxygen para documentar el código, imagen obtenida de [8].

3.5. Proceso de desarrollo de software

El desarrollo se basó en iterar las actividades propuestas por el proceso IS de la siguiente manera:

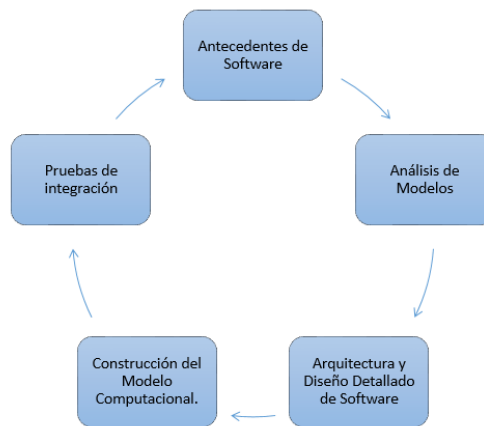


Figura 3.2: Las actividades iteradas durante la duración del Plan del Proyecto, tienen por objetivo al final de esa iteración el entregar una funcionalidad del software.

El ciclo propuesto en la figura 3.2 es con el fin de ir explorando los requerimientos e ir construyendo el sistema, ésto permitió que el sistema fuera evolucionando, cuando el sistema pasa satisfactoriamente las pruebas de integración se procede a las últimas dos actividades del proceso de IS.

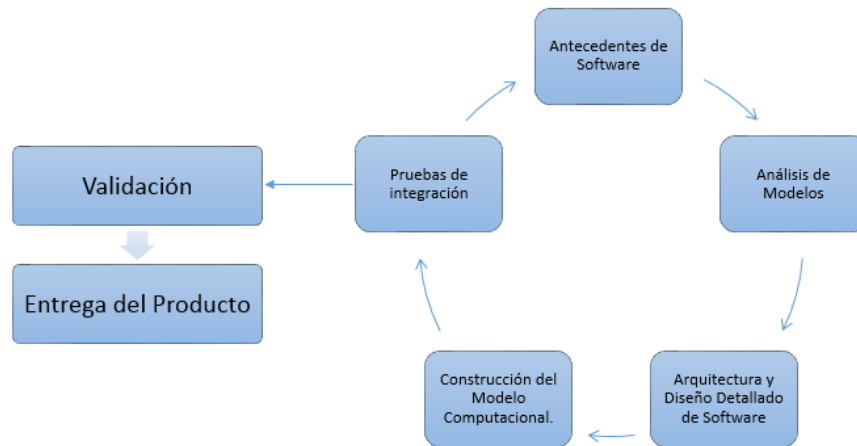


Figura 3.3: Se muestra el ciclo completo de las actividades realizadas, una vez que el sistema se encuentra completo se realizan las siguientes actividades.

Cada una de estas actividades son administradas, ejecutadas y evaluadas por las actividades del proceso de AP: Planeación del Proyecto, Ejecución del Plan del Proyecto y Evaluación y Control del Proyecto. Las últimas actividades relacionadas son las de Cierre del Proyecto y Entrega del Producto de los procesos de AP e IS respectivamente, toda la documentación del proyecto, esta disponible en la tesis del alumno, Mario Arturo Nieto Butron en su trabajo de investigación [21].

3.6. Proceso de Administración del Proyecto según la Guía.

El enunciado de trabajo como elemento de entrada del Proceso de AP a grandes rasgos fue el siguiente:

” Se desea realizar el análisis de sustentabilidad de proyectos de recuperación de energía geotérmica, debido a que estos tienen altas probabilidades de fracaso es usada la simulación numérica. Se busca implementar la simulación de sistemas fracturados geotérmicos en casos particulares donde la finalidad sea la extracción y administración de energía. Se pretende realizar un análisis numérico para establecer la factibilidad de un proyecto de recuperación.”

Este enunciado está de manera completa junto con la documentación en [21]. Para la realización del proyecto, se realizaron planes para cada iteración del ciclo mostrado en el diagrama 3.2, con el enunciado del proyecto se procedió a realizar el primer plan del proyecto, el cual está en el Apéndice C apartado C.1, este plan difiere de los subsecuentes planes, revisar el Apéndice C apartado C.2, el cual es el plan de la última iteración antes del cierre del Proyecto.

El Plan del Proyecto inicial consistió principalmente en actividades de investigación del método numérico seleccionado “*CVFEM*”. Los planes subsecuentes se encargaron de llevar un control de

las tareas y actividades a realizar, muchos de los objetivos que el ET se planteó estuvieron basados en realizar las actividades del Proceso de IS, estos objetivos nos permitieron definir las tareas a realizar durante la iteración actual¹.

Durante todo el proyecto se documentaron los riesgos encontrados, de los cuales presentamos los más importantes y las propuestas de plan de contingencia, tabla 3.1.

Tabla 3.1: Riesgos importantes durante la realización del proyecto.

Riesgos	Plan de contingencia
Abandono del proyecto.	Proporcionar la documentación y trabajo realizado.
Falta de financiamiento.	Ninguno.
Dudas.	Buscar asesoría entre compañeros y profesores de geofísica.
Complejidad del problema.	Ninguno.
Validación del trabajo realizado.	Realizar una investigación en nuevas fuentes de información.

Los riesgos en los cuales no se realizó la propuesta de un plan de contingencia, fue debido a la inexperiencia de nuestra parte como equipo de trabajo, posteriormente en la retroalimentación final se comentaron estos riesgos, sabemos que todo proyecto sin financiamiento llega a su finalización temprana, precisamos que este es un trabajo académico y el financiamiento proviene de la institución (Becas), por otro lado, la complejidad del problema puede provocar un abandono del estudiante en nuestra opinión personal a consecuencia de estrés u otros factores, la mitigación de este riesgo debe estar basado en asesorías e investigaciones más estructuradas en diferentes medios de información, estas observaciones fueron realizadas por el AP.

Se definieron reuniones con el AP, con el objetivo de validar el trabajo realizado y se obtuviera nuevo trabajo a realizar, además de las realizada por el ET para revisar el avance de las tareas a realizar y agregar nuevas tareas, en las cuales se discutieron todos los problemas encontrados, dudas surgidas durante el desarrollo de las tareas, todos estos elementos se trataron de registrar en un reporte de avance, figura 3.4.

¹Revisar el Apéndice C apartados C.1, C.2

Proyecto: Geotermia Aplicación de CVFEM Período Reportado: 02-05-16 al 13-05-16

Responsable: Equipo de Trabajo

1. Reporte de Actividades

Integrante	Actividad	Tiempo Total estimado (horas)	Estado	Observaciones
Integrante del equipo de trabajo 1.	Revisión y discretización de CVFEM	30hr	Iniciado	El modelo es extenso y complejo
Integrante del equipo de trabajo 1.	Mallado	4hr	Pospuesto	El mallado necesita del entendimiento de GMSH
Integrante del equipo de trabajo 2.	Revisión de UML	1hr	Iniciado	En revisión
Integrante del equipo de trabajo 1.	Comprensión de GMSH	10hr	Iniciado	Se analiza el tipo de archivo obtenido y su posible uso.
Integrante del equipo de trabajo 2.	Revisión y modificación del Plan del proyecto de la iteración.	1hr	Iniciado	La modificación está ligada al avance de los objetivos planeados.
ET	Reunión para presentación de avances.	1hr	Finalizado	La reunión se llevó acabo para comentar los inconvenientes teóricos observados y la revisión de tareas asignadas.
Total horas de trabajo		47hr		

Figura 3.4: Actividades del caso de estudio, segunda iteración.

Se muestra una parte del documento de la iteración 2, remarcamos que este documento es el reporte de avance del proyecto, las observaciones tiene el fin de registrar los motivos de no concluir dicha actividad o tarea, el registrar el tiempo dedicado fue útil en la toma de decisiones ante las desviaciones del proyecto. Durante el seguimiento del proyecto también se documentaron las tareas finalizadas en la figura 3.5, como se puede observar las tareas terminadas fueron pocas, hay que resaltar que la complejidad de estas tareas finalizó en iteraciones subsecuentes.

Reporte de Seguimiento
Fecha de creación del documento [13-05-16]
Página 2 de 3

2. Tareas finalizadas

Actividad y/o tarea	Integrante	Fecha de entrega planeada	Fecha real
Reunión ET	ET	13-05-16	13-05-16

Figura 3.5: Reporte de tareas de la iteración 2.

La figura 3.5 documenta una de las reuniones planeadas durante esa iteración, muchas de las tareas propuestas fueron abandonadas debido a que no contribuyeron al avance del proyecto. Uno de los elementos primordiales del proyecto fueron los cierres de cada iteración², en estos cierres se

²ver apéndice C apartado C.4

realizó una retroalimentación del trabajo realizado, tareas pendientes y cuales de estas se pretende seguir realizando, también se dieron propuestas de mejora sobre todos los elementos de trabajo del proyecto.

3.7. Problemas de la Administración de Software según la Guía.

A continuación se presentan los problemas particulares que surgieron a lo largo del proyecto:

1. No se homogenizó correctamente el conocimiento en general, producto de las investigaciones realizadas por cada uno de los integrantes del ET.
2. En el reporte de avance resultó complicado saber exactamente el tiempo dedicado a cada tarea, pero se detectaron tareas complejas que se pueden considerar para planes futuros.
3. Hubo confusión en ciertos elementos que no se comprendieron sobre el proceso aplicado.
4. Los cambios en el proyecto y software no fueron completamente documentados, debido a su sencillez, cantidad y nivel de impacto.
5. Cada actividad del proceso de implementación de software mostrado en el capítulo anterior contiene sus propias complejidades, principalmente en las actividades de pruebas y validación.

Los problemas aquí presentados son algunos de los más importantes, pero podemos mencionar ciertos aspectos de algunos de estos, por ejemplo en el punto 3, se menciona la confusión sobre el proceso utilizado, ya que hubo cierta ambigüedad en el entendimiento de tener una funcionalidad al final del periodo del Plan del Proyecto actual. El ET definió el concepto de funcionalidad como: Tener un componente de la arquitectura completo y funcional.

Las actividades pueden contener su propia complejidad debido a que hay que agregarle las dificultades de la resolución de un problema de ciencia combinada con la administración de dicho proyecto, la abstracción necesaria para realizar la transformación de las ecuaciones en el modelo computacional, en este proyecto resulto en dos meses de trabajo continuo, mientras que las pruebas del tipo Benchmarck agregan una complejidad mas al sistema.

3.8. Implementación de Software según la Guía.

La base teórica de todo el trabajo realizado fue plasmado en la tesis de maestría de Mario Arturo Nieto Butron en Cómputo Científico[21], en este trabajo se registran los resultados obtenidos del sistema generado y el contexto teórico del proyecto, la mayoría de la investigación de la teoría para el proyecto fue realizada durante la actividad de Antecedentes de Software, aunque también se

realizó un poco más de investigación cuando se presentaron dudas sobre el modelo computacional planteado.

Debido a que ya se contaba con el método de solución a implementar (método CVFEM), fue necesaria una discretización, todo el trabajo de investigación realizado, permite desglosar la solución en funcionalidades o componentes del sistema bien definidos.

En la *Especificación de Requerimientos*, Apéndice C apartado C.5, se documentan estas funcionalidades, las cuales son los requerimientos funcionales, mientras que las optimizaciones pasan a ser requerimientos no funcionales, entre las funcionalidades necesarias de manera general se tienen:

1. El sistema debe ser capaz de tratar el dominio de interés discretizado con gmsh³.
2. El sistema debe construir una matriz obedeciendo las expresiones matemáticas obtenidas con el CVFEM.
3. Construir un vector con ayuda del término fuente y de las condiciones de frontera.
4. Obtener la solución del sistema de ecuaciones lineales.
5. Visualizar el resultado obtenido de la ecuación gobernante.

Estas funcionalidades se definieron conforme las iteraciones fueron avanzando, inicialmente se tenía conocimiento de las funcionalidades 1 y 2, por lo tanto fueron unas de las primeras en construirse.

Los requerimientos no funcionales fueron:

1. Minimizar el uso de memoria en el almacenamiento de la matriz.
2. Utilización de técnicas de metaprogramación, en la construcción de la biblioteca de álgebra lineal.
3. Permitir un mantenimiento sencillo.
4. Verificar el rendimiento.

Estos elementos son los alcanzados en la construcción del sistema debido a que se deseaba realizar la implementación del sistema funcional, posteriormente resulta relativamente más sencillo realizar las optimizaciones de memoria, tiempo de ejecución, etc. El Apéndice C, documento de Especificación de Requerimientos C.5, muestra la documentación generada en la actividad de *Análisis de Modelos*.

Para la actividad de Arquitectura y Diseño Detallado de Software, la arquitectura propuesta fue basada en componentes, además aumentaba la posibilidad de hacer el sistema flexible al cambio,

³Revisar [21], para mas información sobre el sistema gmsh

mantenible y que sus componentes puedan ser modificados, también se proporcionó una normatividad base para la codificación. Para la propuesta de la normatividad se consideró el lenguaje de programación seleccionado, este elemento resultó importante debido a que todo el pseudocódigo es basado en reglas bien establecidas y lo hace más sencillo de comprender, se puede revisar el Apéndice C apartado C.6.

La primera propuesta de arquitectura estuvo compuesta de los siguientes componentes de la figura 3.6, basados en las funcionalidades anteriores, cabe resaltar que la arquitectura final puede diferir de la aquí presentada, debido a que el proyecto aún estaba en progreso al momento de la escritura de este documento.

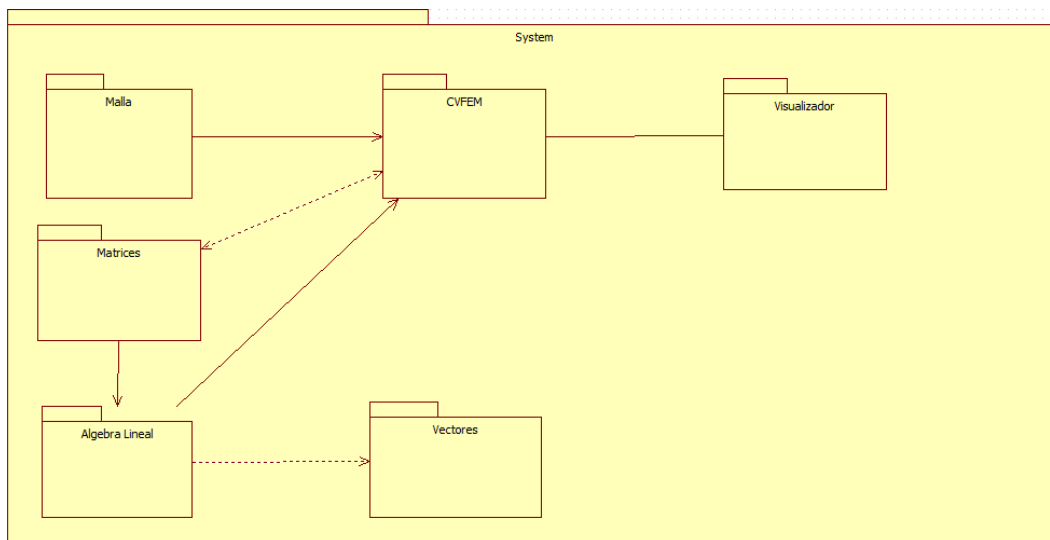


Figura 3.6: Arquitectura en su primera versión, la construcción estuvo basada en esta arquitectura.

Esta arquitectura sufrió modificaciones, hasta lograr tener la siguiente forma, la cual es la propuesta del alumno.

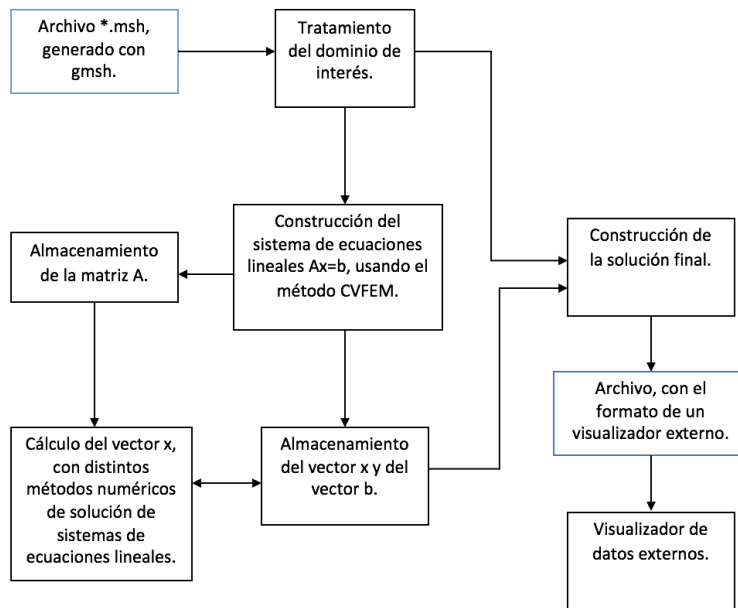


Figura 3.7: Arquitectura en su versión final el dibujo fue proporcionado por el alumno.

Durante la construcción del sistema se realizaron una cantidad considerable de pruebas, la mayoría de ellas fueron hechas con base en cálculos de manera manual o con el apoyo de un software para los cálculos, con el fin de tener la certeza de que el componente construido realice los cálculos solicitados de manera correcta.

La descripción de los componentes de software, contiene los elementos de la arquitectura propuesta, describiendo la implementación de los requerimientos del sistema, la herramienta de Doxygen resultó indispensable en esta parte, ya que permitió documentar las funciones generadas de manera automática.

Resuelve problemas usando el CVFEM	
Main Page	Classes ▾ Files ▾
Class List	
Here are the classes, structs, unions and interfaces with brief descriptions:	
Add	
BICGSTAB	Se implementa el método numerico BICGSTAB para la solución de sistemas de ecuaciones lineales. En este caso sirve para la matriz de tipo CSR
CG	Se implementa el método numerico CG para la solución de sistemas de ecuaciones lineales. En este caso sirve para la matriz de tipo CSR
COO	El CVFEM devuelve matrices con un gran contenido de ceros en sus entradas que no son significativas en su cálculo por lo tanto se requieren utilizar esquemas de almacenamiento, que consideren únicamente la información necesaria, en este caso se implementa el esquema COO. El esquema COO requiere de tres contenedores: data row col en data se almacenan los componentes no cero de la matriz, en row se almacenan las coordenadas i de la matriz, y finalmente en row son almacenadas las coordenadas j. En este caso se usan como contenedores arreglos estáticos, por tal motivo el usuario debiera de indicar de manera aproximada la cantidad de componentes totales que almacenara una matriz, a su vez el sistema multiplicara por 30 esta aproximación de manera que sea poco factible que se lleve
CSR	El CVFEM devuelve matrices con un gran contenido de ceros en sus entradas que no son significativas en su cálculo por lo tanto se requieren utilizar esquemas de almacenamiento, que consideren únicamente la información necesaria, en este caso se implementa el esquema CSR. El esquema requiere CSR de tres arreglos: data col row en data se almacenan los elementos reales no cero de la matriz, en col se almacenan la coordenada j de la matriz y en row se almacenan los indices de inicio y fin de cada renglón del arreglo
CVFEM	Construye la matriz A y el vector b del sistema de ecuaciones $AX=b$
GaussSeidel	Se implementa el método numerico Gauss-Seidel para la solución de sistemas de ecuaciones lineales. En este caso sirve para la matriz de tipo CSR
ICHOL	
ILU	
Jacobi	Se implementa el método numerico Jacobi para la solución de sistemas de ecuaciones lineales. En este caso sirve para la matriz de tipo CSR
JacobiP	
Mesh	Los objetos tipos mesh son contenedores para el dominio de interes generado con la herramienta gmsh
MILU	
Mult	
MV	
Solve	
SOR	Se implementa el método numerico SOR para la solución de sistemas de ecuaciones lineales. En este caso sirve para la matriz de tipo CSR
Sub	
Timer	
Vector	Clase para dar tratamiento a los vectores

Generated by [doxygen](#) 1.8.12

Figura 3.8: Pagina principal de la pagina de html, generada con Doxygen a partir de código fuente del sistema generado,.

La manera en que se realizó la actividad de Pruebas de Integración fue en dos etapas, la primera consistió de la integración de los componentes, debido a que los componentes de software tienen funcionalidades específicas y en la actividad de construcción cada funcionalidad fue probada, se realizaron pruebas de caja negra, se alimentaron los paquetes con datos de entrada conocidos y esperaron resultados de salida conocidos Las pruebas realizadas se encuentran de manera completa en [21], debido a necesidades del alumno la documentación de las pruebas realizadas difieren de la plantilla propuesta.

La segunda etapa de pruebas son las pruebas bien establecidas o puntos de referencia Bechmark, una vez que el sistema esta integrado por todos los componentes, realizamos este tipo de pruebas para verificar que el sistema cumple con los requerimientos [23]. Estos puntos de referencia son problemas bien estudiados en la literatura científica, para los cuales se conoce la solución exacta y pueden ser usados para calibrar el código. Las pruebas se realizan por simple comparación. En este punto serán detectados todos los errores que no fueron encontrados por la primer etapa y las pruebas realizadas en la construcción.

Para el sistema se realizaron 3 pruebas proporcionadas por el AP, los resultados obtenidos por el alumno [21] fueron validados por el AP.

Las pruebas que a continuación se presentan son tomadas directamente del trabajo de investigación realizado por el científico⁴, se utilizó la plantilla mostrada en el apéndice B apartado B.4, para revisar completamente las pruebas realizadas y sus resultados consultar [21].

⁴Se ha excluido parte de las observaciones debido a que no es objetivo de este trabajo hablar del trabajo completo de investigación realizado.

3.9. Prueba de Laplace en dos dimensiones

A continuación presentamos dos de las pruebas realizadas por el alumno Mario Arturo Nieto Butron, con el fin de ejemplificar el uso de las plantillas presentadas en la plantilla B3, la información completa se encuentra en [21].

3.9.1. Descripción de la prueba

Nuestro primer problema consiste en determinar la solución de la ecuación de Laplace en dos dimensiones la cual esta descrita por:

$$-\Delta T(x, y) = 0 \quad (3.1)$$

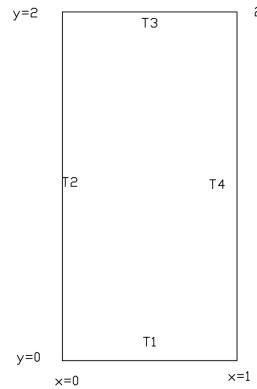


Figura 3.9: Dominio de interés para la ecuación de Laplace

El dominio $\Omega = (0, 1) \times (0, 2)$, es un rectángulo (Ver Fig. 3.9), las condiciones de frontera están dadas por:

$$T_1 = 100, T_2 = T_3 = T_4 = 0$$

La solución analítica del problema es:

$$T(\underline{x}) = T_1 \left[2 \sum_{n=1}^{\infty} \frac{1 - (-1)^n}{n\pi} \frac{\sinh\left(\frac{n\pi(H-y)}{L}\right)}{\sinh\left(\frac{n\pi H}{L}\right)} \sin\left(\frac{n\pi x}{L}\right) \right]$$

3.9.2. Descripción de Resultados

En la figura 3.15, se muestra la solución exacta, la solución aproximada y la distribución del error. Nótese que el mayor problema en la solución lo encontramos en las esquinas, para reducir el error en estos puntos podemos hacer más fina la malla en estos lugares problemáticos.

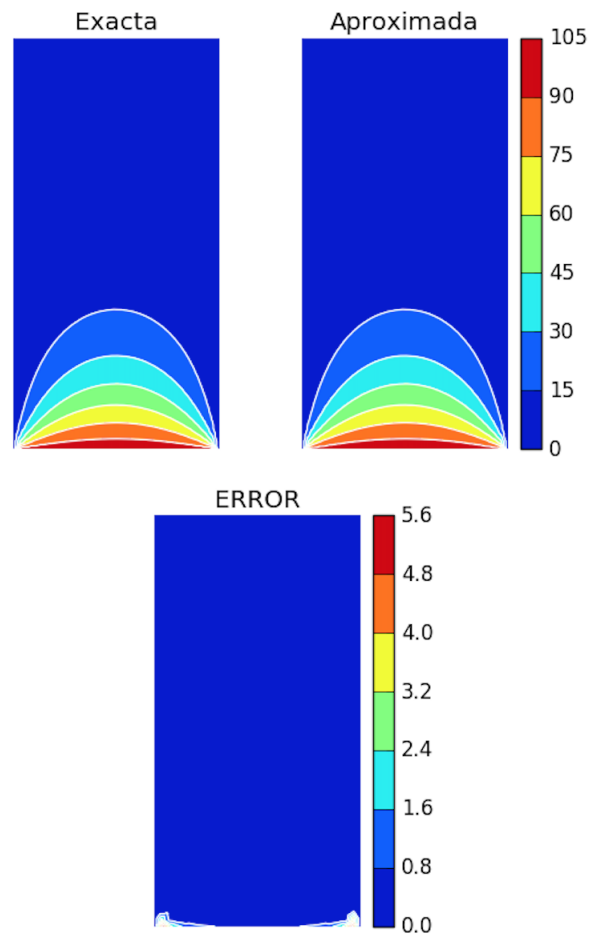


Figura 3.10: Solución para la ecuación de Laplace en estado estacionario

3.9.3. Observaciones

La malla utilizada en esta prueba puede observarse en la figura 3.11, la cual esta compuesta por 1740 nodos y 3482 elementos.

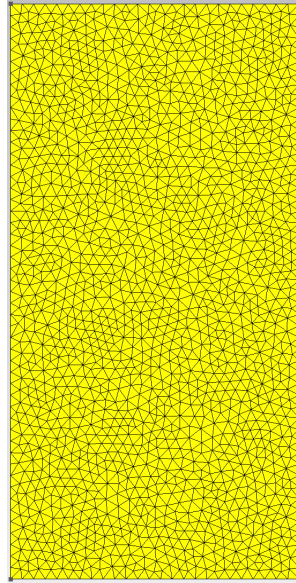


Figura 3.11: Malla para la ecuación de Laplace, generada con la herramienta gmsh.

Para codificar la solución a la ecuación de Laplace usando SYGS, se requiere darle un tratamiento al dominio de interés, para éste fin se declara un objeto de la clase Mesh, el cual recibe un archivo con la discretización del dominio y para identificar fácilmente los nodos frontera de los nodos interiores en un dominio rectangular se requieren sus vértices en código:

```
// Definimos los vertices del dominio
std::vector<double> vertices;
vertices.push_back(0.0);
vertices.push_back(0.0);
vertices.push_back(1.0);
vertices.push_back(2.0);
// Definimos un objeto para el tratamiento de la malla
Mesh mesh("./GMSH/Laplace/mesh10.msh", vertices);
```

Una vez que se genera el objeto mesh tiene como tareas: la lectura del archivo, el reordenamiento de los nodos, la identificación de los nodos frontera, identificación de los elementos soporte para un nodo y del cálculo del área de cada elemento triangular. Es necesario indicar para los nodos frontera si son: Dirichlet o Neumann y que valor tienen. Se utiliza el método `bondaryCondition(int type, double val)`, si `type` es 0 entonces tratamos con un nodo tipo Dirichlet y si es 1 con un nodo tipo Neumann. En éste caso se llaman a los nodos de frontera y se recorren uno a uno, asignado su valor y su tipo según las restricciones del problema, en código:

```
// Cargamos las condiciones de frontera
std::vector<int> border = mesh.bordeNode(); //Nodos frontera
```

```
for (int i = 0; i < border.size(); ++i)
{
    if (mesh.coordY(border[i]) == 0
        && (mesh.coordX(border[i]) != 0
            && mesh.coordX(border[i]) != 1))
    {
        mesh.boundaryCondition(0,100);
    }
    else
    {
        mesh.boundaryCondition(0,0);
    }
}
```

Según la teoría, en los nodos tipo Dirichlet conocemos la solución exacta del problema y podemos no incorporarlos como incógnitas en la matriz de transmisibilidad. Por otro lado los nodos tipo Neumann son lugares en donde no conocemos con precisión la solución del problema, por lo tanto deben ser considerados en la matriz de transmisibilidad. Por lo cual, un paso importante es determinar, cuantos y cuales nodos incógnita tendremos, para asignar los recursos de memoria suficientes, en código:

```
mesh.nodeIncongnite(); // Nodos Incognita
CSR A(mesh.nnodeIncongnite()); // Matriz A del sistema
Vector b(mesh.nnodeIncongnite()); // Vector b del sistema
Vector x(mesh.nnodeIncongnite()); // Vector x del sistema
```

Se construye el sistema de ecuaciones lineales, según el método CVFEM:

```
// Objeto para crear el sistema de ecuaciones
CVFEM cvfem;
cvfem.Ab(A,b,mesh); // Construye A y b
```

Definimos un objeto de la clase BICGSTAB, para hacer uso del método del gradiente conjugado estabilizado, el cual recibe el sistema de ecuaciones a resolver:

```
// Resolvemos el Sistema de ecuaciones
BICGSTAB bicg;
bicg.solve(A,x,b); // BICGSTAB
bicg.report(); // Muestra un reporte
```

Se calcula la solución exacta, la solución aproximada total y las distribución del error las cuales se almacenan en un archivo de texto:

Se almacena la solución aproximada y la exacta (en caso de contar con ella) en un archivo de texto:

```
// Calculamos la solucion exacta
Vector exacta(mesh.nNode()); // Solucion exacta
solver.laplace2D(mesh,"laplace2DE.txt",exacta);
// Vector b para la solucion final
Vector x1(mesh.nNode());
// Almacena la distribucion del error
Vector error(mesh.nNode());
// Almacenamos la solucion aproximada con CVFEM
solver.solution(mesh,x,x1);
solver.storage(mesh,x1,"laplace2DA.txt");
// Calculamos la distribucion del error
error = exacta - x1;
solver.storageE(mesh,error,"error.txt");
```

En la gráfica 3.12 se muestra el calculo del residuo durante cada iteración realizada por el gradiente conjugado y el gradiente conjugado preconditionado con ILU. En la gráfica 3.13 se muestra el calculo del residuo durante cada iteración realizada por el gradiente conjugado estabilizado y el gradiente conjugado estabilizados preconditionado con ILU.

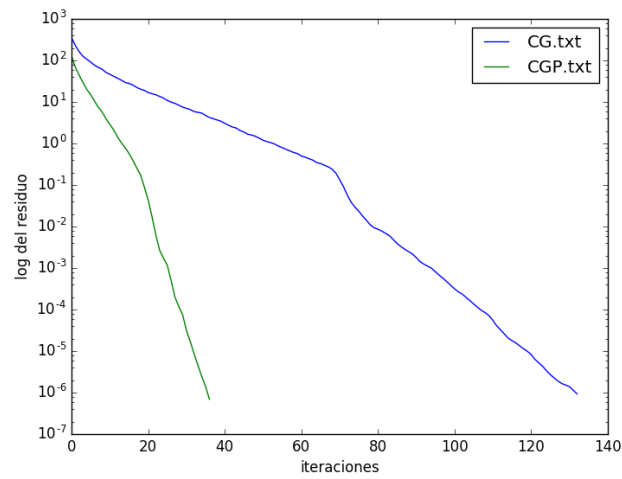


Figura 3.12: La gráfica muestra el residual $\log |r_n|$ vs n , para el método del gradiente conjugado (azul) y el método del gradiente conjugado preconditionado con ILU (verde).

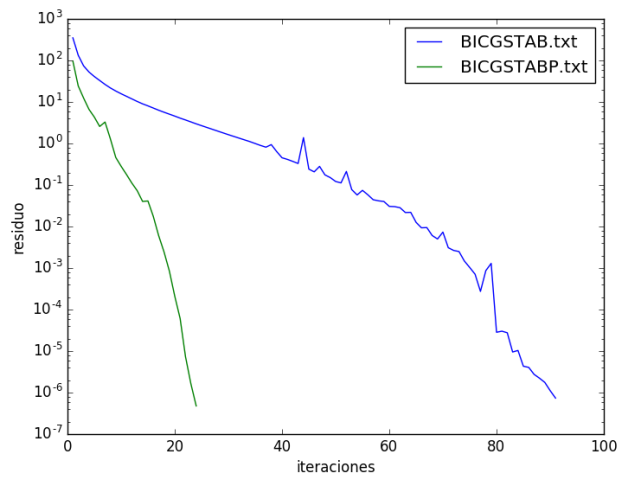


Figura 3.13: La gráfica muestra el residual $\log |r_n|$ vs n , para el método del gradiente conjugado estabilizado (azul) y el método del gradiente conjugado estabilizado preconditionado con ILU (verde).

3.10. Prueba de Laplace en un semicírculo

3.10.1. Descripción

El CFVEM es utilizado en la solución de problemas con dominios complicados, es decir dominios que presentan una geometría no rectangular. En éste contexto determinamos la dirección de flujo en un medio homogéneo en un dominio acotado por dos semicírculos, la ecuación de Laplace (ec. 3.1) describe este fenómeno.

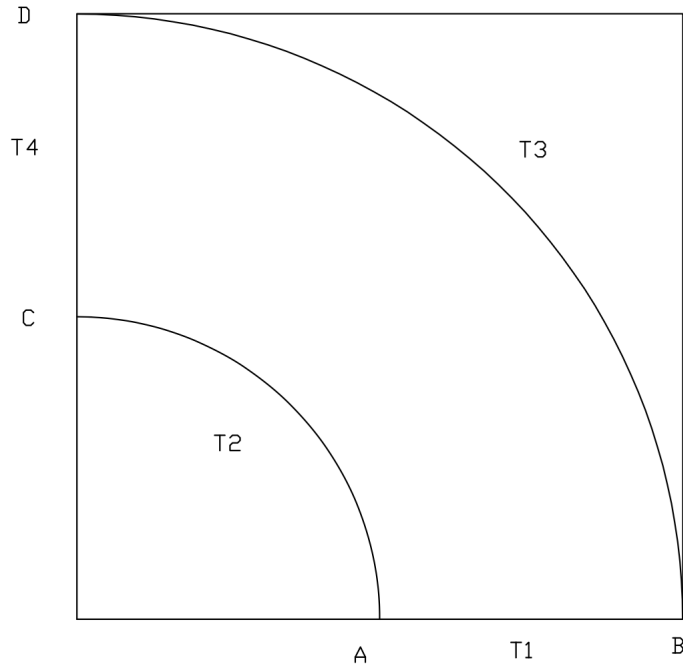


Figura 3.14: Dominio acotado por dos semicírculos para la ecuación de Laplace.

Condiciones de frontera:

$$T_1 = 0, T_2 = \frac{\sin(\theta)}{R_{0A}}, T_3 = \frac{\sin(\theta)}{R_{0B}}, T_4 = \frac{1}{r}$$

Solución analítica:

$$T = \frac{\sin(\theta)}{r}$$

3.10.2. Resultados

Se genera una circulación del flujo en el interior del dominio de interés, según la geometría del problema. En la figura 3.15, se muestra la solución exacta, la aproximada y la distribución del error.

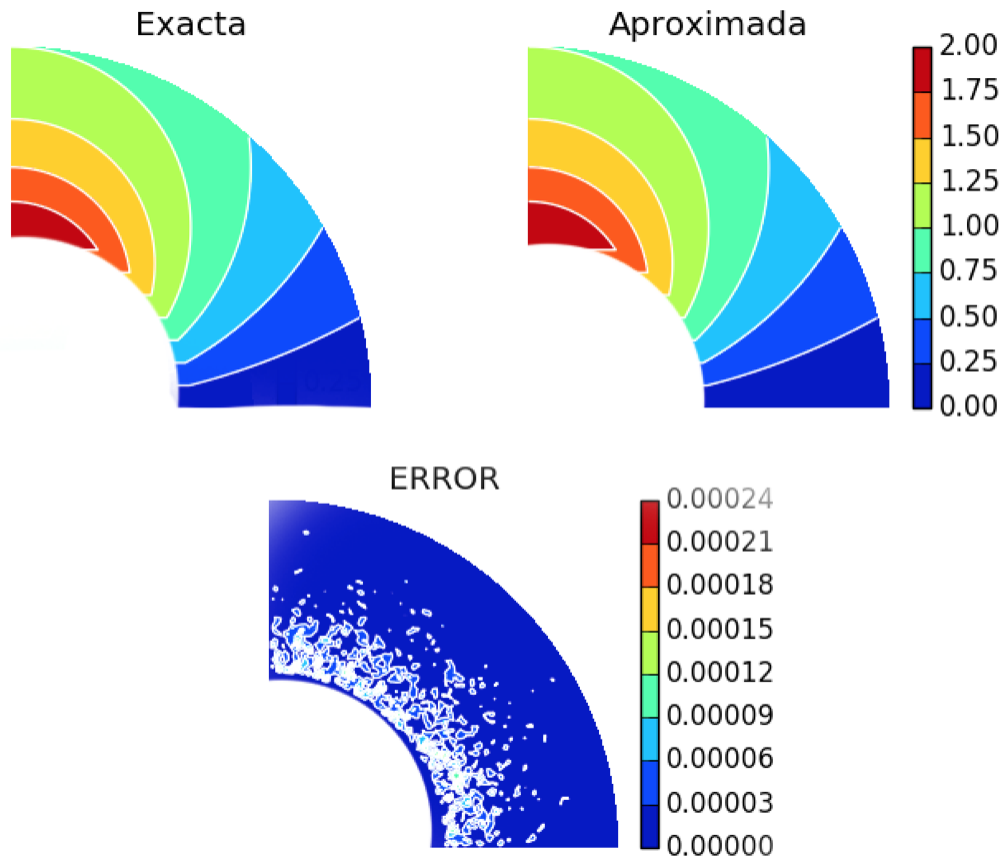


Figura 3.15: Solución para la ecuación de Laplace en un semicírculo.

3.10.3. Observaciones

La malla utilizada en esta prueba puede observarse en la figura 3.16, la cual tiene 2307 nodos y 4615 elementos.

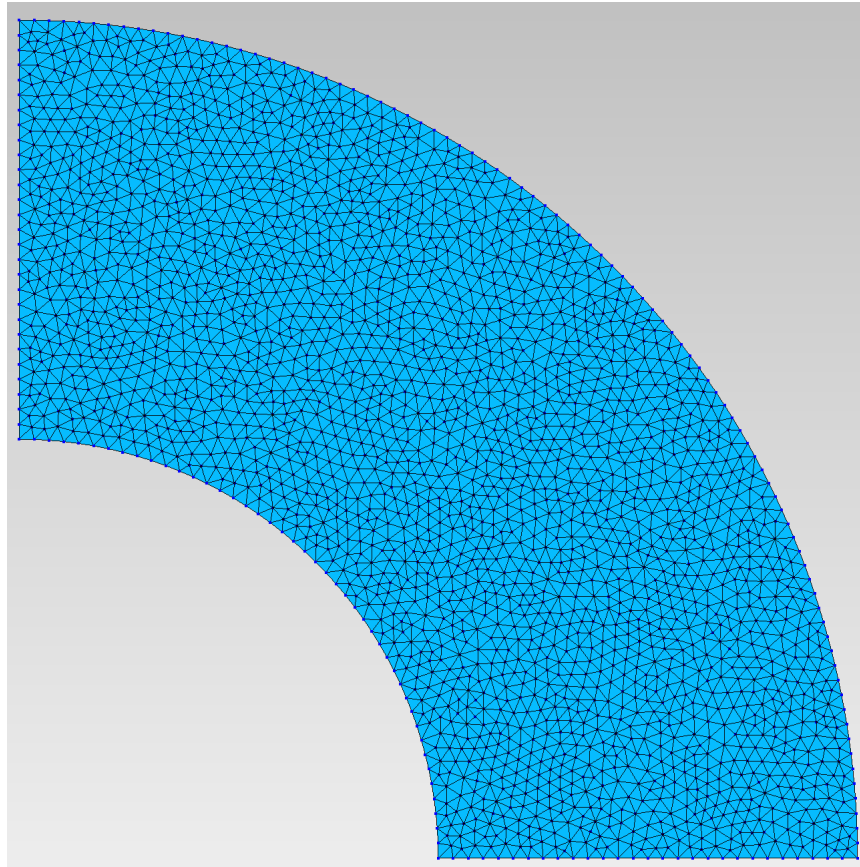


Figura 3.16: Malla para la ecuación de Laplace en un semicírculo.

La codificación resulta ser muy similar al problema anterior, llamamos otro constructor que permita el tratamiento del semicírculo, de esto:

```
// Definimos los radios de los semicírculos
double ra = 0.5; // Radio externo
double rb = 1.0; // Radio interno
// Definimos un objeto para el tratamiento de la malla
Mesh mesh("./GMSH/Laplace/mesh10.msh",ra,rb);
```

Como podemos observar, la plantilla propuesta para las pruebas permite que el científico documente partes fundamentales de las pruebas realizadas, tales como: la descripción de la prueba, resultado y observaciones; éstos tienen la finalidad de posteriormente poder reproducir las pruebas realizadas.

Por otro lado, si observamos el apéndice C apartado C.7 con respecto a las plantillas propuestas en el Apéndice B apartado B.3, difieren debido a que la información que se deseaba registrar era

más amplia, pero esta platilla sirvió de base para hacer el registro de las pruebas de integración de paquete.

3.11. Problemas durante la implementación de Software segun la Guía.

En este proyecto fue indispensable el proceso de AP debido a que lleva el control de todas las actividades a realizar, también nos permitió la toma de decisiones ante los problemas presentados durante la construcción de sistema con base en el trabajo realizado, todos los elementos de este proceso definieron muchos de los objetivos y tareas a realizar en cada una de las iteraciones realizadas.

Los problemas aquí presentados son específicamente al desarrollo del sistema en cuestión, los problemas a los que se presentó el alumno fueron básicamente los siguientes:

1. La estimación de errores de la solución obtenida contra la solución conocida.
2. Tiempos de validación del trabajo realizado.
3. Pruebas de Integración.

Durante las pruebas de integración Benchmark, se presentó el problema de obtención de errores en la solución obtenida, este problema consistió en que los errores obtenidos crecían conforme el número de nodos aumentaba, esto desconcertó al alumno sobre el trabajo realizado, se verificaron todos los elementos construidos, funcionalidades y cálculos realizados, se verificó que estuvieran correctos y no se logró encontrar ningún problema, se revisó la parte teórica de CVFEM para verificar al elemento en la teoría no considerado. El problema encontrado no fue dentro del trabajo realizado, sino a la elección del método de cálculo del error, la verificación de este problema fue solucionada gracias a las opiniones del tutor y experto del área.

El beneficio del análisis de errores y pruebas, fue que se pudo realizar una segunda versión del sistema SYSG con una perspectiva diferente, el desarrollo fue realizado en un tiempo mucho menor al sistema inicial.

Un problema que también creemos importante mencionar es con respecto a los documentos generados en este proceso, hay que tener muy en claro que estos documentos sufrirán modificaciones en todos sus elementos, los documentos que resultaron primordiales fueron: Especificación de Requerimientos y Arquitectura y Diseño Detallado de Software. Ya que permitieron al alumno basar toda la construcción en los elementos establecidos en estos documentos, si bien algunos de ellos sufrieron modificaciones, éstas fueron basadas en la percepción y experiencia que el alumno obtuvo.

Cabe mencionar que el alumno se adaptó a las plantillas propuestas específicamente para Pruebas Benchmark, mientras que para la plantilla de Pruebas de Integración, él realizó las modificaciones a esta plantilla con base en sus necesidades, los elementos propuestos con respecto a la

modificación realizada no difieren, estableciendo los elementos de forma textual en lugar de tener estos elementos en el formato de una tabla. La actividad de validación ha sido excluida debido a que es un elemento descrito completamente en el trabajo de investigación realizado.

Conclusiones

El objetivo de este trabajo de investigación, fue adaptar las tareas contenidas en la Norma ISO/IEC 29110 Perfil Básico con el fin de obtener una Guía para los equipos de desarrollo de software científico, esta Guía pretende reducir los problemas que tienen los proyectos y los sistemas generados por y para la comunidad científica.

Se trabajó de manera conjunta con el proyecto de investigación presentado en el Capítulo 3 [21] para proponer la documentación necesaria con elementos estipulados por la Propuesta de Guía basada en la norma ISO/IEC 29110 Perfil Básico, de tal forma que a los desarrolladores se les permita administrar, evaluar y controlar el ciclo de vida del proyecto, además de la documentación concerniente al producto de software generado.

La Guía dirigida a la comunidad científica que desarrolla software contiene los elementos que se han considerado esenciales y que en la experiencia del caso de estudio se presentaron. La validación realizada al proceso de AP muestra ciertos elementos que fueron bien recibidos, los cuales tratan de cubrir las complejidades que este tipo de proyectos tiene, como pueden ser, requerimientos propensos a cambios, compromiso por los integrantes del equipo de desarrollo, visualización y análisis de los riesgos y desviaciones que se puedan tener en el proyecto, observamos que con el proceso de AP, el trabajo fue constante y se elimina la ambigüedad que se tiene al realizar estos proyectos. Las actividades de ejecución, Evaluación y Control del Proyecto permitieron al investigador, tomar decisiones sobre el trabajo necesario a realizar o proponer nuevas formas de trabajo, se pudo observar un análisis constante del trabajo realizado.

La opinión recibida por parte del Administrador del Proyecto y el Equipo de Trabajo fue fundamental para este trabajo, debido a que realiza recomendaciones para la administración, en proyectos posteriores y establece la realización de tareas de manera ordenada, además de almacenar información importante que puede ser utilizada en proyectos subsecuentes, y no encontró adecuado el excluir el alcance del sistema a generar [21]. El DR. Luis Miguel de la Cruz, Administrador del Proyecto propuso el realizar un sistema que permita al investigador almacenar la información y que se generen los documentos propuestos en los Apéndices A y B, este sistema debe de tener el propósito de facilitarle al investigador el documentar y almacenar la información histórica en una base de datos.

Por otra parte, en el proceso de IS la propuesta de modificar los nombres de las actividades ori-

ginales por aquellos que la Guía presenta, permitieron que los científicos se sintieran identificados, son elementos con los cuales han trabajado constantemente, la validación de la Guía propuesta y los documentos relacionados con el proceso de IS, permitieron al desarrollador hacer la documentación del sistema, lo cual era uno de los propósitos de este trabajo.

Si bien, la parte más compleja ha sido la actividad de pruebas, los documentos generados fueron bien recibidos, por la sencillez que estos presentan y la información a ser registrada en ellos creemos fue la suficiente. Esta Guía puede ser un opción base para la estandarización de un modelo de proceso de desarrollo para la comunidad científica que desarrolla software.

Mario Nieto Butron, ha comentado que las actividades propuestas en la Guía son entendibles para desarrolladores con poca experiencia, pero que aun se puede agregar una actividad más, también agregó que hace falta un documento para el desarrollador que permita llevar un registro claro de los problemas encontrados durante el desarrollo⁵ y registrar la solución a dicho problema.

Otras de las opiniones a rescatar por parte del ET y el AP fue sobre los roles establecidos por la ISO/IEC 29110 Perfil Básico y las competencias de cada uno de estos, el definir roles al ET los hizo sentir poco preparados en comparación a todas las competencias solicitadas, ésto provoca que uno se haga la pregunta ¿Qué tan preparado debe de estar uno para desempeñar dicho rol?. La respuesta que establecimos es: *La Guía es dirigida a la comunidad científica con poco conocimiento en Ingeniería de Software, pero el establecer roles debe tener en los integrantes del equipo el deseo por querer desempeñar dicho rol de la mejor manera posible, basará la investigación en las habilidades necesarias para completar el proyecto*⁶.

Estas opiniones nos permiten establecer que la Guía propuesta es un buen inicio

⁵estos problemas son de carácter matemático

⁶Conocimientos del problema científico a resolver y practicas de Ingeniera de Software correspondientes.

Trabajo a Futuro

Como se pudo constatar en el trabajo realizado aún queda mucho por realizar, por estas razones el trabajo a futuro se puede dividir en los siguientes puntos:

- Realizar una investigación de campo para poder encontrar diferencias entre los equipos de desarrollo de software científico, se propone realizar esta investigación en empresas que realicen software científico y los proyectos de investigación académica.
- Realizar un estudio detallado de las pruebas a realizar para software científico.
- Poner en práctica la Guía modificada en diversos proyectos, con el fin de mostrar su eficiencia y posibles fallas que logren mejorar el trabajo realizado.
- Presentar artículos de investigación los cuales describan la Guía presentada, mostrando su aplicación en algún proyecto, esperando ser publicado en algún foro o revista especializada.
- Realizar una Guía especializada para la comunidad científica, donde se contengan los elementos que pueden ser utilizados en el desarrollo de software: herramientas CASE con tutoriales de uso y definiciones para el desarrollo de la Ingeniería de Software. La Guía tendría el objetivo de promover el uso de la Ingeniería de Software en proyectos de investigación que desarrolle Software Científico, esperando eliminar sistemas sin documentación.

Apéndice A

Lista de Plantillas del Proceso de Administración del Proyecto

A.1. Plantilla para el Plan del Proyecto

1.Introducción

[La introducción ofrece un resumen del contenido del documento y del área de estudio del proyecto.]

2.Enunciado de Trabajo

2.1 Objetivos Generales

[En esta sección se documenta el enunciado de trabajo del proyecto en cuestión, se consideran las metas que se esperan alcanzar con los productos de software terminados.]

3. Objetivos y Tareas

3.1 Objetivos

[Objetivos que se desean alcanzar al final del tiempo asignado al Plan del Proyecto, estos pueden abarcar desde investigación hasta la construcción de software o entrega del producto.]

ID	Objetivos
1	<i>Nombre del Objetivo</i>

Tabla A.1: Lista de objetivos a completar.

3.2 Tareas

[Las tareas registradas pueden ayudar al cumplimiento de los objetivos.]

ID	Funcionalidad o Actividad	Responsable
1	<i>Nombre Identificador de la Actividad u objeto</i>	
	<i>1.1 Tarea a realizar</i>	<i>Nombre del responsable de la tarea</i>

4. Roles

[Listado de roles que ejercerán los miembros del equipo de trabajo, se adjuntan las responsabilidades de cada uno.]

Nombre	Rol	Responsabilidades
<i>[Nombre del miembro del equipo de trabajo]</i>	<i>Rol a desempeñar</i>	<i>[La definición de las responsabilidades del rol están definidas en la tabla de roles 1.3]</i>

Tabla A.2: Lista de roles y responsabilidades de los miembros del equipo de trabajo.

5. Riesgos

[Listado de los riesgos identificados que podrían afectar el desarrollo del proyecto en caso de ocurrencia.]

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
...	...	<i>[Descripción de las medidas a tomar en caso de que se presente]</i>	<i>[Alto/ Medio/ Bajo]</i>	<i>[Identificado/ Controlado/ Mitigado]</i>

Tabla A.3: Tabla de riesgos.

6. Control de versiones

[Método para administrar la configuración del sistema, se debe incluir un mecanismo para la realización de los cambios así como la estructura y reglas de uso del repositorio.]

Estrategia de control de versiones	
Identificación de la configuración	<i>[Establecer lo que será almacenado en el repositorio, suministrar la línea base para clasificar productos y documentos]</i>
Estado de la configuración	<i>[Información de los productos que están en el repositorio]</i>
Responsable	<i>[Establecer quién verificará que se ha cumplido con los acuerdos y cambios a realizar]</i>
Políticas	<i>[Acuerdos para el uso del repositorio, establecer quién puede almacenar, modificar y acceder al repositorio]</i>

Tabla A.4: Tabla de la estrategia de control de versiones.

7. Repositorio

[Ubicación seleccionada para el repositorio.]

Repositorio	
Clasificación	<i>[Distribuida / Centralizada]</i>
Herramienta a utilizar	<i>[CVS / Subversión, GIT/ Github, Bitbuquet]</i>
Ubicación	<i>[Dirección de donde se encuentra el repositorio]</i>
Estructura	<i>[Descripción de la estructura elegida para la organización del repositorio, árbol de directorios]</i>

Tabla A.5: Tabla de las características y ubicación del repositorio.

A.2. Plantilla para el Reporte de Seguimiento

Proyecto: [Nombre del proyecto a reportar]

Periodo a Reportar: [Indicar de manera clara el periodo de tiempo que abarca este reporte. Del [dd-mm-aa.] al [dd-mm-aa.]]

1. Reporte de Actividades

[Listado de todas las actividades y objetivos realizados por el equipo durante el periodo de tiempo reportado.]

Integrante	Actividad u Objetivo	Tiempo Total	Estado	Observaciones
		[El total se obtiene del producto de las horas trabajadas por persona, éste valor es una aproximación]	[Finalizado/ Iniciado/ Pendiente/ Retrasado.]	[Se describen los problemas observados u observaciones importantes]
Total de Horas trabajadas		[Suma total de horas]		

Tabla A.6: Tabla de los Objetivos y actividades a realizar.

2. Tareas finalizadas

[Registro de tareas que han finalizado en el momento de la realización del reporte.]

Actividad y/o Tarea	Integrante	Fecha de entrega	Fecha real
[Nombre de la tarea realizada]	...	[dd-mm-aa]	[dd-mm-aa]

Tabla A.7: Tabla de tareas finalizadas.

3.Productos

[Listado de todos los productos de software realizados por el equipo y su estado actual.]

ID	Producto	Estado
1	<i>[Nombre que permita identificar al producto construido o en construcción durante el periodo actual.]</i>	<i>[Iniciado/ Terminado/ Postergado]</i>

Tabla A.8: Tabla de Productos.

4.Cambios

[Registro de los cambios necesarios tanto en documentación como en los productos de software generados.]

ID	Producto	Descripción	Solicitante	Estado
1	<i>Nombre del solicitante con el objetivo de llevar un control de cambios</i>	<i>Realizado/ Pendiente</i>

Tabla A.9: Tabla que lleva el registro de cambios a realizar.

5.Riesgos

[Listado de los riesgos identificados que podrían afectar el desarrollo del proyecto en la iteración actual en caso de ocurrencia.]

Nombre	Descripción	Plan de contingencia	Impacto	Estado del riesgo
<i>Nombre con que se identifica al riesgo</i>	<i>Descripción del riesgo.</i>	<i>Plan de contingencia en caso de que se presente el riesgo</i>	<i>Alto / Medio / Bajo</i>	<i>Identificado/ Controlado/ Mitigado</i>

Tabla A.10: Tabla de riesgos nuevos y que se han manifestado.

6.Resumen

[Es conveniente resumir las mediciones tomadas en el periodo de tiempo que abarca el seguimiento, para ello se proponen tablas que incluyan la información más relevante de una manera breve y clara.]

Actividades u Objetivos	Cambios	Riesgos
A tiempo:	Solicitados:	Encontrados
Retrasadas:	Rechazados:	Resueltos
Adelantadas:	Realizados:	Postergados
Postergadas:	Postergados:	

Tabla A.11: Resumen de actividades.

A.3. Plantilla para la Solicitud de Cambio

Proyecto	<i>[Nombre del proyecto]</i>
Requerimiento a cambiar	<i>[Especificar el requerimiento a modificar dentro del proyecto]</i>

Descripción

[En esta parte del documento se ingresará la información del cambio, en caso de que se necesite una petición de cambio más elaborada, esta puede ser modificada de acuerdo a las necesidades del proyecto.]

Solicitante	Nombre: <i>[Identificar la(s) persona(s) solicitante(s) del cambio.]</i>
Propuesta de cambio	<i>[Descripción del cambio solicitado.]</i>
Justificación	<i>[Describir el posible impacto/consecuencias de realizar o no el cambio solicitado.]</i>
Impacto y análisis en el producto o proyecto	<i>[Descripción de las ventajas del cambio solicitado o justificación de la necesidad del cambio.]</i>
Alternativas	<i>[Describir opciones de solución, si corresponde y existen.]</i>
Aceptado/ No aceptado	
Fecha de recepción: <i>[mm-dd-aa]</i>	Fecha límite de respuesta <i>[mm-dd-aa]</i>

A.4. Plantilla para el Cierre

1. Resumen

[Es conveniente resumir las mediciones registradas durante la ejecución del Plan del Proyecto, para ello se proponen tablas que incluyan la información más relevante de una manera breve y clara. Los objetivos se registran para tener una perspectiva del avance del proyecto, con el objetivo de tomar decisiones para concluir el proyecto.]

Productos	Cambios
A tiempo:	Solicitados:
Retrasados:	Rechazados:
Adelantados:	Realizados:
Postergados:	Postergados:

Riesgos	Actividades u Objetivos
Encontrados:	Postergados:
Resueltos:	Alcanzados:
Postergados:	

Tabla A.12: Resumen de actividades Cierre.

2. Reporte de productos.

[Se realiza un recuento de los productos, defectos y su estado al final de la ejecución del Plan del Proyecto.]

Nombre del producto	Cambios		Estado del producto
	Encontrados	Corregidos	
			<i>[Terminado/ En proceso/ Retrasado]</i>

Tabla A.13: Tabla de los Objetivos y actividades a realizar.

3. Retroalimentación.

[Listar mejoras y sugerencias del proceso seguido, listar las tareas que se pretenden realizar en la siguiente iteración anexando las tareas del Plan del Proyecto anterior que aún están en proceso y los acuerdos que el equipo ha decidido realizar para el mejoramiento del trabajo.]

Apéndice B

Lista de Plantillas Implementación de Software

B.1. Plantilla para la Especificación de Requerimientos

1.Descripción del producto

1.1. Glosario de términos (Si aplica)

[El glosario ayudará al entendimiento de los términos utilizados dentro del sistema, así como la notación y variables utilizadas para la descripción del problema.]

Término	Definición

2.Requerimientos funcionales

2.1. Funcionalidad 1

[La funcionalidad define una función importante del sistema. Se describe el qué y el cómo se realiza esa funcionalidad, esta descripción esta basada en los modelos matemáticos y numéricos realizados.

Se deben de especificar los diferentes tipos de funciones a realizar y restricciones de variables.

Se deben incluir todas las funcionalidades en las cuales se divide el sistema a construir.]

3.Requerimientos no funcionales

3.1. Caso no funcional 1

[Se espera que los requerimientos no funcionales o básicos en este tipo de sistemas sean el uso eficiente de memoria, tiempo de ejecución, resultados de la solución (manejo de errores).]

4.Restricciones de construcción

[Definir los elementos que deban de ser respetados durante la construcción, pueden ser desde el lenguaje utilizado, hasta el equipo utilizado.]

B.2. Plantilla para la Arquitectura y Diseño Detallado de Software

1. Normatividad Interna

[Especificación de las reglas que los programadores deben seguir durante la codificación, nombre de variables, constante, clases, comentarios, etc.]

2. Arquitectura de Software

[En esta sección se propone la arquitectura seleccionada del sistema, se recomienda que la arquitectura este definida en paquetes que definan una funcionalidad específica, por otro lado se debe elegir una arquitectura particular que se adapte a las necesidades del equipo de trabajo.]

3. Descripción de Componentes de Software

[En esta sección se describe la composición de los elementos que componen la arquitectura, las clases u objetos, así como sus clases y métodos (si aplica).]

B.3. Plantilla para las Pruebas de Integración

[La siguiente tabla es para anexar los elementos de las pruebas de integración, estas pruebas son de caja negra, se pueden proponer otro tipo de pruebas que se adapten a las necesidades del equipo de desarrollo.]

Integración No.	<i>[Elementos a probar en la integración]</i>
Datos de entrada	<i>[Elementos que serán ingresados para realizar la integración, se propone describir los elementos, ya sean un archivo, datos proporcionados por el usuario, etc.]</i>
Flujo esperado	<i>[Describir los pasos que se espera que los datos realicen.]</i>
Resultado esperado	<i>[Describir los resultados esperados, si es necesario el definir un rango de error aceptable, éste deberá de ser especificado en esta sección.]</i>
Cambios realizados	<i>[Detalles del componente que no funcionó y el cambio realizado para que funcionara, éste cambio debe tener la característica de ser puntual o mínima.]</i>
Defectos encontrados	<i>[Describir los defectos encontrados en los componentes que no permitieron realizar la integración.]</i>

B.4. Plantilla para las Pruebas Benchmark

1.Descripción de la Prueba.

[Se describe la parte teórica de las pruebas Benchmark a realizar. Se deben incluir las ecuaciones necesarias y la solución.]

2.Descripción de Resultados.

[Describir la realización de la prueba, los resultados obtenidos, errores calculados y la sustentación de si la prueba es aceptable o no y porqué.]

3.Observaciones.

[Describir elementos puntuales de cambios realizados al pseudocódigo para que la prueba pueda ser reproducible por algún otro usuario.]

Apéndice C

Plantillas del caso de estudio.

Los documentos que a continuación se presentan son del caso de estudio, tiene la finalidad de ejemplificar el uso de las plantillas de la guía presentada en el capítulo 2. La documentación generada durante todo el desarrollo del proyecto esta a su disposición en [21].

C.1. Plan del Proyecto iteración inicial

1.Introducción

El presente documento pretende definir y delimitar los objetivos básicos del proyecto a desarrollarse en el marco de las áreas de Geofísica y la Ingeniería de Software, siguiendo lo propuesto por la Guía realizada, la cual esta basada en la norma ISO/IEC 29110 Perfil Básico, en este documento se encuentran registrados los objetivos generales del software a desarrollar.

2.Enunciado de Trabajo

2.1 Objetivos Generales

Realizar el análisis de sustentabilidad de proyectos de recuperación de energía geotérmica debido a que estos tienen altas probabilidades de fracaso, por lo cual es usada la simulación numérica. Se busca implementar la simulación de sistemas fracturados geotérmicos en casos particulares donde la finalidad sea la extracción y administración de energía geotérmica, se pretende realizar un análisis numérico para establecer la factibilidad de un proyecto de recuperación de energía. El modelo matemático que describe el proceso es el siguiente:

$$\frac{\partial(\varphi_i \rho_j)}{\partial t} + \nabla(\rho_j v_j) = q_j \quad (C.1)$$

$$\frac{\partial(v_i)}{\partial t} + \nabla \cdot (\rho_j h_j \vec{v}_i - k_{T_j} \cdot \nabla T_j) = q_j \quad (\text{C.2})$$

El problema está definido en un medio poroso fracturado, en el cual se aplica “La doble porosidad”, el dominio del problema debe ser discretizado para lo cual se utilizará GMESH, el cual realiza un mallado (discretización), permite aplicar métodos que den solución a (C.1 y C.2), con la característica que la solución de C.1 alimenta la ecuación C.2. El mallado lo realiza GMESH, en cada volumen de control se obtendrá una ecuación lineal, que obtiene la solución de (C.1 y C.2) y esto se realiza sobre lo obtenido por GMESH, lo cual produce un sistema de ecuaciones lineales, que conformarán una matriz de por lo menos 1000x1000 elementos, que es una matriz dispersa, que será resuelta con métodos numéricos del álgebra lineal, se pretende realizar una paralelización utilizando GPU.

3.Objetivos y Tareas

3.1 Objetivos

ID	Objetivos
1	Aplicación de las actividades del proceso IS <ul style="list-style-type: none"> ■ Antecedentes de Software. ■ Análisis de Modelos.
2	Revisión teórica <ul style="list-style-type: none"> ■ Modelo matemático. ■ Métodos de solución de ecuaciones lineales.
3	Plan del proyecto Revisado y Actualizado
4	Actividades para la iteración siguiente

Tabla C.1: Plan del proyecto

3.2 Tareas

ID	Funcionalidad o Actividad	Responsable
1	<i>Antecedentes de Software</i>	
	<i>1.1</i> Métodos de S.E.L	ET
	<i>1.2</i> Matrices dispersas	ET
	<i>1.3</i> Geotermia	ET
	<i>1.4</i> Doble porosidad	ET
	<i>1.5</i> Método CVFEM	ET
2	<i>Documentación</i>	
	<i>2.1</i> Revisión del plan del Proyecto	Administrador
	<i>2.2</i> Actualización del plan del Proyecto	ET

4. Roles

Nombre	Rol	Responsabilidades
<i>Tutor</i>	<i>Administrador</i>	Actividades de validación y experto en el área de estudio
<i>Investigador de C.C</i>	<i>Desarrollador, ET</i>	Investigador, Desarrollador.
<i>Ingeniero de Software</i>	<i>Analista, ET</i>	Actividades de Ingeniería de Software.

Tabla C.2: Lista de roles y responsabilidades de los miembros del equipo de trabajo.

5. Riesgos

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Abandono de Investigación	Abandonar el trabajo por completo	Documentación clara del trabajo realizado	Alto	Identificado

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Falta de financiamiento	Financiamiento para 4 meses	Ninguna	Alto	Identificado

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Aclaración de dudas	Necesidad de asesorías sobre algún tema en específico	Buscar asesoría entre compañeros y experto del área	Alto	Identificado

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Robo o Extravío	Robo total o parcial del equipo de trabajo.	Respaldo de la información y trabajo realizado	Medio	Identificado

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Inexperiencia	Falta de experiencia en el uso de las tecnologías y temas seleccionados	Capacitación constante	Medio	Identificado

Nombre del riesgo	Descripción del riesgo	Plan de contingencia	Impacto del riesgo	Estado del riesgo
Ingeniería de Software	No seguir las prácticas de Ingeniería de Software propuestas.	Reajuste de dichas prácticas	Bajo	Identificado

6. Control de versiones

Estrategia de control de versiones	
Identificación de la configuración	Se almacenará el código fuente del sistema y todo lo relacionado al código, así como la documentación generada e importante para el proyecto. Las versiones de los documentos generados deben de contener la letra inicial de la persona que realiza la edición, seguido del número de la iteración en que se creó y el número de versión. Las carpetas de documentación para la administración estarán separadas de los elementos relacionados al sistema.
Estado de la configuración	Plan del proyecto E01.
Responsable	El equipo de trabajo se hará cargo de los elementos almacenados mientras que el programador es el encargado del código fuente.
Políticas	ET.

7. Repositorio

Repositorio	
Clasificación	Centralizada
Herramienta a utilizar	Bitbucket
Ubicación	http://bitbucket.org/m0nT3cR1s70/sysg/src
Estructura	Por definir

C.2. Plan del Proyecto

1.Objetivos

ID	Objetivos
1	Realizar pruebas de tipo Benchmark.
2	Realizar cambios necesarios.
3	Validación del sistema.
4	Revisar documentación del sistema.

Tabla C.3: Objetivos de la iteración.

2.Tareas

ID	Funcionalidad o Actividad	Responsable
1	<i>Definir pruebas Benchmark</i>	
	1.1 Obtener problemas con solución analítica.	Administrador del proyecto.
	1.2 Entender las pruebas a realizar.	Desarrollador.
2	<i>Realizar Cambios</i>	
	2.1 Realizar los cambios necesarios al sistema para poder iniciar y aplicar las pruebas Benchmark.	Desarrollador
3	<i>Validación</i>	
	3.1 Aplicar el sistema a la pregunta inicial a resolver.	Administrador, Desarrollador.
4	<i>Documentación</i>	
	4.1 Verificar que la documentación esté completa y disponible.	Desarrollador.

Tabla C.4: Tareas a ser realizadas en la iteración.

3.Cambios

ID	Producto	Descripción	Solicitante
1	CVFEM	Modificación de la funcionalidad que construye la matriz y la del vector \mathbf{b} de la ecuación $\mathbf{Ax} = \mathbf{b}$.	Desarrollador.
2	Visualizador.	Construcción de una función para la graficación de una superficie.	Administrador, Desarrollador.
3	CVFEM.	Cálculo de la funcionalidad de permeabilidad aleatoria.	Desarrollador.

Tabla C.5: Cambios a realizar durante la iteración.

4.Riesgos

Nombre	Descripción	Plan de contingencia	Impacto	Estado del riesgo
Fecha de entrega.	La fecha de conclusión del proyecto esta próxima y aún falta mucho por realizar.	Solicitar una extensión de tiempo.	Alto.	Identificado
Reuniones.	Reuniones con el Administrador y experto del tema, deben ser más frecuentes.	Constante comunicación por otros medios: e-mail, teléfono, etc.	Alto.	Controlado.

Tabla C.6: Riesgos encontrados.

C.3. Reporte de Seguimiento

Proyecto: [Simulación de flujo en medios porosos usando CVFE]

Periodo a Reportar: 1-11-2016.] al [30-11-2016]

1.Reporte de Actividades

Integrante	Actividad u Objetivo	Tiempo Total	Estado	Observaciones
Desarrollador	Pruebas Benchmark	45 hr	Finalizado	La obtención de las pruebas de parte del Administrador tomó dos horas, el entendimiento y realización de las pruebas tomó mucho tiempo, fueron 3 pruebas a realizar, y en cada prueba se realizaron los cambios necesarios para el buen funcionamiento del sistema.
Desarrollador	Cambios	40 hr	Finalizado	Los cambios realizados fueron aplicados en conjunto con las pruebas Benchmark, se realizaron cambios en el paquete del mallado, CVFEM.
Desarrollador	Validación	40 hr	Pendiente	La validación se encuentra pendiente, debido a los cambios que se deben realizar.
Desarrollador	Documentación	16 hr	Pendiente	La documentación está siendo revisada y actualizada, aún falta por definir completamente ciertos documentos.
Total de Horas trabajadas		141		

Tabla C.7: Estado actual del proyecto.

2.Tareas

Actividad y/o tarea	Integrante	Fecha de entrega	Fecha real
Obtener la definición del problema benchmark	Desarrollador	[2-11-2016]	[2-11-2016]
Entendimiento de las pruebas	Desarrollador	[4-11-2016]	[16-11-2016]
Realizar cambios	Desarrollador	[11-11-2016]	[16-11-2016]
Validación	Desarrollador	[22-11-2016]	Pendiente
Documentación	Equipo de trabajo	[22-11-2016]	Pendiente

Tabla C.8: Avance de tareas.

3.Productos

ID	Producto	Estado
1	Tratamiento del Dominio	Terminado
2	CVFEM	Terminado
3	Almacenamiento de la Matriz $\underline{\underline{A}}$	Terminado
4	Cálculo del vector \underline{x}	Terminado
5	Almacenamiento del vector \underline{x} y de \underline{b}	Terminado
6	Visualizador	Iniciado

Tabla C.9: Productos.

4.Cambios

ID	Producto	Descripción	Solicitante	Estado
1	Tratamiento del Dominio	<p>Se realizaron los siguientes cambios:</p> <ul style="list-style-type: none"> ■ Reordenamiento de nodos. ■ Distinción entre nodos frontera e interiores. ■ Tratamiento de nodos Neumann y Dirichlet. 	Desarrollador	Realizado
2	CVFEM	<p>Se realizaron los siguientes cambios:</p> <ul style="list-style-type: none"> ■ Análisis de permeabilidad. ■ Distinción de subdominios. ■ Realización de tipos de matrices para nodos tipo Neumann y Dirichlet. 	Desarrollador	Realizado

Tabla C.10: Cambios.

5.Riesgos

Nombre	Descripción	Plan de contingencia	Impacto	Estado del riesgo
Tiempo para realizar cambios	El tiempo de entrega y el tiempo para la realización de los cambios es insuficiente.	Solicitar una extensión de tiempo	Alto	Identificado
Cambios que puedan comprometer el funcionamiento del sistema	Al momento de realizar los cambios, el sistema pueda dejar de funcionar.	Tener las versiones del código disponibles, en caso de que no se halle la solución al problema de manera inmediata.	Alto	Controlado

Tabla C.11: Riesgos encontrados.

6.Resumen

Tareas	Cambios	Riesgos
A tiempo: 4	Solicitados: 5	Encontrados 2
Retrasadas: 2	Rechazados: 0	Resueltos 1
Adelantadas:0	Realizados: 5	Postergados 1
Postergadas: 2	Postergados: 0	

Tabla C.12: Resumen.

C.4. Cierre del Plan del Proyecto

1. Resumen

Productos	Cambios
A tiempo: 5	Solicitados: 0
Retrasados: 0	Rechazados: 0
Adelantados: 0	Realizados: 7
Postergados: 1	Postergados: 0

Riesgos	Actividades u Objetivos
Encontrados: 2	Postergados: 2
Resueltos: 2	Alcanzados: 2

Tabla C.13: Resumen

2. Reporte de productos.

Nombre del producto	Cambios		Estado del producto
	Encontrados	Corregidos	
Tratamiento del Dominio	3	3	[Terminado]
CVFEM	3	3	[Terminado]
Almacenamiento de la matriz $\underline{\underline{A}}$	0	0	[Terminado]
Cálculo del vector \underline{x}	0	0	[Terminado]
Almacenamiento del vector \underline{x} y del vector \underline{b}	0	0	[Terminado]

Tabla C.14: Reporte de Productos.

3. Retroalimentación.

- Terminar la validación del sistema.
- Iniciar el cierre del proyecto.

4.Comentarios

Se tiene el sistema en un estado próximo a finalizar, faltan ciertas modificaciones al sistema relacionadas con la validación, una vez realizadas deben de ser aprobadas por el Administrador.El trabajo es complejo y en ésta parte del proyecto se necesita de la experiencia y apoyo del experto en el área.

Aún no se tiene la certeza del tiempo que se necesita para terminar estas modificaciones pero se trabaja para que todo el sistema esté listo antes de diciembre del 2016, esperando presentar el trabajo para evaluación en el mes de enero del 2017.

C.5. Especificación de Requerimientos

1.Descripción del Producto

El sistema debe ser capaz de leer un dominio en dos dimensiones construido con la herramienta gmsh según la triangulación de Delaunay. Las especificaciones del archivo están en el apéndice B. Con lo cual debe ser capaz de almacenar los nodos y los elementos triangulares, los nodos deben ser ordenados respecto a su coordenada y y después a su componente x .

Se debe construir un sistema de ecuaciones donde la matriz generada debe considerar condiciones de frontera y no debe de almacenar 0's. El sistema debe ser resuelto con métodos numéricos de álgebra lineal. La solución final debe ser almacenada en un archivo con el siguiente formato, coordenada x , coordenada y y solución en ese punto, finalmente debe de poder graficarse la solución obtenida por lo cual se sugiere la utilización de una herramienta externa.

2.Requerimientos funcionales

2.1. Tratamiento del Dominio de interés

- Lee el contenido de un archivo gmsh.
- Almacena las coordenadas x, y de cada nodo del dominio.
- Almacena los nodos que forman un elemento triangular.
- Los nodos son reordenados primero por su coordenada y y después por su coordenada x por ejemplo:

Coordenada x	Coordenada y
0	0
0.5	0
1	0
0.2	0.1
0.5	0.4
0.6	0.5
0.3	0.7

- Se reordenan los elementos según la nueva numeración de los nodos.
- Se deben distinguir los nodos que están en el contorno del dominio y aquellos que están en el interior.

- Se deben distinguir los nodos de contorno en dos tipos:
 - Neumann.
 - Dirichlet
- Para cada nodo contorno se debe almacenar si valor.
- Cada nodo debe identificar que elementos son su soporte; es decir que elementos contienen a ese nodo como uno de sus vértices.
- Cada nodo debe identificar que nodos son sus vecinos.
- Debe calcularse el área de cara elemento triangular.

2.2. CVPFEM

- Debe ser capaz de construir un sistema de ecuaciones de la forma $\underline{\underline{A}}\underline{\underline{x}}=\underline{\underline{b}}$.
- La matriz $\underline{\underline{A}}$ debe ser construida a partir de los coeficientes de transmisibilidad según la teoría y la discretización realizada en [21].
- El vector $\underline{\underline{b}}$ se debe construir a partir de la teoría y la discretización del vector en [21].
- Debe de ser capaz de calcular la permeabilidad en distintos subdominios según lo descrito en el problema [21].

2.3. Almacenamiento de un vector.

- Debe almacenar en memoria los componentes de un vector tipo Double.
- Se definen las operaciones vector-vector
 - Suma.
 - Resta.
 - Producto punto.
- Se definen las operaciones con el esquema de almacenamiento; Matriz por vector.
- Cálculo de una norma euclidiana $\|V\| = \sqrt{v_1^2 + v_2^2 \dots + v_n^2}$
- Cálculo del rms: $V_{rms} = \frac{\sqrt{v_1^2 + v_2^2 \dots + v_n^2}}{n}$

2.3. Almacenamiento de una matriz.

- Almacenamiento de los componentes de una matriz.
- Inicializar una matriz con ceros.

- Acceso a los valores de la matriz.
- Una iteración del algoritmo Jacobi.
- Una iteración del algoritmo Gauss Seidel.

2.4. Álgebra lineal.

- Implementar los métodos para sistemas de ecuaciones lineales.
 - Jacobi.
 - Gauss.
 - Sor.
 - Gradiente Conjugado.
 - BIGSTAB.
- Implementar preconditionadores de la matriz.
 - Jacobi.
 - MILU.
 - ILU.
 - ICHOL.

2.5. Visualizador.

- Graficación de la solución final usando la biblioteca matplotlib.
- Solve:
 - Devuelve la solución final en un archivo de texto segun un visualizador externo.
 - Calcula la distribución del error para los problemas de tipo benchmark usando $|error| = |exacta_{i,j} - calculada_{i,j}|$. Es decir se calcula para cada nodo su error.

3.Requerimientos no funcionales

- El rendimiento es un factor importante.
- Debe permitir un mantenimiento sencillo.
- Minimización del uso de memoria en el almacenamiento de la matriz.

- Utilizar técnicas de programación genérica y metaprogramación.

4. Restricciones de construcción

El lenguaje de programación debe ser C++

C.6. Arquitectura y Diseño Detallado de Software

1. Normatividad Interna

Con la finalidad de hacer más legible el código para el equipo de trabajo y para otros usuarios se tienen las siguientes reglas en la construcción del sistema:

- Los atributos inician con (*Palabra(s)clave(s)*).
- Palabra clave inicia con minúscula, si son 2 palabras, la primera con minúscula y la segunda con mayúscula.
- Nombre de las clases con mayúscula.
- Nombre de la clase = Nombre del archivo = Palabra identificadora.
- Toda clase se escribe en 2 archivos: hpp y cpp a menos que sean clases Template.
- Toda clase es documentada con ayuda de la herramienta Doxygen.
- Los comentarios se hacen por línea y se deja un espacio en blanco antes de iniciar la escritura de los comentarios.
- Al inicio de cada archivo se describe generalmente lo que hace la clase.
- Nombre del autor.
- Descripción del programa de manera general.
- Los métodos inician con minúscula y las palabras clave terminan con P mayúscula.
- Toda clase debe tener un constructor vacío y el constructor copia.
- Los parámetros variables de los métodos inician con minúsculas.

2.Arquitectura de Software

En la construcción del sistema se debe aplicar el POO, el rendimiento y el mantenimiento son requerimientos importantes para satisfacerlos, la arquitectura diseñada consta de 6 clases que pueden ser modificadas con sencillez debido a que cada uno de éstas tiene una tarea específica, además existe una poca comunicación entre ellas. Se presenta una descripción abstracta del diagrama de clases (Ver Fig. C.1) de la arquitectura propuesta.

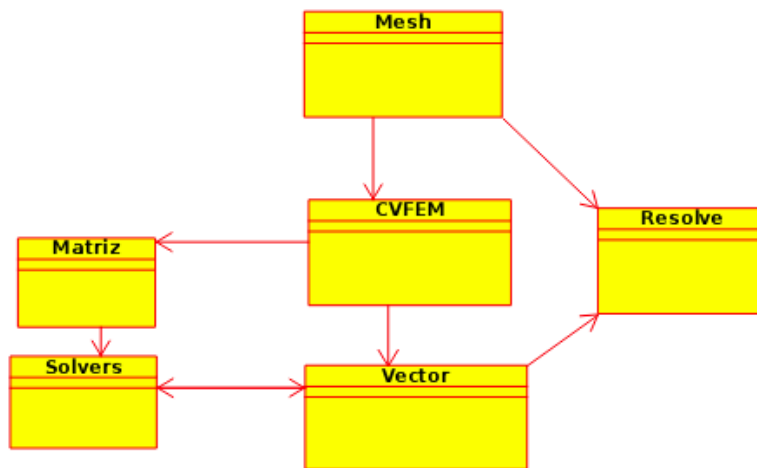


Figura C.1: Diagrama de clases de la arquitectura para SYSG.

Cada clase se comunica con los demás por medio de una interfaz. Por lo tanto si se llega a la conclusión que una clase no satisface los requerimientos o que puede ser optimizada de alguna manera, puede ser reemplazada por otra fácilmente, siempre y cuando la interfaz se mantenga. A continuación se presenta de manera más detallada cada una de las clases.

3. Descripción de Componentes de Software

Mesh

La clase Mesh(Ver figura C.2), tiene como objetivo el tratamiento del dominio y sus funcionalidades principales son:

- Almacenar los nodos y los elementos de la malla.
- Reordenar los nodos en forma ascendente.
- Identificar los nodos frontera y los nodos interiores.
- Identificar los elementos soporte de cada nodo.
- Dado un nodo, identificar los nodos vecinos.
- Calcular el área de cada elemento triangular.
- Almacenamiento de las condiciones de frontera de cada nodo.

```

class Mesh
{
public:
    ~Mesh() {}
    // Nodos
    - _nodes : int
    // Elementos
    - _elements : int
    // Orden de los nodos
    - _coordOrder : std::vector< std::pair< double, double >>
    // Orden de los elementos
    - _elementsOrder : std::vector< std::vector< int >>
    // Nodos de soporte
    - _supportNodeElemint : std::vector< std::vector< int >>
    // Nodos de frontera
    - _nodeBoundary : std::vector< int >
    // Nodos interiores
    - _nodeInterior : std::vector< int >
    // Condiciones de frontera
    - _boundaryCondition : std::vector< double >
    // Tipo de frontera
    - _boundaryType : std::vector< int >
    // Nodos de Dirichlet
    - _nodeDirichlet : std::vector< int >
    // Nodos de Neumann
    - _nodeNeumann : std::vector< int >
    // Nodos de Incongnite
    - _nodeIncongniteP : std::vector< int >
    // Área
    - _area : std::vector< double >
    // Métodos
    + Mesh()
    + Mesh(filename : std::string, vertices : std::vector< double >>6)
    + initialize(filename : std::string, vertices : std::vector< double >>6)
    + readFEM9(filename : std::string, coord : std::vector< std::vector< double >>6, elem : std::vector< std::vector< int >>6)
    + reorderNode(coord : std::vector< std::vector< double >>6)
    + relationNodes(coord : std::vector< std::vector< double >>6, relation : std::vector< int >>6)
    + reorderElement(elem : std::vector< std::vector< int >>6, relation : std::vector< int >>6)
    + borderInteriorNode(vertices : std::vector< double >>6)
    + boundaryConditionType(int, value : double)
    + typeNodeBorder()
    + areaElement()
    + elementSupportNode()
    + nodeNeighbor(node : int) : std::vector< int >
    + nodeIncongnite()
    + pif(vertices : std::vector< std::pair< double, double >>6) : std::vector< int >
    + nNodeIncongnite() : int
    + borderNode() : std::vector< int >6
    + coordX(node : int) : double
    + coordY(node : int) : double
    + valueNode(int) : double
    + typeNode(int) : int
    + Incongnite() : std::vector< int >6
    + nNode() : int
    + showGrid(coord : std::vector< std::vector< double >>6, elem : std::vector< std::vector< int >>6)
    + showNode()
    + showElement()
    + showNodeElement()
    + showRelation(relation : std::vector< int >>6)
    + showNodeF()
    + showArea()
    + showElementSupportNode()
    + showBoundaryCondition()
    + showNodeDirichlet()
    + showNodeNeumann()
    + showNeighbor()
    + showTotal()
}

```

Figura C.2: Clase Mesh.

CVFEM

La clase CVFEM(Ver figura C.3), tiene por objetivo principal formar el sistema de ecuaciones $\underline{Ax} = \underline{b}$, aplicando la expresión discreta a cada vértice de la malla, tomando en cuenta las condiciones de frontera. Sus funcionalidades más importantes son:

- Cálculo de los coeficientes de las funciones de forma.
- Cálculo de los coeficientes de transmisibilidad.
- Construcción de la matriz \underline{A} .
- Cálculo de la permeabilidad.
- Cálculo del vector b .

```

CVFEM
- _tolP : double
- _permeabilidadP : double
- _compresibilidadP : double
- _porosidadP : double
- _densidadP : double
- _gravedadP : double
- _fuenteP : double
- k11 : Vector
- k22 : Vector
+ CVFEM()
+ CVFEM(perme : double, compre : double, fuen : double, poro : double, dens : double, grav : double)
+ CVFEM(cvfem : CVFEM&)
+ Ab(A : CSR&, b : Vector&, mesh : Mesh&)
+ Ab(A : CSR&, b : Vector&, mesh : Mesh&, alta : std::vector< std::pair< double, double >>&, baja : std::vector< std::pair< double, double >>&)
+ matrixCVFEM(mesh : Mesh&, csr : CSR&)
+ permeabilidad(mesh : Mesh&, alta : std::vector< std::pair< double, double >>&, baja : std::vector< std::pair< double, double >>&)
+ permehar(node1 : double, node2 : double) : double
+ permeAverage(node1 : double, node2 : double) : double
+ gPerme() : double
+ sPerme(permeabilidad : double)
+ gCompre() : double
+ sCompre(compresibilidad : double)
+ gDensidad() : double
+ sDensidad(densidad : double)
+ gGravedad() : double
+ sGravedad(gravedad : double)
+ gFuente() : double
+ sFuente(fuente : double) : double
+ gK11() : Vector&
+ gK22() : Vector&
+ baseFunction(a : double, b : double, c : double, x1 : double, x2 : double, K : double) : double

```

Figura C.3: Clase CVFEM.

Matriz

La matriz que se construye con el CVFEM (Ver figura C.4), es dispersa y no estructurada, por lo tanto muchos de sus componentes son cero y aquellos que no son cero se presentan en posiciones irregulares. Los componentes que son cero no son significativos para obtener la aproximación de la solución final, por lo tanto, almacenarlos constituye una pérdida de memoria, conviene entonces usar estrategias de almacenamiento que permitan almacenar y operar sólo la información relevante. Existen distintos esquemas de almacenamiento para matrices dispersas, en este trabajo se eligieron COO y CSR.

- **COO.** Ésta estrategia es la más natural, almacena el valor y las coordenadas i y j de cada componente no cero en tres arreglos de nombre: data, row y col. Su ventaja con relación a otros es que la inserción es constante, sin embargo, la búsqueda de un elemento es una operación ineficiente cuya complejidad es $O(nnz)$, donde nnz es el número de elementos no cero, debido a que los elementos pueden estar desordenados.
- **CSR.** Ésta estrategia, almacena los componentes por renglón en tres arreglos: data, col e irow, en estos: se almacenan los datos, el índice de columna y los índices de inicio y fin de cada renglón. Su ventaja principal es que la complejidad de buscar un elemento dentro de la matriz es $O(\log_2 n)$, donde n es el número máximo de elementos en un renglón.

Sabiendo todo esto y con el fin de minimizar: el tiempo de procesamiento y el almacenamiento, es utilizado el esquema COO para la construcción de la matriz y una vez formado se convierte al esquema CSR.

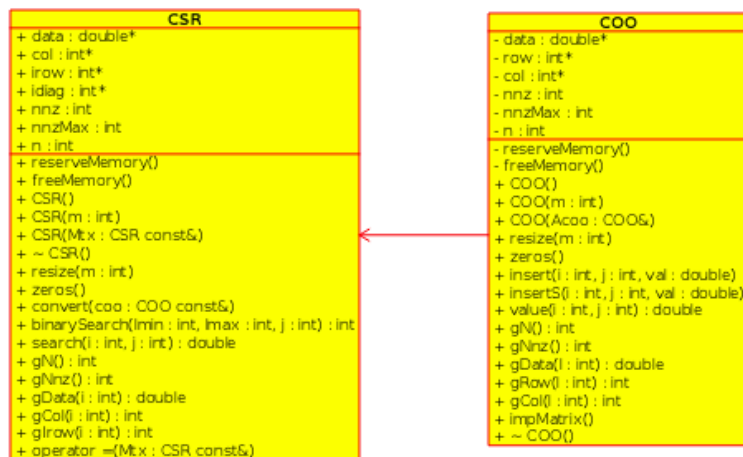
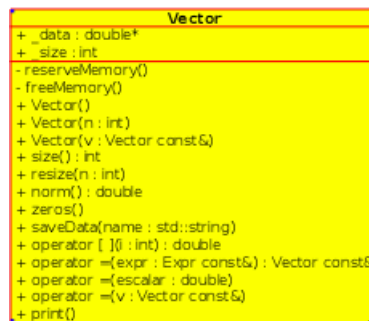


Figura C.4: Diagrama de clases para COO y CSR.

Almacenamiento de Vectores

Se transforma en la clase Vector (Ver figura C.5), su objetivo principal, es almacenar y operar la información de los vectores definidos por CVFEM. Sus operaciones principales son:

- Almacenamiento de un vector dado.
- Cálculo de la norma euclidiana.
- Cálculo de la media cuadrática rms.



```
classDiagram
    class Vector {
        + _data : double*
        + _size : int
        - reserveMemory()
        - freeMemory()
        + Vector()
        + Vector(n : int)
        + Vector(v : Vector const&)
        + size() : int
        + resize(n : int)
        + norm() : double
        + zeros()
        + saveData(name : std::string)
        + operator [ ](i : int) : double
        + operator =(expr : Expr const&) : Vector const&
        + operator =(escalar : double)
        + operator =(v : Vector const&)
        + print()
    }
```

Figura C.5: Clase Vector.

Solvers

Se implementa una librería de métodos iterativos, los cuales son: Jacobi, Gauss Seidel, SOR, Gradiente conjugado (CG) y BICGSTAB. Además, con el fin de mejorar la matriz son implementados preconditionadores los cuales son: Jacobi, ILU y MILU. Pueden ser usados por CG y BICGSTAB. (Ver figura C.6).



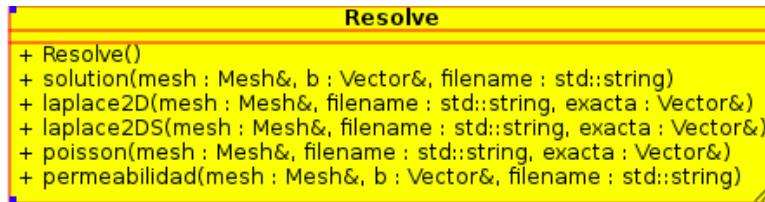
Figura C.6: Diagrama de clases para COO y CSR.

Con el fin de mejorar la eficiencia los métodos numéricos no se usa en ningún caso polimorfismo dinámico.

Resolve

La clase Resolve (Ver figura C.7) tiene como tareas:

- Construcción de un archivo con la solución aproximada del problema para ser leída por un visualizador externo.
- Construcción de la solución exacta para los problemas benchmark.



```

Resolve
+ Resolve()
+ solution(mesh : Mesh&, b : Vector&, filename : std::string)
+ laplace2D(mesh : Mesh&, filename : std::string, exacta : Vector&)
+ laplace2DS(mesh : Mesh&, filename : std::string, exacta : Vector&)
+ poisson(mesh : Mesh&, filename : std::string, exacta : Vector&)
+ permeabilidad(mesh : Mesh&, b : Vector&, filename : std::string)
  
```

Figura C.7: Clase Resolve.

Visualizador de datos externos

Se eligió como visualizador externo la biblioteca Matplotlib, para la visualización de los datos, para tal motivo se desarrolla un programa en python que permita su uso.

Comunicación

Para la comunicación de las clase se debe tomar en cuenta las siguientes consideraciones:

- **Mesh.** Esta no requiere de utilizar ningún servicio de otra clase, para funcionar requiere únicamente del archivo generado con gmsh.
- **CVFEM.** Esta clase construye el sistema $\underline{\underline{A}}\underline{x} = \underline{b}$, por lo tanto requiere de la información de la malla, requiere contenedores para almacenar a la matriz $\underline{\underline{A}}$ y al vector \underline{b} .
- **COO y CSR.** Contenedores para almacenar la matriz generada por la clase CVFEM en formatos optimizados.
- **Vector.** Contenedor, para almacenar el vector \underline{b} y la solución final.
- **Solvers** Sólo requieren la matriz $\underline{\underline{A}}$, al vector \underline{b} y al vector \underline{x} para calcular la solución del sistema.
- **Solve.** Requiere la información de la malla y la solución calculada con los métodos de álgebra lineal.

C.7. Pruebas

Se presenta una de las pruebas realizadas, con el objetivo de mostrar la documentación realizada, la documentación completa del proyecto se encuentra disponible para su revisión en [21]:

Pruebas de Paquete

Se prueba la clase COO Y CSR que implementan esquemas útiles de almacenamiento, se eligió

Datos de entrada

Se requiere una matriz dispersa como la mostrada en la figura

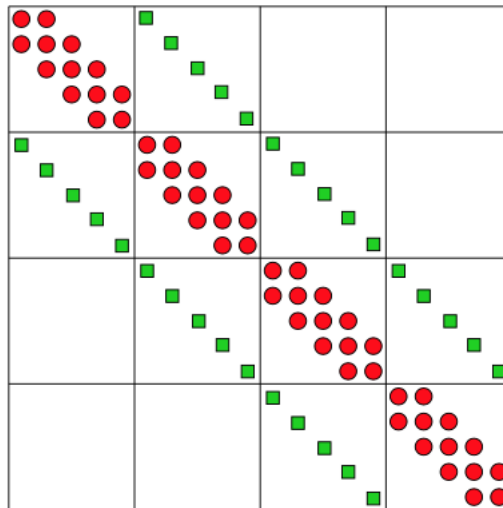


Figura C.8: Matriz dispersa

Flujo esperado

Se define el tamaño de la matriz que será almacenada, de manera temporal se almacena en un formato COO, finalmente se transforma la matriz COO a una matriz CSR, esto en código:

```
{
  COO Acoo(n); // matriz temporal
  timer.tic(); // comienza a medir tiempo
  for(int j=1;j<ny;++j)
  {
    for(int i=1;i<nx;++i)
    {
      if(j>1)
        Acoo.insert(1, 1-(nx-1), -1.);
      if(i>1)
        Acoo.insert(1, 1-1, -1.);
        Acoo.insert(1, 1, 4.);
      if(i<nx-1)
        Acoo.insert(1, 1+1, -1.);
      if(j<ny-1)
        Acoo.insert(1, 1+(ny-1), -1.);
        ++1;
    }
  }
  A.convert(Acoo); // convierte de COO a formato CSR
  Acoo.impMatrix();
}
// destruye Acoo
```

Resultado esperado

Después de compilar y ejecutar el código anterior el resultado obtenido puede verse en la figura C.9. La tabla muestra el tiempo utilizado para la construcción de matrices de distinto tamaño en formato COO y su conversión a CSR.

4	-1	0	-1	0	0	0	0	0
-1	4	-1	0	-1	0	0	0	0
0	-1	4	0	0	-1	0	0	0
-1	0	0	4	-1	0	-1	0	0
0	-1	0	-1	4	-1	0	-1	0
0	0	-1	0	-1	4	0	0	-1
0	0	0	-1	0	0	4	-1	0
0	0	0	0	-1	0	-1	4	-1
0	0	0	0	0	-1	0	-1	4

Figura C.9: Se muestra la matriz almacenada en el formato CSR

Tamaño	Inserción COO [ms]	Conversión CSR [ms]
100x100	0.002	0.002
10000x10000	0.12	0.199
40000x40000	0.584	1.126
90000x90000	1.146	2.018
160000x160000	2.745	3.862
250000x250000	5.185	6.504
360000x360000	6.733	9.288
640000x640000	8.05	16.027
810000x810000	11.155	19.669
1000000x1000000	13.785	28.496

Tabla C.15: Ejecuciones para distintos tamaños de una matriz.

Cambios realizados

Sin cambios.

Defectos encontrados

Sin defectos encontrados

Bibliografía

- [1] *Norma Técnica Peruana NTP-RT-ISO/IEC TR 29110-5-1-2 –INGENIERIA DE SOFTWARE. Perfiles de ciclo de vida para las pequeñas organizaciones (PO). Parte 5-1-2: Guía de gestión e ingeniería: Grupo de perfil genérico. Perfil básico*, vol. Primera Edición. INDECOPI, Calle de la Prosa 104, San Borja (Lima41) Apartado 145, May 2012.
- [2] ABDALA, K., AND ANTONIO, JOSÉ, . Iii. 3 modelos matemáticos,físicos y conceptuales. su uso para el pronostico.
- [3] ANTILLANCA, H. E., AND C., G. Desarrollo de software de investigación.
- [4] BAKER, J. *50 physics ideas you really need to know*. Banshee, 2007.
- [5] BENISTON, M. *From turbulence to climate: numerical investigations of the atmosphere with a hierarchy of models*. Springer Science & Business Media, 2012.
- [6] COPANT. Normas internacionales y normas privadas, 2010.
- [7] CRUZ, L. M., AND RAMOS, E. General template units for the finite volume method in box-shaped domains. *ACM Transactions on Mathematical Software (TOMS)* 43, 1 (2016), 1.
- [8] DIMITRI VAN HEESCH. <http://www.stack.nl/~dimitri/doxygen/>, 1997–2016.
- [9] GOLUB, G. H., AND ORTEGA, J. M. *Scientific computing: an introduction with parallel computing*. Elsevier, 2014.
- [10] HANNAY, J. E., MACLEOD, C., SINGER, J., LANGTANGEN, H. P., PFAHL, D., AND WILSON, G. How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE workshop on Software Engineering for Computational Science and Engineering* (2009), IEEE Computer Society, pp. 1–8.
- [11] HASHAGEN, U., KEIL-SLAWIK, R., AND NORBERG, A. L. *History of Computing: Software Issues: International Conference on the History of Computing, ICHC 2000 April 5–7, 2000 Heinz Nixdorf MuseumsForum Paderborn, Germany*. Springer Science & Business Media, 2002.
- [12] ISO/IEC TR 29110-5-1-2-2011, Software Engineering–Lifecycle Profiles for Very Small Entities (VSEs)– Part 5-1-2 Management and engineering guide: Generic profile group: Basic Profile. Standard, International Organization for Standardization, Geneve, CH, June 2011.

- [13] JAIN, M. K., IYENGAR, S. R., AND JAIN, R. K. *Numerical methods: problems and solutions*. New Age International, 2007.
- [14] KELLY, D. F. A software chasm: Software engineering and scientific computing. *IEEE Software* 24, 6 (2007), 120–119.
- [15] LAPORTE, C. Y., SÉGUIN, N., VILLAS BOAS, G., AND BUASUNG, S. Pequeñas empresas de tecnología: aprovechando las ventajas de las normas de ingeniería de software y sistemas. *Revista ISO Focus+* (2013).
- [16] LEONES, A. A. V. *Modelo computacional en paralelo de flujo y transporte*, 2013.
- [17] LLOPIS, G., AND RUBIO, L. *Física: curso teórico-práctico de fundamentos físicos de la ingeniería*. Editorial Tebar, 1998.
- [18] MARTÍNEZ, R. C. C. Simulación numérica de inyección de agua en yacimientos petroleros empleando el método de líneas de corriente, 2014.
- [19] MORALES, M. E. T. *El proceso de desarrollo y mantenimiento de software propuesto por Competisof de acuerdo al proceso unificado*. PhD thesis, Tesis de Maestría en Ciencia e Ingeniería de la Computación, UNAM, México DF, México, 2010.
- [20] MORRISON, M. *Reconstructing reality: Models, mathematics, and simulations*. Oxford University Press, USA, 2015.
- [21] NIETO, M. A. B. Simulación de flujo en medios porosos usando cvfem, 2017.
- [22] SEGAL, J., AND MORRIS, C. Developing scientific software. *IEEE Software* 25, 4 (2008), 18–20.
- [23] SOMMERVILLE, I., AND GALIPIENSO, M. I. A. *Ingeniería del software*. Pearson Educación, 2005.
- [24] STREMPER, G. C. *Cómputo paralelo para la solución de flujo en medios porosos aplicando funciones de base radial*, 2016.
- [25] TVEITO, A., LANGTANGEN, H. P., NIELSEN, B. F., AND CAI, X. *Elements of Scientific Computing*, vol. 7. Springer Science & Business Media, 2010.
- [26] UNAM, IIMAS. Posgrado en ciencia e ingeniería de la computación, 2016.
- [27] VIERA, M. A. D., SAHAY, P., CORONADO, M., AND TAPIA, A. O. *Mathematical and numerical modeling in porous media: applications in geosciences*. CRC Press, 2012.
- [28] ZAMORA, J. M. F. Simulación de dinámicas moleculares usando unidades de procesamiento gráficas (gpus), 2012.