Instituto Tecnológico de San Juan del Río

Centro de Investigación en ciencia Aplicada y Tecnología Avanzada

Amplificación interactiva de contenido por medio de la detección de la dirección de la mirada.

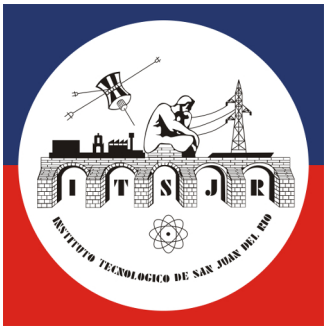
Autor:

J. Carlos Ávila Resendiz

Supervisor:

Dr. Joaquin Salas Rodriguez

4 de junio de 2016


I T S J R
INSTITUTO TECNOLOGICO DE SAN JUAN DEL RIO

Índice general

Introducción **1**

1. GENERALIDADES **5**

1.1. Objetivos 6

1.1.1. Objetivo general 6

1.1.2. Objetivos específicos 6

1.2. Justificación 7

1.3. Caracterización de la empresa 9

1.3.1. Datos generales de la empresa 9

Misión 9

Visión 10

Valores 10

Objetivo 10

Estructura Organizativa 11

1.3.2. Descripción del departamento o área de trabajo 11

1.4. Problemas a resolver 13

1.5. Alcances y limitaciones 14

1.5.1. Alcances 14

1.5.2. Limitaciones 14

2. FUNDAMENTACIÓN TEÓRICA **15**

2.1. Ingeniería del software 16

2.1.1. Concepto de Software 16
2.1.2. Clasificación del software 16
2.1.3. Concepto de la Ingeniería de Software 18
2.2. Herramientas de desarrollo 19
2.2.1. Visual Studio Community 2015 19
2.2.2. Qt Creator 19
2.2.3. PyCharm 20
2.2.4. Sublime-Text 21
2.2.5. Rstudio 22
2.2.6. CMake 23
2.2.7. Sistema de control de versiones Git 24
2.2.8. MariaDB 25
2.2.9. SQLite 25
2.2.10. OpenCV 26
2.2.11. IntraFace 27
2.3. Lenguajes de programación 28
2.3.1. C++ 28
Historia 29
Características 29
Conceptos generales de POO 30
2.3.2. Python 30
Instalacion de Python 31
Herramientas básicas 32
2.3.3. R 32
2.4. Metodologías de desarrollo de software 34
2.4.1. Metodología de desarrollo extremo 34
El proceso de desarrollo extremo 34
Interacción con el cliente 35
Planificación del proyecto 36

Diseño, desarrollo y pruebas 37
Resumen de XP 39

3. DESCRIPCIÓN DE ACTIVIDADES REALIZADAS 41

3.1. Análisis 42
3.2. Diseño 42
3.3. Desarrollo 42
3.4. Pruebas 42
3.5. Implementación 42
3.6. Retroalimentación 42
3.7. Resultados 42
3.8. Conclusiones y recomendaciones 42
3.9. Referencias Bibliográficas & Glosario 42

Introducción

La vista es uno de las principales capacidades sensoriales de los humanos que mas ocupamos en el día a día, nos sirve para percibir el color del mundo, orientarnos en el entorno, ver los peligros, etc.
Técnicamente se llama visión a la capacidad de interpretar nuestro entorno gracias a los rayos de luz que alcanzan el ojo. También se entiende por visión toda acción de ver.
La acción simple de *ver*, es en términos simples una de las mas complejas tareas que como humanos realizamos, en todo momento, la cantidad de factores que influyen en la correcta percepción de los estímulos visuales y su posterior procesado e interpretación, son tareas que abordadas desde la perspectiva de un dispositivo electrónico, son extremadamente complicadas.

No obstante las limitaciones que la tecnología actual presenta al momento de abordar dicha tarea, hay en este campo de la *visión artificial* un iteres constante en buscar y encontrar soluciones, al ya viejo problema de hacer que las máquinas puedan ver el mundo tal como nosotros lo hacemos.
Debe decirse que se han realizado enormes avances en el área desde que la idea fue propuesta, pero remontemos la memoria a mucho antes, quizá hasta los tiempos de grandes filósofos y matemáticos griegos, por ejemplo *Thales de Mileto (640-585 A.C)*, fue capaz de predecir un eclipse y de medir una pirámide gracias a los conocimientos de geometría que poseía, *Euclides* concebía la trigonometría como un conjunto de lineas y puntos, independientemente del sistema de coordenadas.
Posteriormente vino el tiempo renacentista y con el grandes pintores, como *Filippo*

Bruneleschi(1377-1446) que invento la perspectiva, algo que dio mucho sobre lo que estudiar en los años que siguieron, consiguiendo crear a partir de ella todo un mundo de teorías y métodos al respecto.
Todo esto termino dando como resultado la construcción de maquinas y/o dispositivos que ayudaran al artista a capturar una imagen, todas estas máquinas y técnicas sobre la perspectiva dieron como resultado la invención final, la *cámara fotográfica*.
En tiempos mas actuales fue la NASA en 1964 la que revoluciono el campo, logrando procesar por medios digitales las imágenes del satélite MARINER, el avance desde entonces no ha parado en ningún momento.

El presente proyecto es uno más de los trabajos que se realizan ahora mismo en el campo de visión por computadora o visión artificial, como se expone en los primeros párrafos la vista es uno de los sentidos que mas extensamente usamos en el día a día, más aun en un mundo lleno de dispositivos que nos permiten visualizar información por medio de pantallas, específicamente, *computadoras*, sin embargo no todas las personas pueden hacerlo de manera optima o en condiciones que permitan obtener una buena *visión* de las mismas, este segmento de personas son las que presentan alguna debilidad visual.
Por lo tanto, dicho segmento de personas tienen dificultades para sumergirse de manera optima en el medio digital, por ello este proyecto busca una solución que permita un nivel de accesibilidad mucho mayor, ademas se busca de dotar de del elemento interactivo a los propios ojos, pues sera por medio de la detección de la dirección de la mirada que se realice todo el proceso.
El proyecto en si es ambicioso de primeas, pues actualmente pese a que la tecnología de asistencia para magnificar áreas de la pantalla ya existen, no hay una opción que permita a los usuarios hacerlo de forma interactiva, usando para ello solo la vista.

Se abordara la el problema usando las herramientas de la ingeniería de software, las técnicas y herramientas se presentaran en los diferentes capítulos del presente documento, así como los resultados de las investigaciones y procesos para llevar a cavo la tarea

J.C.A.R

expuesta.

J.C.A.R

Capítulo 1

GENERALIDADES

"El auténtico genio consiste en la capacidad para evaluar información incierta, aleatoria y contradictoria."
Winston Churchill, estadista.

1.1. Objetivos

1.1.1. Objetivo general

Desarrollar una aplicación de amplificación interactiva para computadoras con sistema operativo Windows, que asista a personas con bajas capacidades visuales, por medio del seguimiento y estimación de la dirección de la mirada sobre la pantalla de la computadora y en base a ello amplificar la zona de la pantalla en la que enfoca la vista.

1.1.2. Objetivos específicos

- Detección precisa y confiable del movimiento del globo ocular, con la ayuda de software de procesamiento de imágenes digitales.
- Hacer uso de las API's del sistema operativo que proveen las herramientas que magnifican la zona de la pantalla seleccionada.
- Integrar los dos componentes anteriores y de esa forma obtener un magnificador con interacción visual.
- Una vez se cuente con un prototipo, realizar pruebas de campo.

1.2. Justificación

Pese al avance desmesurado de la tecnología en los últimos años en donde las capacidades de los dispositivos se duplica cada cierto tiempo, respondiendo de forma bastante precisa la emblemática ley de Moore hay aun a día de hoy ciertas cuestiones que no han sido abordadas, quizá en gran parte debido al amplio panorama de problemas que se pueden afrontar con soluciones tecnológicas y de alguna forma ayudar a solventar o/y hacer más fácil las mismas.

Aun si los programas de asistencia a personas con capacidades diferentes están a día de hoy cobrando mayor relevancia en prácticamente todos los aspectos sociales, pues en la actualidad las posibilidades de llevar una vida productiva y sin las limitaciones de antaño, son ya una realidad, entre las herramientas que se proporcionan a este sector de la población están las llamadas tecnologías de asistencia o accesibilidad en entornos informáticos, mismas que van desde iconos monocromáticos de un mayor tamaño, hasta lectores de pantalla y lupas, siendo estas últimas el principal componente proporcionado por las herramientas de accesibilidad de los Sistemas Operativos SO actuales, siendo común en los tres mas importantes Linux, Windows, Mac.

Siendo de los tres el segundo, Windows, en el cual se enfocaran los esfuerzos de hacer converger las herramientas de accesibilidad ya mencionadas y las tecnologías de visión por computadora CV, para ofrecer a los discapacitados visuales una forma de hacer uso de la tecnología, mismos que según datos de la OMS de 2002, eran mas de 161 millones de personas, en especifico de computadoras, sin que su limitante visual les impida el poder interactuar con el equipo.

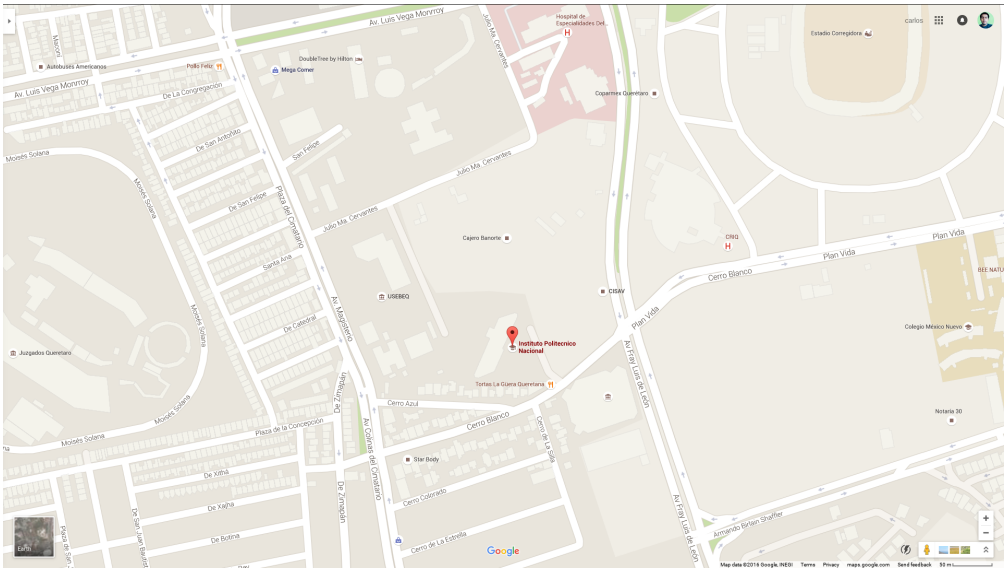
Específicamente el segmento de la población con discapacidad visual en el que se enfoca el desarrollo de este proyecto es el de personas que cuentan con cierto grado de visión, o lo que se conoce como resto visual, pues, siempre que exista un resto visual por mínimo que sea se debe potenciar su uso para alcanzar el máximo desarrollo posible

1.3. Caracterización de la empresa

1.3.1. Datos generales de la empresa

Nombre de la organización: Centro de Investigación en Ciencia Aplicada y Tecnología Avanzada del Instituto Politécnico Nacional `CICATA`

Dirección: Querétaro, Cerro Blanco No.141 Col. Colinas del Cimatario, C.P. 76090, Querétaro, Querétaro México.



Teléfonos: 1 (442) 2290804 o 01 (55) 5729 6000 Ext. 81002

E-Mail: cicata@ipn.mx

Fax: 5395 4147

Misión

Somos un centro de investigación creado por el IPN para fortalecer su impacto a nivel nacional, que atiende necesidades de formación de recursos humanos y de desarrollo tecnológico de la región, a través de proyectos de investigación que contribuyen al

desarrollo social y a la competitividad de los sectores productivo y de servicios, con el respaldo de las capacidades del Instituto, con un enfoque multidisciplinario, innovador y de excelencia, en un marco de sustentabilidad.

Visión

En el 2025, el CICATA-Querétaro se ve como un centro de vanguardia en la investigación y formación de recursos humanos; referente a nivel latinoamericano; con reconocimiento internacional por sus contribuciones de alto impacto y como una de las primeras opciones para alumnos e investigadores, por ser un centro innovador, competitivo, líder y emprendedor.

Valores

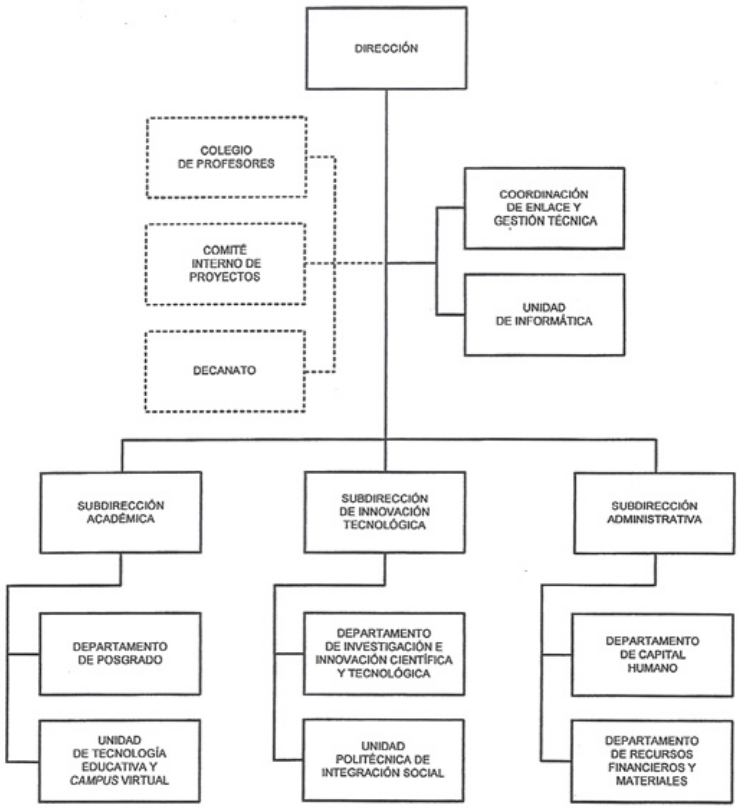
Hemos identificado un conjunto de valores que nos representan y que permiten cumplir nuestra misión y lograr la visión forjada:

- Calidad
- Integridad
- Compromiso
- Asertividad
- Trabajo en equipo
- Aprendizaje continuo

Objetivo

Servir de enlace entre la comunidad científica y los sectores productivos de bienes y servicios, atenderlos y ofrecerles soluciones a sus problemas de desarrollo. Para el cumplimiento de este objetivo, CICATA Querétaro desarrolla programas de investigación científica, tecnológica e innovación con un enfoque interdisciplinario, y asimismo atiende la formación de capital humano de alto nivel, contribuyendo decisivamente al fortalecimiento de la calidad y la competitividad del aparato productivo mexicano.

Estructura Organizativa



1.3.2. Descripción del departamento o área de trabajo

El área de análisis de imágenes puede ser definida como la construcción de algoritmos para la extracción de información presente en las imágenes. Es un área donde una gran variedad de conceptos fundamentales necesitan ser desarrollado e importantes aplicaciones pueden crearse. Esta combinación de teoría y práctica es particularmente atractiva para CICATA Querétaro en virtud de corresponder con su objetivo operativo. El grupo de análisis de imágenes ha presentado desde sus orígenes resultados muy buenos en el renglón de vinculación y desarrollo tecnológico. Se han trabajado proyectos de vinculación con empresas e instituciones tales como TAMSA y el IFE, por mencionar algunos. En la actualidad con un grupo de cuatro investigadores esta tendencia

se ha mantenido. En este momento se estudian temas de interferometría, colorimetría, industrial, metrología, óptica, análisis de imágenes, procesamiento de imágenes, reconstrucción tridimensional e interpretación visual de la actividad.

1.4. Problemas a resolver

- Amplificar áreas de la pantalla donde se enfoque la vista usando la vista.
- Detectar y seguir la dirección de la mirada.
-

1.5. Alcances y limitaciones

1.5.1. Alcances

El sistema de seguimiento e identificación de la dirección de la mirada persigue el objetivo de permitir la interacción usuario-maquina usando en el proceso únicamente los ojos y, proporcionar interactividad a esta parte del cuerpo.

El uso directo e inmediato de la detección de la dirección de la mirada es el de proporcionar una forma interactiva para magnificar zonas de la pantalla donde se enfoque la mirada en determinado momento.

Inicialmente el sistema será desarrollado para funcionar en maquinas con sistema operativo Windows.

1.5.2. Limitaciones

Ciertamente el proyecto presenta algunas dificultades técnicas y algunas otras funcionales, entre las cuales encontramos las siguiente:

Las condiciones no controladas, se espera unas condiciones de operación dentro de ciertos parámetros, es por tanto importante considerar el rendimiento en entornos donde las mismas se encuentres fuera de norma, provocando un funcionamiento incierto.

La API de magnificación de Windows esta solo disponible en para arquitecturas de 32 bits, lo que imposibilita el desarrollo del sistema para arquitecturas de 64 bits.

J.C.A.R

Capítulo 2

FUNDAMENTACIÓN TEÓRICA

2.1. Ingeniería del software

2.1.1. Concepto de Software

Existen varias definiciones similares aceptadas para software, pero probablemente la más formal sea la siguiente:

"Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación" (IEEE Std, 1993).

Considerando esta definición, el concepto de software va más allá de los programas de computación en sus distintos estados: código fuente, binario o ejecutable; también su documentación, los datos a procesar e incluso la información de usuario forman parte del software: es decir, abarca todo lo intangible, todo lo «no físico» relacionado.

El término software fue usado por primera vez en este sentido por John W. Tukey en 1957. En la ingeniería de software y las ciencias de la computación, el software es toda la información procesada por los sistemas informáticos: programas y datos.

2.1.2. Clasificación del software

Si bien la definición anterior de software es, en cierto modo arbitraria, y a veces confusa, a los fines prácticos se puede clasificar al software en tres grandes tipos [1]:

- **Software de sistemas:** Su objetivo es desvincular adecuadamente al usuario y al programador de los detalles informáticos en particular que se use, aislándolo especialmente del procesamiento referido a las características internas de: memoria, discos, puertos y dispositivos de comunicaciones, impresoras, pantallas y teclados. El software de sistema le procura al usuario y programador adecuadas

interfaces de alto nivel, controlador, herramientas y utilidades que permiten el mantenimiento del sistema global.

Incluye entre otros:

- Sistemas Operativos
- Controladores de dispositivos
- Herramientas de diagnostico
- Servidor de utilidades
- Herramientas de corrección y optimización.

- **Software de programación:** Es el conjunto de herramientas que permiten al programador desarrollar programas informáticos, usando diferentes alternativas y lenguajes de programación, de una manera práctica. Incluyen básicamente:

  - Editores de texto
  - Compiladores
  - Interpretes
  - Enlazadores
  - Depuradores

  - Entornos de desarrollo integrado (IDE). Agrupan las anteriores herramientas, usualmente en un entorno visual, de forma que el programador no necesite introducir múltiples instrucciones para compilar , interpretar y depurar. Cuentan con una interfaz gráfica de usuario GUI.

- **Software de aplicacion:** es aquel que permite a los usuarios llevar a cabo una o varias tareas específicas, en cualquier campo de actividad susceptible de ser automatizadas o asistido, con especial énfasis en los negocios. Incluye entre muchos otros:

  - Control de sistemas.
  - Automatización industrial
  - Software educativo
  - Software empresarial
  - Base de datos
  - Telecomunicaciones
  - Videojuegos
  - Software medico

- Aplicaciones ofimáticas
- Software de calculo numérico y simbólico
- Diseño asistido (CAD)

2.1.3. Concepto de la Ingeniería de Software

La ingeniería de software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza. En esta definición existen dos frases clave:

Disciplina de la ingeniería: Los ingenieros hacen que las cosas funcionen. Aplican teorías, métodos y herramientas donde sean convenientes, pero las utiliza de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías o métodos aplicables para resolverlos.

Todos los aspectos de producción de software: La ingeniería de software no solo comprende los procesos técnicos del desarrollo de software, sino también actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

en general los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma mas efectiva de producir software de calidad [1] pp.6.

2.2. Herramientas de desarrollo

Como se describe en la sección 2.1.2 hay diferentes clases y tipos de software, entre los que se encuentran las herramientas de desarrollo o software de desarrollo, así como software de aplicación, en este proyecto se han usado variedad de ellas, de las cuales se da una descripción a continuación:

Software de desarrollo

2.2.1. Visual Studio Community 2015

VISUAL STUDIO COMMUNITY 2015 es un entorno de desarrollo integrado creado y distribuido por MICROSOFT, para el sistema operativo de la misma. Soporta una gran variedad de lenguajes de programación tales como, C++, C#, VISUAL BASIC, .NET, F#, JAVA, PYTHON, RUBY, PHP, al igual que entornos de desarrollo web como ASP.NET MVC, DJANGO, ETC., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco. Esta version en particular tiene la particularidad de ser gratuita e incluir todas las características de la versión EXPRESS, enfocada en desarrolladores individuales, proyectos de código abierto, investigación académica, educación y pequeños equipos de profesionales.

2.2.2. Qt Creator

Qt Creator es un IDE (entorno de desarrollo integrado) multiplataforma que se ajusta a las necesidades de los desarrolladores Qt. Este es parte de Qt Project [1].

Qt Creator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt.

[1] *Qt Project*

- Editor de código con soporte para C+, QML y ECMAscript
- Herramientas para la rapida navegacion del codigo
- Resaltado de sintaxis y auto-completado de código
- Control estático de código y estilo a medida que se escribe
- Soporte para refactoring de código
- Ayuda sensitiva al contexto
- Plegado de código (code folding)
- Paréntesis coincidentes y modos de selección

2.2.3. PyCharm

PYCHARM es un entorno integrado de desarrollo (IDE) usado para la programacion en PYTHON (2.3.2). Provee herramientas de análisis de código, depurador gráfico, pruebas unitarias, integración con sistemas de control de versiones y soporte para el desarrollo con DJANGO.
Desarrollado por la compañía JetBrains [2].

Es multi-plataforma, funcionando en Windows, Mac OS X y Linux, cuenta con una version profesional liberada bajo licencia propietaria y con una versión comunitaria bajo los términos de Apache Licence [3], aunque esta última cuenta con algunas limitaciones.

Entre sus características mas destables se encuentran las siguientes:

- Asistencia en el análisis de código
- Auto-completado de código
- Errores de sintaxis
- Resaltado de lineas
- Vistas de navegación
- Vistas de estructuras
- Integración con Python Debugguer.
- Integración con CVS.

[2] JetBrains: https://en.wikipedia.org/wiki/JetBrains
[3] Apache: https://en.wikipedia.org/wiki/Apache_License

2.2.4. Sublime-Text

Sublime Text es un editor de texto y editor de código fuente está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de `Vim`, con el tiempo fue creando una identidad propia, por esto aún conserva un modo de edición tipo vi llamado Vintage mode.

Características:

- Minimapa. consiste en una previsualización de la estructura del código
- Multi selección: Hace una selección múltiple de un término por diferentes partes del archivo.
- Multi cursor: Crea cursores con los que podemos escribir texto de forma arbitraria en diferentes posiciones del archivo.
- Multi layout: Trae siete configuraciones de plantilla podemos elegir editar en una sola ventana o hacer una división
- Soporte nativo para infinidad de lenguajes: Soporta de forma nativa 43 lenguajes de programación y texto plano.
- Remarcado de sintaxis: El remarcado de sintaxis es completamente configurable a través de archivos de configuración del usuario.
- Búsqueda dinámica: Se puede hacer búsqueda de expresiones regulares o por archivos, proyectos, directorios, una conjunción de ellos o todo a la vez.
- Keybindings: Todas las teclas pueden ser sobrescritas a nuestro gusto.
- Etc.

Se puede descargar desde el sitio web Sublime-Text [4] y evaluar de forma gratuita. Sin embargo no es software libre o de código abiertoy se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene

[4] Subl-Text: https://www.sublimetext.com/

fecha de caducidad.

2.2.5. Rstudio

RStudio es un entorno de desarrollo integrado (IDE) para R (2.3.3). Esta disponible de forma libres y en edición comercial, se ejecuta en entornos multiplataforma (Windows, Mac OS X y Linux) o sobre navegadores web conectados a RStudio Server o RStudio Server Pro (Debian/Ubuntu, RedHat/CentOS y SUSE Linux).

RStudio tiene la misión de proporcionar el entorno informático estadístico R. Permite un análisis y desarrollo para que cualquiera pueda analizar los datos con R.

Entre sus características se encuentran la siguientes:

- Consola integrada
- Resaltado de sintaxis
- Ejecución directa de código R.
- Herramientas para gráficos
- Historial
- Depurador interactivo
- Fácil administración de espacios de trabajo.
- Ayuda y documentación integrada
- Edición con Sweave y R Markdown

Esta disponible desde la web oficial del proyecto RStudio [5] en sus dos versiones.

2.2.6. CMake

CMake es una herramienta multiplataforma de generación o automatización de código. El nombre es una abreviatura para *'cross platform make' (make multiplataforma)*; más allá del uso de *'make'* en el nombre, CMake es una suite separada y de más alto nivel que el sistema `make` común de Unix, siendo similar a las autotools.

[5] RStudio: https://www.rstudio.com/

CMake es una familia de herramientas diseñada para construir, probar y empaquetar software. CMake se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma.

Desarrollado por *Andy Cedilnik, Bill Hoffman, Brad King, Ken Martin, Alexander Neundorf* se puede acceder a la ultima version estable en CMake [6].

Entre las principales funcionalidad se encuentran las siguientes:

- Ficheros de configuración escritos en un lenguaje de scripting específico para CMake
- Análisis automático de dependencias para C, C++, Fortran, y Java
- Soporte para SWIG, Qt, FLTK, a través del lenguaje de scripting de CMake
- Soporte para varias versiones de Microsoft Visual Studio, incluyendo la 6, 7, 7.1, 8.0, 9.0 y 10.0
- Detección de cambios en ficheros usando timestamps tradicionales
- Soporte para builds paralelos
- Compilador cruzado
- Soporte para builds multiplataforma
- Integrado con DART (SOFTWARE), CDASH, CTEST Y CPACK, una colección de herramientas para prueba y liberación de software

Software de aplicación

2.2.7. Sistema de control de versiones Git

Git es un software de control de versiones diseñado por *Linus Torvalds*, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario

[6] CMake: www.cmake.org

o front end como Cogito o StGIT.

Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que usan Git, en particular, el grupo de programación del núcleo Linux.

Entre las características más relevantes se encuentran:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones.
- Gestión distribuida, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos.

La documentación completa, tutoriales de uso y medios de instalación se pueden encontrar en la pagina oficial de Git [7]

2.2.8. MariaDB

MariaDB es un sistema de gestión de bases de datos derivado de MySQL con licencia GPL. Está desarrollado por *Michael (Monty) Widenius (fundador de MySQL)* y la comunidad de desarrolladores de software libre.

Introduce dos motores de almacenamiento nuevos, uno llamado Aria -que reemplaza con ventajas a MyISAM- y otro lamado XtraDB -en sustitución de InnoDB. Tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente. Este SGBD surge a raíz de la compra de *Sun Microsystems*.

[7] Git: https://git-scm.com/

MariaDB es un fork directo de MySQL que asegura, permanecerá una versión de este producto con licencia GPL. La ultima versión estable se puede encontrar en la web oficial del proyecto MariaDB [8].

2.2.9. SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por *D. Richard Hipp*.

A diferencia de los sistema de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo.

El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina cliente. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

En su versión 3, SQLite permite bases de datos de hasta 2 Terabytes de tamaño, y también permite la inclusión de campos tipo BLOB.

Librerías

2.2.10. OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de

[8] MariaDB: https://mariadb.org/

movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel (IPP).

2.2.11. IntraFace

IntraFace es una librería que incluye algoritmos para análisis de patrones faciales, se comenzó a desarrollar en 2010 en la universidad Carnigie Mellon & University of Pittsburgh, actualmente la ultima version de la misma es la 1.0, soportada por Human Sensing Laboratory.
Para este proyecto se usa una version anterior de la misma, provista para fines educativos y/o investigación sin fines de lucro.

2.3. Lenguajes de programación

Un lenguaje de programación es una herramienta que nos permite comunicarnos e instruir a la computadora para que realice una tarea especifica. Cada lenguaje de programación posee una sintaxis y un léxico particular, es decir, la forma de escribirse, que es diferente en cada uno por la forma en que fue creado y por la forma que trabaja su compilador para revisar, acomodar y reservar el programa en memoria.

Existen muchos lenguajes, sin embargo en este proyecto solo nos interesan los siguientes:

- C++
- Python
- R

2.3.1. C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por *Bjarne Stroustrup*. La intención de su creación fue el extender al lenguaje de programación C, agregando mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido, por un lado es un lenguaje que sigue muy ligado a hardware subyacente, manteniendo una considerable potencia para la programación a bajo nivel, pero se le han añadido elementos que le permiten un estilo de programación de alto nivel de abstracción.

La definición oficial nos dice que C++ es un lenguaje de propósito general basado en C, al que se le han añadido nuevos tipos de datos, plantillas, clases, mecanismos de acepciones, espacio de nombres (STD), funciones inline, sobrecarga de operadores, referencias, manejo de memoria persistente y algunas utilidades adicionales en la librería estándar de C++.

Historia

El comité para el estándar ANSI C fue formado en 1983 con el objetivo de crear un lenguaje uniforme a partir del C original, desarrollado por *Kernighan y Ritchie* en 1972, en la AT&T. Hasta entonces el estándar lo marcaba el libro escrito en 1978 por estos dos autores. El lenguaje C++ se comenzó a desarrollar en 1980. Su autor fue Bjarne Stroustrup, también de la AT&T. Al comienzo era una extensión del lenguaje C que fue denominada *C with classes*. Este nuevo lenguaje comenzó a ser utilizado fuera de la AT&T en 1983. El nombre C++ es también de ese año, y hace referencia al carácter del operador incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la AT&T comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI (seguido algún tiempo después por un comité ISO) para estandarizarlo a nivel americano e internacional.

Características

Antes, mencionar que tanto C como C++ son lenguajes compilados, y no interpretados. Esta diferencia es muy importante, ya que afecta mucho a muchos aspectos relacionados con la ejecución del programa.

Como lenguaje de programación orientado a objetos se basa en una filosofía completamente diferente a la se su predecesor, exige el cambio completo de paradigma al programador, las propias características de la Programación Orientada a Objetos de C++ son la causa principal de ello.

Conceptos generales de POO

La programación orientada a objetos (POO) se fundamenta en algunos principios o pilares que la definen:

Clase: Es una plantilla que define la estructura de un conjunto de objetos, que al ser creados se llamarán las instancias de la clase. Esta estructura está compuesta por

la definición de los atributos y la implementación de las operaciones (métodos).

Objeto: Es la implementación de una instancia de clase, es decir, una ocurrencia de esta, que tiene los atributos definidos por la clase, y sobre la que se puede ejecutar las operaciones definidas en ella.

Identidad: Característica de cada objeto que lo diferencia de los demás, incluyendo de aquellos que pudieran s pertenecer a la misma clase y tener los mismos valores en sus atributos.

Herencia: Es la capacidad que tienen las clases para heredar propiedades y métodos de otras clases.

2.3.2. Python

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses *Monty Python*. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible [4].

Se trata de un lenguaje interpretado o de script, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.
Python es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo, muy rápido y, lo que es más importante, divertido.

Python no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.
Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Ligh & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles.

Instalacion de Python

Existen varias implementaciones distintas de Python: CPython, Jython, IronPython, PyPy, etc

CPython es la más utilizada, la más rápida y la más madura. Cuando la gente habla de Python normalmente se refiere a esta implementación. En este caso tanto el intérprete como los módulos están escritos en C.

Jython es la implementación en Java de Python, mientras que IronPython es su contrapartida en C# (.NET). Su interés estriva en que utilizando estas implementaciones se pueden utilizar todas las librerías disponibles para los programadores de Java y .NET.

PyPy, por último, como habréis adivinado por el nombre, se trata de una implementación en Python de Python.

CPython está instalado por defecto en la mayor parte de las distribuciones Linux y en las últimas versiones de Mac OS. Para comprobar si está instalado abre una terminal y escribe `python`. Si está instalado se iniciará la consola interactiva de Python y obtendremos parecido a lo siguiente:

```
Python 3.5.1 (default, Mar  3 2016, 09:29:07)
[GCC 5.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

si no te muestra algo parecido no te preocupes, instalar Python es muy sencillo. Puedes descargar la versión correspondiente a tu sistema operativo desde la web de Python [9].

[9] Python: https://www.python.org/

Herramientas básicas

Existen dos formas de ejecutar código Python. Podemos escribir líneas de código en el intérprete y obtener una respuesta del intérprete para cada línea (sesión interactiva) o bien podemos escribir el código de un programa en un archivo de texto y ejecutarlo.

Para el trabajo del proyecto se uso PyCharm (2.2.3) como IDE de desarrollo, anque en algunas ocaciones se uso editor de texto, SUBLIME-TEXT (2.2.4) , para la edición rápida.

2.3.3. R

R es una implementación de software libre del lenguaje S pero con soporte de alcance estático. Se trata de uno de los lenguajes más utilizados en investigación por la comunidad estadística, siendo además muy popular en el campo de la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo o graficación [3] .

R es parte del sistema GNU y se distribuye bajo la licencia GNU GPL. Está disponible para los sistemas operativos Windows, Macintosh, Unix y GNU/Linux.

2.4. Metodologías de desarrollo de software

2.4.1. Metodología de desarrollo extremo

La programación extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software. Usadas conjuntamente proporcionan una nueva metodología de desarrollo software que se puede englobar dentro de las metodologías ligeras, que son aquéllas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible (pero no más). La programación extrema, dentro de las metodologías ágiles, se puede clasificar dentro de las evolutivas.

Una de las características de eXtreme Programming es que muchos de, si no todos, sus ingredientes son de sobra conocidos dentro de la rama de la ingeniería del software desde hace tiempo, incluso desde sus comienzos. Los autores de han seleccionado los que han considerados como los mejores y han profundizado en sus relaciones y en cómo se refuerzan unos a otros. El resultado ha sido una metodología única y compacta. Por eso, aunque se pueda alegar que la programación extrema no se base en principios nada nuevos, se ha de aclarar que, en conjunto, es una nueva forma de ver el desarrollo de software.

El proceso de desarrollo extremo

La programación extrema parte del caso habitual de una compañía que desarrolla software, generalmente software a medida, en la que hay diferentes roles: un equipo de gestión, un equipo de desarrolladores y los clientes. La relación con el cliente es totalmente diferente a lo que se ha venido haciendo en las metodologías tradicionales que se basan fundamentalmente en una fase de captura de requisitos previa al desarrollo y una fase de validación posterior al mismo.

J.C.A.R

Interacción con el cliente

En la programación extrema al cliente no sólo se le pide que apoye al equipo de desarrollo, en realidad podríamos decir que es parte de él. Su importancia es capital a la hora de abordar las historias de los usuarios y las reuniones de planificación, como veremos más adelante. Además, será tarea suya retroalimentar al equipo de desarrolladores después de cada iteración con los problemas con los que se ha encontrado, mostrando sus prioridades, expresando sus sensaciones... Existirán métodos como pruebas de aceptación que ayudarán a que la labor del cliente sea lo más fructífera posible.

En resumen, el cliente se encuentra mucho más cercano al proceso de desarrollo. Se elimina la fase inicial de captura de requisitos y se permite que éstos se vayan definiendo de una forma ordenada durante el tiempo que dura el proyecto. El cliente puede cambiar de opinión sobre la marcha y a cambio debe encontrarse siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario.

El proceso de captura de requisitos de XP gira entorno a una lista de características que el cliente desea que existan en el sistema final. Cada una de estas características recibe el nombre de historias de usuarios y su definición consta de dos fases:

En la primera fase el cliente describe con sus propias palabras las características y el responsable del equipo de desarrollo le informa de la dificultad técnica de cada una de ellas y por lo tanto de su coste.

La segunda fase consiste en coger las primeras historias que serán implementadas (primera iteración) y dividirlas en las tareas necesarias para llevarlas a cabo.

Este proceso es una de las principales diferencias con las metodologías tradicionales. Aunque las historias de usuarios guardan cierta relación con otras técnicas como los casos de uso de UML, su proceso de creación es muy diferente. En lo que al cliente se refiere no se le exige que especifique exactamente lo que quiere al principio con un documento de requisitos de usuario. La parte que se mantiene con este documento es

que es el cliente el que tiene que escribir lo que quiere, no se permite que alguien del equipo de desarrolladores lo escriba por él.

Planificación del proyecto

La planificación debe de seguir unas ciertas premisas. La primordial es que las entregas se hagan cuanto antes y que con cada iteración el cliente reciba una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo habrá para trabajar en ella posteriormente. Se aconsejan muchas entregas y muy frecuentes. De esta forma, un error en una parte esencial del sistema se encontrará pronto y, por tanto, se podrá arreglar antes.

Sin embargo, los requisitos anteriores en cuanto a la planificación no deben suponer horas extra para el equipo de desarrollo.

Pero lo mejor de todo es que a la hora de planificar uno se puede equivocar. Es más, todos sabemos que lo común es equivocarse y por ello la metodología ya tiene previsto mecanismos de revisión. Por tanto, es normal que cada 3 a 5 iteraciones se tengan que revisar las historias de los usuarios y re-negociar nuevamente la planificación.

Diseño, desarrollo y pruebas

El desarrollo es la pieza clave de todo el proceso de programación extrema. Todas las tareas tienen como objetivo que se desarrollo a la máxima velocidad, sin interrupciones y siempre en la dirección correcta. También se otorga una gran importancia al diseño y establece que éste debe ser revisado y mejorado de forma continua según se van añadiendo funcionalidades al sistema

La clave del proceso de desarrollo de XP es la comunicación. La gran mayoría de los problemas en los proyectos de desarrollo son provocados por falta de comunicación en

el equipo, así que se pone un gran énfasis en facilitar que la información fluya lo más eficientemente posible.

Como ya hemos visto con anterioridad, uno de los principios de la programación extrema es la simplicidad. El diseño debe ser lo más simple posible, pero no más simple. El paradigma KISS (*'Keep It Small and Simple'*)se lleva hasta las últimas consecuencias. Por ejemplo, se hace énfasis en no añadir funcionalidad nunca antes de lo necesario, por las sencillas razones de que probablemente ahora mismo no sea lo más prioritario o porque quizás nunca llegue a ser necesaria.

Supongamos que ya hemos planificado y dividido en tareas, como se ha comentado en los párrafos anteriores. Lo lógico sería empezar ya a codificar. Pues no. Nos encontramos con otro de los puntos clave de la programación extrema (y que sí es innovador en ella): las pruebas unitarias se implementan a la vez hay que el código de producción. De hecho cada vez que se va a implementar una pequeña parte se escribe una prueba sencilla y luego el código suficiente para que la pase. Cuando la haya pasado se repite el proceso con la siguiente parte.

Esta forma de usar las pruebas unitarias ayuda a priorizar y comprobar la evolución del desarrollo y que ofrecen retroalimentación inmediata. Ya no hay imprescindibles dos equipos diferenciados que desarrollan y prueban cada uno por su cuenta. Ahora el ciclo se basa en implementar una prueba unitaria, codificar la solución y pasar la prueba, con lo que se consigue un código simple y funcional de manera bastante rápida. Por eso es importante que las pruebas se pasen siempre al 100

Las pruebas unitarias no se han de confundir con las pruebas de aceptación que han sido mencionadas con anterioridad. Éstas últimas son pruebas realizadas por el cliente o por el usuario final para constatar que el sistema hace realmente lo que él quiere.

La programación extrema viene a perseguir lo que se ha venido a llamar integración continua. De esta forma, haciéndolo cada vez con pequeños fragmentos de código, se evita la gran integración final.

En todo desarrollo de programación extrema debería existir, por tanto, una versión siempre integrada a la sincronización por parte de los desarrolladores con el repositorio

central debe darse como mínimo una vez al día, de manera que los cambios siempre se realicen sobre la última versión. De esta forma nos podemos asegurar de que las modificaciones que hacemos no se estén haciendo sobre una versión obsoleta

El proceso de desarrollo no lo va a hacer un desarrollador en solitario, sino siempre con otra persona, algo que se ha venido a llamar programación por parejas. Una pareja de desarrolladores debe compartir ordenador, teclado y ratón. El principal objetivo es realizar de forma continua y sin parar el desarrollo una revisión de diseño y de código. Las parejas deben ir rotando de forma periódica para hacer que el conocimiento del sistema se vaya difundiendo por el equipo (facilitándose que el código sea de todos), a la vez que se fomentan el entrenamiento cruzado.

Resumen de XP

Todos los punto anteriormente detallados, bien los podemos definir y resumir en la siguiente lista, obviamente cada uno de ellos lleva consigo un gran significado de fondo, pero en general y conociendo la metodología se puede uno basar en la siguiente lista para cerciorarse de que se están cumpliendo los puntos que la metodología estipula que se deben llevar a cabo.

1. El juego de la planificación (the planning game).
2. Pequeñas entregas (small releases).
3. Metáfora (metaphor).
4. Diseño simple (simple design).
5. Pruebas (testing).
6. Refactorización (refactoring).
7. Programación por parejas (pair programming).
8. Propiedad colectiva (collective ownership).
9. Integración continua (continous integration).
10. 40 horas semanales (40-hour week).
11. Cliente en casa (on-site costumer).

J.C.A.R

12. Estándares de codificación (coding standards).

Se puede considerar como el resumen de bolsillo de la metodología de desarrollo extremo, y llevarla a cada lugar en el que se este desarrollando.

J.C.A.R

Capítulo 3

DESCRIPCIÓN DE ACTIVIDADES REALIZADAS

3.1. Análisis

3.2. Diseño

3.3. Desarrollo

3.4. Pruebas

3.5. Implementación

3.6. Retroalimentación

3.7. Resultados

3.8. Conclusiones y recomendaciones

3.9. Referencias Bibliográficas & Glosario

Glosario

Linux

GNU/Linux Sistema Operativo creado y distribuido por miles de personas alrededor del mundo 7

Mac

Sistema Operativo creado y distribuido por Apple 7

SO

Es el sistema o conjunto de aplicaciones que permiten que una computadora lleven a cabo sus funciones. 7

Windows

Sistema Operativo creado y distribuido por Microsoft. 7

J.C.A.R

Bibliografía

[1] **Ian Sommerville**, Ian Sommerville, María Isabel Alfonso Galipienso [*ingeniería del software, Séptima edición.*]. Pearson Educación.S.A., Madrid, 2005. ISBN: 84-7829-074-5

[2] **Beck** Kent Beck [.*Extreme Programming Explained: Embrace Change"*], Addison-Wesley Pub Co; ISBN: 0201616416, 1.ª Edición octubre 1999

[3] **Development Core Team (2008)**. R Foundation for Statistical Computing, [*R: A language and environment for statistical computing.*] Vienna, Austria. ISBN 3-900051-07-0, http://www.R-pro ject.org

[4] **Rául Gonzáles Duque**, Rául Gonzáles Duque [*Python para todos.*] Este libro se distribuye bajo una licencia Creative Commons Reconocimiento 2.5 España

[5] **David** David E.Brumbaugh [*Object-Oriented Developement with C++*]. Edt.Wile, 1994.

[6] **Fowler** Martin Fowler [*Variations on a Theme of XP*], http://www.martinfowler.com/articles/xpVariation.html

[7] **Harrison** Peter Harrison [*Evolutionary Programming*], http://www.devcentre.org/research/evoprogramming.htm