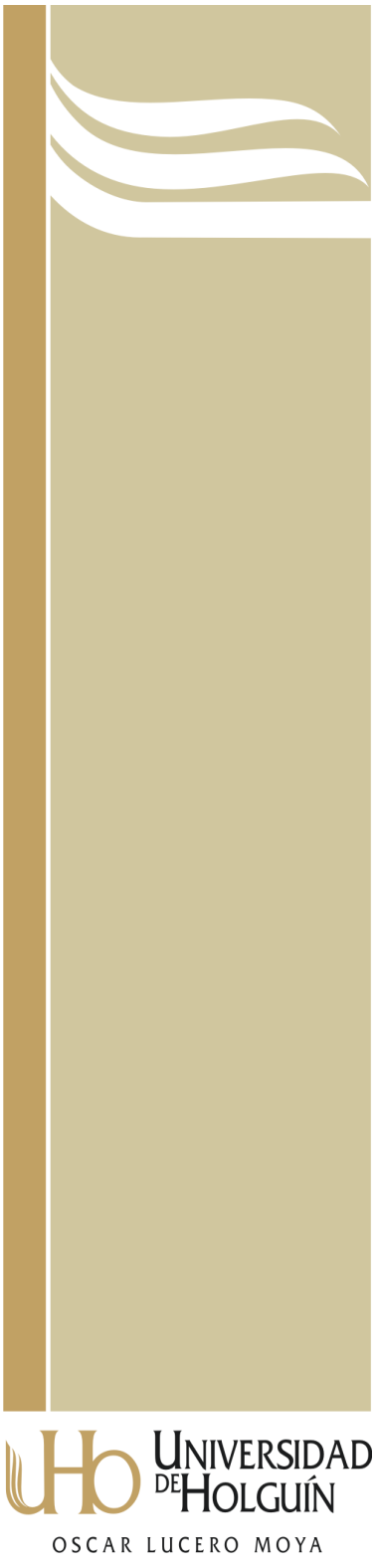
UNIVERSIDAD DE HOLGUÍN "OSCAR LUCERO MOYA"
FACULTAD DE INFORMÁTICA Y MATEMÁTICA

Herramienta para aplicar Análisis de Componentes Independientes a registros de movimientos oculares sacádicos

Tesis en Opción al Título de Ingeniero Informático

Abel Antonio Fernández Higuera

Holguín, 2014



UNIVERSIDAD DE HOLGUÍN "OSCAR LUCERO MOYA"
FACULTAD DE INFORMÁTICA Y MATEMÁTICA

Herramienta para aplicar Análisis de Componentes Independientes a registros de movimientos oculares sacádicos

Tesis en Opción al Título de Ingeniero Informático

Autor: Abel Antonio Fernández Higuera

Tutores: M. Sc. Roberto Antonio Becerra García
Dr. C. Rodolfo Valentín García Bermúdez
Dr. C. Gonzalo Joya Caparrós

Holguín, 2014

DEDICATORIA

A mis padres, las personas que siempre y pase lo que pase, han estado, están, y estarán a mi lado.

AGRADECIMIENTOS

A mis padres, por todo lo que han hecho para que yo llegue hasta donde estoy hoy. Todo lo que soy y seré, sin duda alguna, se lo debo a ellos.

A mi tía Ene, que más que una tía ha sido otra madre para mi, y siempre ha estado ahí.

A toda mi familia en general que siempre me ha apoyado en todo y sería muy difícil mencionarlos sin omitir algún nombre.

A mi novia Lianet, que se ha convertido en mi musa y una parte muy importante de mi vida, gracias por existir.

A Regino, Sureya y Lisbeth, que aunque sea poco el tiempo que llevo conociéndolos, siempre me han tratado como parte de la familia.

A todos mis compañeros de aula, por hacerme pasar cinco años increíbles.

A Idertator, que siempre ha sido mi modelo de informático, un millón de gracias por confiar en mi y enseñarme tantas cosas. A pesar de su tiempo tan ocupado y sus millones de diplomantes, pocas veces me tocó enfrentarme a su piloto automático.

A Fofi, que a pesar de su complicado tiempo y lo complejo que sea su calendario, siempre ha estado ahí. Muchas gracias.

A Michel, Andrés, Mayito, Lisandra, y toda la tribu de indios en general, por estar ahí y ayudar siempre en lo que haga falta.

En fin, a todas las personas que de una forma u otra han estado ahí, muchas gracias.

RESUMEN

El análisis de los movimientos oculares constituye una herramienta útil para el estudio de una gran variedad de disfunciones neurológicas entre las que se encuentra la Ataxia Espinocerebelosa Tipo 2 (SCA2, en inglés *Spinocerebellar Ataxia Type 2*). Existe un método computacional de separación de señales conocido como Análisis de Componentes Independientes (ICA) muy útil para analizar por separado las componentes que intervienen en el modelo de generación de sácadas. Este trabajo aborda el proceso de diseño y construcción de una herramienta que posibilita la utilización de ICA en los registros de movimientos oculares sacádicos que se obtienen en el Centro de Investigación y Rehabilitación de Ataxias Hereditarias (CIRAH) de la ciudad de Holguín. Como resultado de la investigación, se obtiene una herramienta informática que permite aplicar ICA a este tipo de registros, lo que contribuye al aumento de la calidad de los servicios investigativos y en consecuencia asistenciales que se ofrecen en el CIRAH.

ABSTRACT

The eye movement analysis is a useful tool for the study of a variety of neurological disorders including the Spinocerebellar Ataxia Type 2 (SCA2). A computational method for signal separation known as Independent Component Analysis (ICA) is helpful in analyzing separately the components involved in the model generating saccades. This investigation discusses the process of designing and building a tool which enables the use of ICA in the records of saccadic eye movements obtained at the Center for Research and Rehabilitation of Hereditary ataxias (CIRAH) in the city of Holguin. As a result of the research, a software tool was obtained in order to allow the application of ICA to these records, this application contributes to improving the quality of investigative services and care offered in CIRAH.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

Cuba es el país que presenta la mayor concentración de enfermos con ataxias hereditarias a nivel internacional. Las ataxias hereditarias son un grupo de enfermedades clínicas, patológicas y genéticamente heterogéneas que se encuentran dentro de los desórdenes neurodegenerativos causados por la degeneración del cerebelo y sus vías aferentes y eferentes, la médula espinal, los nervios periféricos y el tronco cerebral [4].

En Cuba existen cerca de 800 enfermos y 8000 presintomáticos que tienen riesgos de desarrollar alguna ataxia hereditaria en los próximos años. La forma molecular más frecuente es la SCA2 con una prevalencia de 43 casos por cada 100,000 habitantes en la provincia Holguín, cifra que no ha sido superada por ninguna otra región o país a escala internacional [4].

Por esta razón se crea Centro de Investigación y Rehabilitación de las Ataxias Hereditarias (CIRAH) de la ciudad de Holguín, el cual es el encargado de realizar entre otras contribuciones científicas, asistenciales y sociales, la instauración de un programa de neurorehabilitación multifactorial, la caracterización neurofisiológica de los sistemas somáticos y autosómicos, la identificación de factores modificadores de la edad de inicio y el curso evolutivo de la enfermedad, la instauración de un programa de diagnóstico prenatal y presintomático para la SCA2 y el desarrollo de un modelo celular y animal transgénico para esta enfermedad.

Es importante señalar que el término ataxia, no se refiere a una enfermedad específica ni a un determinado diagnóstico, sino a un estado patológico de la coordinación de los movimientos [5]. A menudo esta palabra se utiliza para describir un trastorno de la marcha que se manifiesta por inestabilidad, incoordinación y aumento de la base de sustentación. Resulta de una disfunción

a nivel del cerebelo y/o sus vías, así como alteraciones en la médula espinal, nervios periféricos o una combinación de estas tres condiciones.

Los movimientos oculares tienen un rol muy importante en la identificación de las disfunciones en un amplio rango de enfermedades neurológicas, y entre estos, los movimientos sacádicos. Estos últimos son utilizados para cambiar bruscamente el campo visual y proveen de una útil herramienta en la exploración de las funciones neurales. Se ha comprobado que las estructuras neurales encargadas de generar este tipo de movimientos, son blancos importantes de la SCA2 [2]. Estas consideraciones convierten a las características asociadas a los movimientos oculares en criterios de evaluación muy sensibles y de alto valor diagnóstico endofenotípico, para el diagnóstico y seguimiento evolutivo de esta enfermedad.

En el proceso de medición de los movimientos oculares se emplea en el CIRAH un equipo de la firma alemana OtoScreen con el cual se obtienen los electrooculogramas. Este equipo realiza las mediciones utilizando el método conocido como electrooculografía, que consiste en la medición de los potenciales eléctricos en la zona cercana a los ojos, ofreciendo como salida, los valores de los potenciales asociados a la señal de estímulo visual aplicada y la respuesta del paciente en un fichero ASCII.

Con el objetivo de obtener una plataforma informática que permita realizar el procesamiento de electrooculogramas realizados a pacientes con SCA2 según los protocolos establecidos por el CIRAH, se ha desarrollado en la Universidad de Holguín "Oscar Lucero Moya "una herramienta denominada NSEog encargada de realizar esta tarea.

Las afectaciones de pacientes con SCA2 provocan que los registros oculares realizados a los pacientes presenten graves afectaciones en la forma de onda de los movimientos sacádicos, menores velocidades máximas, y además, la ocurrencia de movimientos involuntarios, parpadeos y temblores, modifican sustancialmente la forma de las sácadas [2].

Los modelos biológicos científicamente establecidos han determinado la existencia de dos centros neuronales independientes, que se encargan de la generación y control de los movimientos oculares sacádicos, los cuales no son directamente medibles por medio de la exploración electrofisiológica de las estructuras del sistema nervioso central [6].

Existe un método de separación de señales conocido como Análisis de Componentes Independientes (ICA) que permite la separación de varias fuentes de señal que aparecen mezcladas en los registros digitales capturados mediante el uso de equipos que miden la actividad eléctrica humana. Los resultados de separar los 2 centros neuronales utilizando ICA a partir de señales sacádicas permiten la evaluación del impacto de la SCA2 sobre cada uno de estos por separado. Por otro lado, los parámetros asociados a estas componentes pueden emplearse en la clasificación automática de diferentes estadíos de la enfermedad.

La **problemática** gira en torno a que no existe una herramienta conocida, capaz de realizar el Análisis de Componentes Independientes en registros de movimientos oculares para lograr obtener las dos componentes involucradas el sistema sacádico.

Esto plantea el siguiente **problema**: ¿Cómo elaborar una herramienta que permita realizar el Análisis de Componentes Independientes en registros de movimientos oculares sacádicos de pacientes con SCA2?

El **objeto de estudio** de este trabajo son las herramientas para aplicar el Análisis de Componentes Independientes a registros oculares sacádicos, cuyo **campo de acción** es el Análisis de Componentes Independientes en registros de movimientos oculares sacádicos de pacientes con SCA2.

Con el fin de dar solución al problema antes expuesto, se define como **objetivo general**, elaborar una herramienta que permita realizar el Análisis de Componentes Independientes en registros de movimientos oculares sacádicos de pacientes con SCA2.

Para guiar la investigación se trazaron las siguientes **preguntas científicas**:

- ¿Qué fundamentos teóricos sustentan la elaboración de una herramienta informática que permita realizar el Análisis de Componentes Independientes en registros de movimientos oculares sacádicos de pacientes con SCA2?
- ¿Cuáles son los métodos y herramientas más adecuadas para darle solución al problema planteado?
- ¿Cómo diseñar e implementar una herramienta informática que permita realizar el Análi-

sis de Componentes Independientes en estudios sacádicos de pacientes con SCA2 de forma que cumpla con los requisitos necesarios para su aplicación en el CIRAH?

- ¿Será sostenible en el tiempo el producto informático a desarrollar?

Para darles respuesta a las preguntas científicas y cumplir el objetivo trazado, se proponen las siguientes **tareas**:

1. Elaborar los fundamentos teóricos de la investigación.
2. Seleccionar los métodos y herramientas adecuadas para elaborar la herramienta informática.
3. Diseñar la arquitectura de la herramienta informática.
4. Implementar la herramienta informática.
5. Valorar el estado de aceptación de la herramienta informática propuesta mediante pruebas de aceptación de la metodología XP.

Durante el transcurso de la investigación se emplearon un grupo de métodos teóricos y empíricos que guiaron este proceso y facilitaron su estructuración y desarrollo. A continuación se detallan como fueron utilizados cada uno de los mismos:

Métodos teóricos

Histórico-lógico: se utilizó para comprender el estudio de ICA a través de la evolución histórica de los métodos y técnicas empleadas en esta área de conocimiento.

Sistémico: se empleó para detectar los sistemas y subsistemas que componen la solución propuesta, así como las relaciones y flujos de ejecución entre estos.

Modelado: se utilizó para representar los procesos y entidades implicados en el ICA.

Métodos empíricos

Entrevista: se empleó para recolectar información, comprender mejor el negocio y extraer los requisitos funcionales y no funcionales de la solución que se propone.

Experimental: se utilizó para verificar el funcionamiento de la herramienta propuesta y comprobar la factibilidad de los datos obtenidos de la misma.

El presente documento consta de 3 capítulos con sus respectivas conclusiones parciales, introducción, conclusiones generales, recomendaciones, referencias bibliográficas y anexos. Los capítulos se describen de la siguiente manera:

Capítulo 1. Marco Teórico: Se explican los conceptos y criterios que se utilizaron para el diseño de la herramienta.

Capítulo 2. Exploración y Planificación: Se hace un análisis de los requerimientos de la aplicación y se planifica el proceso de implementación. Se hace la presentación del negocio mediante Historias de Usuario.

Capítulo 3. Implementación y Prueba: Primeramente se explica el diseño de la aplicación y luego el flujo de iteraciones utilizadas para su implementación. Este capítulo finaliza con las pruebas realizadas para comprobar la calidad de la aplicación.

1

Marco Teórico

Para llevar a cabo la investigación propuesta se necesita analizar y detallar ciertos aspectos teóricos que sirven como base para la misma. Además, es importante señalar que su contextualización en la institución a la que se destina el producto, es de vital importancia para el éxito de los resultados que se esperan obtener. En el siguiente capítulo se estudiarán además las tecnologías y herramientas disponibles en el mercado, de forma que se seleccionen las más adecuadas para el desarrollo de la solución propuesta. Los siguientes epígrafes describen en detalle cada una de estas cuestiones.

1.1. La Ataxia y el CIRAH

La enfermedad conocida como ataxia espino cerebelosa se refiere a un estado patológico de la coordinación de los movimientos, caracterizada por trastornos de la marcha que se manifiestan por inestabilidad, descoordinación y aumento de la base de sustentación, como resultado de una disfunción a nivel del cerebelo o de sus vías, así como alteraciones en la médula espinal, nervios periféricos o una combinación de estas tres condiciones [5].

En el año 2000 se crea el CIRAH como proyecto encaminado a buscar y brindar a las personas que padecen ataxias hereditarias soluciones y tratamientos que mejoren sus condiciones de vida. Para ello asume la atención a pacientes enfermos y a sus descendientes con riesgos de contraer la enfermedad. El objetivo principal del centro es la investigación científica, así como la búsqueda y desarrollo de programas destinados a la rehabilitación físico - motora, psicológica

y del lenguaje. Además se desarrollan ensayos clínicos con el propósito de crear un protocolo de tratamiento de la enfermedad.

1.2. Movimientos oculares

Los movimientos oculares son elementos muy útiles en la detección de muchas enfermedades neurológicas. Entre estos, los movimientos de persecución y sacádicos, son necesarios para mantener el seguimientos de objetos que se mueven, y proveen de una útil herramienta en la exploración de las funciones neurales.

Una clasificación de los movimientos oculares divide a los mismos en dos grandes categorías: los movimientos de estabilización, que tratan de fijar en la retina una imagen estable y los sacádicos, referidos al movimiento de los ojos dentro del campo visual para traer objetos de interés al área de visión más precisa [2].

Fundamentalmente existen cuatro tipos de movimientos oculares [5], cada uno controlado por un sistema neural distinto pero que comparten la misma vía final común, las neuronas motoras que llegan a los músculos extra-oculares.

Los movimientos sacádicos: movimientos súbitos y enérgicos de tipo espasmódico, ocurren cuando la mirada cambia de un objeto a otro. Colocan nuevos objetos de interés en la fóvea y disminuyen la adaptación en la vía visual, que podría ocurrir si la mirada se fijara en un solo objeto por períodos prolongados.

Los movimientos suaves de persecución (de búsqueda): movimientos oculares de seguimiento que se producen cuando se observa un objeto en movimiento.

Los movimientos vestibulares (movimientos de ajuste): ocurren como respuesta a estímulos iniciados en los conductos semicirculares, para mantener la fijación visual mientras se mueve la cabeza.

Los movimientos de convergencia: aproximan los ejes visuales entre sí cuando se enfoca la atención en objetos cercanos al observador.

1.2.1. Movimientos oculares sacádicos

Los sacádicos constituyen uno de los movimientos más característicos de los ojos. Son movimientos fundamentalmente voluntarios y los utilizamos para dirigir la mirada a un objeto que nos llama la atención. Su objetivo no es otro que el de situar la imagen visual frente a la fóvea, que es la región de la retina que dispone de mayor agudeza visual [7].

Tras cada movimiento sacádico, los ojos permanecen relativamente quietos durante periodos de tiempo muy breves que se denominan fijaciones. El objetivo de los periodos de fijación es enfocar una zona concreta de la escena para percibir y asimilar la información visual que hay en ella.

El sistema de movimientos sacádicos provee a los investigadores de una herramienta poderosa para explorar el control cognitivo del comportamiento. Este es un sistema del comportamiento cuya salida puede ser medida con precisión excepcional, y cuya entrada puede ser controlada o manipulada, de muchas maneras diferentes.

Este tipo de movimientos es de suma importancia para el estudio de la SCA2 pues el enlentecimiento sacádico es uno de los rasgos clínicos más comunes en sujetos portadores de la mutación de este tipo de ataxia, lo cual ha permitido incluso la confirmación en retrospectiva de la misma.

1.2.2. Características de los registros a procesar

Los registros oculográficos se obtienen a partir de electooculografías, que no son más que exámenes cuyo objetivo es colocar electródos cerca de los músculos de los ojos para medir el movimiento de éstos. Estos estudios, normalmente se realizan para estudiar las alteraciones del sueño.

Esta técnica fue introducida por Fenn y Hursh en 1934 la cual mide el potencial de la retina-córnea colocando electródos en la piel alrededor del ojo como se muestra en la siguiente Figura, convirtiéndose luego en una señal que mide el ángulo de rotación de los ojos.

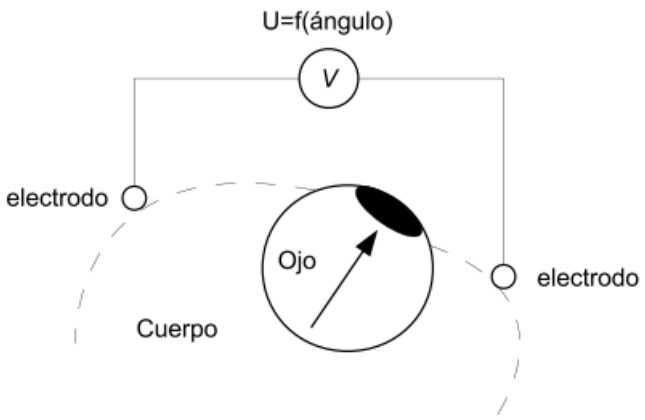


Figura 1: Principio de funcionamiento de la electrooculografía. Tomado de [1]

Es importante señalar que el potencial de la retina-córnea depende directamente de la iluminación que llega a la retina, utilizándose frecuentemente para la medición de la respuesta del ojo a la luz [8]. Debido a esto, cuando se utiliza en la medición de movimientos oculares se recomienda mantener condiciones de iluminación constantes.

La técnica tiene como principales ventajas su poca invasividad, ayudando en la cooperación final de los sujetos y la posibilidad de registrar amplios movimientos horizontales ($\pm 40^{\circ}$) con una resolución de 1° [2].

La señal resultante al aplicar la técnica de electrooculografía se denomina **electrooculograma**. Estos se almacenan en archivos de texto o binarios dependiendo del equipo utilizado para su captura. En la actualidad se utiliza en el CIRAH un electronistagmógrafo de la marca alemana Otoscreen para realizar el registro de electrooculogramas que produce archivos de texo ASCII[1] con extensión CSV[2]. Su estructura es bastante sencilla y separada por pruebas, facilitándose de esta manera su lectura e interpretación.

Para la captura de los movimientos oculares sacádicos, antisacádicos, de persecución suave y nistágmicos que se estudian en el CIRAH se utilizan una serie de pruebas. En estas pruebas

[1] Código normalizado americano para el intercambio de la información (en inglés *American Standard Code for Information Interchange*)

[2] Documento en formato abierto, sencillo para representar datos en forma de tablas, en las que las columnas se separan por comas

se coloca al paciente a una determinada distancia de un monitor donde aparecerá un estímulo visual. Luego se fija la cabeza del paciente y se termina la conFiguración de la prueba con la colocación de los electrodos del electronistagmógrafo. Antes del comienzo de la prueba se le pide al paciente que siga con la mirada el estímulo que aparece en la pantalla. La ejecución de la prueba consiste básicamente en el registro de la respuesta del paciente según el estímulo visual establecido [2].

1.3. Análisis de Componentes Independientes (ICA)

Un problema fundamental de muchas disciplinas relacionadas con la investigación es encontrar una representación aceptable de datos multivariables, por ejemplo, vectores aleatorios. Por razones de simplicidad conceptual y computacional esta representación es mostrada como una transformación lineal de los datos originales. El Análisis de Componentes Independientes (ICA por sus siglas en idioma inglés), es un método que pretende encontrar una distribución lineal de datos no Gaussianos, de forma que los componentes que la forman sean estadísticamente independientes, o tan independientes como sea posible[9]. Es importante señalar que ICA es un caso especial de separación a ciega de señales.

1.3.1. Separación ciega de señales

La separación ciega de señales es un problema importante en el procesamiento de las mismas. Por lo general, las señales capturadas por sensores, son mezclas de varias fuentes, en principio independientes, las cuales se ven alteradas al transmitirse por la actuación sobre ellas de un medio material que las perturba. El objetivo de la separación de fuentes consiste en recuperar las señales originales partiendo de estas mezclas. Esta técnica puede ser aplicada en campos tales como el procesamiento de señales en radar, sonar y en la voz, para realzar la señal original perturbada sobre otras señales [2].

Un problema clásico para entender ICA y la Separación Ciega de Señales es el llamado *cocktail-party problem*. Dicho problema consiste en tratar de separar todas las voces de n personas en una habitación hablando simultáneamente, suponiendo que se cuenta con al menos n micrófonos para capturar dichas voces [9].

Actualmente el área de la Separación Ciega de señales es aplicable a una gran variedad de problemas reales especialmente en el campo de la ingeniería biomédica, reconocimiento de voz, econometría, minería de datos, etc. Es importante señalar que las técnicas basadas en separación ciega de señales no necesitan un conjunto de datos de entrenamiento y no asumen ningún tipo de conocimiento a priori.

1.3.2. Fundamentación teórica

Denotando un vector aleatorio ovservado $X = [X_1, X_2, \ldots, X_m]^T$ cuyos n elementos son mezclas de m elementos independientes de un vector aleatorio $S = [S_1, S_2, \ldots, S_m]^T$ dado por $X = AS$ donde A representa una matriz de $m \times m$ elementos. El objetivo de ICA es encontrar una matriz de separación W, que obtendrá Y, como la mejor aproximación posible de S [10]:

$$Y = WX \cong S \tag{1}$$

Para aplicar ICA se deben asumir primero cinco conceptos fundamentales que se detallarán a continuación [9]:

1. La independencia estadística de cada una de las fuentes S_i del vector de fuentes S. Este concepto se detallará en el siquiente epígrafe.

2. La matriz de mezclas debe ser cuadrada, o sea, que el número de mezclas debe ser igual al número de fuentes y las mezclas deben ser linealmente independientes entre ellas.

3. La única fuente de estocasticidad en el modelo es el vector fuente S que no tiene ruido externo. Con esto se busca que el modelo esté libre de ruido.

4. Se asume que los datos son centrados (media cero). Además, para algunos algoritmos, los data debe ser pre-procesada anteriormente, a veces, el vector X debe ser blanqueado. Se dice que un vector aleatorio está blanqueado si su media es cero para todos sus componentes y su matriz de covarianzas es la matriz identidad I [2].

5. Las señales fuentes deben no tener una función de densidad de probabilidad Gaussiana, excepto para una fuente simple que puede ser gaussiana.

1.3.3. Independencia estadística

La hipótesis esencial que determina los algoritmos de separación ciega de señales es la independencia estadística de las fuentes pues los valores de las señales originales no dan ninguna información acerca del resto de las fuentes. Esta exigencia se considera restrictiva desde el punto de vista estadístico, sin embargo, no así en la práctica, puesto que en la mayor parte de las situaciones es bastante factible suponer que señales s_i, generadas por distintos procesos serán independientes entre sí.

Se dice que un vector aleatorio x es estadísticamente independiente si la función de densidad de probabilidad conjunta de x es factorizable en el producto de las funciones de densidad de probabilidad marginales de sus componentes, y x es independiente si [10]:

$$p_x(x) = p_{x_1}(x_1) * p_{x_2}(x_2) * \ldots * p_{x_n}(x_n) = \prod p_{x_1}(x_1) \tag{2}$$

Considerando simplemente la independencia estadística de señales originales e imponiendo que no más de una de estas tenga una función de distribución Gaussiana, ya se está en disposición de dar solución al problema de separación ciega de señales. Esto se realizará buscando un conjunto de coeficientes que transformen las señales mezcladas u observadas x en un conjunto de estimaciones y tales que dichas estimaciones sean independientes, como se analizó en el epígrafe anterior.

1.3.4. Modelo de mezcla lineal instantáneo

El modelo de mezcla lineal instantáneo o más conocido como lineal sin memoria es el más simple de los propuestos y es además el más tratado por los autores especializados en el tema [2].

Dicho modelo supone que el valor de las observaciones $(x_i(t))$ en un instante dado es una función lineal de los valores de las fuentes originales $(s_i(t))$ en ese mismo instante de tiempo

y no influyen en la mezcla en ningún caso los valores de las fuentes en los instantes de tiempo anteriores. Una función lineal sobre un conjunto de variables se reduce a un conjunto de coeficientes reales que multiplican a los originales. De esta forma, se puede describir este modelo matemático mediante la siguiente expresión [2]:

$$x_i(t) = a_{i1}s_1(t), \ldots, a_{ij}s_j(t), \ldots, a_{ip}s_p(t) i = 1, \ldots, q \tag{3}$$

de forma resumida, utilizando vectores de variables y una matriz de coeficientes reales se puede definir que:

$$x(t) = A \times s(t), donde A = \begin{pmatrix} a_{11} & \ldots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{i1} & \ldots & a_{ij} \end{pmatrix} \tag{4}$$

La matrix A se denomina matriz de mezcla, y es definitivamente la encargada de realizar la combinación lineal de fuentes S. Al asumir que el número de observaciones es igual al número de fuentes, la matriz de mezclas A es cuadrada, tal y como se mostró en la ecuación anterior.

Por tanto, el problema de la separación de señales se pudiera resolver si se tuviera una matriz W denominada matriz de separación, cuya inversa es la equivalente a la matriz de mezclas A y de sus mismas dimensiones, con lo que al multiplicarla por las señales mezcladas se podría obtener una reconstrucción o estimación $y(t)$ de las fuentes originales. De esta forma la ecuación del modelo de separación es:

$$Y = WX \cong S \tag{5}$$

Reduciendo el problema de la separación ciega de señales lineal e instantánea mediante la aplicación del análisis de componentes independientes, recordemos que la solución será válida si las estimaciones $y(t)$ son equivalentes a las fuentes $s(t)$ salvo permutaciones y escalados invertibles [10].

1.3.5. ICA y la separación ciega de señales

Al ser la independencia estadística un principio básico para la resolución del problema de separación ciega de señales, relaciona dicho problema con el uso de la técnica de ICA.

Se puede definir el ICA del vector aleatorio x, en su forma más genérica, cómo la técnica de búsqueda de la transformación lineal $s = W * x$ tal que las componentes s_i son lo más independiente posibles, en el sentido de maximizar una determinada función $F(s_1, s_2, ...s_m)$ que cuantifica la independencia [2].

El uso extensivo de la técnica de ICA para el problema de la separación ciega de señales, hace que dichos conceptos a veces se confundan. No obstante, debe quedar claro que ICA tan sólo es la herramienta o técnica que nos permite resolver un problema (el de la separación ciega de señales) en el caso concreto en que asumimos que las señales originales son estadísticamente independientes. De hecho, existen numerosas líneas de investigación que van más allá de esta aparente equivalencia entre ICA y la separación ciega de señales.

1.3.6. Algoritmos para realizar ICA

Existen varias formas para medir la independencia y cada una de esta involucra una serie de algoritmos para desarrollar ICA, cuyos resultados desembocan en matrices de separación ligeramente diferentes. Principalmente, hay dos familias de algoritmos de ICA [10]. Algunos algoritmos están basados en la minimización de la información mutua y otros en la maximización de la no-Gaussianidad.

1.3.7. Algoritmos basados en la minimización de la información mutua

La información mutua es definida por un par de variables aleatorias como:

$$I(X; H) = H(X) - H(X|Y) \tag{6}$$

donde $H(X|Y)$ es la entropía condicional y $H(X)$ es la entropía de X. La entropía condicional esta dada por la ecuación:

$$H(X|Y) = H(X,Y) - H(Y) \tag{7}$$

donde $H(X,Y)$ es la entropía conjunta de X y Y y H(Y) es la entropía de Y. Formalmente, la entropía para determinadas variables fue definida por Shannon en 1848 como:

$$H(X) = -\sum_{x} P(x) \log P(x) \tag{8a}$$

$$H(Y) = -\sum_{y} P(y) \log P(y) \tag{8b}$$

$$H(X,Y) = -\sum_{x,y} P(x,y) \log P(x,y) \tag{8c}$$

donde $P(x)$ es la probabilidad de que X se encuentre en un estado x. La entropía suele ser vista como una medida de incertidumbre. Entonces, volviendo a la ecuación 5 , la información mutua puede ser vista como una reducción de la incertidumbre respecto a la variable X después de una observación en Y [10]. Por consiguiente teniendo una algoritmo que busque minimizar la información mutua estaríamos buscando componentes (variables latentes) que son máximamente independientes.

Entre estos algoritmos se encuentra Infomax el cual se describe en el algoritmo 1 el cual surge después de hacer cierta manipulación sobre la ecuación 5.

Algoritmo 1: ICA basado en la minimización de la información mutua

1 $\rightarrow$ Inicializar $W(0)$ (por ejemplo con valores aleatorios)
2 $\rightarrow$ $W(t+1) = W(t) + \eta(t)(I - f(Y)Y^T)W(t)$
3 $\rightarrow$ **if** *no converge* **then**
4 $\quad$ **go to** 2

donde t representa un determinado paso de aproximación, $\eta(t)$ una función general que especifique al longitud del paso para la actualización de la matriz de separación W (este valor es usualmente una función exponencial o una constante), $f(Y)$ una función no-lineal usualmente

seleccionada conforme al tipo de distribución (sub o super Gaussiana), I la matriz identidad de dimensiones $m \times m$ y T es el operador transpuesta. En el caso de las distribuciones super-Gaussianas $f(Y)$ es usualmente seleccionada como:

$$f(Y) = \tanh(Y) \tag{9a}$$

y para las distribuciones sub-Gaussianas $f(Y)$ sería:

$$f(Y) = Y - \tanh(Y) \tag{9b}$$

1.3.8. Algoritmos basados en la maximización de la no-Gaussianidad

Otra manera para estimar las componentes independientes consiste en centrarse en la no-Gaussianidad. Desde entonces se asume que cada fuente subyacente no esta normalmente distribuida, una forma de extraer las componentes es forzando cada una de ellas a alejarse tanto como sea posible de la distribución normal. La neguentropía o entropía negativa puede ser usada para estimar la no-Gaussianidad, la misma es una medida de distancia de la normalidad definida por:

$$N(H) = H(X_{Gaussiana}) - H(X) \tag{10}$$

donde X sería un vector aleatorio conocido por ser no-Gaussiano, $H(X)$ es la entropía, y $H(X_{Gaussiana})$ es la entropía de un vector aleatorio Gaussiano cuya matriz de covarianzas es igual a la de X. Sin embargo, es difícil calcular la neguentropía usando la ecuación anterior, por lo cual se usan aproximaciones. Por ejemplo Hyvärinen y Oja [9] proponen la siguiente aproximación:

$$N(V) = E(\phi(V)) - E(\phi(V))^2 \tag{11}$$

donde V es una variable estándar no-Gaussiana (media 0 y varianza 1), U, una variable estándar Gaussiana y $\phi()$ una función no cuadrática usualmente $\tanh()$. Basado en estos conceptos surge el algoritmo FastICA detallado en el algoritmo 2.

Algoritmo 2: ICA basado en la maximización de la no-Gaussianidad

```
→ Inicializar W(0) (por ejemplo con valores aleatorios)
→ W_i^+ = E(φ'(W_i^T X))W_i − E(φ0(W_i^T X))
→ W_i = W_i^+ / ||W_i^+||
→ if i==1 then
    if no converge then
        go to 2
    else
        i = i+1
        go to 1
else
    W_i^+ = W_i − Σ_{j=1}^{i+1} W_i^T W_j W_j
    W_i = W_i^+ / ||W_i^+||
```

donde W_i es el vector columna de la matriz de separación, W_i^+ es una variable temporal usada para calcular W_i, $\phi\prime()$ es la derivada de $\phi()$. Una vez que W_i converge, el siguiente W_{i+1} debe ser ortogonal a este.

Es importante señalar que existen varias implementaciones de estos algoritmos basadas en ambos criterios para la obtención de las componentes independientes. Entre los algoritmos más conocidos no mencionados anteriormente se encuentran además, JADE, TDSEP, y CUBICA, aunque los más usados son sin dudas los primeros. Estudios han demostrado que los mejores resultados en la aplicación de ICA a registros de movimientos oculares sacádicos son los obtenidos mediante Infomax [2].

1.3.9. ICA y los movimientos oculares sacádicos

Un modelo aceptado del sistema sacádico afirma que la generación de las sácadas contiene una componente pulso y una componente escalón. El pulso consiste en un arranque de alta frecuencia que rápidamente mueve el ojo a una nueva posición, mientras que la componente escalón mantiene los ojos estables en la nueva posición [11].

Dentro del campo de la neurofisiología no se han encontrado muchas referencias al empleo de

ICA en el procesamiento de movimientos oculares, a diferencia de otros estudios, como pueden ser los electroencefalogramas (EEG) y electrocardiogramas (ECG), donde se encuentran centenares de investigaciones realizadas acerca del tema [12].

Fundamentalmente se han encontrado los trabajos realizados en el estudio de los movimientos sacádicos de vergencia realizados por Semmlow [13] en colaboración con otros autores. En colaboración con otros autores, desarrollando una metodología paralela aplicación de ICA a este tipo de movimientos oculares, este grupo de trabajo realizó la comprobación de la validez de su enfoque, por medio de la simulación.

En todos los casos se realizan secuencias de seguimiento de un estímulo, que conforma sácadas o movimientos sacádicos de vergencia. Luego cada sácada es aislada y considerada como una fuente independiente, para la aplicación de ICA, que realiza la extracción de las componentes pulsos y escalón presentes en la programación de los movimientos sacádicos, las cuales se analizaron anteriormente en este capítulo [13].

El modelo de generación de sácadas, que plantea la existencia de dos centros neuronales independientes, que se activan en diferentes instantes de tiempo, luego de la aparición de un estímulo visual, generando una componente tipo pulso no controlada y otra componente sostenida con forma de escalón, plantea la posibilidad de realizar la separación de ambas componentes a partir de un conjunto de observaciones utilizando ICA [2]. En la Figura 2 se pueden observar las dos componentes del sistema sacádico.

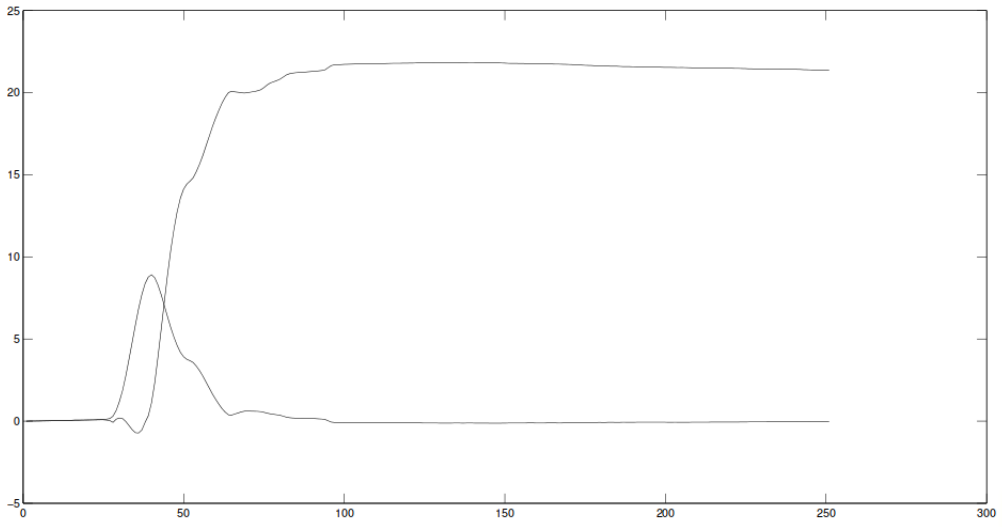


Figura 2: Componentes pulso y escalón después de aplicar ICA. Tomado de [2]

Para entender de una mejor forma la aplicación de ICA se puede analizar una prueba sacádica usando un estímulo a 30^o. La misma fue aplicada a cinco pacientes enfermos con ataxia y a cinco sujetos sanos, diagnosticados y clasificados en el CIRAH, y registrados con un electronistagmógrafo.

En ese trabajo, para aplicar ICA, cada respuesta a los cambios de estímulos fue considerada como una señal, y las sácadas a la derecha fueron invertidas para hacer un ensamble con las sácadas a la izquierda. Como resultado, son obtenidas las componentes pulso y escalón involucradas en la generación de las sácadas, como una marcada diferencia entre sujetos sanos y enfermo, como se muestra en la Figura 1 donde a la izquierda se observa el ensamble de sácadas, y al centro y la derecha las componentes de sujetos sanos y enfermos respectivamente. El resultado de esta prueba mostró resultados importantes para estudiar los efectos de la SCA2 en el sistema oculomotor [12]

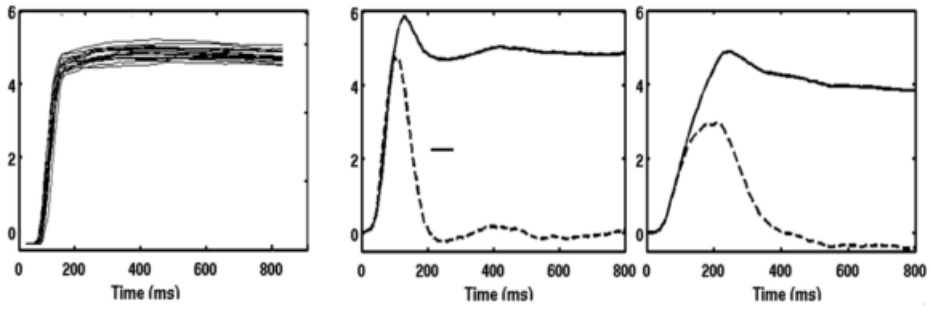


Figura 3: Diferencias de las componentes sacádicas en sujetos sanos y enfermos. Tomado de [2]

1.3.10. Requisitos para la aplicación de ICA en movimientos oculares sacádicos

Para la aplicación de ICA a los movimientos oculares sacádicos, es necesario la obtención de un conjunto de sácadas en respuesta a un estímulo visual, luego cada una de estas sácadas será tratada como una observación. En este caso, las señales producidas por los componentes neuronales de tipo pulso y escalón, constituirán las variables latentes s y la matriz de mezcla A se determinará a partir de la variabilidad presente en cada una de las respuestas. En este caso la aplicación de ICA se puede realizar a partir del cumplimiento de los siguientes requisitos [2]:

Independencia de las fuentes: El modelo que se utiliza se basa en la existencia de dos centros neuronales independientes en la generación de las componentes transitoria (tipo pulso) y sostenida (con forma de escalón), del mecanismo de generación de las sácadas. Aunque estas fuentes neuronales son independientes, la sincronización debida a que ambas se producen a consecuencia del mismo estímulo puede generar algún tipo de correlación temporal en el tramo inicial de las mismas. En la medida que la respuesta continúa este efecto debido al estímulo va disminuyendo, de modo que las componentes se convierten en totalmente independientes. La solución comúnmente adoptada para evitar esta posible correlación debida a la sincronización a causa del estímulo es realizar la evaluación de la matriz de mezcla sin tomar en cuenta la parte inicial de las respuestas.

Fuentes no Gaussianas: Las dos componentes neuronales que están presentes según el modelo pulso-escalón, no tienen distribución gaussiana, siendo este uno de los requisitos

para la aplicación de ICA.

Mezcla lineal instantánea:] Al considerar los centros generadores de las componentes pulso y escalón, como dos elementos neuronales diferentes que se activan de manera independiente, es razonable considerar que en los electródos estará presente una combinación lineal e instantánea de ambos, a partir de la suficiente evidencia científica que existe al respecto.

Estacionalidad espacial de las fuentes: Los centros neuronales involucrados no constituyen partes móviles en las estructuras del sistema nervioso central ubicadas en la cabeza. Los electrodos son fijados en zonas donde tampoco hay movimientos apreciables de la piel. También existe evidencia científica anterior del cumplimiento de esta condición, que confirma que las proyecciones espaciales de las componentes se mantienen invariables a lo largo del tiempo, más específicamente en el caso de las fuentes que reflejan el procesamiento de información relacionada con eventos, se asume generalmente como la suma de la actividad de generadores espacialmente estacionarios.

Menor cantidad de fuentes que de observaciones: Como se ha planteado anteriormente, se toman como observaciones cada una de las respuestas al estímulo, para obtener solamente las dos componentes independientes asociadas a la generación de las sácadas.

1.3.11. Procesamiento de las sácadas para aplicar ICA

Para la aplicación del ICA a los registros de movimientos oculares sacádicos se estableció una secuencia de pasos lógicos que permitiesen la conformación del conjunto de observaciones necesarias [2].

La data de los registros oculares, previamente procesada por el programa que identifica los puntos de comienzo y fin de las sácadas, posteriormente es editada por los especialistas médicos usando la plataforma NSEog, antes analizada, que marcan como no válidas las sácadas que no cumplen con los requisitos de presentar pestañeos, ruido excesivo, movimientos del paciente entre otros.

Luego de seleccionadas las sácadas válidas, las sácadas son ensambladas en una matriz S de vectores verticales, en la cual cada una de n las columnas se corresponde con una observación, compuesta por las m filas correspondientes a los puntos de cada observación [2].

Es importante señalar que no se incluyen en dicho ensamblaje las sácadas que presentan alguno de los siguientes problemas:

- Maracadas como no válidas.
- Solamente se ha detectado un punto de inicio o fin.
- Tienen latecias inferiores a 100ms.
- La fijación posterior a esta presenta aberraciones.

En la siguiente Figura se muestra un emsamblaje de sácadas de un sujeto sano, listas para aplicar ICA.

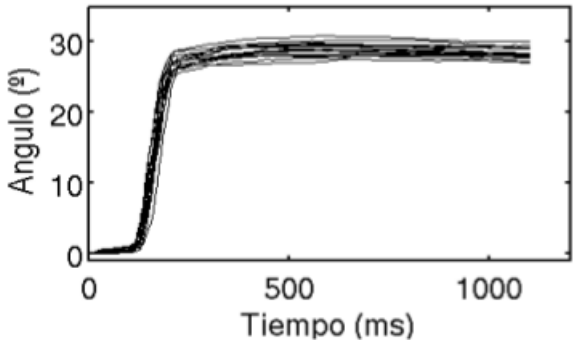


Figura 4: Ensamblaje de sácadas para aplicar ICA. Tomado de [2]

1.4. Tecnologías y herramientas disponibles

En el desarrollo de esta aplicación es vital el análisis de las tecnologías y herramientas disponibles, ya que las mismas guiarán en gran medida el proceso de desarrollo del software. En el caso particular de este producto informático, la elección de algunas de las tecnologías no constituye una opción para el desarrollador pues la plataforma sobre la cual se apoya dicho producto ya está desarrollada y los mismos deben ser completamente compatibles. En los siguientes

epígrafes se hará una revisión de las tecnologías y herramientas disponibles para desarrollar la herramienta propuesta.

1.4.1. Lenguaje de programación Python

Un lenguaje de programación no es más que un idioma artificial usado para expresar operaciones que pueden ser llevadas a cabo por máquinas computadoras. Los mismos pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Cada lenguaje de programación cuenta con cierto nivel de expresividad, lo que influye en gran manera en los distintos escenarios donde los mismos puedan aplicarse. Dicha expresividad, viene dada por los distintos paradigmas que los mismos implementan.

Entre los lenguajes de programación existentes, se encuentra Python, un lenguaje cuya popularidad y usabilidad, es cada día más notable. Multiparadigma, de una sintaxis de legibilidad admirable, Python se ha convertido en una opción importante para una inmensa comunidad de desarrolladores por su innegable adaptabilidad para resolver un gran número de problemas.

Python posee, entre otras, las siguientes características que lo hacen elegible para desarrollar la herramienta propuesta [14]:

- Sintaxis clara y legible, usando la identación como estructura en bloques.
- Comunidad libre y activa que proporciona un buen soporte a las tecnologías basadas en el lenguaje.
- Fácil de aprender y de enseñar, debido a su parecido con el pseudocódigo y la limpieza de su sintaxis.
- Portable a la mayoría de los sistemas operativos de escritorio disponibles en el mercado.
- Buen soporte de la comunidad científica a través de las bibliotecas NumPy y SciPy.

Es importante señalar que la herramienta para aplicar ICA sobre registros oculares sacádicos se construye como un *plugin* para toda una plataforma desarrollada en este lenguaje, por lo

que se hace obligatorio por cuestiones objetivas, que la misma sea programada en el mismo lenguaje de programación.

1.4.2. Framework Qt

En el desarrollo de interfaces gráficas de usuario se encuentran un gran número de *frameworks* compatibles con tecnologías seleccionadas, entre estos se encuentran GTK, TK y QT.

De este conjunto de *frameworks*, Qt es tecnológicamente superior al resto. El mismo consiste en un marco de trabajo de plataforma cruzada para la creación de aplicaciones de escritorio, empotradas y móviles. La misma posee una interfaz neutral, independiente de la plataforma para prácticamente todo. En la Figura 5 se muestra su alcance a través de su arquitectura.

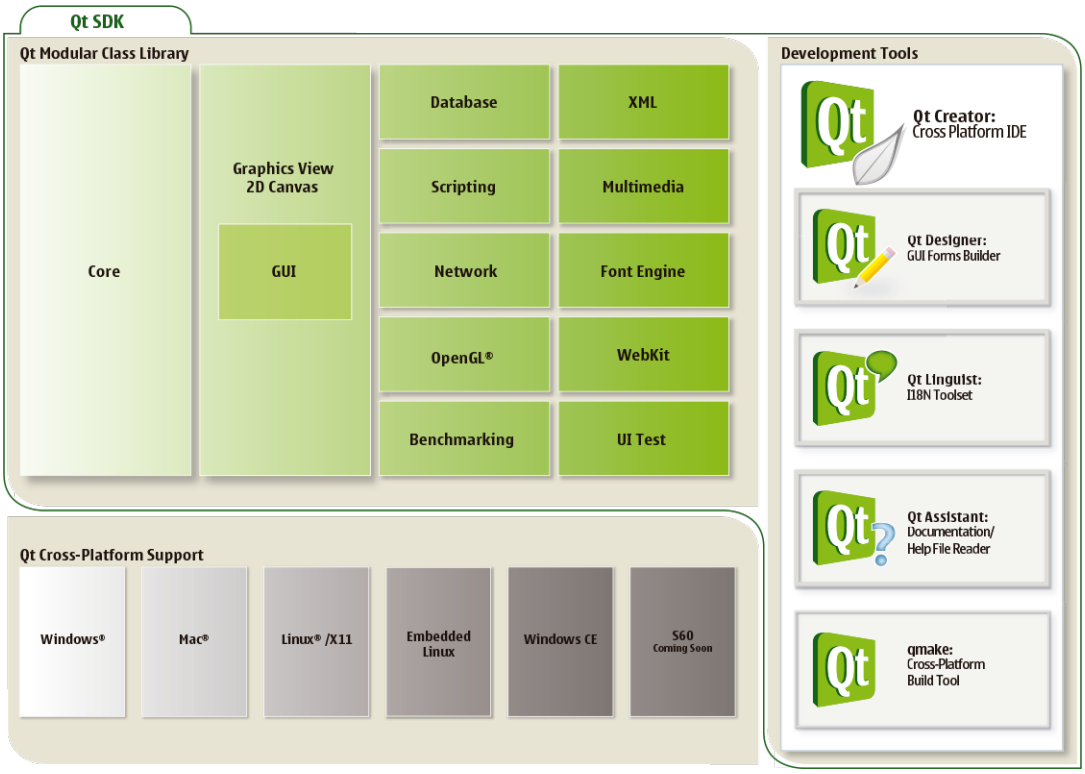


Figura 5: Arquitectura del Framework Qt

La característica de plataforma cruzada que provee el framework potencia el rendimiento de las aplicaciones y es capaz de adaptarse a la cultura visual de cada sistema operativo en los

que se compilan estas. Además cuenta con un API [3] intuitiva y excelentemente documentada, disminuyendo la curva de aprendizaje y aumentando la productividad de los desarrolladores. Posee un modelo de componentes sencillo y poderoso que facilita la creación de widgets [4] a la medida.

1.4.3. PyQt

El desarrollo de Qt se ha realizado fundamentalmente en C++, pero se han creado diversos envoltorios para permitir que se utilicen desde otros lenguajes de programación. Los envoltorios no son más que mecanismos utilizados por los lenguajes de programación para reutilizar código desarrollado en otros lenguajes. En Python se pueden encontrar dos de estos de probada calidad: PyQt y PySide.

La combinación de Python Qt mediante PyQt hace posible desarrollar los programas existentes en cualquier plataforma y ejecutarlos en las cualquier plataforma sin cambio alguno. Además, no se requiere compilación pues Python es un lenguaje interpretado y no se requieren cambios en el código fuente para adaptarlo a los diferentes sistemas operativos gracias a la abstracción de Qt de los detalles de cada plataforma. PyQt es usado para escribir todo tipo de aplicaciones GUI, desde simples aplicaciones, hasta herramientas de visualización usadas por científicos e ingenieros [15].

1.4.4. Matplotlib

Python es un lenguaje interpretado que cuenta en su núcleo con un gran número de funcionalidades y un poderoso aspecto modular que permite expandir las funcionalidades del mismo con módulos externos que permitan incluir más potencialidades en el mismo.

Los módulos están organizados dentro de paquetes. Matplotlib es una colección estructurada de módulos con el mismo propósito: desarrollar gráficos en 2D y 3D con alta calidad y productividad. La misma permite incluir sus herramientas dentro de aplicaciones desarrolladas con los diferentes entornos de interfaces gráficas de usuario usados en Python (GTK+, Qt, entre

[3] Interfaz de Programación de Aplicaciones del inglés *Application Programming Interface*
[4] Componente de interfaz visual

otros) y permite además salvar las gráficas obtenidas en los formatos más conocidos (PNG, PS, entre otros)

Matplotlib tiene dos propósitos fundamentales. Primero, se pueden incluir sus gráficas en aplicaciones de tipo, ahorrando al desarrollador horas de trabajo y produciendo a su vez imágenes de gran calidad con un buen rendimiento. Además, se puede usar de forma muy simple en un intérprete de Python como por ejemplo IPython con el fin de visualizar resultados de determinada prueba o experimento sin necesidad de crear una aplicación para esto.

En la herramienta informática propuesta tiene una vital importancia pues se realizan un gran número de tareas de este tipo y aunque en el NSEog se usa un componente gráfico implementado para etiquetar los movimientos oculares en un estudio, y dicho componente realice esta tarea con gran calidad y buen rendimiento, el mismo no está diseñado para adaptarse a otras tareas dentro del software

1.4.5. NumPy y SciPy

Los lenguajes de programación C, C++ y Fortran tienen muchos beneficios, pero no son completamente interactivos y son considerados muy complejos por muchos. Las alternativas comerciales más comunes, son, entre otras, Matlab, Maple, y Mathematica. Estos productos poseen un poderoso lenguaje pero poseen muchas limitaciones que los separan de lenguajes de propósito general [16].

En Python, a diferencia de otros lenguajes de programación donde existen un gran número de bibliotecas orientadas al cálculo númerico y científico, el desarrollo de este tipo de herramientas se ha centrado en solo dos: NumPy y SciPy. Esto, si lugar a dudas, constituye un aspecto positivo para la comunidad científica que usa este lenguaje.

Ambas bibliotecas son de código abierto liberadas bajo licencia BSD[5] que proveen rutinas matemáticas y numéricas comunes en la forma de funciones precompiladas eficientes. Desde el momento de su creación, las mismas han sido crecientes en funcionalidades, estando cada día más cerca de competidores comerciales como Matlab.

[5] Desarrollo de Software de Berkeley, en inglés *Berkeley Software Development*

Numpy (del inglés *Numerical Python*) es una biblioteca de código abierto para computación científica. La misma permite trabajar con arreglos y matrices en una forma natural. Esta librería contiene una larga lista de funciones matemáticas ampliamente usadas incluyendo algunas para álgebra lineal, transformaciones de Fourier, y rutinas de generación de números aleatorios, entre otros [16].

SciPy es una colección de algoritmos matemáticos y funciones construidas como una extensión de NumPy. La misma añade un poder significativo al trabajo científico con Python, facilitando a los usuarios comandos y clases de alto nivel para la manipulación y visualización de datos [17].

1.4.6. Python MDP

El uso de Python en la neurociencia computacional ha tenido un desarrollo creciente en los últimos tiempos. La maduración de las dos librerías antes mencionadas (NumPy y SciPy), dan acceso a una larga colección de funciones científicas que compiten con alternativas comerciales como *Matlab* [18].

MDP[6] es una biblioteca de algoritmos ampliamente usados en el procesamiento de datos que cuenta con la posibilidad de combinar estos con el fin de construir software más complejo para procesar datos. La misma ha sido diseñada como un *framework* para científicos que sirva para programar procesamiento de datos.

Desde la perspectiva del usuario, MDP consiste en una colección de módulos o unidades, que procesan datos. Este contienen, entre otros, algoritmos para aprendizaje supervisado y no supervisado, clasificación y análisis de componentes principales e independientes [18].

MDP es distribuido bajo licencia BSD y desde su primera publicación en 2004 ha registrado decenas de miles de descargas lo que lo convierte en un producto ampliamente usado por la comunidad científica de Python. Es importante señalar que esta biblioteca cuenta con dependencias mínimas ya que solamente requiere la instalación de NumPy [18]. Es además extensible pues cuenta con una arquitectura bastante flexible para realizar clases que dependan

[6] Herramienta modular para el procesamiento de datos (del inglés *The Modular toolkit for Data Processing*).

de las ya implementadas en dicha biblioteca de clases y así incrementar sus funcionalidades con algoritmos y procedimientos que hasta el momento la misma no contenga.

1.4.7. Entornos de Desarrollo Integrado (IDE)

En la actualidad, para desarrollar software que cumpla con las exigencias actuales de la mayoría de los clientes es preciso contar con modernas herramientas que faciliten el trabajo de codificar, construir, desplegar y probar. En la mayoría de los casos estas herramientas se encuentran en un solo paquete denominado Entorno de Desarrollo Integrado (IDE). Los mismos no son más que una herramienta que integra un grupo de tecnologías para ponerlas en función del desarrollador.

Los IDEs de mayor calidad en la actualidad constituyen plataformas que extienden sus funcionalidades a través de *plugins*. Los de mayor relevancia son:

Geany: Es un editor de texto ligero basado en Scintilla con características básicas de entorno de desarrollo integrado (IDE). Está disponible para distintos sistemas operativos, como GNU/Linux, Mac OS X, BSD, Solaris y Microsoft Windows. Es distribuido como software libre bajo la Licencia Pública General de GNU. Tiene soporte para muchos lenguajes de programación distintos.

Eclipse: Creado inicialmente como entorno de desarrollo para el lenguaje Java, se ha convertido en una rica plataforma de desarrollo, donde ya no solo se realizan aplicaciones de programación, sino que se ha extendido su uso a otros escenarios como la oficina, el diseño, etc. Cuenta con soporte para un gran número de lenguajes entre los que se encuentra Python usando Pydev. Cuenta con binarios para las mayores plataformas de escritorio.

PyCharm: De reciente desarrollo, este IDE se convierte en una poderosa alternativa para los desarrolladores que utilicen el lenguaje de programación Python. El mismo, desarrollado por JetBrain, cuenta con un gran número de tecnologías y funcionalidades integradas que sin duda mejoran de forma notable el proceso de desarrollo de software. Es importante destacar que el mismo tiene como desventaja su carácter privativo.

En el proceso de desarrollo de este producto informático se utilizará el IDE Eclipse por las ventajas que brinda al programador en cuanto al uso del lenguaje de programación.

1.5. Metodologías de Desarrollo

Desarrollar un buen software depende de un sin número de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el 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. [19]

Dentro del desarrollo de software y a la altiva necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para nuestros clientes, se generan grandes cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que, unas se adaptan mejor que otras, al contexto del proyecto brindando mejores ventajas.

Las metodologías tradicionales (formales) se focalizan en documentación, planificación y procesos. (Plantillas, técnicas de administración y revisiones) mientras que las metodologías ágiles se centran más en el proceso de desarrollo y la colaboración con el cliente. Además, estas últimas se centran en el concepto de que la capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan. En la siguiente tabla se pueden observar las principales diferencias entre las metodologías ágiles y las tradicionales.

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Tabla 1: Diferencias entre las metodologías tradicionales y ágiles

Entre las principales metodologías tradicionales se encuentran los ya tan conocidos RUP[7] y MSF[8]. Por otra parte, entre los métodos ágiles mas conocidos se encuentran ICONIX, AUP[9], SCRUM y XP[10] que nacen como respuesta a los problemas de las metodologías tradicionales.

1.5.1. Metodologías ágiles

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "ágil "aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria

[7] Proceso Unificado de Rational (del inglés *Rational Unified Process*)
[8] *Microsoft Solution Framework*
[9] *Agil Unified Process*
[10] Programación Extrema (del inglés *Extreme Programming*)

del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas [20].

Como resultado de esta nueva teoría se crea un **Manifiesto Ágil** [3] cuyas principales ideas son:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

1.5.2. Programación Extrema (XP)

La programación extrema es una metodología de desarrollo ligera (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay al rededor de la programación.

Los objetivos de XP son muy simples: primeramente se encuentra la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo

en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.[21]

En cotraste con las metodologías tradicionales, XP cuenta, entre otras, con las siguientes características fundamentales [22]:

- Esta orientada a quien desarrolla y usa el software.
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.

Uno de los problemas del desarrollo de software, es la casi imposibilidad de precisar los verdaderos requerimientos del sistema al inicio del desarrollo. Por más preciso que sea el proceso de captura de requerimientos y más detallado su análisis y diseño inicial, en cualquier etapa del ciclo de vida siempre van a aparecer necesidades imprescindibles.

En la siguiente Figura se muestra el costo del cambio en relación al tiempo. Tradicionalmente, entre más tarde aparezca la necesidad de un cambio, el costo de implementación de esta se elevará exponencialmente [22].

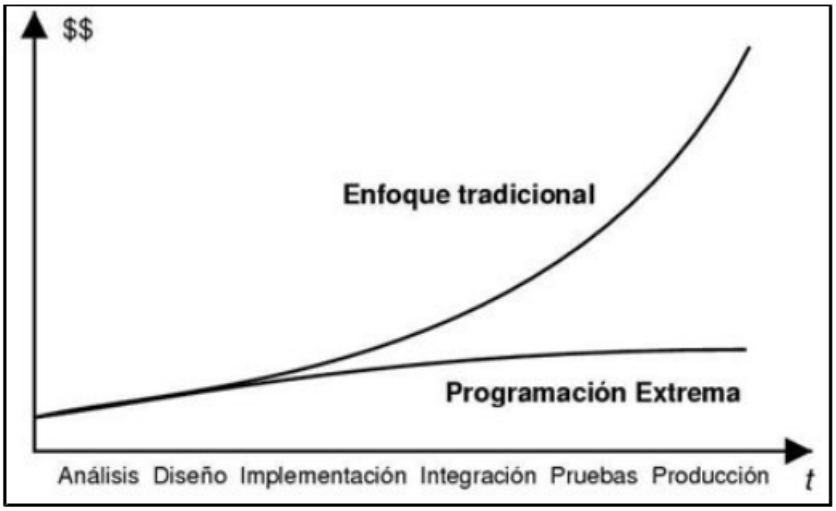


Figura 6: Costo del cambio durante el ciclo de vida. Tomado de [3]

Si se aplica correctamente, XP mantiene dicho costo en un nivel prácticamente independiente con respecto a la etapa del ciclo de vida, como puede observarse en la curva inferior de la Figura antes mostrada.

La programación extrema puede describirse someramente por los siguientes puntos:

1. Empieza en pequeño y añade funcionalidad con retroalimentación continua.
2. El manejo del cambio se convierte en parte sustantiva del proceso.
3. El costo del cambio no depende de la fase o etapa.
4. No introduce funcionalidades antes de que sean necesarias.
5. El cliente y el usuario se convierten en miembros del mismo equipo.

Historias de Usuario

Las historias de usuario en XP no so más que la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Un ejemplo de éstas se puede ver en la Figura 2

Historia de Usuario	
No.: Id	**Nombre:** Nombre de la historia
Usuarios: Todos	
Prioridad en el negocio: Alta, Media, Baja	**Nivel de Complejidad:** Alta, Media, Baja
Estimación: En semanas laborables	**Iteración asignada:** 1
Descripción: Breve descripción de la historia	
Información Adicional (Observaciones): Aquí se pone a qué requisitos da cumplimiento la historia entre otras consideraciones	

Tabla 2: Ejemplo de una Historia de Usuario

Tareas

Las historias a su vez se subdividen en tareas concretas que permiten dividir una funcionalidad del negocio en pequeñas unidades estimables por los programadores. Al completar las mismas, se puede saber el por ciento de trabajo realizado en cada historia de usuario y llevar las trazas de la misma [23]. Además, pueden existir también tareas independientes de las historias de usuario que den cumplimiento a funcionalidades específicas. Las tareas son escritas por los programadores y se puede observar un ejemplo de éstas en la Figura 23.

Tarea	
No.: ♯	**Número de la HU:** ♯
Nombre de la tarea: Nombre de la tarea	
Tipo de tarea: Tipo de tarea	**Estimación:** Días de duración de la tarea
Fecha de inicio: Fecha de inicio	**Fecha de Fin:** Fecha de fin
Programador responsable: Nombre del programador responsable	
Descripción: Descripción de la tarea	

Tabla 3: Ejemplo de Tareas

Pruebas de Aceptación

El otro artefacto usado por XP para documentar el proceso son las Pruebas de Aceptación como las que se muestran en la Figura 27 con las cuales se describen los escenarios donde se ponen a prueba funcionalidades del sistema desplegado [23]. Este tipo de pruebas determina cuando las historias de usuario han sido completadas.

Caso de Prueba de Aceptación	
Código: Código de la prueba	**Historia de usuario:** ♯
Nombre: Nombre de la prueba	
Descripción: Descripción breve de la prueba	
Condiciones de ejecución: Pre-condiciones para ejecutar la prueba	
Entrada/Pasos de ejecución: Descripción del caso de prueba paso a paso	
Resultado esperado: Resultado esperado de la prueba	
Evaluación de la prueba: Prueba satisfactoria o no	

Tabla 4: Ejemplo de pruebas de aceptación

Roles En todo equipo de desarrollo debe existir cierta jerarquía para poder funcionar de forma correcta. En XP existen un grupo de roles que dan solución a este problema. Los mismos, de acuerdo con la propuesta original de Beck [20, 23] son:

Programador: Escribe las pruebas unitarias y produce el código del sistema.

Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

Encargado de pruebas (Tester): Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Encargado de seguimiento (Tracker) Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

Entrenador (Coach): Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.

Gestor (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Ciclo de vida El ciclo de vida ideal de XP consiste de seis fases [23]: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. Las mismas se detallan a continuación:

Exploración: El cliente define las historias de usuario que desea incluir en la primera liberación del producto. Al mismo tiempo los miembros del equipo se familiarizan con las

herramientas, tecnologías y prácticas que usarán durante el proyecto. Se probarán las tecnologías que se usarán y las posibilidades de arquitectura para el sistema, construyendo un prototipo del sistema. La fase de exploración puede tomar desde algunas semanas hasta meses, dependiendo de cuan familiar es la tecnología para los programadores.

Planeación: En esta fase se establece el orden de prioridad para las historias de usuario y un acuerdo entre el cliente y el equipo de desarrollo sobre el contenido del primer lanzamiento del sistema. Los programadores estiman la cantidad de esfuerzo requerido por cada historia de usuario y se llega a un acuerdo en el cronograma del proyecto. Esta fase toma usualmente varios días.

Iteraciones a liberar: Esta fase incluye varias iteraciones del sistema antes del primer lanzamiento. El cronograma establecido en la fase de planeación se separa en un número de iteraciones que tomarán de una a cuatro semanas de implementar. La primera iteración crea un sistema con la arquitectura de todo el sistema. Esto se logra seleccionando las historias que fuerzan a construir la estructura de todo el sistema. El cliente decide las historias que serán seleccionadas para cada iteración. Las pruebas funcionales diseñadas por el cliente deben correr al final de cada iteración. Al final de la última iteración el sistema está listo para producción

Producción: La fase de producción requiere una serie extra de pruebas y el chequeo del funcionamiento del sistema antes de que pueda ser liberado al cliente. Además, en esta fase aún se pueden realizar nuevas decisiones y cambios si se incluyen en el lanzamiento actual. Durante esta fase las iteraciones se deben apresurar desde tres a una semana. Las ideas y sugerencias pospuestas son documentadas para ser implementadas durante la fase de mantenimiento.

Mantenimiento: Después del primer lanzamiento de producción para el uso por el cliente, el proyecto XP debe mantener el sistema corriendo y además producir nuevas iteraciones. Para lograr esto, la fase de mantenimiento requiere un esfuerzo en las tareas de soporte del cliente. De esa manera, la velocidad de desarrollo se puede desacelerar después que el sistema entre en producción. Esta fase puede requerir incorporar nuevas personas al

equipo de desarrollo y realizar cambios en la estructura actual del mismo.

Muerte: La fase final o muerte del proceso se encuentra cercana cuando el cliente no tiene más historias para implementar. Esto requiere que el sistema satisfaga otras necesidades del cliente como son rendimiento y confiabilidad. Este es el momento en el que la documentación del sistema es finalmente escrita y no se realizan más cambios al diseño o la arquitectura del proyecto. La muerte del proyecto puede ocurrir también cuando el sistema no entrega el resultado esperado o si se hace muy caro seguir su desarrollo.

En los Capítulos 2 y 3 se mostrará en detalle la implementación de la metodología XP en el desarrollo de la herramienta informática propuesta.

1.6. Conclusiones del capítulo

En este capítulo se han analizado los principales aspectos teóricos que sustentan esta investigación y luego de realizar este proceso se pueden arribar a las siguientes conclusiones.

Según la bibliografía consultada el Análisis de Componentes Independientes es adecuado para separar las señales de los centros neuronales que generan los movimientos oculares sacádicos.

La selección del lenguaje de programación Python y las bibliotecas asociadas al mismo, proveen de un ambiente acorde a las políticas planteadas en el capítulo. Además de proporcionar un balance investigativo-productivo sin paralelo en otros lenguajes.

La metodología de desarrollo XP brinda un marco de trabajo adecuado para la implementación de la solución propuesta, acorde también a las políticas trazadas en el capítulo.

2

Exploración y Planificación

En este capítulo se detallará la aplicación de las dos primeras fases de la metodología XP en la solución propuesta. De la misma forma se seleccionarán las tecnologías y herramientas a utilizar. El objetivo principal de el mismo es describir las principales características del sistema, exponiéndose de manera clara los requerimientos a cumplir por el mismo que dan paso a las historias de usuario.

2.1. Exploración

Los requerimientos del sistema representan las tareas que el mismo debe ser capaz de realizar para funcionar correctamente. Es por ello que la elaboración detallada de estos, para ser utilizados como base de las futuras pruebas sea de vital importancia en el proceso de desarrollo de la aplicación.

2.1.1. Requerimientos funcionales

RF1. Pre-procesar las sácadas

RF1.1: Seleccionar manualmente sácadas a procesar.

RF1.2: Filtrar las sácadas según varios criterios de forma automática.

RF1.3: Visualizar las sácadas seleccionadas.

RF1.4: Alinear las sácadas seleccionadas según varios criterios.

RF1.5: Construir la matriz de mezclas.

RF2. Aplicar ICA

RF2.1: Seleccionar el algoritmo de ICA a utilizar.

RF2.2: Establecer los parámetros requeridos por el algoritmo seleccionado.

RF2.3: Aplicar el algoritmo seleccionado.

RF2.4: Mostrar los resultados del ICA.

RF3. Exportar los resultados del ICA.

RF3.1: Seleccionar el formato a utilizar.

RF3.2: Exportar los datos al formato seleccionado.

RF3.3: Guardar los datos obtenidos dentro de un estudio (Fichero eogs).

2.1.2. Requerimientos no funcionales

Los requisitos no funcionales son aquellos que constituyen propiedades o cualidades que el producto informático desarrollado debe poseer. Estas características son las que lo hacen más atractivo, usable, rápido y confiable. Es importante destacar que muchos de estos requisitos se obtienen de la plataforma NSEog pues la herramienta propuesta está estrechamente relacionada con la misma.

- **Apariencia o Interfaz de Usuario**

RNF1: Usar solamente bibliotecas que permitan el uso de la cultura de cada sistema.

RNF2: Usar interfaces que se adapten a la resolución de pantalla del usuario.

RNF3: Usar colores, íconos y componentes agradables para el usuario.

- **Usabilidad**

RNF4: Incluir atajos de teclado para facilitar el uso de la aplicación por parte del usuario.

RNF5: No utilizar barras de menús ni otros elementos diseñados específicamente para plataformas de escritorio de forma que se facilite la transición de la herramienta a plataformas móviles.

- **Ayuda y Documentación**

RNF6: Crear un manual de usuario de la herramienta para facilitar el trabajo de los usuarios con la misma.

RNF7: Documentar los módulos de la aplicación utilizando un generador de documentación.

- **Portabilidad**

RNF8: La aplicación debe poder utilizarse en las plataformas Windows, Linux y Mac Os.

- **Rendimiento**

RNF9: la plataforma debe poderse ejecutar en el equipamiento actual que posee el CIRAH con un rendimiento adecuado.

2.1.3. Personas relacionadas con la aplicación

Las personas relacionadas con la aplicación son aquellas que obtienen algún beneficio de la misma. Las aplicaciónes orientadas al procesamiento de señales biomédicas se orientan generalmente a especialistas de la misma rama.

En el caso de la herramienta propuesta, la persona relacionada es el **especialista**. Esta persona debe tener los conocimientos suficientes sobre la aplicación de Análisis de Componentes Independientes a registros oculares sacádicos, para ser capaz de explotar esta herramienta de la forma correcta. Este rol es tomado generalmente por el personal médico que conduce alguna investigación sobre el tema.

2.1.4. Historias de usuario

Como se analizó en el capítulo anterior, La historia es la unidad de funcionalidad en un proyecto XP. En lo adelante se muestran las historias de usuario relacionadas con la herramienta informática propuesta.

Historia de Usuario	
No.: 1	**Nombre:** Crear prototipo no funcional del plugin
Usuarios: Todos	
Prioridad en el negocio: Alta	**Nivel de Complejidad:** Alta
Estimación: 1 semana	**Iteración asignada:** 1
Descripción: Esta aplicación constituirá una base para comenzar la implementación sobre ella.	
Información Adicional (Observaciones): Esta aplicación base contendrá los componentes y las clases necesarios con una estructura definida para trabajar pero sin ninguna funcionalidad,	

Tabla 5: Historia de Usuario No.1 "Crear prototipo no funcional del plugin"

Historia de Usuario	
No.: 2	**Nombre:** Pre-procesar las sácadas para realizar ICA
Usuarios: Todos	
Prioridad en el negocio: Alta	**Nivel de Complejidad:** Alta
Estimación: 2 semanas	**Iteración asignada:** 1
Descripción: El especialista debe ser capaz de adicionar y eliminar restricciones con sus respectivos valores que permitan filtrar las sácadas dejando solo las adecuadas para realizar ICA sobre ellas, además deberá poder realizar esta misma operación de forma visual	
Información Adicional (Observaciones): Esto da cumplimiento al requisito funcional 1 "Pre-procesar las sácadas para realizar ICA"	

Tabla 6: Historia de Usuario No.2 "Pre-procesar las sácadas para realizar ICA"

Historia de Usuario	
No.: 3	**Nombre:** Aplicar ICA
Usuarios: Todos	
Prioridad en el negocio: Alta	**Nivel de Complejidad:** Alta
Estimación: 3 semanas	**Iteración asignada:** 2
Descripción: El especialista debe ser capaz de seleccionar el algoritmo a utilizar (Informax, Jade, FastICA), y establecer los parámetros necesarios para el funcionamiento del mismo.	
Información Adicional (Observaciones): Esto da cumplimiento al requisito funcional 2 "Aplicar ICA."	

Tabla 7: Historia de Usuario No.3 "Aplicar ICA "

Historia de Usuario	
No.: 4	**Nombre:** Exportar los datos del ICA
Usuarios: Todos	
Prioridad en el negocio: Alta	**Nivel de Complejidad:** Alta
Estimación: 3 semanas	**Iteración asignada:** 3
Descripción: El especialista debe ser capaz exportar los datos obtenidos del análisis así como guardar dichos resultados dentro del mismo estudio en que está trabajando.	
Información Adicional (Observaciones): Esto da cumplimiento al requisito funcional 3 "Exportar los datos del ICA."	

Tabla 8: Historia de Usuario No.4 "Exportar los datos del ICA"

Cabe destacar que en la estimación del tiempo de duración de cada historia se utilizó como unidad la semana laborable que consiste en 5 días, cada uno con 8 horas laborables. Todo esto se realiza con la práctica de la metodología utilizada que afirma que se debe trabajar como máximo 40 horas a la semana.

2.1.5. Selección de tecnologías y herramientas

Primeramente se toma como premisa que mientras sea posible, se deberán seleccionar herramientas de software libre o de código abierto. Esto se realiza para apoyar la sostenibilidad del sistema y apoyar la política de migración e independencia tecnológica existente en el país.

Debido a que la herramienta propuesta surge como una extensión de una plataforma de procesamiento de registros ya existentes, su desarrollo se realiza en el **lenguaje de programación** Python, que es en la que la misma está desarrollada. Este lenguaje es el que mejor cumple con los estándares necesarios por el NSEog, además de que su sintaxis se encuentra estandarizada internacionalmente, y existen intérpretes de gran calidad y para todos los sistemas operativos.

Una vez seleccionado el lenguaje se seleccionan la **bibliotecas** necesarias para desarrollar la aplicación. Se selecciona Matplotlib para la visualización de los datos por su calidad en gráficos 2D, su gran variedad de opciones de edición y personalización y su perfecta integración con PyQt5, el framework que se utiliza en todo el NSEog para las interfaces visuales. También se usan Numpy, Scipi y MDP para utilizar los algoritmos y las estructuras necesarias que estos brindan.

Cómo **Entorno de Desarrollo Integrado** se utilizará Geany pues el mismo cuenta con un gran soporte para el lenguaje de programación Python.

Como **Herramienta CASE** se selecciona Visual Paradigm pues su calidad es superior al resto de sus competidores. Para la modelación se utilizará la versión desarrollada por la comunidad.

2.2. Planificación

2.2.1. Iteraciones

Para la selección del trabajo de cada iteración se tuvo en cuenta que este no tuviera más de 4 semanas de desarrollo, siguiendo las prácticas de la metodología seleccionada. En la tabla 9 se muestra la distribución de las Historias de Usuario por cada iteración.

Distribución de las historias de usuario por iteración		
Iteraciones	Orden de las historias de usuario a implementar	Tiempo
1	1. Crear prototipo no funcional del plugin.	3 semanas
	2. Pre-procesar las sácadas para realizar ICA.	
2	3. Aplicar ICA.	3 semanas
3	4. Exportar los datos del ICA	3 semanas.

Tabla 9: Distribución de las historias de usuario por iteración

2.2.2. Plan de Entregas

El plan de entregas es el compromiso final del equipo de trabajo con los clientes. Esta es una cuestión de vital importancia para ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y la moral de los implicados.

La estimación es uno de los más complicados temas de la metodología XP, y es por ello que resulta de vital importancia tener bien claros los requerimientos acordados con el cliente, el estilo de trabajo del equipo de desarrollo y el tiempo con el que dispone el cliente para tener en sus manos la solución. El cronograma de entregas del producto informático propuesto se expone en la tabla 10.

Plan de Entregas		
1^{ra} **Iteración**	2^{da} **Iteración**	3^{ra} **Iteración**
11 de Abril 2014	2 de Mayo 2014	23 de Mayo 2014

Tabla 10: Cronograma de Entregas

2.3. Conclusiones del capítulo

En este capítulo se completaron de manera satisfactoria las dos primeras fases del ciclo de vida de la solución propuesta: los artefactos de las Historias de Usuario, el plan de iteraciones y el de las entregas, los cuales fueron detallados de forma clara y concisa.

Importante es destacar que separar el proceso en iteraciones permitió tener un mayor control sobre el desarrollo del producto, logrando así establecer un calendario de entrega acorde a las exigencias del cliente.

3

Implementación y prueba

XP no define un conjunto de artefactos a utilizar en la solución que utilice dicha metodología. La misma deja en manos del equipo de desarrollo, la determinación del uso de tantos diagramas UML como sean necesarios, para que el proceso de desarrollo sea más simple y compresible. El presente capítulo hace alusión al diseño de la herramienta, los diagramas usados, las tareas generadas por cada historia de usuario, el proceso de pruebas y por último la valoración de sostenibilidad del producto informático propuesto. En los siguientes epígrafes se tratan con mayor profundidad cada uno de estos aspectos.

3.1. Arquitectura de la plataforma NSEog

La arquitectura de la plataforma de procesamientos NSEog presenta tres tipos principales de componentes: las bibliotecas, los plugins y las aplicaciones. La herramienta informática propuesta constituye uno de los plugins que esta plataforma carga para aumentar sus funcionalidades. En la Figura 3.1 se puede visualizar dicha arquitectura y el lugar que ocupa el producto informático propuesto dentro de la aplicación. Además en la Figura 3.1 se muestra el diagrama de clases de la aplicación.

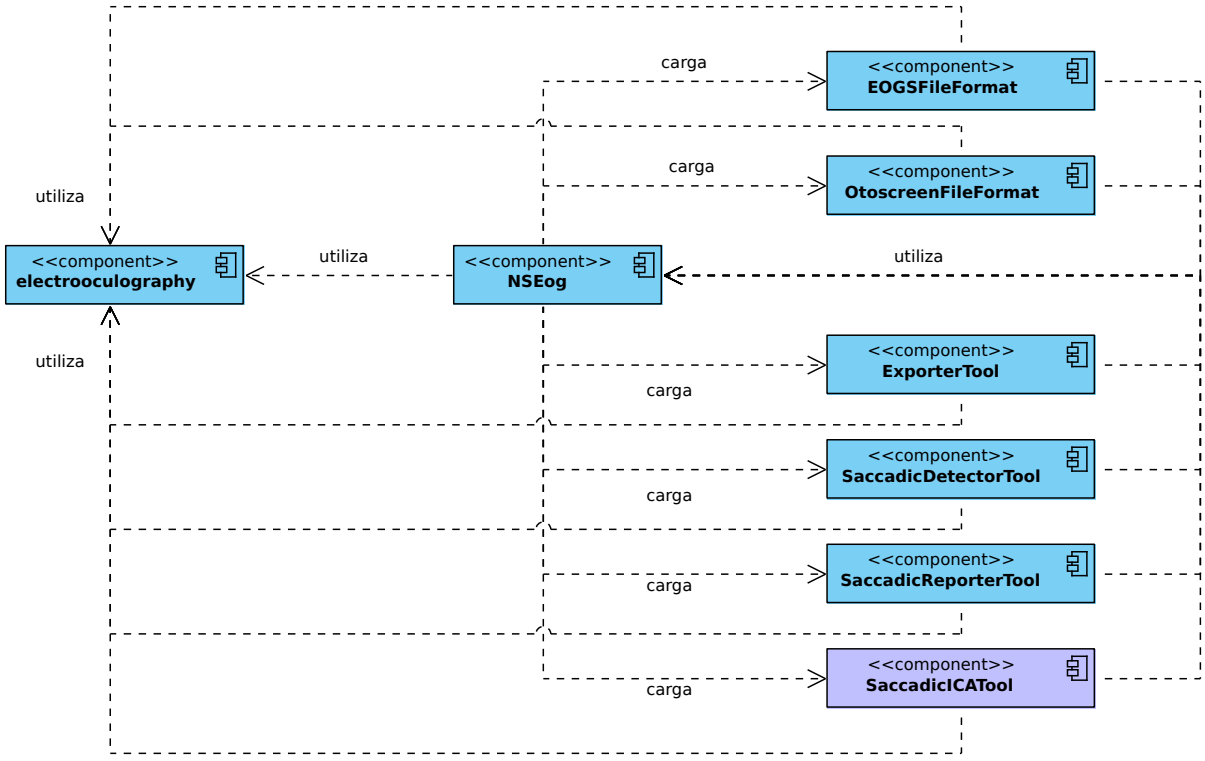


Figura 7: Diagrama de componentes de la plataforma NSEog

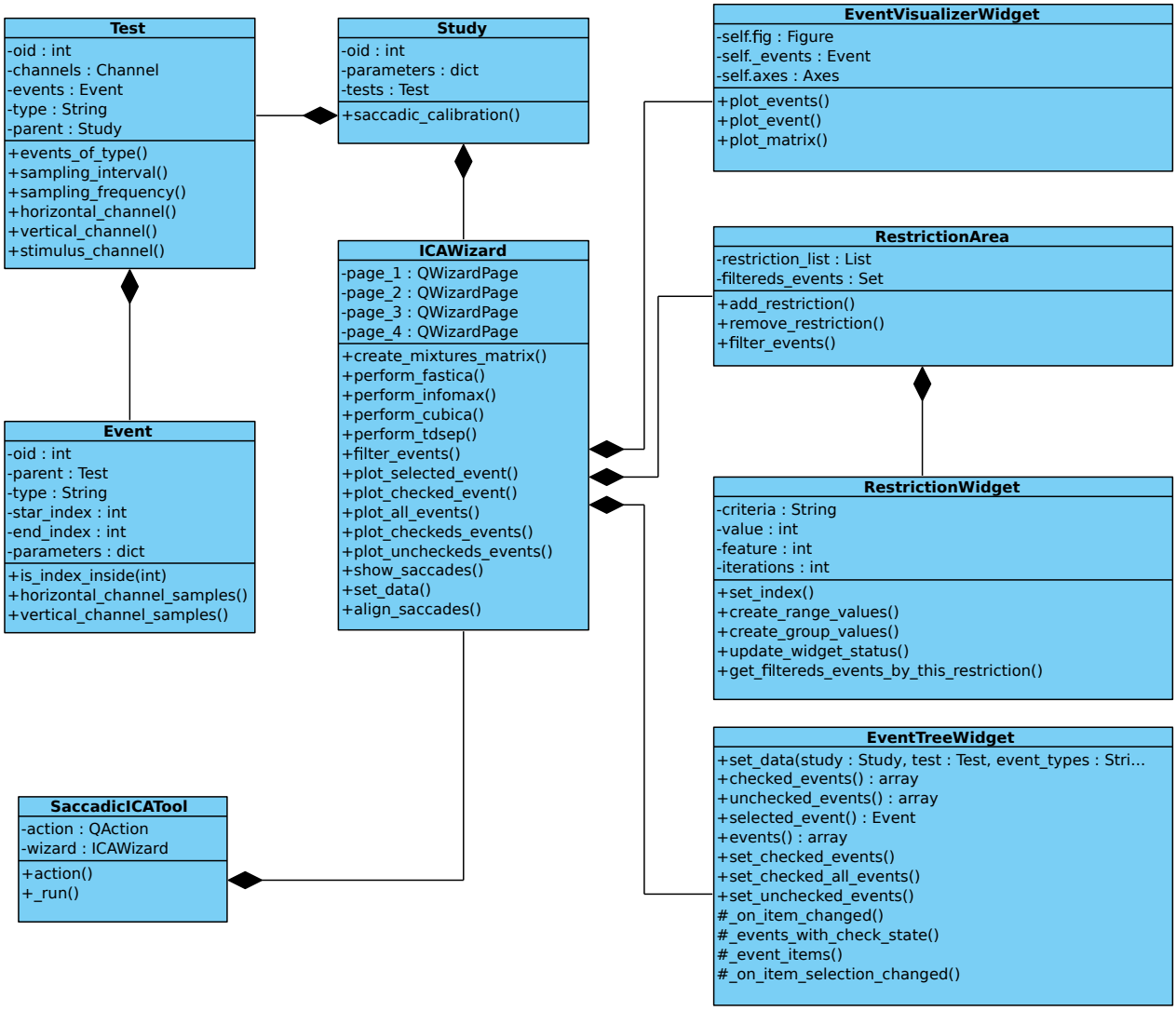


Figura 8: Diagrama UML de la herramienta propuesta

3.2. Implementación

La metodología escogida sugiere comenzar la implementación partiendo de una arquitectura lo más flexible posible para evitar grandes cambios en las próximas iteraciones y en los cambios que generalmente el cliente propone. Puesto que la solución propuesta posee un gran componente técnico, es necesario desde un inicio tener bien definido una estructura. Trazada la

misma se procede a la primera iteración.

3.2.1. Iteración 1

El principal objetivo de esta iteración es crear un prototipo no funcional del plugin para luego trabajar sobre el éste. Además tiene también como objetivo lograr una versión de la herramienta que permita el pre-procesamiento de las sácadas antes de realizar el análisis.

Para ello se trazan las siguientes 6 tareas:

1. **Tarea 1:** Crear prototipo no funcional del plugin
2. **Tarea 2:** Selección manual de sácadas
3. **Tarea 3:** Filtrado automático de sácadas
4. **Tarea 4:** Visualización de las sácadas seleccionadas
5. **Tarea 5:** Alineación de las sacádas según varios criterios
6. **Tarea 6:** Construcción de la matriz de mezclas

Tarea	
No.: 1	**Número de la HU:** 1
Nombre de la tarea: Crear prototipo no funcional del plugin	
Tipo de tarea: Diseño	**Estimación:** 3 días
Fecha de inicio: 27 de febrero del 2014	**Fecha de Fin:** 4 de marzo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Crear un prototipo no funcional del plugin para luego trabajar sobre el mismo. Construir todas las clases e interfaces, aunque sin ninguna funcionalidad. La herramienta se compone de una clase ICAWizard(QWizard) con 4 pantallas para en cada una realizar una funcionalidad. Además se debe construir un plugin de NSEog construyendo una herramienta ITool siguiendo las convenciones para el desarrollo de este tipo de componentes para dicha plataforma	

Tabla 11: Tarea 1 "Crear prototipo no funcional del plugin"

Tarea	
No.: 2	**Número de la HU:** 2
Nombre de la tarea: Selección manual de sácadas	
Tipo de tarea: Diseño	**Estimación:** 7 días
Fecha de inicio: 4 de marzo del 2014	**Fecha de Fin:** 13 de marzo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Implementar una clase *Restriction* para manejar los criterios para filtrar las sácadas y dejar solamente las sácadas seleccionadas. Además, se debe implementar una clase *RestrictionArea* que contenga las restricciones, dándole la posibilidad al usuario de adicionar y eliminar las mismas. Esta última clase también debe contener todos los métodos necesarios para el filtrado de los datos.	

Tabla 12: Tarea 2: "Selección manual de sácadas"

Tarea	
No.: 3	**Número de la HU:** 2
Nombre de la tarea: Filtrado automático de sácadas	
Tipo de tarea: Diseño	**Estimación:** 7 días
Fecha de inicio: 13 de marzo del 2014	**Fecha de Fin:** 24 de marzo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Filtrar las sácadas seleccionadas dentro de la *prueba* en la que se esté trabajando dentro del *estudio* abierto. Para ello se seleccionan varios criterios que dependan del conjunto de eventos como la media, la mediana y la desviación estándar, y otros criterios que dependan de los eventos por separado como un rango de Amplitud, Velocidad Máxima, Latencia y Duración. Después de realizar el filtrado, las saćadas seleccionadas para realizar el ICA deberán quedar contenidas dentro de un conjunto de eventos(*set()*), denominado *filtered_events*, el que luego se usará en la mayoría de los componentes del plugin	

Tabla 13: Tarea 3: "Filtrado automático de sácadas"

Tarea	
No.: 4	**Número de la HU:** 2
Nombre de la tarea: Visualización de las sácadas seleccionadas	
Tipo de tarea: Diseño	**Estimación:** 6 días
Fecha de inicio: 24 de marzo abril del 2014	**Fecha de Fin:** 1 de abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Implementar una clase *EventVisualizer*, usando las potencialidades de la biblioteca Matplotlib para visualizar los eventos. La misma debe ser capaz de incluirse dentro de cualquier interfaz de la herramienta y debe brindar las opciones de marcar con varios colores los eventos dependiendo de su estado dentro de la selección (marcadas o seleccionadas). Este componente se debe usar en tres de las pantallas del plugin	

Tabla 14: Tarea 4: "Visualización de las sácadas seleccionadas"

Tarea	
No.: 5	**Número de la HU:** 2
Nombre de la tarea: Alineación de las sacádas según varios criterios	
Tipo de tarea: Diseño	**Estimación:** 6 días
Fecha de inicio: 1 de abril del 2014	**Fecha de Fin:** 9 de abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Se deberán alinear todos los eventos previamente filtrados por el especialista según las restricciones establecidas. Para esto se establecerá un ancho de ventana determinado por el usuario y se establecerán dos criterios para realizar la alineación(Punto de Inicio del Estímulo y Punto de Inicio del Evento)	

Tabla 15: Tarea 5: "Alineación de las sacádas según varios criterios"

Tarea	
No.: 6	**Número de la HU:** 2
Nombre de la tarea: Construcción de la matriz de mezclas	
Tipo de tarea: Diseño	**Estimación:** 2 días
Fecha de inicio: 9 de abril del 2014	**Fecha de Fin:** 11 abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Construir una matriz con las sácadas previamente alineadas de cada fila de la matriz tendrá una sácada contenida	

Tabla 16: Tarea 6: "Construcción de la matriz de mezclas"

3.2.2. Iteración 2

En esta iteración se tiene como objetivo lograr una versión del plugin que permita aplicar el Análisis de Componentes Independientes a la matriz de sácadas previamente construida a partir de los criterios definidos por el especialista.

Para ello se trazaron las siguientes 4 tareas:

1. **Tarea 7:** Selección del algoritmo de ICA a utilizar
2. **Tarea 8** Captura de los parámetros requeridos por cada algoritmo
3. **Tarea 9:** Aplicación del Análisis de Componentes Independientes
4. **Tarea 10:** Visualización de los resultados del ICA realizado

Tarea	
No.: 7	**Número de la HU:** 3
Nombre de la tarea: Selección del algoritmo de ICA a utilizar	
Tipo de tarea: Diseño	**Estimación:** 2 días
Fecha de inicio: 11 de abril del 2014	**Fecha de Fin:** 14 abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Crear un mecanismo de selección del algoritmo usando un objeto *QCombobox* dentro de la tercera pantalla del plugin. Al seleccionar el mismo, este debe ajustar la interfaz para recibir los parámetros necesarios para que estos algoritmos funcionen.	

Tabla 17: Tarea 7: "Selección del algoritmo de ICA a utilizar"

Tarea	
No.: 8	**Número de la HU:** 3
Nombre de la tarea: Captura de los parámetros requeridos	
Tipo de tarea: Diseño	**Estimación:** 10 días
Fecha de inicio: 16 de abril del 2014	**Fecha de Fin:** 29 abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Crear un mecanismo mediante el cual se puedan entrar los datos del algoritmo seleccionado de forma dinámica. Según el algoritmo que se escoja, serán los campos de entrada contenidos en la aplicación.	

Tabla 18: Tarea 8: "Captura de los parámetros requeridos por cada algoritmo"

Tarea	
No.: 8	**Número de la HU:** 3
Nombre de la tarea: Aplicación del ICA	
Tipo de tarea: Diseño	**Estimación:** días
Fecha de inicio: 14 de abril del 2014	**Fecha de Fin:** 16 abril del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Después de haber seleccionado el algoritmo a utilzar se debe aplicar el mismo. Para esto, se debe hacer uso de la biblioteca Python-MDP para los algoritmos JADE, FastICA y CubICA y se deberá implementar el algoritmo InfomaxICA a partir de un algoritmo ya implementado en el lenguaje de programación Matlab.	

Tabla 19: Tarea 9: "Aplicación del Análisis de Componentes Independientes"

Tarea	
No.: 10	**Número de la HU:** 3
Nombre de la tarea: Visualización de los resultados	
Tipo de tarea: Diseño	**Estimación:** 4 días
Fecha de inicio: 29 de abril del 2014	**Fecha de Fin:** 2 de mayo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Visualizar los datos del ICA realizado utilizando el componente EventVisualizer previamente seleccionado. Se deberá escoger por parte del especialista que utiliza la herramienta sobre que prueba mostrar el resultado del ICA realizado.	

Tabla 20: Tarea 10: "Visualización de los resultados"

3.2.3. Iteración 3

En esta iteración se tiene como objetivo lograr una versión del plugin terminado. El mismo debe permitir, para completar sus requerimientos, la posibilidad de exportar los resultados del ICA realizado

Para ello se trazaron las siguientes 3 tareas:

1. **Tarea 11:** Selección del formato a utilizar
2. **Tarea 12:** Exportar datos
3. **Tarea 13:** Guardar ICA en estudio eogs.

Tarea	
No.: 11	**Número de la HU:** 4
Nombre de la tarea: Selección del formato a utilizar	
Tipo de tarea: Diseño	**Estimación:** 2 días
Fecha de inicio: 5 de mayo del 2014	**Fecha de Fin:** 7 de mayo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Seleccionar el formato a utilizar para exportar los resultados del ICA el cuál podrá ser HTML o PDF. Esto se realizará usando un objeto QCombobox para que el especialista lo haga manualmente y este se colocará en la última pantalla del plugin.	

Tabla 21: Tarea 11: "Selección del formato a utilizar"

Tarea	
No.: 12	**Número de la HU:** 4
Nombre de la tarea: Exportar datos	
Tipo de tarea: Diseño	**Estimación:** 6 días
Fecha de inicio: 7 de mayo del 2014	**Fecha de Fin:** 14 de mayo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Usar las bibliotecas necesarias para exportar los resultados. Además, se debe permitir al usuario la posibilidad de escoger donde guardar dichos resultados.	

Tabla 22: Tarea 12: "Exportar datos"

Tarea	
No.: 13	**Número de la HU:** 4
Nombre de la tarea: Exportar datos	
Tipo de tarea: Diseño	**Estimación:** 2 días
Fecha de inicio: 14 de mayo del 2014	**Fecha de Fin:** 16 de mayo del 2014
Programador responsable: Abel Antonio Fernández Higuera	
Descripción: Guardar los datos dentro del estudio en el que se esté trabajando dentro de un fichero .eogs siguiendo las convenciones de la plataforma para este tipo de procesos	

Tabla 23: Tarea 13: "Exportar datos"

3.3. Pruebas

Las pruebas de aceptación también son conocidas como pruebas de caja negra, ya que es el propio cliente quien la realiza en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones realizadas en las historias de usuario.

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Adicionar restricciones para filtrar las sácadas	
Descripción: Prueba la funcionalidad de adicionar restricciones	
Condiciones de ejecución: Se debe contar con un estudio $*.eogs$ y los sácadas previamente detectadas y anotadas.	
Entrada/Pasos de ejecución: Al abrirse el Wizard del plugin para realizar ICA en su primera primera página aparece una restricción con un botón de + al final de la misma. Presionar el botón para adicionar una nueva restricción. Al hacerlo se adicionará una nueva restricción y el botón de la resultante cambiará el + por un - para si la misma luego se desea eliminar.	
Resultado esperado: Se adicionan las restricciones nuevas y se cambia la funcionalidad del botón para eliminarlas futuramente.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 24: Caso de prueba de aceptación HU2_P1

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Filtrar las sácadas según los criterios seleccionados	
Descripción: Prueba la funcionalidad de filtrar las sácadas	
Condiciones de ejecución: Se debe contar con un estudio $*.eogs$ y los sácadas previamente detectadas y anotadas. Se deben tener varias restricciones adicionadas en la primera página del plugin.	
Entrada/Pasos de ejecución: Adicionar varias restricciones, seleccionar su tipo e introducir los valores según corresponda. Luego, presionar el botón Filtrar Sácadas. Se mostrará en el gráfico de la izquierda las sácadas obtenidas después de filtrarlas según cada prueba.	
Resultado esperado: Se eliminan las sácadas que no cumplen con las restricciones y se muestran las resultantes en el componente visual de la izquierda de la pantalla.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 25: Caso de prueba de aceptación HU2_P2

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Seleccionar las sácadas visualmente	
Descripción: Prueba la funcionalidad de seleccionar visualmente las sácadas para realizarles ICA	
Condiciones de ejecución: Se debe contar con un estudio $*.eogs$ y los sácadas previamente detectadas y anotadas y filtradas según las restricciones adicionadas por el usuario.	
Entrada/Pasos de ejecución: Seleccionar la prueba a estudiar en el Combobox de la esquina superior derecha de la pantalla. En el árbol de eventos de la derecha del componente, deben mostrarse todas las sácadas previamente filtradas y al hacer clic en cada una de estas si se seleccionan se deben mostrar en rojo y si se desmarcan se deben quitar del componente visual a la izquierda de la pantalla.	
Resultado esperado: Se visualizan y seleccionan las sácadas e acuerdo a lo esperado.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 26: Caso de prueba de aceptación HU3_P1

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Aplicar ICA	
Descripción: Prueba la funcionalidad de aplicar ICA al ensamblaje de sácadas elaborado.	
Condiciones de ejecución: Se debe contar con un estudio $*.eogs$ y los sácadas previamente detectadas y anotadas y filtradas según las restricciones adicionadas por el usuario.	
Entrada/Pasos de ejecución: Seleccionar el algoritmo a utilizar e introducir los parámetros necesarios para el correcto funcionamiento del mismo. Luego, hacer clic en el botón Aplicar ICA y se debe mostrar en el componente visual de la izquierda de la pantalla los resultados del ICA realizado.	
Resultado esperado: Luego de aplicar el ICA, se muestran correctamente los resultados del ICA realizado.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 27: Caso de prueba de aceptación HU4_P1

3.4. Utilización de la aplicación

Para trabajar con la herramienta propuesta existe un flujo de trabajo que contine los siguientes pasos:

1. Ejecutar la plataforma de procesamientos NSEog lo cuál abrirá una ventana como la que se muestra en la Figura **??**.
2. Crear o abrir un estudio para trabajar con el mismo.
3. Detectar las sácadas de forma automática si no han sido marcadas aún para que la herramienta pueda funcionar usando el diálogo mostrado en la Figura 9
4. Abrir el plugin dentro del NSEog
5. Adicionar las restricciones y ejecutar el filtrado usando el diálogo que se observa en la Figura **??**.
6. Seleccionar visualmente las sácadas válidas para realizar el procesamiento usando el diálogo que se observa en la Figura **??**.

Figura 9: Diálogo de detección automática de sácadas

7. Seleccionar el ICA realizado y aplicarlo.

8. Exportar los datos del ICA realizado y guardarlo en el estudio abierto.

3.5. Valoración de la propuesta

Para la valoración de todo producto informático se tienen las siguientes dimensiones de sostenibilidad: administrativa, socio-humanista, ambiental y tecnológica. Su valoración en las cuatro dimensiones dan una idea de las ventajas que el mismo proporciona en cada uno de estos aspectos.

En la dimensión administrativa se puede afirmar que el producto es sostenible ya que su costo de elaboración y aplicación de la propuesta se reduce a un precio mínimo, debido a que se desarrolló por un estudiante de informática y está dirigido al equipamiento con que se cuenta en la actualidad en el CIRAH usando solo software libre o de código abierto. Lo cual es debido a que la alternativa era contratar a una empresa nacional o extranjera para que realizara un sistema a medida, lo que costaría miles de dólares a la clínica. Además, el procesamiento de estos registros se hará más ágil al usar un producto más eficiente.

La solución propuesta tiene una componente social de alto valor, esto esta dado a que las ataxias hereditarias son un problema de salud muy serio para nuestra comunidad con más de

800 enfermos y 8000 presintomáticos y las investigaciones llevadas a cabo en el CIRAH tienen un impacto directo en nuestra sociedad más cercana. Esta herramienta realizará su modesta contribución para encontrar terapias que ayuden a mitigar los efectos de esta enfermedad en nuestra población. Además contribuirá a que los especialistas del CIRAH ganen en comprensión de esta enfermedad al poder analizar por separado las dos componentes que intervienen en el modelo de generación de sácadas, por lo que se considera a la propuesta sostenible en la dimensión socio-humanista.

En cuanto a la dimensión ambiental, aun cuando este no tiene impacto directo sobre el medio ambiente, contribuye a su preservación, reduciendo el riesgo de estrés psicológico o de salud a los usuarios que lo utilicen. Lo planteado se logra debido a que las herramientas que este aporta, mejoran el proceso que se realizaba con el mismo fin de la propuesta. Esto ocurre gracias al uso eficiente de las interfaces de hardware (sean mouse o teclado) en cada uno de los escenarios que se brindan en la propuesta. Otro de los aspectos es que las interfaces gráficas son sencillas y agradables, lo que repercute de forma directa en la salud de los usuarios que lo utiliza. Todo esto indica que la solución propuesta es sostenible en esta dimensión.

La dimensión tecnológica de la propuesta, se considera sostenible debido a que los especialistas que utilizarán la misma en el CIRAH tienen la preparación necesaria para su explotación. Además de que en el diseño de esta se tuvo en cuenta el equipamiento de esta entidad y otras instituciones del sistema de salud cubano.Debido a que la propuesta responde a una necesidad directa del CIRAH a un mínimo costo, podemos concluir que la propuesta es sostenible en las cuatro dimensiones analizadas.

La valoración que da el cliente acerca del producto final también es de vital importancia. A este es al que se destina la aplicación y por lo tanto su opinión es definitiva respecto a la factibilidad de la solución. La metodología XP ofrece como ya se había visto anteriormente las Pruebas de Aceptación que diseña y ejecuta el cliente y son aceptadas por el equipo de trabajo como acuerdo de trabajo realizado. Lo que quiere decir que el trabajo se considera completado cuando todas las pruebas de aceptación son satisfactorias para el cliente. En el caso de la solución propuesta, todas las pruebas resultaron satisfactorias para el cliente.

3.6. Conclusiones del capítulo

En el presente capítulo se abordaron todos los temas referentes a el diseño, la implementación y la estrategia de pruebas de la solución. Se presentó el diseño de la estructura del plugin y las tareas que se llevaron a cabo para construirla.

La efectividad del desarrollo dirigido por pruebas y la aplicación de las pruebas de aceptación demuestran ser muy altas en el proceso de desarrollo de software. Ambas juegan un papel fundamental en el proceso de construcción de la herramienta con una metodología ágil, aún cuando el equipo de desarrollo es de un solo integrante.

Conclusiones generales

Como resultado de la investigación, se obtiene una herramienta que con las funcionalidades que brinda, permite extender las funcionalidades de la plataforma de procesamiento de electro-oculogramas NSEog, lo que contribuye al aumento de la calidad de los servicios investigativos y en consecuencia asistenciales que se ofrecen en el CIRAH.

La cuidadosa elección de las interfaces gráficas de usuario permitió obtener una herramienta acorde a los estándares visuales de las aplicaciones en este campo.

El uso de Matplotib dentro de PyQt5 permitió lograr gráficos de gran calidad dentro del producto informático propuesto, ganando además en productividad en el proceso de desarrollo del mismo.

El uso de PyQt5 permitió que la herramienta se adapte a la cultura de cada sistema operativo en el que se despliega, facilitando su uso por parte del especialista.

El uso de la metodología XP con un enfoque ágil, orientada a equipos de desarrollo pequeños, y apoyada por la interacción constante diseñador-programador-especialistas permitió la obtención de un producto en consonancia con los objetivos de este proyecto.

A partir de los aspectos que se han especificado en los párrafos precedentes, se considera que los objetivos planteados al acometer la elaboración de este producto fueron cumplidos.

La herramienta para aplicar análisis de componentes independientes a registros de movimientos oculares sacádicos, ha sido concebida y desarrollada teniendo en cuenta el estado actual de la neurofisiología, en relación con la caracterización y evaluación de los movimientos oculares sacádicos, adaptada a las características del equipamiento, los usuarios y la información a ser

procesada, y utilizando las tecnologías y herramientas de elaboración de aplicaciones informáticas de uso corriente internacionalmente, por lo que responde a las expectativas y necesidades de los especialistas del CIRAH.

Recomendaciones

Para garantizar la continuidad del trabajo y el enriquecimiento de la herramienta propuesta se realizan las siguientes recomendaciones:

1. Incorporar al producto informático propuesto otros algoritmos para realizar ICA con el fin de aumentar sus potencialidades y analizar los resultados de la aplicación de los mismos.
2. Realizar un estudio en profundidad del algoritmo FastICA en función de mejorar los resultados obtenidos pues los mismos se alejan bastante de los esperados

Bibliografía

[1] R. A. Becerra Garcia, Plataforma de Procesamiento de Electrooculogramas. Caso de Estudio Pacientes con SCA2., Ph.D. thesis, Universidad de Holguin (2013).

[2] V. R. Garcia Bermudez, Procesamiento de registros oculares sacadicos en pacientes de ataxia SCA2. Aplicacion del Analisis de Componentes Independientes., Ph.D. thesis, Universidad de Granada (2010).

[3] A. Cockburn, Agile Software Development, Cockburn, 2001.

[4] L. Velazquez, Neurophysiological clinical study of 70 patients with type 2 spinocerebellar ataxia, Revista de Neurologia 30 (2) (2000) 109–115.
URL http://www.scopus.com/scopus/inward/record.url?eid=2-s2.0-0034673006&partnerID=40&rel=R8.2.0http://www.scopus.com/scopus/record/display.url?eid=2-s2.0-0034673006&view=basic&origin=inward&txGid=_ZMsGYGYB1P2QEKbj7KBLSu%3a51

[5] Velázquez, Luis, Ataxia espino cerebelosa tipo 2. Principales aspectos neurofisiológicos en el diagnóstico, pronóstico y evaluación de la enfermedad., Ediciones Holguín., 2006.

[6] Y.-F. Chen, Y.-Y. Lee, M.-W. Huang, Y.-C. Du, T. Chen, Analysis of decomposed saccadic pulses and steps by independent component analysis for patients with brain stem lesions, 2005, pp. 136–140.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-60749102465&

partnerID=40&md5=280332fc43b9ad1a98fd8aee1ddb5383http://www.scopus.com/record/display.url?eid=2-s2.0-60749102465&origin=inward&txGid=v4Vv6LqCMUL0tGFDeljI5f9%3a295

[7] COA Centro de Optometría Avanzada.
URL http://coabilbao.com/sacadicos.html

[8] M.-U. Manto, The wide spectrum of spinocerebellar ataxias (SCAs), The Cerebellum 4 (1) (2005) 2–6. doi:10.1080/14734220510007914.
URL http://dx.doi.org/10.1080/14734220510007914http://www.springerlink.com/content/xj882j324p545310/

[9] A. Hyvärinen, E. Oja, Independent Component Analysis: Algorithms and Applications, Networks, Neural 1 (1) (2000) 411–430.

[10] D. Langlois, S. Chartier, D. Gosselin, An Introduction to Independent Component Analysis: InfoMax and FastICA algorithms, Tutorials in Quantitative Methods for Psychology 6 (1) (2010) 31–38.

[11] F. Rojas, R. V. Garcia, J. Gonzalez, L. Velazquez, R. Becerra, O. Valenzuela, B. San-Roman, Identification of Saccadic Components in Spinocerebellar Ataxia Applying an Independent Component Analysis Algorithm, Neurocomputing.
URL http://www.sciencedirect.com/science/article/B6V10-45M6G4R-5/2/be1c7b9ddf4b59d44f5f07e02f93e80b

[12] R. V. Garcia, F. Rojas, J. Gonzalez, I. Rojas, L. Velazquez, N. Canales, R. Becerra, ICA separation of saccadic pulse-step components in Cuban ataxia SCA2 patients. Changes in time to maximum values., Havana, Cuba, 2011.

[13] J. L. Semmlow, W. Yuan, Adaptive Modification of Disparity Vergence Components: An Independent Component Analysis Study, Invest. Ophthalmol. Vis. Sci. 43 (7) (2002) 2189–2195.
URL http://www.iovs.org/cgi/content/abstract/43/7/2189http://www.

iovs.org/cgi/content/full/43/7/2189http://www.iovs.org/cgi/reprint/43/7/2189.pdf

[14] M. Lutz, Learning Python 4th Edition, 4th Edition, O'Reilly, 2009.

[15] M. Summerfield, Rapid GUI Programming whit Python and Qt, Prentice Hall, 2008.

[16] I. Idris, NumPy 1.5 Beginner's Guide, PACKT, Birmingham, UK, 2011.
URL http://www.packtpub.com

[17] T. E. Oliphant, SciPy Tutorial, no. September, 2003.

[18] MDP-Developers, Modular toolkit for Data Processing Tutorial, 2011.

[19] R. G. Figueroa, C. J. Solis, A. A. Cabrera, Metodologias Tradicionales vs Metodologias Agiles.

[20] J. Canos, P. Letelier, M. Penades, Métodologías Ágiles en el Desarrollo de Software.

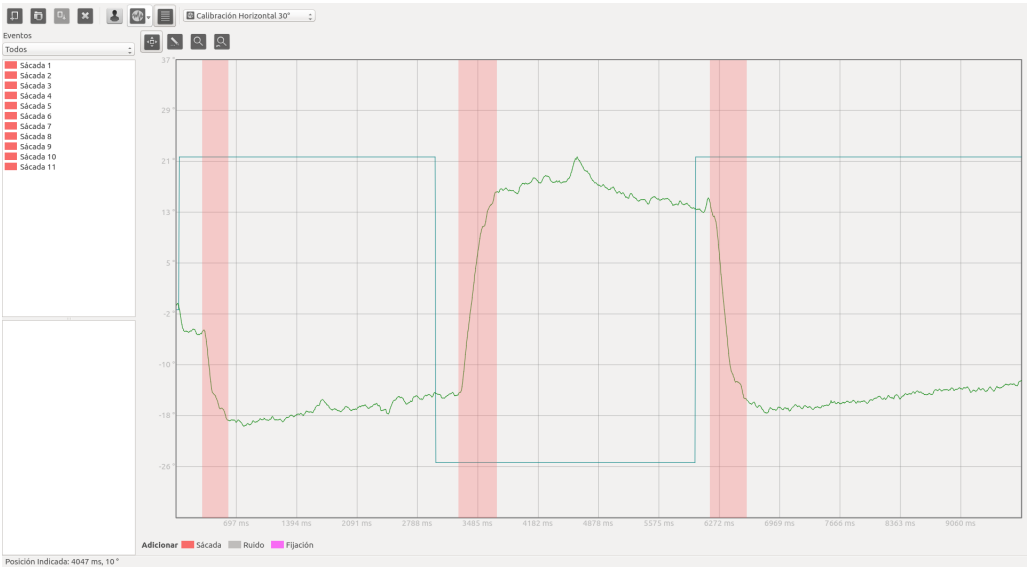
[21] M. C. Solis, Una explicación de la programación extrema (XP) (2003).
URL http://www.apolosoftware.com/

[22] A. Aguilar, Introduccion a la Programacion Extrema (2002).

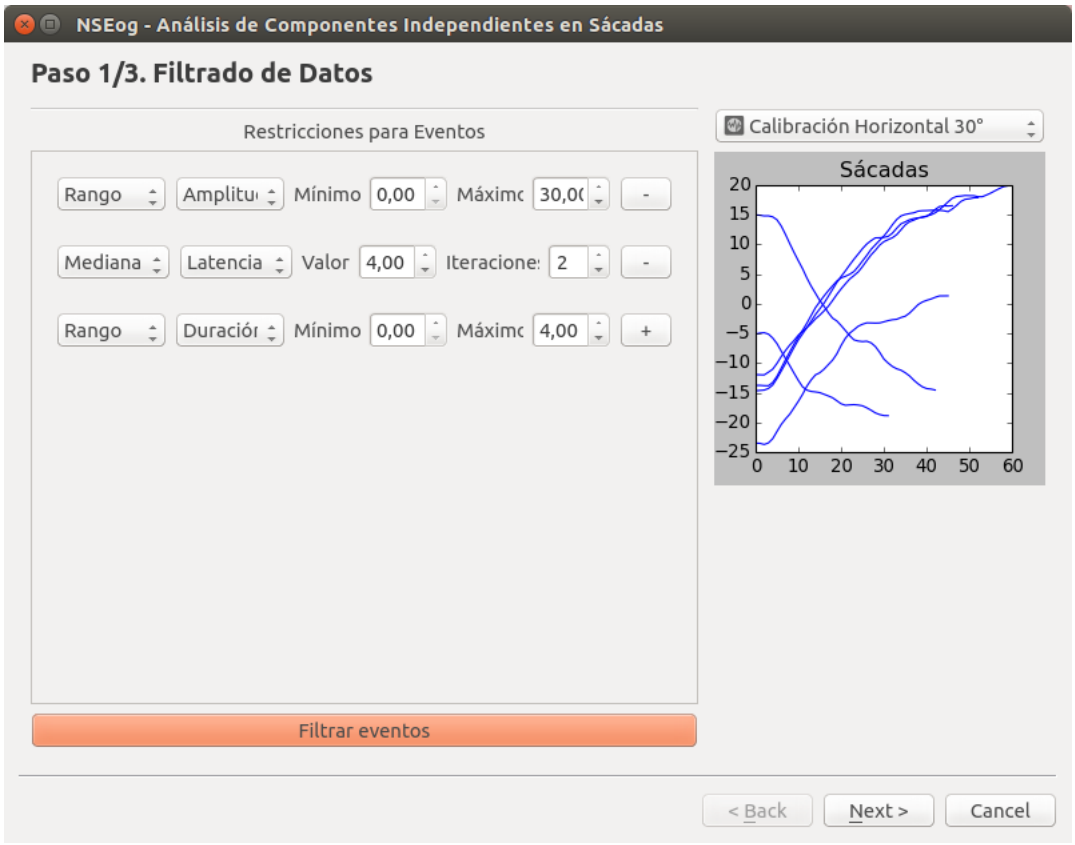
[23] K. Beck, M. Fowler, Planning Extreme Programming Kent Beck Martin Fowler, first edit Edition, Addison-Wesley Longman Publishing Co., Inc., 2000.

Anexos

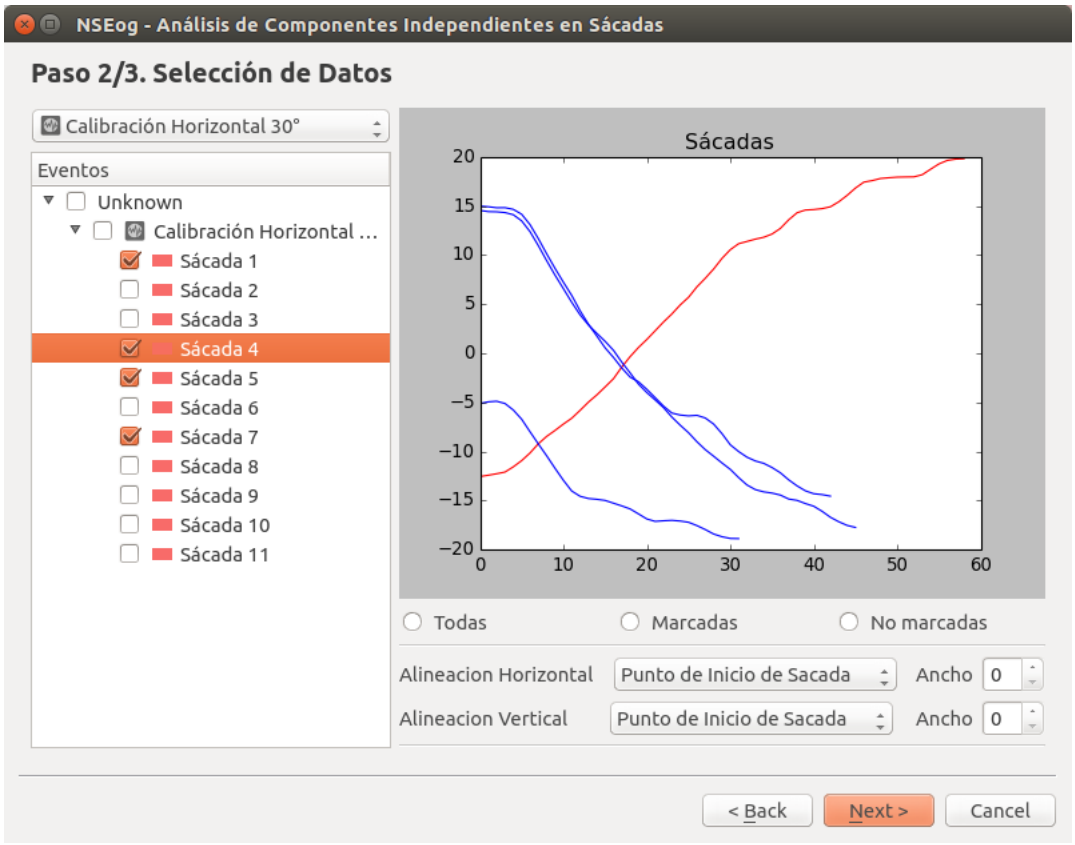
Anexo A: Interfaces Gráficas de Usuario



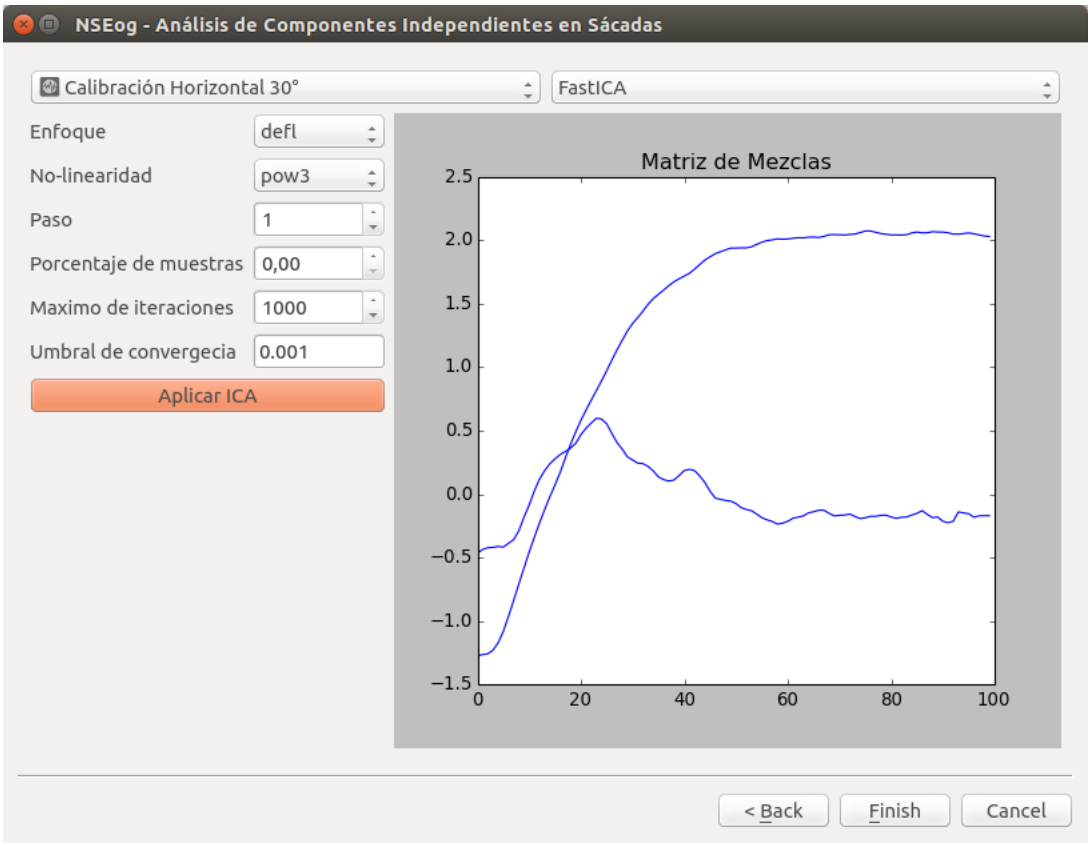
- Pantalla principal de la plataforma NSEog



- Diálogo de filtrado de sácadas



- Diálogo de selección y alineación de sácadas



- Pantalla de las Componentes Independientes