Instituto Superior Minero Metalúrgico
Dr. Antonio Núñez Jiménez
Facultad Metalúrgia - Electromecánica
Moa, Holguín

Trabajo de Diploma

para optar por el título de Ingeniería en Informática

Título:

"Desarrollo de una interfaz gráfica de usuario para el preprocesador meteorológico AERMET"

Autor:
Dariel Raúl Subirós Muñoz.

Tutores:
Ing. Roiky Rodríguez Noa.
Lic. Liban Montes de Oca González.

Consultante:
Dr. Allan Pierre Conde.

Moa, Junio de 2009.
"Año del 50 Aniversario del Triunfo de la Revolución."

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez" (ISMMM) y al Centro de Estudios del Medio Ambiente (CEMA) para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del 2009.

Dariel Raúl Subirós Muñoz

_____

Firma autor

Ing. Roiky Rodríguez Noa.

_____

Firma tutor 1

Lic. Liban Montes de Oca González.

_____

Firma tutor 2

AGRADECIMIENTOS

Quiero agradecer a todas las personas que de una forma u otra me han servido de apoyo para el desarrollo de esta investigación y en especial:

A Dios por darme la oportunidad de ser su hijo.

A mi madre que aunque no está aquí conmigo, hoy cumplo su más grande deseo.

A mi abuela Mirtha por comportarse conmigo como nadie en el mundo.

A mi papá Claudio y mi hermanito por darme su corazón.

A Arturo por su inigualable disposición.

A todo el departamento de informática del CEMA.

A mi mejor amigo Obed Matos Prats por aconsejarme y estar siempre ahi.

A Ramonita por servirme sin condiciones.

A mis suegros por acogerme como su hijo varón.

A mi esposa por su amor, belleza, comprensión y ser lo especial que busca todo hombre, ..pero que ya yo encontré.

A todos, gracias…

DEDICATORIA

El tiempo perdido no se recupera, es por ello que los lamentos de nada sirven y lo ideal es actuar a tiempo y estar ahí cuanto te necesiten. Yo dedico esta tesis a la memoria de mi bella mamá, y me prometo a mi mismo aprovechar el tiempo al máximo, en especial compartiendo con aquellas personas que me aman.

PENSAMIENTO

"Hay muchos mitos acerca del amor, muchos jueces que comentan y reproducen lo que oyen acerca de él, pero nada se compara con vivirlo y es más, las mejores y peores experiencias de la vida, se la debemos a él..."

Dariel Subirós

RESUMEN

La contaminación del aire en Moa, hace necesaria la aplicación de modelos de dispersión de contaminantes, para estudiar los efectos producidos por el escape de gases de las chimeneas de las fábricas y otras fuentes de contaminación en la localidad.

Dentro de los softwares de modelos de dispersión existentes, el recomendado por la Agencia de Protección Ambiental de los Estados Unidos (USEPA) para estudios detallados de áreas menores de 50 kilómetros (km) o locales, es el American Metheorological Society/ Environment Protection Agency (AERMOD, por sus siglas en inglés). El sistema de modelos AERMOD[1], contiene un programa central (AERMOD) y, dos pre-procesadores de datos de entrada que son componentes regulatorios del sistema: AERMAP[2] el pre-procesador de datos de terreno y AERMET[3], el pre-procesador de datos meteorológicos.

La forma original de configurar los datos de entrada para el AERMET no es capaz de brindar al usuario facilidad de interacción. En esta investigación se desarrolló una interfaz gráfica de usuario (GUI) amigable para permitir el uso eficiente de este preprocesador meteorológico y, además incorpora métodos de graficación de parte de la información preprocesada.

Para lograr los objetivos planteados se hizo un estudio de las características principales del AERMET. Se utilizaron herramientas de software libre y la metodología ágil de desarrollo de software XP.

[1] *Technology Transfer Network Support Center for Regulatory Atmospheric* http://www.epa.gov/scram001/dispersion_prefrec.htm#aermod.
[2] *User´s Guide for the AERMOD Terrain Preprocessor (AERMAP),* U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards Emissions, Monitoring, and Analysis Division Research Triangle Park, North Carolina 27711.
[3] *User´s Guide for the AERMOD Meteorological Preprocessor (AERMET),* U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards Emissions, Monitoring, and Analysis Division Research Triangle Park, North Carolina 27711, 03-002, November 2004.

ABSTRACT

The pollution of the air in Moa, makes necessary the application of pollution dispersion models, to study the effects pduct of escape of gases of the chimneys of the factories and other sources of contamination in the town.

Among the available softwares of pollution dispersion models, the one recommended by the Agency of Environmental Protection of the United States (USEPA) for detailed studies of areas smaller than 50 kilometers (km) or local, is the American Metheorological Society / Environment Protection Agency (AERMOD, by it’s initials in English). The modelling system AERMOD, contains a central program (AERMOD) and, two pre-processors of input data that are regulatory components of the system: AERMAP the terrain data pre-processor and AERMET, the pre-processor of meteorological data.

The original way of configuring the input data for the AERMET is not able to offer to the user interaction in an easy way. In this investigation a friendly user's graphic interface(GUI) was developed to allow the efficient use of this meteorological preprocessor and, it also incorporates plotting methods of some generated data.

To achieve the outlined objectives a study of the main characteristics of the AERMET was made. Tools of free software and the agile methodology of software development XP was used.

ÍNDICE DE TABLAS E ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

Introducción

En la actualidad, el resultado del desarrollo y progreso tecnológico ha originado diversas formas de contaminación, las que afectan el equilibrio físico y mental del hombre, motivo por lo cual la contaminación se convierte en un problema más crítico que en épocas pasadas. [**1**]

Precisamente la contaminación del aire, es una preocupación a escala mundial, razón por la que en todo el planeta, el hombre realiza estudios acerca de la proyección de los contaminantes atmosféricos. En estos estudios se tienen en cuenta factores característicos de la fuente contaminante y la localización de los receptores es importante a la hora de trazar las estrategias necesarias, para las que se hace imprescindible el uso de modelos de calidad del aire, que son útiles para identificar la contribución de las fuentes a la contaminación del aire y sirven de apoyo a la hora de tomar decisiones y buscar soluciones para mitigar la contaminación atmosférica. [**2**]

Debido a la contaminación existente en Moa la aplicación de estos modelos sería de gran ayuda a la hora de extraer información real sobre la calidad y composición del aire.

Dentro de los modelos de calidad del aire se encuentran los de dispersión, estos a su vez se dividen en modelos de dispersión local o regional, en dependencia de la distancia de la fuente en la que se dispersan los contaminantes. El modelo recomendado por la Agencia de Protección Ambiental de los Estados Unidos (USEPA) [**3**] para estudios detallados de áreas menores de 50 kilómetros (km) es el American Metheorological Society/Environment Protection Agency (AERMOD, por sus siglas en inglés). El sistema de modelos AERMOD, contiene un programa central (AERMOD) y, dos pre-procesadores de datos de entrada que son componentes regulatorios del sistema: AERMAP el pre-procesador de datos de terreno y AERMET, el pre-procesador de datos meteorológicos.

Este sistema de modelación fue implementado utilizando FORTRAN, lenguaje de programación que permanece como el idóneo para desempeñar tareas de computación numérica de alto rendimiento [**4**], como las que realiza el AERMET y aunque el FORTRAN tiene esta excepcional característica, dentro de la literatura revisada no se encontró información acerca de librerías para la construcción de interfaces gráficas de usuario (GUI) en este lenguaje de programación.

El AERMET utiliza un fichero de configuración y varios de datos. El fichero de configuración debe tener el formato exacto que se necesita a la hora de pasárselo al preprocesador; configurar el fichero es muy complejo y solo un usuario con experiencia puede realizarlo, eso se debe en gran parte a que el AERMET no presenta ninguna interfaz de usuario para ayudar en su configuración, pues para ello se debe usar un editor de texto común; además, AERMET no posee ningún método de graficación visual para mostrar datos de salida.
Dadas las condiciones anteriores concluimos nuestro **problema científico** de esta manera: ¿Cómo facilitar al usuario la configuración de los de datos de entrada para el AERMET e interpretación de los resultados generados?

Objeto de Estudio: El preprocesador de datos meteorológicos AERMET.
Campo de acción: Las interfaces de entrada y salida de datos del AERMET.

Hipótesis: Si se desarrolla una interfaz gráfica de usuario amigable, se facilitaría la configuración de los de datos de entrada para el AERMET e interpretación de los resultados generados.

Objetivo General: Desarrollar un software que sirva de interfaz visual para el AERMET y facilite su interacción con el usuario.

Este problema se enmarca en los siguientes **objetivos específicos**:

- Diseñar e implementar una interfaz gráfica de usuario para la configuración de los datos de entrada del AERMET.
- Diseñar e implementar una interfaz gráfica de usuario para representar gráficamente datos generados por el AERMET para ayudar en su interpretación.
- Crear el manual de usuario.

Para dar cumplimiento a los objetivos y resolver la situación problémica planteada, proponemos las siguientes **tareas de investigación**:

- Estudio de las principales características del AERMET.
- Analizar el formato y orden de entrada de los datos al AERMET.
- Analizar el formato de los datos de salida del AERMET.
- Selección de metodología de desarrollo de software, lenguaje y herramientas que se utilizarán.
- Validación del sistema propuesto.

La metodología utilizada es fundamentalmente cualitativa, se emplearon métodos empíricos y teóricos.

Métodos empíricos:

Entre los métodos empíricos usados podemos citar **la entrevista y la observación** para la recopilación de la información. La entrevista permitió determinar los principales requerimientos del sistema y funcionalidades que necesita plasmadas en las historias de usuarios. La observación fue útil para entender el comportamiento del sistema y sus especificaciones.

Métodos teóricos:

Entre los métodos teóricos podemos encontrar:

- **Análisis y síntesis:** mediante el análisis y síntesis de la documentación disponible conocimos el funcionamiento actual de los modelos de dispersión y en la confección del informe final.
- **Hipotético-deductivo:** en la elaboración de la hipótesis, a partir de la cual se realizaran deducciones que arriben a la solución del problema.
- **Histórico-lógico:** para investigar el desarrollo que ha tenido el tema (antecedentes) y apoyar los conocimientos que sobre este existen.
- **Modelación:** la modelación permitió realizar una representación de la realidad, se logró detectar problemas en la forma actual de procesar la información y encontrar las funcionalidades que debe de tener el sistema que se propone, que lo harán más completo y le brindarán satisfacción al usuario con un producto de mayor calidad.

El presente trabajo consta de introducción, cuatro capítulos, conclusiones, recomendaciones, bibliografía, glosario de términos:

Capítulo 1: Contiene la fundamentación teórica del tema, donde se abordan los lenguajes de programación y las tecnologías que se utilizan en el desarrollo de la aplicación. También se exploran soluciones existentes similares al campo de acción para tener una guía de las posibles automatizaciones que se pueden realizar.

Capítulo 2: Aborda los aspectos funcionales para el desarrollo del sistema, se definen los procesos fundamentales por medio de las historias de usuarios creadas por el cliente y se realiza la planificación de entrega de los diferentes módulos que componen la aplicación.

Capítulo 3: Este capítulo contiene los aspectos relacionados con el diseño e implementación del sistema. Se presenta la técnica de tarjetas CRC y además las tareas de ingeniería de cada módulo del sistema para garantizar la entrega en la planificación establecida.

Capítulo 4: Este capítulo está dedicado a las pruebas que se le realizan al funcionamiento del software, las pruebas de aceptación del cliente. Las pruebas se realizan por módulos para la aceptación de cada uno de forma independiente.

Capítulo 5: En este capítulo se realiza un estudio de factibilidad del proyecto, se utilizara la Metodología Costo Efectividad (Beneficio), la cual plantea la conveniencia de la ejecución del proyecto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se describe de forma general los aspectos relacionados con el objeto de estudio y el campo de acción en que se trabaja. Este capítulo constituye la base teórica para la comprensión del trabajo que se desarrolla y los principales aspectos que se abordan son:

1.1 Interfaces

1.1.1 Interfaz

En software, interfaz es parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Esa parte de un programa está constituida por un conjunto de comandos y métodos que permiten estas intercomunicaciones.
Interfaz también hace referencia al conjunto de métodos para lograr interactividad entre un usuario y una computadora. Una interfaz puede ser del tipo GUI, o línea de comandos, etc. También puede ser a partir de un hardware, por ejemplo, el monitor, el teclado y el mouse, son interfaces entre el usuario y el ordenador. [**5**]

1.1.2 Interfaz Gráfica de usuario

En el contexto del proceso de interacción persona - ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.
La interfaz gráfica de usuario (GUI, por sus siglas en inglés) es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.
La GUI surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. [**6**]

1.2 Preprocesador meteorológico.

Los modelos de dispersión de contaminantes atmosféricos son herramientas que permiten calcular la proyección de un contaminante. Estos requieren numerosos datos meteorológicos, algunos de ellos son medidos de forma rutinaria en las estaciones meteorológicas -denominados **primarios**-, pero otros no lo son y por tanto deben ser inferidos de los primeros, a estos se les llama **secundarios**. [**7**] Generar estos datos secundarios es el principal objetivo de un preprocesador meteorológico.

Ejemplo de datos **primarios:**

- Dirección del viento.
- Velocidad del viento.

Ejemplo de datos **secundarios:**

- Categorías de estabilidad atmosférica: A, B, C, D, E, F.
- Altura de la capa de mezcla.

1.2.1 El preprocesador meteorológico AERMET

AERMET es el preprocesador meteorológico del AERMOD, que procesa la capa límite y otros parámetros necesarios para el funcionamiento del modelo y acepta datos de fuentes ubicadas tanto en el sitio como fuera de este. AERMET crea dos archivos: un archivo de datos de superficie y un archivo de datos de viento. [**8**]

AERMET tiene tres pasos:

1) Extracción y chequeo de los datos.
2) Unión de todos los datos meteorológicos
3) Cálculo de variables secundarias y creación de los archivos meteorológicos que AERMOD necesita.

En la versión de AERMET (04300) los pasos 1 y 2 se resuelven con STAGE1N2.exe y el 3 con STAGE3.exe, pero en la versión 06341 ambos ejecutables se unen en Aermet.exe. Tanto en la versión 04300 como en la 06341 se necesitan tres archivos de control para cada

paso. Se crea un archivo .bat para correr consecutivamente los tres pasos de AERMET, que permite no tener que renombrar los archivos de control y de salida de cada paso.

AERMET se puede alimentar con los siguientes datos meteorológicos:

1. Datos in situ (en el sitio específico donde se realizaron las mediciones), en formato libre que el usuario configura dado un listado posible de variables. Incluye datos de torres de gradientes para dos o más niveles en la altura.
2. Datos de superficie con un formato pre-establecido.
3. Datos de sondeo.

1.3 Otros softwares relacionados al objeto de estudio y campo de acción que han implementado interfaces gráficas.

1.3.1 AERMOD VIEW 6.0

El ISC-Aermod View es software basado en un modelo de dispersión de contaminantes recomendado por la USEPA que contiene entre otros componentes dos programas de preprocesado de bases de datos meteorológicos. Estos programas se nombran AERMET View y RAMMET View, y son necesarios para la modelación en ISC-AERMOD view. Estos usan los siguientes módulos:

- Hourly
- Upper Air Sectors
- On-Site
- Output
- WRplot

1.3.2 CALPUFF View

CALPUFF View es un pre y post procesador para el CALPUFF, el cual es un sistema de modelación de transporte y dispersión de contaminantes. Este también posee un preprocesador de datos meteorológicos llamado CALMET, que brinda facilidades de graficación y análisis de datos en tres dimensiones.

1.3.3 Características.

Estos softwares que trabajan la misma temática e implementan interfaces visuales amigables poseen muy buenas características, pero también las desventajas de no ser libres ni gratuitos.

Se debe mencionar que como estos softwares están basados en los modelos asesorados por la EPA, y en el caso del AERMOD (incluyendo el AERMET y los demás preprocesadores) como modelo regulatorio, poseen la siguiente característica:

- No implementa una nueva versión del modelo, sino que usa la misma establecida como modelo regulatorio.
- La interfaz que muestra es solo la fachada para ayudar al usuario en el trabajo con el mismo.

Esto se debe a que como el modelo es regulatorio, cualquiera que implemente una interfaz debe hacerlo corriendo el ejecutable creado por la EPA, a menos que la misma EPA actualice el modelo.

De acuerdo con los softwares expuestos y sus características, pensamos que usando el mismo software AERMET de consola obtenido de la web de la EPA, podemos ganar ventaja de la situación si implementamos nosotros mismos en el país las interfaces visuales requeridas, como ejemplo nuestro caso, las interfaces gráficas de entrada y salida del AERMET.

1.4 Tendencias y tecnologías actuales

1.4.1 Política de Migración a Software libre

Se denomina software libre a todo aquel que permita a los usuarios ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. A menudo es confundido con el software gratuito, sin embargo no se trata de una cuestión de precio sino de libertad. Precisamente, las cuatro libertades que se definen son:

- La libertad de ejecutar el programa para cualquier propósito.
- La libertad de estudiar cómo trabaja el programa y adaptarlo a sus necesidades (El acceso al código fuente es una condición necesaria).

- La libertad de redistribuir copias para que pueda ayudar al vecino.
- La libertad de mejorar el programa y publicar sus mejoras y versiones modificadas en general para que se beneficie toda la comunidad (El acceso al código fuente es una condición necesaria). [**9**]

Las ventajas especialmente económicas que brindan las soluciones libres y las aportaciones de la comunidad de desarrollo han permitido un constante crecimiento del software libre hasta superar en ocasiones, como en el caso de los servidores web, al mercado propietario. Estas ventajas hacen que nuestro país siga una política de migración hacia el software libre y como parte de este proceso se decide para el desarrollo de la aplicación la utilización de herramientas y tecnologías pertenecientes al software libre.

1.4.2 Lenguajes de Programación

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. [10] A continuación se realiza un estudio de los principales lenguajes de programación existentes que se utilizan para la creación de aplicaciones de escritorio, como son Delphi, C++ y Java.

Delphi.

Es un entorno de desarrollo de software (IDE) diseñado para la programación de propósito general con énfasis en la programación visual. Es producido comercialmente por la empresa estadounidense CodeGear. En sus diferentes variantes, permite producir archivos ejecutables para Windows, Linux y la plataforma .NET.

Está basado en una versión moderna de Pascal, denominada Object Pascal. Borland en los últimos años defendía que el nombre correcto del lenguaje es también Delphi, posiblemente debido a pretensiones de marca, aunque en sus mismos manuales el nombre del lenguaje aparecía como Object Pascal, por lo que la comunidad de programadores no ha adoptado mayoritariamente este cambio (supuesta aclaración, según Borland). Object Pascal expande las funcionalidades del Pascal estándar:

Soporte para la programación orientada a objetos (habitualmente llamada POO) también existente desde Turbo Pascal 5.5, pero más evolucionada en cuanto a:

- Encapsulación: declarando partes privadas, protegidas, públicas y publicadas de las clases
- Propiedades: concepto nuevo que luego han adaptado muchos otros lenguajes. Las propiedades permiten usar la sintaxis de asignación para setters y getters.
- Simplificación de la sintaxis de referencias a clases y punteros.
- Soporte para manejo estructurado de excepciones, mejorando sensiblemente el control de errores de usuario y del sistema.
- Programación activada por eventos (event-driven), posible gracias a la técnica de delegación de eventos. Esta técnica permite asignar el método de un objeto para responder a un evento lanzado sobre otro objeto. Fue adoptada por Niklaus Wirth, autor del Pascal Original, e incorporada a otros de sus lenguajes como Component Pascal.

C++.

Es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. El nombre C++ fue propuesto por Rick Masciatti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases".

Se puede decir que abarca tres paradigmas de la programación:

- La programación estructurada.
- La programación genérica.
- La programación orientada a objetos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución

Está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos

(obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

Java.

Es un lenguaje de programación desarrollado por Sun Microsystems a principios de los años 1990. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

Las características principales que nos ofrece Java son:

- **Simple**

Ofrece toda la funcionalidad de un lenguaje potente. Debido a que C y C++ son los lenguajes más difundidos, Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.

Elimina muchas de las características de otros lenguajes para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica).

No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

- **Orientado a objetos**

Con el objetivo de mantener la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, necesitan ser construidas y destruidas en espacios de memoria.

Incorpora funcionalidades como por ejemplo, la resolución dinámica de métodos mediante una interfaz específica llamada RTTI (RunTime Type Identification) que define la interacción entre objetos excluyendo variables de instancias o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

- **Distribuido**

Se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.

Proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

- **Robusto**

Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo.

Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

También implementa los arrays auténticos, en vez de listas enlazadas de punteros, con comprobación de límites, para evitar la posibilidad de sobreescribir o corromper memoria

resultado de punteros que señalan a zonas equivocadas. Estas características reducen drásticamente el tiempo de desarrollo de aplicaciones en Java.

- **Arquitectura neutral**

Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (run-time) puede ejecutar ese código objeto, sin importar en modo alguno la máquina en que ha sido generado.

Actualmente existen sistemas run-time para Solaris 2.x, SunOs 4.1.x, Windows 95, Windows NT, Linux, Irix, Aix, Mac, Apple y probablemente haya grupos de desarrollo trabajando en el porting a otras plataformas.

- **Seguro**

La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el casting se eliminan para prevenir el acceso ilegal a la memoria. Cuando se usa Java para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.

El código Java pasa muchos tests antes de ejecutarse en una máquina. El código se pasa a través de un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal -código que falsea punteros, violaderechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto.

El Cargador de Clases también ayuda a Java a mantener su seguridad, separando el espacio de nombres del sistema de ficheros local, del de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo Caballo de Troya, ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior.

Las clases importadas de la red se almacenan en un espacio de nombres privado, asociado con el origen. Cuando una clase del espacio de nombres privado accede a otra clase, primero se busca en las clases predefinidas (del sistema local) y luego en el espacio de nombres de la clase que hace la referencia. Esto imposibilita que una clase suplante a una predefinida.

- **Multihebra**

Soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Fundamentación de la selección de Java como lenguaje de programación.

Atendiendo las características de los lenguajes antes mencionados hemos seleccionado Java para el desarrollo de nuestra aplicación por contar con las siguientes características:
Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- Aritmética de punteros.
- No existen referencias.
- Registros (struct).
- Definición de tipos (typedef).
- Macros (#define).
- Necesidad de liberar memoria (free).

Aunque, en realidad, lo que hace es eliminar las palabras reservadas (struct, typedef), ya que las clases son algo parecido. Tiene ventajas en cuanto a seguridad ya que al tener la característica de ser interpretado el código Java pasa a través de un verificador de byte-codes que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal -código que falsea punteros, viola derechos de acceso sobre objetos o intenta cambiar el tipo o clase de un objeto. El Cargador de Clases, separando el espacio de nombres del sistema de ficheros local, del de los recursos procedentes de la red. Esto limita cualquier aplicación del tipo Caballo de Troya, ya que las clases se buscan primero entre las locales y luego entre las procedentes del exterior.

1.4.3 Ambiente de Desarrollo Integrado (IDE) elegido: NetBeans 6.5.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun

MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

La Plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear aplicaciones. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

El IDE NetBeans 6.0 es una herramienta para programadores, pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, JavaEE, web, EJB y aplicaciones móviles). Entre sus características se encuentran un sistema de proyectos basado en Ant, control de versiones y refactoring.

El IDE NetBeans IDE 6.0 corre en diferentes sistemas operativos entre los que podemos mencionar linux, Mac OS X, Solaris y Windows. Como se puede observar se ha lanzado con soporte para la mayoría de las plataformas. Como requerimiento se necesita tener previamente instalado el JDK 5.0 o 6.0.

1.5 Arquitectura

La Arquitectura es el esqueleto o base de una aplicación. Representa la organización fundamental de un sistema. Desde los pequeños programas hasta los sistemas más grandes poseen una estructura y un comportamiento que los hace clasificables según su "arquitectura". En la Web es muy común la utilización de la arquitectura "3-capas", "n-capas", "MVC", entre otras. El lenguaje de programación JAVA, seleccionado anteriormente, implementa a su vez el patrón arquitectónico MVC, es por ello que adoptamos esta arquitectura para el desarrollo de la propuesta de solución. Modelo Vista Controlador (MVC). Es un patrón de diseño de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- **Modelo:** Representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- **Vista:** Presenta el modelo en un formato adecuado, como en una página Web que le permite al usuario interactuar con ella, usualmente un elemento de interfaz de usuario.
- **Controlador:** Responde a eventos, usualmente acciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) logrando un mantenimiento más rápido y sencillo de las aplicaciones. Ejemplo, para el caso de la web, si se fuera a mostrar una misma aplicación en un navegador estándar, como en un navegador de un dispositivo móvil, sólo es necesario crear una vista nueva por cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (Aplicación de escritorio, HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

1.5.1 Ventajas del Modelo Vista Controlador

- La separación del Modelo de la Vista, es decir, separa los datos de la representación visual de los mismos.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Permite el escalamiento de la aplicación en caso de ser requerido.

1.6 Metodologías de Desarrollo De Software

1.6.1 Proceso de desarrollo de software

La calidad en el desarrollo y mantenimiento del software se ha convertido hoy en día en uno de los principales objetivos estratégicos de las organizaciones, debido a que cada vez más,

los procesos principales dependen de los sistemas informáticos para su buen funcionamiento. En los últimos años se han publicado diversos estudios y estándares en los que se exponen los principios que se deben seguir para la mejora de los procesos de software.

El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo".

Las piedras angulares del proceso de desarrollo del software son: el proyecto, las personas y el producto; siendo las características del cliente, el entorno de desarrollo y las condiciones del negocio, elementos que influyen en el proceso. Existe una estrecha relación entre personas, proyecto, producto y proceso. Estos términos son conocidos como las cuatro «P» en el desarrollo de software. El resultado final de un proyecto software es un producto, donde intervienen personas a través de un proceso de desarrollo de software que guía los esfuerzos de las personas implicadas en el proyecto.

1.6.2 Metodologías

Hoy en día, llevar a cabo el desarrollo de un buen software depende de un gran número de actividades y etapas donde elegir la mejor metodología para el equipo influye directamente en el futuro éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo.

Una metodología para el desarrollo de un proceso de software es un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas informáticos.

Las metodologías existentes en la actualidad se dividen en dos grandes grupos atendiendo a sus características: las metodologías tradicionales (RUP, MSF) y las metodologías ágiles (XP, SCRUM). Las primeras están pensadas para el uso exhaustivo de documentación

durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente.
Teniendo en cuenta ambos enfoques damos paso al análisis de dos de las metodologías más usadas actualmente.

El Proceso Unificado de Desarrollo (RUP).

Es un proceso para el desarrollo de un software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Como tres características esenciales está dirigido por casos de uso: que orientan al proyecto a la importancia para el usuario y lo que se quiere, está centrado en la arquitectura:
que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y es iterativo e incremental: donde divide el proyecto en mini-proyectos donde los casos de uso y al arquitectura cumplen sus objetivos de manera depurada. RUP propone cuatro etapas para el desarrollo de un producto: Inicio, Elaboración, Construcción y Transición, cada una de ellas compuesta de una o varias iteraciones. Estas etapas revelan que para producir una versión del producto en desarrollo se emplean todas las actividades de ingeniería pero con diferente énfasis; en las primeras versiones se hace más énfasis en el modelado del negocio, requisitos, análisis y diseño; mientras en las posteriores el énfasis recae sobre las actividades de implementación, pruebas y despliegue. Además contempla flujos de trabajo de soporte que involucran actividades de planificación de recursos humanos tecnológicos y financieros. El Proceso Unificado de Desarrollo tiene 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Flujos de trabajo:

- **Modelamiento del negocio:** Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- **Requerimientos:** Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

- **Análisis y diseño:** Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- **Implementación:** Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- **Prueba (Testeo):** Busca los defectos a los largo del ciclo de vida.
- **Instalación:** Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.
- **Administración del proyecto:** Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- **Administración de configuración y cambios:** Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- **Ambiente:** Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Es una de las metodologías más generales y más usadas de las que existen en la actualidad, pues está pensada para adaptarse a cualquier proyecto. Constituye además, una propuesta de proceso para el desarrollo de software orientado a objeto, utilizando UML (Unified Model Language), para describir todo el proceso basándose en componentes. Este lenguaje es estándar, con él se puede visualizar, especificar, construir y documentar los artefactos de un sistema.

XP (Extreme Programming).

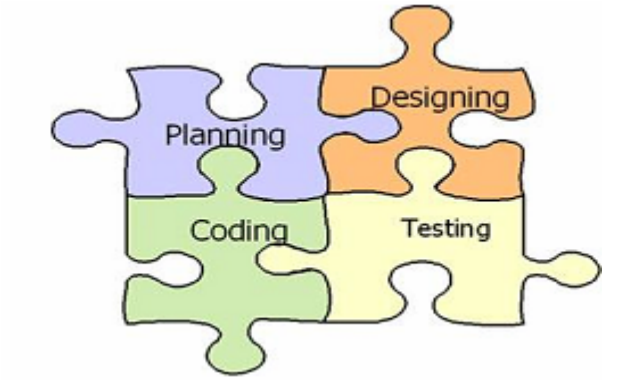

Ilustración 1: Metodología XP

Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los programadores que la practican consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Es utilizada para proyectos de corto plazo, equipo pequeño y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. [**11**]Las características fundamentales son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del

código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.

- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Ventajas

- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para los clientes, ya conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- Permite definir en cada iteración cuales son los objetivos de la siguiente.
- Permite la retroalimentación.
- La presión esta a lo largo de todo el proyecto y no en una entrega final.

Desventajas

- Delimitar el alcance del proyecto con nuestro cliente.

Para mitigar esta desventaja se plantea definir un alcance a alto nivel basado en la experiencia.

Fundamentación de la selección de la metodología de desarrollo de software.

XP y RUP son dos grandes metodologías que después de analizar sus principales características y los aspectos más sobresalientes de cada una de ellas, se ha determinado la implantación de XP, una metodología ligera, con menos requerimientos de documentación y planificación para el desarrollo de la aplicación. Si se escoge RUP traería grandes dificultades como son:

Multitud de artefactos: El hecho de realizar varios artefactos y mantenerlos actualizados consume mucho tiempo.

El poco personal de desarrollo: Al ser solo una persona a cargo del desarrollo de la aplicación, este tomaría varios roles en cada etapa y sería muy complejo cumplir con las actividades de cada uno de ellos.

En futuras versiones: Se piensa añadirle otras funcionalidades, destacando que se desea lograr una funcionalidad mínima del software lo mas pronto posible.

Se necesita: La documentación mínima necesaria para el futuro soporte y mantenimiento del producto final.

Requisitos cambiantes: Los cambios en un proceso de desarrollo son inevitables, al aparecer un nuevo requisito hace que se tenga que comenzar una nueva iteración para dar cumplimiento a su funcionalidad. Como el proyecto esta en plena investigación y en un ambiente de desarrollo sujeto a cambios repentinos se sugieren una gran adaptatividad y pronta respuesta, lo cual RUP no ofrece.

Planificación inexistente: La planificación que se realiza en las fases iniciales está sujeta a muchas variaciones en dependencia de los cambios que se experimenten en los requisitos. Por tanto se hace muy difícil planificar actividades específicas si no se tiene claro que se debe hacer realmente.

Los inconvenientes planteados pueden ser eliminados con la utilización de la metodología XP.

1.7 Conclusiones

En este capítulo hemos abordados los conceptos fundamentales asociados al dominio del problema, relacionados con el objeto de estudio y el campo de acción. Además realizamos un estudio de lo más utilizado en cuanto a los múltiples lenguajes, metodologías y tecnologías para el desarrollo de aplicaciones existentes, escogiendo así lo que presenta mayor ventaja con respecto a las características de nuestro sistema. Entre lo seleccionado se encuentra JAVA como lenguaje de programación y NetBeans como IDE de programación; además, se decide realizar la aplicación sobre la base de la metodología ágil XP, pues nos permitirá obtener resultados funcionales observables a corto plazo.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

Introducción

En el presente capítulo se describen los parámetros principales que constituyen la entrada y salida de datos del AERMET, que es el punto de partida para construcción de las interfaces de entrada y salida del mismo. Se presenta de la primera fase de la metodología, las historias de usuarios realizadas por el cliente, así como toda la planificación de entrega para su implementación.

2.1 Especificaciones de configuración para los datos de entrada de AERMET

Como no se tiene un formato o contenido estándar para los datos de sitios "on-site", el usuario debe especificar el fichero de configuración para el AERMET. El formato de los datos de sitios específicos es razonablemente flexible, y esta sujeto a las siguientes reglas:

1) Los datos para un período de observación pueden ser uno o mas records, y los records para el período debe ser contiguos (lo identificaremos aquí como grupo de observación).
2) Pueden existir hasta 12 observaciones igualmente espaciadas por hora, ej.: con un intervalo de cada 5 minutos (5 minutos es el mínimo intervalo permitido).
3) El mismo juego de variables tiene que aparecer para todos los períodos de observación pero no las mismas variables tienen que estar presente en cada record en el período de observación, ej.: Suponiendo una medición por hora, cada record puede estar dado por una medición en cada nivel de la torre instrumentada, y es perfectamente admisible que en cada nivel se esté midiendo un grupo de variables que no tiene que ser igual a ninguno de los otros medidos en los otros niveles.
4) La información de fecha/tiempo para cada observación debe de estar contenida en el primer record del grupo; esto puede darse en cualquier orden dentro del primer record y tienen que estar en formato de número entero; las variables meteorológicas del sitio específico o Estación de Superficie pueden aparecer en el primer record.
5) Las variables presentes dentro de cada observación deben ser un subconjunto de aquellas listadas en los anexos ANEXO 1 y ANEXO 2 (Aunque el usuario puede orientar al AERMET que salte campos).

6) Las variables de nivel simple (ej: heat flux y las mixing heights observadas) tienen que ser leídas entes de cualquier variables de nivel múltiple (ej: temperature).

2.1.1 Palabras claves o de especificación para el AERMET

EL sistema de modelación AERMOD cuenta en general con una forma de configuración orientada por ficheros con palabras claves, que son las especificaciones consecuentes a los datos que representan, es decir, seguida de una palabra clave se encontrará el o los datos que corresponden al uso de dicha palabra, también puede darse el caso de que luego de una palabra clave le siga un bloque de palabras claves que a su vez pertenecen a dicha palabra. Las palabras claves que a continuación se muestran representan las reglas o restricciones bajo las que se ha construido la interfaz visual perteneciente a este trabajo.

Palabras clave y explicación:

JOB: Inicio del fichero de configuración donde se especifican los ficheros de mensajes.
MESSAGES (pertenece a JOB): Fichero de mensajes escritos por AERMET.
REPORT (pertenece a JOB): Fichero resumen de reportes escritos por AERMET.
ONSITE: La palabra ONSITE indica que le sigue un bloque de palabras clave que pertenecen a ONSITE. Las palabras clave básicas del bloque ONSITE son:

- **DATA** – Especifica el nombre del fichero de entrada con los datos.
- **XDATES** – Especifica el período de tiempo a analizar dentro de los datos del fichero.
- **LOCATION** – Indica el identificador de la estación, latitud, longitud y el valor de tiempo para convertirlo a la hora local estándar.
- **QAOUT** – Especifica el nombre del fichero de salida del proceso de análisis; esta palabra también es usada para identificar el fichero de entrada al paso 2.
- **READ** – Define el orden de las variables por orden de aparición en el fichero de datos; esta palabra es repetida cuantos records existan por período de observación.
- **FORMAT** – Define el formato de las variables por orden de aparición en el fichero de datos; por cada READ debe existir un FORMAT.

- **THRESHOLD** – Define la velocidad de viento mínima permitida en los datos de sitio específico; cualquier valor de viento por debajo de este valor es considerado como calma.
- **RANGE** – Modifica los valores por defecto de los límites menor y mayor así como el valor a escribir en caso de algún valor fuera de rango para la variable meteorológica especificada.

Todas estas palabras excepto la última son obligatorias. El orden de estas palabras dentro del bloque de palabras clave pertenecientes a ONSITE no es importante.

AERMET a partir de su preprocesamiento genera dos archivos: un archivo de datos de superficie y un otro de datos de viento, cuyas extensiones son .SFC y .PFL respectivamente. El archivo .PFL contiene un perfil de los datos de viento que se graficarán en la interfaz de salida.

2.2 Personas relacionadas con el sistema

Persona(s) relacionadas con el sistema	Justificación
Especialista	Esta es la persona que tiene cierto conocimiento en la materia de dispersión de contaminantes, y esta encargada de reconocer los datos que están en el fichero con la data meteorológica; se encarga también de graficar.

Tabla 1: Personas relacionadas con el sistema.

2.3 Planificación

La planificación es la primera fase de la metodología XP, en la misma quedan definidos los procesos que el cliente quiere automatizar y el tiempo en que terminará la construcción la aplicación.

2.4 Exploración

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Estos constituyen el resultado directo de la interacción entre los clientes y desarrolladores a través de reuniones donde las lluvias de ideas determinan no solo los requerimientos del proyecto sino también las posibles soluciones. De forma general se describen brevemente las características que el sistema debe tener desde el punto de vista del cliente. Para definir las historias de usuario se emplea la siguiente plantilla.

Historia de Usuario	
Número: (Número de la Historia de Usuario)	**Usuario:** (Usuario entrevistado para obtener la función requerida a automatizar)
Nombre historia: (Nombre de la historia de usuario que sirve para identificarla mejor entre los desarrolladores y el cliente)	
Prioridad en negocio: (Importancia de la historia para el cliente) (Alta / Media / Baja)	**Riesgo en desarrollo:** (Dificultad para el programador) (Alto / Medio / Bajo)
Puntos estimados: 2	**Iteración asignada:** (Iteración a la que corresponde)
Programador responsable: Dariel Subirós	
Descripción: (Se especifican las operaciones por parte del usuario y las respuestas que dará el sistema)	
Observaciones: (Algunas observaciones de interés, como glosario, información sobre usuarios, etc.)	

Tabla 2: Representación de una historia de usuario.

A continuación se muestran las historias de usuarios confeccionadas por el cliente, teniendo estas las funcionalidades principales a integrar en la aplicación.

Historia de Usuario	
Número: 1	**Usuario:** Liban Montes
Nombre historia: Identificación de Estación y fichero con data meteorológica.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Dariel Subirós	
Descripción: El usuario debe seleccionar el fichero de entrada con la data meteorológica, e identificar la Estación de Superficie a la que pertenece.	
Observaciones: Se debe validar el fichero de entrada: El fichero con los datos "on-site" no tiene un formato específico, pero teniendo en cuenta las posibles características que debe poseer el fichero, se emitirán mensajes de error en caso de fallo. Las características posibles son: ❖ Solo puede tener números. ❖ Debe contener al menos fecha y hora	

Tabla 3 : H.U. Identificación de Estación y fichero con data meteorológica.

Historia de Usuario	
Número: 2	**Usuario:** Allan Pierra
Nombre historia: Configuración de formato de datos meteorológicos	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Dariel Subirós	
Descripción: El usuario debe identificar formato de los datos meteorológicos de entrada.	
Observaciones: Atendiendo las reglas de READ y FORMAT el usuario de forma visual debe identificar las variables que se encuentran en el fichero. El número de variables por record debe corresponderse con el número de columnas identificadas al leer el fichero.	

Tabla 4 : H.U. Configuración de formato de datos meteorológicos.

Historia de Usuario	
Número: 3	**Usuario:** Allan Pierra
Nombre historia: Configuración de Leyenda	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Dariel Subirós	
Descripción: Se crea la leyenda para mostrar los datos del fichero de salida de perfil del AERMET. Se debe permitir al usuario crear una Leyenda nueva, modificarla y visualizarla.	
Observaciones: La leyenda viene con una configuración por defecto, pero se puede variar al gusto del usuario: • Puede redefinir Clases de Viento • Puede redefinir Colores correspondientes a cada Clase de Viento	

Tabla 5 : H.U. Configuración de Leyenda.

Historia de Usuario	
Número: 4	**Usuario:** Liban Montes
Nombre historia: Configuración de Opciones de graficación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Dariel Subirós	
Descripción: El usuario debe configurar las opciones de graficación que desee o usar las opciones por defecto.	
Observaciones: Se divide 360° entre la cantidad de direcciones especificadas por el usuario, y se obtiene la cantidad de *Rayos* a representar. **Nota 1 :** El arco tiene una amplitud de 360°/Cantidad de direcciones **Nota 2**: Rayo es la suma de las mediciones de clases de viento en el arco calculado.	

Tabla 6 : H.U. Configuración de Opciones de graficación.

Historia de Usuario	
Número: 5	Usuario: Liban Montes
Nombre historia: Graficación	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 3	**Iteración asignada:** 5
Programador responsable: Dariel Subirós	
Descripción: Se debe graficar la información extraída mediante la Historia de Usuario 4	
Observaciones: Se realizan las tareas de graficación que representan datos de viento extraídos desde el fichero de salida del AERMET con extensión .PFL	

Tabla 7 : H.U. Graficación.

2.5 Planificación de entregas

En esta parte se establece la prioridad de cada historia de usuario así como una estimación del esfuerzo necesario de cada una de ellas con el fin de determinar un cronograma de entregas.

Las estimaciones de esfuerzo asociado a la implementación de las historias se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación(6 días). Las historias generalmente valen de 1 a 3 puntos. Por otra parte, se mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del

sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.5.1 Estimación de esfuerzos por historias de usuario

Historia de usuario	Número	Puntos Estimados
Identificación de Estación y fichero con data meteorológica	1	2
Configuración de formato de datos meteorológicos	2	3
Configuración de Leyenda	3	3
Configuración de Opciones de graficación	4	2
Graficación	5	3

Tabla 8: Estimaciones de esfuerzo por historias de usuario.

El plan de entregas se realiza teniendo en cuenta las unidades funcionales que se quieren entregar y cada uno de estos módulos abarca un número de historias de usuarios a implementar para dar cumplimiento al funcionamiento del mismo.

2.5.2 Plan de entregas

Módulo	Historia(s) de Usuario que abarca
• Configuración de datos meteorológicos	1,2
• Graficación de datos meteorológicos	3,4,5

Tabla 9: Plan de entregas.

2.5.3 Plan de duración de las iteraciones

Con respecto a las Historias de Usuario previamente presentadas realizamos una planificación en cinco iteraciones basándonos en el tiempo y procurando obtener la funcionalidad relacionada en la misma iteración.

Iteración	Orden de implementación por Historias de Usuario	Duración total de la iteración en semanas
1	• Identificación de Estación y fichero con data meteorológica.	2
2	• Configuración de formato de datos meteorológicos	3
3	• Configuración de Leyenda. • Configuración de Opciones de graficación.	2+3 = 5
4	• Graficación.	3

Tabla 10: Duración de las iteraciones

Combinando el plan de entrega y el plan de iteraciones se harán *releases o liberaciones* al sistema en las fechas mostradas a continuación:

Iteración \ Módulo	Configuración de datos meteorológicos.	Graficación de datos meteorológicos.
Final 1ra Iteración	14 febrero 2009	
Final 2a Iteración	7 marzo 2009	
Final 3ra Iteración		11 abril 2009
Final 4ta Iteración		2 mayo 2009

Tabla 11: Tabla de releases

2.6 Conclusiones

La configuración de los datos de entrada y la representación de los datos de salida del AERMET, junto con las historias de usuarios creadas por el cliente es el objeto de automatización de la presente investigación. A partir de estas historias se planifica la entrega del software, la misma se realiza conjuntamente con el cliente que es el actor principal en la etapa de planificación de la metodología seleccionada. Se decide realizar el software en cuatro iteraciones de 2, 3, 5 y 3 semanas de trabajo cada una para garantizar la entrega al cliente en el tiempo establecido.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Introducción

En este capítulo se presentan las fases diseño e implementación de la metodología XP. Uno de los artefactos principales es la creación de las tarjetas CRC (Clase-Responsabilidades-Colaboración) las cuales permiten brindar un mayor enfoque orientado a objetos. Por otra parte se describen cada una de las tareas confeccionadas para llevar cumplir con el desarrollo de cada una de las historias de usuario detectadas.

3.1 Diseño

3.1.1 Modelo-Vista-Controlador (MVC) en Java Swing

Java tiene como características fundamentales su portabilidad a un gran número de plataformas (Java es en sí una plataforma), su simplicidad y su extenso conjunto de librerías.
Su arquitectura está profundamente basada en MVC, lo que proporciona un alto grado de extensibilidad y de personalización de los componentes de la librería.
Permite "conectar" y "desconectar" estilos de interfaz de usuario (llamados "look and feels") que modifican la forma en que se muestra y se comporta toda la interfaz de usuario, así, la misma aplicación puede verse como una aplicación Windows o como una aplicación Motif7 simplemente conmutando el look and feel en tiempo de ejecución.

Ventajas

La implementación de Swing del patrón de diseño MVC presenta muchas ventajas:
Permite la creación de interfaces de usuario de una manera sencilla y rápida, permitiendo el manejo del patrón MVC pero ocultando los detalles de su implementación. El mecanismo de eventos de Java se adapta perfectamente al mecanismo de notificaciones de MVC. Al estar los modelos separados de la vista, las posibilidades de extensión de la librería y de personalización de componentes ya existentes son enormes. Permite al usuario crear sus propias estructuras de datos y adaptar la interfaz de usuario a ellas y no a la inversa, como sucede con librerías ya implementadas.

3.2 Tarjetas CRC

El uso de las tarjetas C.R.C (Class, Responsabilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica.

Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

Esta nueva técnica de diseño es adoptada como alternativa a los diagramas UML de las clases, pues en estas se plasman las responsabilidades que tienen cada objeto y las clases con las que tienen que interactuar para darles respuesta brindando así la información que se necesita a la hora de implementar.

Clase	
Responsabilidades	**Colaboraciones**
Clase A	Clase B

Tabla 12: Plantilla de Tarjeta CRC.

Para una mejor comprensión de las tarjetas C.R.C de nuestro sistema procedemos a agruparlas por los módulos identificados en el Plan de Entregas en el capítulo 2.

3.2.1 Tarjetas CRC del módulo #1:Configuración de datos meteorológicos

VisualAermet	
Responsabilidades	**Colaboraciones**
Validar configuración de datos	FormatEditting
Escribir fichero de entrada del AERMET	VisualAermet.FormattedModel

Tabla 13: Tarjeta CRC de VisualAermet.

VisualAermet.FormattedModel	
Responsabilidades	**Colaboraciones**
Almacenar variables configuradas	

Tabla 14: Tarjeta CRC de VisualAermet.

FormatEditting	
Responsabilidades	**Colaboraciones**
Trabajo con variables	MetRecordsFormatting VisualAermet.FormattedModel
Aplicar restricciones	

Tabla 15: Tarjeta CRC de FormatEditting.

FileDataToModel	
Responsabilidades	**Colaboraciones**
Validar fichero escogido	
Extraer records a configurar	

Tabla 16: Tarjeta CRC de FileDataToModel.

MetRecordsFormatting	
Responsabilidades	**Colaboraciones**
Insertar variables	MetRecordsFormatting.OnSiteVariable MetRecordsFormatting.Multilevel
Eliminar variables	
Modificar variables	
Buscar variables	

Tabla 17: Tarjeta CRC de MetRecordsFormatting.

OnsiteHeights	
Responsabilidades	**Colaboraciones**
Insertar alturas de sitio específico	OnsiteHeights.MyCellRenderer
Eliminar alturas de sitio específico	
Mostrar alturas de sitio específico	

Tabla 18: Tarjeta CRC de OnsiteHeights.

3.2.2 Tarjetas CRC del módulo #2:Graficación de datos meteorológicos

VisualAermetPlotter	
Responsabilidades	**Colaboraciones**
Mostrar gráfico de Rosa de Vientos según Leyenda	MetData VisualAermetPloter.MyCellRenderer
Mostrar tabla de frecuencias	
Mostrar gráfico de barras	

Tabla 19: Tarjeta CRC de VisualAermetPlotter.

Preferentes	
Responsabilidades	**Colaboraciones**
Mostrar leyenda	VisualAermetPlotter
Modificar leyenda	

Tabla 20: Tarjeta CRC de Preferences.

WindRose	
Responsabilidades	**Colaboraciones**
Crear Rosa de Vientos y Leyenda	Legend MetData.ray

Tabla 21: Tarjeta CRC de WindRose.

Legend	
Responsabilidades	**Colaboraciones**
Almacenar leyenda	Legend.Rango

Tabla 22: Tarjeta CRC de Legend.

MetData	
Responsabilidades	**Colaboraciones**
Crear rayos	MetData.Ray MetData.Ray.WD_WS Legend
Obtener tabla de frecuencias	

Tabla 23: Tarjeta CRC de MetData.

Ray	
Responsabilidades	**Colaboraciones**
Almacenar conjunto mediciones por dirección	MetData.Ray.WD_WS

Tabla 24: Tarjeta CRC de MetData.

Ray.WD_WS	
Responsabilidades	**Colaboraciones**
Almacenar par de dirección-velocidad de viento	MetData.Ray.WD_WS

Tabla 25: Tarjeta CRC de MetData.

BarChart	
Responsabilidades	**Colaboraciones**
Crear gráfico de barras	MetData

Tabla 26: Tarjeta CRC de BarChart.

3.3 Implementación

En la metodología XP se convierte en un integrante más del equipo de desarrollo el cliente pues él crea las historias de usuario bajo la supervisión de los desarrolladores. Estas historias quedan confeccionadas cuando el cliente es capaz de identificar con precisión la funcionalidad deseada, además, también debe estar presente cuando se realicen las pruebas de aceptación para cada historia, por lo que su presencia es imprescindible.

En XP generalmente cada historia de usuario se divide en tareas de ingeniería (TI) o tareas de programación. Estas se crean para obtener una mejor planificación de la historia; con

ellas se pretende cumplir con las funcionalidades básicas que luego conformarán las funcionalidades generales de cada historia.

A continuación se presentan las Tareas de Ingeniería agrupadas por las respectivas historias de usuario a las que pertenecen.

3.3.1 Tareas por Historias de Usuario

Tareas de la Historia de Usuario #1:

Tarea	
Número tarea: 1	**Número historia:** 1
Nombre tarea: Identificación de Estación de Superficie y selección de fichero de entrada.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 2 febrero 2009	**Fecha fin:** 7 febrero 2009
Programador responsable: Dariel Subirós	
Descripción: Se valida la correcta apertura del fichero escogido por el usuario para la importación al sistema de la data meteorológica así como los datos de identificación de la estación de superficie	

Tabla 27: TI Identificación de Estación de Superficie y selección de fichero de entrada.

Tarea	
Número tarea: 2	**Número historia:** 1
Nombre tarea: Creación de la interfaz gráfica principal.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9 febrero 2009	**Fecha fin:** 14 febrero 2009
Programador responsable: Dariel Subirós	
Descripción: Se codifica la interfaz inicial del programa	

Tabla 28: TI Se crea la interfaz gráfica principal.

Tareas de la Historia de Usuario #2:

Tarea	
Número tarea: 1	**Número historia:** 2
Nombre tarea: Visualización al usuario del contenido del fichero de entrada.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 16 febrero 2009	**Fecha fin:** 18 febrero 2009
Programador responsable: Dariel Subiros	
Descripción: Mostrar al usuario los datos leídos desde el fichero de entrada	

Tabla 29: TI Visualización al usuario del contenido del fichero de entrada.

Tarea	
Número tarea: 2	**Número historia:** 2
Nombre tarea: Establecimiento del formato de los datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 19 febrero 2009	**Fecha fin:** 25 febrero 2009
Programador responsable: Dariel Subiros	
Descripción: El usuario debe identificar las variables de entrada según la información mostrada en la Tarea 1	

Tabla 30: TI Establecimiento del formato de los datos.

Tarea	
Número tarea: 3	**Número historia:** 2
Nombre tarea: Creación de fichero de configuración del Aermet.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 26 febrero 2009	**Fecha fin:** 4 marzo 2009
Programador responsable: Dariel Subiros	
Descripción: Se debe crear el fichero de configuración del Aermet para que este reconozca sus variables desde el fichero con la Data Meteorológica	

Tabla 31: TI Creación de fichero de configuración del Aermet.

	Tarea
Número tarea: 4	**Número historia:** 2
Nombre tarea: Ejecutar el Aermet.	
Tipo de tarea : Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 5 marzo 2009	**Fecha fin:** 7 marzo 2009
Programador responsable: Dariel Subiros	
Descripción: Se debe correr el Aermet de forma transparente al usuario pasándole los ficheros de entrada que este necesita	

Tabla 32: TI Ejecutar el Aermet.

Tareas de la Historia de Usuario #3:

	Tarea
Número tarea: 1	**Número historia:** 3
Nombre tarea: Creación de nueva Leyenda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9 marzo 2009	**Fecha fin:** 14 marzo 2009
Programador responsable: Dariel Subiros	
Descripción: Se debe permitir al usuario crear una nueva Leyenda	

Tabla 33: TI Creación de nueva Leyenda.

	Tarea
Número tarea: 2	**Número Historia:** 3
Nombre tarea: Modificación de Leyenda.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16 marzo 2009	**Fecha fin**: 21 marzo 2009
Programador responsable: Dariel Subiros	
Descripción: Se debe permitir al usuario modificar la Leyenda	

Tabla 34: TI Modificación de Leyenda.

	Tarea
Número tarea: 3	**Número historia:** 3
Nombre tarea: Visualización de Leyenda	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23 marzo 2009	**Fecha fin:** 28 marzo 2009
Programador responsable: Dariel Subiros	
Descripción: Se muestra la Leyenda en una tabla, donde el usuario puede crear una nueva leyenda(HU 3, Tarea 1) o modificar la existente(HU 3, Tarea 2)	

Tabla 35: TI Visualización de Leyenda.

Tareas de la Historia de Usuario #4:

	Tarea
Número tarea: 1	**Número historia:** 4
Nombre tarea: Extracción de datos para la graficación desde el fichero.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 30 marzo 2009	**Fecha fin:** 11 abril 2009
Programador responsable: Dariel Subirós	
Descripción: Esta tarea depende directamente de la Historia de Usuario 3, pues dada la Leyenda y las direcciones de viento seleccionadas serán extraídos los datos desde el fichero de entrada.	

Tabla 36: TI Extracción de datos para la graficación desde el fichero.

Tareas de la Historia de Usuario #5:

	Tarea
Número tarea: 1	**Número historia:** 5
Nombre tarea: Graficar Rosa de los Vientos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13 abril 2009	**Fecha fin:** 18 abril 2009
Programador responsable: Dariel Subiros	
Descripción: Se lee la Leyenda, la matriz de datos y se grafica.	

Tabla 37: TI Graficar Rosa de los Vientos.

Tarea	
Número tarea: 2	**Número historia:** 5
Nombre tarea: Mostrar en una tabla acumulados según direcciones y clases de viento (tabla de frecuencias).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20 abril 2009	**Fecha fin:** 25 abril 2009
Programador responsable: Dariel Subiros	
Descripción: Se muestra al usuario la información extraída según la Leyenda y Opciones de Graficación escogidas(Historias de Usuario 3 y 4 respectivamente)	

Tabla 38: TI Graficar acumulados según direcciones y clases de viento.

Tarea	
Número tarea: 3	**Número historia:** 5
Nombre tarea: Graficación de tabla de Contingencia Doble (Gráfico de barras).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 27 abril 2009	**Fecha fin:** 2 mayo 2009
Programador responsable: Dariel Subiros	
Descripción: Se lee la configuración de la Leyenda y de Opciones de Configuración (Historias de Usuario 3 y 4 respectivamente) y se crea un fichero en formato de imagen con el resultado, luego este se carga al programa y se muestra al usuario	

Tabla 39: TI Graficación de tabla de Contingencia Doble.

3.4 Conclusiones

Enfocándose en la programación orientada a objetos dentro de la fase de diseño de la metodología XP, se crearon las tarjetas CRC. Para lograr la completa implementación de cada historia de usuario en la fecha acordada con el cliente, estas se dividieron en tareas de ingeniería. A cada TI se le asignó un tiempo de desarrollo que se cumplió de manera eficiente garantizando así el objetivo principal de su confección.

CAPÍTULO 4: PRUEBA

Introducción

En el presente capítulo se muestran las pruebas de aceptación confeccionadas por el cliente para comprobar que la aplicación funcione de forma correcta. Estas pruebas fueron llevadas a cabo antes de cada entrega que se realizó durante todo el desarrollo del proyecto.

4.1 Pruebas de Aceptación (PA).

Uno de las mejores características de la metodología XP es el proceso de pruebas. Esta metodología propone probar tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos no deseados a la hora de realizar modificaciones y refactorizaciones. XP propone la realización de pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente. [**10**]

En este proyecto al codificar no se sigue la regla de XP que aconseja crear pruebas unitarias con entornos de desarrollo antes de programar. Las obtuvieron de la descripción de requisitos plasmados en las historias de usuarios, y estas especifican las barreras que deben pasar las distintas funcionalidades del programa, procurando codificar teniendo en cuenta las pruebas que se deben vencer. Para realizar las pruebas de aceptación el cliente utiliza la siguiente plantilla.

Prueba de aceptación
HU: Nombre de la historia de usuario que va a comprobar su funcionamiento.
Nombre: Nombre del caso de prueba.
Descripción: Descripción del propósito de la prueba.
Condiciones de ejecución: Precondiciones para que la prueba se pueda realizar.
Entrada/Pasos de ejecución: Pasos para probar la funcionalidad.
Resultado esperado: Resultado que se desea de la prueba.
Evaluación de la prueba: Aceptada o Denegada.

Tabla 40: Plantilla Prueba de Aceptación.

A continuación se presentan las pruebas llevadas a cabo para verificar buen el funcionamiento de cada módulo en las entregas que se le hacen al cliente cumpliendo con lo establecido en el cronograma de entregas.

4.1.1 Pruebas del módulo #1:Configuración de datos meteorológicos

Prueba de aceptación
HU: Identificación de Estación y fichero con data meteorológica.
Nombre: Validación de fichero de entrada.
Descripción: El especialista escoge el fichero con la data meteorológica.
Condiciones de ejecución:
Entrada/Pasos de ejecución: El especialista intenta escoger un fichero de entrada.
Resultado esperado: Se emite un mensaje de error en caso de fichero incorrecto o uno de información en caso de éxito al leer el fichero.
Evaluación de la prueba: Satisfactoria.

Tabla 41: PA Validación de fichero de entrada.

Prueba de aceptación
HU: Configuración de formato de datos meteorológicos.
Nombre: Visualización del contenido del fichero de entrada.
Descripción: Se muestran al usuario el o los records de muestra extraídos desde el fichero.
Condiciones de ejecución: El especialista debe haber escogido un fichero válido.
Entrada/Pasos de ejecución: El usuario pasa a la pestaña de "Records"
Resultado esperado: Se muestran al usuario el o los records de muestra leídos del fichero.
Evaluación de la prueba: Satisfactoria.

Tabla 42: PA Visualización del contenido del fichero de entrada.

Prueba de aceptación
HU: Configuración de formato de datos meteorológicos.
Nombre: Establecimiento del formato de los datos.
Descripción: El usuario crea el formato de los datos de entrada.
Condiciones de ejecución: El especialista debe haber escogido un fichero válido.
Entrada/Pasos de ejecución: El usuario: • Pulsa el botón "Editar". • Inserta las variables desde las tablas.
Resultado esperado: Se muestran en la tabla de "Formato Actual" el formato establecido por nombre de variables.
Evaluación de la prueba: Satisfactoria.

Tabla 43: PA Establecimiento del formato de los datos.

Prueba de aceptación
HU: Configuración de formato de datos meteorológicos.
Nombre: Creación de fichero de configuración del Aermet.
Descripción: Se crea el fichero con las variables clave según la configuración actual.
Condiciones de ejecución: Se debe haber establecido el formato de las variables leídas del fichero.
Entrada/Pasos de ejecución: • El usuario establece el formato de los datos. • El usuario intenta guardar el fichero de configuración.
Resultado esperado: Se emite un mensaje de error en caso de: • No se hallan configurado alguno de los records leídos • No se halla escogido ningún tipo de altura. • Escogidas "Alturas de Sitio especifico" y no se hallan configurado según cantidad de records de muestra leídos. • Se insertan variables multiniveles sin haber entrado alturas. • No coincida el número de variables leídas por record y las configuradas por el usuario • La fecha escogida para el análisis sea sea mayor que la fecha fin Se emite un mensaje de información en caso de éxito.
Evaluación de la prueba: Satisfactoria.

Tabla 44: PA Creación de fichero de configuración del AERMET.

4.1.2 Pruebas de aceptación al módulo #2:Graficación de datos meteorológicos

Prueba de aceptación
HU: Configuración de Leyenda.
Nombre: Creación de nueva Leyenda.
Descripción: El especialista puede crear una nueva leyenda.
Condiciones de ejecución:
Entrada/Pasos de ejecución: • Se muestra la ventana de configuración de la leyenda. • Se seleccionan inicio, ancho y repeticiones de clases de viento.
Resultado esperado: Se crea nueva leyenda.
Evaluación de la prueba: Satisfactoria.

Tabla 45: PA Creación de nueva Leyenda.

Prueba de aceptación
HU: Configuración de Leyenda.
Nombre: Modificación de leyenda.
Descripción: Se modifica la leyenda.
Condiciones de ejecución: • Que exista una leyenda(Esta siempre existe)
Entrada/Pasos de ejecución: • Se muestra la ventana de configuración de la leyenda. • Se seleccionan inicio, ancho y repeticiones de clases de viento o se cambia el color de clase de viento, se adiciona un rango al final o elimina el último.
Resultado esperado: Se modifica la leyenda.
Evaluación de la prueba: Satisfactoria.

Tabla 46: PA Modificación de leyenda.

Prueba de aceptación
HU: Configuración de Leyenda.
Nombre: Visualización de Leyenda.
Descripción: Se muestra al usuario la configuración de la leyenda.
Condiciones de ejecución:
Entrada/Pasos de ejecución: • Se muestra la ventana de leyenda.
Resultado esperado: Se visualiza la leyenda.
Evaluación de la prueba: Satisfactoria.

Tabla 47: PA Visualización de Leyenda.

Prueba de aceptación
HU: Configuración de Opciones de graficación.
Nombre: Extracción de datos para la graficación desde el fichero.
Descripción: Se extraen los datos desde el fichero según los parámetros que entre el usuario.
Condiciones de ejecución: • Leyenda. • Direcciones de viento.
Entrada/Pasos de ejecución: • Se deben configurar la leyenda y las direcciones de viento deseadas.
Resultado esperado: Se extraen exitosamente los datos desde el fichero
Evaluación de la prueba: Satisfactoria.

Tabla 48: PA Extracción de datos para la graficación desde el fichero.

Prueba de aceptación
HU: Graficación.
Nombre: Graficar Rosa de los Vientos.
Descripción: Utilizando clases de sistema(clases de Java), se leen las configuraciones y se grafica
Condiciones de ejecución: • Leyenda. • Direcciones de viento.
Entrada/Pasos de ejecución: • Se deben configurar la leyenda y las direcciones de viento deseadas.
Resultado esperado: Se crea un gráfico que representa la rosa de los vientos deseada.
Evaluación de la prueba: Satisfactoria.

Tabla 49: PA Graficar Rosa de los Vientos.

Prueba de aceptación
HU: Graficación.
Nombre: Mostrar Tabla de Frecuencias.
Descripción: Se muestra en una tabla la distribución de los acumulados de vientos según direcciones de viento y velocidades.
Condiciones de ejecución: • Leyenda. • Direcciones de viento.
Entrada/Pasos de ejecución: • Se debe mostrar la tabla de frecuencias.
Resultado esperado: Se grafican en una tabla los acumulados de viento.
Evaluación de la prueba: Satisfactoria.

Tabla 50: PA Mostrar Tabla de Frecuencias.

Prueba de aceptación
HU: Graficación.
Nombre: Mostrar Gráfico de Barras.
Descripción: Se muestra una tabla de contingencia doble (clases de viento y %)
Condiciones de ejecución: • Leyenda. • Direcciones de viento.
Entrada/Pasos de ejecución: • Se debe mostrar un gráfico de barras.
Resultado esperado: Se grafica muestra un gráfico de barras.
Evaluación de la prueba: Satisfactoria.

Tabla 51: PA Mostrar Tabla de Frecuencias.

4.2 Conclusiones

Con la realización de las pruebas de aceptación el cliente se asegura de que las funciones implementadas cumplan su objetivo satisfactoriamente, probando individualmente cada módulo y asignándole la evaluación correspondiente. Todas las pruebas que se realizaron fueron positivas y el cliente estuvo conforme, cumpliendo entonces la aplicación con las historias de usuarios definidas inicialmente.

CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

Introducción

Para estudiar la factibilidad de este proyecto se utilizará la **Metodología Costo Efectividad (Beneficio),** la cual plantea que la conveniencia de la ejecución de un proyecto se determina por la observación conjunta de dos factores:

- El costo, que involucra la implementación de la solución informática, adquisición y puesta en marcha del sistema hardware/software y los costos de operación asociados
- La efectividad, que se entiende como la capacidad del proyecto para satisfacer la necesidad, solucionar el problema o lograr el objetivo para el cual se ideó, es decir, un proyecto será más o menos efectivo con relación al mayor o menor cumplimiento que alcance en la finalidad para la cual fue ideado (costo por unidad de cumplimiento del objetivo).

5.1 Efectos Económicos

- ✓ Efectos directos
- ✓ Efectos indirectos
- ✓ Efectos externos
- ✓ Intangibles

Efectos directos:

- ✓ POSITIVOS:
- ➢ Se facilitará la configuración de los datos de entrada al preprocesador meteorológico AERMET.
- ➢ Se facilitará la interpretación de los resultados mediante la graficación.
- ➢ Se mejorará la eficiencia y explotación del preprocesador meteorológico AERMET.

- ✓ NEGATIVOS:
- Para usar la aplicación es imprescindible el uso de un ordenador, aparejado a los gastos que este trae de consumo de corriente eléctrica y mantenimiento.

Efecto indirecto:

- Los efectos económicos observados que pudiera repercutir sobre otros mercados no son perceptibles, aunque este proyecto no está construido con la finalidad de comercializarse.

Externalidades:

- Se contará con una herramienta disponible que permitirá a los especialistas en materia de medio ambiente modelar la dispersión de contaminantes en el aire.

Intangibles:

- En la valoración económica siempre hay elementos perceptibles por una comunidad como perjuicio o beneficio, pero al momento de ponderar en unidades monetarias esto resulta difícil o prácticamente imposible.

A fin de medir con precisión los efectos, deberán considerarse tres situaciones:

✓ **SITUACIÓN SIN PROYECTO**

El fichero de configuración "AERMET.INP" se le pasa al AERMET conteniendo las Palabras de Especificación (PE) necesarias (JOB, DATA), que son: Inicio de la configuración (incluye ficheros de reportes para el AERMET) y bloque de palabras clave de sitio específico. Para la entrada de los datos al AERMET sin este proyecto debemos crear fichero de texto plano, editarlo y seguir los siguientes pasos:

1. Escribir PE JOB.
    a) Escribir debajo e interlineados los nombres de ficheros de mensajes.
2. Escribir PE ONSITE, y debajo cada una de las siguientes PE del bloque ONSITE:
    a) DATA.
    b) QOUT.
    c) XDATES.
    d) LOCATION.
    e) READ.
    f) FORMAT.
    g) THRESHOLD.

Nota 1: Por cada READ debe haber un FORMAT. Habrán tantos READ y FORMAT como número de records por períodos existan.

Nota 2: Mientras más variables tenga cada record mayor trabajoso serán los READ y FORMAT a configurar.

- ✓ **SITUACIÓN CON PROYECTO**

Para la entrada de los datos al AERMET con el proyecto debemos seguir los siguientes pasos.

1. Selección del fichero de entrada.
2. Configurar XDATES, LOCATION y THRESHOLD.
3. Configurar contenido.
4. Leer del fichero.
5. Configurar formato.
6. Crear fichero de entrada del AERMET.

5.2 Beneficios Y Costos Intangibles en el proyecto

COSTOS:

- ✓ Resistencia al cambio.

BENEFICIOS:

- ✓ Mayor comodidad para los usuarios.
- ✓ Mejora en la calidad de la información.
- ✓ Menor tiempo empleado en la introducción de los datos.
- ✓ Facilidad a la hora de interpretar los datos de concentración.

5.3 Ficha de costo

Para determinar el costo económico del proyecto se utilizará el procedimiento para elaborar Una Ficha De Costo de un Producto Informático [Dra. Ana Ma. Gracia Pérez, UCLV].

Para la elaboración de la ficha se consideran los siguientes elementos de costo, desglosados en moneda libremente convertible y moneda nacional.

Costos en Moneda Libremente Convertible:

- ✓ **Costos Directos**.
    1. Compra de equipos de cómputo: No procede.
    2. Alquiler de equipos de cómputo: No procede.
    3. Compra de licencia de Software: No procede.
    4. Depreciación de equipos: $ 60.78.
    5. Materiales directos: No procede.

    Total: $ 60.78.
- ✓ **Costos Indirectos**.
    1. Formación del personal que elabora el proyecto: No procede.
    2. Gastos en llamadas telefónicas: No procede.
    3. Gastos para el mantenimiento del centro: No procede.
    4. Know How: No procede.
    5. Gastos en representación: No procede.

    Total: $0.00.
- ✓ **Gastos de distribución y venta.**
    1. Participación en ferias o exposiciones: No procede.
    2. Gastos en transportación: No procede.
    3. Compra de materiales de propagandas: No procede.

    Total: $0.00.

Costos en Moneda Nacional:

- ✓ **Costos Directos.**
    1. Salario del personal que laborará en el proyecto: $100.00.
    2. El 12% del total de gastos por salarios se dedica a la seguridad social: No procede.
    3. El 0.09% de salario total, por concepto de vacaciones a acumular: No procede.
    4. Gasto por consumo de energía eléctrica: $ 5.94.
    5. Gastos en llamadas telefónicas: No procede. 6.
    6. Gastos administrativos: No procede.

- ✓ **Costos Indirectos.**
    1. Know How: $ 108,75.

Total: $ 214.69.

Como se hizo referencia anteriormente, la técnica seleccionada para evaluar la factibilidad del proyecto es la Metodología Costo-Efectividad. Dentro de esta metodología la técnica de punto de equilibrio aplicable a proyectos donde los beneficios tangibles no son evidentes el análisis se basa exclusivamente en los costos. Para esta técnica es imprescindible definir una variable discreta que haga variar los costos. Teniendo en cuenta que el costo para este proyecto es despreciable, tomaremos como costo el tiempo en minutos empleado para introducir los datos de entrada al preprocesador de datos meteorológicos AERMET y la variable sería la cantidad de niveles en la torre de donde se tomaron las mediciones, para lo cual tenemos cuatro valores.

Valores de la variable (Solución manual):

- 1 Fichero de 3 niveles, 1 OBS/HOUR y 10 variables. (17 min.).
- 1 Fichero de 6 niveles, 1 OBS/HOUR y 10 variables. (34 min.).
- 1 Fichero de 9 niveles, 1 OBS/HOUR y 10 variables. (51 min.).
- 1 Fichero de 12 niveles, 1 OBS/HOUR y 10 variables. (68 min.).

Valores de la variable (Solución con el programa):

- 1 Fichero de 3 niveles, 1 OBS/HOUR y 10 variables. (0.83 min.).
- 1 Fichero de 6 niveles, 1 OBS/HOUR y 10 variables. (1.66 min.).
- 1 Fichero de 9 niveles, 1 OBS/HOUR y 10 variables. (2.5 min.).
- 1 Fichero de 12 niveles, 1 OBS/HOUR y 10 variables. (3.3 min.).

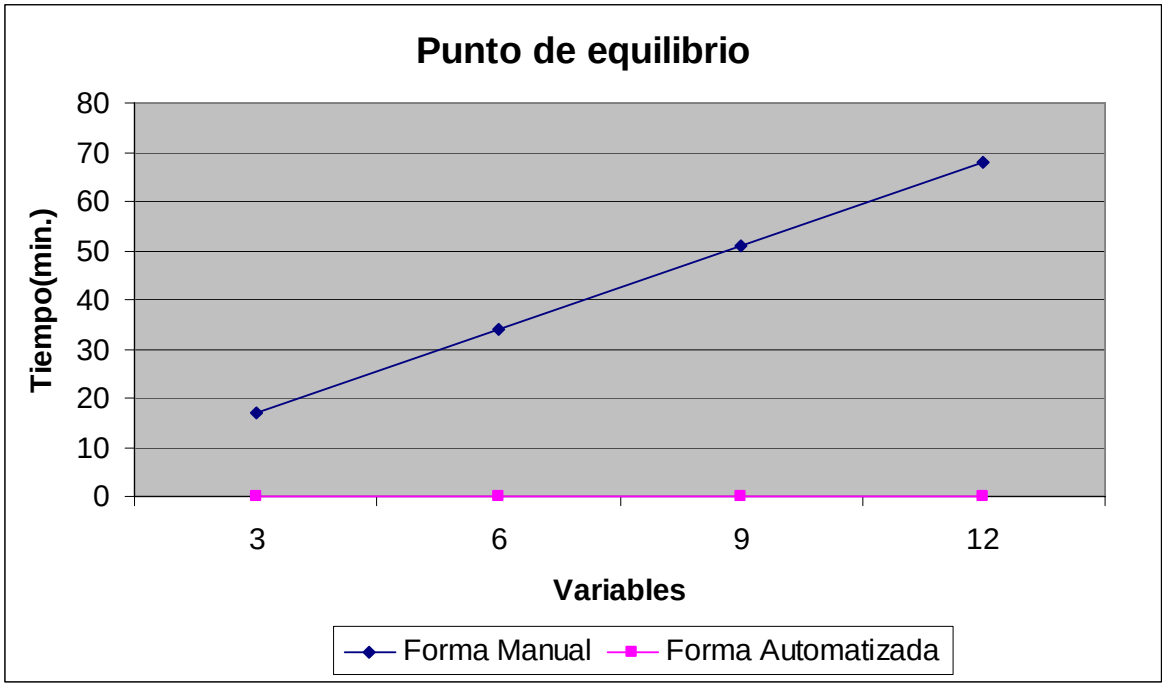

Ilustración 2: Gráfico de punto de equilibrio de soluciones.

Teniendo en cuenta los resultados reflejados en la gráfica en cuanto al Punto de Equilibrio queda demostrada la factibilidad del sistema evidenciado por la relación entre la complejidad del problema (cantidad de fuentes y receptores) y el tiempo que demora la introducción de los datos de forma manual y automatizada.

Configurar los datos de un fichero de 3 niveles, 1 OBS/HOUR y 10 variables me tomo un tiempo de aproximadamente 17 minutos.

5.4 Conclusiones

En este capítulo se realizó el estudio de factibilidad mediante La Metodología Costo Efectividad (Beneficio), se analizó los efectos económicos, los beneficios y costos intangibles, así como se calculó el costo de ejecución del proyecto mediante la ficha de costo arrojando como resultado $ 60.78 CUC. y $ 214.69 MN demostrándose la factibilidad del proyecto.

CONCLUSIONES GENERALES

El preprocesador de datos meteorológicos AERMET, no tiene interfaz alguna para la configuración de sus datos de entrada, lo que dificulta el trabajo con este programa y además, tampoco posee forma de representar gráficamente datos de salida generados durante su preprocesamiento.

Para dar solución a estos problemas se diseñó e implementó una aplicación que contiene una interfaz gráfica de usuario, para configurar los datos de entrada del AERMET y mostrar datos generados.

Con el objetivo de desarrollar de la aplicación se llevaron a cabo las siguientes actividades:

- Se hizo un análisis de los sistemas similares existentes de los cuales se obtienen ejemplos de cómo podrían solucionarse algunas funcionalidades que se requieren para la aplicación.
- Se llevó a cabo un estudio de las principales metodologías, lenguajes y herramientas que se consideraron factibles para el desarrollo del sistema.
- Se realizó todo el proceso de desarrollo del software siguiendo las fases de la metodología XP, lo cual queda plasmado en el presente documento.

Como resultado de la investigación se logró desarrollar un software en el que se da cumplimiento a las especificidades de los objetivos propuestos. Con el valor fundamental de simplificar la demora que produce el procesamiento manual de la información, disminuir el grado de errores, mejorar la interacción del usuario con el programa, y además para contribuir a elevar la calidad del desarrollo del trabajo.

RECOMENDACIONES

- Aplicar el software en el departamento del Centro de Estudios del Medio Ambiente (CEMA) para obtener un mayor rendimiento del mismo.
- Extender las funcionalidades acorde a nuevos requisitos que surjan en el departamento del CEMA.
- Implementar en futuras versiones el soporte de datos de entrada de superficie la graficación de clases de estabilidad de viento.
- Realizar un estudio más profundo de este sistema en vista a perfeccionarlo en nuevas versiones del software.

REFERENCIAS BIBLIOGRÁFICAS

[1] La contaminación como un problema global del Medio Ambiente , Sahilis Álvarez Pérez

[2] http://www.cubasolar.cu/biblioteca/Ecosolar/Ecosolar17/HTML/articulo06.htm [En línea] 15 junio 2009.

[3] TURTÓS CARBONELL, L. **;** et al**.** *Sistema de Modelos AERMOD para dispersión local de contaminantes atmosféricos.* 2007. http://www.cubaenergía.cu [En línea] 1ro Julio 2009.

[4] http://es.wikipedia.org/wiki/Fortran [En línea] 15 junio de 2009.

[5] http://es.wikipedia.org/wiki/Interfaz[En línea] 15 junio de 2009.

[6] http://es.wikipedia.org/wiki/Interfaz_gráfica_de_usuario[En línea] 15 junio de 2009.

[7] http://www.monografias.com/trabajos67/estimar-emisiones-altura-capa-limite/estimar-emisiones-altura-capa-limite2.shtml [En línea] 15 junio 2009.

[8] Turtós Carbonell, L et.al. Proyecto Programa Ramal Nuclear "Sistema de Modelos AERMOD para dispersión local de contaminantes atmosféricos" Código: PRN/5-2/2

[9] GNU Operating System. La Definición de Software Libre. [En línea] marzo de 2009. http://www.gnu.org/philosophy/free-sw.es.html

[10] Gutierrez, Jorge A. Saavedra. El Mundo Informático. Software Libre. [En línea] 2007. http://jorgesaavedra.wordpress.com/2007/05/05/lenguajes-de-programacion/

[11] J.J Gutierrez, M.J Escalona, M. Mejias, J.Torres. Pruebas del sistema en programación extrema. [En línea]

BIBLIOGRAFÍA

Barbone, Víctor A. González. XP: Extreme Programming . [En línea] http://iie.fing.edu.uy/~nacho/blandos/seminario/XProg1.html .

Lsi.us.es. PSISEXTREMA.pdf. [En línea] 15 junio 2009 http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf

Monografías.com. Datos meteorológicos. [En línea] 2009. http://www.monografias.com/trabajos67/estimar-emisiones-altura-capa-limite/estimar-emisiones-altura-capa-limite2.shtml

Patricio Letelier, Ma. Carmen Penadés. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2009 http://www.willydev.net/descargas/masyxp.pdf .

Torres, Raúl. Lenguajes de Programación. [En línea] 2009. http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml .

Wikipedia.Interfaz.[En línea]. 12 junio 2009 http://es.wikipedia.org/wiki/Interfaz

Sanchez, María A. Mendoza. Informatizate. Metodologías De Desarrollo De Software. [En línea] 2009. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Wikipedia.*FORTRAN.*[En Línea] 1 julio 2009. http://es.wikipedia.org/wiki/FORTRAN

Wikipedia.Interfaz gráfica de usuario.[En línea]. 12 junio 2009 http://es.wikipedia.org/wiki/Interfaz_gráfica_de_usuario

GLOSARIO DE TÉRMINOS

AERMOD: El código AERMOD (**A**merican Meteorology Society – **E**.P.A. **– R**egulatory **MOD**el)

GPL: ***G****eneral* ***P****ublic* ***L****icense* / Licencia Pública General. Orientada principalmente a proteger la libre distribución, modificación y uso de software.

Patrón: Es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

GUI: ***G****raphical* ***U****ser* ***I****nterface* / Interfaz Gráfica de Usuario. Programa software que gestiona la interacción con el usuario de manera gráfica mediante el uso de íconos, menú, mouse, etc.

IDE: ***I****ntegrated* ***D****evelopment* ***E****nvironment* / Entorno de Desarrollo Integrado. Entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

Linux: Sistema Operativo libre creado por el finlandés Linus Torvalds.

Metodología Ágil: Constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Programación Extrema(XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo**.**

RUP: El Proceso Unificado Racional o RUP (Rational Unified Process), es un proceso desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos y roles.

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Software Libre: es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.

ANEXO 1

Variables y valores por defecto del análisis de calidad para las variables de nivel simple y de fecha/tiempo aceptadas por AERMET

Variable name	Description	Units	Type	Missing Indicator	Lower Bound	Upper Bound
HPLX	Surface heat flux	watts/square meter	<	999	-100	800
USTR	Surface friction velocity	meters/second	<	999	0	2
MHGT	Mixing height	meters	<	9999	0	4000
ZOHT	Surface roughness length	meters	<	999	0	2
SAMT	Snow amount	centimeters	<=	999	0	250
PAMT	Precipitation amount	centimeters	<=	999	0	100
INSO	Insolation	watts/square meter	<	9999	0	1250
NRAD	Net radiation	watts/square meter	<	999	-100	800
DT01	Temperature diff.(U - L)[1]	°C	<	9	-2	5
DT02	Temperature diff.(U - L)[1]	°C	<	9	-2	5
DT03	Temperature diff.(U - L)[1]	°C	<	9	-2	5
US01	User's scalar #1	user's units	<	999	0	100
US02	User's scalar #2	user's units	<	999	0	100
US03	User's scalar #3	user's units	<	999	0	100
ALTP	Altimeter pressure	inches mercury*100	<=	-9999	2700	3200
SLVP*	Sea level pressure	millibars *10	<	-9999	9000	10999
PRES*	Station pressure	millibars *10	<	-9999	9000	10999
CLHT*	Ceiling height	kilometers *10	<=	-9999	0	300
TSKC*	Sky cover (total/opaque)	tenths	<=	99	0	10
OSDY	Day		<=	-9	1	31
OSMO	Month		<=	-9	1	12
OSYR	Year		<=	-9	0	99
OSHR	Hour		<=	-9	0	24
OSMN	Minutes		<=	-9	0	60

ANEXO 2

Variables y valores por defecto del análisis de calidad para las variables de nivel múltiple aceptadas por AERMET

Variable name	Description	Units	Type	Missing Indicator	Lower Bound	Upper Bound
HTnn	Height	meters	<	9999	0	4000
SAnn	Std. dev. horizontal wind	degrees	<	99	0	35
SEnn	Std. dev. vertical wind	degrees	<	99	0	25
SVnn	Std. dev. v-comp. of wind	meters/second	<	99	0	3
SWnn	Std. dev. w-comp. of wind	meters/second	<	99	0	3
SUnn	Std. dev. u-comp. of wind	meters/second	<	99	0	3
TTnn*	Temperature	°C	<	99	-30	35
WDnn*	Wind direction	degrees from north	<=	999	0	360
WSnn*	Wind speed	meters/second	<	999	0	50
VVnn	Vertical wind component	meters/second	<	999	0	5
DPnn	Dew-point temperature	°C	<	99	-65	35
RHnn	Relative humidity	whole percent	<=	999	0	100
V1nn	User's vector #1	user's units	<	999	0	100
V2nn	User's vector #2	user's units	<	999	0	100
V3nn	User's vector #3	user's units	<	999	0	100